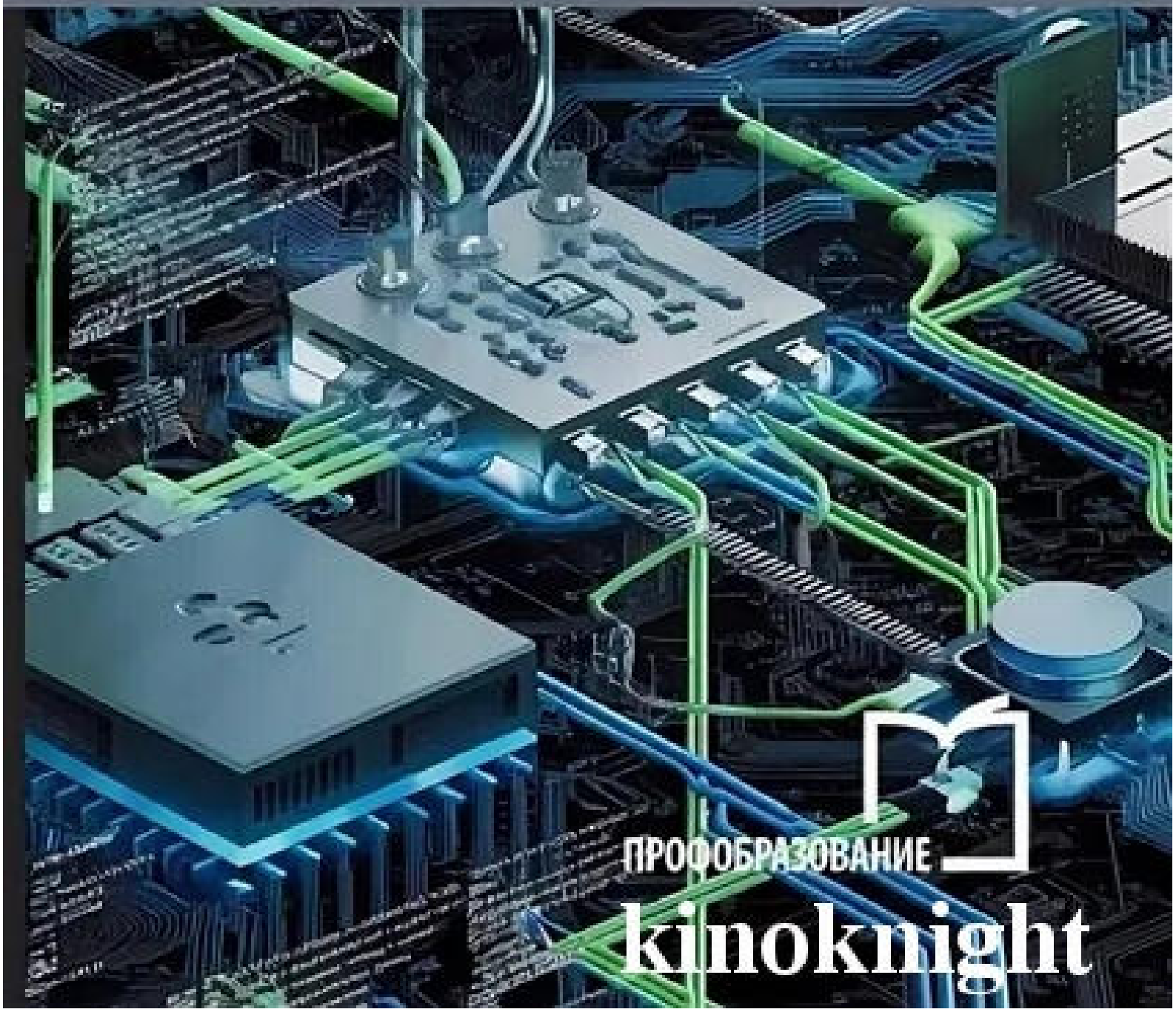


О.В. Болдырихин, В.А. Алексеев

Архитектура компьютерных систем

Учебное пособие для СПО

Рекомендовано УМО СПО



ПРОФОБРАЗОВАНИЕ

kinoknight

УДК 004.2
ББК 32.97
Б79

Авторы:

Болдырихин О.В. — ст. преподаватель кафедры автоматизированных систем управления Липецкого государственного технического университета;

Алексеев В.А. — канд. техн. наук, доц. кафедры автоматизированных систем управления Липецкого государственного технического университета

Рецензенты:

Громов Ю.Ю. — д-р техн. наук, проф., научный руководитель кафедры информационных систем и защиты информации, директор Института автоматики и информационных технологий Тамбовского государственного технического университета;

Негода В.Н. — д-р техн. наук, доц., проф. кафедры вычислительной техники Ульяновского государственного технического университета

Болдырихин, Олег Владимирович.

Б79 Архитектура компьютерных систем : учебное пособие для СПО / О.В. Болдырихин, В.А. Алексеев. — Москва : Ай Пи Ар Медиа ; Саратов : Профобразование, 2026. — 517 с. — (Среднее профессиональное образование). — Текст : электронный.

ISBN 978-5-4497-3619-2 (Ай Пи Ар Медиа)
ISBN 978-5-4488-2234-6 (Профобразование)

Учебное пособие формирует полное представление об основных деталях взаимодействия между аппаратным и программным обеспечением ЭВМ. Рассмотрены устройство и порядок функционирования ЭВМ, детали работы процессора и компьютера при выполнении программ.

Подготовлено в соответствии с Федеральным государственным образовательным стандартом среднего профессионального образования.

Учебное пособие предназначено для изучения дисциплин «Архитектура компьютерных систем», «Архитектура аппаратных средств» студентами укрупненной группы специальностей «Информатика и вычислительная техника».

Учебное электронное издание

ISBN 978-5-4497-3619-2 (Ай Пи Ар Медиа)
ISBN 978-5-4488-2234-6 (Профобразование)

© Болдырихин О.В., Алексеев В.А., 2026
© ООО Компания «Ай Пи Ар Медиа», 2026
© ООО «Профобразование», 2026

Учебное издание

Болдырихин Олег Владимирович
Алексеев Владимир Александрович

Редактор *М.В. Леман*
Технический редактор, компьютерная верстка *Н.Г. Шиндина*
Корректор *Т.П. Сударикова*
Обложка *Я.А. Кирсанов, С.С. Сизиумова*, фотобанк Freepik

Подписано к использованию 16.01.2026. Объем данных 14 Мб.

ООО «Профобразование»
8-800-511-14-70 (бесплатный звонок по России)
E-mail: office@profspo.ru, sale@profobr.pro

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 7

1. ОБЩИЕ СВЕДЕНИЯ ОБ ЭВМ 11
1.1. СУЩНОСТЬ И НАЗНАЧЕНИЕ ЭВМ 11
Информация и информационные процессы 11
Количество информации 21
ЭВМ как программно-аппаратная основа информационно-коммуникационных технологий 27
1.2. ОСНОВЫ И КРАТКАЯ ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ 32
Предпосылки создания электронных вычислительных устройств. Представление информации. Алгоритм 32
Двоичная система счисления 38
Математическая логика 46
Первые цифровые вычислительные машины 60
Поколения ЭВМ и появление персональных компьютеров 72
1.3. ОСНОВНЫЕ СТРУКТУРНЫЕ ПРИНЦИПЫ УСТРОЙСТВА ЭВМ 78
Архитектура фон Неймана 78
Процессор ЭВМ как универсальный алгоритм 83
Программирование ЭВМ 87
Язык ассемблера 91
1.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 1 102
Задание 1. Основные устройства ЭВМ 102
Задание 2. Математическая логика. Создание обрабатывающих и запоминающих устройств ЭВМ из логических элементов 107
Задание 3. Основные принципы устройства и работы ЭВМ 124

2. АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ. СТРУКТУРА ЭВМ 134
2.1. ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ МАШИН 134
Технико-эксплуатационные показатели ЭВМ 134
Классификационные признаки и основные типы вычислительных машин 143
Аналоговые, химические и нейронные компьютеры 154
Оптические и фотонные компьютеры 159
Молекулярный компьютер. Мемристоры 163
2.2. ОБЩАЯ СТРУКТУРА ЭВМ 168
Структуризация ЭВМ 168

Общая схема структуры ЭВМ 171
Порты ввода-вывода. Дешифраторы. Ресурсы 178
Внешние устройства ЭВМ 182
Запоминающие устройства ЭВМ 185
Интерфейс процессора с памятью и внешними устройствами 193
2.3. АРХИТЕКТУРА ПЭВМ 197
Набор микросхем (чипсет) ПЭВМ. Архитектура северного и южного моста 197
Развитие архитектуры ПЭВМ 206
Контроллер-концентратор платформы (PCH) 215
2.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 2 221
Задание 1. Конфигурация ПЭВМ 221
Задание 2. Оценка производительности ПЭВМ 229

3. МИКРОПРОЦЕССОР. БАЗОВАЯ СТРУКТУРА И СОВРЕМЕННЫЕ ТЕХНОЛОГИИ. 235
3.1. ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ ПРОЦЕССОРОВ 235
Определение и характеристики микропроцессоров 235
Классификация процессоров по архитектуре 244
Специализированные процессоры 251
3.2. ХАРАКТЕРИСТИКИ, НАЗНАЧЕНИЕ ВЫВОДОВ И СТРУКТУРА УНИВЕРСАЛЬНОГО МИКРОПРОЦЕССОРА 260
Классический микропроцессор i8080 260
Микропроцессор i8086. Семейство x86 278
Процессоры архитектуры IA-32 и Intel 64 296
3.3. СИСТЕМА КОМАНД ПРОЦЕССОРА 318
Программная модель процессора. Характеристики команды 318
Способы адресации 322
Система команд процессоров IA-32 и Intel 64 337
3.4. СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В МИКРОПРОЦЕССОРАХ 353
Конвейерная обработка 353
Технология Hyper-Threading 362
Технология виртуализации 367
3.5. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 3 377
Задание 1. Характеристики процессора. Оценка производительности процессора 377
Задание 2. Система команд процессора. Подпрограммы и стек. Способы адресации и флаги 386

4. ФУНКЦИОНИРОВАНИЕ И ОБМЕН ДАННЫМИ В ЭВМ 395
4.1. ФУНКЦИОНИРОВАНИЕ ЭВМ 395

Последовательность действий процессора при выполнении программы ... 395
Состояния и режимы работы процессора ... 399
Прерывания и исключения ... 405
4.2. ИНТЕРФЕЙСЫ ПАМЯТИ И ВВОДА-ВЫВОДА В ЭВМ ... 412
Классификация интерфейсов ЭВМ ... 412
Интерфейс PCIe ... 418
Интерфейс USB ... 425
Интерфейсы внешней памяти. SATA, SAS и NVMe ... 433
Сетевые интерфейсы. Ethernet ... 442
4.3. ОСНОВНЫЕ ПОДСИСТЕМЫ ВВОДА-ВЫВОДА ПЭВМ ... 446
Ввод с клавиатуры ... 446
Видеосистема ... 453
Аудиосистема ... 462
4.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 4 ... 471
Задание 1. Интерфейсы ПЭВМ ... 471
Задание 2. Ввод данных с клавиатуры ... 475
Задание 3. Вывод информации на дисплей ... 477

5. ПАРАЛЛЕЛИЗМ И МНОГОЗАДАЧНОСТЬ. КОМПЛЕКСНЫЕ ЭВМ ... 481
5.1. ТЕРМИНОЛОГИЯ И КЛАССИФИКАЦИЯ В ОБЛАСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ... 481
Виды и уровни параллелизма ... 481
Классификация ЭВМ по признаку параллелизма ... 484
5.2. ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ ... 492
Суперкомпьютеры рейтинга Top500 ... 492
Суперкомпьютер Frontier ... 496
Архитектура суперкомпьютера Sunway TaihuLight ... 499
5.3. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 5 ... 506
Задание 1. Динамика рейтинга Top500 по основным техническим характеристикам суперкомпьютерных систем ... 506
Задание 2. Получение списка систем с заданными характеристиками ... 509

ЛИТЕРАТУРА ... 511

ВВЕДЕНИЕ

Вне зависимости от конкретных задач, выполняемых по роду профессиональной деятельности, любой специалист в области информационных технологий должен знать и понимать устройство и порядок функционирования компьютерных систем (вычислительных систем, ЭВМ).

Учебное пособие предназначено для формирования соответствующих знаний и навыков у ИТ-специалистов на уровне, позволяющем использовать компьютеры и разрабатывать для них программы с пониманием особенностей их работы, т. е. работы процессора и компьютера в целом при выполнении программ.

ЭВМ — это сложная система. Существует большое число типов ЭВМ: суперкомпьютеры, настольные компьютеры, ноутбуки, мобильные устройства (смартфоны и планшеты), встроенные системы (специализированные компьютеры, использующиеся для управления оборудованием технологических процессов в промышленности, телекоммуникационых систем, узлами автомобилей и т.п.). Составляющие их элементы: процессоры, запоминающие устройства, устройства и интерфейсы ввода-вывода — также сложны и разнообразны. Но при этом, основные принципы устройства и работы всех вышеперечисленных типов компьютеров одни и те же. Рассмотрение в данном учебном пособии всех деталей и аспектов устройства и функционирования компьютерных систем направлено на изучение этих общих основных принципов. Основное внимание уделяется, конечно, наиболее распространенным в настоящее время типам ЭВМ, но дается также представление и о других интересных и перспективных способах вычислений.

Главным элементом ЭВМ является центральный процессор, поэтому особенности конкретной ЭВМ во многом обусловлены его характеристиками. Универсальные процессоры, использующиеся в качестве центральных, рассматриваются наиболее подробно. Приведена

также информация об основных типах, назначении и характеристиках специализированных процессоров.

Один из основных признаков классификации процессоров — сложность системы команд и самих команд. По этому признаку выделяется два класса — RISC- и CISC-процессоры.

RISC (Restricted (Reduced) Instruction Set Computer) — компьютер (процессор) с сокращенным набором команд [1]. У RISC-процессоров система команд меньше, команды более простые и выполняются быстрее, чаще всего за один такт. Соответственно, программа для RISC-процессоров длиннее.

CISC (Complex (Complete) Instruction Set Computer) — компьютер (процессор) со сложным (полным) набором команд. У CISC-процессоров система команд больше, команды более сложные и выполняются медленнее, но программа для них короче.

Второй важный признак классификации процессоров и компьютеров — организация запоминающего устройства (памяти). В памяти компьютера находится программа, выполняемая процессором. Это называется концепцией хранимой программы. Программа — это совокупность команд и данных. Первый вариант организации памяти: основное запоминающее устройство является конструктивно единым, в нем хранятся и команды программы, и данные. Такая архитектура называется принстонской. Второй вариант: запоминающее устройство является раздельным, одна часть хранит только команды, другая часть — только данные. Такая архитектура называется гарвардской. Чаще всего компьютеры на основе RISC-процессоров имеют гарвардскую архитектуру, а на основе CISC-процессоров — принстонскую.

Большинство мобильных устройств использует RISC-процессоры. Большинство настольных систем и суперкомпьютеров — CISC-процессоры. Поэтому в данном учебном пособии большее внимание будет уделяться рассмотрению принстонской CISC-архитектуры на примере процессоров семейства x86 производства Intel или AMD, которые являются наиболее распространенными.

Для закрепления теоретического материала в конце каждого раздела приводятся практические задания, которые в зависимости от учебного плана могут быть включены в лабораторные или практические работы. Они представляют собой небольшие экспериментальные исследования, выполняемые с помощью различных программных средств.

В результате изучения материалов учебного пособия студент приобретает следующие

1) ***знания:***

– назначение, принципы устройства и работы ЭВМ в целом и ее составных частей: процессора, запоминающих устройств, устройств ввода-вывода и их интерфейсов;

– принципы обработки информации с использованием ЭВМ, хранения информации в памяти ЭВМ, передачи информации между элементами ЭВМ;

– способы и инструментальные средства программирования ЭВМ, ассемблер, программирование ЭВМ на низком уровне;

– принципы формирования обрабатывающих и запоминающих устройств ЭВМ из базовых логических элементов: И, ИЛИ, НЕ;

– технико-эксплуатационные показатели, классификационные признаки и основные типы вычислительных машин;

– архитектура и характеристики основных элементов и интерфейсов ПЭВМ;

– характеристики и классификация процессоров;

– система команд процессора, способы адресации;

– особенности процессоров принстонской CISC-архитектуры на примере микропроцессоров семейства x86;

– современые технологии в микропроцессорах: конвейерная обработка, технология Hyper-Threading, технология виртуализации;

– порядок обмена данными и функционирования ЭВМ;

– характеристики и типы параллельных вычислительных систем;

2) ***умения:***

– оценивать технико-эксплуатационные показатели ЭВМ в целом, показатели и характеристики их составных частей: процессоров, чипсетов, запоминающих устройств, устройств и интерфейсов ввода-вывода;

– создавать обрабатывающие и запоминающие устройства ЭВМ в виде схем из базовых логических элементов с использованием программных средств схемотехнического моделирования;

– программировать на языке ассемблера;

– использовать инструментальные средства программирования на языке ассемблера: компилятор, компоновщик, отладчик;

– исследовать ход программы с помощью отладчика, отслеживать состояние элементов ЭВМ в ходе выполнения программы;

– отслеживать изменение стека при выполнении команд, влияющих на него: вызова и возврата из подпрограмм, помещения в стек, извлечения из стека и др. — а также при возникновении аппаратных прерываний и возврата из их обработчиков;

– определять способы адресации, используемые в командах процессора;

– определять влияние команд на флаги;

3) ***навыки:***

– моделирования обрабатывающих и запоминающих устройств ЭВМ;

– анализа характеристик элементов и интерфейсов ЭВМ;

– расчета пиковой производительности процессора, пропускной способности интерфейсов и других показателей ЭВМ;

– использования инструментальных средств программирования ЭВМ на языке ассемблера;

– исследования состояния элементов ЭВМ в ходе выполнения программы.

1. ОБЩИЕ СВЕДЕНИЯ ОБ ЭВМ

1.1. СУЩНОСТЬ И НАЗНАЧЕНИЕ ЭВМ

Информация и информационные процессы

Для человека одним из важнейших понятий является информация. Самого человека можно назвать информационной системой или информационной машиной. В процессе жизнедеятельности человеку необходимо принимать и обрабатывать информацию, а затем на основе результатов обработки принимать и реализовывать решение. В самых простых индивидуальных видах деятельности человек воспринимает информацию об окружающей среде, существенную для выполнения стоящей задачи, например, передвижения, и соответствующим образом выстраивает последовательность действий. В более сложных, коллективных видах деятельности возникают дополнительные информационные процессы — коммуникационные. Человек — существо социальное, люди должны согласованно действовать для достижения определенных целей, для этого существуют язык, письменность и информационно-коммуникационные технологии.

Информация — это одно из наиболее общих понятий, которыми оперирует наука, наряду с материей и энергией.

Выделяется два вида материи — *вещество* и *поле*. Точнее сказать, две стороны или два ее проявления, поскольку все материальные объекты обладают свойствами и вещества, и поля. Это называется корпускулярно-волновым дуализмом. С веществом связано понятие частицы или, другими словами, корпускулы, а волна — это изменяющееся во времени и распространяющееся в пространстве поле.

Корпускулярно-волновой дуализм материальных объектов проявляется, например, в явлениях фотоэффекта и дифракции электронов на кристаллической решетке. Фотоэффект — это испускание электронов веществом под воздействием света. В этом явлении проявля-

ются корпускулярные свойства электромагнитного поля или волны, которые выражаются в существовании фотонов как квантов (частиц) электромагнитного поля. Дифракция — это огибание волной препятствия. При дифракции электронов на кристаллической решетке проявляются волновые свойства вещества.

Энергия — второе важнейшее понятие науки. Энергия — это способность совершать работу. Все материальные объекты обладают энергией. Энергия связана с еще одним важным понятием — массой. Эту связь устанавливает знаменитая формула Эйнштейна:

$$E = mc^2,$$

где c — скорость света в вакууме, одна из фундаментальных физических постоянных.

И, наконец, третье важнейшее понятие науки — информация. Существует множество его определений. Самым распространенным является следующее. Информация — сведения о чем-либо: явлениях, процессах, объектах и т. п. Однако информация и сведения — это синонимы. Таким образом, это определение ничего не дает для понимания информации. Просто излагает его другими словами.

Норберт Винер, создатель кибернетики, науки об управлении, дал такое определение: «Информация есть информация, не материя и не энергия» (Information is information, not matter or energy) [2, 3]. В этом определении отражен тот факт, что информация является наиболее общим понятием. Его нельзя подвести под другое более общее понятие так же, как и материю. Обычная форма определения, когда для определяемого понятия указывается более общее понятие, а затем указываются видовые отличия определяемого понятия, для информации не подходит. Поэтому Винер просто противопоставляет эти три понятия.

Можно привести еще одно определение: «Информацией является любая связь между системами, в результате которой увеличивается

негэнтропия хотя бы одной из этих систем» [4]. **Негэнтропия** — это мера порядка в системе. Поэтому получение информации в норме должно приводить к увеличению порядка. Для понимания этого и полезно данное определение. В нем в качестве более общего понятия по отношению к информации приводится связь (взаимодействие). Однако в измеряемых процессах связь или взаимодействие материальны, таким образом, все это понятия одного самого высокого уровня общности.

Наконец, есть еще одно очень важное понятие науки — **время**. Человеку понятие времени необходимо для описания изменения так же, как понятие пространства необходимо для описания движения. Понятия изменения и движения связаны между собой, поэтому было введено понятие четырехмерного пространства-времени. Время можно трактовать как результат превращения энергии в информацию [5, 6]. Утверждается, что время появляется там, где идут необратимые процессы, где изменяется энтропия, что, в свою очередь, связано с выбором из множества вариантов и вероятностью [7]. Примерами этого могут служить образование Вселенной или игра в бильярд. Но надо заметить, что в таких процессах существует именно «стрела времени», «незамкнутое» время. В процессах строго периодических время, можно сказать, замыкается, «идет по кругу». Любой момент времени в таких процессах можно заменить его суммой с величиной, кратной периоду:

$$y = A\sin(\omega t) = A\sin(\omega(t + kT)) = A\sin(\omega t + \omega kT) =$$
$$= A\sin(\omega t + \frac{2\pi}{T}kT) = A\sin(\omega t + 2\pi k) =$$
$$= A\sin(\omega t).$$

Именно такие периодические изменения берутся в качестве эталона времени. Год — это время обращения Земли вокруг Солнца. Сутки — это время обращения Земли вокруг своей оси. Секунда долгое время определялась как определенная часть года. Сейчас секунда определяется как время, за которое совершается 9 192 631 770 перио-

дов колебаний в излучении, соответствующем переходу между сверхтонкими уровнями основного состояния атома цезия-133.

Период колебания — величина, обратная частоте. А частота электромагнитной волны излучения связана с энергией соотношением:

$$E = h\nu, \tag{1.1}$$

где E — энергия кванта;

h — постоянная Планка;

ν — частота излучения.

В системах с периодическими процессами тоже происходит превращение энергии. В консервативных системах полная энергия остается постоянной, но она периодически переходит из одной формы в другую, например, в механическом маятнике без трения потенциальная энергия груза переходит в кинетическую и обратно, в идеальном электронном колебательном контуре энергия электрического поля конденсатора — в энергию магнитного поля катушки индуктивности и обратно. Примерно то же самое происходит и в электромагнитной волне: вихревое электрическое поле порождает вихревое магнитное, оно в свою очередь — электрическое и т. д., что и приводит к распространению волны в пространстве. В диссипативных периодических системах (автоколебательных системах), каковыми являются различные генераторы колебаний, энергия также переходит из одной формы в другую, но при этом она расходуется, рассеивается, а этот расход компенсируется энергией источника питания. Причем энергия источника должна поступать в колебательный контур не постоянно, а периодически, синхронно с колебаниями в нем, что обеспечивается положительной обратной связью.

Такие генераторы имеются в любых радиопередающих устройствах и, конечно, в компьютерах. Там он называется генератором тактовых импульсов. Формируемые им тактовые импульсы представляют собой периодический прямоугольный сигнал, который поступает на соответствующий вход процессора и задает дискретное время, в

соответствии с которым в компьютере происходят все процессы. **Такт** — это один период тактового сигнала. Тактовая частота является важнейшей характеристикой процессора. Чем больше тактовая частота, тем больше производительность процессора и компьютера в целом. Теоретическая или пиковая производительность процессора определяется формулой, похожей на формулу (1.1) энергии кванта:

$$E = I / T = (I / Clock) \cdot (Clock / T) = IPC \cdot F, \qquad (1.2)$$

где I — количество выполненных команд;

T — время;

$Clock$ — количество тактов;

F — тактовая частота;

IPC — Instructions Per Clock, количество команд, выполняемых процессором за такт.

Вместо постоянной Планка здесь стоит величина, характеризующая степень внутреннего параллелизма процессора. При отсутствии параллелизма она меньше единицы, при наличии — больше или, по крайней мере, равна единице.

Также в любом электронном устройстве, в том числе ЭВМ, присутствует источник питания. И на примере любого электронного устройства, особенно ЭВМ, очень хорошо видно, как энергия источника питания превращается в информацию. Благодаря источнику питания в устройстве идут процессы, которые приводят к появлению какой-либо информации: картинке на экране компьютера, звуку в динамиках и т. п.

Итак, все три понятия: материя, энергия и информация, — связаны между собой. Материя обладает энергией. Информация не материальна, но является продуктом превращения энергии. В процессе превращения энергии в информацию возникает время. Иногда говорят, что информация требует материального носителя. На самом деле, она просто становится видна, воспринимаема, когда имеет матери-

альный носитель. Материальным носителем информации может быть и вещество, и поле (волна).

«Вещественным» носителем информации может быть все, что угодно: папирус, камень, береста, дерево, кожа, веревка (завязывание узелков) и т. д. В настоящее время используются такие ***носители информации***, как:

– бумага;
– пленка;
– магнитная лента или магнитный диск;
– лазерный диск;
– кристалл полупроводника.

Многие из перечисленных носителей выходят из широкого употребления, но продолжают использоваться в определенных областях. По-прежнему везде используется бумага. Расширяется применение микросхем, где носителем информации является кристалл кремния. В большинстве современных персональных компьютеров уже нет магнитных носителей и приводов лазерных компакт-дисков, а в качестве внешней памяти используются накопители SSD (Solid State Drive, твердотельный накопитель), основу которых составляют микросхемы постоянной памяти. В качестве носителей кратковременной (динамической) информации в компьютерах используются микросхемы оперативной памяти, в которых информация быстро изменяется при функционировании компьютера и теряется при выключении его питания.

Пользователю в процессе его работы за компьютером информация большей частью выводится так же динамически на экране монитора. Гораздо реже происходит вывод информации на бумагу с помощью принтера. Внутри человека существует ***два основных вида информации***:

– изначальная, например, содержащаяся в ДНК;
– приобретенная или полученная, находящаяся в памяти (мозге).

Человек воспринимает информацию с помощью органов чувств. Как известно, у человека ***пять внешних чувств***:

– зрение;
– слух;
– осязание;
– обоняние;
– вкус.

В работе за компьютером осязание задействовано при использовании средств ввода данных: клавиатуры, мыши и т. п. В основном воспринимается информация с помощью зрительного аппарата, для чего используется ее визуальное представление с помощью какого-либо дисплея, но также применяется и ее аудиопредставление с использованием звуковой карты (платы) и динамиков, и в определенной степени задействуется и слуховой аппарат человека.

В качестве носителей для визуального представления информации пользователю используются светодиоды, жидкие кристаллы, люминофор ЭЛТ, газоразрядные приборы и т. п. Перечисленные устройства и вещества применяются для создания соответствующих дисплеев. В некоторые моменты, когда определенный этап работы закончен, документ подготовлен, он распечатывается на бумаге.

«Полевым» носителем информации в принципе также может быть любое поле. В настоящее время с практической целью используется только электромагнитное поле в виде волн оптического, инфракрасного диапазонов и радиодиапазона.

Информацию можно классифицировать по разным признакам: содержанию, форме представления, способу кодирования и т. д. В контексте нашей дисциплины наиболее важные деления таковы:

– по множеству принимаемых значений:
  – непрерывная (аналоговая);
  – дискретная (цифровая);
– с точки зрения структуры компьютерной программы:
  – команды (информация о действии);
  – данные (входные операнды или результаты действия);

– сигналы на шинах (линиях связи) ЭВМ:
  – адрес;
  – данные;
  – управление.

Определение информации через негэнтропию, приведенное выше, приводит к принципу относительности или полезности информации: ценность (потребительское качество или количество) информации определяется возможностью ее принять, понять и использовать.

Процесс, в котором осуществляются какие-либо операции над информацией, называется информационным.

Операции, выполняемые с информацией:

– генерация (создание);
– извлечение (data mining);
– отбор;
– получение (прием);
– восприятие;
– передача;
– представление;
– преобразование;
– обработка;
– хранение;
– использование (применение).

Представление — это вид преобразования информации, при котором происходит перевод ее в форму, удобную для восприятия человеком.

Информационные процессы происходят повсеместно в природе. Примером информационного процесса в живой природе на уровне популяции может служить самоорганизация коллективных амеб.

Информационный процесс на уровне отдельных органов — формирование изображения на сетчатке глаза.

Информационный процесс в технике — формирование информации о температуре охлаждающей жидкости двигателя автомобиля с помощью датчика, ее передача и представление водителю при помощи индицирующего прибора.

Информационный процесс в обществе — накопление и структуризация знаний в форме печатных (библиографических) изданий в библиотеках.

Примеры процессов передачи информации в прошлом — костры на башнях, флажный семафор. В современности — радио, телевидение, Интернет.

Как уже было отмечено, информация важна не сама по себе, а как одно из средств, необходимых для решения какой-либо задачи, повышения негэнтропии, порядка в каком-либо смысле: улучшения качества жизни, повышения эффективности технологического процесса и т. п. Во всех случаях это связано с понятием управления. **Управление** — это процесс выработки и реализации решения на основе полученной информации. Совокупность средств, осуществляющих этот процесс, называется управляющей системой (УС). Выработка и реализация решения всегда подразумевает воздействие на некоторый объект или процесс для достижения заданной цели. Такое воздействие называется управляющим, а объект или процесс, на который оно направлено — объектом управления (ОУ). Получение управляющей системой информации об объекте управления и реализация управляющего воздействия на него дает кольцо, цикл или замкнутый контур управления (рис. 1.1).

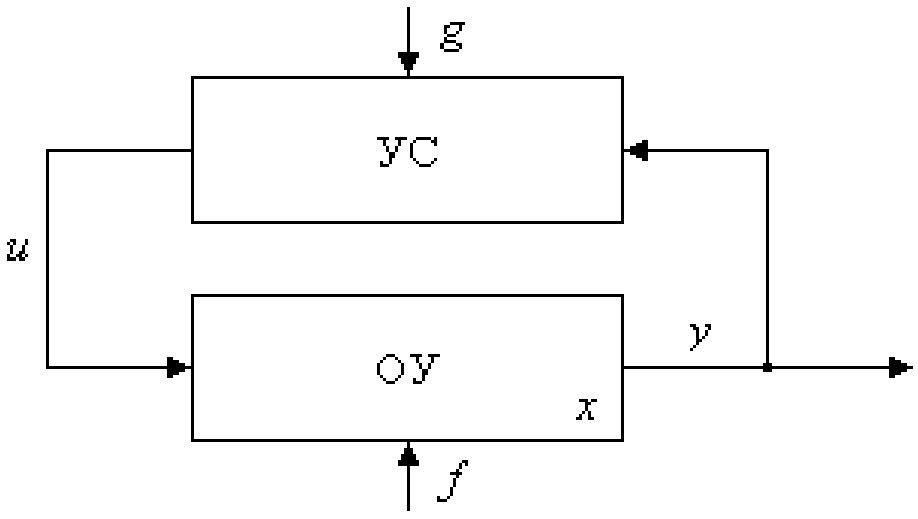


Рис. 1.1. Замкнутый контур управления

Информационные связи в системе управления:

g — задание или цель;

u — управляющее воздействие;

x — внутренняя информация о состоянии объекта управления;

y — извлеченная информация о состоянии объекта управления (выход);

f — возмущающие воздействия.

В замкнутом контуре управления реализуется последовательность информационных процессов: получение, передача, обработка, представление, применение информации. В общем случае в сложных системах контуров управления может быть множество. Управляющая система в настоящее время практически в любом процессе управления строится на основе ЭВМ. ЭВМ может быть одна, или их может быть много, разного типа, с разным набором аппаратных и программных средств, но если даже нет специальных аппаратных средств ввода-вывода для получения информации о состоянии объекта управления и реализации управляющего воздействия, таких как датчики и исполнительные механизмы, эти контуры, так или иначе, присутствуют. Например, студент изучает дисциплину «Архитектура компьютерных систем» на дистанционном курсе. Объект управления — процесс изучения дисциплины. Студент получает теоретическую информацию и задания с использованием своего компьютера, на котором имеется какой-либо браузер (веб-клиент), и удаленного компьютера с веб-сервером и системой управления обучением (Learning Management System, LMS), который и выдает всю эту информацию клиенту. Это управляющая информация. Студент выполняет задания, проходит тесты и т. п. — это информация о состоянии объекта управления, которую получает и оценивает управляющая система. Это первый, основной контур управления. В процессе обучения студент выполняет практические задания на своем компьютере с использованием необходимых программных средств. Например, он создает программу и запускает ее в отладчике для исследования ра-

боты компьютера по ее выполнению. Здесь управляющая система — студент и соответствующие программные и аппаратные средства. Объект управления — процесс выполнения программы. Студент управляет этим процессом, инициируя очередные шаги, задавая контрольные точки и т. п. Информация о состоянии объекта управления — значения регистров процессора, портов ввода-вывода и переменных в памяти. Это другой контур управления.

Главной частью самого компьютера является устройство управления процессора, которое тоже реализует цикл управления — чтение команд и данных программы из памяти, дешифрацию команд и их выполнение. Для обеспечения согласованной работы всех частей компьютера в этом процессе устройство управления формирует управляющие сигналы, которые выдаются через соответствующие выводы процессора и в совокупности составляют шину управления.

Количество информации

Существует несколько подходов к определению количества информации. Одним из них является вероятностный подход.

Вероятность какого-либо события будем оценивать через относительную частоту его появления в полном множестве возможных событий. Например, при подбрасывании монеты возможны два события (исхода) — выпадение орла и выпадение решки. Они составляют полное множество событий и равноправны, т. е. нет оснований считать, что более предпочтительным является осуществление какого-то одного из них. Сумма вероятностей всех событий полного множества равна единице. Поэтому вероятность выпадения и орла, и решки — одна вторая.

Для опеределения количества информации с использованием вероятностного подхода используется три постулата:

1. Информации в событии тем больше, чем меньше его вероятность.

2. Количество информации в нескольких независимых событиях равняется сумме количеств информации в каждом из этих событий.

3. В детерминированном событии, т. е. таком событии, которое произойдет с вероятностью, равной единице, количество информации равняется нулю.

Они приводят к следующей формуле определения количества информации:

$$I_a = \log \frac{1}{P_a} = -\log P_a, \qquad (1.3)$$

где a — событие;

I_a — количество информации в событии a;

P_a — вероятность события a.

То есть количество информации в событии — это логарифм величины, обратно пропорциональной вероятности события, или логарифм вероятности события со знаком минус.

В этом случае выполняются все три постулата. Например, количество информации в двух независимых событиях:

$$I_{12} = \log \frac{1}{P_{12}} = \log \frac{1}{P_1 \cdot P_2} = \log \left(\frac{1}{P_1} \cdot \frac{1}{P_2} \right) = \log \frac{1}{P_1} + \log \frac{1}{P_2} = I_1 + I_2.$$

Основание логарифма в принципе может быть любым. Удобно выбирать в качестве основания логарифма количество возможных исходов. Например, при бросании монеты возможны два исхода. Соответственно, основание логарифма в этом случае можно взять равным двум.

Поскольку широко распространен натуральный логарифм, то он и используется чаще всего вне зависимости от количества исходов.

Подсчитаем количество информации в событии «выпадение орла» при бросании монеты:

$$I_a = \log_2 \frac{1}{P_a} = \log_2 2 = 1.$$

Оно равно единице. Единица информации, определяемая таким образом, называется «бит» (англ. *bit — binary digit*, двоичная цифра). **Бит** — это количество информации, содержащееся в событии, представляющем собой выбор одного из двух равновероятных вариантов: орел или решка, да или нет, истина или ложь, ноль или единица, — две цифры двоичной системы счисления, которая и используется в ЭВМ на физическом аппаратном уровне.

Понятно, что в событии «выпадение решки» также содержится один бит информации.

Конечно, не всегда события полного множества (возможные исходы опыта) равновероятны. Например, известно, что бутерброд из-за смещения центра масс чаще падает маслом вниз. Вероятность этого исхода больше, а количество информации в нем, соответственно, меньше. Если бутерброд упадет маслом вверх, это будет удивительно, информационная насыщенность такого исхода будет высока.

В таких случаях желательно бывает знать среднее количество информации в возможных исходах опыта. Пусть, например, вероятность того, что бутерброд упадет маслом вниз, равна 0,8. Тогда вероятность противоположного исхода — 0,2. Среднее количество информации в опыте «падение бутерброда» будет определяться следующим образом:

$$\bar{I} = 0{,}8 \log_2 \frac{1}{0{,}8} + 0{,}2 \log_2 \frac{1}{0{,}2} = -0{,}8 \log_2 0\,,8 - 0{,}2 \log_2 0\,,2 \approx 0{,}722.$$

Здесь мы не просто сложили количество информации в первом исходе с количеством информации во втором исходе и поделили на два, а добавили в качестве множителей к ним вероятности исходов. Ведь они различны. Поэтому среднее должно быть не простым, а «взвешенным». Весами и являются как раз вероятности исходов.

В общем случае справедливо соотношение:

$$\bar{I} = \sum_{i=1}^{n} P_i I_i = \sum_{i=1}^{n} P_i \log \frac{1}{P_i} = -\sum_{i=1}^{n} P_i \log P_i .$$

Это формула математического ожидания дискретной случайной величины. **Случайная величина** — это какая-либо мера случайного события. Например, в теории надежности систем основное случайное событие — отказ (утрата системой работоспособности). Ему соответствуют непрерывная случайная величина — наработка на отказ (время от начала работы системы до отказа) — и дискретная случайная величина — число отказов на заданном интервале времени. В нашем случае случайной величиной является количество информации, соответствующее событию (исходу опыта).

В случае равновероятных исходов среднее количество информации равно единице (если брать в качестве основания логарифма число возможных исходов). Например, для того же опыта бросания монеты с двумя равновероятными исходами:

$$\bar{I} = 0{,}5 \log_2 \frac{1}{0{,}5} + 0{,}5 \log_2 \frac{1}{0{,}5} = 0{,}5 \log_2 2 + 0{,}5 \log_2 2 = 1 .$$

Это максимальное значение. Неравновероятные исходы всегда дают значение, меньшее единицы.

В информационно-коммуникационных технологиях событием, несущим информацию, является прием сообщения. В цифровой технике сообщения являются дискретными. Дискретное сообщение состоит из множества символов. Пусть число символов алфавита, который используется для формирования сообщений, равно *m*. Пусть длина сообщения фиксирована, что характерно для интерфейсов (каналов связи) компьютерных систем, и составляет *n* символов. Тогда

среднее количество информации, приходящееся на один символ алфавита, равно:

$$\bar{I} = \sum_{i=1}^{m} P_i \log \frac{1}{P_i} = -\sum_{i=1}^{m} P_i \log P_i,$$

где p_i — вероятность i-го символа.

Тогда среднее количество информации в сообщении:

$$\bar{I} = -n \sum_{i=1}^{m} P_i \log P_i.$$

Если символы алфавита имеют одинаковую вероятность, то:

$$\bar{I} = -n \sum_{i=1}^{m} \frac{1}{m} \log \frac{1}{m} = n \sum_{i=1}^{m} \frac{1}{m} \log m = n \log m.$$

Если в качестве основания логарифма взять m, то:

$$\bar{I} = n.$$

Таким образом, в случае равной вероятности появления символов в сообщении вероятностный подход к определению количества информации приводит к обычному «объемному» или «емкостному» подходу, который применим не только к приему информации, но и к передаче и хранению. На нижнем физическом уровне в компьютерных системах и сетях используется двоичная система счисления и, соответственно, только два символа — 0 и 1. Они имеют одинаковую вероятность, и, как уже было отмечено, один двоичный символ несет один бит информации. Таким образом, количество информации, хранимой, передаваемой или принимаемой в компьютерных си-

стемах, измеряется в количестве двоичных символов — нулей или единичек в информационных блоках, т. е. в битах, байтах и производных (кратных) единицах:

1 байт = 2^3 бит = 8 бит;
1 Кбайт (килобайт) = 2^{10} байт = 1 024 байта;
1 Мбайт (мегабайт) = 2^{20} байт = 1 024 Кбайта;
1 Гбайт (гигабайт) = 2^{30} байт = 1 024 Мбайта;
1 Тбайт (терабайт) = 2^{40} байт = 1 024 Гбайта;
1 Пбайт (петабайт) = 2^{50} байт = 1 024 Тбайта;
1 Эбайт (эксабайт) = 2^{60} байт = 1 024 Пбайта.

Однако надо заметить, что в кратных единицах скорости передачи или пропускной способности приставки соответствуют степеням не двойки, а десяти, как и в физике:

1 Кбит/с = 1 000 бит/с;
1 Мбит/с = 1 000 Кбит/с = 1 000 000 бит/с;
1 Гбит/с = 1 000 Мбит/с = 1 000 000 Кбит/с = 1 000 000 000 бит/с.

Обозначение производных единиц байта можно сократить:
1 Кбайт = 1 Кб.

Обозначение производных единиц бита обычно не сокращают, чтобы не было путаницы. Но иногда в литературе, документации и в различных приложениях для байтов используется сокращение с прописной буквой «Б»:

1 Кбайт = 1 КБ (KB),

а для бита — со строчной:

1 Кбит = 1 Кб (Kb).

ЭВМ как программно-аппаратная основа информационно-коммуникационных технологий

Электронная вычислительная машина (ЭВМ) или **компьютер** — универсальное устройство для автоматической цифровой обработки информации, осуществляемой под управлением программы, выполненное на электронных приборах.

Таким образом, компьютерная система имеет две неразрывные составляющие — аппаратную и программную, аппаратное обеспечение (hardware) и программное обеспечение (software). Универсальность этой системы определяется тем, что она может выполнять различные программы, состоящие из различных команд и использующие различные данные. Конкретная задача по обработке информации с помощью компьютера определяется используемой программой, находящейся в его памяти. Если нужно написать статью, то мы используем текстовый редактор. Если нужно прослушать музыку или посмотреть фильм — соответствующий проигрыватель. Есть программы для написания других программ — инструментальные системы программирования. Также есть программы для запуска и обеспечения работы других программ — операционные системы.

Конечно, универсальность имеет определенную степень. Есть компьютеры более универсальные и менее универсальные, специализированные. **Специализированные** — это, например, компьютеры для управления оборудованием определенного технологического процесса на каком-либо производстве, где в цикле управления выполняется одна и та же последовательность информационных процессов. В таких компьютерах управляющая программа является единственной, менять ее требуется нечасто или вовсе не требуется. Такого рода компьютеры имеются в настоящее время практически во всех типах электронных устройств, используемых в различных областях: в домашнем хозяйстве и быту (мультиварки, кофемашины, газовые котлы и т. п.), в промышленности, сельском хозяйстве, транспорте, конечно, в военном деле и т. д. Такие компьютеры называются

встраиваемыми (embedded system), потому что они в виде отдельной платы входят в состав системы, которой управляют. Как правило, основой встраиваемых систем являются микроконтроллеры — более полные в структурно-функциональном плане по сравнению с микропроцессорами интегральные схемы. Они помимо собственно процессора содержат в своем составе запоминающие устройства, параллельные и последовательные порты ввода-вывода, таймеры для задания и определения временных параметров, а также то или иное множество дополнительных устройств.

В настоящее время все больше встраиваемых систем имеют интерфейсы для подключения к сети Интернет. Они могут взаимодействовать между собой с помощью стандартных средств этой сети, аппаратных, программных и информационных, обмениваться информацией в автоматическом режиме, без участия человека. Это называется интернетом вещей (IoT, Internet of Things). Использование таких устройств в промышленности называется промышленным интернетом вещей. Интернет вещей интенсивно развивается во всех областях деятельности человека. ***Сферы использования интернета вещей:***

1. *Производство.* На предприятиях используется множество IoT-датчиков, с помощью которых собирается оперативная информация о ходе технологических процессов на различных этапах производства, благодаря чему можно повысить производительность и улучшить качество процесса, сократить издержки, снизить процент брака, обеспечить более безопасные условия труда, запланировать техническое обслуживание и т. п.

2. *Энергетика.* С помощью датчиков собираются данные о состоянии оборудования, параметрах сети и потреблении энергии, что позволяет автоматически перераспределять нагрузку, уведомлять оператора о необходимости проведения диагностики и ремонта и т. п.

3. *Сельское хозяйство.* В растениеводстве с помощью датчиков может контролироваться уровень влажности почвы и если он упадет ниже заданного значения, автоматически включится полив. В живот-

новодстве с помощью датчиков могут собираться показатели состояния животных и отправляться на обработку в облачный сервис.

4. *Строительство.* С помощью датчиков может собираться информация с различных объектов на строительной площадке: техники, материалов, зданий и даже персонала. Эти данные затем могут поступать в централизованную систему для обработки, анализа и принятия решений.

5. *Транспорт и логистика.* С помощью установленных камер и датчиков можно проводить оценку габаритов продукции и на основе полученной информации осуществлять оптимальную загрузку транспортных средств, повышая эффективность их использования, отслеживать перемещение товаров и транспортных средств.

6. *Медицина.* IoT-технологии могут использоваться для ухода за пациентами, сбора анализов, отслеживания состояния и перемещения пациентов.

7. *Розничная торговля.* Bluetooth-маяки позволяют отслеживать перемещение покупателей по торговым залам, что можно использовать для оптимизации размещения товаров и т. п.

8. *Общественная безопасность.* Интеллектуальные камеры с технологией распознавания лиц могут идентифицировать и отслеживать людей в многолюдных местах, предупреждая соответствующие службы о потенциальных угрозах.

9. *Умный город.* Здесь с помощью устройств и технологий IoT может решаться целый комплекс задач: управление ресурсами, оптимизация транспортных потоков с помощью гибкого управления светофорами, мониторинг качества воздуха и т. п.

Универсальные вычислительные системы — это, например, персональные компьютеры. Их можно использовать для решения самых разнообразных задач, для чего требуется запускать различные программы. Для запуска различных программ на исполнение и при необходимости для управления их работой или обеспечения их работы предназначена специальная программа, точнее, комплекс программ, — операционная система (ОС — OS, Operating System). Есть

еще одна важная совокупность программ — базовая система ввода-вывода (Basic Input-Output System, BIOS) или, в современной терминологии, встроенное программное обеспечение (firmware), которая тоже предназначена для обеспечения работы других программ, в том числе и операционной системы. BIOS находится в постоянной памяти (ROM BIOS) компьютера и начинает работать сразу после включения питания. Она тестирует оборудование, инициализирует необходимые параметры, некоторые из которых можно поменять с помощью программы Setup BIOS, например, дату и время, настройки оборудования, а потом загружает операционную систему с диска в оперативную память компьютера.

ЭВМ в настоящее время редко используются по отдельности. Домашние компьютеры с помощью маршрутизаторов подключены к сети Интернет через какого-либо поставщика услуг (провайдера). Рабочие компьютеры на предприятиях и в организациях соединены в локальную сеть также с выходом в Интернет через прокси-сервер. В компьютерных сетях выделяются такие виды ЭВМ, как рабочие станции и серверы. Рабочие станции используются сотрудниками так же, как и персональные компьютеры — для выполнения своих профессиональных обязанностей. **Серверы** — это более производительные компьютеры по сравнению с рабочими станциями. Серверы используются совместно многими пользователями для решения общих задач, например, для хранения общих файлов, размещения баз данных или веб-приложений и т. п. Не очень мощные серверы выглядят как рабочие станции, только имеют более производительный процессор или несколько процессоров. Мощные серверы представляют собой шкафы (стойки) с соответствующим оборудованием. Большой универсальный высокопроизводительный отказоустойчивый сервер со значительными ресурсами ввода-вывода, большим объемом оперативной и внешней памяти называется мейнфреймом (mainframe). Такого типа серверы впервые были разработаны компанией IBM в рамках знаменитой серии System/360, и термин «мейнфрейм» происхо-

дит от названия типовых процессорных стоек этой системы. Вообще, сервер («обслуживатель») — это устройство, которое предоставляет определенную услугу, в нашем контексте информационную, другому компьютеру (клиенту) и в конечном итоге его пользователю. Поскольку ЭВМ — это программно-аппаратное устройство, то термин сервер используется в двух смыслах — аппаратном и программном. Сервер — это и компьютер, и программы, установленные на нем, которые предоставляют определенную информацию. Таких программ-серверов множество — веб-сервер (HTTP-сервер, HyperText Transfer Protocol), принимающий запросы от веб-браузеров и выдающий им веб-страницы или данные для их формирования, файловый сервер (FTP-сервер, File Transfer Protocol), выдающий соответствующим клиентам файлы, DHCP-сервер (Dynamic Host Configuration Protocol), автоматически назначающий IP-адреса компьютерам сети, и т. д. Понятие клиента также понимается и в аппаратном, и в программном смысле, и как компьютер, и как определенная программа. Например, браузер (обозреватель) Интернета — это веб-клиент, который отображает веб-страницы, которые ему предоставляет веб-сервер. Каждого типа таких программ, и серверов, и клиентов, существует множество. Самым популярным веб-сервером является Apache. Наиболее распространенные веб-клиенты (браузеры) Yandex, Firefox и многие другие.

Ну и конечно, следует здесь также упомянуть о наиболее распространенном в настоящее время типе вычислительных систем — смартфоне. Это тоже персональное устройство, но более специализированное, предназначенное для коммуникации в самом широком смысле. Смартфоны также подключены к сети Интернет и по производительности не уступают персональным компьютерам. Они так же, как и другие компьютеры, работают под управлением операционной системы, поскольку на них требуется запускать множество различных программ (приложений). Их основой является такой тип микросхем, как система на кристалле (SoC, System on Chip). Система на кристал-

ле — это высокопроизводительный микроконтроллер. Она так же, как и микроконтроллер, имеет в своем составе все необходимые элементы компьютера — процессор, память, интерфейсы ввода-вывода. Но процессор SoC имеет высокую производительность, сравнимую с производительностью процессоров для персональных компьютеров и серверов.

1.2. ОСНОВЫ И КРАТКАЯ ИСТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Предпосылки создания электронных вычислительных устройств. Представление информации. Алгоритм

К созданию вычислительных устройств, в том числе электронных, привело множество идей и достижений в разных областях науки и техники. Можно выделить философские, лингвистические, математические, физические и технические предпосылки создания ЭВМ, которые, конечно, тесно связаны между собой самой концепцией алгоритмических вычислений, представления и преобразования информации в соответствии с алгоритмом.

Важнейшим для человека способом представления и передачи информации является язык. Естественный язык выражает информацию об окружающем мире. Окружающий мир (объективную реальность) можно представить в виде совокупности объектов, их свойств и действий, а также в виде отношений между объектами. В человеческом восприятии им соответствуют понятия, которые обозначаются словами и их сочетаниями. **Естественный язык** — это способ кодирования понятий для их представления и передачи. Для каждого типа элементов окружающего мира в языке имеются соответствующие части речи — существительные, прилагательные, глаголы и т. д. Выше было сказано о корпускулярно-волновой природе всех материальных

объектов, т. е. в природе существует как дискретное, так и непрерывное. Но язык, и устный, и письменный, дискретен. В устной речи слова — последовательности фонем, в письменной — последовательности букв — знаков алфавита. Язык, дискретный способ кодирования информации, выражает как дискретное, так и непрерывное. В мире существует огромное количество языков. В каждом из них свой алфавит, который содержит то или иное количество букв. Отдельная тема — выражение количественных свойств и закономерностей окружающего мира. Это уже не столько тема языка, сколько отдельной науки — математики. И в ней важнейшее место занимает вопрос представления чисел. Числа, как и слова, записываются в виде последовательности символов — цифр. В качестве цифр могут выступать как буквы, так и отдельные специальные символы, например, арабские цифры. Число различных цифр, используемых для записи чисел, как и число букв алфавита, может варьироваться. Самый распространенный вариант — десять цифр от нуля до девяти в десятичной системе счисления. Но есть и другие варианты. Непраздный вопрос — каков наилучший вариант, наилучший алфавит? В общем виде на этот вопрос, конечно, ответить нельзя. Поэтому и существуют различные языки и различные системы записи чисел (системы счисления). Но в определенном смысле наилучшим, оптимальным является алфавит, состоящий из двух символов. Это не минимальный вариант. Минимальным является алфавит из одного символа, но он не оптимальный, его можно использовать только в очень простых случаях, поскольку многие числа и слова из одного символа были бы очень длинными:

$1_{10} = 1_1$,
$2_{10} = 11_1$,
$3_{10} = 111_1$,
и т. д.

Это единичная (унарная) система счисления. Для такой системы вероятность появления символа в слове равна единице, поскольку

символ только один, а количество информации, соответственно, равно нулю. Информация появляется только в словах и сообщениях, поскольку нет различных слов, состоящих из одинакового количества символов. Кроме того, с использованием такого алфавита нельзя записать одно из важнейших чисел — ноль. Вообще-то, можно, но с использованием дополнительных «костылей», например:

$0_{10} = 1_1$,

$1_{10} = 11_1$,

$2_{10} = 111_1$,

и т. д.

Но самое плохое, что такая система счисления является непозиционной, что существенно затрудняет ее использование в технических устройствах.

Поэтому оптимальным вариантом является минимальное количество *различных* символов. И это количество — два. Это минимальное количество символов для записи различных слов одинаковой длины, что дает возможность построения уже позиционной системы счисления. Это и есть идея двоичной формы представления информации. Если говорить именно о представлении чисел, то это двоичная система счисления. Однако ее можно использовать для представления любой информации, не только числовой, но и текстовой, и графической, и звуковой и т. д. Именно так она и используется в ЭВМ, как это мы и увидим в дальнейшем.

Здесь, в контексте рассуждений о двоичном алфавите необходимо также упомянуть о двоичной логике Аристотеля, важнейшим положением которой является закон исключенного третьего: А или не-А истинно, третьего не дано. Здесь «А» — некоторое высказывание (утверждение). Другими словами, в двоичной логике Аристотеля любое высказывание имеет только два значения — истина или ложь. Здесь двоичность не в формулировке высказываний, а в их возможных значениях. Чтобы было понятно, что это тоже не единственный

вариант, приведем критику логики Аристотеля Гегелем: «Дух является зеленым» или «Дух не является зеленым». Какое из этих двух утверждений истинное? В более серьезной и основательной форме закон исключенного третьего критиковали более серьезные ученые, например, голландский философ и математик Брауэр, работы которого привели к созданию интуиционистской логики. Так что в общем плане критика справедлива, но нужно понимать, что любая теория, любой формализм имеет свои ограничения применимости.

Итак, идея двоичности — глубокая и в то же время простая философская идея: все гениальное просто. (Но не все простое — гениально.) Она существует в тех или иных формах в различных религиях и философиях: добро и зло, верх и низ (небо и земля), мужское и женское, и т. п. В логике — истина и ложь.

Наиболее широко и формально эта идея проявилась в древней китайской философии. Поэтому можно считать, что двоичная форма представления информации, двоичная система счисления возникла в Древнем Китае. Наиболее ранним источником, в котором появляется двоичная система, является «И цзин» (Книга Перемен) или «Чжоу И», около 700 г. до н. э. В Книге Перемен содержится 64 понятия, выраженных в форме гексаграмм — слов из шести элементов, символов. Каждая гексаграмма состоит из двух триграмм — слов из трех символов. Символ представляет собой черточку. Черточки используются двух видов — сплошные и прерывистые (пунктирные). Сплошные черточки символизируют активное начало: свет, твердость и т. п. (ян), а прерывистые — пассивное начало: тьму, мягкость и т. п. (инь). Две крайние триграммы: три прерывистые черточки — земля (кунь), три сплошные черточки — небо (цянь). Хотя изначально Книга Перемен предназначалась для гадания, она оказала огромное влияние на последующее развитие китайской философии и культуры в целом и далее через идею двоичной системы обозначений — на мировую науку, в частности на математическую логику, информатику и вычислительную технику.

Первым человеком в Европе, понявшим огромное значение двоичной системы, был Готфрид Вильгельм Лейбниц. В основном он известен как создатель дифференциального и интегрального исчисления, но помимо этого Лейбниц внес большой вклад и в другие разделы математики, а также философию, физику и вычислительную технику. В 1673 г. Лейбниц создал счетную машину, которая могла выполнять четыре арифметические операции над десятичными числами. Предыдущая счетная машина, разработанная Паскалем, выполняла только сложение и вычитание. Однако главная заслуга Лейбница в области компьютерной науки состоит в другом. Основной идеей его творчества была идея универсальной знаковой системы («универсальной характеристики»), которая позволяла бы представлять любые существующие понятия (любую информацию) и выводить новые. Лейбниц ценил двоичную логику Аристотеля и понимал важность удобных обозначений в математике и вообще в языке. Двоичная система наилучшим образом подходила для представления и логических значений, и чисел, и вообще любых понятий. Лейбниц узнал о ней от одного из странствующих монахов-миссионеров, сразу оценил по достоинству, начал развивать и пропагандировать. По его предложению в одном из немецких княжеств была выбита медаль в честь диадической системы, как она в то время называлась (рис. 1.2). На медали была изображена табличка, состоящая из двух столбцов: в правом стояли числа от 0 до 17 в десятичной системе, в левом — те же числа в двоичной системе. Вверху была надпись на латинском языке «2, 3, 4, 5
и т. д. Для получения их всех из нуля достаточно единицы». Надпись внизу гласила: «Картина создания. Изобрел Г(отфрид) Г(ильом) Л(ейбниц). Год MDCXCVII». (MDCXCVII —1697.)



Рис. 1.2. Медаль в честь двоичной системы счисления

Слово «изобрел» является здесь, конечно, преувеличением, но вклад Лейбница в развитие идеи двоичной символики действительно высок. Лейбниц использовал в качестве двоичных символов цифры 0 и 1, привел соответствие между десятичными числами и двоичными, примеры сложения, вычитания, умножения и деления двоичных чисел.

Лейбниц не смог воплотить свою идею универсальной знаковой системы. Однако более чем через сто лет после его смерти она частично воплотилась в виде математической логики Джорджа Буля (булевой алгебры или алгебры логики). А когда были изобретены и разработаны физические способы представления двоичных значений и созданы соответствующие технические устройства: ламповые, а затем и полупроводниковые диоды, триоды (транзисторы) и триггеры, — эта идея воплотилась в ЭВМ.

Винер писал: «Если бы мне пришлось выбирать в анналах истории наук святого — покровителя кибернетики, то я выбрал бы Лейбница. Философия Лейбница концентрируется вокруг двух основных идей, тесно связанных между собой: идеи универсальной символики и идеи логического исчисления» [2].

«...Machina rationatrix {думающая машина, рассуждающая машина} есть не что иное, как calculus ratiocinator {исчисление умозаключений} Лейбница, снабженное двигателем» [2]. Генератор такто-

вых импульсов — это и есть двигатель или «моторчик» компьютера. В каждом такте процессор выполняет определенные действия, которые задают все остальные действия компьютера. Выполнение той или иной задачи по обработке информации в ЭВМ представляет определенную последовательность таких действий. И здесь мы подходим к еще одному важному разделу математики — теории алгоритмов, которая наряду с двоичной системой и двоичной логикой лежит в основе концепции современных ЭВМ. **Алгоритм** (алгорифм) — это точное предписание, которое задает последовательность действий (в применении к ЭВМ — вычислительный или алгоритмический процесс) по обработке информации, начиная с определенных исходных данных, направленную на получение полностью определяемого этими исходными данными результата. Термин «алгоритм» происходит от латинского написания имени персидского математика IX века Аль-Хорезми. Алгоритмами являются, например, правила сложения, вычитания, умножения и деления столбиком. Именно в таком узком смысле, как правила счета в десятичной системе счисления, и понимался алгоритм в средневековой Европе, поскольку о принципах позиционной системы счисления Европа узнала только в XII в. благодаря латинскому переводу книги Аль-Хорезми. **Программа для ЭВМ** — это алгоритм, изначально записанный на каком-либо языке программирования, а затем переведенный на язык процессора другой программой, компилятором, также по определенному алгоритму. Сам процессор при выполнении программы работает по алгоритму, заложенному в него при разработке и изготовлении.

Двоичная система счисления

Рассмотрим основы двоичной системы счисления, которые нам потребуются в дальнейшем: представление целых и дробных чисел, перевод чисел из одной системы счисления в другую.

Итак, в двоичной системе существует только два знака для представления информации. По понятным причинам наиболее удобными являются цифры 0 и 1. Значениями логических переменных и логических функций в алгебре логики также могут быть только 0 или 1 (ложь и истина). Физически в цифровых электронных устройствах двоичным знакам соответствуют определенные уровни, точнее диапазоны напряжения, например, +5 вольт и 0 вольт (плюс-минус некоторое допустимое отклонение). В принципе, это могут быть любые уровни и диапазоны, главное, чтобы их можно было легко различить, т. е. с вероятностью, близкой к единице, определить, когда значение равно нулю, а когда — единице.

Для оперирования двоичными числами полезно знать степени двойки, записанные десятичными числами (табл. 1.1).

Таблица 1.1

Положительные и отрицательные степени двойки

n	**2^n**	**–n**	**$2^{(-n)}$**
0	1	0	1,00000000000000000000000000
1	2	−1	0,50000000000000000000000000
2	4	−2	0,25000000000000000000000000
3	8	−3	0,12500000000000000000000000
4	16	−4	0,06250000000000000000000000
5	32	−5	0,03125000000000000000000000
6	64	−6	0,01562500000000000000000000
7	128	−7	0,00781250000000000000000000
8	256	−8	0,00390625000000000000000000
9	512	−9	0,00195312500000000000000000
10	1024	−10	0,00097656250000000000000000
11	2048	−11	0,00048828125000000000000000
12	4096	−12	0,00024414062500000000000000
13	8192	−13	0,00012207031250000000000000
14	16384	−14	0,00006103515625000000000000
15	32768	−15	0,00003051757812500000000000
16	65536	−16	0,00001525878906250000000000

17	131072	−17	0,00000762939453125000000
18	262144	−18	0,00000381469726562500000
19	524288	−19	0,00000190734863281250000
20	1048576	−20	0,00000095367431640625000

Интересно отметить, что существуют колебания значений разрядов дробных чисел — отрицательных степеней двойки. Они прослеживаются по диагоналям разрядов в последнем столбце. В первой (значение 5) и второй (значение 2) диагонали колебаний нет. Далее чередуются значения 1 и 6 (период колебания можно трактовать как равный единице). В следующей диагонали чередуются 0, 3, 5, 8. Период равен трем. Далее идут более сложные колебания.

В двоичной записи положительным степеням соответствуют единицы с соответствующим числом нулей:

$2^0 = 1$,
$2^1 = 10$,
$2^2 = 100$,
и т. д.

Отрицательным — единица в соответствующем разряде после запятой:

$2^{-1} = 0{,}1$,
$2^{-2} = 0{,}01$,
$2^{-3} = 0{,}001$,
и т. д.

Это справедливо для любой позиционной системы счисления.

В компьютерной науке используются в основном три системы счисления — двоичная, десятичная и шестнадцатеричная. Десятичная система является наиболее привычной для человека, а двоичная —

наиболее подходящей для создания вычислительных устройств, как было обосновано выше. Однако двоичные числа слишком длинные, поэтому для их более короткого представления используется шестнадцатеричная система. Сразу отметим, что именно для представления. «Внутри» компьютера существуют только двоичные числа. В шестнадцатеричном виде, для более короткой записи они представляются в отладчиках, шестнадцатеричных редакторах, листингах программ. Шестнадцатеричная система выбрана, потому что $16 = 2^4$. Поэтому каждой шестнадцатеричной цифре соответствует тетрада двоичных цифр, и перевод из двоичной системы в шестнадцатеричную и обратно выполняется очень просто, простой заменой знаков. Надо запомнить следующую таблицу (табл. 1.2).

Таблица 1.2

Соответствие между цифрами и числами шестнадцатеричной, десятичной и двоичной систем счисления

Десятичная	Двоичная	Шестнадцатеричная
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E

15	1111	F
16	10000	10

Для перевода чисел из одной системы счисления в другую можно использовать формулу, определяющую структуру числа в позиционной системе счисления:

$$a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-m} = \sum_{i=-m}^{n} a_i \cdot b^i =$$

$$= a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \dots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot$$
$$\cdot b^{-2} + \dots + a_{-m} \cdot b^{-m},$$

где a — цифра системы счисления;

b — степень системы счисления;

i — номер разряда.

Пример:

$$10011_2 = 1 \cdot 10^{100} + 0 \cdot 10^{11} + 0 \cdot 10^{10} + 1 \cdot 10^1 + 1 \cdot 10^0 \xrightarrow{2 \to 10} 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{10}$$

Нижний индекс в исходном числе и результате обозначает систему счисления, а стрелочка в этом выражении означает момент перехода к обозначениям той системы счисления, в которую осуществляется перевод.

$$19_{10} = 1 \cdot 10^1 + 9 \cdot 10^0 \xrightarrow{10 \to 2} 1 \cdot 1010^1 + 1001 \cdot 1010^0 = 1010 + 1001 = 10011_2;$$

$$687_{10} = 6 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0 \xrightarrow{10 \to 16} 6 \cdot A^2 + 8 \cdot A^1 + 7 \cdot A^0 = 6 \cdot 64 + 8 \cdot A + 7 = 258 + 50 + 7 = 2AF_{16}$$

$$2AF_{16} = 2 \cdot 10^2 + A \cdot 10^1 + F \cdot 10^0 \xrightarrow{16 \to 10} = 2 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 512 + 160 + 15 = 687_{10}.$$

Перевод из двоичной системы в шестнадцатеричную и обратно, как уже было сказано, осуществляется прямой заменой знаков без осуществления вычислений.

Нужно обратить внимание, что запись числа, равного основанию системы счисления, во всех системах одинакова — 10.

Видно, что при переводе этим способом необходимо оперировать правилами выполнения арифметических действий той системы счисления, в которую осуществляется перевод. Эти правила привычны только для десятичной системы. Таким образом, этот способ удобен при переводе чисел в десятичную систему счисления.

При переводе из десятичной системы более удобен другой способ — деление на основание системы, в которую осуществляется переход. При этом способе арифметические действия осуществляются в той системе, из которой осуществляется переход, а замена знаков производится на последнем этапе.

Пример:

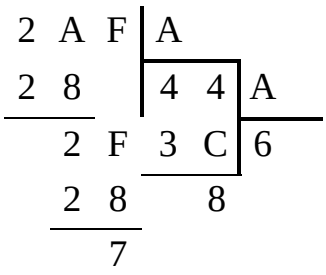
$$687_{10} = 2AF_{16}:$$

$$\begin{array}{r|l} 6\ 8\ 7 & 1\ 6 \\ \cline{2-2} 6\ 4 & 4\ 2 \\ \cline{1-1} 4\ 7 & 3\ 2 \\ \cline{1-1} 3\ 2 & 1\ 0 \to \mathrm{A} \\ \cline{1-1} 1\ 5 \to \mathrm{F} & \end{array} \qquad \begin{array}{|l} \hline 1\ 6 \\ \hline 2 \to 2 \\ \\ \\ \end{array}$$

Это становится более понятным, если воспользоваться следующим представлением, соответствующим такого рода разложению:

$$(2 \cdot 16 + 1\ 0) \cdot 16 + 15 = 2 \cdot 16^2 + 10 \cdot 16 + 15 \to 2 \cdot 10^2 + \mathrm{A} \cdot 10^1 + \mathrm{F} \cdot 10^0 = 2AF_{16}.$$

И в другую сторону можно сделать так же:



Но, как было отмечено, здесь нужно руководствоваться арифметическими правилами системы, из которой осуществляется перевод.

Для дробных чисел справедливо все сказанное. Нужно лишь учесть, что степени основания в дробной части являются отрицательными числами.

Перевод двоичной дроби в десятичную:

$$0{,}1101_2 \xrightarrow{2\to10} 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} = 0{,}5 + 0{,}25 + 0{,}0625 = 0{,}8125_{10}.$$

Перевод десятичной дроби в двоичную осуществляется последовательным умножением на два исходной дроби и дробных частей получающихся чисел:

$$\begin{aligned} 0,8125 \cdot 2 &= 1,625 \\ 0,625 \cdot 2 &= 1,25 \\ 0,25 \cdot 2 &= 0,5 \\ 0,5 \cdot 2 &= 1,0 \end{aligned}$$

Процесс заканчивается, когда дробная часть становится равной нулю. Целые части получившихся чисел будут стоять в соответствующих разрядах двоичной дроби. Оформить это удобнее в следующем виде:

0	8125
1	625
1	25
0	5
1	0

Первый столбец такой таблички и представляет собой результирующую двоичную дробь. Конечная десятичная дробь даст конечную двоичную дробь только в том случае, если она представляет собой конечную сумму отрицательных степеней двойки (табл. 1.1). Таким образом, пятерка в младшем разряде десятичной дроби является необходимым, но не достаточным условием того, что ей соответствует конечная двоичная дробь.

Таким же образом осуществляется и перевод десятичной дроби в шестнадцатеричную, только умножать нужно на 16:

$$0{,}8125_{10} = 0{,}D_{16}.$$

0	8125
13	0

Видно, что перевод между шестнадцатеричными и двоичными дробями, как и между целыми числами, осуществляется простой заменой шестнадцатеричных цифр соответствующими двоичными числами или обратно:

$$0{,}D_{16} = 0{,}1101_2.$$

Арифметические операции в различных системах счисления выполняются одинаково. Необходимо только выучить таблицы сложения и умножения для каждой системы счисления.

Здесь мы рассмотрели основные математические аспекты, связанные с двоичной системой. Существует также множество технических деталей и вариантов ее использования для представления информации в ЭВМ. Это представление дробных двоичных чисел в двух формах: с фиксированной и плавающей точкой (запятой), — а также различные двоичные коды: дополнительный код для представления отрицательных чисел; коды для представления символов, например, ASCII (American Standard Code for Information Interchange) и UTF-8 (Unicode Transformation Format); коды для повышения надежности передачи и обработки информации с контролем по четности, нечетности и др.

Математическая логика

Существует три основные логические операции: две бинарные — И (конъюнкция) и ИЛИ (дизъюнкция), — и одна унарная — НЕ (отрицание или инверсия). Их смысл тот же, что и в соответствующих союзах и частице естественного языка. Обозначать их можно по-разному. Можно использовать систему обозначений, принятую в арифметике — операцию И обозначать знаком умножения, ИЛИ — сложения. Операция НЕ часто обозначается чертой сверху, но в текстовых редакторах ее рисовать неудобно, поэтому ее мы будем использовать на рисунках, а в тексте — подчеркивание или знак решетки, #. Логические выражения (функции) составляются из логических операндов и логических операций. Для двух операндов A и B возможны 16 (2^4) различных логических функций, каждая из которых имеет определенный смысл и соответствующее название (табл. 1.3).

Таблица 1.3

Логические функции двух логических переменных

AB / F	00	01	10	11	Формула	Название
F_0	0	0	0	0	$F_0=0$	Константа ноль
F_1	0	0	0	1	$F_1=A\cdot B$	Конъюнкция (И)
F_2	0	0	1	0	$F_2=\underline{A\to B}=A\cdot\underline{B}$	Запрет (неимпликация от А к В)
F_3	0	0	1	1	$F_3=A$	Тождество А
F_4	0	1	0	0	$F_4=\underline{B\to A}=\underline{A}\cdot B$	Запрет (неимпликация от В к А)
F_5	0	1	0	1	$F_5=B$	Тождество В
F_6	0	1	1	0	$F_6=A\oplus B=\underline{A}\cdot B+A\cdot\underline{B}$	Исключительное ИЛИ (неравнозначность)
F_7	0	1	1	1	$F_7=A+B$	Дизъюнкция (ИЛИ)
F_8	1	0	0	0	$F_8=A\downarrow B=\underline{A+B}=\underline{A}\cdot\underline{B}$	ИЛИ-НЕ (стрелка Пирса)
F_9	1	0	0	1	$F_9=A\equiv B=\underline{A\oplus B}=A\cdot B+\underline{A}\cdot\underline{B}$	Исключительное ИЛИ-НЕ (равнозначность)
F_{10}	1	0	1	0	$F_{10}=\underline{B}$	Отрицание В
F_{11}	1	0	1	1	$F_{11}=B\to A=A+\underline{B}$	Импликация от В к А
F_{12}	1	1	0	0	$F_{12}=\underline{A}$	Отрицание А
F_{13}	1	1	0	1	$F_{13}=A\to B=\underline{A}+B$	Импликация от А к В
F_{14}	1	1	1	0	$F_{14}=A\mid B=\underline{A\cdot B}=\underline{A}+\underline{B}$	И-НЕ (штрих Шеффера)
F_{15}	1	1	1	1	$F_{15}=1$	Константа единица

Каждая строчка таблицы определяет соответствующую функцию посредством еще одной таблицы, называемой таблицей истинности. Например, вторая строчка определяет конъюнкцию (табл. 1.4).

Некоторые из этих функций являются постоянными (две константы — ноль и единица), некоторые — переменными А, В и их отрицаниями, а большая часть — логическими операциями, причем не только базовыми. Это и импликация, и равнозначность (эквивалентность) и др., которые представляются в виде выражения с базовыми

операциями, но имеют и собственные обозначения. Впрочем, набор базовых операций, базис, может быть любым.

Таблица 1.4

Таблица истинности операции И (конъюнкции)

A	B	F = A · B
0	0	0
0	1	0
1	0	0
1	1	1

Логические операции технически можно реализовать с помощью каких-либо переключателей («замкнуто-разомкнуто» — «истина-ложь» — «единица-ноль»): механических, электромеханических (реле) или электронных (электровакуумных ламп (триодов) или транзисторов) (рис. 1.3, 1.4).

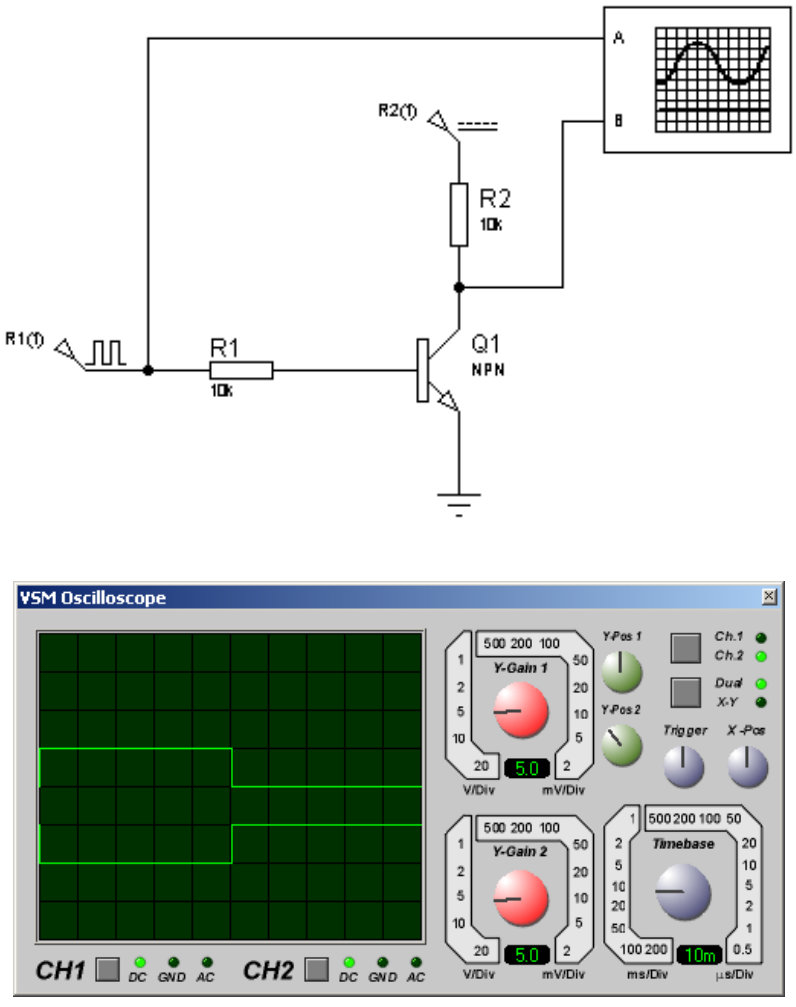


Рис. 1.3. Инвертор (операция НЕ) на транзисторе NPN

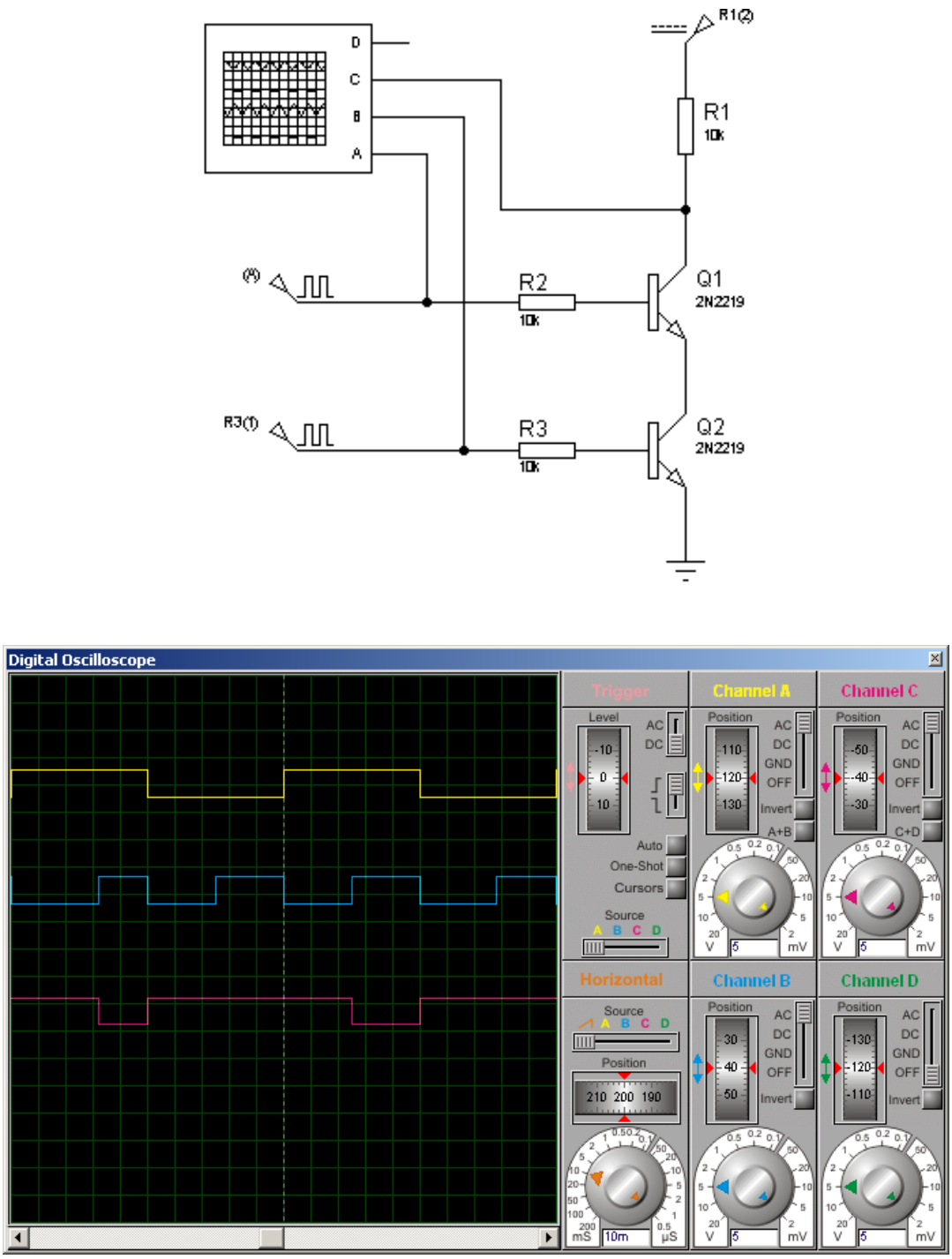


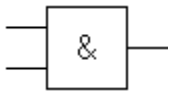
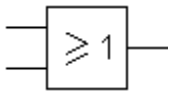
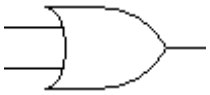
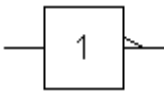
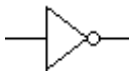
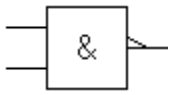
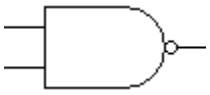
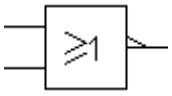
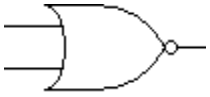
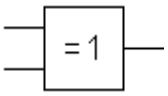
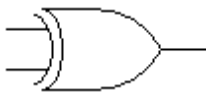
Рис. 1.4. Логический элемент И-НЕ на двух транзисторах NPN

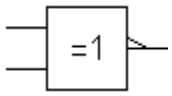
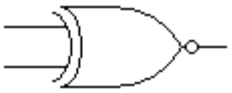
Схемы и диаграммы на рисунках созданы с помощью демонстрационной версии программного комплекса Proteus.

В настоящее время логические операции реализуются с помощью микросхем, основными элементами которых являются те же транзисторы (табл. 1.5).

Таблица 1.5

Логические микросхемы

№	Операция	Международное обозначение (соответствует ГОСТ 2.743-91)	Американское обозначение	Таблица истинности
1	И			X1 X2 Y 0 0 0 0 1 0 1 0 0 1 1 1
2	ИЛИ			X1 X2 Y 0 0 0 0 1 1 1 0 1 1 1 1
3	НЕ			X Y 0 1 1 0
4	И-НЕ (НЕ-И)			X1 X2 Y 0 0 1 0 1 1 1 0 1 1 1 0
5	ИЛИ-НЕ (НЕ-ИЛИ)			X1 X2 Y 0 0 1 0 1 0 1 0 0 1 1 0
6	Исключительное ИЛИ (неравнозначность)			X1 X2 Y 0 0 0 0 1 1 1 0 1 1 1 0

<table>
<tr><th>№</th><th>Операция</th><th>Международное обозначение (соответствует ГОСТ 2.743-91)</th><th>Американское обозначение</th><th>Таблица истинности</th></tr>
<tr><td>7</td><td>Исключительное ИЛИ-НЕ (равнозначность или эквивалентность)</td><td>
</td><td></td><td>
<table>
<tr><td>X1</td><td>X2</td><td>Y</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

Как легко заметить, кружок в обозначении выходного вывода логического элемента означает инверсию, т. е. отрицание.

Отметим сразу, что в системе команд любого процессора имеются команды для основных логических операций.

Микросхема (интегральная схема) — это с конструктивной точки зрения электронный элемент, а с функционально-логической — устройство (схема), состоящее из транзисторов, резисторов, других необходимых элементов и выполняющее некоторую функцию, некоторое преобразование информации. Электронные элементы микросхемы со всеми связями между ними реализуются в сложном многоэтапном технологическом процессе на кристалле полупроводника, чаще всего кремния. Затем кристалл (чип) заключается в корпус, через который делаются необходимые выводы для входных и выходных сигналов. Основной элемент полупроводниковой микросхемы — транзистор. Именно количество транзисторов на кристалле определяет такую важную характеристику микросхемы, как степень интеграции.

Сколь угодно сложную логическую функцию любого числа логических переменных можно представить в виде логического выражения в некотором базисе (базовом наборе логических операций) и реализовать в виде схемы из соответствующих логических элементов. И с технической точки зрения наиболее удобными базисами являются И-НЕ (НЕ-И, Not AND, NAND) или ИЛИ-НЕ (НЕ-ИЛИ, Not OR, NOR), поскольку для реализации любой функции достаточно только одного типа таких элементов. Конечно, сложные логические

устройства реализуются чаще всего не в виде платы из отдельных логических микросхем, а в виде более сложной микросхемы. Но это не самое главное и не самое интересное. Самое главное и интересное — из логических элементов в ЭВМ делается все без исключения — не только сложные логические функции, но и арифметика, и память, и «моторчик» (генератор тактовых импульсов). Получается всего два уровня: логические элементы делаются из переключателей (транзисторов на данном этапе развития электронной техники), а все остальное — из логических элементов.

Сумматор или **полусумматор** — это превращение формальной логики в арифметику.

Для создания устройства, реализующего какую-либо логическую функцию, если нельзя ее написать сразу, необходимо составить таблицу истинности устройства, по таблице истинности составить логическое выражение в виде СДНФ (совершенной дизъюнктивной нормальной формы) или СКНФ (совершенной конъюнктивной нормальной формы), а затем минимизировать его и привести к необходимому логическому базису.

Для получения таблиц истинности полусумматора и сумматора представим сложение двух двоичных чисел в столбик. Складываются младшие разряды слагаемых, получается значение младшего разряда суммы (0 или 1) и переноса в следующий разряд (0 или 1). Это делает полусумматор. Для получения значений последующих разрядов суммы складываются соответствующие разряды слагаемых и перенос из предыдущего разряда. При этом также получается перенос в следующий разряд. Это делает сумматор.

Таким образом, в таблице истинности полусумматора две входные логические переменные, соответствующие младшим разрядам слагаемых, и две выходные логические функции: сумма и перенос в следующий разряд (табл. 1.6).

Таблица 1.6

Таблица истинности полусумматора

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Здесь даже не надо составлять СДНФ или СКНФ. Сразу видно, что сумма — это Исключительное ИЛИ или неравнозначность:

$$S = A \oplus B,$$

а перенос — конъюнкция:

$$C = A \cdot B.$$

Таким образом, полусумматор можно реализовать в виде схемы, изображенной на рис. 1.5.

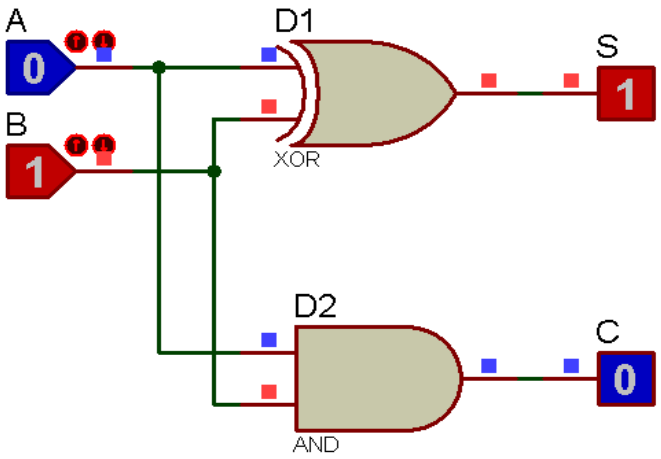


Рис. 1.5. Схема полусумматора

В таблице истинности сумматора результирующие функции те же, а входных переменных — три: текущие разряды слагаемых и перенос из предыдущего разряда (табл. 1.7).

Таблица 1.7

Таблица истинности сумматора

C0	A	B	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Здесь также сразу видно, что:

$$S = C0 \oplus (A \oplus B),$$
$$C = C0 \cdot (A \oplus B) + (A \cdot B).$$

Схема сумматора изображена на рис. 1.6.

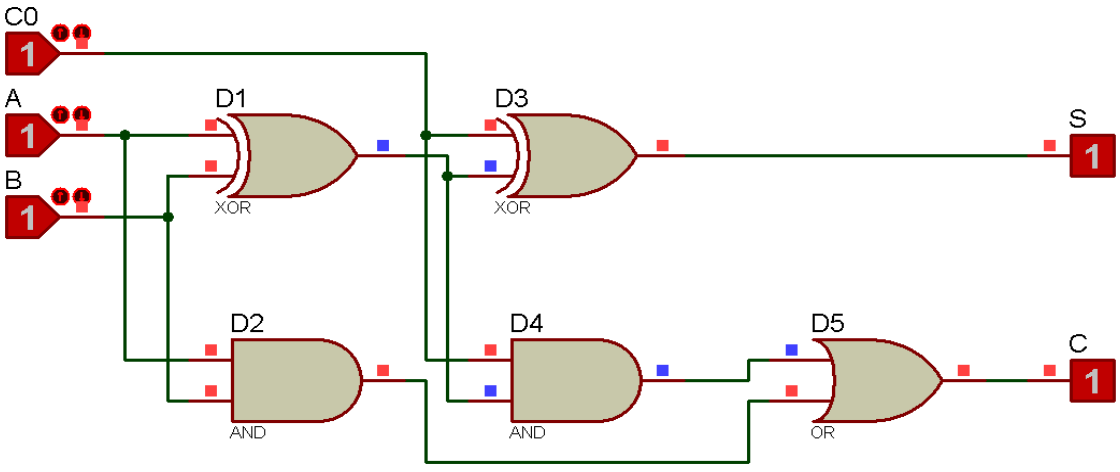


Рис. 1.6. Схема сумматора

Чтобы создать устройство сложения восьмиразрядных двоичных чисел (байтов), нужно взять один полусумматор и семь сумматоров.

Общее свойство схем такого типа, подобных сумматору и полусумматору, — однозначная зависимость между входом и выходом (не взаимнооднозначная!), т. е. одной комбинации входных значений соответствует одно и только одно значение выхода. Такие схемы называются комбинационными или, другими словами, относятся к группе комбинационной логики. У них нет внутреннего состояния, т. е. памяти и, соответственно, времени. Во времени они существуют не на интервале, а в точке. В компьютере время идет, «моторчик» крутится, «часы тикают», но при изменении логических значений на входе такого типа схемы предыдущее значение выхода сразу «забывают».

Как же можно реализовать схемы с памятью из схем без памяти? Нужно вспомнить один из основных системотехнических принципов. Система только тогда является системой, когда имеет свойства, отличные от свойств ее отдельных элементов. И новые свойства системы порождают именно связи между элементами.

Формальную логику в память превращает такое устройство, как триггер. Схемы с памятью называются последовательностными или, другими словами, относятся к классу последовательностной логики. У них теряется однозначная зависимость между входом и выходом, появляется внутреннее состояние (память), текущее значение выхода зависит не только от входа, но и от предыдущего или предыдущих состояний, их последовательности. Соответственно во времени они существуют уже не в точке, а на некотором интервале. **Триггер** — это «атом» последовательностных логических схем, минимальное устройство с памятью. Он может находиться в одном из двух состояний, 0 или 1, и, соответственно, хранить один бит информации. Триггер является важнейшим «кирпичиком» мироздания. Триггеры существуют не только в электронике, но и повсюду в природе и соответствующих областях знания: физике, биологии, психологии и т. д. Примеры механического триггера — тумблер, механизм раскрыва-

ния-закрывания зонта и т. п. Более общее название такого рода элементов — элементы с бистабильностью, т. е. такие, у которых имеется только два устойчивых состояния и одно неустойчивое между ними.

Триггер можно реализовать из двух логических элементов И-НЕ или ИЛИ-НЕ, соединив их перекрестными обратными связями, так что выход каждого элемента подключен к одному из входов другого элемента (рис. 1.7).

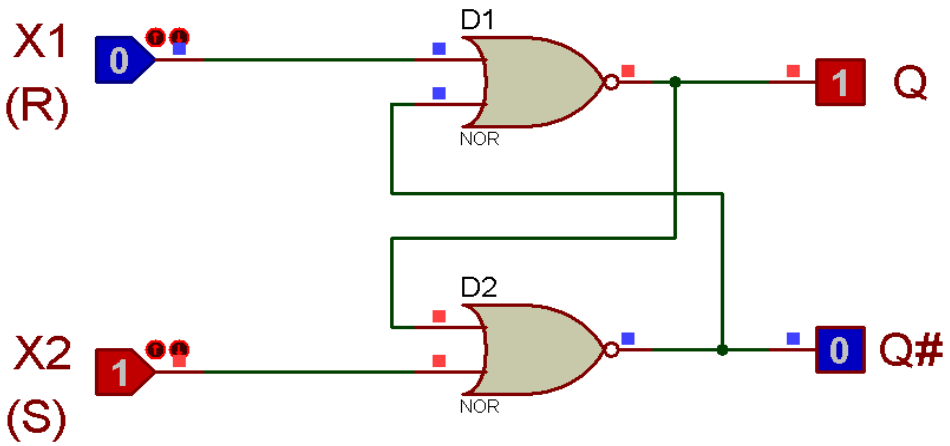


Рис. 1.7. Триггер из элементов ИЛИ-НЕ

У триггера два выхода — *прямой* и *инверсный*. Их значение должно быть противоположным. Здесь мы использовали еще одно, самое распространенное обозначение элемента ИЛИ-НЕ. Мы помним его таблицу истинности (табл. 1.5). Составим на ее основе таблицу истинности триггера, начав со значения на входе X1 = 0, X2 = 1 (табл. 1.8).

Таблица 1.8

Таблица истинности триггера

X1	X2	Q	Q#
0	1	1	0
0	0	1	0
1	0	0	1
0	0	0	1
1	1	0	0

В соответствии с таблицей истинности элемента ИЛИ-НЕ выход примет значение: Q = 1, Q# = 0.

Если теперь на X2 подать 0, ничего не изменится. Теперь подадим на X1 единицу. Выход примет значение Q = 0, Q# = 1. Сбросим X1 в ноль — выход остался прежним. Теперь видно, что одному значению входа соответствует два различных значения выхода. Однозначное соответствие нарушилось. Выход зависит не только от входа, но и от предыдущего состояния. Наконец, если на вход подать две единички, на выходе будет два нуля, что для триггера недопустимо, и такую ситуацию необходимо исключить тем или иным способом.

С учетом сказанного, таблицу истинности триггера удобнее представить в другом виде, введя для выходных сигналов индексы, обозначающие дискретное время (табл. 1.9).

Таблица 1.9

Другая таблица истинности триггера

X1 (R)	**X2 (S)**	**Q_n**	**Q_n#**	**Состояние**
0	0	Q_{n-1}	Q_{n-1}#	Хранение
0	1	1	0	Установка (запись единицы)
1	0	0	1	Сброс (запись нуля)
1	1	0	0	Запрещенное (недопустимое)

Первая строка соответствует хранению информации, вторая и третья — записи информации, четвертая должна быть исключена. Теперь можно ввести новые обозначения входных сигналов в соответствии с их смыслом по отношению к прямому выводу: X1 — R (Reset, сброс), X2 — S (Set, установка). Когда S активен (равен 1), а R неактивен, прямой выход устанавливается, т. е. в триггер записывается 1, когда R активен (равен 1), а S неактивен, прямой выход сбрасывается, т. е. в триггер записывается 0. Когда оба входных сигнала не-

активны, триггер хранит записанный в него бит информации. Ну и ситуация, когда они оба активны, не имеет смысла.

Триггер можно реализовать непосредственно из переключателей, транзисторов. И конечно, существуют микросхемы триггеров различных типов. Описанный выше триггер — это асинхронный RS-триггер. Существуют и синхронные триггеры. У них на один из входов подается тактовый сигнал, по которому в триггер записывается определенное значение в соответствии с информационными входами.

Для хранения большего количества информации нужно взять соответствующее число триггеров. Устройство, которое может хранить более одного бита информации, называется регистром. **Регистр** — это совокупность триггеров. Объем информации соответствует степеням двойки: 8, 16, 32, 64 бита и т. д. Для хранения еще большего объема информации предназначены микросхемы памяти. Подробно тема памяти будет рассмотрена в соответствующем разделе. Здесь пока что отметим, что из триггеров состоит статическая память, к которой относятся отдельные регистры, регистры процессора и других микросхем, а также кэш-память. В динамической памяти основой запоминающей ячейки является конденсатор, и запоминание осуществляется посредством физического процесса заряда и разряда конденсатора. Конденсатор заряжен — 1, разряжен — 0.

Итак, память получилась из логики введением обратных связей. Точно так же из логики можно получить колебания, только здесь достаточно одного самого простого логического элемента — инвертора (элемента НЕ) — и перекрестных связей уже не нужно, обратная связь одна (рис. 1.8).

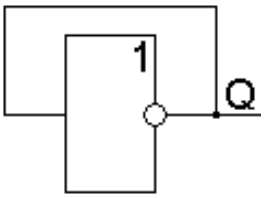


Рис. 1.8. Генератор прямоугольных колебаний из инвертора

Этому простейшему способу соответствует парадокс, называемый «лжец» в самой простой его формулировке: «Я лгу». Или: «Высказывание, которое я сейчас произношу, является ложным». Инвертор на рисунке 1.8 как раз и является таким «лжецом».

Более сложной, но более понятной формулировкой этого парадокса является следующая. На первой стороне листа бумаги написано: «На обратной стороне этого листа напечатано истинное высказывание». А на второй стороне: «На обратной стороне этого листа напечатано ложное высказывание».

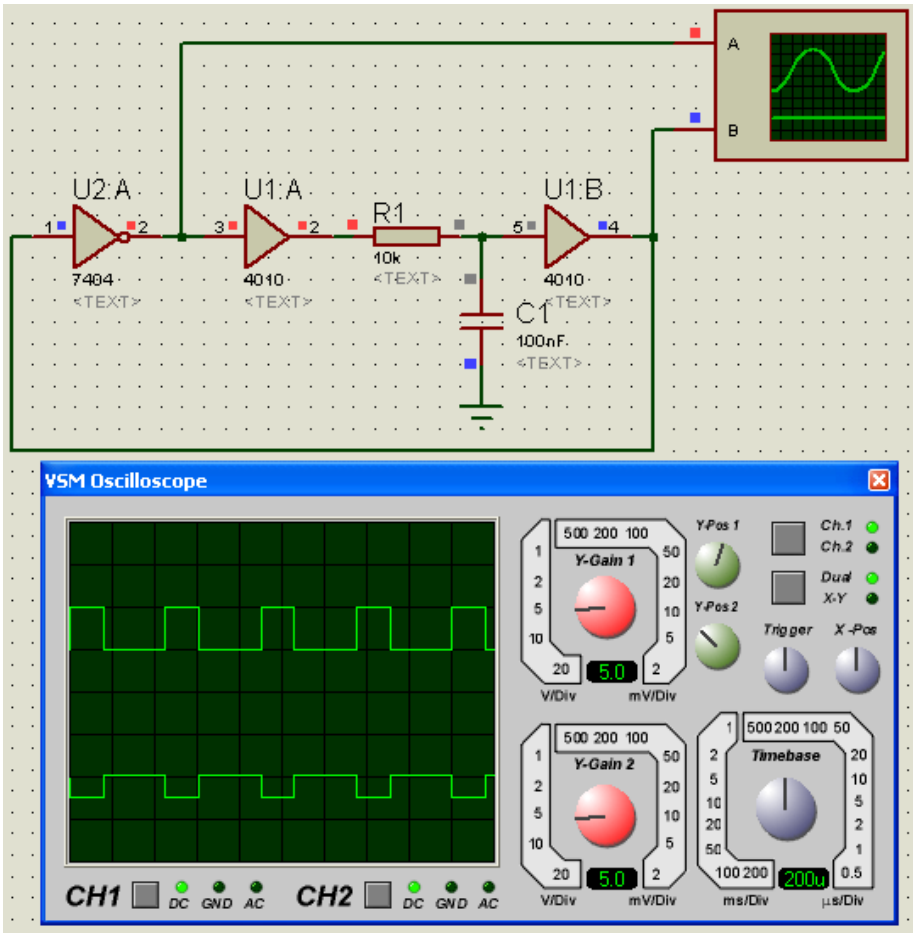


Рис. 1.9. Генератор из инвертора с задержкой в цепи обратной связи

Проведем эксперимент. Прочитаем первое утверждение и поверим ему, то есть сочтем истинным. Переворачиваем лист и читаем второе высказывание, считая его истинным. В силу этого мы должны принять, что первое высказывание ложно. Переворачиваем лист и чи-

таем первое высказывание, считая его ложным. Снова переворачиваем, и читаем второе высказывание, считая его ложным. Переворачиваем лист и читаем первое высказывание, считая его истинным. На этом эксперимент заканчивается. Совершился полный период колебаний. Для этого нам пришлось перевернуть лист четыре раза. То есть один переворот соответствовал четверти периода колебаний. Величина периода зависит от скорости переворачивания листа и скорости чтения текста. Это детали, которые выходят за рамки самого парадокса, но они важны при его технической реализации в тактовом генераторе. Ведь он должен генерировать прямоугольный сигнал определенного периода и, соответственно, частоты. Чтение и переворачивание листа в словесной формулировке парадокса соответствует введению схемы задержки в цепь обратной связи (рис. 1.9).

Период и частота колебаний такого генератора будут определяться параметрами схемы задержки, в частности постоянной времени RC-цепочки.

Первые цифровые вычислительные машины

Z1 — механический двоичный программируемый компьютер — был создан в 1938 г. [8]. Его разработал и изготовил на собственные деньги немецкий инженер Конрад Цузе (Konrad Zuse), работавший на предприятии Хеншель. Это был первый компьютер на основе двоичной логики Буля и двоичной системы счисления. Он включал все типы устройств, характерных для современных компьютеров: устройство управления, арифметическое устройство, память, устройства ввода-вывода. Таким образом, можно сказать, Z1 предвосхищал архитектуру фон Неймана. Единственное отличие, команды программы с перфоленты не вводились во внутреннюю память, а исполнялись сразу.

Основные характеристики Z1 таковы:

− занимаемая площадь 4 м^2, масса около 1 т;

− основа — механические элементы с электроприводом;

− тип обрабатываемых данных — вещественные числа с плавающей точкой в двоичном формате;

− тактовая частота 1 Гц, средняя производительность — 1 умножение за 5 с;

− ввод-вывод в десятичном формате;

− устройства ввода: клавиатура на основе пишущей машинки, устройство чтения перфолент;

− устройство вывода — ламповая панель.

Z1 работал недостаточно надежно, и в 1940 г. был создан следующий компьютер — Z2, уже частично электромеханический. Устройство управления и арифметическое устройство Z2 были выполнены на основе телефонных реле. Частота была увеличена до 5 Гц.

Следующая машина Цузе, Z3, была завершена в 1941 г. Она была уже целиком построена на телефонных реле, т. е. являлась полностью электромеханической. Количество реле составляло 2 600, тактовая частота — 5–10 Гц. Арифметическое устройство при вычислениях использовало два регистра, организованных в виде стека. Но, надо отметить, условные переходы и циклы отсутствовали. Конечно, Цузе знал про более совершенную элементную базу, электронные лампы, уже давно существовавшие в то время, и планировал использовать их, но не получил финансирование.

После войны серия Z была продолжена. Для программирования своих компьютеров Цузе разработал первый в мире высокоуровневый язык программирования, названный им Планкалкюль (нем. *Plankalkül*, исчисление планов).

ABC, компьютер Атанасова — Берри (Atanasoff-Berry Computer), был создан в 1942 г. профессором математики и физики колледжа штата Айова Джоном Атанасовым (John Atanasoff) с помощью своего аспиранта Клиффорда Берри (Clifford Berry). Это было специализированное непрограммируемое электронное вычислительное устройство. ABC был разработан исключительно для решения систем линейных уравнений. Однако его преимуществами были использование двоичной системы счисления и памяти на конденсаторах. Ввод-вывод осуществлялся в десятичном формате. Тип данных — вещественные числа с фиксированной точкой. Арифметико-логическое устройство было реализовано на лампах, а устройство управления — на реле. Органы управления — переключатели и пере-

мычки. Для ввода данных использовались стандартные перфокарты IBM, для вывода — дисплей на передней панели.

Colossus Mark 1, первый компьютер из серии Colossus, начал использоваться в 1943 г. в Великобритании. Это был специализированный программируемый электронный цифровой компьютер, разработанный для декодирования шифрованных немецких сообщений. Его основным разработчиком был инженер-исследователь телефонной связи главного почтового управления Томми Флауэрс (Tommy Flowers). Colossus Mark 1 содержал 1 500 электровакуумных ламп. Программа задавалась переключателями и штекерами. Данные вводились с перфоленты. Устройством вывода была электрическая пишущая машинка. Имелся также дисплей в виде ламповой панели. В 1944 г. вступила в строй усовершенствованная модель Colossus Mark 2. Mark 2 содержал 2 500 ламп и имел более высокую производительность (быстродействие). Позже было изготовлено еще несколько компьютеров этой серии.

Компьютер ASCC (Automatic Sequence Controlled Calculator), более известный как Harvard Mark I, начал работу в 1944 г. в Гарвардском университете. Это было электромеханическое программируемое устройство. Автором идеи был американский физик Говард Эйкен (Howard Aiken). Навеяна она была Аналитической и Разностной машинами Чарльза Бэббиджа. Эйкен занимался исследованием движения заряженных частиц в электровакуумных лампах и столкнулся с необходимостью численного решения нелинейных обыкновенных дифференциальных уравнений. Компьютер был спроектирован и изготовлен в компании IBM рядом инженеров под непосредственным руководством Эйкена.

Некоторые характеристики Harvard Mark I:

− габариты: длина 16 м, высота 2,4 м, глубина 0,6 м;

− масса 4,3 т;

− энергопотребление 3,7 кВт;

− 765 тысяч деталей (в том числе 3 500 реле);

− 800 км проводов.

Поскольку реализация осуществлялась IBM, то в основу конструкции были положены решения, использовавшиеся в это время в счетно-перфорационных машинах. По всей длине компьютера про-

ходил непрерывно вращавшийся вал, приводивший в действие все его узлы. Вычисления осуществлялись с использованием десятичной арифметики. Компьютер имел несколько вычислительных блоков, работавших параллельно, и программировался с помощью перфолент. Причем использовались два устройства считывания с двух отдельных перфолент. С одной ленты вводились команды, с другой — данные. Такой принцип раздельной памяти команд и данных впоследствии получил название «гарвардской архитектуры». Для ввода констант использовались поворотные переключатели.

Компьютер выполнял операции сложения и вычитания за 0,3 с (время одного оборота приводного вала), умножения и деления — в среднем за 6 и 15,3 с.

Команды условного перехода не было, но в 1946 г. ASCC был дополнен еще одним устройством считывания со специальной перфоленты, позволявшим организовать ветвление программ.

В дальнейшем было изготовлено еще несколько компьютеров этой серии, последний из которых, Harvard Mark IV (1952 г.), уже был полностью электронным устройством.

ENIAC (Electronic Numerical Integrator and Calculator) был создан в 1945 г. в Электротехнической школе Мура Пенсильванского университета. Основными разработчиками ENIAC были профессор физики Джон Мочли (John Mauchly) и инженер Преспер Эккерт (J. Presper Eckert). В сентябре 1944 г. в качестве научного консультанта к проекту присоединился известный математик Джон фон Нейман (John von Neumann), работавший в Институте перспективных исследований (IAS, Institute for Advanced Study) в Принстоне.

Основные характеристики ENIAC:

− 18 000 электровакуумных ламп, 7 200 кристаллических диодов, 1 500 реле, 70 000 резисторов, 10 000 конденсаторов;

− масса 27 т;

− память данных — 20 аккумуляторов для хранения десятичных чисел и выполнения операций над ними;

− потребляемая мощность 150 КВт;

− производительность — 357 операций умножения или 5 000 операций сложения в секунду;

− тактовая частота 100 КГц;

− устройство ввода-вывода — табулятор перфокарт;
− десятичная система счисления;
− программирование — коммутация проводов;
− средняя наработка на отказ 20 ч.

Таким образом, производительность ENIAC была намного больше, чем Harvard Mark I, но надежность оставляла желать лучшего. Он отказывал в среднем каждые 20 часов.

Еще во время работы над ENIAC рабочей группой была начата разработка нового компьютера, более производительного, простого и надежного, позже названного EDVAC (Electronic Discrete Variable Automatic Computer). В 1944 г. Эккерт разработал для него память, которая содержала бы как данные, так и команды программы. В 1945 г. фон Нейман изложил концепцию EDVAC в своем знаменитом «Первом проекте отчета по EDVAC» (First Draft of a Report on the EDVAC), который предназначался для внутреннего использования, но был распространен в ряде правительственных и образовательных учреждений куратором проекта от армии Германом Голдстайном (Herman H. Goldstine). На документе стояло только имя фон Неймана, поэтому изложенные в нем положения стали называться впоследствии архитектурой фон Неймана, хотя авторами основных архитектурных решений были Мочли и Эккерт.

В марте 1946 г. Мочли и Эккерт покинули школу Мура и основали компанию Electronic Control Company, которая позднее была переименована в Eckert-Mauchly Computer Corporation. В 1949 г. компания выпустила свою первую электронную вычислительную машину BINAC (Binary Automatic Computer), которая для ввода информации вместо перфокарт впервые использовала магнитную ленту. В 1950 г. их компания, испытывавшая финансовые затруднения, была приобретена компанией Remington Rand и преобразована в подразделение UNIVAC, в котором в 1951 г. Мочли и Эккерт разработали свой первый коммерческий массовый компьютер UNIVAC I (UNIVersal Automatic Computer).

После завершения проекта ENIAC фон Нейман продолжил работу в Институте перспективных исследований в Принстоне, где под его непосредственным руководством в 1951 г. был создан компьютер IAS machine.

Уход ведущих разработчиков из проекта EDVAC привел к тому, что работы над ним затянулись на три года. В результате первой действующей и практически используемой машиной, реализовавшей архитектуру фон Неймана, стал британский компьютер EDSAC (Electronic Delay Storage Automatic Computer), созданный Моррисом Уилксом (Maurice Wilkes) в Кембриджском университете в 1949 г.

EDVAC (Electronic Discrete Variable Automatic Computer) был поставлен школой Мура армейскому заказчику в 1949 г., но заработал только в 1952 г. после длительного процесса устранения неполадок. Это был двоичный компьютер с хранимой программой, т. е. с архитектурой фон Неймана. EDVAC включал следующие компоненты:

− устройство чтения-записи магнитной ленты;

− блок управления с осциллографом;

− диспетчерский блок для приема команд от блока управления и памяти и передачи их в другие блоки;

− вычислительный блок для выполнения арифметических операций над парой чисел и отправки результата в память после проверки на дублирующем блоке;

− таймер;

− блок сдвоенной памяти, состоящий из двух наборов по 64 ртутных акустических линий задержки емкостью по восемь слов в каждой линии;

− три временных хранилища (регистра) для линий задержки, каждый из которых емкостью в одно слово.

Основные характеристики EDVAC:

− тип данных — вещественные числа с плавающей точкой;

− производительность около 340 операций в секунду;

− 5 937 электровакуумных ламп и 12 000 диодов;

− энергопотребление 56 кВт;

− площадь 45,5 м^2;

− масса 7,8 т.

После завершения EDVAC школа Мура больше не занималась созданием компьютеров. Основные характеристики первых компьютеров представлены в табл. 1.10.

Таблица 1.10

Основные характеристики первых компьютеров

Компьютер	Элементная база	Универсальность	Арифметика	Программирование	Память	Ввод-вывод
Z1, 1938 г.	Механика	Универсальный	Двоичная	Перфолента, клавиатура	Металлические пластины	Клавиатура, устройство чтения перфолент, ламповая панель
Z2, 1940 г.	Механика, реле			Перфолента, перфокарты, клавиатура	Металлические пластины	
Z3, 1941 г.	Реле			Перфолента, клавиатура	Реле	
ABC, 1942 г.	Электровакуумные лампы, реле	Специализированный	Двоичная	Непрограммируемый	На конденсаторах	Переключатели, перемычки, считыватель перфокарт, дисплей
Colossus Mark 1, 1943 г.	Электровакуумные лампы	Специализированный	Двоичная	Переключатели и штекеры	Отсутствовала	Ввод — перфолента. Вывод — электрическая пишущая машинка, ламповая панель
ASCC (Harvard Mark I), 1944 г.	Механика, реле	Универсальный	Десятичная	Две перфоленты: с командами и данными	Зубчатые колеса и переключатели	Устройства чтения перфолент, поворотные переключатели
ENIAC, 1945 г.	Электровакуумные лампы	Универсальный	Десятичная	Коммутация проводов	20 аккумуляторов для хранения данных	Устройство чтения перфокарт IBM, перфоратор
EDVAC, 1952 г.	Электровакуумные лампы и диоды	Универсальный	Двоичная	Ввод программы во внутреннюю память	На основе ртутных линий задержки	Устройство чтения-записи магнитной ленты, осциллограф

В нашей стране электроника и вычислительная техника развивались так же успешно. 27 апреля 1918 г. на заседании Российского общества радиоинженеров Михаил Александрович Бонч-Бруевич выступил с докладом «Комбинированные характеристики катодных реле» (электронных радиоламп) [9].

В докладе говорилось о созданном им устройстве с двумя устойчивыми рабочими состояниями, которое он использовал в Твери летом 1917 г. для приема радиосигналов (рис. 1.10).

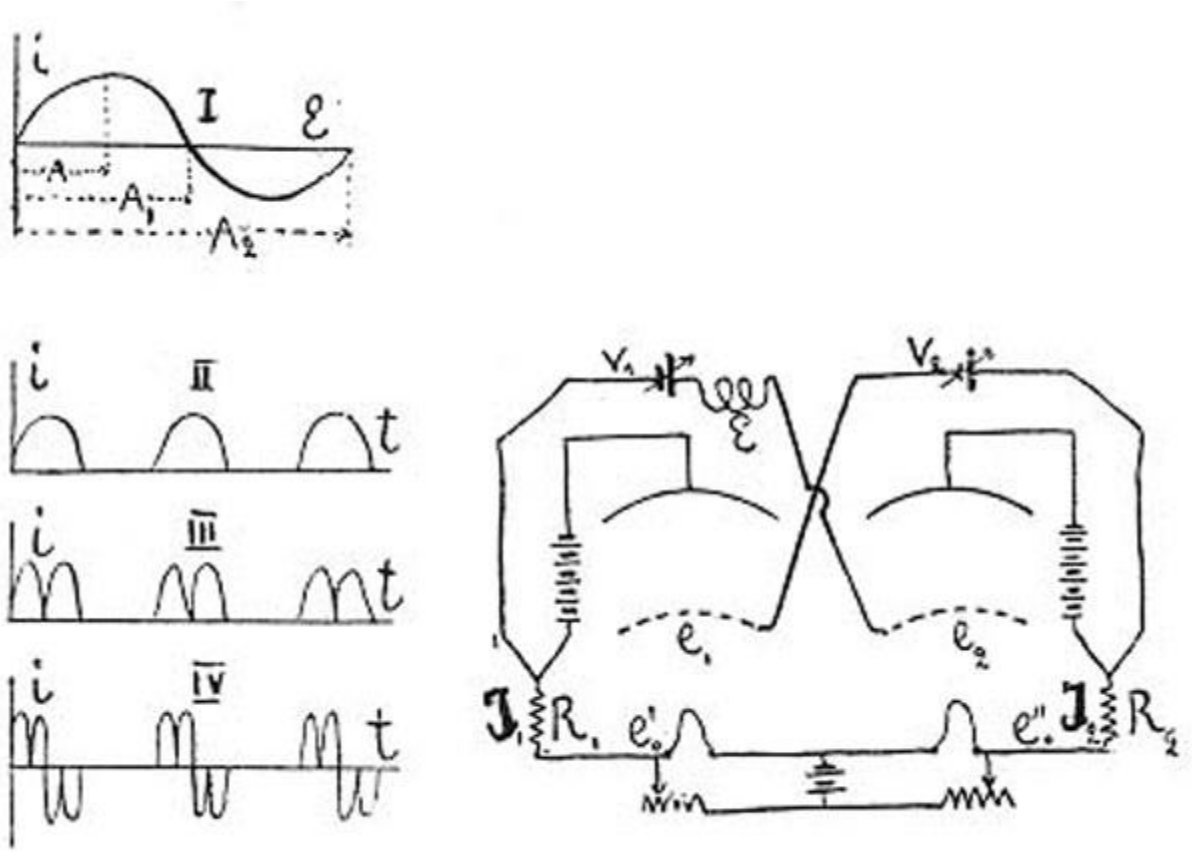


Рис. 1.10. Схема триггера М. А. Бонч-Бруевича

Впоследствии такое устройство получило название триггер (flip-flop) и стало одним из основных элементов ЭВМ.

Считающиеся изобретателями триггера Вильям Экклз (William Eccles) и Фрэнк Джордан (Frank Jordan) подали заявку на патент в 1918 г.

4 декабря 1948 г. чл.-корр. АН СССР д-р техн. наук Исааку Семеновичу Бруку и инженеру Баширу Искандеровичу Рамееву было выдано авторское свидетельство № 10475 — первый советский патент на автоматическую цифровую ЭВМ [10].

Сейчас 4 декабря отмечается как Всероссийский день информатики.

В 1935 г. Брук создал свою первую вычислительную машину «электрический стол переменного тока» — аналоговое устройство для моделирования и расчета сетей электроснабжения (хранится в Московском политехническом музее).

Затем занимался разработкой механического интегратора дифференциальных уравнений.

После войны в лаборатории Брука была создана электронная аналоговая машина ЭДА, интегрировавшая дифференциальные уравнения до 20-го порядка.

В августе 1948 г. Брук и Рамеев представили первый в нашей стране проект «Автоматическая цифровая электронная машина» с описанием принципиальной схемы ЭВМ и арифметических операций в двоичной системе (рис. 1.11).

Брук и Рамеев реализовали принцип хранения управляющей программы в памяти, что соответствует архитектуре фон Неймана. Программа записывалась на перфоленте и вводилась в машину, результаты вычислений выдавались на другой ленте, и полученная информация снова вводилась с нее в машину для следующего цикла.

В 1950–1951 гг. под руководством Исаака Семеновича Брука была разработана ЭВМ М-1, первая в мире ЭВМ, в которой логические схемы были выполнены на полупроводниковых элементах.

Основные характеристики М-1:

− система счисления — двоичная;

− формат чисел — 25-разрядные вещественные с фиксированной точкой;

− два вида внутренней памяти:

1) медленная, на магнитном барабане — 256 чисел;

2) быстрая, на электростатических трубках — 256 чисел;

− быстродействие:

1) с магнитной памятью — 20 операций в секунду;

2) с электронной памятью для операций сложения — 20 000 операций в секунду, для операций умножения — 500 операций в секунду;

− ввод информации и программ с перфоленты;

− вывод результатов на широкоформатном телетайпе.

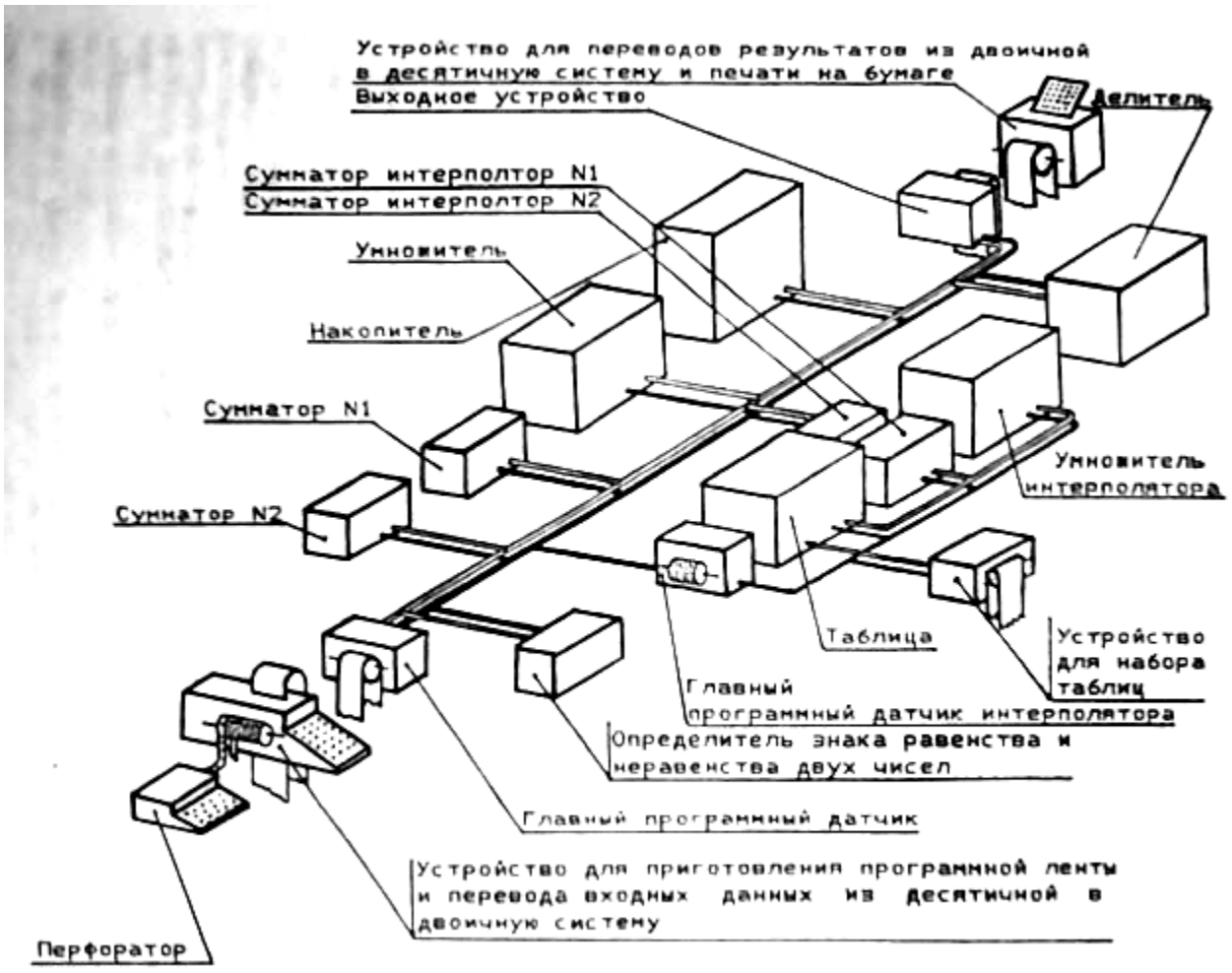


Рис. 1.11. Схема автоматической цифровой вычислительной машины Брука и Рамеева

В 1953 г. на основе разработок Брука и Рамеева была создана первая отечественная серийная ЭВМ «Стрела». Ее генеральным кон-

структором был Юрий Яковлевич Базилевский, Рамеев — его заместителем. Как заместитель главного конструктора Рамеев работал над всем проектом, но самостоятельно разработал арифметико-логическое устройство и память на магнитном барабане.

«Стрела» имела следующие основные характеристики:

− производительность 2 000 операций в секунду;

− 8 000 электронных ламп и 2 000 полупроводниковых диодов;

− оперативное запоминающее устройство (ОЗУ) на электронно-лучевых трубках;

− постоянное запоминающее устройство (ПЗУ) на полупроводниковых диодах;

− внешние запоминающие устройства: накопители на магнитной ленте, последний вариант «Стрелы» использовал память на магнитном барабане;

− ввод данных — с перфокарт и с магнитной ленты, вывод данных — на магнитную ленту, перфокарты и широкоформатный принтер;

− потребляемая мощность 150 кВт;

− занимаемая площадь 300 м^2.

«Стрела» эксплуатировалась в математическом институте им. Стеклова АН СССР, МГУ им. М. В. Ломоносова, ВЦ АН СССР, ядерных центрах, ЦНИИ-27/ВЦ-1 Министерства обороны, Центре управления космическими полетами. «Стрела» использовалась для расчетов орбит запускаемых искусственных спутников, управления полетами пилотируемых кораблей, для расчетов при разработке первой советской водородной бомбы РДС-6с, аэродинамических расчетов для первого советского пассажирского реактивного самолета Ту-104.

МЭСМ (Малая электронная счетная машина) выполнила первую программу 4 августа 1950 г. Была разработана Сергеем Алексеевичем Лебедевым в созданной им Лаборатории моделирования и вычислительной техники Киевского института электротехники АН УССР. Это также была двоичная машина с внутренней оперативной и постоянной памятью. В МЭСМ, как и практически во всех отечественных ЭВМ, был реализован принцип хранимой программы с общей памя-

тью для команд и данных. Работа над машиной носила исследовательский характер, МЭСМ создавалась как макет или модель Большой электронной счетной машины (БЭСМ) для экспериментальной проверки принципов построения универсальных цифровых ЭВМ, и первое время буква «М» в названии означала «модель». Но после того, как машина доказала пригодность для решения реальных практических задач, она была переименована в Малую электронную счетную машину.

Вскоре после этого Лебедев начал работу над компьютером БЭСМ в Институте точной механики и вычислительной техники АН СССР (ИТМ и ВТ), созданном в 1948 г., а на его место был приглашен Виктор Михайлович Глушков.

БЭСМ-1 была создана осенью 1952 г. в ИТМ и ВТ. Главный конструктор — Сергей Алексеевич Лебедев.

Основные характеристики БЭСМ-1:

− 4 000 электронных ламп и 5 000 полупроводниковых диодов;

− параллельное 39-разрядное арифметико-логическое устройство для обработки вещественных чисел с плавающей запятой;

− общее адресное пространство памяти для команд и данных 2K 39-разрядных ячеек;

− оперативное запоминающее устройство емкостью 2 048 чисел на электронно-лучевых трубках, как и у «Стрелы»;

− внешние запоминающие устройства: четыре накопителя на магнитных лентах по 30 000 слов и два на магнитных барабанах по 5 120 слов;

− ввод программы и исходных данных с перфоленты со скоростью 20 слов в секунду, вывод на фотопечатающее устройство;

− производительность 8–10 тыс. оп/с;

− потребляемая мощность 80 кВт.

По результатам международной конференции в Дармштадте (октябрь 1953 г.) БЭСМ-1 оказалась самой быстродействующей в Европе.

На ее основе был создан целый ряд военных и гражданских ЭВМ, вплоть до самой производительной и знаменитой — БЭСМ-6.

Поколения ЭВМ и появление персональных компьютеров

После появления первых ЭВМ их число стало расти экспоненциально. И даже не все самые известные и интересные из них здесь можно описать. Для этого нужна специальная книга по истории вычислений. Развитие вычислительной техники идет по двум основным направлениям — совершенствование архитектуры и элементной базы. С точки зрения элементной базы выделяются так называемые поколения ЭВМ:

− *первое поколение* — компьютеры на электронных лампах, 1940–1950-е гг.;

− *второе поколение* — компьютеры на транзисторах, 1950–1960-е гг.;

− *третье поколение* — компьютеры на интегральных схемах, 1960–1970-е гг.;

− *четвертое поколение* — компьютеры на сверхбольших интегральных схемах или микропроцессорах, с начала 1970-х гг. по настоящее время.

Эти сроки соответствуют основным вехам развития элементной базы:

− 1947 г. — изобретение транзистора;

− 1958, 1959 гг. — изобретение интегральной схемы;

− 1971 г. — изобретение микропроцессора.

Точнее их указать невозможно, поскольку в разных странах разрабатывалось множество различных типов и моделей компьютеров. Кроме того, в каждом десятилетии некоторое время выпускались еще компьютеры предыдущих поколений.

В рамках такого деления все современные ЭВМ относятся к четвертому поколению. Надо отметить, что в середине восьмидесятых годов прошлого века в Японии была принята программа создания ЭВМ пятого поколения. Она была широко разрекламирована, получила широкую известность во всем мире, но оказалась абсолютно

провальной. Предполагалось, что ЭВМ пятого поколения будут интеллектуальными системами, программируемыми с использованием не процедурного (алгоритмического) способа, а декларативного. В «обычной» процедурной программе описывается последовательность действий, которую должна выполнить ЭВМ, т. е. алгоритм. Декларативная программа представляет собой описание предметной области в терминах фактов, понятий и отношений между ними. Исходные данные формулируются в виде утвердительных предложений. Задача ставится в форме вопросительного предложения. Алгоритм здесь тоже реализуется, но самой системой программирования и, конечно, процессором в ходе поиска решения задачи.

В качестве такого декларативного языка программирования ЭВМ пятого поколения был выбран «Пролог» (Prolog, Programming in logic). Он основан на исчислении предикатов первого порядка. Это один из разделов логики наряду с алгеброй логики (математической логикой) и исчислением высказываний. Особенность его в том, что предикат — это логическая функция предметных переменных. Предметные переменные принимают значения из некоторого множества, определяемого предметной областью. Это может быть, например, множество действительных чисел или множество некоторых терминов или имен. А значение предиката — логическое двоичное, «истина» или «ложь». Скоро выяснилось, что этот дополнительный уровень использования двоичной логики в ЭВМ не только не делает ЭВМ интеллектуальной, но и вообще не дает никаких преимуществ, только создает дополнительные трудности. Энтузиасты, конечно, продолжают развивать «Пролог», но доля его использования по сравнению с другими языками программирования в настоящее время незначительна.

Так что вычислительная техника развивается в соответствии с принципом наименьшего действия, сформулированного Лейбницем. Архитектура фон Неймана — самый простой и очевидный, но не самый лучший способ вычислений. Тем не менее около ста процентов современных вычислительных систем имеют в своей основе эту ар-

хитектуру. Каждый человек в настоящее время использует такие компьютеры в своей работе и вообще в повседневной жизни. Такие однопользовательские системы, персональные компьютеры, появились еще в 60-х гг. прошлого столетия, но стремительное развитие получили после создания микропроцессора, т. е. в рамках ЭВМ четвертого поколения.

В качестве одной из первых персональных ЭВМ можно упомянуть ЭВМ «МИР», разработанную в 1965 г. под руководством Глушкова. Она имела микропрограммное управление и аппаратно реализованный машинный язык, близкий по возможностям к языкам программирования высокого уровня. Микропрограммное управление заключается в том, что устройство управления ЭВМ для формирования управляющих сигналов использует так называемую микропрограмму, находящуюся в постоянной памяти. ЭВМ «МИР» выпускалась серийно и предназначалась для использования в учебных заведениях, конструкторских бюро, научных организациях. Конечно, стоила она очень дорого, дороже автомобиля, по ценам 1973 г. — 39 500 руб., и была не маленькой.

Появление микропроцессора позволило создать компактные и производительные, по-настоящему доступные рядовому пользователю компьютеры. Первоначально они назывались микро-ЭВМ, позже появился термин «персональный компьютер» (Personal Computer, PC), а микро-ЭВМ стали называть интегральные схемы, которые помимо процессора содержали память, интерфейсы ввода-вывода и другие традиционные узлы ЭВМ. Другое название таких микросхем — микроконтроллеры.

Первые микро-ЭВМ были построены еще на основе микропроцессора Intel 8008 (1972 г.), второго по счету (первым был Intel 4004, 1971 г.). Это были SCELBI и Mark-8, выпущенные в 1974 г., однако они не получили широкого распространения.

Первым популярным персональным компьютером был Altair 8800. Он был разработан в 1974 г. на основе процессора Intel 8080

в компании MITS (Microinstumentation and Telemetry Systems). Главный разработчик — Генри Робертс (Henry Roberts).

Altair 8800 был изображен на обложке журнала Popular Electronics за январь 1975 г. Статью в журнале прочитали Пол Аллен (Paul Allen) и Билл Гейтс (Bill Gates). Они написали для компьютера Altair 8800 интерпретатор языка БЕЙСИК и вскоре после этого создали компанию Microsoft.

MITS удалось договориться с Intel о цене на процессоры i8080 в $75, хотя установленная Intel цена на них составляла в то время $360. Благодаря этому Altair 8800 продавался по цене около $500. Это во многом способствовало популярности Altair 8800, хотя был и еще один важный фактор — архитектура открытой шины S-100, что позволяло всем желающим разрабатывать свои внешние устройства для этого компьютера.

В 1976 г. появились компьютеры Apple I компании Apple Computer, в 1977 г. — Apple II, TRS-80 компании Tandy, Commodore PET компании Commodore International.

Отечественная персональная ЭВМ этого класса — Электроника НЦ-8010 [12]. Она была разработана на основе микросхемы К1801ВЕ1 в НИИТТ (НИИ точной технологии, в настоящее время компания «Ангстрем», г. Зеленоград) и выпущена в мае 1981 г.

Широкое распространение получили отечественные персональные ЭВМ семейства ДВК (диалоговый вычислительный комплекс) [14]. Они использовались в различных областях, в том числе в системах управления технологическими процессами в промышленности. ПЭВМ семейства ДВК были также разработаны в НИИТТ. Было выпущено несколько серий ПЭВМ этого семейства. ДВК-1 (Электроника НМС 01100.1 или Электроника МС 0501) была разработана в 1981 г. Впоследствии было разработано и выпущено еще несколько серий — ДВК-2, ДВК-3 и ДВК-4. Были созданы эскизные модели ДВК-5, прообразы современных ноутбуков, но в серию они не пошли из-за изменившейся экономической ситуации. В каждой из серий ДВК было

множество модификаций. Архитектурно все они соответствовали популярным мини-ЭВМ PDP-11 компании DEC (Digital Equipment Corporation), но использовали более совершенную элементную базу, микропроцессоры К1801ВМ1, К1801ВМ2, К1801ВМ3, дополнительные микросхемы этого и некоторых других семейств и поэтому имели существенно меньшие размеры, что позволяло их относить к классу микро-ЭВМ. Еще одним их важным свойством была защищенная память. ДВК выпускались зеленоградским заводом «Квант», построенном специально для их производства. За примерно десятилетний период было выпущено около 230 тыс. различных моделей ДВК. В нашей стране в качестве ПЭВМ ДВК со временем были вытеснены IBM-совместимыми персональными компьютерами (ПК), но в качестве встраиваемых компьютеров в составе различных систем управления они работали еще много лет.

IBM-совместимые ПК — это, конечно, самый распространенный тип ПЭВМ. Многие даже считают его первым и единственным. Первая модель IBM PC (фирменный номер модели — IBM 5150) появилась в августе 1981 г. Команду разработчиков возглавлял Дон Эстридж (Don Estrige). За короткое время IBM PC приобрели огромную популярность. Она так же, как и у Altair, была обусловлена открытой архитектурой, что позволяло сторонним компаниям осуществлять ремонт, обслуживание, а также производство периферийных устройств для IBM PC.

Обычно для своих компьютеров IBM разрабатывала свою элементную базу, но здесь разработчики поступили иначе: для создания IBM 5150 были использованы микросхемы компании Intel:

− микропроцессор i8088;
− контроллер прямого доступа к памяти i8237;
− контроллер прерываний i8259A;
− таймер i8254;
− контроллер параллельного порта i8255A.

Это стало еще одним фактором популярности IBM PC. Intel с момента своего создания в 1968 г. была лидером микроэлектронной индустрии. Один из основателей Intel Роберт Нойс (Robert Norton Noyce) является изобретателем интегральной схемы наряду с Джеком Килби (Jack St. Clair Kilby). Микросхемы памяти впервые были разработаны и выпущены Intel. Первый микропроцессор, Intel 4004, также был изобретен в Intel.

Основные характеристики IBM 5150 были таковы:

– ОЗУ от 16 до 256 Кбайт;

– системная шина ISA (Industry Standard Architecture);

– внешняя память — пятидюймовые дисководы на гибких магнитных дисках, жесткий диск отсутствовал.

Материнская плата модели 5150 имела пять разъемов (слотов) системной шины ISA для подключения различных плат расширения: видеоадаптера, контроллера гибких дисков и др.

Важным новшеством была микросхема ПЗУ с BIOS (Basic Input-Output System, базовая система ввода-вывода). BIOS — это набор подпрограмм, реализующих низкоуровневые функции для взаимодействия с аппаратным обеспечением компьютера, создающих необходимую программную среду для операционной системы и других программ.

В компьютере использовалась операционная система MS-DOS тогда еще небольшой малоизвестной компании Microsoft.

Стоимость в зависимости от конфигурации составляла от $1 500 до $3 000.

Семейство IBM PC стало быстро развиваться. Вскоре появилось следующее поколение — IBM PC/XT (eXtended Technology) — с жестким диском, а затем IBM PC/AT (Advanced Technology) — на основе процессора i80286 с защищенным режимом работы.

Множество других компаний стало производить подобные персональные компьютеры. Они стали называться IBM-совместимыми. Производились они и в нашей стране. В составе серии широко распространенных отечественных компьютеров ЕС ЭВМ (единой системы ЭВМ) были персональные компьютеры ЕС ПЭВМ, которые являлись аналогами персональных компьютеров семейства IBM PC [11].

1.3. ОСНОВНЫЕ СТРУКТУРНЫЕ ПРИНЦИПЫ УСТРОЙСТВА ЭВМ

Архитектура фон Неймана

В основе практически всех современных вычислительных систем, в том числе суперкомпьютеров, лежит архитектура фон Неймана.

Одним из первоисточников (неопубликованным), в котором системно были изложены основные принципы устройства и функционирования ЭВМ, был отчет, составленный Джоном фон Нейманом об итогах работы группы по главе с Джоном Мочли и Преспером Еккертом над компьютером EDVAC, «First Draft of a Report on the EDVAC» (Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945). Сформулированные в нем основные положения и называются архитектурой фон Неймана.

Основные принципы архитектуры фон Неймана (в скобках приведены оригинальные названия частей вычислительной системы из работы «First Draft of a Report on the EDVAC»):

1. Машина состоит из пяти основных блоков (рис. 1.12): арифметико-логического устройства (CA, central arithmetical part), устройства управления (CC, central control part), запоминающего устройства (M, memory), устройства ввода (I, input) и устройства вывода (O, output). Первые два устройства вместе составляют центральный блок (C, Central), в настоящее время называемый процессором (CPU, Central Processing Unit).

2. Запоминающее устройство содержит команды и данные (концепция хранимой программы). Данные могут включать в себя переменные — поименованные области памяти, к которым можно обращаться по имени (которое для процессора представляет собой адрес) для чтения или записи (изменения их значения).

3. Устройство управления считывает команды из памяти в соответствии со значением адреса в специальном регистре — счетчике

команд, декодирует их и организует выполнение, формируя необходимые управляющие сигналы. Для выполнения арифметических и логических операций используется арифметико-логическое устройство.

4. Устройства ввода и вывода используются для ввода информации, в том числе программы, в систему (в запоминающее устройство) и, соответственно, вывода ее из системы.

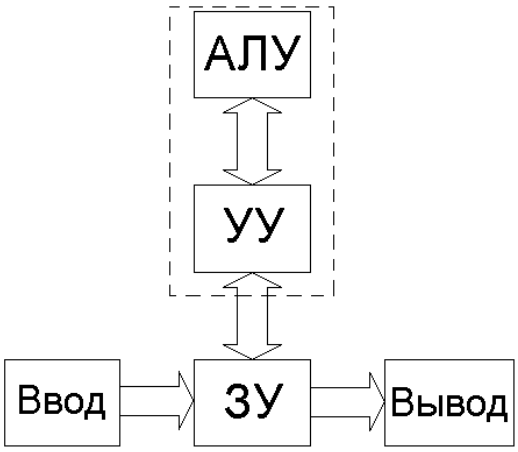


Рис. 1.12. Схема ЭВМ архитектуры фон Неймана

Команды также называют операциями или инструкциями. В этом контексте все эти термины являются синонимами.

Помимо пяти основных блоков фон Нейман упоминает еще одну составляющую компьютера — носитель информации (R), т. е. внешнюю память, с которого информация вводится в основную память компьютера с помощью устройства ввода и на который выводится информация из основной памяти компьютера с помощью устройства вывода.

Главными чертами архитектуры фон Неймана являются разделение в компьютере обрабатывающего и запоминающего устройств и, соответственно, концепция хранимой программы. Существуют типы вычислительных устройств, в которых это не так. В мозге человека, например, и в нейрокомпьютере запоминающие и обрабатывающие элементы неотделимы друг от друга. Таким образом, в фоннеймановском компьютере необходим информационный канал, свя-

зывающий запоминающее устройство с обрабатывающим, причем этот канал один и для команд, и для данных. Пропускная способность этого канала во многом ограничивает производительность компьютера и называется «бутылочным горлышком» (bottleneck) архитектуры фон Неймана. При этом, с одной стороны, команды и данные хранятся вместе, а с другой — запоминающее устройство не является единым и монолитным, а наоборот, должно состоять из различных частей, каждая из которых предназначена для хранения определенного типа информации — неизменяющейся (постоянная память), изменяющейся (оперативная память), быстро изменяющейся (регистры процессора).

Устройства ввода и вывода компьютера фон-неймановской архитектуры можно объединить в одно устройство ввода-вывода (УВВ). Устройства ввода-вывода называются также внешними или периферийными устройствами. Арифметико-логическое устройство в совокупности с устройством управления образует процессор. Таким образом, любая ЭВМ содержит всего три типа основных устройств — процессор, память и устройства ввода-вывода (рис. 1.13).

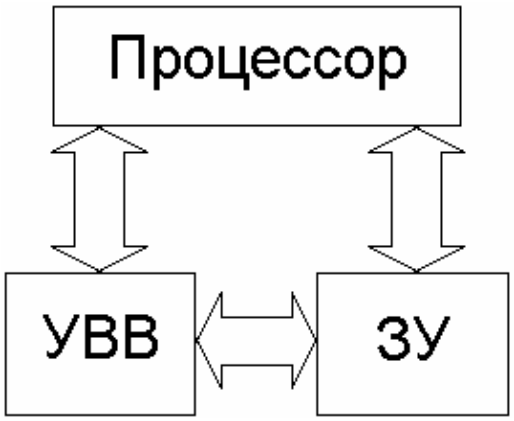


Рис. 1.13. Треугольник фон Неймана

В архитектуре фон Неймана информация с устройства ввода поступает непосредственно в память и выдается из памяти на устройство вывода, минуя процессор. В современных компьютерах это называется прямым доступом к памяти (DMA, Direct Memory Access).

Однако во многих случаях все устройства взаимодействуют, обмениваясь информацией через процессор. Это называется программируемым вводом-выводом (PIO, Programming Input-Output).

Выделенные три типа элементов существуют в компьютере не только на верхнем уровне, но и на более низких. Этих уровней может быть множество, что в целом создает иерархическую структуру компьютера (рис. 1.14).

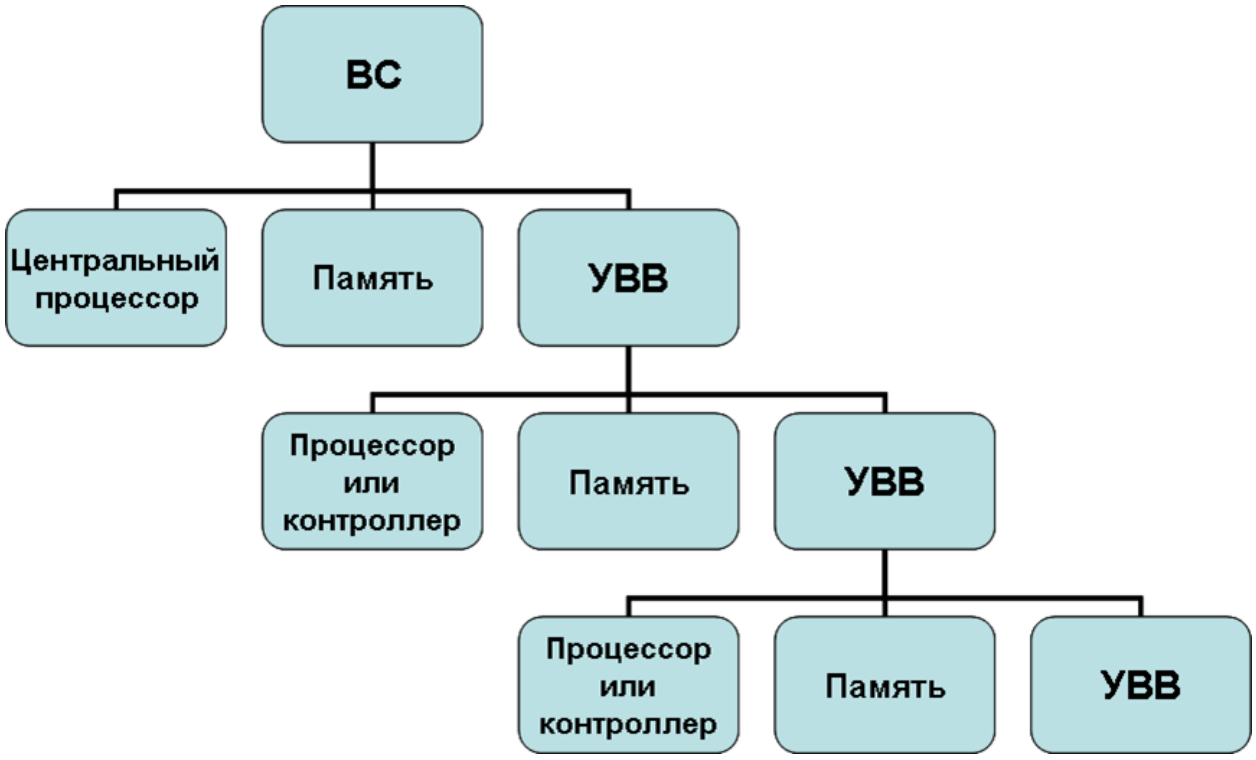


Рис. 1.14. Иерархическая структура ЭВМ

Причем эта структура многослойная. На рис. 1.14 изображен как иллюстрация иерархической структуры только один из множества слоев, относящихся к одному из устройств ввода-вывода. Это можно пояснить на примере такого устройства ввода-вывода, как клавиатура. На верхнем уровне иерархии во всех случаях находятся центральный процессор и основная память, а устройством ввода-вывода этого уровня является в целом подсистема клавиатуры, которая детализируется на следующем, среднем уровне. Ее центральным элементом является контроллер клавиатуры, который ранее существовал как от-

дельная микросхема, а сейчас вместе с другими контроллерами входит в состав более крупных микросхем. Память этого уровня — регистры того же контроллера, например, порт 60h, в котором оказываются скан-коды нажимаемых клавиш. Устройство ввода-вывода этого уровня — собственно клавиатура. Она детализируется на нижнем уровне. Здесь процессор — это микро-ЭВМ клавиатуры, память — запоминающие устройства той же микро-ЭВМ, а устройство ввода-вывода — наборное поле, матрица клавиш, собственно кнопочки — оконечное устройство ввода.

Такая же иерархия существует и для других подсистем ввода-вывода, например, видеосистемы. Таким образом, в этой иерархической структуре существует множество не только уровней, но и слоев.

Интересно отметить, что фон Нейман в своем черновике проводил аналогию между устройством компьютера и человеческого организма: “The three specific parts CA, CC (together C) and M correspond to the associative neurons in the human nervous system. It remains to discuss the equivalents of the sensory or afferent and the motor or efferent neurons. These are the input and the output organs of the device. (Три специфические части CA, CC (вместе C) и M соответствуют ассоциативным нейронам в нервной системе человека. Остается обсудить эквиваленты сенсорных, или афферентных, и моторных, или эфферентных, нейронов. Это входные и выходные органы устройства)”.

Дисциплину «Архитектура компьютерных систем» в блоке компьютерных специальностей и направлений можно сравнить с «Анатомией и физиологией» для медицинских специальностей. Не все медики будут хирургами, но каждый из них должен знать и понимать устройство и функционирование человеческого организма. Точно так же каждый специалист в области информационных технологий должен знать устройство компьютера и понимать, что происходит внутри него при работе программ, функционировании информационных и автоматизированных систем.

Процессор ЭВМ как универсальный алгоритм

В качестве математической основы процессора можно привести следующую теорему [15]: «Существует такой нормальный алгоритм U, называемый универсальным нормальным алгоритмом, который для любого нормального алгоритма A и любого входного слова P из области определения этого последнего алгоритма переводит слово AP, полученное приписыванием слова P к алгоритму A, в слово, являющееся соответствующим выходным словом, в которое алгоритм A перерабатывает слово P»:

$$U(AP) = A(P).$$

Автором понятия нормального алгоритма является Андрей Андреевич Марков (младший), на книгу которого ссылается Глушков [16]. Там же приведена и указанная теорема, но в формулировке использованы менее привычные обозначения, принятые в математической среде.

Теорема утверждает возможность создания устройства, которое может выполнить любое заданное преобразование информации. Таким устройством и является процессор. Процессор реализует алгоритм (U) выполнения алгоритма (A). Как следует из теоремы, процессор должен получать информацию двух типов — команды (алгоритм A) и данные или операнды (входное слово P алгоритма A). Совокупность команд и данных и представляет собой программу.

Необходимую универсальность процессора обеспечивают следующие типы команд:

1. Передачи данных. В любой программе требуется передавать данные, например, из основной памяти в регистр процессора или обратно.

2. Передачи управления (перехода, ветвления). Только при решении простых задач команды программы могут выполняться одна за другой так, как они расположены в тексте программы (в памяти). Для

решения большинства задач линейной структуры программы недостаточно, требуется осуществлять переходы в заданную точку с возвратом или без, в зависимости от выполнения некоторых условий выполнять или не выполнять тот или иной фрагмент программы, идти по одной из нескольких возможных ветвей, выполнять определенный фрагмент программы несколько раз.

3. Арифметических и логических операций. Это основной тип команд, с помощью которых и осуществляются вычисления. Они выполняются АЛУ, арифметико-логическим устройством процессора.

4. Машинного управления (управления процессором). Первые три типа команд относятся непосредственно к алгоритму A, они содержат информацию о задаче. Команды четвертого типа определяют условия выполнения задачи, например, должны быть разрешены прерывания или нет. Таким образом, они относятся уже в большей степени к алгоритму U, хотя и содержатся технически в алгоритме A. Можно сказать, что с их помощью алгоритм A влияет на алгоритм U.

В каждом конкретном процессоре разбиение команд на группы и подгруппы может варьироваться, но большинство команд относится к одному из перечисленных типов. Некоторые сложные команды относятся к двум типам, например, CMPXCHG (сравнение и обмен) — к арифметическим и логическим командам и командам передачи данных. Но никаких других типов не существует.

В работе «First Draft of a Report on the EDVAC» фон Нейман всего один раз использует слово «регистр». Однако он указывает, что память может быть сложным устройством, состоящим из различных частей, каждая из которых выполняет свою задачу. В частности, фон Нейман говорит, что арифметико-логическое устройство должно хранить результаты промежуточных вычислений в отдельных запоминающих элементах. Таким образом, часть памяти оказывается в процессоре в виде регистров. Регистры имеются также и в устройствах ввода-вывода компьютера. Они могут быть различных размеров: 8, 16, 32 бита и т. д.

В соответствии с рассмотренным математическим обоснованием в каждый момент времени процессор должен знать месторасположение следующей команды и операндов, как входных (где взять исходные данные), так и выходных (куда поместить результат операции).

Для этого в процессоре имеются следующие основные регистры:

– *счетчик команд* (счетчик программы (program counter, PC) или указатель команд (instruction pointer, IP)) — содержит адрес очередного байта (или слова) команды, который должен быть считан из памяти команд;

– *регистр команд* — содержит код операции команды, введенный из памяти по адресу в счетчике команд;

– *аккумулятор* (accumulator, A, накопительный регистр или регистр результата) — содержит входной операнд или один из входных операндов команды и результат ее выполнения.

Поскольку на протяжении определенных участков программы команды выполняются последовательно, одна за другой так, как они расположены в тексте программы (в памяти), то счетчик команд инкрементируется при чтении каждого байта (слова) команды:

$$IP \leftarrow IP + 1.$$

Если же нужно нарушить последовательность выполнения команд, то используется явное задание адреса:

$$IP \leftarrow \text{новое значение}$$

или

$$IP \leftarrow IP + \text{смещение}.$$

Первый вариант, когда в IP записывается новое значение, называется абсолютной адресацией, а второй, когда к текущему значению IP прибавляется некоторое смещение, положительное или отрицательное, называется относительной адресацией.

Помимо указанных регистров процессор содержит множество других, которые могут иметь свою специализацию. Например, регистр флагов соедржит признаки результата арифметических и логических команд: знак, наличие переноса, переполнения и т. д.

Устройство управления ЭВМ реализует непрерывный двухшаговый цикл «выборка-исполнение» (рис. 1.15).

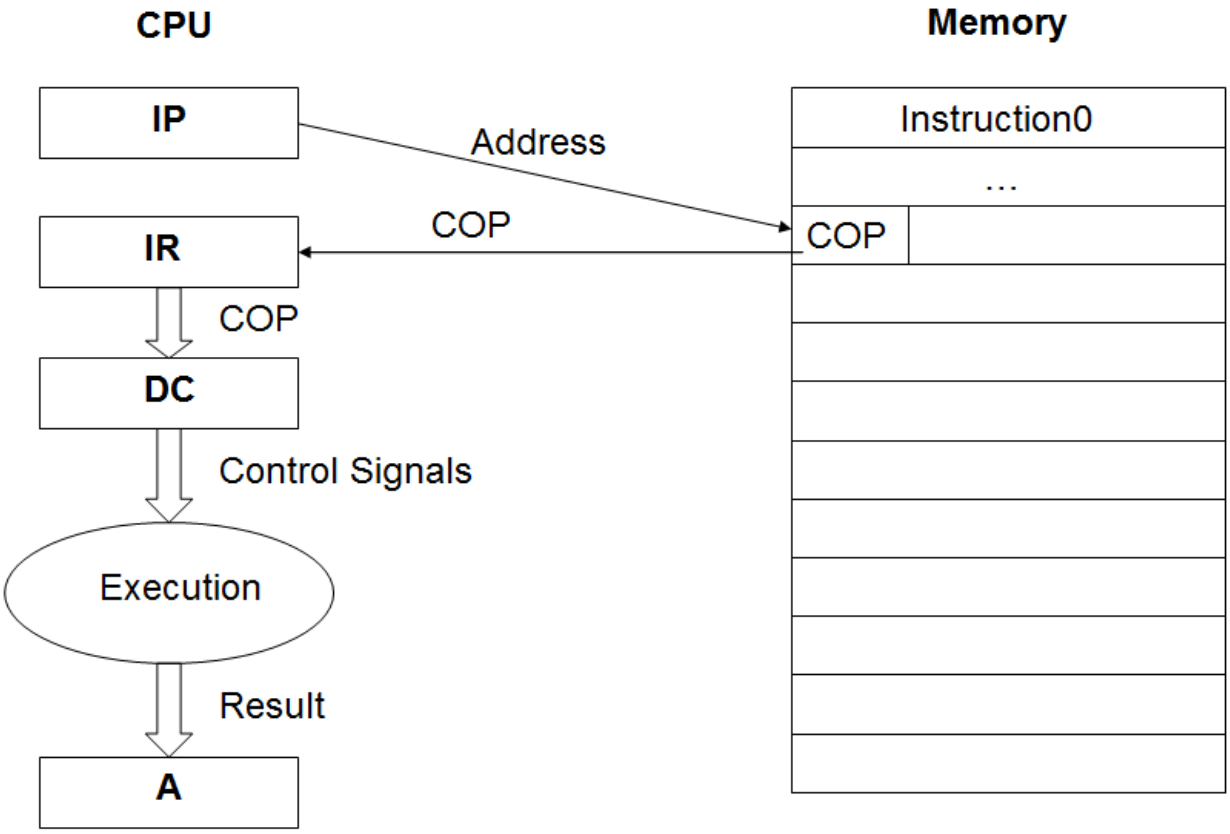


Рис. 1.15. Выборка и исполнение команд:
IP — Instruction Pointer, указатель команд;
IR — Instruction Register, регистр команд;
DC — Decoder, дешифратор команд;
A — Accumulator, аккумулятор, основной регистр данных;
COP — Code of Operation, код операции команды

На шаге выборки осуществляется извлечение команды из памяти по адресу, содержащемуся в программном счетчике, и помещение ее кода операции в регистр команд. На шаге исполнения — дешифрация и собственно исполнение. В зависимости от команды на этапе

исполнения процессор извлекает из своих регистров, памяти или портов ввода-вывода операнды, выполняет требуемую операцию и сохраняет результаты. Можно говорить и о трехшаговом цикле «выборка-дешифрация-исполнение». Это зависит от того, с какой степенью детализации рассматривается процесс выполнения команды. Можно ввести и большее количество шагов. Как будет видно в дальнейшем, они впрямую связаны со стадиями (этапами) конвейера в конвейерных процессорах (см. п. 3.2, 3.4). Цикл от начала чтения команды до завершения ее выполнения — это цикл команды. Он состоит из нескольких машинных циклов (циклов шины или транзакций) — промежутков времени, за которые происходит один обмен данными по шине между процессором и памятью или устройством ввода-вывода. Машинный цикл состоит из тактов.

Программирование ЭВМ

В процессе разработки и изготовления ЭВМ создается ее аппаратная структура, набор технических средств, на которых затем будет выполняться программа. Поэтому программный аспект также прорабатывается в этом процессе, и работающая ЭВМ — это совокупность аппаратного и программного обеспечений. В составе программного обеспечения выделяется системное и прикладное программное обеспечение. Прикладное программное обеспечение — это программы, предназначенные для выполнения задач пользователя в данной предметной области — численного интегрирования дифференциальных уравнений, решения линейных алгебраических уравнений и т. п. Системное программное обеспечение — все остальные программы. Они обеспечивают создание и выполнение прикладных программ. К системному программному обеспечению относится BIOS, операционная система, СУБД (система управления базами данных), различные так называемые утилиты: проверки и обслуживания дисков, получения информации о конфигурации компьютера и т. п. Основное назначение операционной системы — запуск других программ на исполне-

ние. Из этой главной ее функции следуют все остальные: распределение памяти, реализация интерфейса с пользователем, организация хранения информации во внешней памяти, обеспечение взаимодействия программ с оборудованием, реакции на различные события, в том числе действия пользователя и т. д. Важным типом системных программ являются инструментальные системы программирования — средства создания других программ, как прикладных, так и системных.

Выделяются ***два уровня программирования***:

1. Низкий — уровень команд процессора.
2. Высокий — уровень алгоритмических действий решаемой задачи.

На низком уровне можно программировать в машинных кодах (на машинном языке, языке процессора) или на языке ассемблера. На высоком уровне используется язык программирования высокого уровня (ЯВУ). Существует множество языков ассемблера и огромное множество языков высокого уровня. Одному оператору языка высокого уровня может соответствовать множество команд процессора. Одной команде ассемблера соответствует ровно одна команда процессора. Поэтому у каждой модели (семейства или архитектуры) процессора свой язык ассемблера. Есть ассемблеры x86, ARM, AVR, RISC-V и др.

Как уже было отмечено, и команды, и данные в ЭВМ представляются в цифровой двоичной форме. Для человека более удобной является символьная, точнее буквенная запись, поскольку цифры — это тоже символы. Для облегчения работы человека по программированию ЭВМ и были разработаны языки программирования низкого и высокого уровня.

Для изучения архитектуры компьютерных систем необходим именно язык низкого уровня, язык ассемблера. Он не скрывает деталей программно-аппаратной реализации ЭВМ и в то же время удобен для запоминания команд. Основная часть команды на языке ассемблера и называется мнемоникой.

При программировании на каком-либо языке, отличном от машинного, неважно, высокого или низкого уровня, требуется специальная программа, которая переводит разрабатываемую программу с языка программирования на язык процессора, машинный язык. Существует ***два варианта перевода и исполнения разрабатываемой программы*** (табл. 1.11):

− построчно, когда очередная строка (блок, конструкция, оператор) программы переводится и сразу исполняется;

− целиком, когда переводится вся программа в целом и только после этого исполняется.

Таблица 1.11

Способы и уровни программирования

Способы перевода / **Уровни**	**Целиком**	**Построчно**
Высокий (ЯВУ)	Компилятор с ЯВУ	Интерпретатор
Низкий (Язык ассемблера)	Ассемблер	–

Интерпретатор — это программа, которая выполняет программу, написанную на языке программирования, построчно. Примеры интерпретируемых языков программирования — JavaScript и Python. Имеется множество различных интерпретаторов для этих языков. Интерпретаторы JavaScript встроены во все современные браузеры.

Компилятор — это программа, которая преобразует программу, написанную на языке программирования, на язык процессора (в машинный код) целиком. Примеры компилируемых языков компилирования — C, C++, Rust. Файл с программой на языке программирования называется исходным, а с программой на языке процессора — объектным.

Компилятор и интерпретатор являются видами транслятора. **Транслятор** — это программа, которая переводит программу с одного языка программирования на другой.

Ассемблер — это компилятор, который переводит программу с языка ассемблера на язык процессора. Часто ассемблером называют и сам язык ассемблера. То есть можно говорить: «Программировать на языке ассемблера» или «Программировать на ассемблере».

Интерпретация и компиляция могут сочетаться. Для этого вводится промежуточный уровень между языком программирования и языком процессора, который можно назвать языком виртуальной машины или среды исполнения. Программа с языка высокого уровня компилируется в промежуточный код, который затем исполняется виртуальной машиной в режиме интерпретации. Так происходит в среде .NET или JAVA. Есть интерпретаторы Python, которые компилируют программу в байт-код (bytecode), который затем выполняется виртуальной машиной Python (Python Virtual Machine, PVM).

Помимо компилятора для создания и запуска программы требуется еще множество программ, основными из которых являются:

- текстовый редактор (editor);
- компоновщик (редактор связей, linker);
- отладчик (debugger);
- загрузчик (loader).

С помощью текстового редактора осуществляется ввод и редактирование программы на языке программирования.

Компоновщик необходим, потому что разрабатываемая программа создается для запуска в определенной операционной системе, может состоять из нескольких файлов и использовать код, созданный ранее, традиционно располагающийся в так называемых библиотеках. Поэтому после компиляции программу необходимо скомпоновать. Компоновщик объединяет все фрагменты объектного кода в один исполняемый файл и добавляет информацию, необходимую для запуска программы в данной операционной системе.

Отладчик позволяет детально исследовать разрабатываемую программу, прогнав ее покомандно или с заданными точками останова, наблюдая за состоянием компьютера — значением регистров процессора, ячеек памяти, портов ввода-вывода.

Для выполнения программу нужно поместить в определенную область памяти и передать ей управление. Это делает еще одна программа — загрузчик. Загрузчик входит в состав операционной системы. Для загрузки самой операционной системы используются два загрузчика: начальный, входящий в состав BIOS (Basic Input-Output System), базовой системы ввода-вывода, расположенной в ПЗУ компьютера, и расположенный на диске в загрузочном секторе. В операционных системах Windows на основе NT (New Technology) загрузочный код содержится также в файле NTLDR (NT Loader).

Современные инструментальные системы программирования предоставляют так называемую интегрированную среду разработки (IDE, Integrated Development Environment). В таких средах все функции отдельных инструментов программирования доступны из одного главного окна приложения. Это удобно для разработки программ, но не для изучения данной дисциплины. Поэтому необходимо рассмотреть все инструменты по отдельности, чтобы затем использовать интегрированные среды со знанием дела. Такую возможность нам предоставит инструментальный пакет Turbo Assembler (TASM). Существует множество других ассемблеров x86: FASM, MASM, WASM — но для учебных целей лучше всего подходит TASM.

Язык ассемблера

Как уже было сказано, язык ассемблера — это язык низкого уровня. Каждой команде на языке ассемблера соответствует одна машинная команда и наоборот, т. е. между записью команд процессора на машинном языке и языке ассемблера существует взаимнооднозначное соответствие. Ассемблер преобразует команды, записанные на языке ассемблера, в команды на машинном языке, т. е. в числа, понятные процессору. Помимо команд в языке ассемблера существуют и элементы другого типа — директивы. Директивы в языке ассемблера — это команды самому ассемблеру, необходимые для компиляции

программы, например, выделения определенных областей памяти данных под переменные и заполнения их определенными значениями. В любом деле фигурирует информация различных уровней. Есть информация непосредственно о задаче, а есть информация об информации — метаинформация. Это, например, различные пояснения, указания и т. п. Здесь есть аналогия с выделением типов команд. Команды передачи данных, передачи управления и арифметическо-логические команды несут информацию о задаче, а команды машинного управления — об условиях ее выполнения, т. е. тоже представляют собой метаинформацию. Таким образом, текст программы на языке ассемблера содержит два типа предложений: команды и директивы.

Общая форма записи команд на языке ассемблера x86 такова:

label: **mnemonic** argument1, argument2, argument3

Команды можно записывать большими или маленькими буквами.

В любой команде должна присутствовать информация двух типов: основная — информация о действии, которое должен выполнить процессор по данной команде, и дополнительная — информация, необходимая для выполнения данного действия.

Мнемоника — это основная и обязательная часть команды. Она несет информацию первого типа — о действии. Можно сказать, что мнемоника — это имя класса кодов операции, которым соответствует определенное действие, например, передача данных из регистра в регистр, сложение, вычитание и т. п. Мнемоника представляет собой либо слово полностью, либо его сокращение, либо аббревиатуру, например:

ADD — сложить;

SUB — subtract, вычесть;

CMP — compare, сравнить;

MOV — move, переместить, передвинуть;

IN — input, ввести, ввод;

OUT — output, вывести, вывод;

JMP — jump, прыгнуть, прыжок;

CALL — вызвать (подпрограмму), вызов;

RET — return, возврат из подпрограммы;

INT — interrupt, прервать, прерывание;

IRET — interrupt return, возврат из обработчика прерывания;

LOOP — цикл, сделать петлю, виток;

STI — stand interrupt, разрешить прерывания.

Чтобы запомнить мнемонику, нужно обязательно знать ее расшифровку.

Остальные части команды присутствуют опционально. Метка требуется для передачи управления на данную команду из другого места программы. В программе на машинном языке она соответствует абсолютному или относительному адресу данной команды в памяти.

Аргументы — это информация второго типа — операнды для выполнения действия или их адреса. Если аргументов несколько, то приемником является первый аргумент. Аргументы могут представлять собой имена регистров, переменных, числа и допустимые выражения, состоящие из всех этих элементов и математических операций. Надо иметь в виду, что аргументы в ассемблерной записи команды могут отсутствовать, но операнды в ней при этом имеются. В этом случае они подразумеваются по умолчанию логикой выполняемого действия.

Подробно система команд процессора будет описана в п. 3.3. Но начинать программировать на ассемблере нужно уже сейчас. Для этого будет достаточно приводимых примеров. Если какие-то детали покажутся неясными, ничего страшного. Одно из основных качеств исследователя — терпимость к неопределенности.

Рассмотрим в качестве примера фрагмент программы сложения двух чисел (табл. 1.12).

Таблица 1.12

Фрагмент программы сложения двух чисел

№	Адрес (IP)	Команда на машинном языке	Команда на языке ассемблера	Регистр команд	Аккумулятор ax
1	0000	B8 02 00	mov ax, 2	B8	0002
2	0003	05 04 00	add ax, 4	05	0006

Напомним, что внутри компьютера содержатся только двоичные числа, но машинный код, программа на языке процессора для сокращения записи записывается в виде шестнадцатеричных чисел.

В этих командах по два аргумента: первый — регистр, второй — число, находящееся в самой команде (непосредственный операнд).

В начале выполнения этого фрагмента значение IP равно 0000. По этому адресу процессор читает из памяти код операции первой команды B8 и помещает его в регистр команд. IP инкрементируется. Процессор декодирует код операции и определяет, что это команда передачи непосредственных данных размером в два байта в аккумулятор. Таким образом, далее так же по значению IP из памяти читаются второй и третий байты команды и помещаются в аккумулятор. В большинстве компьютеров принят так называемый обратный порядок хранения слов (little endian order). Если число занимает в памяти больше одного байта, то сначала, по младшему адресу, располагается младший байт, а затем последующие байты. Поэтому в машинном коде первой команды второй и третий байты не 00 02, а 02 00. Надо отметить, что если память команд адресуется побайтно, как это имеет место в архитектуре x86, то IP инкрементируется после чтения каждого байта команды. Отметим также, что адресация (способ нахождения операнда), при которой операнд находится в самой команде, называется непосредственной, а когда в регистре — регистровой. То есть, в каждой их этих двух командах присутствует регистровая адресация и непосредственная.

Перед выполнением второй команды IP равен 0003. По этому адресу процессор читает из памяти код операции второй команды 05 и также помещает его в регистр команд. По результатам дешифрации кода операции из памяти читаются второй и третий байты команды и прибавляются к значению аккумулятора, который в итоге становится равным 0006.

Теперь для полноты картины коснемся языка высокого уровня и рассмотрим соответствие между языком высокого уровня, языком ассемблера и языком процессора. Проведем эксперимент. Напишем простую программу сложения двух целых чисел на языке С:

```
int A = 1, B = 2, S;
void main()
{
	S = A + B;
}
```

Скомпилируем ее и дизассемблируем. Это можно сделать в любой среде программирования на языке С. Дизассемблирование — процесс, обратный ассемблированию, получение кода на языке ассемблера по машинному коду.

Мы увидим, что телу главной функции, которое состоит всего из одной строки с оператором присваивания и операцией сложения, будет соответствовать фрагмент кода на языке ассемблера и машинном языке, состоящий из трех строк, трех команд (табл. 1.13).

Таблица 1.13

Соответствие между фрагментами кода на различных языках

№	ЯВУ С	Язык ассемблера	Язык процессора
1	S = A + B	mov AX, A	A1 AA 00
2		add AX, B	03 06 AC 00
3		mov S, AX	A3 90 02

Видно, что язык высокого уровня скрывает важнейшие детали устройства и работы компьютера.

Оператору присваивания переменной S результата сложения значений переменных A и B соответствуют три команды ассемблера и процессора. В командах так же, как и в предыдущем примере, по два аргумента. Но теперь один из них не число, в отличие от предыдущего примера, а имя переменной, которое на машинном языке, языке процессора, представляет собой ее адрес в памяти. То есть в команде находится не операнд, а его адрес, а сам операнд находится в памяти по данному адресу.

По команде mov AX, A процессор передает значение переменной A в аккумулятор. AX становится равным единице. Отметим, что размер целочисленных переменных в языке C — 16 бит. Таков же и размер регистра AX. На машинном языке команда mov AX, A представляется как трехбайтное число. Первый байт A1 — это код операции команды, который содержит информацию о действии (передать данные из памяти данных) и об операнде-приемнике, т. е. куда передать данные (регистр AX, аккумулятор). Код операции будет помещен процессором в регистр команд для дешифрации и выработки необходимых управляющих сигналов для выполнения команды. Второй и третий байты — это адрес переменной A. Здесь он тоже 16-битный. Отметим, что адресация, при которой в команде располагается адрес операнда, называется прямой. Есть и другие способы адресации — косвенная (адрес операнда находится в регистре) и неявная (операнд в регистре, но имя регистра в команде отсутствует, потому что принято по умолчанию для данной команды).

Почему адреса переменных оказались именно такими? Все три переменные, A, B и S, объявлены как глобальные. Они существуют на протяжении всего времени работы программы и доступны из любых ее частей. Такие переменные располагаются в памяти в отдельной области (сегменте) данных. Указанные адреса отсчитываются от начала этого сегмента и называются смещениями. Компилятор рас-

положил эти переменные в сегменте данных так, что их смещения приняли именно такие конкретные значения.

Вторая команда прибавляет к аккумулятору значение переменной B и результат помещает в аккумулятор. Таким образом, в аккумуляторе оказывается сумма переменных A и B. Длина команды сложения — четыре байта. Первые два байта — 03 06 — код операции с уточнением способа адресации. Следующие два байта — адрес переменной B, 00AC. 00AC – 00AA = 2 — таким образом, переменные A и B располагаются в памяти одна за другой.

Третья команда передает результат сложения из аккумулятора в переменную S. Ее адрес — 0290. Компилятор расположил ее на некотором расстоянии от переменных A и B.

Полный текст программы на языке ассемблера, выполняющей сложение двух переменных и помещающей результат сложения в третью переменную, мог бы выглядеть так:

```
;sum.asm — программа, вычисляющая S = A + B
data   segment              ;объявление начала сегмента данных
       a dw 01h             ;объявление и инициализация переменной A
       b dw 02h             ;объявление и инициализация переменной B
       s dw ?               ;объявление переменной S
data   ends                 ;объявление конца сегмента данных
code   segment              ;объявление начала сегмента команд (кода)
       assume ds:data       ;регистр DS будет указывать на сегмент data
begin: mov dx, data         ;точка входа в программу, DX = data
       mov ds, dx           ;теперь DS указывает на сегмент data, DS = data
       mov ax, a            ;AX = A = 1
       add ax, b            ;AX = AX + B = A + B = 1 + 2 = 3
       mov s, ax            ;S = AX = 3
       mov ax, 4c00h        ;функция завершения программы MS-DOS с
       int 21h              ;кодом возврата 0 прерывания 21h
code   ends                 ;объявление конца сегмента кода
       end    begin         ;конец программы
```

Более подробно понятие сегмента мы рассмотрим в дальнейшем, а пока отметим, что в различных процессорах используются различные средства структурирования памяти и, соответственно, программы. Сегмент — это одно из таких средств. Программа включает два сегмента — сегмент данных с именем data и сегмент кода с именем code. Процессор должен находить элементы выполняемой программы в памяти. Мы уже поняли, что команды он находит по значению регистра IP. Но это значение не абсолютный адрес, а относительный, отсчитываемый от начала сегмента, так называемое смещение. А адрес начала сегмента процессор определяет по значению специального регистра — сегментного регистра. Имена и количество сегментов в программе могут быть любыми с определенными ограничениями. Сегментных регистров в процессоре несколько, каждый из них указывает на свой сегмент. На сегмент с командами указывает регистр CS (Code Segment). Его в данном случае инициализирует операционная система. На сегмент с переменными указывает регистр DS (Data Segment). Его нужно инициализировать в программе явным образом.

Если в компьютере используется операционная система, то программа должна удовлетворять ее требованиям, т. е. операционная система накладывает определенные ограничения на формат и структуру программы. Для правильного завершения программы в операционной системе MS-DOS необходимо вызвать функцию 4C системного прерывания 21h, что и делается в двух последних строчках сегмента команд.

Для компиляции программы можно использовать любой ассемблер. Мы будем использовать пакет Turbo Assembler (TASM) компании Borland. Имя файла самого ассемблера, компилятора, — tasm.exe.

Командная строка для компиляции (табл. 1.14):

tasm sum

Если расширение имени файла стандартное — .asm, то его можно не писать. В результате будет создан файл с объектным кодом sum.obj.

Компоновщик пакета TASM находится в файле tlink.exe. Соответственно, командная строка для компоновки:

tlink sum

В результате будет создан исполняемый файл sum.exe. Если его запустить, то никакого видимого эффекта не будет. Программа никак не взаимодействует с внешними устройствами, ничего не вводит и ничего не выводит. Исходные данные читаются процессором из памяти, результат также помещается в память. Исследовать работу программы можно с помощью отладчика, который находится в файле td.exe:

td sum

Таблица 1.14

Компиляция и компоновка программы

№	Командная строка	Преобразование
1	tasm filename	filename.asm -> filename.obj
2	tlink filename	filename.obj -> filename.exe

Расширения имен файлов писать не обязательно, если они стандартные. В этом случае компилятор и компоновщик правильно найдут нужные им файлы.

Более подробно все эти действия описаны в следующем пункте.

Если при компиляции задать параметр l (listing):

tasm /l sum,

то кроме файла с объектным кодом sum.obj будет создан файл с листингом sum.lst:

```
1                                   ;sum.asm — программа, вычисляющая S = A + B
2       0000                        data    segment
3       0000  0001                          a dw 01h
4       0002  0002                          b dw 02h
5       0004  ????                          s dw ?
6       0006                        data    ends
7       0000                        code    segment
8                                           assume ds:data
9       0000  BA 0000s              begin: mov dx, data
10      0003  8E DA                         mov ds, dx
11      0005  A1 0000r                      mov ax, a
12      0008  03 06 0002r                   add ax, b
13      000C  A3 0004r                      mov s, ax
14      000F  B8 4C00                       mov ax, 4c00h
15      0012  CD 21                         int 21h
16      0014                        code    ends
17                                  end     begin
```

Листинг — это таблица, содержащая четыре столбца:

1. Номер строки.
2. Адрес внутри сегмента, т. е. от его начала (смещение).
3. Содержимое памяти по данным адресам (машинный код). В сегменте данных — это значения переменных, в сегменте кода — это машинные коды команд.
4. Ассемблерный код.

Комментарии в листинге также сохраняются, просто здесь они для экономии места опущены.

Слова (двухбайтные значения) в листинге представлены в прямом, а не в обратном порядке.

Из листинга видно, что командам процессора соответствуют только строки 9–15. Остальные строки соответствуют директивам ассемблера.

Директивы бывают двух типов:

− отображаемые в машинный код;

− непосредственно не отображаемые в машинный код.

Для директив первого типа столбец с машинным кодом непустой. Для директив второго типа — пустой.

В рассматриваемой программе имеются следующие директивы:

− segment — директива объявления начала сегмента;

− dw (define word) — директива определения данных (выделения и инициализации области памяти) размером 16 бит;

− ends (end segment) — директива объявления конца сегмента;

− assume — директива назначения сегменту сегментного регистра;

− end — директива указания конца программы.

Есть также другие директивы определения данных:

− db (define byte) — директива определения области памяти в 8 бит;

− dd (define double word) — директива определения области памяти в 32 бита.

В дальнейшем нам понадобятся директивы определения подпрограммы:

− proc (procedure) — начало подпрограммы;

− endp (end procedure) — конец подпрограммы.

Подпрограмма — это фрагмент кода, выполняющий определенную задачу, выполняемый множество раз, вызываемый командой вызова подпрограммы CALL и заканчивающийся командой возврата из подпрограммы RET. Подпрограмма является важным элементом структурирования программ. В языках высокого уровня подпрограммы — это функции и процедуры.

Строки 3–5 содержат директивы определения переменных A, B и S. В совокупности строки 3–5 и 9–15 содержат информацию для процессора — программу, включающую команды и данные. Они непосредственно отображаются на память компьютера, где по указанным адресам располагаются соответствующие значения.

1.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 1

Задание 1. Основные устройства ЭВМ

Основную информацию о составе аппаратного обеспечения компьютера можно получить средствами операционной системы, в том числе утилит, входящих в ее состав, без использования специальных программ анализа конфигурации.

В числе таких средств можно выделить:

− параметры системы: SystemSettings.exe (рис. 1.16);

− диспетчер задач: Taskmgr.exe (рис. 1.17);

− консоль управления: mmc.exe и диспетчер устройств: devmgmt.msc (рис. 1.18);

− сведения о системе: msinfo32.exe (рис. 1.19);

− средство диагностики DirectX: dxdiag.exe (рис. 1.20);

− редактор реестра: regedit.exe: Компьютер\HKEY_LOCAL_MACHINE\HARDWARE (рис. 1.21);

− командная строка: cmd.exe, команда systeminfo (рис. 1.22).

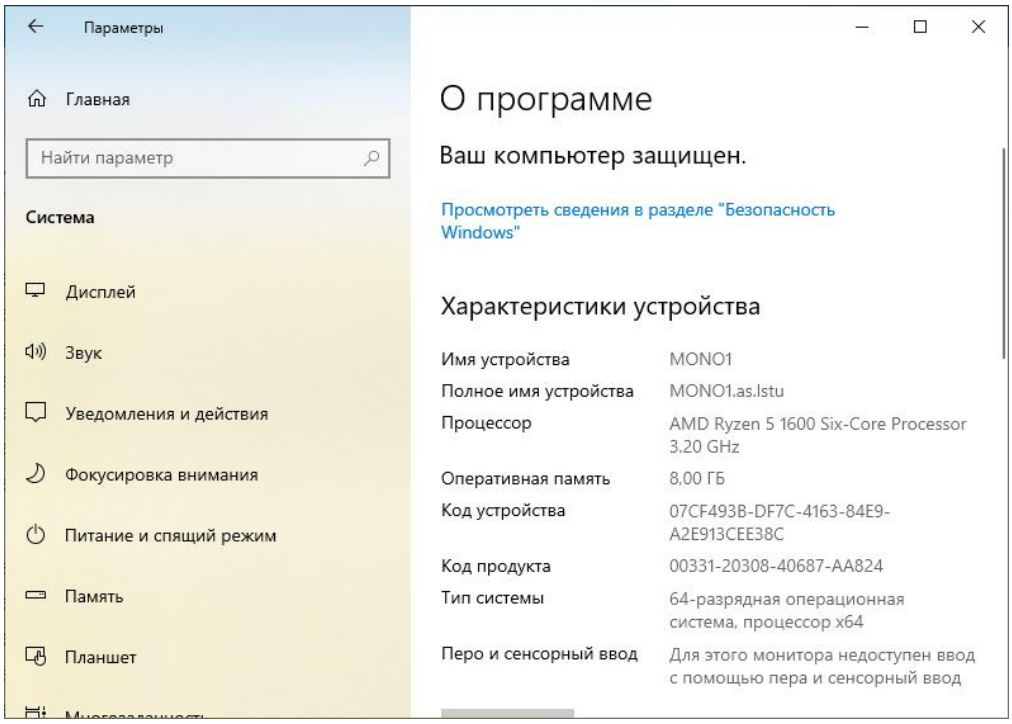


Рис. 1.16. Параметры системы

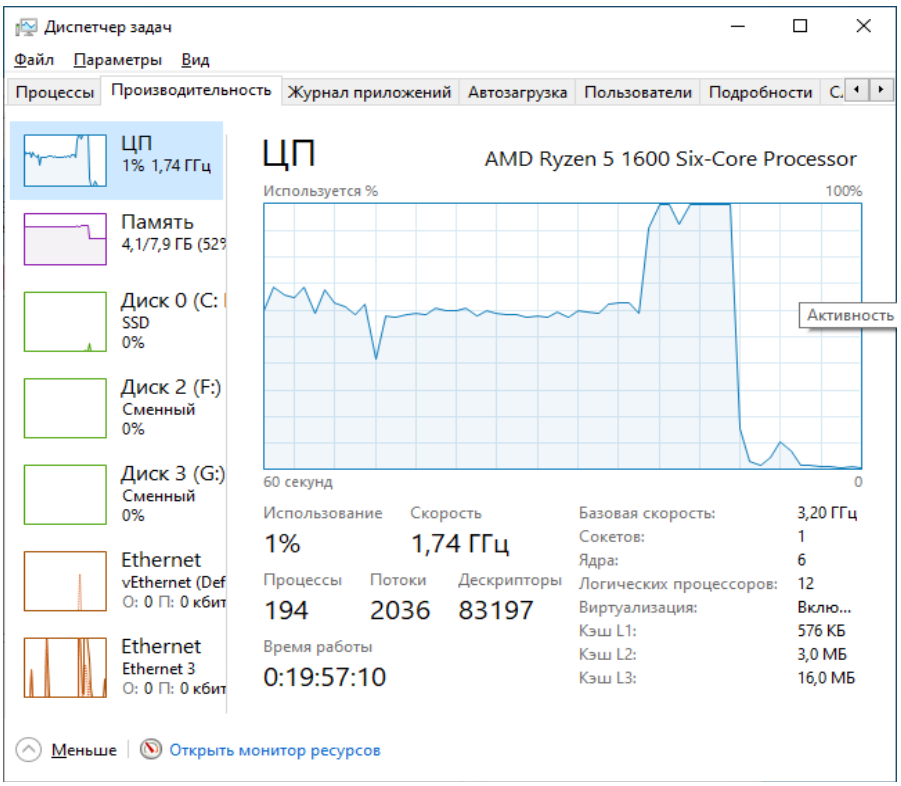


Рис. 1.17. Диспетчер задач

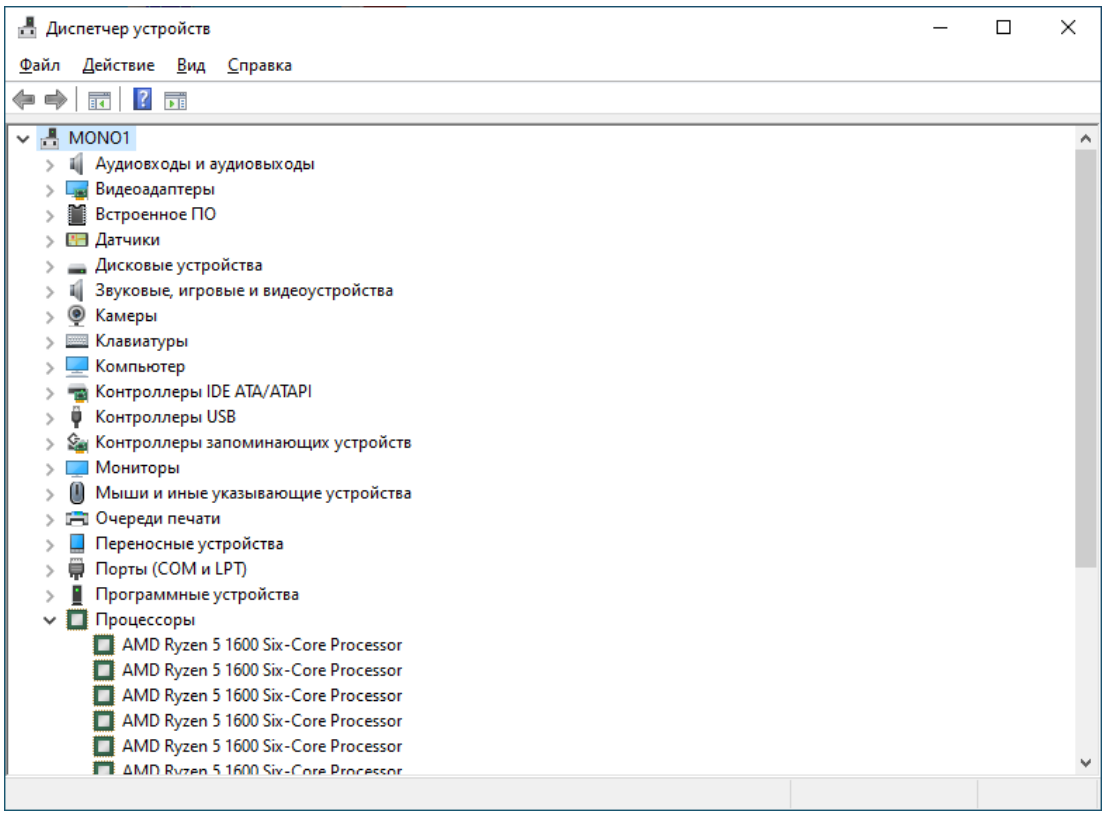


Рис. 1.18. Диспетчер устройств

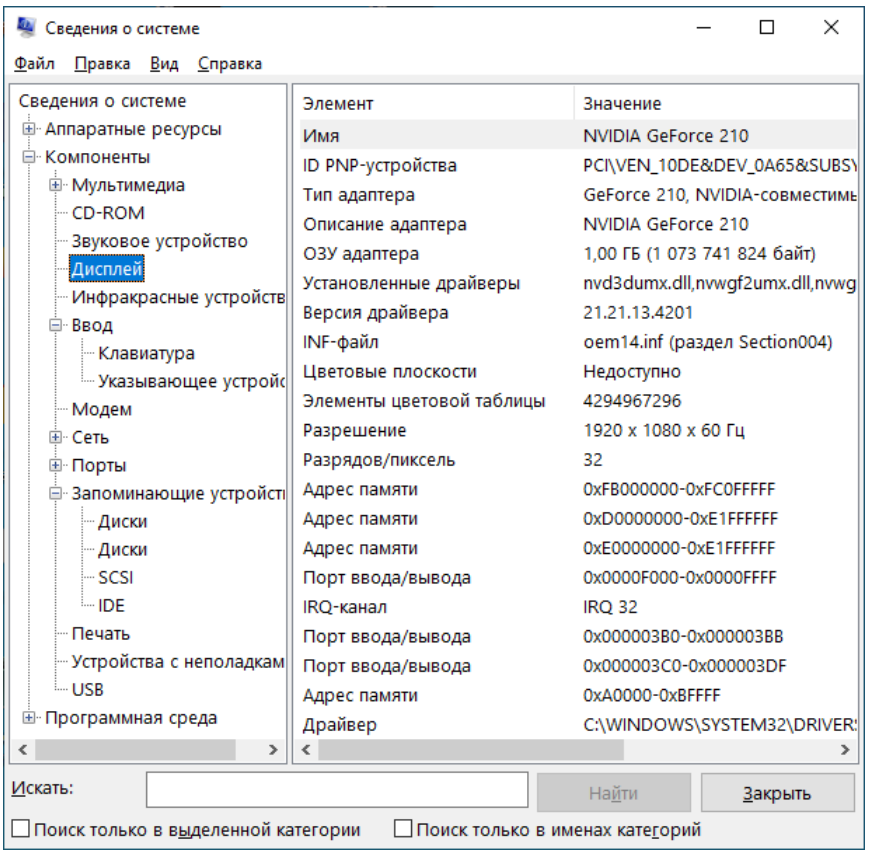


Рис. 1.19. Сведения о системе

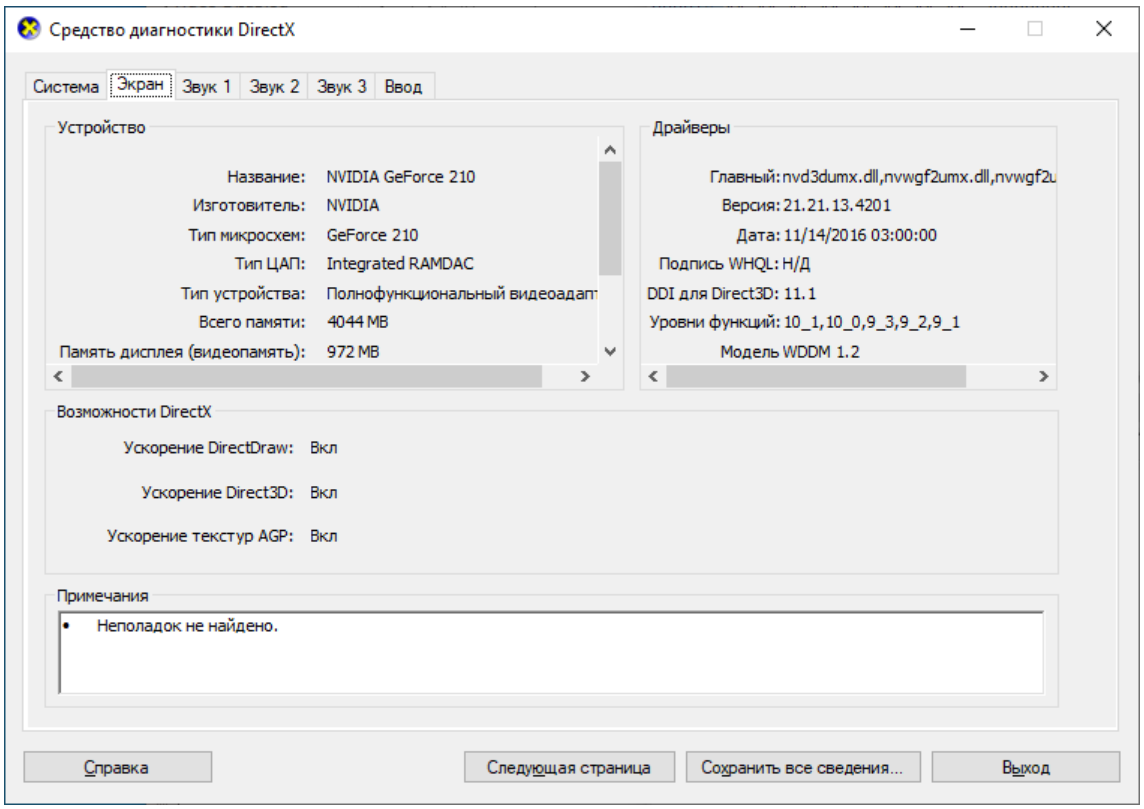


Рис. 1.20. Средство диагностики DirectX

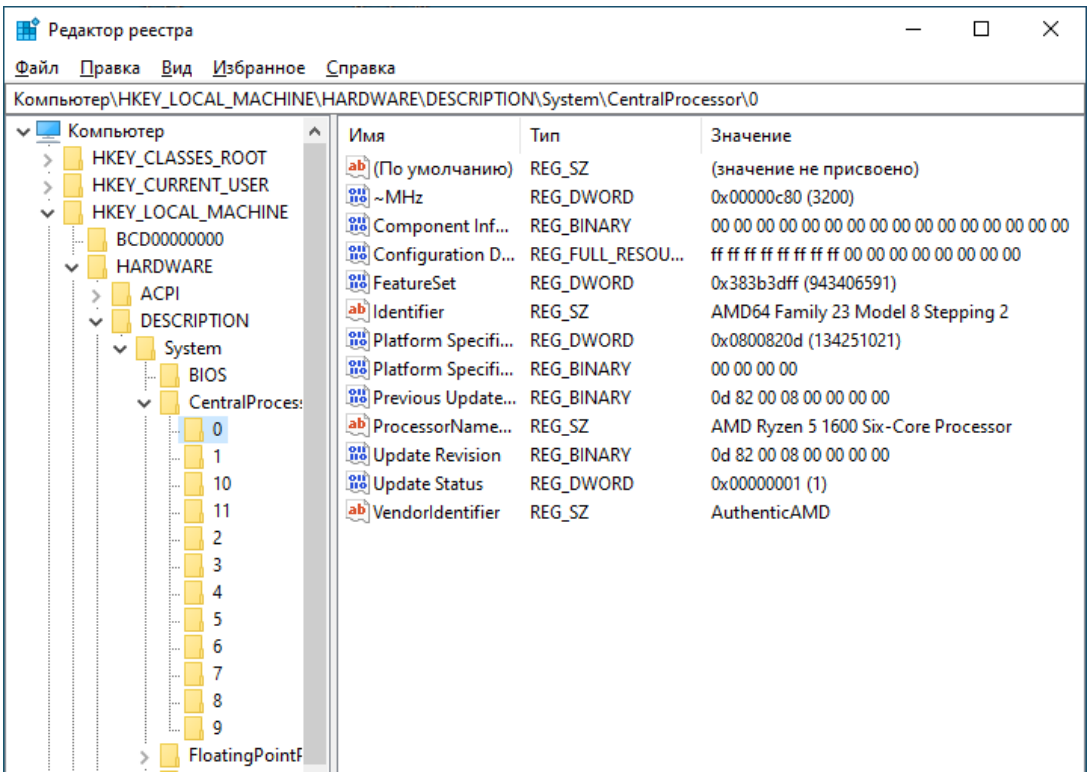


Рис. 1.21. Редактор реестра

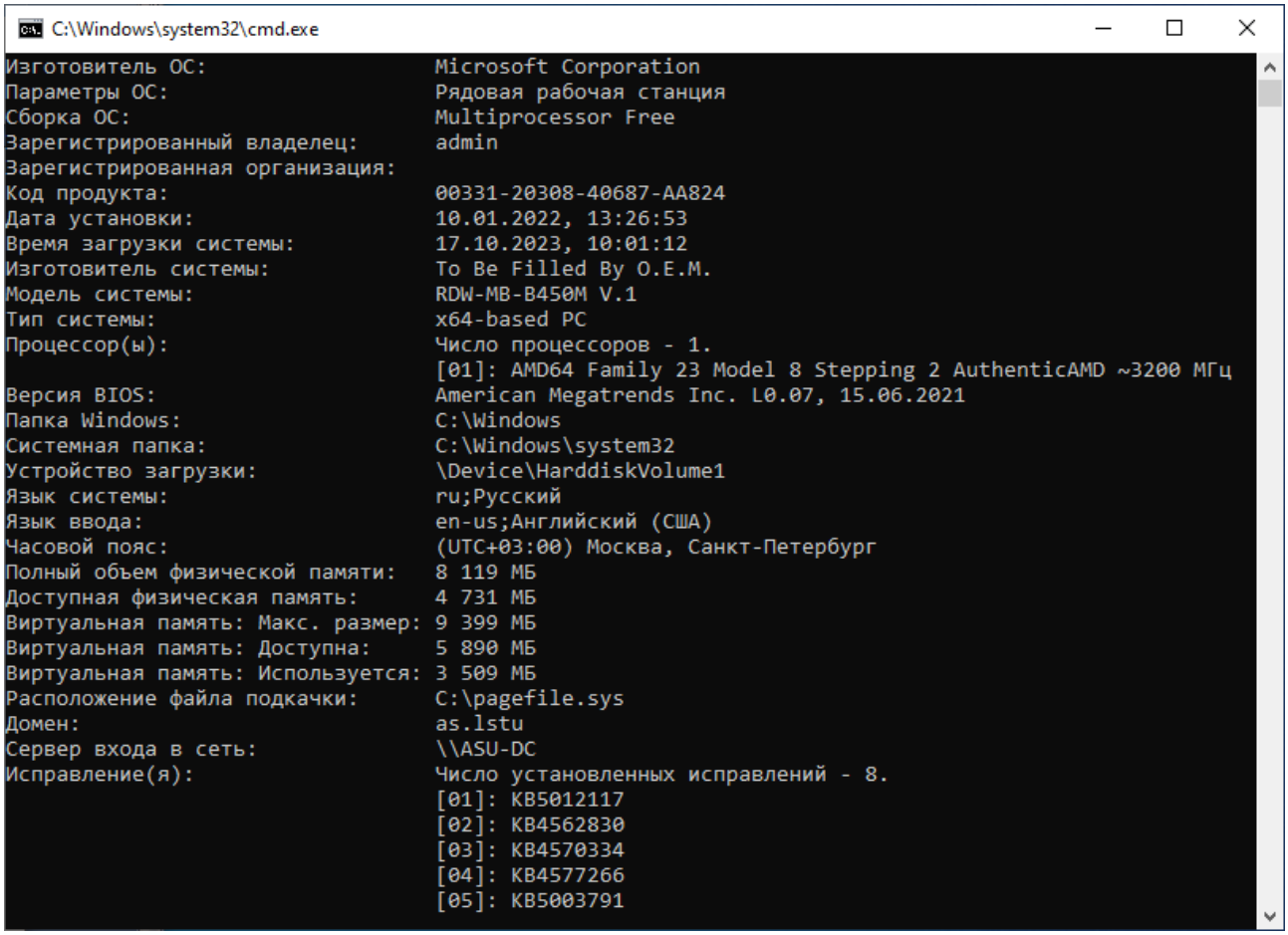


Рис. 1.22. Командная строка

С помощью данных средств определить:

− модель и основные характеристики процессора (CPU, Central Processing Unit): количество ядер, количество логических процессоров, тактовую частоту, объем кэш-памяти различных уровней;

− объем основной оперативной памяти (RAM, Random Access Memory);

− объем видеопамяти (Video RAM, VRAM);

− состав устройств ввода-вывода (периферийных устройств);

− состав и объем устройств внешней памяти (дисков, накопителей).

Полученную информацию оформить в виде таблицы (табл. 1.15).

Таблица 1.15

Состав и характеристики основных устройств компьютера

№ п/п	Программа	CPU	RAM	VRAM	Устройства ввода-вывода	Внешняя память
1	Параметры системы					
2	Диспетчер задач					
3	Диспетчер устройств					
4	Сведения о системе					
5	Средство диагностики DirectX					
6	Редактор реестра					
7	Командная строка					

В отчете должны быть скриншоты результатов использования указанных средств, подписи рисунков (скриншотов) и таблиц, а также ссылки на рисунки и таблицы.

Единицы измерения частоты и информационной емкости запоминающих устройств должны быть написаны по-русски: ГГц, Мбайт и т.п.

Задание 2. Математическая логика. Создание обрабатывающих и запоминающих устройств ЭВМ из логических элементов

Изучение математическо-логических основ устройства компьютера удобно изучать с помощью программного комплекса Proteus. Его демонстрационную версию можно скачать с сайта компании Labcenter Electronics [17]. Proteus имеет стандартный интуитивно-понятный графический интерфейс (рис. 1.23).

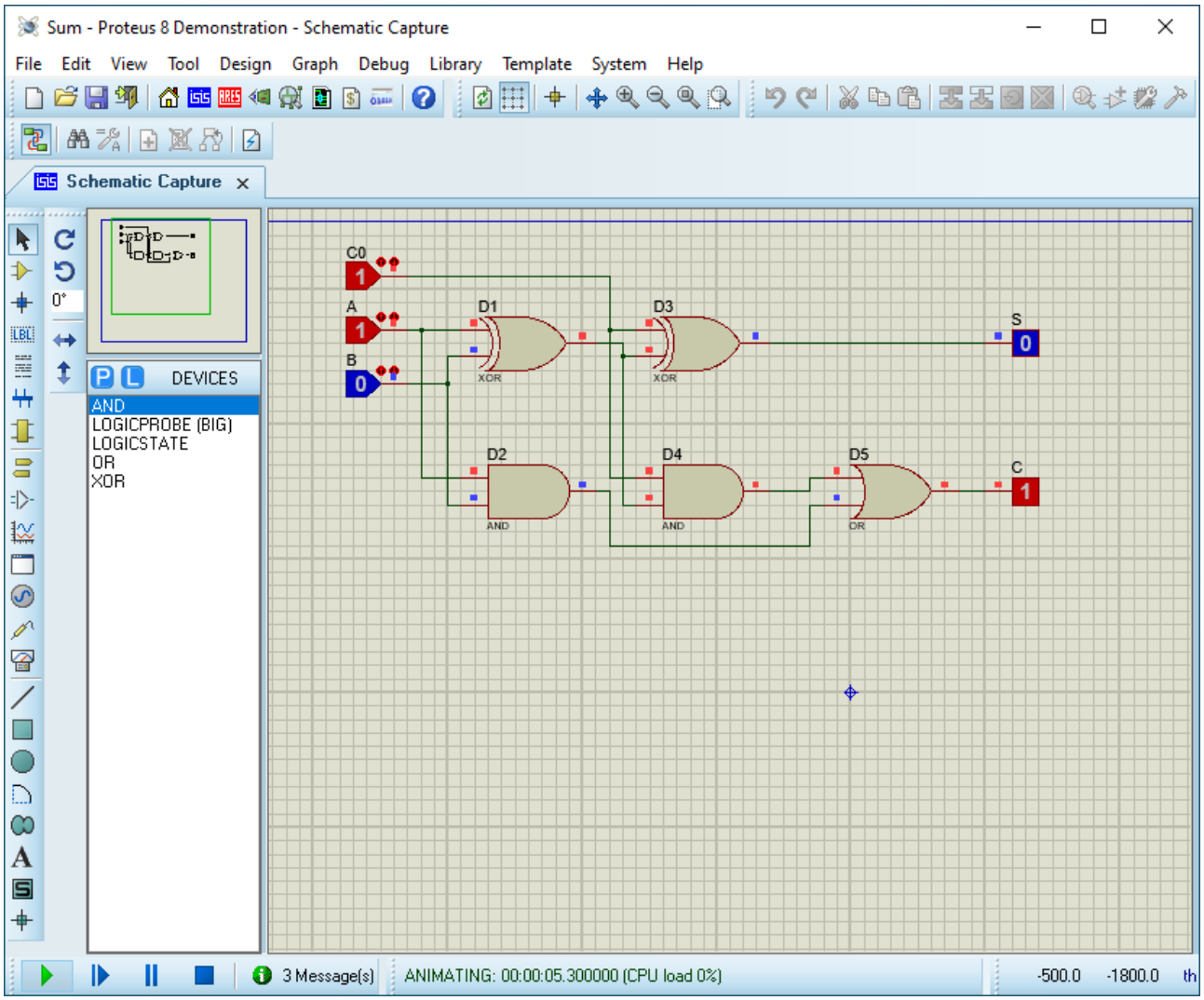


Рис. 1.23. Главное окно Proteus

В Proteus можно собрать схему электронного устройства и запустить ее в режиме имитационного моделирования. В Proteus имеется обширная библиотека электронных элементов от базовых: резисторов и конденсаторов — до микропроцессоров и микроконтроллеров. Для Proteus существует огромное число доступных руководств и учебных материалов, поэтому здесь мы не будем уделять внимание деталям работы в этой программе, тем более, что большое число встроенных примеров от самых простых до достаточно сложных позволяет быстро освоить этот инструмент без использования дополнительных материалов. Что касается теоретической подготовки по электроники, то школьного объема знаний по соответствующему разделу курса физики будет вполне достаточно.

Процесс выполнения этого задания включает три этапа:

1. Создание логического устройства.
2. Создание арифметического устройства.
3. Создание запоминающего устройства.

1. Создание логического устройства.

Исходной информацией для создания логического устройства является его заданная таблица истинности. По таблице истинности необходимо составить логическую функцию устройства, а затем по полученной логической функции создать схему устройства в Proteus. Исследовать работу схемы, убедиться, что созданное устройство работает правильно. Вариантами для выполнения этого задания могут быть заданные таблицы истинности. Дополнительными признаками формирования вариантов могут быть заданный логический базис для построения схемы, серия микросхем и т. д.

Пример 1

Дана таблица истинности по варианту (табл. 1.16).

Таблица 1.16

Данная таблица истинности по варианту

X1	X2	X3	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

По таблице истинности составляется СДНФ (совершенная дизъюнктивная нормальная форма) или СКНФ (совершенная конъюнктивная нормальная форма). СДНФ — это дизъюнкция конъюнкций. СКНФ – это конъюнкция дизъюнкций. Дизъюнкция — это логическая операция «ИЛИ», конъюнкция — «И». Каждая конъюнкция в СДНФ и каждая дизъюнкция в СКНФ называется термом.

Если в столбце значений выходной переменной (Y) меньше единичек, то удобнее составить СДНФ, а если меньше нулей, то — СКНФ. Для данного варианта единичек меньше, чем нулей, поэтому составляем СДНФ:

$$Y = \overline{X1}\,\overline{X2}\,\overline{X3} \vee \overline{X1}\,X2\,X3 \vee X1\,\overline{X2}\,\overline{X3}.$$

Термов в СДНФ столько, сколько единичек в столбце Y. Входная переменная в терм СДНФ входит с отрицанием, если ее значение в данной строке равно нулю, и без отрицания, если ее значение равно единице. Упрощаем формулу:

$$Y = \overline{X1}\,\overline{X2}\,\overline{X3} \vee \overline{X1}\,X2\,X3 \vee X1\,\overline{X2}\,\overline{X3} = \overline{X2}\,\overline{X3}(\overline{X1} \vee X1) \vee$$
$$\vee\, \overline{X1}\,X2\,X3 = \overline{X2}\,\overline{X3} \vee \overline{X1}\,X2\,X3.$$

Можно сразу использовать для получения логической функции карту Карно:

X2 X3		00	01	11	10
X1	0	1	0	1	0
	1	1	0	0	0

Или, по-другому:

	X2			
X1	1	0	1	0
	1	0	0	0
	X3			

В этой форме таблицы карты Карно жирная линия для каждой входной логической переменной соответствует ее значению «1».

Для получения минимальной ДНФ «склеиваем» соседние ячейки с единицами. Левая и правая стороны таблицы считаются соединенными, т.е. на самом деле эта таблица — кольцо, и первый столбец является соседним для четвертого. В «склейке» должно быть четное число единиц. Одна и та же единичка может участвовать в разных «склейках». Если остаются «несклеивающиеся» единички, их тоже учитываем:

	X2			
X1	1	0	1	0
	1	0	0	0
	X3			

«Склейке» в первом столбце таблицы соответствуют следующие значения входных переменных:

X1 = 0 и 1,

X2 = 0,

X3 = 0.

Таким образом, термы $\overline{X1}\,\overline{X2}\,\overline{X3}$ и $X1\,\overline{X2}\,\overline{X3}$ «склеиваются» по переменной X1, т.е. она пропадает, и остается терм:

$\overline{X2}\,\overline{X3}$.

Оставшейся одинокой единичке соответствуют следующие значения входных переменных:

X1 = 0,

X2 = 1,

X3 = 1,

а значит, следующий терм:

$\overline{X1}\,X2\,X3$.

В итоге получается та же самая логическая функция:

$$Y = \overline{X2}\,\overline{X3} \vee \overline{X1}\,X2\,X3.$$

Для ее схемотехнической реализации потребуется три логических элемента «НЕ», один двухвходовый логический элемент «И», один трехвходовый логический элемент «И» и один двухвходовый логический элемент «ИЛИ» (рис. 1.24).

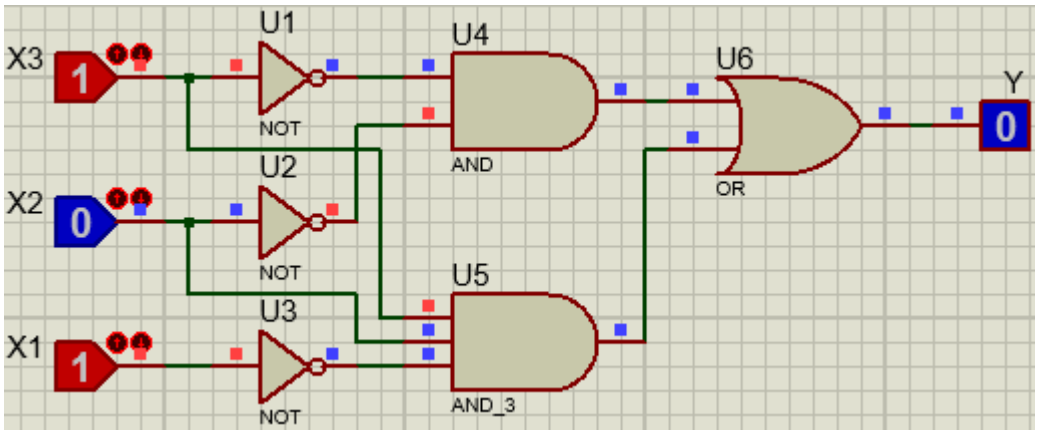


Рис. 1.24. Схема устройства, реализующего логическую функцию

$$Y = \overline{X2}\,\overline{X3} \vee \overline{X1}\,X2\,X3$$

Пример 2

Дана таблица истинности по варианту (табл. 1.17).

Таблица 1.17

Данная таблица истинности по варианту

X1	X2	X3	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Для данного варианта в столбце выходной переменной Y меньше нулей, чем единичек, поэтому составляем СКНФ:

$$Y = (X1 \vee \overline{X2} \vee \overline{X3})(\overline{X1} \vee X2 \vee X3)(\overline{X1} \vee \overline{X2} \vee \overline{X3}).$$

Термов в СКНФ столько, сколько нулей в столбце Y. Входная переменная входит в терм с отрицанием, если ее значение в данной строке равно единице, и без отрицания, если ее значение равно нулю.

Упрощаем формулу:

$$Y = (X1 \vee \overline{X2} \vee \overline{X3})(\overline{X1} \vee X2 \vee X3)(\overline{X1} \vee \overline{X2} \vee \overline{X3}) =$$
$$= ((\overline{X2} \vee \overline{X3}) \vee (X1\overline{X1}))(\overline{X1} \vee X2 \vee X3) = (\overline{X2} \vee \overline{X3})(\overline{X1} \vee X2 \vee X3).$$

Карта Карно для данного варианта:

X2 X3		00	01	11	10
X1	0	1	1	0	1
	1	0	1	0	1

Или

	X2			
X1	1	1	0	1
	0	1	0	1
		X3		

Для каждой входной логической переменной жирная линия соответствует ее значению «1».

«Склеиваем» соседние ячейки с нулями. В «склейке» должно быть четное число нулей. Один и тот же нуль может участвовать в разных «склейках». Если остаются «несклеивающиеся» нули, их тоже учитываем:

	X2			
X1	1	1	0	1
	0	1	0	1
		X3		

«Склейке» в третьем столбце таблицы соответствуют следующие значения переменных:

X1 = 0 и 1,

X2 = 1,

X3 = 1.

Таким образом, термы $(X1 \vee \overline{X2} \vee \overline{X3})$ и $(\overline{X1} \vee \overline{X2} \vee \overline{X3})$ «склеиваются» по переменной X1, т.е. она пропадает, и остается терм:

$(\overline{X2} \vee \overline{X3})$.

Оставшемуся нулю соответствуют следующие значения переменных:

X1 = 1,

X2 = 0,

X3 = 0,

а значит, следующий терм:

$(\overline{X1} \vee X2 \vee X3)$.

В итоге получается та же самая логическая функция:

$$Y = (\overline{X2} \vee \overline{X3})(\overline{X1} \vee X2 \vee X3).$$

Для ее схемотехнической реализации потребуется три логических элемента «НЕ», один двухвходовый логический элемент «ИЛИ», один трехвходовый логический элемент «ИЛИ» и один двухвходовый логический элемент «И» (рис. 1.25).

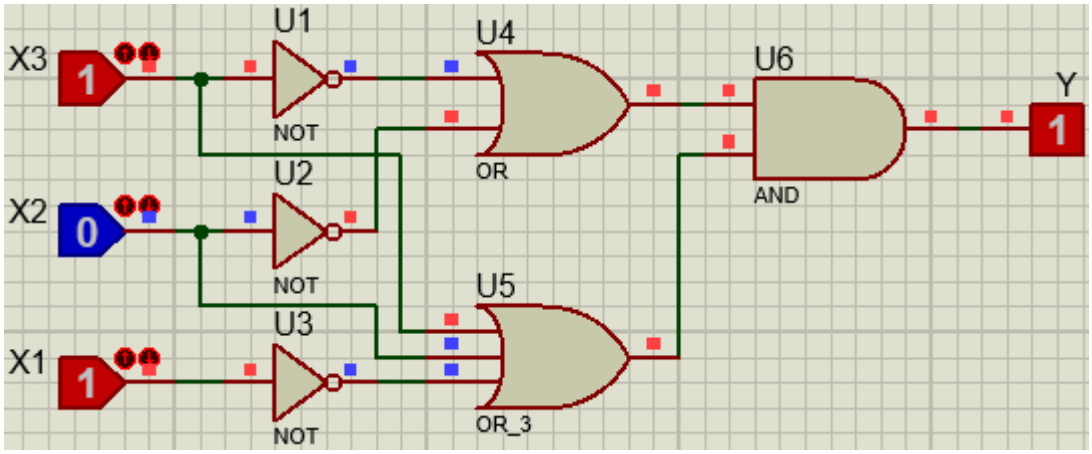


Рис. 1.25. Схема устройства, реализующего логическую функцию $Y = (\overline{X2} \vee \overline{X3})(\overline{X1} \vee X2 \vee X3)$

2. Создание арифметического устройства.

Для заданного арифметического устройства составить таблицу истинности. По составленной таблице истинности получить логическую функцию. По полученной логической функции создать схему данного арифметического устройства в Proteus. Исследовать его работу, убедиться, что созданное устройство работает правильно.

Признаками формирования вариантов могут быть выполняемая арифметическая операция, тип и разрядность обрабатываемых чисел, серия микросхем, логический базис.

Пример

Сумматор двухразрядных знаковых целых чисел, представленных в дополнительном коде, на микросхемах серии TTL 74S.

Диапазон знаковых целых чисел в двух разрядах: –2÷1. В дополнительном коде они представляются следующим образом:

$-2_{10} = 10_2$,

$-1_{10} = 11_2$,

$+0_{10} = 00_2$,

$+1_{10} = 01_2$,

Составляем таблицу истинности сумматора (табл. 1.18).

Таблица 1.18

Таблица истинности сумматора двухразрядных знаковых чисел

A1	A0	B1	B0	O	C	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
1	1	0	1	0	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	0	1	1	0

Здесь C — перенос из старшего разряда суммы (S1);

O — признак переполнения, указывающий на то, что результат получился неправильный.

Дальнейшая последовательность действий такая же, как в первом задании. Только теперь выходных переменных четыре, и для каждой из них нужно составить по таблице истинности логическую функцию, а по ней схему из микросхем заданной серии.

Карта Карно для выхода S0:

B1 B0		00	01	11	10
A1A0	00	0	1	1	0
	01	1	0	0	1
	11	1	0	0	1
	10	0	1	1	0

B1

A1				A0
	0	1	1	0
	1	0	0	1
	1	0	0	1
	0	1	1	0

B0

Делаем склейки:

B1

A1				A0
	0	1	1	0
	1	0	0	1
	1	0	0	1
	0	1	1	0

B0

У карты Карно логической функции четырех переменных связаны попарно все противоположные стороны: верхняя и нижняя, левая и правая. То есть, она фактически является тором (бубликом). Таким образом, здесь имеется только две склейки.

Для желтой склейки имеем следующий терм:

$A0\overline{B0}$.

Для оранжевой склейки следующий:

$\overline{A0}B0$.

Логическая функция младшего разряда суммы такова:

$S0 = A0\overline{B0} \vee \overline{A0}B0 = A0 \oplus B0$.

Смысл этого выражения вполне очевиден — младший разряд суммы равен единице тогда и только тогда, когда младшие разряды слагаемых имеют противоположные значения.

Карта Карно для выхода S1:

B1 B0		00	01	11	10
A1A0	00	0	0	1	1
	01	0	1	0	1
	11	1	0	1	0
	10	1	1	0	0

B1

A1				
	0	0	1	**1**
	0	1	0	1
	1	0	1	0
	1	1	0	0

A0

B0

Здесь почти ничего не склеилось. В левом нижем и правом верхнем углах имеется по две склейки, угловые единички (выделены жирным) участвуют в каждой из двух склеек. Плюс две отдельные единички. В итоге для S1 получается логическая функция с шестью термами:

$$S1 = A1\overline{B1}\,\overline{B0} \vee A1\overline{A0}\,\overline{B1} \vee \overline{A1}\,\overline{A0}B1 \vee \overline{A1}B1\overline{B0} \vee \overline{A1}A0\overline{B1}B0 \vee A1A0B1B0.$$

Единицу в старшем разряде суммы дает большое число комбинаций входных значений.

Логическую функцию S1 можно еще несколько упростить:

$$S1 = A1\overline{B1}(\overline{B0} \vee \overline{A0}) \vee \overline{A1}B1(\overline{A0} \vee \overline{B0}) \vee A0B0(\overline{A1}\,\overline{B1} \vee A1B1),$$

$$S1 = (\overline{A0} \vee \overline{B0})(A1\overline{B1} \vee \overline{A1}B1) \vee A0B0(\overline{A1}\,\overline{B1} \vee A1B1),$$

$$S1 = (\overline{A0} \vee \overline{B0})(A1 \oplus B1) \vee A0B0(\overline{A1 \oplus B1}),$$

$$S1 = \overline{A0B0}(A1 \oplus B1) \vee A0B0(\overline{A1 \oplus B1}),$$

$$S1 = (A0B0) \oplus (A1 \oplus B1).$$

Карта Карно для выхода C:

B1 B0		00	01	11	10
A1A0	00	0	0	0	0
	01	0	0	1	0
	11	0	1	1	1
	10	0	0	1	1

Склейки получаются такие. Первая:

B1

	0	0	0	0
	0	0	1	0
A1	0	1	1	1	A0
	0	0	1	1

B0

Дает терм:

$A1B1$.

Вторая:

B1

	0	0	0	0
	0	0	1	0
A1	0	1	1	1	A0
	0	0	1	1

B0

Дает терм:

$A1A0B0$.

Третья:

A1 \ B1 B0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	1	1	1
10	0	0	1	1

A0

Дает терм:

$A0B1B0$.

Логическая функция переноса C такова:

$$C = A1B1 \vee A1A0B0 \vee A0B1B0 = A1B1 \vee A0B0(A1 \vee B1).$$

Карта Карно для выхода O:

A1A0 \ B1 B0	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	0	0	1
10	0	0	1	1

Склейки получаются такие:

A1 \ B1 B0	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	0	0	1
10	0	0	1	**1**

A0

Логическая функция признака переполнения O такова:

$$O = A1\overline{A0}B1 \vee A1B1\overline{B0} \vee \overline{A1}A0\overline{B1}B0 = A1B1(\overline{A0} \vee \overline{B0}) \vee \overline{A1}\,\overline{B1}A0B0 =$$
$$= A1B1\overline{A0B0} \vee \overline{A1}\,\overline{B1}A0B0.$$

Таким образом, получили логические функции для всех выходных переменных сумматора двухразрядных знаковых чисел:

$$S0 = A0 \oplus B0,$$
$$S1 = (A0B0) \oplus (A1 \oplus B1),$$
$$C = A1B1 \vee A0B0(A1 \vee B1),$$
$$O = A1B1\overline{A0B0} \vee \overline{A1}\,\overline{B1}A0B0.$$

Соответствующая схема представлена на рис. 1.26.

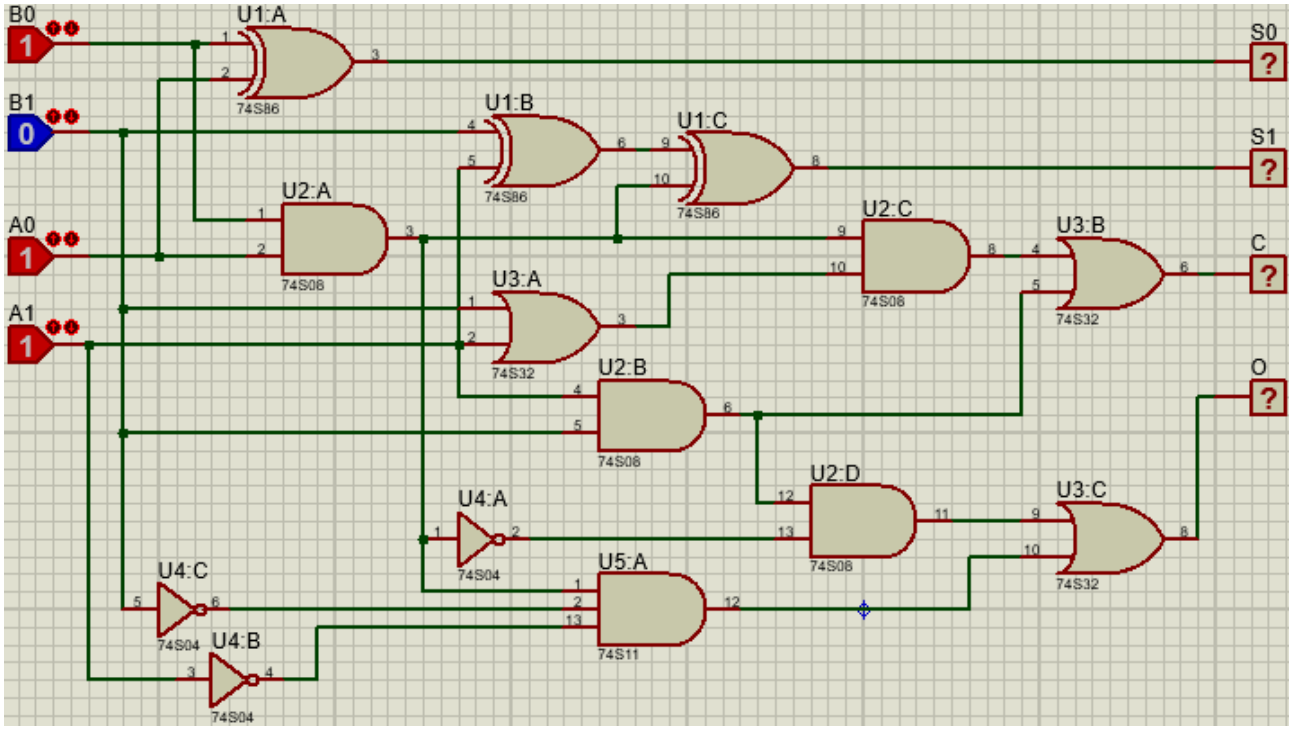


Рис. 1.26. Сумматор двухразрядных знаковых чисел на микросхемах серии TTL 74S с использованием элементов XOR

Однако в Proteus для логических элементов XOR серии 74S нет моделей, и исследовать работу схемы в имитационном режиме нельзя. Поэтому составим схему по формулам с операциями NOT, OR и AND (рис. 1.27):

$$S0 = A0\overline{B0} \vee \overline{A0}B0,$$

$$S1 = (\overline{A0} \vee \overline{B0})(A1\overline{B1} \vee \overline{A1}B1) \vee A0B0(\overline{A1}\,\overline{B1} \vee A1B1),$$

$$C = A1B1 \vee A0B0(A1 \vee B1),$$

$$O = A1B1(\overline{A0} \vee \overline{B0}) \vee \overline{A1}\,\overline{B1}A0B0.$$

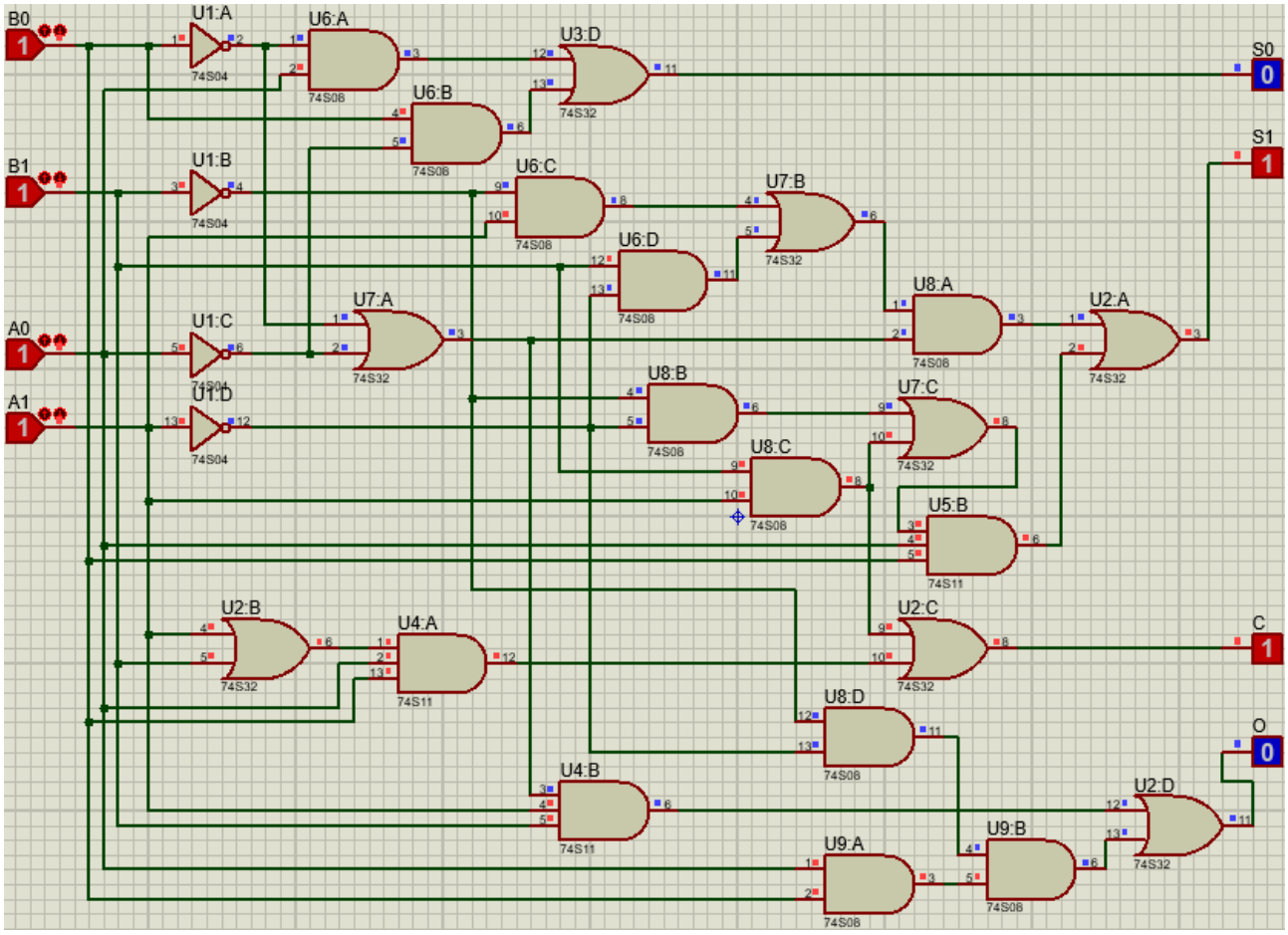


Рис. 1.27. Сумматор двухразрядных знаковых чисел
на микросхемах серии TTL 74S

3. Создание запоминающего устройства.

В Proteus создать схему минимального запоминающего устройства, триггера, из логических элементов. Исследовать работу схемы триггера, составить таблицу истинности.

Варианты для выполнения этого задания — тип триггера, логический базис, серия микросхем.

Пример

Создать RS-триггер в логическом базисе И-НЕ с использованием микросхем серии TTL 74S.

Берем два элемента 74S00 и соединяем их обратными перекрестными связями (рис. 1.28).

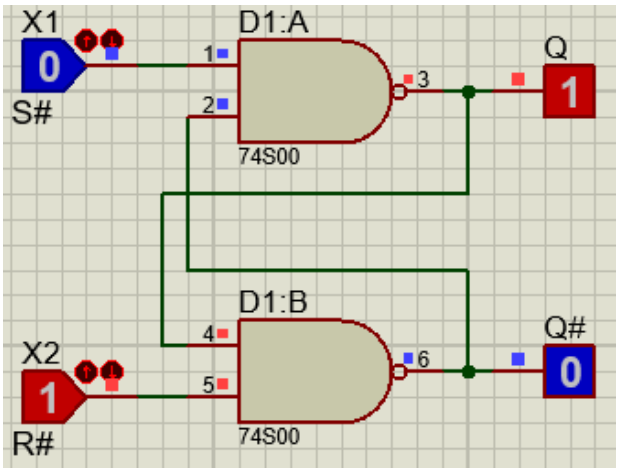


Рис. 1.28. Триггер из двух элементов И-НЕ серии TTL 74S

У триггера два выхода — прямой Q и инверсный Q#. По смыслу данного устройства они должны принимать противоположные значения.

Исследуем работу триггера, начиная со строки X1=0, X2=1 (табл. 1.19).

Таблица 1.19

Переключение триггера из состояния в состояние

№	X1	X2	Q	Q#
1	0	1	1	0
2	1	1	1	0
3	1	0	0	1
4	1	1	0	1
5	0	0	1	1

В первой строке прямой выход Q принимает значение 1. Значит, такая комбинация значений входных сигналов соответствует установке триггера или записи в него единички. В следующей строке значение выходных сигналов не изменяется. Значит, она соответствует состоянию, когда триггер хранит ранее записанное в него значение. В третьей строке выход Q принимает значение 0. Значит, такая комбинация значений входных сигналов соответствует сбросу триггера или записи в него нуля. Следующая строка снова соответствует состоянию хранения. И наконец, в последней строке прямой и инверсный выходы принимают одинаковое значение (1), значит, такая комбинация входных значений должна трактоваться, как запрещенная. Видно, что состоянию хранения информации соответствуют две единички на входе. Значит, единица — неактивный уровень входного сигнала. Таким образом, в первой строке активный уровень имеет сигнал X1, значит, его можно назвать сигналом установки S#. В третьей строке активный уровень имеет сигнал X2, значит, его можно назвать сигналом сброса R#. В итоге, получаем следующую таблицу истинности этого триггера (табл. 1.20).

Таблица 1.20

Таблица истинности триггера

X1 (S#)	**X2 (R#)**	**Q_n**	**Q_n#**	**Состояние**
0	0	1	1	Запрещенное (недопустимое)
0	1	1	0	Установка (запись единицы)
1	0	0	1	Сброс (запись нуля)
1	1	Q_{n-1}	Q_{n-1}#	Хранение

Здесь было необходимо ввести индекс для выходных сигналов триггера, который означает номер такта (момента дискретного времени), в котором изменились значения входных сигналов.

Задание 3. Основные принципы устройства и работы ЭВМ

Как было отмечено выше, языки высокого уровня не очень хорошо подходят для изучения архитектуры ЭВМ. Они были созданы для облегчения труда программистов по созданию прикладного и системного программного обеспечения. Но здесь нам нужен «микроскоп», который покажет все необходимые детали. Современные операционные системы типа Windows с графическим интерфейсом и очередью сообщений тоже не подойдут для решения наших задач. Они скрывают оборудование посредством HAL (Hardware Abstraction Layer), сами переключают процессор в защищенный режим и практически ничего не позволяют делать запускаемым программам самостоятельно. Потому нам понадобится максимально простая среда: операционная система FreeDOS или MS-DOS, оболочка Volkov Commander (или подобная) и любой ассемблер. Лучше всего для учебных целей подходит Turbo Assembler. Он имеет наиболее простой и ясный синтаксис, в структуре исходного кода программы, написанной на этом языке, четко видно содержимое памяти компьютера после запуска программы. Поскольку чаще всего во всех учебных лабораториях на компьютерах установлены операционные системы Windows или Linux с соответствующим дополнительным программным обеспечением, то необходимо использовать виртуальные машины с нужной нам средой или эмуляторы типа DOS-Box или Bochs. Конечно, в домашних условиях лучше всего иметь «чистый» DOS на загрузочном диске. Можно создать загрузочную флешку. Также в Интернете есть сайты, предоставляющие соответствующую среду. Можно использовать и их. Мы будем ориентироваться на эмулятор Bochs [18]. Несмотря на некоторые его недостатки, это наиболее универсальный вариант и для домашних, и для рабочих условий.

Bochs (читается «бокс») — свободная программа для эмуляции компьютера с процессором архитектуры x86. В используемом экземпляре Bochs виртуальный компьютер сконфигурирован с одним жестким диском (C:) с операционной системой FreeDOS (каталог DOS), инструментальной системой программирования на языке ассемблера Turbo Assembler компании Borland (каталог TASM) и примерами программ на ассемблере (каталог SAMPLE). (Понятие ката-

лога (директории) в операционной системе DOS — синоним понятия папки в операционной системе Windows.) Bochs может работать с буфером обмена Windows, что может быть полезно при редактировании программ и составлении отчета. Программу можно редактировать в каком-нибудь текстовом редакторе Windows, вставляя текст в соответствующие файлы в Bochs через буфер обмена. Листинг программы, сгенерированный в Bochs, можно вставить в отчет также через буфер обмена.

Подробная последовательность действий по созданию программы: запуск необходимых программных средств, редактирование программы, компиляция, компоновка, запуск программы в отладчике — представлена в табл. 1.21.

Таблица 1.21

Последовательность действий в процессе работы с Bochs и TASM

Действие	Результат
Скопировать папку BochsInstl в рабочую папку	В рабочей папке появится папка BochsInstl
Запустить программу bochs.exe из папки BochsInstl	Появится окно консоли программы Bochs со списком вариантов действий
Из списка вариантов действий оставить предлагаемый по умолчанию 6 — Begin simulation, нажать клавишу Enter	Появится главное окно (Display) программы Bochs. Ниже заголовка окна, как обычно, панель инструментов. Нижняя строка — строка статуса. В рабочей области окна — командная строка операционной системы FreeDOS: C:\>_ В ней можно вводить команды. Удобства в работе здесь предоставляет оболочка операционной системы, например, файловый менеджерVolkov Commander

Действие	Результат
Запустить Volkov Commander — в командной строке ввести: vc\vc	В рабочей области Bochs появится экран Volkov Commander: две панели, в каждой из которых отображается содержимое какого-либо каталога (папки), в данном случае корневого каталога диска C. В строке ниже панелей — командная строка. В нижней строке — подсказка о значениях функциональных клавиш. (Если появится сообщение об ошибке, что нет диска D, ввести с клавиатуры C, нажать Enter)
Для возможности чтения и ввода русских букв запускаем русификатор клавиатуры — rk.com: — выделяем его стрелкой вниз, нажимаем Enter	Запускается русификатор клавиатуры, визуально ничего не изменяется. Переключение раскладки — Ctrl + LeftShift
В правой панели перейти в папку SAMPLE\SMPL1. Переход между панелями — Tab. Навигация в панели — стрелки и т. п. Для входа в папку выделяем ее, нажимаем Enter. Выход из папки на один уровень вверх — выделяем верхнюю строку с двоеточием, Enter. Выход сразу в корневую папку — Ctrl + \	В правой панели появится содержимое папки SAMPLE\SMPL1
В левой панели перейти в папку TASM	В левой панели появится содержимое папки TASM
Скопировать пример sum16.asm из папки Smpl1 в папку TASM — в правой панели выделяем файл sum16.asm, нажимаем F5 (Copy), в появившемся окне проверяем, что копируется нужный файл в нужное место, подтверждаем (Enter)	В папке TASM появляется файл sum16.asm

Действие	Результат
Переходим в папку TASM, выделяем файл sum16.asm, открываем в режиме просмотра — F3 (View), изучаем структуру программы на языке ассемблера	На экране появляется содержимое файла sum16.asm — текст программы сложения двух чисел, находящихся в переменных A и B, на языке ассемблера. Внизу подсказка о значениях функциональных клавиш
Выходим из режима просмотра — Esc	На экране — панели Volkov Commander
Переименовываем файл sum16.asm в lab1.asm: Ctrl + F6, вводим новое имя, Enter	Имя файла изменяется на lab1.asm

Открываем файл lab1.asm в режиме редактирования — F4 (Edit), вносим изменения в программу в соответствии с вариантом:
Перед первой переменной объявляем пустую переменную:
dw ?
Изменяем имена переменных A, B и S на X, Y и R (и в месте объявления, и в месте использования), добавляем переменную Z;
Задаем значения переменных с исходными данными: X, Y, Z;
Добавляем нужные команды. Например, после команд:
mov ax, X
add ax, Y
добавляем новую строку и вводим в ней команду:
sub ax, Z
В результате выполнения этих команд в регистре ax будет значение выражения X + Y − Z.
Добавляем еще одну строку и вводим в ней нужную команду сдвига, например:
	На экране текст программы, в котором присутствует курсор, вводимые с клавиатуры символы появляются в месте курсора, курсор перемещается на следующую позицию. Курсор перемещаем в нужную позицию стрелками, Home (в начало строки), End (в конец строки) и т. п.
	Удаление символа слева от курсора — Backspace, над курсором — Delete.
	Добавление новой строки — Enter.
	Внизу подсказка о значениях функциональных клавиш.
	При редактировании можно использовать буфер обмена Windows. В панели инструментов Bochs есть кнопки Copy и Paste. Из Bochs в буфер обмена копируется содержимое экрана. Поэтому, если весь текст файла не помещается на экране, копировать нужно по частям, удаляя при этом ненужные

Действие	Результат
shl ax, 1 – сдвиг на 1 влево — умножение на 2 Добавляем еще одну строку и вводим в ней нужную логическую команду, например: and ax, 0F0F0h	строки
Сохраняем отредактированный файл — F2 (Save) — и выходим из режима редактирования (Esc)	На экране — панели Volkov Commander
Компилируем программу — в командной строке вводим: tasm lab1 (tasm.exe — компилятор ТурбоАссемблера). Нажимаем Enter. При этом мы должны находиться в папке TASM. Можно создать нужный текст в командной строке и по-другому: Выделяем файл с компилятором tasm.exe, нажимаем Ctrl + Enter, в командной строке появляется tasm.exe. Выделяем файл с текстом программы (исходный файл) lab1.asm, нажимаем Ctrl + Enter, в командной строке оказывается tasm.exe lab1.asm Для создания файла с листингом программы компилятор нужно запустить с параметром *l*: tasm /l lab1 Ранее введенные в командной строке команды можно в ней повторить, нажав ↑. При этом панели Volkov Commander должны быть убраны	Происходит компиляция — перевод программы с языка ассемблера на язык процессора (машинный язык). Если ошибок в программе нет, компиляция проходит успешно, и в папке TASM появляется файл с программой на языке процессора (объектный файл) lab1.obj Если есть ошибки, компилятор выведет на экран перечень строк с ошибками с названием ошибок. Снова редактируем исходный файл, устраняем ошибки, компилируем. Посмотреть сообщение компилятора можно, убрав панели Volkov Commander нажатием Esc или Ctrl + O. Вернуть панели — так же Esc или Ctrl + O. Если компилятору задан параметр *l*, кроме объектного файла появится файл листинга lab1.lst. В листинге помимо текста программы присутствуют номера строк, адреса переменных и команд, машинный код
Компонуем программу — в командной строке вводим:	Происходит компоновка — создание исполняемого файла. В папке

Действие	Результат
tlink lab1 (tlink.exe — компоновщик (редактор связей, линкер) ТурбоАссемблера). Нажимаем Enter. Можно сделать и как на предыдущем этапе: Выделяем файл с компоновщиком tlink.exe, нажимаем Ctrl + Enter, в командной строке появляется tlink.exe. Выделяем объектный файл lab1.obj, нажимаем Ctrl + Enter, в командной строке оказывается tlink.exe lab1.obj. Нажимаем Enter	TASM появляется исполняемый файл lab1.exe
Открываем исполняемый файл в отладчике — td lab1 (td.exe — отладчик ТурбоАссемблера). Нажимаем Enter. Или выделяем файл с отладчиком td.exe, нажимаем Ctrl + Enter, в командной строке появляется td.exe. Выделяем исполняемый файл lab1.exe, нажимаем Ctrl + Enter, в командной строке оказывается td.exe lab1.exe. Нажимаем Enter. В открывшемся окне отладчика соглашаемся с его сообщением, что в программе нет таблицы символов — Enter	Появляется экран отладчика. Верхняя строка — главное меню, нижняя — подсказка о значениях функциональных клавиш: F7 — Trace (пошаговое выполнение команд: одна команда); F8 — Step (пошаговое выполнение команд: одна команда, полностью цикл или подпрограмма); и т. д. В основной части рабочей области пять полей, по часовой стрелке: сегмент команд (адрес, машинный код, ассемблерный код); регистры процессора; флаги (отдельные биты регистра флагов); сегмент стека; сегмент данных. Переход между этими полями — Tab по часовой стрелке, Shift + Tab — против часовой стрелки (как в Windows). При пошаговом выполнении ко-

Действие	Результат
	манд изменяющиеся регистры и флаги выделяются белым цветом. Чтобы посмотреть данные в памяти, **после** инициализации сегментных регистров переходим в поле сегмента данных (Tab или Shift + Tab), нажимаем Ctrl + G и вводим адрес начала сегмента: ds:0000 или es:0000. Выход из отладчика — Alt + X

Пример простой программы без ввода-вывода, переходов и циклов, с основными арифметическими и логическими командами

Программа вычисляет значение выражения:

$$R = (X + Y - Z) \cdot 2 \text{ and } 0F0F.$$

В программе три сегмента: сегмент данных с исходными данными, сегмент данных с результирующей переменной и сегмент команд:

```
;в сегменте indat находятся переменные X, Y и Z
indat segment          ;начало сегмента indat
        dw ?           ;пустая переменная, нужна для bochs
        X dw 05h       ;переменная X
        Y dw 0Ah       ;переменная Y
        Z dw 02h       ;переменная Z
indat ends             ;конец сегмента indat
;в сегменте outdat находится переменная R, в которую запишется результат
outdat segment         ;начало сегмента outdat
        R dw ?         ;R не инициализирована, в нее запишется результат
outdat ends            ;конец сегмента outdat
```

```
code segment            ;начало сегмента code
;сегментные регистры будут указывать на соответствующие сегменты:
        assume cs:code, ds:indat, es:outdat
begin: mov dx, indat    ;точка входа в программу, инициализация
        mov ds, dx      ;сегментного регистра ds
        mov dx, outdat  ;инициализация сегментного
        mov es, dx      ;регистра es
        mov ax, X       ;в регистр AX (аккумулятор) читается переменная X
        add ax, Y       ;к X прибавляется Y: ax = X + Y
        sub ax, Z       ;ax = X + Y – Z
        shl ax, 1       ;сдвиг ax влево на 1 бит: ax = (X + Y – Z) * 2
        and ax, 0FF0h   ;ax = (X + Y –Z) * 2 and 0FF0
        mov R, ax       ;запись результата в переменную R
        mov ax, 4c00h   ;запись в ax номера функции завершения программы
        int 21h         ;вызов прерывания DOS для завершения программы
code ends               ;конец сегмента code
        end begin   ;конец программы (команда с меткой begin — точка входа)
```

Скомпилируем программу, скомпонуем ее и откроем исполняемый файл в отладчике. В процессе пошагового выполнения программы исследуем работу процессора по выполнению команд — изменение значений регистров и переменных. (На регистр флагов пока что можно внимания не обращать). Заполним таблицу (табл. 1.21).

Таблица 1.21

Результаты исследования

№	Адрес команды	Команда на машинном языке (машинный код)	Команда на языке ассемблера (ассемблерный код)	Регистр команд	Указатель команд IP	Изменившиеся регистры и переменные
1	0000	BA9243	mov dx, 4392	BA	0003	dx = 4392
2	0003	8EDA	mov ds, dx	8EDA	0005	ds = 4392
3	0005	BA9343	mov dx, 4393	BA	0008	dx = 4393

4	0008	8EC2	mov es, dx	8EC2	000A	es = 4393
5	000A	A10200	mov ax, [0002]	A1	000D	ax = 0005
6	000D	03060400	add ax, [0004]	0306	0011	ax = 000F
7	0011	2B060600	sub ax, [0006]	2B06	0015	ax = 000D
8	0015	D1E0	shl ax, 1	D1E0	0017	ax = 001A
9	0017	25F00F	and ax, 0FF0	25	001A	ax = 0010
10	001A	26A30000	mov es:[0000], ax	26A3	001E	es:[0000] = 0010
11	001E	B8004C	mov ax, 4C00	B8	0021	ax = 4C00
12	0021	CD21	int 21	CD	001A	

Отладчик в командах mov dx, indat и mov dx, outdat показывает конкретные значения баз этих сегментов. Они становятся известны только после загрузки программы в память операционной системой. Вместо имен переменных в ассемблерном коде отладчик показывает их внутрисегментные адреса (смещения). В машинном коде команды mov R, ax (mov es:[0000], ax) появился дополнительный байт 26. Это префикс замены сегмента (см. п. 3.3). По умолчанию процессор ищет переменные в основном сегменте данных, по сегментному регистру DS. Если переменная располагается в другом сегменте, требуется явное указание этого, для чего и используется префикс замены сегмента.

Регистр команд отладчик не показывает. Но его содержимое можно определить теоретически. В него помещается та часть команды, которая должна быть декодирована. А это все, кроме смещения (адреса от начала сегмента) и непосредственного операнда.

Операцию сложения процессор выполняет по команде ADD, операцию вычитания — по команде SUB.

Операцию умножения на 2 можно реализовать с помощью команды логического сдвига влево SHL (SHift Left), операцию целочисленного деления на 2 — с помощью команды логического сдвига вправо SHR (SHift Right) для беззнаковых чисел или арифметического сдвига вправо SAR для знаковых чисел.

Логические команды:

AND — команда поразрядной логической операции "И".

OR — команда поразрядной логической операции "ИЛИ".

XOR — команда поразрядной логической операции "Исключительное ИЛИ".

Более подробную информацию о командах процессора можно получить из книги Зубкова «Ассемблер» (п. 2.3. Основные непривилегированные команды) [19].

Задание:

1. Составить блок-схему алгоритма программы для вычисления значения заданного выражения с арифметическими и логическими операциями.

2. Провести ручной расчет по алгоритму для заданных исходных значений.

3. Написать по составленному алгоритму программу на языке ассемблера.

4. Выполнить программу в отладчике в пошаговом режиме (покомандно) и после выполнения каждой команды фиксировать состояние указателя команд, других регистров, задействованных в программе, ячеек памяти данных (переменных), определить значение регистра команд.

В программе должны быть 16-битные переменные для исходных данных (операндов выражения) и результата (рассчитанного значения выражения).

Результаты исследования работы процессора по выполнению программы оформить в виде таблицы (табл. 1.21).

В отчете привести также скриншот отладчика со значением результирующей переменной.

2. АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ. СТРУКТУРА ЭВМ

2.1. ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ ВЫЧИСЛИТЕЛЬНЫХ МАШИН

Технико-эксплуатационные показатели ЭВМ

Показатели — это характеристики, имеющие числовое выражение.

Технико-эксплуатационные показатели ЭВМ — это численные величины, характеризующие ее свойства и возможности в соответствии с основным предназначением, обработкой информации, и связанные с ними. Основные свойства ЭВМ, как и любой другой технической системы, — это эффективность (производительность) и надежность. Они характеризуются множеством показателей. Их можно разделить на общие, характеризующие ЭВМ в целом, и частные, характеризующие отдельные части ЭВМ.

Общие технико-эксплуатационные показатели ЭВМ:

- показатели надежности: средняя наработка на отказ и др.;
- показатели производительности:
  - пиковая производительность;
  - среднее время прохождения задачи;
  - реальная производительность по различным тестовым задачам;
- потребляемая мощность;
- габариты, масса.

Потребляемая мощность, габариты и масса непосредственно не определяют вычислительную способность ЭВМ, но тесно связаны с ней и имеют большое значение при ее эксплуатации: каков должен быть размер помещения для установки ЭВМ, какова должна быть система питания и охлаждения и т. п.

Надежность ЭВМ — это ее свойство сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции, в заданных режимах и условиях эксплуатации. Один из основных показателей надежности — средняя наработка на отказ или среднее время между отказами (MTBF, Mean Time Between Failures). Для компьютера ENIAC, как мы помним, она составляла всего 20 часов. У современных ЭВМ средняя наработка на отказ неизмеримо больше. Это связано с элементной базой ЭВМ четвертого поколения, с использованием микропроцессоров и других сверхбольших интегральных схем, что дает относительно небольшое число отдельных конструктивных элементов и электрических связей между ними. А сами по себе микросхемы достаточно надежны. Из-за высокой надежности современных ЭВМ и их компонентов численно определить значения их показателей надежности непросто. Но иногда ориентировочные значения MTBF приводятся в документации или пресс-релизах. Например, MTBF материнской платы Intel Endeavor (1996 г.) составляет 30 000 ч, мини-ПК Polywell B760L2 (2023 г.) — 100 000 ч, жесткого диска Seagate HGST Ultrastar He10 (2014 г.) — 2,5 миллиона часов.

Главным свойством ЭВМ является ее производительность (быстродействие, вычислительная мощность). **Производительность ЭВМ** — это ее способность выполнять определенный объем вычислений за определенный интервал времени. Существует множество видов и показателей производительности. ЭВМ — это устройство обработки информации, и по идее основным показателем производительности должно быть количество обрабатываемой информации в единицу времени. Но в отличие от машин обработки материальных объектов, например, станков, это определить затруднительно. Поэтому используются другие показатели производительности. Один из показателей — среднее время прохождения задачи. Это среднее время выполнения задачи от начала ввода исходных данных до окончания вывода результатов. Усреднение здесь используется, как и в любом эксперименте, по причине того, что существует множество фак-

торов, которые влияют на время выполнения задачи, и поэтому оно будет варьироваться от запуска к запуску. Конечно, таких задач для каждой ЭВМ может быть сколько угодно, и у каждой задачи будет свое среднее время прохождения. Поэтому более предпочтительными являются унифицированные показатели. ГОСТ 16325-88 «Машины вычислительные электронные цифровые общего назначения. Общие технические требования» дает следующее определение производительности. Производительность ЭВМ ($P_к$), команд/с, — величина, обратная среднестатистическому времени выполнения команды при решении достаточно большого количества задач:

$$P_к = \frac{\sum_{(j)} b_j}{\sum_{(j)} b_j t_j}, \qquad (2.1)$$

где b_j — вес команды j-го типа;

t_j — среднее время выполнения команды j-го типа.

Фактически определяемая таким образом производительность — это средневзвешенное число команд, выполняемых в единицу времени. Использовать эту формулу непросто. Нужно как-то определить веса и среднее время выполнения команд различного типа. В ГОСТ 16325-88 приводятся соответствующие формулы и таблицы с весами, но нет гарантий, что они одинаково хорошо подойдут для всех типов ЭВМ и всех видов задач.

Быстродействие по тому же стандарту является другим показателем и определяется как количество коротких команд (типа сложения с фиксированной точкой), выполняемых за 1 с.

Проблема определения производительности сильно упрощается для компьютеров на RISC-процессорах, в которых команды выполняются за один такт. В этом случае производительность численно равна тактовой частоте, на которой работает процессор. Правда, надо заметить, что и здесь не все так просто. Несмотря на простоту системы команд RISC-процессоров, есть команды более простые и более

сложные, и даже при наличии конвейера более сложные команды выполняются более, чем за один такт. Поэтому в итоге в качестве наиболее универсального показателя вводится понятие пиковой (теоретической) производительности — теоретического максимума быстродействия вычислительной системы, который рассчитывается, как и для процессора, по формуле (1.2), которая приводилась в первом разделе:

$$E = I / T = (I / Clock) \cdot (Clock / T) = IPC \cdot F,$$

где I — количество выполненных команд;

T — время;

$Clock$ — количество тактов;

F — тактовая частота;

IPC — Instructions Per Clock, количество команд, выполняемых за такт.

Определение пиковой производительности близко к определению быстродействия по стандарту ГОСТ 16325-88.

При наличии параллельно работающих исполнительных блоков вычислительной системы, каковыми могут быть отдельные конвейеры процессора, ядра процессора, процессоры в целом или компьютеры в целом, максимально возможное число команд, выполняемых за один такт, больше единицы.

Формула (1.2) явным образом показывает наличие двух факторов или способов повышения производительности вычислительной системы — увеличение тактовой частоты и распараллеливание потока команд по исполнительным блокам.

При определении пиковой производительности считается, что величина IPC постоянна, т. е. в каждом такте выполняется максимально возможное число команд. В реальной ситуации это не так, и вводится другой показатель — реальная производительность, который определяется в соответствии с концепцией формулы (2.1), но

рассчитывается не теоретически, а экспериментально с помощью различных тестовых задач определенной направленности.

Реальная производительность компьютера всегда ниже пиковой по двум причинам.

Первая — в определении пиковой производительности предполагается, что вся необходимая информация для выполнения программы, и команды, и данные, всегда доступна процессору. Однако эта информация не присутствует в процессоре в полном объеме, а читается им из памяти. Канал между процессором и памятью — это так называемое «бутылочное горлышко» (bottle neck) фон-неймановской архитектуры. В какие-то промежутки времени процессор не выполняет вычисления, а читает информацию из памяти. Эта проблема решается увеличением пропускной способности канала с памятью и многоуровневой организацией памяти с наличием кэш-памяти на определенных уровнях.

Вторая причина состоит в том, что не все параллельные блоки при их наличии могут быть загружены в каждый момент времени. Для их полной загрузки необходимо, чтобы у исполняемой программы была необходимая степень внутреннего параллелизма, которую можно определить как долю команд, которые могут быть выполнены параллельно. Таким образом, создание параллельного кода — второй способ увеличения реальной производительности параллельной вычислительной системы.

И пиковая, и реальная производительность определяются отдельно для целочисленных команд и команд с плавающей точкой. Соответственно, используются такие единицы производительности:

– IPS (Instructions Per Second) — количество целочисленных операций в секунду;

– FLOPS (Floating point Operations Per Second), Flop/s — количество операций над числами с плавающей точкой в секунду.

Поскольку наиболее высокопроизводительные компьютеры используются в основном для выполнения сложных научно-технических расчетов, более важны операции над вещественными

числами, представленными в формате с плавающей точкой, поэтому чаще используется FLOPS и кратные единицы:

- MFlop/s (Mega Floating point operations per second), 10^6 Flop/s;
- GFlop/s (Giga Floating point operations per second), 10^9 Flop/s;
- TFlop/s (Tera Floating point operations per second), 10^{12} Flop/s;
- PFlop/s (Peta Floating point operations per second), 10^{15} Flop/s;
- EFlop/s (Exa Floating point operations per second), 10^{18} Flop/s.

Разрядность операндов в формуле (1.2) не учитывается, но оказывает существенное влияние на значение производительности. При определении производительности необходимо указывать разрядность чисел, используемых в вычислениях. Один и тот же процессор может оперировать с различными типами данных, в том числе различной разрядности, и производительность для каждого из них будет различаться.

На данный момент для 64-разрядных вычислений пределом является эксафлопсная производительность. Его впервые достигла система с соответствующим названием («предел») — суперкомпьютер Frontier, размещенный в Окриджской национальной лаборатории министерства энергетики США. Frontier появился в рейтинге Top500 в июне 2022 г. Надо отметить, что прогнозы были более оптимистичные, в 2010–2011 гг. разными специалистами предполагалось появление эксафлопсного компьютера в 2018 г. Основные характеристики Frontier на момент появления в рейтинге были таковы [20]:

- производитель и архитектура — HPE Cray EX235a;
- процессоры AMD EPYC 64C 2GHz с общим количеством ядер 8730112;
- ускоритель — графический процессор AMD Instinct MI250X;
- реальная производительность по тесту HPL (High Performance Linpack) R_{max} — 1,102 EFlop/s;
- пиковая производительность R_{peak} — 1,68565 EFlop/s;
- потребляемая мощность 21,1 МВт.

Top500 — проект, в рамках которого два раза в год, в июне и ноябре, составляется и публикуется список 500 наиболее произво-

дительных компьютеров мира. На сайте Top500 (top500.org) представлено много полезной и интересной информации об архитектуре вычислительных систем. Самый последний компьютер в списке Top500 достиг реальной производительности в 1 PFlop/s в июне 2019 г.

Для определения реальной производительности компьютера при операциях над 64-разрядными числами с плавающей точкой в рейтинге Top500 используется тест HPL (High Performance Linpack). Тест Linpack заключается в решении плотной системы линейных уравнений AX = B размерности n×n, что является наиболее распространенной задачей в самых различных областях науки и техники. Его название происходит от пакета прикладных программ LINPACK, созданного еще в прошлом веке и предназначенного для решения задач линейной алгебры. Тест HPL является модификацией Linpack для параллельных вычислительных систем. Основной список Top500 упорядочивается именно по показателю реальной производительности R_{max} по тесту HPL.

В Top500 используются также дополнительные показатели производительности, связанные со спецификой теста HPL. Это максимальная размерность системы уравнений N_{max}, соответствующая R_{max}, а также размерность N1/2, при которой достигается половина производительности R_{max}.

Для Frontier N_{max} = 24 440 832.

Конечно, ни одна конкретная вычислительная задача не может объективно отражать реальную производительность ЭВМ, поэтому используются и другие тесты. В Top500 в дополнение к HPL используется тест HPCG (High-Performance Conjugate Gradient). Он реализует вычисления с разреженной матрицей, при которых существенное влияние на производительность оказывает пропускная способность канала памяти и внутреннего межсоединения. Объем данных для теста HPCG таков, что они не могут поместиться целиком в кэш-памяти, поэтому их постоянно приходится читать из основной памяти. Список, упорядоченный по показателю HPCG, конечно, отличает-

ся от основного списка. Так, например, тот же Frontier по тесту HPCG имеет реальную производительность 14,054 PFlop/s, на два порядка меньшую, чем по тесту HPL, и занимает уже не первое, а второе место. Это показывает, насколько значения реальной производительности зависят от вычислительной задачи и какое влияние на общую производительность компьютера оказывает канал взаимодействия с памятью, «бутылочное горлышко» фон-неймановской архитектуры.

С производительностью напрямую связан еще один общий показатель компьютера — потребляемая мощность. Чем больше производительность, тем больше потребляемая мощность. У суперкомпьютеров из Top500 высокая производительность, но и огромная потребляемая мощность. У Frontier, например, этот показатель равен 21,1 МВт. Поэтому эти компьютеры требуют серьезной системы охлаждения. Часто они охлаждаются водой, которая затем используется для отопления, т. е. система охлаждения суперкомпьютеров совмещена с системой отопления зданий, где они эксплуатируются.

Зависимость между производительностью и потребляемой мощностью прямая, но не линейная, поэтому используется еще один важный показатель — удельная производительность на единицу потребляемой мощности или энергоэффективность. В Top500 помимо основного списка и списка HPCG есть еще один — Green500, в котором ранжирование осуществляется по значению показателя энергоэффективности, определяемого как частное R_{max} по тесту HPL и потребляемой мощности. Frontier имеет энергоэффективность 52,227 GFlops/watts и в том же Green500 за ноябрь 2022 года занимает шестое место. Интересно отметить, что все компьютеры первой десятки этого Green500 используют процессоры архитектуры x86: Intel Xeon с ускорителями NVIDIA или AMD EPYC с ускорителями AMD Instinct. Суперкомпьютер Fugaku производства Fujitsu с процессорами архитектуры ARM, которые изначально разрабатывались по критерию минимального энергопотребления, занимает в этом рейтинге 43 место. Fugaku занял первое место в рейтинге Top500 за июнь 2020 г.

Технико-эксплуатационные показатели Fugaku на тот момент были таковы:

- пиковая производительность 513,8547 PFlop/s;
- реальная производительность R_{max} по тесту HPL 415,53 PFlop/s;
- потребляемая мощность 28,335 МВт;
- энергоэффективность 15 GFlops/watts.

При этом в вычислениях с 16-битными числами Fugaku демонстрировал эксафлопсную производительность.

Есть еще один рейтинг — Graph500 (graph500.org), в котором используется тест, в минимальной степени нагружающий вычислительную подсистему и в максимальной — коммуникационную. В его основе лежит поиск в большом ненаправленном графе. Единица измерения реальной производительности по этому тесту — TEPS (Traversed Edges Per Second). В рейтинге Graph500 также есть несколько списков:

- BFS (Breadth-First Search, поиск в ширину);
- SSSP (Single-Source Shortest Paths, вычисление кратчайших путей из одного источника);
- Green Graph 500.

Основной список — BFS. Первое место в списке BFS за ноябрь 2022 г. занимает Fugaku с показателем 10 2955 GTEPS. В списке SSSP Fugaku занимает третье место с показателем 2 126,45 GTEPS.

Помимо общих технико-экономических показателей ЭВМ существует множество частных показателей, характеризующих отдельные элементы компьютера: внутреннюю и внешнюю память, устройства ввода-вывода, каналы передачи информации:

- пропускная способность (Bandwidth) канала процессор-память;
- пропускная способность интерфейсов ввода-вывода;
- разрядность обрабатываемых данных и каналов передачи информации;
- характеристики оперативной памяти: емкость (объем, размер), тактовая частота, тайминги и др.;
- характеристики кэш-памяти: количество уровней и их объем, время доступа, коэффициент попаданий (Hit Rate) и др.;

– характеристики видеосистемы: тактовая частота графического процессора, объем и частота видеопамяти, пропускная способность интерфейса, характеристики монитора и т. д.;

– характеристики подсистемы хранения: состав и емкость накопителей, пропускная способность интерфейса и т. д.

Для самых высокопроизводительных систем хранения данных (СХД) также имеется отдельный рейтинг – IO500 [21]. В текущий список IO500 (июнь 2025 г.) входит сегмент СХД на базе решения RSC Storage onDemand (RSC Group или РСК — российская компания-разработчик решений в области высокопроизводительных вычислений, центров обработки данных и облачных платформ). Он поддерживает систему DAOS (Distributed Asynchronous Object Storage), лидирует среди российских систем хранения данных и занимает в рейтинге 139-е место. Другой сегмент СХД на базе файловой системы Lustre суперкомпьютера «Говорун» Объединенного института ядерных исследований (ОИЯИ) в Дубне размещается в этом рейтинге на 167-м месте. А в момент появления в рейтинге IO500 в июне 2018 г. СХД суперкомпьютера «Говорун» заняла 9-е место [22].

Классификационные признаки и основные типы вычислительных машин

Классификационные признаки — это в основном нечисловые характеристики (свойства) системы, по которым выделяется ряд ее типов или классов. Но и числовые характеристики (показатели, в том числе технико-эксплуатационные) тоже могут выступать в качестве классификационных признаков.

Как и технико-эксплуатационные показатели, классификационные признаки вычислительных машин могут быть общими, характеризующими их в целом, и частными, характеризующими их отдельные элементы.

Основные общие классификационные признаки и выделяемые по ним типы ЭВМ:

- *способ вычислений*:
  - алгоритмические;
  - неалгоритмические;
- *синхронизация* (наличие тактового генератора):
  - синхронные;
  - асинхронные;
- *тип обрабатываемой информации*:
  - аналоговые (АВМ);
  - цифровые (ЦВМ);
  - гибридные;
- *система счисления*:
  - двоичная;
  - десятичная и др.;
- *разрядность;*
- *организация памяти*:
  - принстонская архитектура;
  - гарвардская архитектура;
- *элементная база*:
  - механические;
  - электромеханические (на реле);
  - электронные:
    - на электронных лампах (первое поколение);
    - на дискретных полупроводниковых элементах (второе поколение);
    - на интегральных схемах (третье поколение);
    - на сверхбольших интегральных схемах, микропроцессорах (четвертое поколение);
  - оптические, фотонные, молекулярные и т. д.;
- *универсальность*:
  - универсальные;
  - специализированные;
  - проблемно-ориентированные;

− *производительность*:

− суперкомпьютеры;

− большие ЭВМ (Mainframe);

− мини-ЭВМ;

− микро-ЭВМ;

− *параллелизм* (см. п. 5.1):

− SISD (Single Instruction stream, Single Data stream), одиночный поток команд, одиночный поток данных;

− SIMD (Single Instruction stream, Multiple Data stream), одиночный поток команд, множественный поток данных;

− MISD (Multiple Instruction stream, Single Data stream), множественный поток команд, одиночный поток данных;

− MIMD (Multiple Instruction stream, Multiple Data stream), множественный поток команд, множественный поток данных.

Конечно, эти признаки неортогональны, т. е. связаны между собой, имеют «пересечения». Высокопроизводительные вычислительные системы, суперкомпьютеры — параллельные системы класса MIMD. Аналоговые компьютеры — неалгоритмические, химические — аналоговые и т. п.

Подавляющее большинство современных вычислительных машин — это фон-неймановские алгоритмические синхронные универсальные цифровые двоичные ЭВМ. В компьютере архитектуры фон Неймана и команды, и данные программы представляются в одной и той же числовой двоичной форме и хранятся вместе в одном и том же запоминающем устройстве. Фон Нейман был профессором математики в принстонском Институте перспективных исследований, поэтому фон-неймановскую архитектуру называют также принстонской. Другая возможность — команды и данные программы хранить в физически различных запоминающих устройствах: памяти команд и памяти данных. Она была реализована в Гарвардском университете в компьютере Mark I и называется гарвардской архитектурой. Но и в принстонской, и в гарвардской архитектуре основополагающей является

концепция хранимой программы — важнейший принцип фон-неймановской архитектуры. Основной признак вычислительных систем этого типа — отделение обрабатывающего и запоминающего устройств друг от друга и информационный канал между ними. Или несколько каналов — не принципиально. В параллельных вычислительных системах множество обрабатывающих и запоминающих устройств и еще больше вариантов организации взаимодействия между ними, но главный признак остается — разделение обрабатывающего и запоминающего устройств. Поэтому и самые высокопроизводительные параллельные вычислительные системы, суперкомпьютеры, по сути, остаются фон-неймановскими.

Информация в цифровых двоичных ЭВМ представлена в виде двоичных чисел, каждый бит которых представлен какой-либо физической величиной, принимающей только два хорошо различимых (практически с вероятностью единица) значения. Во внутренней части компьютера, в микросхемах и других электронных элементах эта величина — электрическое напряжение. Во внешних устройствах это могут быть и другие величины: в магнитных дисках и лентах — направление вектора намагниченности домена магнитной поверхности диска, в однократно записываемых оптических (лазерных) дисках — наличие выемок (ямок), в перезаписываемых (CD-RW, DVD-RW) — состояние ячейки (кристаллическое или аморфное).

Данный учебное пособие, соответственно, направлено на изучение компьютерных систем именно такого типа. Но нужно охарактеризовать и другие типы вычислительных устройств. Все они интересны с научной, теоретической точки зрения, многие из них имеют большие перспективы с практической, технической точки зрения.

По признаку элементной базы все современные ЭВМ являются электронными. Электромагнитное взаимодействие в настоящее время наиболее удобно для практического использования в разных технических устройствах, в том числе, вычислительных. Электроника сейчас повсюду. Однако помимо электромагнетизма есть и другие области:

оптика, квантовая физика, молекулярная физика, химия и т. д. В них проводится множество интересных исследований по разработке способов вычислений, альтернативных транзисторным. Существуют концепции и реализации таких вычислительных устройств, как:

- квантовый компьютер;
- химический компьютер;
- нейрокомпьютер;
- оптический компьютер;
- молекулярный компьютер;
- биологический компьютер;
- ДНК-компьютер.

Конечно, они тоже являются электронными в большей или меньшей степени. Без электроники обойтись нельзя, но надо помнить, что в современных электронных компьютерах основой схем, осуществляющих хранение и обработку информации, являются переключатели в виде полупроводниковых транзисторов. Такими переключателями могут быть и устройства, работающие по другим физическим принципам. В первых компьютерах использовались механические и электромеханические переключатели, затем электронные лампы. Сейчас в рамках множества различных концепций вычислений разрабатываются новые типы переключателей.

Также надо отметить, что в электронных компьютерах имеется множество неэлектронных элементов — оптических, механических и др. Без них тоже нельзя обойтись. В настоящее время развивается направление, связанное с микроминиатюризацией такого рода элементов. Оно связано с понятием MEMS (Micro-Electro Mechanical Systems, микроэлектромеханические системы). Это использование механических, пневматических и гидравлических элементов в электронике, компьютерной технике и автоматике на новом уровне, новом витке технологического развития. MEMS — это система из электронных, механических и других элементов на одной кремниевой подложке. Технология MEMS дает возможность построить полную

систему управления на одном кристалле с датчиками и исполнительными механизмами. ***К MEMS относятся:***

− микродатчики: например, датчики ускорения (акселерометры) в смартфонах для подсчета шагов и автомобилях для детектирования столкновения и выпуска защитных подушек, в дисководах для обнаружения ротационных перемещений, влияющих на позиционирование дорожки, датчики давления и т. д.;

− микроактуаторы;

− микродетали (гидравлические и пневмоклапаны, струйные сопла принтера, подвеска головки дисковода для ее точной подстройки, гироскопы);

− микроинструменты (скальпели и пинцеты);

− микромашины (моторы, насосы, турбины);

− микророботы;

− микролаборатории;

− системы инерциальной навигации;

− цифровые мультизеркальные устройства (DMD, Digital Micromirror Device) для цифровой обработки света (DLP, Digital Light Processing);

− электромеханическая энергонезависимая память.

Микроактуатор — исполнительный механизм миниатюрных размеров, устройство преобразования какой-либо энергии в механическую, т. е. движение. Например, вид микроактуатора — микродвигатель. Используются следующие ***методы активации в микроактуаторах:***

− электростатический;

− магнитный;

− пьезоэлектрический;

− гидравлический;

− тепловой.

Важной частью MEMS является технология создания трехмерных структур на кремниевой подложке. Основные способы создания трехмерных структур:

− объемная микрообработка, например, глубинное объемное травление;

− поверхностная микрообработка — последовательное наложение тонких пленок с удалением вспомогательных слоев;

− технология LIGA (Roentgen LIthography Galvanik Abformung) — комбинация рентгеновской литографии, гальванотехники и прессовки (формовки).

Таким образом, MEMS — это совокупность множества различных современных технологий: полупроводниковых, механических, оптических, гидравлических и др.

Далее кратко охарактеризуем некоторые специфичные типы вычислительных устройств, которые не распространены в настоящее время, но интересны с теоретической и практической точки зрения и могут получить развитие в будущем.

Такие признаки вычислительных устройств, как система счисления и разрядность относятся только к цифровым ЭВМ. О системах счисления много было сказано в главе 1.2. В настоящее время все ЭВМ являются двоичными, но мы помним примеры десятичных ЭВМ: Harvard Mark I и ENIAC. Были машины и на других системах счисления.

В 1959 г. в МГУ под руководством Николая Петровича Брусенцова была разработана очень интересная ЭВМ «Сетунь» на основе троичной системы счисления [10, 23]. Идея использования троичной системы восходит к задаче о выборе наилучшей системы гирь для взвешивания на рычажных весах, сформулированной Ленардо Пизанским (Фибоначчи). Он определил, что если при взвешивании груза гири можно ставить только на одну чашу весов, то минимальным будет набор гирь с весами, равными степеням двойки. Например, для взвешивания грузов массой до 31 грамма достаточно пяти гирь: 1, 2, 4, 8 и 16 граммов. Чтобы можно было взвешивать грузы до 63 граммов, потребуется еще одна гиря в 32 грамма. И т. д. Если же гири можно ставить на обе чаши весов, то оптимальным является набор гирь, веса которых равны степеням тройки: 1, 3, 9, 27, 81 и т. д.

Леонардо Пизанский бывал в Алжире и странах востока (Египте, Сирии, Византии), где изучал математику у арабских учителей и познакомился с математическими трудами на арабском языке. Он глубоко воспринял идею позиционной системы счисления и написал ряд трудов, главный из которых, «Книга абака» (1202 г.), очень способствовал распространению в Европе десятичной системы. Интересно, что после решения задачи о гирях Фибоначчи не пришел к идее двоичной и троичной системы счисления.

Через несколько сотен лет после этого троичная система оказалась полезной для выполнения денежных расчетов в Англии, где было много различных денежных единиц: фартинги, фунты, шиллинги, пенсы и т. д. — и при расчетах требуется преобразование одних единиц в другие и обратно. Такая задача встала перед Томасом Фаулером (Thomas Fowler), и для ее решения он в 1840 г. создал механическую троичную счетную машину. Она была сделана из дерева и имела внушительные размеры: длина — 6 футов, глубина — 3 фута, высота — 1 фут (примерно 1,8×0,9×0,3 м).

В ЭВМ «Сетунь» вычисления в троичной системе были реализованы на базе троичного ферритодиодного элемента. Но троичные цифры представлялись в виде пары бит, т. е. в машине использовался двухбитный троичный код. Единицами измерения были триты. Разрядность операндов в «Сетуни» составляла один трайт — шесть тритов (примерно 9,5 бит). Машина была проста, надежна и имела достаточную для своего времени производительность. Тактовая частота процессора составляла 200 кГц. Вычисления в троичной системе примерно в 1,5 раза быстрее двоичных. На Казанском заводе математических машин было изготовлено 46 ЭВМ «Сетунь», большинство из которых эксплуатировалось в различных институтах для решения научно-технических задач.

«Сетунь» была единственной в мире троичной ЭВМ. Однако перспективы использования троичной системы для вычислений сохраняются, поскольку использование троичной системы дает ряд преимуществ [23]. Еще в начале 1970-х гг. были разработаны и запа-

тентованы троичные триггеры. Имеются теоретические обоснования преимущества использования троичной системы в квантовых компьютерах, поскольку это упростит их конструкцию за счет уменьшения необходимого количества логических элементов.

Еще один перспективный способ вычислений — квантовые вычисления. Квантовый компьютер — это вычислительное устройство, в котором для передачи и обработки информации используются квантовые явления, в частности квантовая суперпозиция и квантовая запутанность. Наименьшей единицей информации в квантовом компьютере является не бит, а так называемый кубит (квантовый бит). Он имеет так называемые собственные состояния (значения): 0 и 1, как и обычный бит, но при этом его текущее состояние представляет собой их суперпозицию, определяемую волновой функцией, включающей так называемые амплитуды вероятностей двух его собственных состояний. Сумма квадратов амплитуд вероятностей равна единице. При измерении состояния кубита можно получить только собственное состояние — 0 или 1, каждое с соответствующей вероятностью, равной квадрату амплитуды. При измерении состояние кубита необратимо разрушается. Квантовая запутанность — это наличие корреляции между состояниями двух и более кубитов.

В настоящее время не существует работающего универсального квантового компьютера, но имеются специализированные экспериментальные и даже коммерческие системы, выполняющие фиксированные алгоритмы. Проводится множество интенсивных исследований в этой области в разных странах, в том числе в России, и регулярно появляются сообщения о новых разработках (рис. 2.1).



Рис. 2.1. Фрагмент действующего российского квантового компьютера (по материалам Первого канала, программа «Время» от 13.07.2023)

В настоящее время появилось понятие квантового процессора — Quantum Processing Unit (QPU). На конференциях появляются сообщения о создании экспериментальных образцов квантовых процессоров. Предполагается, что они будут интегрироваться с центральным, графическим и другими типами процессоров в составе компьютерных систем.

Несколько слов нужно сказать о таком классификационном признаке, как синхронизация.

Синхронные и асинхронные компьютеры выделяются по признаку наличия какого-либо тактирования, какого-либо способа задания дискретного времени, «часов» системы — генератора тактовых импульсов, вращающегося вала, как в Harvard Mark I, и т. п. Этот признак относится также только к цифровым ЭВМ. В настоящее время практически все компьютеры являются синхронными. Однако возможен и асинхронный компьютер, и соответственно, асинхронный процессор. В синхронных системах какие-нибудь действия и изменение состояния происходят в каждом такте. В асинхронных — только

при наступлении определенных событий: изменении входной информации, поступлении сигналов из внешней среды, действий пользователя и т. п. Как это всегда бывает в сложных системах, типы по одному и тому же признаку могут комбинироваться, т. е. значения одного признака могут быть разными на разных уровнях, в разных частях системы и в разные периоды ее функционирования. Например, как уже было сказано, все современные компьютеры синхронные. Но процессор по определенной команде (halt, break и т. п.) может перейти в состояние останова. Генератор может продолжать работу, но процессор не изменяет своего состояния и ничего не делает, кроме наблюдения за значением некоторых сигналов, по которым он и может выйти из состояния останова и выполнить необходимые действия, но уже в обычном синхронном режиме. Например, процессор может выйти из состояния останова по сигналу прерывания, который всегда является асинхронным, и выполнить специальную подпрограмму — обработчик прерывания. Генератор тактовых импульсов тоже может изменять свое поведение, снижать частоту при уменьшении вычислительной нагрузки или вовсе останавливаться. В большинстве вычислительных систем имеется множество генераторов и множество тактируемых устройств. В некоторых режимах отключаются все генераторы и устройства кроме некоторых, которые должны среагировать на внешний сигнал и вывести остальные элементы из состояния останова.

Другой пример — память. Она тоже может быть асинхронной или синхронной. Пример асинхронной памяти — EDO (Extended Data Out). Она не требует тактирования. Использовалась с процессором Pentium. Потом появилась синхронная память — SDRAM (Synchronous Dynamic Random Access Memory). В компьютере есть отдельный генератор, который формирует тактовые импульсы, поступающие на ее соответствующие входы.

Есть интерфейсы ввода-вывода, которые могут работать в синхронном или асинхронном режиме в зависимости от настройки. Это, например, USART (Universal Synchronous-Asynchronous Reciver-

Transmitter, универсальный синхронно-асинхронный приемопередатчик).

Память и интерфейсы ввода-вывода будут подробнее рассмотрены в соответствующих разделах.

Аналоговые, химические и нейронные компьютеры

В большинстве компьютеров, как уже было сказано, информация обрабатывается по алгоритму. Но есть и другие, неалгоритмические способы вычислений и соответствующие компьютеры:

− аналоговые;

− химические;

− нейронные или нейроморфные.

Аналоговые компьютеры выделяются в первую очередь по типу обрабатываемой информации. Но по другому классификационному признаку — способу вычислений — они являются неалгоритмическими. По признаку элементной базы они могут быть механическими, электромеханическими, электронными, пневматическими, гидравлическими. Конечно, они могут иметь различную степень универсальности.

В начале развития вычислительной техники внимание в равной мере уделялось созданию как цифровых, так и аналоговых вычислительных машин. Использовались термины: АВМ — аналоговая вычислительная машина, и ЦВМ — цифровая вычислительная машина. Были и неэлектронные цифровые вычислительные машины (Z1, Z2, Z3, Harvard Mark I). Поэтому использовался также термин ЦЭВМ — цифровая электронная вычислительная машина. Со временем подавляющее большинство ЭВМ стало цифровыми, неэлектронные цифровые машины исчезли, и в аббревиатуре ЦЭВМ первая буква пропала. Под термином ЭВМ понимается именно цифровой компьютер. АВМ — аналоговая вычислительная машина. История аналоговой вычислительной техники не менее интересна, чем цифровой, но здесь

невозможно уделить ей много места. Аналоговые вычислительные машины создавал, например, Брук. В 1935 г. он разработал такую АВМ, как расчетный стол переменного тока, использовавшийся для расчета и моделирования энергетических систем. Затем под его руководством были созданы механический, а позднее электронный интегратор дифференциальных уравнений ЭДА (электронный дифференциальный анализатор) [10].

В АВМ информация представлена непрерывными физическими величинами. В настоящее время большинство АВМ, конечно, электронные и такой физической величиной является, соответственно, электрическое напряжение. В АВМ используется полностью другой принцип вычислений, чем в ЭВМ. Хранимая программа в них отсутствует. Соответственно, нет и алгоритма. **Основные элементы АВМ** — это электронные блоки, каждый из которых имеет определенную функцию преобразования информации, входного сигнала в выходной — интегрирование, дифференцирование и т. п. Имеются блоки для операций сложения, вычитания, умножения. Соединения блоков между собой непостоянные. Их можно задавать с помощью переключателей, формируя определенную аппаратную схему для необходимых вычислений. Инициализация АВМ происходит путем задания требуемой аппаратной структуры. После этого кнопкой «Старт» запускается процесс, в ходе которого изменяются напряжения в тех или иных точках схемы, значения которых можно измерить в любой момент времени. Они и представляют собой вычисляемую информацию.

В дальнейшем АВМ получили меньшее развитие, чем цифровые, по причине меньшей точности и невозможности полностью избавиться от шума, который приводит к искажению сигнала. Но для решения некоторых задач, например, интегрирования систем нелинейных дифференциальных уравнений высокого порядка в реальном времени, они незаменимы.

Существуют гибридные вычислительные машины. В них присутствуют как цифровые, так и аналоговые блоки, соответственно, они сочетают преимущества как тех, так и других.

Химический компьютер, по-другому реакционно-диффузионный компьютер или компьютер Белоусова — Жаботинского (BZ-computer), — это вычислительная система, в которой данные представлены концентрациями различных химических веществ, а вычисления выполняются посредством химических реакций. К созданию концепции химических вычислений привело открытие, которое в 1951 г. сделал Борис Павлович Белоусов. Он обнаружил автоколебательную химическую реакцию, т. е. такую, в которой происходило периодическое изменение концентраций веществ, участвующих в реакции. До этого считалось, что колебания в химических процессах невозможны, и Белоусов даже не смог опубликовать результаты своего исследования ни в одном журнале. Только спустя несколько лет это удалось сделать в незначительном сборнике. К счастью, эта статья получила известность и стала одной из самых цитируемых. Открытие Белоусова дало толчок к развитию синергетики, теории динамических систем, детерминированного хаоса и концепции химических вычислений. Посмертно Белоусову была присуждена Ленинская премия.

Впоследствии эту реакцию исследовал Анатолий Маркович Жаботинский, он обнаружил пространственное распределение концентраций, т. е. волны, предложил объяснение и создал математическую модель реакции. Наличие колебаний и волн означает возможность передачи и обработки информации.

Начиная с конца восьмидесятых годов прошлого века до настоящего времени было представлено множество разработок, демонстрирующих возможность вычислений с использованием реакции Белоусова — Жаботинского.

Еще один тип неалгоритмического вычислительного устройства — нейрокомпьютер. Он представляет собой аппаратную реализацию нейронной сети — системы из нейронов и аксонов, их соеди-

нителей, моделирующей работу человеческого мозга. Но нейронная сеть может быть реализована программно на ЭВМ фон-неймановской архитектуры. Такие реализации существуют с середины прошлого века, но только сейчас они вышли на качественно новый уровень с использованием высокопроизводительных вычислительных систем — суперкомпьютеров. Большое распространение сейчас получили искусственные нейронные сети (Artificial Neural Network, ANN), использующие так называемые большие языковые модели (Large Language Model, LLM), имитирующие мыслительную и разговорную деятельность человека и позиционирующиеся как системы генеративного искусственного интеллекта (Generative Artificial Intelligence, Generative AI, GenAI, GAI). Это, например, GigaChat и Kandinsky (Кандинский), разработанные Сбером, ChatGPT (Generative Pre-trained Transformer, генеративный предобученный трансформер), разработанный компанией OpenAI, DeepSeek от одноименной компании, YandexGPT — нейросеть от Яндекса, использующаяся Алисой. Все они требуют огромных вычислительных затрат, поскольку, как уже было сказано, работают на суперкомпьютерах, которые потребляют огромное количество энергии и соответственно, имеют низкую энергоэффективность. Поэтому больший интерес представляют не программные, а аппаратные реализации нейронных сетей — нейрокомпьютеры и нейронные или нейроморфные процессоры.

Первым нейрокомпьютером считается MARK 1 (еще один MARK 1), построенный Франком Розенблаттом (Frank Rosenblatt) в 1958 г. MARK 1 был основан на перцептроне, одной из первых моделей нейросетей, первоначально программно смоделированном на компьютере IBM 704. Он мог распознавать некоторые буквы английского алфавита. Велись подобные исследования и разработки и в нашей стране [11]. С появлением интегральных схем нейросети стали аппаратно реализовываться уже на их основе. В качестве примера можно привести нейропроцессоры, разработанные IBM в рамках программы SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) агентства DARPA (Defense Advanced Research

Projects Agency, Агентство перспективных исследовательских проектов Министерства обороны США). Они были представлены 18 августа 2011 г. под названием когнитивные вычислительные чипы (cognizers). Было изготовлено два действующих прототипа, разработанных по 45-нм технологии SOI-CMOS, включающих по 256 нейронов. Один из прототипов содержал 262 144 программируемых, другой — 65 536 самообучающихся синапсов. На действующих образцах был продемонстрирован ряд приложений: модель мозга кошки, навигация, машинное зрение, распознавание образов, ассоциативная память, классификация.

Руководитель исследовательского проекта IBM Research Дхармендра Модха так охарактеризовал новый процессор: «Наш чип представляет собой явный отказ от традиционной архитектуры фон Неймана. Все функции хранения объединены с функциями обработки, создавая в некотором роде социальную сеть нейронов, программное обеспечение которой хранится в синапсах» [24].

В 2013 г. компания Qualcomm представила процессор Zeroth, соответствующий классу мобильных процессоров, использующихся в смартфонах, отличительной особенностью которого было наличие нейропроцессора в качестве одного из специализированных ядер (Neural Processing Unit, NPU). На прототипе Zeroth был создан обучающийся робот.

В середине 2016 г. IBM создала искусственный нейрон из сплава германия, сурьмы и теллура (GeSbTe, или GST), который может находиться в двух состояниях, кристаллическом и аморфном, и переключаться между ними под воздействием тепла. В кристаллическом состоянии GST является проводником, а в аморфном — диэлектриком.

В настоящее время существует множество действующих образцов нейроморфных процессоров IBM, Intel и других компаний, в том числе российской компании «Мотив Нейроморфные Технологии». Отечественный нейроморфный процессор имеет название «Алтай» (рис. 2.2).



Рис. 2.2. Модуль с восемью нейроморфными процессорами «Алтай»

«Алтай» имеет характеристики, не уступающие зарубежным образцам:

− 256 нейроядер, 131 072 нейронов, 67 млн. синапсов;

− площадь кристалла 64 мм²;

− энергопотребление 0,1–0,5 Вт;

− техпроцесс 28 нм;

− производительность около 67 млрд. действий в секунду или 2 200 кадров в секунду.

При этом его важным преимуществом по сравнению с зарубежными аналогами является возможность неограниченной масштабируемости нейронной сети, создаваемой на его основе.

По производительности «Алтай» сравним с современными графическими процессорами при существенно меньшем энергопотреблении, что обусловлено использованием импульсных сигналов.

Оптические и фотонные компьютеры

Идея оптического компьютера заключается в том, что для представления, передачи и обработки информации в нем используются световые сигналы. Но свет — это электромагнитная волна, и хотя оптика — это отдельный раздел физики, без электроники ни в каком современном техническом устройстве обойтись нельзя. И электроника — комплексная область, в которую входит множество разделов: вакуумная электроника, полупроводниковая электроника, наноэлектроника, квантовая электроника, оптоэлектроника и т. д. Чисто оптических устройств и элементов немного: линза, зеркало, призма, диафрагма, световод, соединительные устройства, муфты и разветвители. Для реализации вычислительной системы только оптическими устройствами и приборами, без использования электронных элементов, обойтись невозможно. Полностью оптических компьюте-

ров не существует даже в виде простейших опытных образцов. Имеются примеры реализации и аналоговых, и цифровых оптических компьютеров, но большинство элементов в них все равно электронные. Световые сигналы необходимо генерировать с помощью полупроводниковых лазеров или светодиодов. А они являются электронными элементами.

В настоящее время использование чисто оптических элементов дает существенные преимущества только в составе электронных компьютеров и компьютерных сетей. Для передачи информации широко используются волоконно-оптические линии связи. Пропускная способность волоконно-оптических кабелей намного больше, чем электрических. Еще несколько лет назад они использовались только в магистральных каналах, но сейчас имеются уже во многих квартирах и частных домах, где доступ в Интернет осуществляется с использованием технологии GPON (Gigabit-capable Passive Optical Network). Как видно из названия, пропускная способность линии связи составляет гигабиты в секунду. Слово «пассивный» в названии технологии означает, что на участке сети от терминала поставщика услуг Интернета до абонентского терминала типа ONT (Optical Network Terminal) нет промежуточных электронных устройств: коммутаторов и переключателей, — которые в данной терминологии являются активными, а только пассивные соединительные устройства: оптические кроссы с разветвителями, оптические муфты и т. п. Терминал ONT — это маршрутизатор (router), к разъему которого со стороны поставщика подключается не электрический кабель, а оптоволоконный. А оконечные устройства в доме: компьютеры, телевизоры, планшеты, смартфоны — подключаются к терминалу ONT, как и к любому маршрутизатору, либо по Wi-Fi, либо обычными электрическими кабелями типа «витая пара».

Ведутся также разработки по реализации оптических каналов передачи данных и оптических коммутаторов непосредственно на кремниевой подложке, т. е. в составе полупроводниковой микросхемы.

Еще один пример использования оптики в компьютерах — хранение информации на оптических (лазерных) дисках. Правда, сейчас они уже выходят из употребления. Зато широко используются такие оптоэлектронные элементы, как оптопары (оптроны) и оптореле.

Оптопары применяются для гальванической развязки различных частей электронных устройств. Они представляют собой заключенные в одном корпусе излучающий элемент, чаще всего светодиод, и фотоэлемент — фототранзистор, фотодиод, фототиристор или фоторезистор. Разновидностью оптрона является оптореле (твердотельное реле). Это оптрон с полевым транзистором или фотосимистором. Они используются для коммутации сигналов, в том числе для управления мощными устройствами: нагревателями и двигателями.

Наиболее распространенные устройства ввода — мыши — являются в настоящее время оптическими или лазерными. Да и механические мыши, которые использовались ранее, имели в своем составе оптоэлектронные элементы — пары «светодиод — фотодиод», разделенные вращающимся при перемещении мыши диском с прорезями для прерывания светового потока от светодиода к фотодиоду и определения таким образом параметров перемещения мыши.

Конечно, различные типы компьютерных дисплеев: на электронно-лучевых трубках, жидкокристаллические, светодиодные — это тоже оптоэлектронные устройства.

Итак, оптика в электронной вычислительной технике используется повсеместно — для передачи, хранения, отображения информации и т. д. Но к чисто оптическим элементам можно отнести только волоконно-оптические кабели и некоторые соединительные устройства: оптические муфты, кроссы и т. п. Большинство типов элементов, использующих оптические эффекты, являются оптоэлектронными.

Для полностью оптических вычислений необходимо реализовать логические элементы на основе оптических (фотонных) переключателей или оптических транзисторов, в которых свет включается или выключается светом, т.е. прохождением фотонов так же управляют фотоны. Предлагается множество различных вариантов реализации оптических транзисторов, некоторые из них продемонстрированы экспериментально, но по многим параметрам (эффективность, сложность, стоимость, размеры) они в настоящее время проигрывают электронным. Основная проблема здесь — снижение мощности входного переключающего светового сигнала. В 2020 г. в Сколковском институте науки и технологий был изобретен однофотонный переключатель, который кардинально решает эту проблему [25]. На

изобретение был выдан патент «Способ переключения макроскопического состояния поляритонов при помощи одного фотона». До этого в том же институте был разработан поляритонный транзистор, в котором состояние с нуля на единицу переключалось при помощи сигнала мощностью порядка 10–100 тыс. фотонов. Но, конечно, однофотонный переключатель еще нужно воплотить в коммерчески оправданном техническом устройстве. Для вычислительных устройств с такими переключателями стали использовать термин «Фотонный компьютер». Вычислительная система такого типа разрабатывается сейчас в лаборатории фотонных вычислительных устройств Национального центра физики и математики в Сарове. Предполагается, что ее производительность будет на четыре порядка выше, чем у современных самых мощных суперкомпьютеров (рис. 2.3) [26].



Рис. 2.3. Прогнозируемая производительность фотонного компьютера

В заключение этого параграфа надо отметить, что фотоны используются для вычислений и в квантовых компьютерах, а также для передачи информации в квантовых сетях. Но там, в отличие от фотонных компьютеров и обычных волоконно-оптических линий связи, используются квантовые явления: суперпозиция и запутанность. В Национальном центре физики и математики ведется работа и над созданием квантовых систем.

Молекулярный компьютер. Мемристоры

Молекулярный компьютер — вычислительное устройство, реализованное с использованием молекулярных технологий (нанотехнологий), в котором вычисления осуществляются на молекулярном уровне, т. е. переключатели реализованы из отдельных молекул. Молекулы могут быть органическими, в частности молекулами ДНК, поэтому биологический компьютер или ДНК-компьютер — это разновидности молекулярного компьютера. Вообще-то, любой организм, в том числе человеческий, является биологическим компьютером. Вспомним название главной книги Винера — «Кибернетика, или управление и связь в животном и машине». В организме ДНК — это постоянная память, мозг неразрывно объединяет устройство управления, арифметико-логическое устройство и оперативную память, устройства ввода-вывода — это органы чувств, речи и движения. Однако здесь речь идет не о естественных, а об искусственных, технических системах.

Основная идея молекулярных вычислений состоит в следующем. При изготовлении транзисторных переключателей в микросхемах в рамках сложного многоэтапного технологического процесса — фотолитографии, травления и т. д. — происходит формирование необходимых структур на кристалле кремния путем удаления ненужных областей в объеме кристалла, а затем добавления нужных на макроуровне посредством напыления, ионной имплантации и т. п. Можно провести аналогию с ваянием: берется глыба мрамора и отсе-

кается все ненужное. В молекулярных компьютерах, наоборот, происходит сборка молекул для формирования структур с заданными свойствами. В этом состоит сущность нанотехнологий, поэтому здесь используется также термин «нанокомпьютер».

Проводится множество исследований в направлении создания молекулярных нанокомпьютеров. Существуют концепции вычислительных систем такого рода и экспериментальные реализации их отдельных элементов. Наиболее интересными и перспективными являются разработки, связанные с понятиями crossbar latch (поперечной защелки, координатной защелки) и мемристора. Кроссбар *(crossbar)* — это матричная структура элементов, в данном контексте проводников, расположенных перпендикулярно друг другу и в точках пересечения связанных тем или иным способом. То есть пересечения связаны не простым электрическим контактом, а некоторым переключателем — диодным, транзисторным, молекулярным двухпозиционным или мемристорным. Мемристор *(memristor, memory resistor)* — резистор с «памятью», резистор, сопротивление которого изменяется в зависимости от приложенного напряжения и сохраняется при снятии напряжения. Идею мемристора как четвертого базового элемента электрических цепей, наряду с конденсатором, резистором и катушкой индуктивности, выдвинул в 1971 г. Леон Чуа (Leon Ong Chua) [27]. Сочетание кроссбара и мемристора, реализованных на молекулярном уровне, предоставляет большие возможности для обработки и хранения информации. Разработки в этой области проводятся в ведущих исследовательских организациях многих стран, в том числе России. Одними из первых были результаты, полученные в компании Hewlett-Packard.

Hewlett-Packard представила crossbar latch в начале двухтысячных годов. Патент Molecular crossbar latch на данное изобретение был выдан в 2003 г. на имя Филиппа Кукеса (Philip Kuekes) [28]. Устройство и работа crossbar latch были описаны в журнале прикладной физики в 2005 г. [29]. Элементарная структура crossbar latch имеет размеры порядка единиц нанометров и является альтернативой транзи-

стору. В лаборатории по квантовым исследованиям (Quantum Science Research) компании Hewlett-Packard было продемонстрировано, что crossbar latch может выполнять логические операции и хранить информацию.

Crossbar latch представляет собой молекулярный переключатель, состоящий из сигнальной линии, пересеченной двумя линиями управления. В точке пересечения они не соприкасаются, а проходят, как мосты, один над другим. Сигнальная линия лежит снизу, а линии управления проходят над ней. При этом в месте пересечения с верхних проводников на нижний спускаются молекулярные цепочки, соединяющие их между собой.

Работа crossbar latch состоит в том, что электрические импульсы замыкают или размыкают цепочки в противофазе, т. е. когда замкнуто одно соединение, разомкнуто другое и наоборот. Это устойчивое состояние и представляет собой 0 или 1.

Еще большие возможности появляются, если в кроссбаре использовать не двухпозиционные переключатели, а мемристоры. Как электронный элемент мемристор был создан в 2008 г. в той же компании Hewlett-Packard коллективом разработчиков во главе со Стэнли Уильямсом (Richard Stanley Williams). Он представлял собой твердотельный элемент на основе тонкой (5 нм) двухслойной пленки диоксида титана. Один из слоев пленки был обеднен кислородом, и кислородные вакансии могли перемещаться между слоями под действием приложенного напряжения, что приводило к изменению сопротивления элемента с сохранением его значения после снятия напряжения.

Мемристор на основе диоксида титана был создан также в Тюменском государственном университете в 2012 г. В других организациях были созданы мемристоры и на других основах: оксиде кремния, полимерах и т. д. В Институте физики полупроводников им. А.В. Ржанова Сибирского отделения РАН в сотрудничестве с некоторыми другими организациями был создан мемристор на основе

графена [30]. Разрабатываются мемристоры и на основе органических молекул.

На основе кроссбара и мемристоров можно создавать логические элементы и запоминающие устройства [31, 32]. А затем на основе таких мемристорных элементов и состоящих из них узлов — вычислительные системы. Одну из первых попыток создания компьютера на основе мемристоров предприняла та же самая Hewlett-Packard.

Концепция такого компьютера с названием The Machine была представлена в июне 2014 г. на конференции HP Discover в Лас-Вегасе. The Machine по технико-эксплуатационным показателям сравнивалась с суперкомпьютерной системой K computer компании Fujitsu (рис. 2.4).

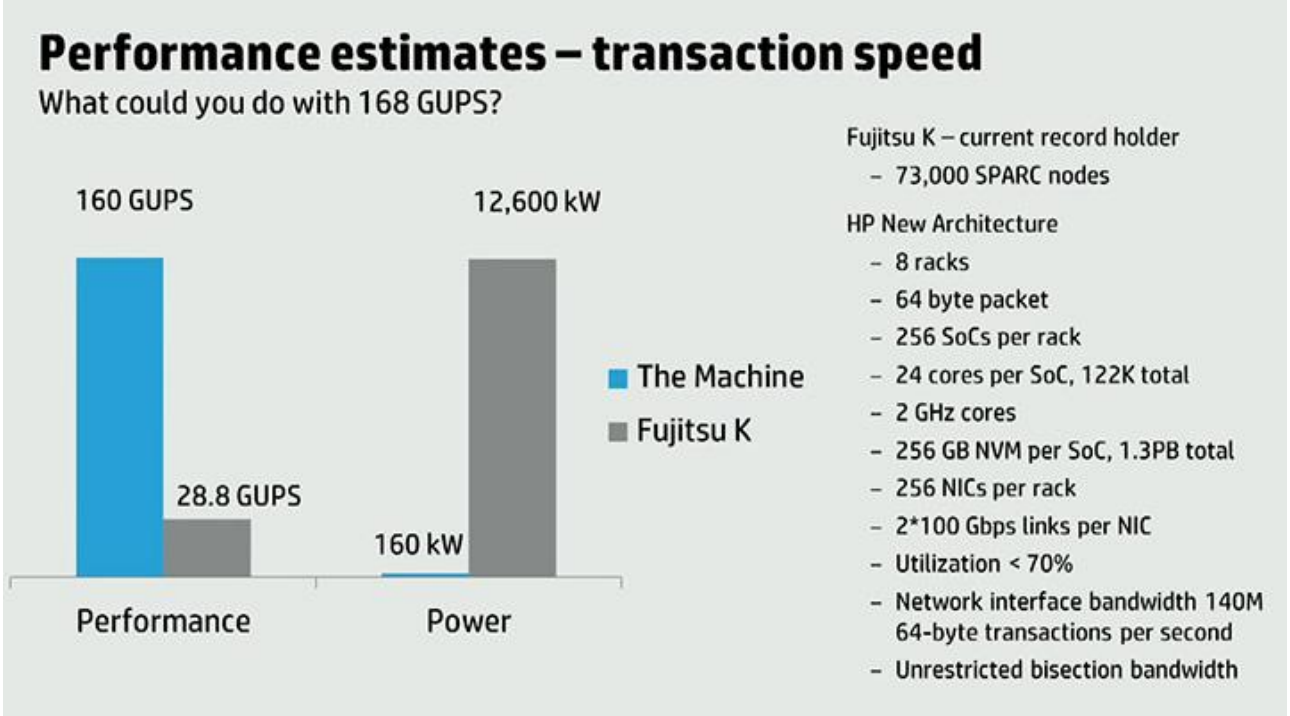


Рис. 2.4. Сравнение The Machine и Fujitsu K Computer

K computer занял первое место в 37-м рейтинге Top500 (июнь 2011 г.). Это был первый компьютер, преодолевший рубеж производительности в 10 PFlop/s. Правда, произошло это после модернизации уже в следующем, 38-м, ноябрьском рейтинге Top500. Его реальная

производительность по тесту Linpack составила 10,51 PFlop/s. Затем он еще долгое время оставался в первой десятке. Производительность The Machine должна была более, чем в пять раз, превысить производительность K computer, а энергопотребление при этом должно было быть меньше почти в 80 раз.

Основная особенность The Machine — единое запоминающее устройство на основе мемристоров, объединяющее функции оперативной, постоянной и внешней памяти. Коммуникационная подсистема предполагалась полностью оптической.

Работающий прототип The Machine должен был быть выпущен в 2016 г. Но этого не произошло. В мае 2017 г. появилось сообщение о реальной демонстрации прототипа The Machine. При этом сообщались следующие характеристики: 1 280 ядер Cavium ARM (сорок 32-ядерных процессоров), разделяемая память объемом 160 Тбайт, оптические межсоединения с контроллером на основе программируемой логической матрицы, масштабируемость до 4 096 Йоттабайт разделяемой памяти. Однако эти характеристики были далеки от заявленных в 2014 г. О мемристорах уже не упоминалось.

Использование мемристоров в компьютерах фон-неймановской архитектуры коммерчески неоправданно. Большие перспективы применение мемристоров имеет для создания нейроморфных вычислительных устройств. В отличие от фон-неймановской вычислительной системы, где обрабатывающее и запоминающее устройство разделены и связаны каналом передачи информации, мозг — это одновременно и обрабатывающее, и запоминающее устройство. Мемристор позволяет создать подобное мозгу техническое вычислительное устройство, в котором обработка и хранение информации совмещены [33, 34, 35]. Из мемристора можно сделать синапс нейроморфного компьютера, поскольку фукционирование мемристора аналогично работе биологического синапса: его пропускная способность зависит от интенсивности входного сигнала и может принимать множество различных значений. То есть мемристор, как и резистор, является аналоговым устройством. Таким образом, на новом витке спирали в

виде мемристоров аналоговая техника возвращается в вычислительную технику.

Исследования в области разработки нейроморфных вычислительных систем на основе мемристоров ведутся во многих научных организациях, в том числе в Курчатовском институте. Там созданы нанокомпозиты, массивы мемристоров на их основе и прототипы нейроморфных систем, демонстрирующие высокую производительность и низкое энергопотребеление [36, 37]. Мемристоры имеют малые размеры, и соответственно, плотность их размещения на единице площади достаточно высока. Они потребляют мало энергии, поскольку она не требуется им для поддержания текущего состояния, а необходима лишь для его изменения, т. е. в мемристорных системах можно использовать импульсные сигналы. А информационная емкость мемристоров больше одного бита, так как число состояний, в которых они могут находиться, больше двух.

2.2. ОБЩАЯ СТРУКТУРА ЭВМ

Структуризация ЭВМ

Можно провести следующие основные деления ЭВМ на части, аналогичные делениям, принятым для человеческого организма:

1. Аппаратная (твердая, hard) часть — программная (мягкая, soft) часть (скелет — мягкие ткани).
2. Подсистемы (системы):

– математическая — вычисления с плавающей точкой;
– физическая — физические расчеты;
– сигнальная (ММХ) — обработка сигналов;
– звуковая (аудиосистема) — обработка звуковых сигналов;
– графическая (видеосистема) — обработка видеоинформации;
– коммуникационная — обработка пакетов;
– безопасности;

– прерываний;

– прямого доступа к памяти;

– ввода-вывода;

– ввода информации с клавиатуры и т. д.

3. Элементы (органы):

– *центральные* (внутренние):

  – процессор (обрабатывающее устройство);

  – память (запоминающее устройство);

– *периферийные* (внешние) — устройства ввода-вывода.

Процессор и память — это «мозг» компьютера, устройства ввода-вывода — «органы чувств и речи».

Надо отметить, что многие подсистемы, устройства ввода-вывода и их интерфейсы имеют главный элемент, называемый контроллером или процессором:

– *математическая* — устройство с плавающей точкой (Floating Point Unit, FPU);

– *физическая* — физический процессор (Physics Processing Unit, PPU);

– *сигнальная* — сигнальный (Digital Signal Processor, DSP) или медийный процессор, устройство MMX;

– *звуковая* — аудиоконтроллер, в состав которого входит DSP;

– *графическая* — графический процессор (Graphics Processing Unit, GPU);

– *коммуникационная* — сетевой процессор (Network Processor, NP);

– *безопасности* — криптографический процессор (Platform Security Processor, PSP, или Trusted Platform Module, TPM);

– *прерываний* — контроллер прерываний (Programmable Interrupt Controller, PIC);

– *прямого доступа к памяти* — контроллер ПДП (Direct Memory Access Controller, DMA Controller);

– *ввода-вывода* — контроллер ввода-вывода;

– *ввода информации с клавиатуры* — контроллер клавиатуры и микро-ЭВМ клавиатуры.

Это приводит к тому, что любая ЭВМ имеет иерархическую структуру (см. п. 1.3).

Можно провести также и еще одно деление — *деление на уровни* [38]:

Уровень 5 — язык высокого уровня.

Уровень 4 — уровень языка ассемблера.

Уровень 3 — уровень операционной системы.

Уровень 2 — уровень архитектуры команд.

Уровень 1 — уровень микроархитектуры.

Уровень 0 — цифровой логический уровень.

Уровень −1 — уровень физических устройств.

Уровень физических устройств — это электронные элементы самого нижнего уровня: резисторы, конденсаторы, транзисторы и т. п.

Цифровой логический уровень — это логические элементы, которые строятся из транзисторов с использованием резисторов и конденсаторов. А из них в ЭВМ строится все остальное (см. п. 1.2).

Уровень микроархитектуры — это детали устройства ядра процессора: операционные, исполнительные блоки и блоки, обеспечивающие их работой — доставку и распределение команд и т. п.

В компьютерной программе первые три уровня никак не проявляются. Это чисто аппаратная картина. На следующих уровнях уже начинает проявляться программный аспект. Уровень архитектуры команд (см. п. 1.3, 3.3) — это система команд процессора, какие функциональные возможности разработчик заложил в процессор с точки зрения их непосредственного использования в программе.

Уровень языка ассемблера соответствует уровню архитектуры команд, поскольку, как уже было сказано в первом разделе, команды, записанные на языке ассемблера, отображаются в команды процессора. Но ассемблер добавляет некоторые новые краски. Одна конкретная модель процессора — это одно конкретное множество команд. Но ассемблеров для одной модели процессора существует множество.

Они различаются синтаксисом, некоторыми удобствами для программиста по оформлению структуры программы и т. п. В программе на ассемблере помимо команд присутствуют директивы (см. п. 1.3). Уровень системы команд и уровень языка ассемблера — это граница между аппаратным и программным обеспечением, но такая граница, которая не разделяет, а наоборот, соединяет их. Это место их соприкосновения и взаимодействия.

Уровень операционной системы — это уже первый чисто программный уровень. Операционная система, помимо всего прочего, реализует так называемый HAL (Hardware Abstraction Layer, уровень абстракции аппаратного обеспечения) — интерфейс между аппаратными средствами и запускаемыми операционной системой программами. Так что программам уже не нужно обращаться напрямую к аппаратуре, а можно делать это опосредованно через соответствующие элементы операционной системы. Конечно, в этом учебном пособии мы не будем пользоваться этими возможностями операционной системы, за некоторыми исключениями, поскольку здесь перед нами стоит другая цель — самим разобраться в том, как в программе происходит работа с аппаратурой непосредственно.

Наконец, самый высокий уровень — это **уровень языка высокого уровня**. Чаще всего программист пишет программу на языке высокого уровня с использованием соответствующих инструментальных средств, которые работают в соответствующей операционной системе. Соответственно, в программе появляются особенности, связанные с языком, системой программирования и операционной системой.

Общая схема структуры ЭВМ

Рассмотрим схему структуры ЭВМ, более детальную по сравнению с рассмотренной ранее схемой архитектуры фон Неймана (рис. 2.5). Эта схема является общей для всех ЭВМ этой архитектуры.

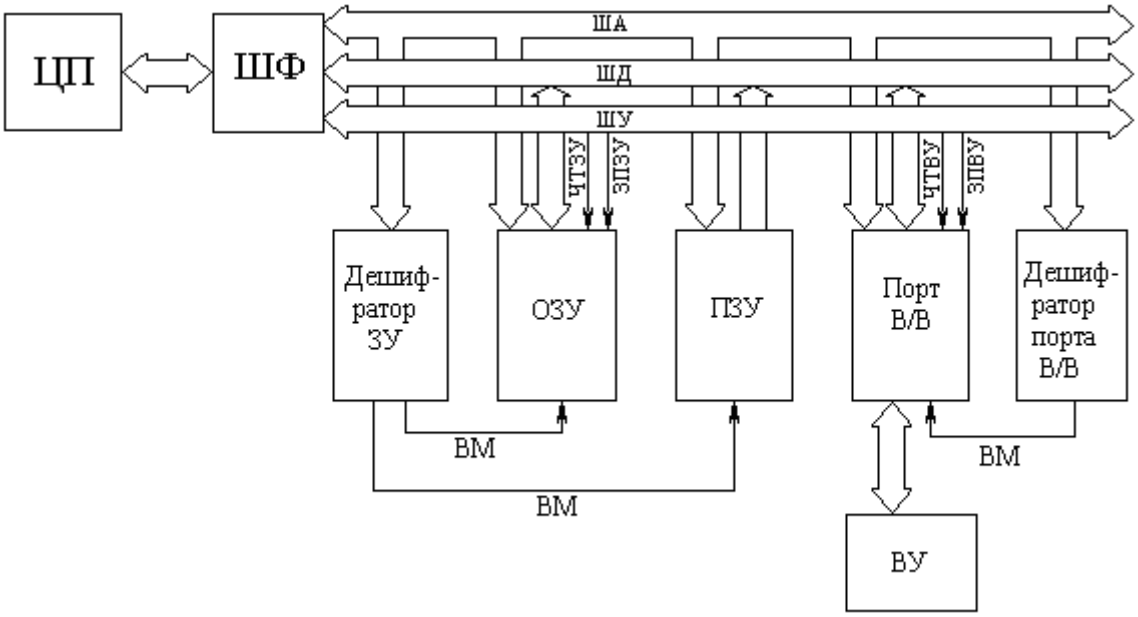


Рис. 2.5. Общая схема структуры ЭВМ

Представленные на ней элементы и связи между ними присутствуют в любой ЭВМ. На схеме не представлены специфичные элементы, которыми ЭВМ могут различаться: специализированные процессоры (сопроцессоры), контроллеры прерываний, прямого доступа к памяти и др. Также не представлен общий для всех синхронных компьютеров элемент — генератор тактовых импульсов. Он подразумевается по умолчанию, к тому же он может входить в состав процессора.

Элементы, изображенные на схеме:

− ЦП — центральный процессор (CPU, Central Processing Unit);

− ШФ — шинный формирователь (в самом широком смысле этого слова);

− ЗУ — запоминающее устройство (M, Memory);

− ОЗУ — оперативное запоминающее устройство (RAM, Random Access Memory);

− ПЗУ — постоянное запоминающее устройство (ROM, Read Only Memory);

− В/В — ввод/вывод (I/O, Input/Output);

− ВУ — внешнее устройство (устройство ввода-вывода), например, клавиатура, дисплей, датчики, исполнительные механизмы и т. п.;

− ША — шина адреса (AB, Address Bus);

− ШД — шина данных (DB, Data Bus);

− ШУ — шина управления (CB, Control Bus);

− ЧТЗУ (ВУ) — чтение ЗУ (ВУ) (R, RD — Read, RC — Receive);

− ЗПЗУ (ВУ) — запись ЗУ (ВУ) (W, WR — Write, TR — Transmit);

− ВМ — выбор микросхемы (CS, Crystal Select).

На схеме элементы ЭВМ распределены по трем уровням:

1. Процессор.

2. Память и другие микросхемы: дешифраторы и порт ввода-вывода.

3. Устройство ввода-вывода.

На схеме архитектуры фон Неймана (рис. 1.12) присутствовали только основные типы элементов: процессор, разделенный на две части: устройство управления и АЛУ, а также запоминающее устройство и устройства ввода и вывода. На общей схеме показаны необходимые интерфейсные и дополнительные элементы: шинный формирователь, порт ввода-вывода и дешифраторы.

Интерфейс — это средство взаимодействия элементов между собой. Процессор взаимодействует с остальными устройствами через некоторое интерфейсное устройство, которое в общем смысле можно назвать шинным формирователем. В зависимости от специфики ЭВМ он может быть реализован посредством универсальных микросхем коммутаторов — шинных формирователей — или специализированных — системных контроллеров, в том числе таких, как контроллеры-концентраторы памяти и ввода-вывода микросхем чипсета (см. п. 2.3).

Для повышения эффективности взаимодействия процессора с устройствами ввода-вывода может использоваться контроллер прерываний, который также реализуется с помощью универсальных или специализированных микросхем. Для организации взаимодействия между устройствами ввода-вывода и запоминающим устройством без участия процессора может использоваться контроллер прямого доступа к памяти. В персональных компьютерах контроллер прерыва-

ний и контроллер прямого доступа к памяти входят в состав микросхем чипсета вместе с другими контроллерами и интерфейсами. Помимо этих контроллеров в составе чипсета есть также контроллер памяти.

Запоминающее устройство обязательно включает две основные части: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ). Как уже говорилось выше, запоминающее устройство должно состоять из различных частей, каждая из которых предназначена для хранения информации определенного типа. ПЗУ содержит неизменяющуюся информацию — управляющую программу, множество базовых подпрограмм, выполняющих низкоуровневые функции: взаимодействие с оборудованием, задание начальных настроек (даты и времени, параметров оборудования и т. п). ПЗУ является энергонезависимой памятью, при выключении питания компьютера информация в нем сохраняется. ОЗУ является энергозависимой памятью, при выключении питания информация в нем пропадает. ОЗУ содержит программы, загружаемые с диска при их запуске, в том числе данные программ: глобальные переменные, данные, помещаемые в специальную область памяти — стек, в том числе локальные переменные подпрограмм, и т. п. Таким образом, после включения питания компьютера первую команду процессор должен читать из ПЗУ. Процессор также содержит элементы запоминающего устройства — регистры. Раньше их называли сверхоперативной памятью. Во всех универсальных современных компьютерах присутствует еще один важный вид запоминающего устройства — кэш-память. Она может быть многоуровневой, реализована в виде отдельных микросхем или как часть процессора. Кэш-память является чисто технологическим элементом, она не адресуется, соответственно, не занимает адресного пространства процессора и служит в качестве промежуточного буфера между основной оперативной памятью и внутренними элементами процессора для ускорения обмена информацией и повышения производительности компьютера.

Блок ВУ на общей схеме символизирует множество устройств ввода-вывода, которое является специфичным для конкретной модели ЭВМ. В состав любой ЭВМ должны входить, по крайней мере, одно устройство ввода и одно устройство вывода. Внешнее устройство изображено на более низком уровне, чем память, поскольку для взаимодействия с этим типом устройств необходим еще один интерфейсный элемент — порт ввода-вывода. Устройства ввода-вывода бывают самые разные, и оконечные элементы этих устройств — кнопочки, индикаторы и т. п. Поэтому необходимы некоторые унифицированные элементы, микросхемы, в определенной степени похожие на микросхемы памяти, содержащие буферные запоминающие элементы — регистры, как и у процессора, в которые записывается информация от внешнего устройства для приема ее процессором или от процессора для передачи во внешнее устройство.

Дешифратор формирует сигнал выбора микросхемы (ВМ или CS, Crystal Select). Чаще всего он имеет активный низкий уровень. Этот сигнал подключает или отключает микросхему от шины. Поскольку к шинам адреса, данных и управления множество устройств подключено параллельно, для взаимодействия с процессором должно быть включено только одно устройство, а остальные выключены.

Элементы ЭВМ связаны между собой посредством сигнальных линий, которые объединяются в шины. В ЭВМ выделяется три типа шин:

− шина данных;

− шина адреса;

− шина управления.

Основная характеристика шины данных и шины адреса — разрядность, т. е. количество линий, проводников для передачи информации.

Разрядность шины данных определяет максимальный размер операндов, которыми процессор может обмениваться с памятью или устройствами ввода-вывода за один раз, — 8, 16, 32 или 64 бита.

Разрядность шины адреса определяет размер адресного пространства:

$$AS = 2^{ABW}, \qquad (2.2)$$

где AS (Address Space) — размер адресного пространства;

ABW (Address Bus Width) — ширина шины адреса.

Размер адресного пространства — это максимальное число ячеек памяти, которое может адресовать процессор, что в свою очередь, определяет максимальный объем памяти, которую может иметь компьютер. Конечно, имеются различные способы расширения адресного пространства с помощью дополнительных регистров, но они были актуальны, когда у процессоров было небольшое число адресных линий. Например, если адресная шина 16-битная, то размер адресного пространства составляет 2^{16} = 64 К, где 1 К = 1024. Если шина данных 8-битная, то максимальный объем памяти — 64 Кбайта. У современных 64-разрядных процессоров адреса в программе в 64-битном режиме 64-битные. Шина данных тоже 64-битная, но память адресуется побайтно. Таким образом, максимальный теоретический объем памяти составляет 2^{64} = 16 Эбайт. Этого, мягко говоря, более чем достаточно, такого объема памяти нет даже у эксафлопсных суперкомпьютеров. Поэтому размер физического адреса у современных процессоров не больше 52 бит.

Из рис. 2.5 видно, что дешифратор связан именно с шиной адреса. Но на него поступают не все адресные сигналы, а только их часть, несколько старших адресных сигналов. Младшая часть адресных сигналов поступает непосредственно на устройство — ОЗУ, ПЗУ или порт ввода-вывода. Как выделяются эти части, зависит от конфигурации компьютера, свойств процессора, запоминающих и внешних устройств. Таким образом, старшая часть адресных линий адресует устройство в целом, а младшая часть — определенный адресуемый элемент внутри устройства — ячейку памяти в ОЗУ или ПЗУ, регистр в порту, интерфейсе, адаптере или контроллере ввода-вывода. Число

таких элементарных адресуемых элементов в устройстве связано с числом младших адресных сигналов, поступающих на него, той же формулой (2.2).

Шина управления является наиболее информационно насыщенной. Такой характеристики, как разрядность, у нее нет. Она характеризуется просто множеством управляющих сигналов, которое сильно варьируется в зависимости от модели процессора, но, тем не менее, у всех этих конкретных множеств есть пересечение — минимально необходимое множество управляющих сигналов, которое присутствует во всех процессорах. К нему относятся сигналы, определяющие направление передачи информации в ЭВМ. Названия этих сигналов могут быть разные, но смысл один и тот же — куда идут данные, к процессору или от процессора. На схеме они обозначены как чтение и запись запоминающего устройства, чтение и запись внешнего устройства: ЧТЗУ, ЗПЗУ, ЧТВУ, ЗПВУ.

Направление передачи информации в ЭВМ определяется относительно процессора. Процессор — это центральный элемент компьютера, он и является точкой отсчета. Если информация движется к процессору, то это прием. Прием из памяти — чтение, из внешнего устройства — ввод. Если информация движется от процессора, то это передача. Передача в память — запись, во внешнее устройство — вывод (табл. 2.1).

Таблица 2.1

Направления передачи информации в ЭВМ

Направление	Память	УВВ
От процессора — передача (transmission)	Запись (writing)	Вывод (output)
К процессору — прием (receiving)	Чтение (reading)	Ввод (input)

Это наиболее правильная терминология, но ее не всегда придерживаются даже в официальной технической документации и учебной литературе. Поэтому и здесь мы будем использовать терми-

ны «запись» и «чтение» в более широком смысле, как синонимы терминов «передача» и «прием».

Порты ввода-вывода. Дешифраторы. Ресурсы

Порты ввода-вывода с аппаратной точки зрения — это предназначенные для подключения внешних устройств платы, микросхемы или части микросхем, в которых имеются запоминающие элементы, регистры, к которым процессор может обращаться для записи или чтения, как и к памяти, по определенным адресам. Поэтому с программной точки зрения порты ввода-вывода — это адреса этих регистров. Эти адреса могут располагаться в основном адресном пространстве (адресном пространстве памяти) или в отдельном адресном пространстве — пространстве ввода-вывода. По этому признаку выделяется два типа интерфейсов ввода-вывода:

1. Ввод-вывод по принципу доступа к памяти.
2. Изолированный ввод-вывод.

Преимущество изолированного ввод-вывода в том, что он не занимает адресного пространства памяти. Недостаток — программная часть ограничивается командами IN и OUT. Для ввода-вывода по принципу доступа к памяти можно использовать весь набор команд, но недостаток в том, что он занимает часть адресного пространства. Как и другие типы взаимодействия, они могут комбинироваться.

Адреса портов ввода-вывода, как правило, короче, чем адреса памяти, поскольку для их адресации используются не все адресные линии процессора, а некоторая их часть. Например, в компьютерах с процессорами x86, адреса памяти могут быть 16, 32 или 64 бита, а адреса портов — 8 или 16 бит.

Многие внешние устройства работают под управлением специального устройства — контроллера или адаптера. Например, выводом информации на монитор управляет видеоадаптер (видеокарта).

Вводом информации с клавиатуры управляет контроллер клавиатуры. В таких случаях порты ввода-вывода находятся в соответствующих контроллерах.

Контроллеры или адаптеры устройств ввода-вывода помимо регистров могут иметь в своем составе память, которая может занимать определенный диапазон в адресном пространстве процессора.

Для повышения эффективности взаимодействия элементов компьютера между собой и, соответственно, увеличения его производительности используются такие контроллеры, как контроллер прерываний и контроллер прямого доступа к памяти. Они назначают связанным с ними устройствам ввода-вывода номера прерываний и каналов прямого доступа к памяти. Механизм прерываний очень важен в компьютерных системах. При взаимодействии процессора с внешними устройствами инициатива может принадлежать процессору или внешнему устройству. В первом случае способ взаимодействия называется опросом. При опросе процессор постоянно обращается к внешнему устройству, чтобы определить, есть ли у него данные для обмена. Понятно, что это занимает много процессорного времени. Взаимодействие по прерыванию, когда инициатива принадлежит внешнему устройству, сильно разгружает процессор. В этом случае он выполняет основную программу, а внешнее устройство, когда появляется необходимость обмена данными с процессором, формирует сигнал прерывания, по которому процессор ответвляется от основной программы и переходит к обработчику прерывания, в котором и выполняется обмен.

Дешифратор, как уже было сказано выше, формирует сигнал выбора микросхемы по нескольким старшим адресным линиям. Только одна конкретная комбинация нулей и единичек на этих линиях включит устройство. На младших адресных линиях при этом может находиться любое значение от минимального, когда все адресные

линии равны нулю, до максимального, когда все адресные линии равны единице. Таким образом, посредством дешифраторов назначаются диапазоны адресов, которые занимает устройство в адресном пространстве памяти или ввода-вывода. Это один из видов ресурсов, выделяемых в ЭВМ различным устройствам. Всего имеется четыре вида таких ресурсов:

– диапазоны в адресном пространстве памяти;
– диапазоны в пространстве ввода-вывода (номера портов);
– номера линий запросов прерывания;
– номера каналов прямого доступа к памяти.

Надо иметь в виду, что это именно аппаратные ресурсы, их не надо путать с другими видами ресурсов в ЭВМ, например, процессорным временем — интервалом времени работы процессора, выделяемым отдельным задачам в многозадачной системе.

Значения аппаратных ресурсов различных устройств можно посмотреть с помощью утилиты msinfo32 («Сведения о системе») или диспетчера устройств операционной системы. Его можно запустить через свойства компьютера, с использованием проводника или командной строки:

C:\Windows\System32\devmgmt.msc

В диалоговом окне свойств устройства есть вкладка «Ресурсы». Посмотрим ресурсы стандартной клавиатуры (рис. 2.6). У нее имеется два вида ресурсов — *диапазон ввода-вывода*, который включает два порта — 60 и 64 (значения приведены в шестнадцатеричном формате), и *номер прерывания* (IRQ 1). Запомним эти значения, они нам понадобятся в дальнейшем (см. п. 4.3).

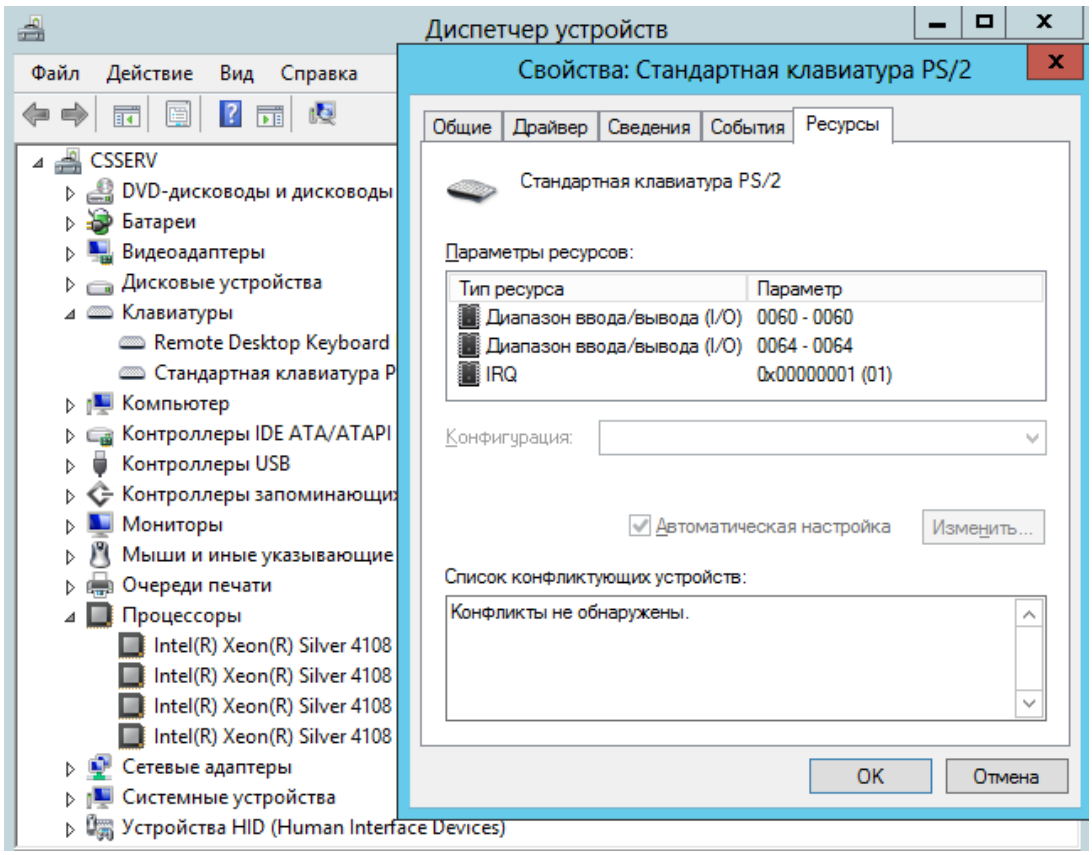


Рис. 2.6. Ресурсы клавиатуры

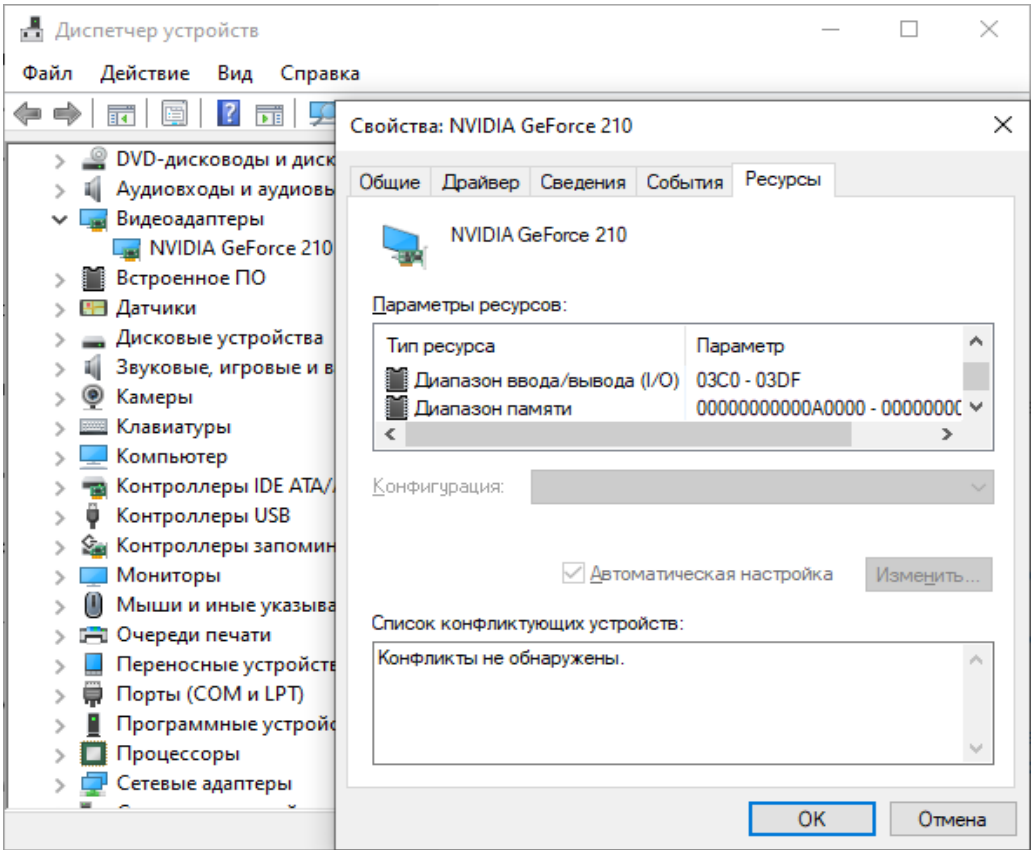


Рис. 2.7. Ресурсы видеоадаптера

Теперь посмотрим ресурсы видеоадаптера (рис. 2.7). Видеоадаптер имеет три вида ресурсов — *диапазоны памяти*, *диапазоны ввода-вывода* и *номер прерывания*. Они будут так же использованы в разделе 4.

Внешние устройства ЭВМ

Внешние устройства можно классифицировать следующим образом:

1. По направлению передачи информации:

1.1. Устройства ввода (клавиатура, мышь, микрофон, видеокамера, датчики).

1.2. Устройства вывода (монитор, проектор, колонки или наушники, исполнительные механизмы (актуаторы)).

1.3. Устройства ввода-вывода (накопители на магнитных дисках, твердотельные накопители (SSD, Solid State Drive)).

2. По источнику-назначению информации:

2.1. Пользователь.

2.2. Внешняя память.

2.3. Другие компьютеры (сетевые коммуникационные устройства: сетевые платы, модемы и т. п.).

2.4. Объект управления (устройства связи с объектом: датчики, преобразователи, исполнительные механизмы).

В свою очередь, устройства взаимодействия пользователя с машиной имеют свои признаки классификации и выделяемые по ним типы:

1. Интерактивность взаимодействия:

1.1. Диалоговые средства (монитор, клавиатура, мышь, видеокамера и звуковые устройства).

1.2. Недиалоговые средства (принтер, сканер, многофункциональное устройство).

2. Используемые органы и системы человека:

2.1. Зрение (монитор).

2.2. Слух (колонки, наушники).

2.3. Речевой аппарат (микрофон).

2.4. Опорно-двигательный аппарат, осязание (клавиатура, мышь, сенсорная панель (touchpad), различные игровые устройства).

Устройства взаимодействия пользователя с машиной вместе с соответствующим программным обеспечением составляют так называемый интерфейс «человек-компьютер» (ИЧК) или человеко-машинный интерфейс. Основное требование к ИЧК — дружественность. Его качество определяет эффективность использования диалоговой вычислительной системы.

Устройства внешней памяти классифицируются в основном по типу носителя информации:

1. Магнитная лента (стример (ленточный накопитель), магнитофон, видеомагнитофон).
2. Магнитный диск (накопитель на жестких магнитных дисках (НЖМД или HDD, Hard Disk Drive), накопитель на гибких магнитных дисках (НГМД или FDD, Floppy Disk Drive).
3. Оптический диск (приводы CD-ROM, DVD-ROM).
4. Магнитооптический диск (магнитооптический дисковод).
5. Кристалл полупроводника (Flash-память различного типа).

Устройства связи с объектом используются в таких компьютерных системах, как автоматизированные системы управления (АСУ). **АСУ** — это компьютерная система, использующая такие устройства ввода-вывода, как датчики и исполнительные механизмы, а также необходимые для них интерфейсные элементы: модули ввода, вывода и т. п. (рис. 2.8) [39].

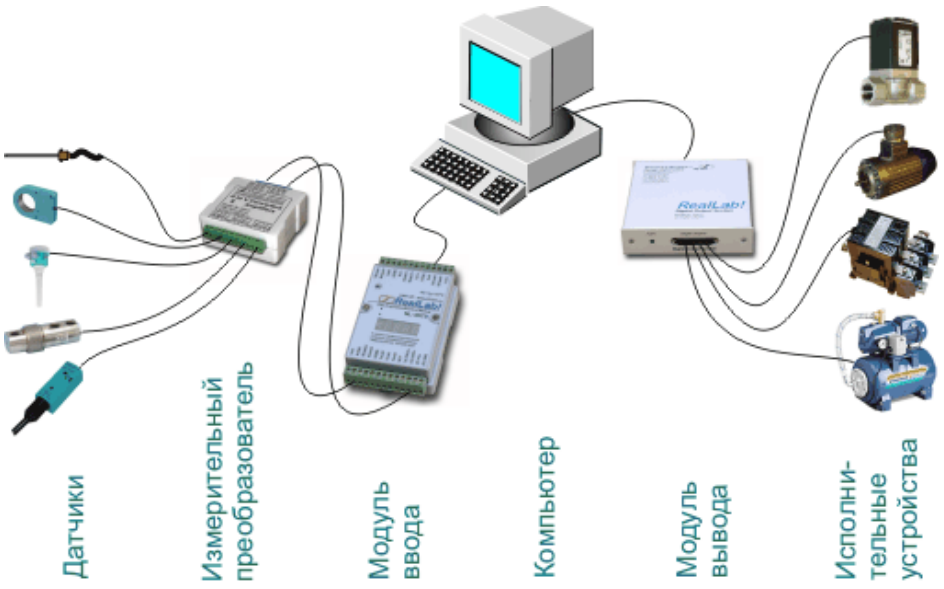


Рис. 2.8. Элементы автоматизированной системы управления

Датчики, изображенные на рисунке, являются самыми простыми, состоящими только из первичного преобразователя, выдающего электрический сигнал, соответствующий значению измеряемой величины. Многие современные датчики являются сложными устройствами, которые включают помимо первичного преобразователя вторичный или измерительный преобразователь, аналого-цифровой преобразователь, цифровой интерфейс и даже микропроцессор, т. е. являются, по сути, также вычислительными системами.

Множество датчиков имеется в смартфонах:

- акселерометр для измерения ускорения и подсчета шагов;
- гироскоп для определения ориентации в пространстве;
- магнитометр (компас);
- датчик Холла;
- барометр;
- датчик приближения;
- датчик освещенности;
- термометр;
- гигрометр (датчик влажности);
- пульсометр;
- дозиметр и др.

Конечно, набор датчиков в смартфоне зависит от его модели и стоимости. Помимо перечисленых датчиков в смартфоне могут присутствовать и другие, дополнительные. Например, для измерения количества сделанных шагов в качестве отдельного датчика может присутствовать шагомер. Информацию с датчиков обрабатывают соответствующие программы смартфона, обеспечивая необходимую реакцию, например, поворот изображения на экране. Таким образом, обычный смартфон — это сложная автоматизированная система управления.

Запоминающие устройства ЭВМ

Запоминающие устройства ЭВМ делятся на внешние и внутренние. Внутренние запоминающие устройства занимают определенные диапазоны адресов в адресном пространстве, соответственно, доступ к ним осуществляется по адресам.

Внешние запоминающие устройства относятся к устройствам ввода-вывода и имеют соответствующий контроллер. Доступ к ним осуществляется через интерфейсы и порты ввода-вывода.

К внешним запоминающим устройствам относятся:

- стример (ленточный накопитель);
- накопитель на жестких магнитных дисках (НЖМД или HDD, Hard Disk Drive) или просто жесткий диск;
- накопитель на гибких магнитных дисках (НГМД или FDD, Floppy Disk Drive);
- привод компакт дисков (CD-ROM, DVD-ROM и др.);
- твердотельный накопитель (SSD, Solid State Drive);
- USB Flash и т. п.

Магнитные носители все реже используются в персональных компьютерах, но по-прежнему незаменимы в data-центрах. Для повышения надежности HDD-накопителей в них используется технология S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology, технология самоконтроля, анализа и отчетности). Она позволяет определить и оценить состояние накопителя с помощью встроенных средств, выявить потенциальные проблемы и спрогнозировать возможные отказы. SSD-накопители также поддерживают S.M.A.R.T. Данные, которые позволяет получить технология S.M.A.R.T., можно посмотреть с помощью специальных программ, информационно-диагностических утилит, например, HWiNFO.

Продолжают развиваться накопители на магнитных лентах (стримеры). Они используются в мейнфреймах для резервного копи-

рования и долговременного хранения информации благодаря следующим достоинствам:

- большой емкости;
- низкой стоимости;
- невысоким требованиям к условиям хранения носителя;
- высокой надежности;
- низкому энергопотреблению.

В 2020 г. IBM представила ленточный накопитель емкостью до 580 Тбайт с плотностью записи 49 Гбит/см^2.

Операционная система организует хранение информации во внешней памяти в виде так называемых файлов. Для этого используется файловая система, являющаяся важной частью операционной системы. **Файл** — это поименованная единица хранения информации во внешней памяти. Структурирование файлов осуществляется с помощью каталогов (папок, директорий). Каталог самого верхнего уровня в иерархической структуре файлов и каталогов на диске называется корневым. Файловая система определяет все характеристики и параметры хранения информации: максимальный размер имен файлов и каталогов, максимальный размер файла, набор атрибутов файла, ограничение доступа к файлам и т. д.

Внешние накопители делятся на разделы, в которых создаются логические диски (тома). Долгое время в качестве внешней памяти использовались исключительно диски: магнитные, оптические и магнитооптические. Поэтому термин «диск» используется и для твердотельных накопителей, в которых дисков нет, а информация хранится в микросхемах Flash-памяти.

Минимальной адресуемой единицей хранения информации на внешнем носителе с точки зрения его контроллера является сектор.

Минимальной единицей чтения-записи диска операционной системой является кластер. Кластер включает несколько секторов, хотя может состоять и из одного сектора. Таким образом, файл на диске занимает множество кластеров и секторов. Важным параметром фай-

ловой системы является размер идентификатора (индекса) кластера. Он определяет количество кластеров на диске.

При открытии файла или запуске исполняемого файла операционная система читает информацию из файла и размещает ее в оперативной памяти. При сохранении файла происходит обратный процесс.

Существует множество различных файловых систем: FAT, NTFS и др. Операционная система может поддерживать одну или несколько конкретных файловых систем.

Внутренние запоминающие устройства делятся на два типа по признаку изменения информации в них в процессе работы ЭВМ:

1. Оперативное запоминающее устройство (ОЗУ, RAM — Random Access Memory). Информация в нем изменяется, т. е. не только читается, но и записывается.

2. Постоянное запоминающее устройство (ПЗУ, ROM — Read Only Memory). Информация в нем не изменяется, т. е. только читается.

RAM переводится как память с произвольным доступом. Такое название было выбрано как противоположное памяти с последовательным доступом — перфолентам и магнитным лентам, при чтении информации с которых необходимо последовательно обратиться ко всем ячейкам, начиная с первой. Произвольный доступ означает обращение по любому адресу в любом такте или цикле. В этом смысле ROM также является памятью с произвольным доступом.

ОЗУ является энергозависимой, а ПЗУ — энергонезависимой памятью. При выключении питания компьютера информация в оперативной памяти теряется. Информация в ПЗУ сохраняется при выключении питания, но в перепрограммируемых ПЗУ ее можно изменить или в составе самой ЭВМ, или в специальном устройстве — программаторе ПЗУ.

Существует также понятие энергонезависимой RAM — NVRAM (Non-Volatile Random-Access Memory). К нему может быть отнесена и стираемая программируемая постоянная память EPROM (Erasable

Programmable Read-Only Memory) и новые типы RAM: FRAM (FeRAM) и MRAM (см. далее).

ОЗУ персональных компьютеров содержит выполняющиеся в настоящий момент программы, в том числе операционную систему, загружаемые с устройств внешней памяти, и необходимые для их выполнения данные.

ПЗУ персональных компьютеров содержит BIOS (Basic Input-Output System, базовую систему ввода-вывода) или в современной терминологии Firmware (встроенное программное обеспечение, «фирменное обеспечение»). Поэтому оно называется ROM BIOS, EFI (Extensible Firmware Interface) или UEFI (Unified Extensible Firmware Interface).

В составе персональных компьютеров есть также так называемая CMOS RAM — оперативная память, в которой хранятся настройки BIOS, которые можно поменять с помощью программы Setup BIOS. CMOS (Complementary Metal-Oxide-Semiconductor) — технология полупроводниковых интегральных схем с наиболее низким энергопотреблением. CMOS RAM питается от батарейки и поэтому сохраняет содержимое при выключении питания.

Содержимое ROM BIOS современных компьютеров может быть перезаписано новой версией BIOS, причем в составе самого компьютера. Такая возможность имеется у ПЗУ, изготовленных по технологии Flash, — Flash-ROM. Однако это делается, конечно, не в процессе обычной работы компьютера, а в особом режиме с помощью специальных программных средств.

ПЗУ имеется и в составе современных модулей оперативной памяти DIMM SDRAM. Это микросхема SPD EEPROM (Serial Presence Detect EEPROM), которая идентифицирует тип модуля, различные параметры организации, временные параметры, характеризующие производительность (тайминги).

Классификация ПЗУ:

1. Однократно программируемые:

1.1. Масочные ПЗУ — программируемые производителем в процессе изготовления путем наложения маски.

1.2. Электрически программируемые ПЗУ (ЭППЗУ) — программируемые потребителем после приобретения пережиганием перемычек электрическим путем в программаторе.

2. Многократно программируемые (перепрограммируемые) (Erasable Programmable Read-Only Memory, EPROM):

2.1. Со стиранием ультрафиолетовым светом и электрической записью — УФППЗУ.

2.2. Электрически стираемое программируемое ПЗУ (ЭСППЗУ или EEPROM, Electrically Erasable Programmable Read-Only Memory).

2.3. Flash-ROM — разновидность EEPROM, бывает двух типов:

2.3.1. NOR (Not OR) — 2-мерная матрица (тот же crossbar). Используется в микроконтроллерах, ROM BIOS.

2.3.2. NAND (Not AND) — 3-мерный массив. Используется в USB-накопителях (флешках), накопителях SSD, SD-картах (Secure Digital Memory Card), смартфонах.

2.4. 3D XPoint (читается “3D crosspoint”). Устройства 3D XPoint компании Intel выходят под торговой маркой Optane. Бывают следующих типов:

2.4.1. Intel Optane Memory в форм-факторе M.2, использующийся в качестве ускорителя основного накопителя.

2.4.2. Intel Optane Persistent Memory Module — SSD-накопитель в формате DIMM-модуля памяти DDR4, использующийся в серверах с процессорами Intel Xeon Scalable второго поколения в дополнение к модулям оперативной памяти.

2.4.3. Intel Optane SSD, использующийся в серверах в качестве основного накопителя.

Основные качественные признаки классификации ОЗУ и выделяемые по ним типы:

1. *Архитектура*: FPM (Fast Page Mode), EDO (Extended Data OUT) и др.

2. *Запоминающий элемент*: конденсатор — динамическая память (DRAM, Dynamic RAM), триггер — статическая память (SRAM, Static RAM).

3. *Конструктивное исполнение*: SIMM (Single Inline Memory Module, модуль памяти с однорядным расположением контактов), DIMM (Dual Inline Memory Module, модуль памяти с двухрядным расположением контактов), SO-DIMM (Small Outline DIMM), Mini-DIMM, MicroDIMM и т. п.

4. *Наличие синхронизации тактовыми импульсами*: асинхронная память, синхронная память (SDRAM, Synchronous DRAM).

5. *Умножение скорости передачи по сравнению с частотой тактовых импульсов*: RDRAM (Rambus DRAM, выпускается в конструктивах под названием RIMM (Rambus Inline Memory Module)) и DDR (Double Data Rate, память с удвоенной скоростью передачи данных, конструктив DIMM).

6. *Многоканальность*: двухканальная, трехканальная и т. д.

Основные количественные характеристики ПЗУ и ОЗУ:

1. Объем.

2. Размер ячейки: бит, тетрада, байт и т. д.

3. Время доступа.

Для синхронной оперативной памяти еще одной важной характеристикой является тактовая частота и количество тактов, требующихся для доступа, — так называемые тайминги.

Надо отметить, что во многих современных модулях оперативной памяти существует возможность программного конфигурирования количества и размера ячеек.

Как уже было сказано, канал между процессором и памятью является фактором, ограничивающим производительность компьютера, «бутылочным горлышком» фон-неймановской архитектуры.

Существует два способа увеличения пропускной способности любого тракта (даже обычной автомобильной дороги):

1. Увеличение ширины (разрядности шины).

2. Увеличение скорости (тактовой частоты).

Влияет на производительность также время доступа к памяти для чтения или записи. Основная оперативная память всегда имеет недостаточную производительность, поэтому для увеличения производительности компьютера используется многоуровневая организация памяти. Уровни выделяются по ***степени близости к ядру процессора***, минимальное количество уровней — два:

1. Основная оперативная память — динамическая, менее дорогая и быстрая, более емкая.
2. Кэш-память — статическая, более дорогая и быстрая, менее емкая.

Существуют различные виды кэш-памяти, выделяемые по разным признакам. По ***признаку конструктивного исполнения*** (физической реализации) можно выделить:

1. Кэш-память в виде отдельных микросхем.
2. Кэш-память в одном корпусе с процессором, но на отдельном кристалле.
3. Кэш-память на одном кристалле с процессором.

Эти типы физической реализации, конечно, различаются архитектурными особенностями и другими характеристиками.

Кэш-память может быть разделена на уровни, которые также характеризуются увеличением производительности и уменьшением емкости по мере близости к ядру процессора.

В большинстве современных процессоров кэш-память находится на кристалле процессора и делится на два или три уровня, которые обозначаются L1, L2 и L3 соответственно.

Кэш-память является промежуточным буфером между основной памятью и процессором и не занимает его адресного пространства. Поэтому доступ к ней осуществляется не по адресам, а по содержимому, т. е. ассоциативно, но с использованием некоторых элементов адресации для ускорения доступа.

В настоящее время разрабатывается и выпускается множество новых типов запоминающих устройств, наиболее интересными из которых являются:

− память на основе мемристоров (см. п. 2.1);

− сегнетоэлектрическая память (Ferroelectric RAM, FeRAM или FRAM);

− магниторезистивная память (MRAM, Magnetoresistive RAM).

Все эти перечисленные типы сочетают свойства оперативной и постоянной памяти.

FRAM устроена так же, как и DRAM, запоминающим элементом в ней является конденсатор, но вместо диэлектрического слоя в ней используется слой сегнетоэлектрика (ферроэлектрика). Это обеспечивает ее энергонезависимость и, соответственно, возможность использования в качестве постоянной памяти.

MRAM является наиболее новой и интересной памятью, сочетающей в себе магнитные и полупроводниковые технологии. Можно сказать, это возвращение к памяти на магнитных сердечниках, которая широко использовалась в качестве основной внутренней памяти в 1950–70-х гг., на новом технологическом витке исторического развития. Для хранения информации в MRAM используются магнитные моменты, как и в магнитных дисках. MRAM так же, как и FRAM, обладает энергонезависимостью.

В нашей стране производителем микросхем MRAM является компания «Крокус Наноэлектроника» [40]. Микросхемы MRAM имеют множество преимуществ. При этом у них такой же интерфейс, что и у обычных микросхем RAM. Поэтому для применения памяти MRAM не требуется изменение конструкции электронного устройства и замена других электронных элементов.

Основные особенности микросхем MRAM:

− энергонезависимость;

− высокая производительность: время доступа для чтения 35 нс, для записи 35/90/120/150 нс;

− неограниченное число циклов записи и чтения;

− срок службы до 20 лет;

− широкий диапазон рабочих температур.

Интерфейс процессора с памятью и внешними устройствами

Взаимодействие процессора с другими устройствами осуществляется с помощью дешифратора. Как было сказано выше, на него поступает старшая часть адресных сигналов, а младшая часть поступает непосредственно на микросхему памяти или порта ввода-вывода. Дешифратор формирует сигнал выбора микросхемы CS#. Активный низкий уровень этого сигнала формируется только при одном конкретном значении старших адресных сигналов при полной дешифрации и при некотором диапазоне значений — при частичной дешифрации.

Помимо сигнала включения микросхемы на нее необходимо подать сигналы, определяющие направление передачи информации — чтение или запись. Из ПЗУ информация может только читаться, поэтому достаточно активировать сигнал, разрешающий выдачу данных. Сущность интерфейса процессора с микросхемой ПЗУ иллюстрирует пример на рис. 2.9.

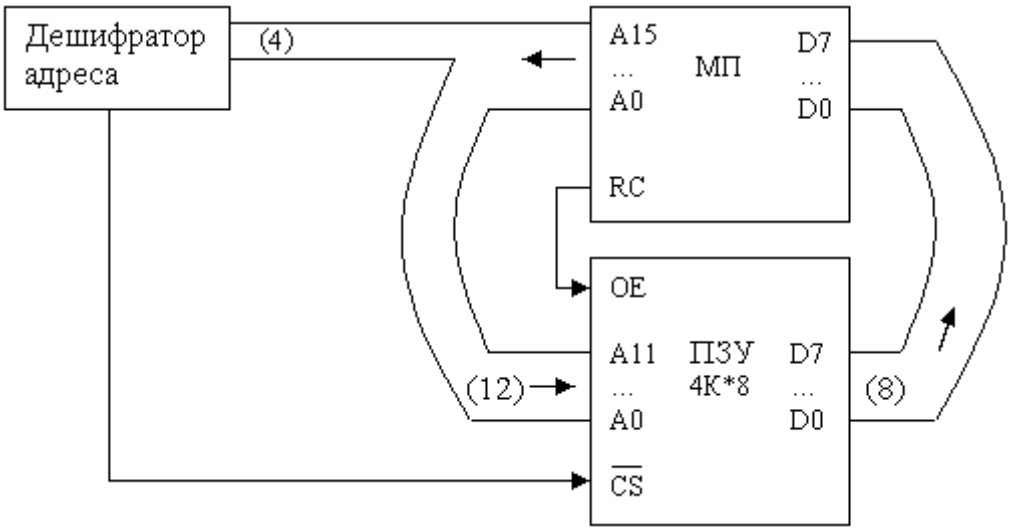


Рис. 2.9. Интерфейс процессора с ПЗУ

В приведенном примере шина адреса процессора 16-разрядная. ПЗУ имеет 12 адресных входов. Поэтому на вход дешифратора по-

ступает четыре старших адресных сигнала — A15–A12. Сигналы A11–A0 поступают непосредственно на микросхему ПЗУ. Разрешает выдачу данных из микросхемы памяти сигнал OE (Output Enable) — разрешение выхода. На вход OE поступает сигнал RC процессора. В данном примере оба сигнала имеют активный высокий уровень. В цикле чтения данных процессором сигнал RC устанавливается в 1. Если значение адреса, выданного процессором, находится в диапазоне, соответствующем микросхеме ПЗУ в адресном пространстве, дешифратор включит сигнал CS# и ПЗУ выдаст значение выбранной ячейки, которое по шине данных поступит в процессор.

Пример интерфейса процессора с микросхемой ОЗУ показан на рис. 2.10. Она может, как читаться, так и записываться, поэтому на нее подаются и сигнал чтения RC, и сигнал записи TR#. Сигнал RC так же поступает на вход OE, сигнал записи TR# — на вход R/W# — чтения/записи микросхемы ОЗУ. Значения управляющих сигналов при чтении ОЗУ:

TR# = 1 (неактивный уровень);

RC = 1 (активный уровень).

При записи:

TR# = 0 (активный уровень);

RC = 0 (неактивный уровень).

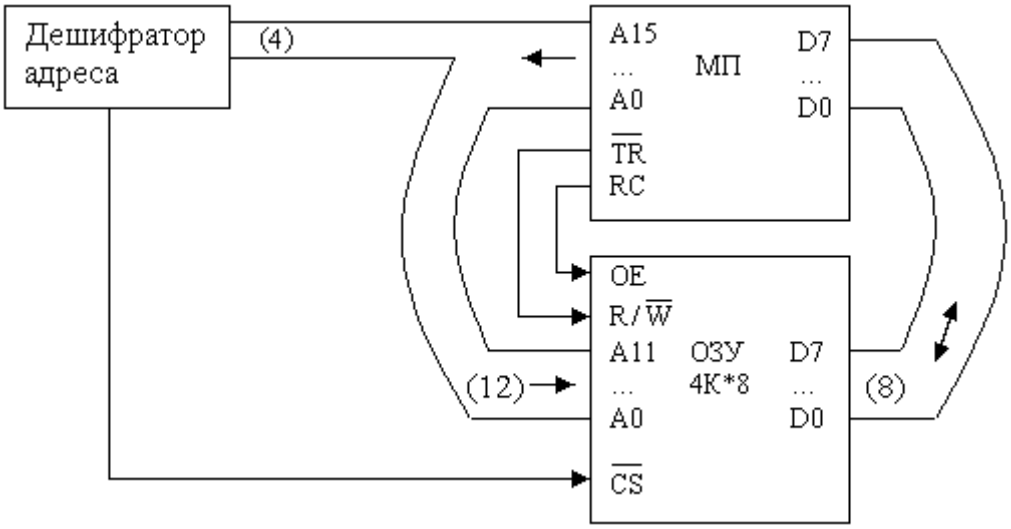


Рис. 2.10. Интерфейс процессора с ОЗУ

Приведенные примеры иллюстрируют основные принципы взаимодействия процессора с внутренней памятью, поэтому шинный формирователь в них не присутствует.

Конечно, у различных микросхем памяти множество входных и выходных сигналов различается. Большинство современных микросхем оперативной памяти имеет матричную организацию, при которой ячейки расположены в виде матрицы, по строкам и столбцам. В матричных ЗУ адрес, поступающий к памяти, делится на две части. Одна определяет адрес строки, другая — адрес столбца. В таких микросхемах кроме сигнала CS используются сигналы RAS (Row Address Strobe, строб адреса строки) и CAS (Column Address Strobe, строб адреса столбца). Вначале подается сигнал RAS, а затем с некоторой задержкой — CAS. Сигналов RAS и CAS может быть несколько, в зависимости от количества матриц ячеек памяти. Например, в 30-контактных модулях SIMM по одному сигналу RAS и CAS. В 72-контактных — по четыре.

У синхронной оперативной памяти SDRAM важной характеристикой производительности или пропускной способности являются тайминги — временные параметры, связанные с сигналами RAS и CAS и выражаемые в тактах. Этих параметров может быть три или четыре:

− CL, CAS Latency, латентность — задержка между отправкой памяти адреса столбца и началом передачи данных, т. е. число тактов, требуемое на чтение первого значения из памяти, когда нужная строка уже открыта;

− T_{RCD}, RAS to CAS Delay, задержка строба столбца относительно строба строки — число тактов между открытием строки и доступом к столбцу в ней;

− T_{RP}, RAS Precharge Time, время предварительной зарядки для открытия следующей строки — число тактов между командой на предварительный заряд банка (закрытие строки) и началом открытия следующей строки;

− T_{RAS}, RAS Active Time, время активности строки — число тактов, в течение которых строка памяти может быть доступна для чтения или записи.

Тайминги записываются в виде последовательности трех или четырех чисел в перечисленном порядке, например: 2-2-2 или 2-2-2-6. Иногда указывается только один параметр, например, CL2.

Рекомендуемые значения таймингов для наиболее распространенных частот системной шины записаны в микросхеме SPD, находящейся на планке модуля памяти вместе с основными микросхемами.

Тайминги оказывают существенное влияние на реальную производительность компьютера при небольшом размере кэш-памяти и практически не оказывают такого влияния при достаточном ее размере, который характерен практически для всех современных процессоров.

Взаимодействие процессора с внешними устройствами осуществляется через порты ввода-вывода. Порт в этом интерфейсе занимает место микросхемы памяти, но является промежуточным, буферным элементом (рис. 2.11).

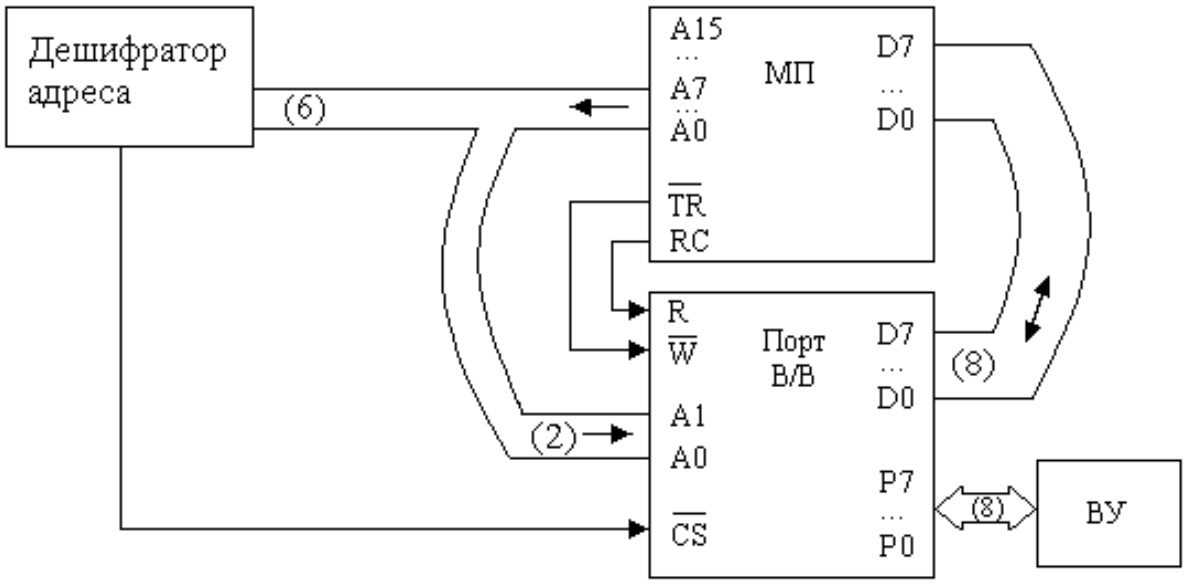


Рис. 2.11. Интерфейс процессора с ВУ

Для взаимодействия с внешними устройствами требуется, как правило, меньшее число адресных линий. В примере на рис. 2.11 используется младшая половина адресной шины. Шесть адресных сигналов поступает на дешифратор, и два — на порт. Для связи с устройствами ввода-вывода порт может иметь разное число каналов и разное число линий данных в канале. На рисунке показан только один 8-разрядный канал. Такой порт называется параллельным. Если используется только одна линия и, соответственно, в каждом такте в одну сторону может быть передан только один бит, то такой порт называется последовательным. Классификация интерфейсов ввода-вывода, характеристики наиболее распространенных из них будут рассмотрены в разделе 5.

2.3. АРХИТЕКТУРА ПЭВМ

Набор микросхем (чипсет) ПЭВМ. Архитектура северного и южного моста

В ПЭВМ дополнительные и интерфейсные устройства, необходимые для организации ее работы, взаимодействия основных типов устройств: процессора, памяти и устройств ввода-вывода — входят в состав так называемого чипсета (набора микросхем). Микросхемы чипсета имеют высокую степень интеграции, каждая из них включает множество контроллеров и интерфейсов, и соответственно, число микросхем в чипсете небольшое. Долгое время чипсет включал только две основные микросхемы:

− контроллер-концентратор памяти (MCH, Memory Controller-Hub) — северный мост (north bridge);

− контроллер-концентратор ввода-вывода (ICH или IOCH, Input-Output Controller-Hub) — южный мост (south bridge).

Это наиболее удобная и логичная архитектура, она стала классической, поскольку в наибольшей степени соответствует понятию

треугольника фон Неймана — для каждого типа устройств в ней предназначен свой контроллер, хотя, в микросхему MCH входит также интерфейс для подключения видеоадаптера. До нее в составе чипсета было большее количество микросхем, а после, отметим сразу, осталась только одна — PCH (Platform Controller-Hub, контроллер-концентратор платформы).

Начнем рассмотрение архитектуры северного и южного моста с наиболее популярного в свое время чипсета Intel 440BX. Микросхемы чипсета отражены в списке системных устройств диспетчера устройств (рис. 2.12).

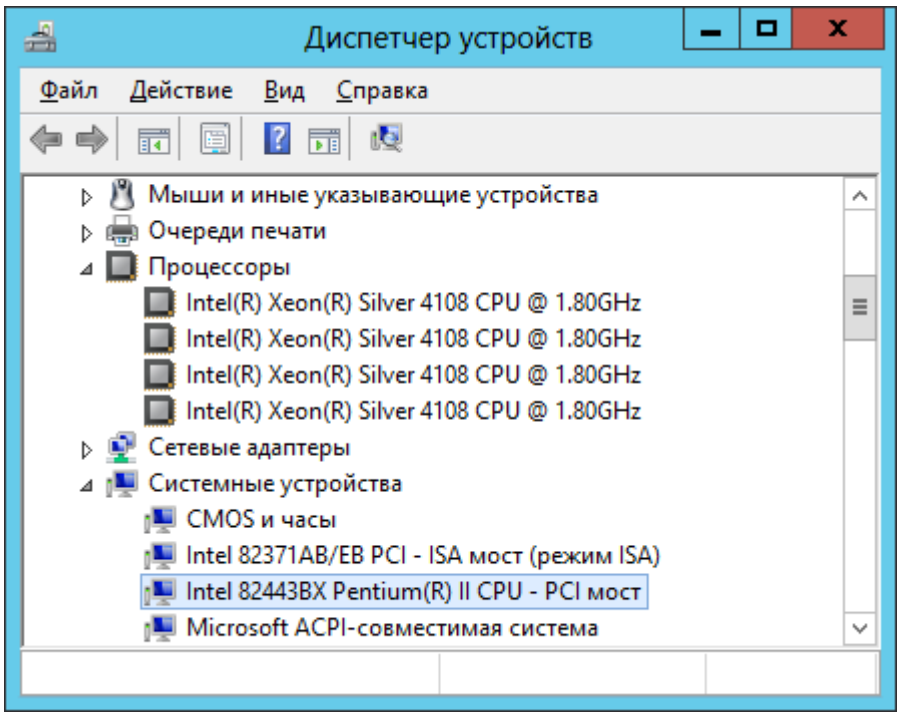


Рис. 2.12. Микросхемы чипсета в списке системных устройств

Чипсет Intel 440BX включает следующие две микросхемы северного и южного моста (рис. 2.13) [41].

- 82443BX — контроллер-концентратор памяти;
- 82371EB — контроллер-концентратор ввода-вывода.

Основные характеристики чипсета i440BX:

- процессор Pentium II, Pentium III или Celeron;
- поддержка двухпроцессорной системы;

– системная шина 64 бита 100 МГц;
– интерфейс с памятью SDRAM 64 бита 100 МГц.

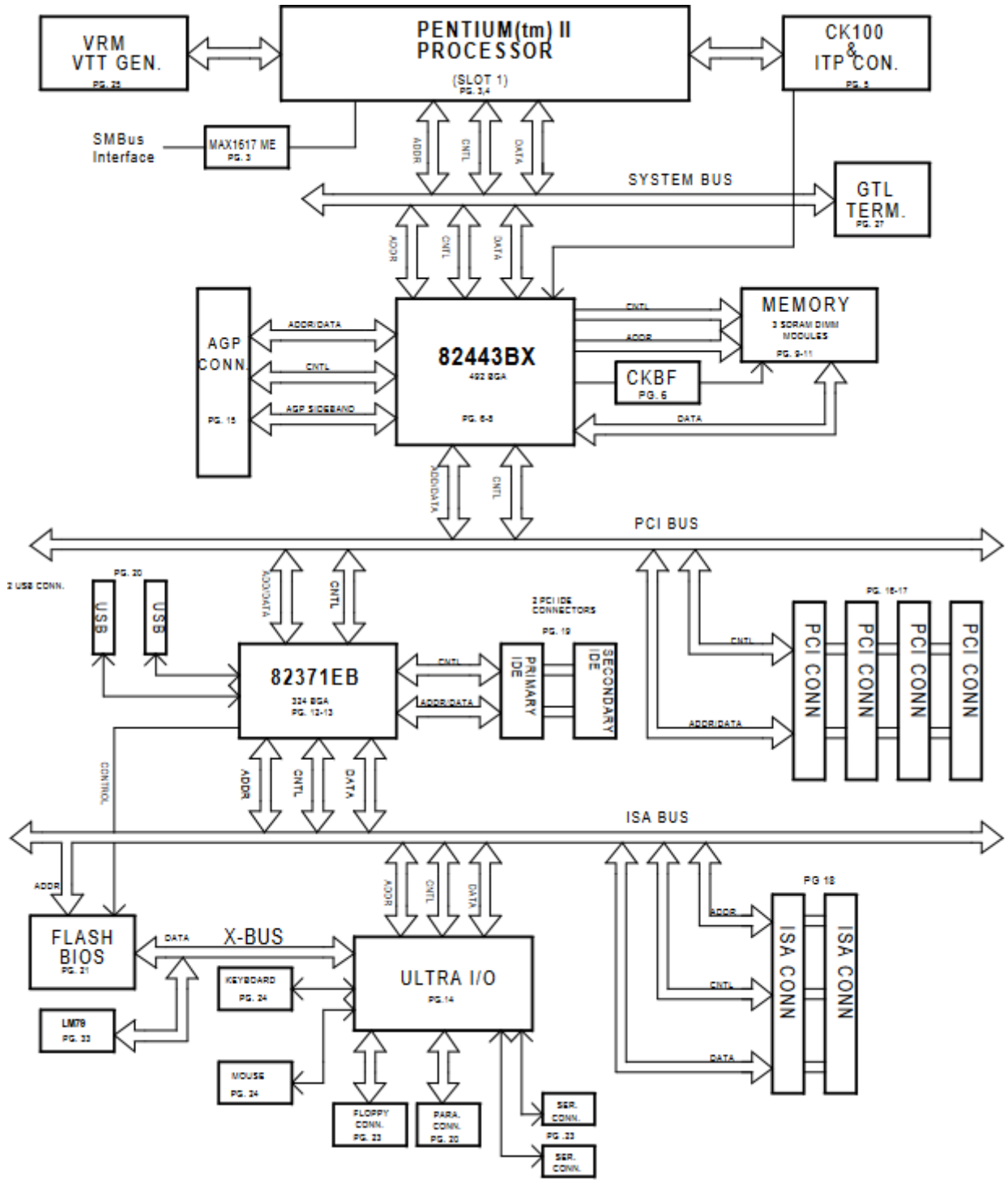


Рис. 2.13. Структурная схема компьютера с чипсетом Intel 440BX

Процессор изображен вверху схемы. Справа от него генератор тактовых импульсов с частотой 100 МГц, слева — регулятор напря-

жения (VRM, Voltage Regulator Module). ITP CON (ITP Test Connector) — опциональный разъем для тестирования. Микросхема MAX1617 ME — цифровой датчик температуры с интерфейсом SMBus (System Management Bus, шина управления системой). SMBus — это последовательная шина, основанная на популярном двухпроводном интерфейсе I2C. Она используется для получения информации о параметрах конфигурации и состояния системы, например, температуре процессора, переключения в режим пониженного энергопотребления и т. п.

Ниже мы видим три шины процессора — адреса, управления и данных, — которые составляют в совокупности системную шину. GTL TERM (GTL Termination Resistors) — это согласующие резисторы.

Микросхема 82443BX — контроллер главного моста (Host Bridge Controller) или просто северный мост [42]. Она подключается к системной шине и реализует интерфейс с памятью и видеоадаптером. Память имеет тип SDRAM, конструктивно реализованный в модуле DIMM (см. п. 2.2). CKBF (Clock Buffer) — буфер тактовых импульсов для памяти.

Интерфейс с видеоадаптером имеет тип AGP (Accelerated Graphics Port). ***Его основные характеристики:***

- шина данных 32 разряда;
- тактовая частота 66 МГц;
- пропускная способность 266 Мбайт/с.

Для AGP 2X, 4X, 8X частота и скорость умножаются на соответствующий коэффициент. Напряжение питания AGP и AGP 2X — 3,3 В, AGP 4X и 8X — 1,5 В. Интерфейс AGP был разработан в 1996 г. и использовался на протяжении многих лет, пока не был вытеснен более производительным интерфейсом PCI Express.

Северный мост реализует также интерфейс PCI (Peripheral Component Interconnect), разработанный Intel, через который подключаются все остальные устройства, в том числе вторая микросхема чипсета 82371EB, а также платы расширения.

Первая версия шины PCI имела следующие характеристики:

– тактовая частота 33 МГц;

– шина данных 32 разряда;

– пропускная способность 133 Мбайт/с.

PCI 2.1 работает на частоте 33–66 МГц и имеет разрядность 32 или 64 бита. Максимальная пропускная способность (66 МГц, 64 бита) 533 Мбайт/с. Позволяет подключать до десяти внешних устройств. Важным свойством этой шины является автоматическое конфигурирование внешних устройств с использованием технологии «Plug and Play» (Plug-n-Play, PnP). В целом технология PnP широко используется в персональных компьютерах, начиная с операционной системы Windows 95, в которой она и была впервые реализована. Технология PnP позволяет операционной системе автоматически обнаружить и определить новое устройство, установить для него необходимые драйверы и осуществить его конфигурирование. Конечно, эту технологию должны поддерживать и сами устройства, чтобы дать операционной системе необходимую информацию о себе.

Имеется серверный вариант шины PCI — PCI-X, которая может работать с частотой 100, 133, 266 и 533 МГц.

К шине PCI можно подключать через соответствующие разъемы (слоты) различные платы: сетевые адаптеры, видеоадаптеры, звуковые платы и т. п.

Микросхема южного моста 82371EB (PIIX, PCI-to-ISA / IDE Xcelerator) реализует следующие интерфейсы [43]:

– ISA (Industry Standard Architecture);

– IDE (Integrated Drive Electronics);

– USB (Universal Serial Bus);

– SMBus.

Она также содержит:

– контроллер прямого доступа к памяти (два контроллера 82C37, совместимые с i8237);

– контроллер прерываний (два контроллера 82C59, совместимые с i8259A);

– часы реального времени (Real-Time Clock, RTC).

Шина ISA (Industry Standard Architecture) — одна из первых шин персональных компьютеров. Она имеет следующие характеристики:

– частота 8 МГц;
– шина данных 8 или 16 бит;
– шина адреса 24 бита;
– пропускная способность до 16 Мбайт/с.

К слотам шины ISA так же можно подключать различные платы: сетевые, звуковые, видеокарты и т. д.

Интерфейс IDE (Integrated Drive Electronics) по-другому называется ATA (AT Attachment). Разъем — 40 контактов (данные — 16). Имеет возможность подключения до двух устройств. Первоначально использовался для подключения жестких дисков, затем появилось расширение для поддержки иных типов накопителей: приводов CD-ROM, CD-R и DVD-ROM, накопителей LS-120 и ZIP, магнитооптики, стримеров и т. п., — которое называется ATAPI (ATA Packed Interface).

Интерфейс IDE поддерживает два типа обмена данными:

1. Программируемый ввод-вывод (PIO, Programmed Input/Output).
2. Прямой доступ к памяти (DMA, Direct Memory Access).

При программируемом вводе-выводе обмен данными производится через регистры процессора под его непосредственным управлением. В режиме прямого доступа к памяти, обеспечиваемого соответствующим контроллером, контроллер интерфейса IDE пересылает данные между диском и оперативной памятью, не загружая центральный процессор.

Существует несколько разновидностей интерфейса IDE, совместимых снизу вверх друг с другом. Пропускная способность зависит от разновидности интерфейса и способа обмена данными:

– PIO Mode 3 — 11,1 Мбайт/с;
– PIO Mode 4 и Single Word DMA Mode 2 — 16,7 Мбайт/с;
– Multi Word DMA Mode 2 — 20 Мбайт/с;

– Ultra ATA (Ultra DMA, ATA-33, DMA-33) при использовании режима DMA Mode 3 — 33,3 Мбайт/с;

– Ultra ATA/66 — 66 Мбайт/с;

– Ultra ATA/100 — 100 Мбайт/с.

Надо отметить, что в более производительных компьютерах, использовавшихся в качестве серверов, для подключения различных накопителей вместо IDE использовался интерфейс SCSI (Small Computer System Interface). К нему можно было подключить до 7 или 15 устройств, помимо контроллера (хост-адаптера).

Разъемы интерфейса SCSI отличаются большим разнообразием. Их можно разделить на два типа — для подключения устройств внутри и вне системного блока. Количество контактов: 25, 50 или 68. Существует множество разновидностей интерфейса SCSI, отличающихся шириной шины данных — 8 или 16 бит (Wide SCSI) и частотой тактовых импульсов:

– SCSI-1 — 5 МГц, 5 Мбайт/с;

– Fast SCSI (SCSI-2) — 10 МГц, 10 Мбайт/с;

– Fast Wide SCSI — 10 МГц, 20 Мбайт/с;

– Ultra SCSI — 20 МГц, 20 Мбайт/с;

– Ultra Wide SCSI — 20 МГц, 40 Мбайт/с;

– Ultra 2 SCSI — 40 МГц, 40 Мбайт/с;

– Ultra2 Wide SCSI — 40 МГц, 80 Мбайт/с;

– Ultra3 SCSI — 40 МГц, 160 Мбайт/с;

– Ultra-320 SCSI — 80 МГц, 320 Мбайт/с;

– Ultra-640 SCSI — 160 МГц, 640 Мбайт/с.

Интерфейс USB (Universal Serial Bus, универсальная последовательная шина) используется для подключения различных внешних устройств: принтеров, сканеров, Flash-накопителей и т. п. Он появился в 1996 г. До этого в персональных компьютерах был только один тип последовательного интерфейса, так называемый COM-порт (Communication Port) (на рис. 2.13 Ser. Conn., Serial Connector). Это маркетинговое название интерфейса RS-232 или UART (Universal Asynchronous Receiver-Transmitter) — разновидности USART (Univer-

sal Synchronous-Asynchronous Receiver-Transmitter) без синхронного режима. В дальнейшем, как мы увидим, тенденция к переходу от параллельных интерфейсов к последовательным в компьютерных системах будет усиливаться.

В первых двух версиях интерфейса USB для подключения внешних устройств используется четырехпроводный кабель, при этом два провода в дифференциальном включении (т.е. когда сигнал представляет собой разность сигналов на двух проводниках) используются для приема и передачи данных, а два провода — для питания внешнего устройства (максимальная сила тока — 500 мА).

В чипсете i440BX интерфейс USB версий 1.0 и 1.1. USB 1.0 работает в режиме Low-speed со скоростью передачи до 1,5 Мбит/с. Этого достаточно для подключения клавиатуры, мыши, джойстика. В последующих версиях добавлялись новые режимы и росла скорость:

− USB 1.1 (1998 г.): Full-Speed (аудио, сжатое видео) — 12 Мбит/с;

− USB 2.0 (2000 г.): High-speed (видеоустройства, устройства хранения информации) — 480 Мбит/с;

− USB 3.0 (2008 г.) — 5 Гбит/с;

− USB 3.1 (2013 г.) — 10 Гбит/с;

− USB 3.2 (2017 г.) — 20 Гбит/с;

− USB4 (2019 г.) — 40 Гбит/с;

− USB4 2.0 (2022 г.) — 80 Гбит/с.

Вместе с микросхемами северного и южного моста используется еще одна микросхема — контроллер ввода-вывода. На рисунке 2.12 она обозначена как ULTRA I/O. Это общее название микросхем такого типа. В других случаях она может называться Super I/O, Advanced I/O и т. п. Такие микросхемы выпускает множество различных производителей с конкретными наименованиями, например, IT8673F компании ITE (Integrated Technology Express). Контроллер ввода-вывода включает все остальные интерфейсы и контроллеры, необходимые в персональном компьютере (рис. 2.14) [44]:

− последовательные порты (COM-порты) UART;

– параллельный порт;
– контроллер дисковода гибких дисков (флоппи-дисковода);
– контроллер клавиатуры и др.

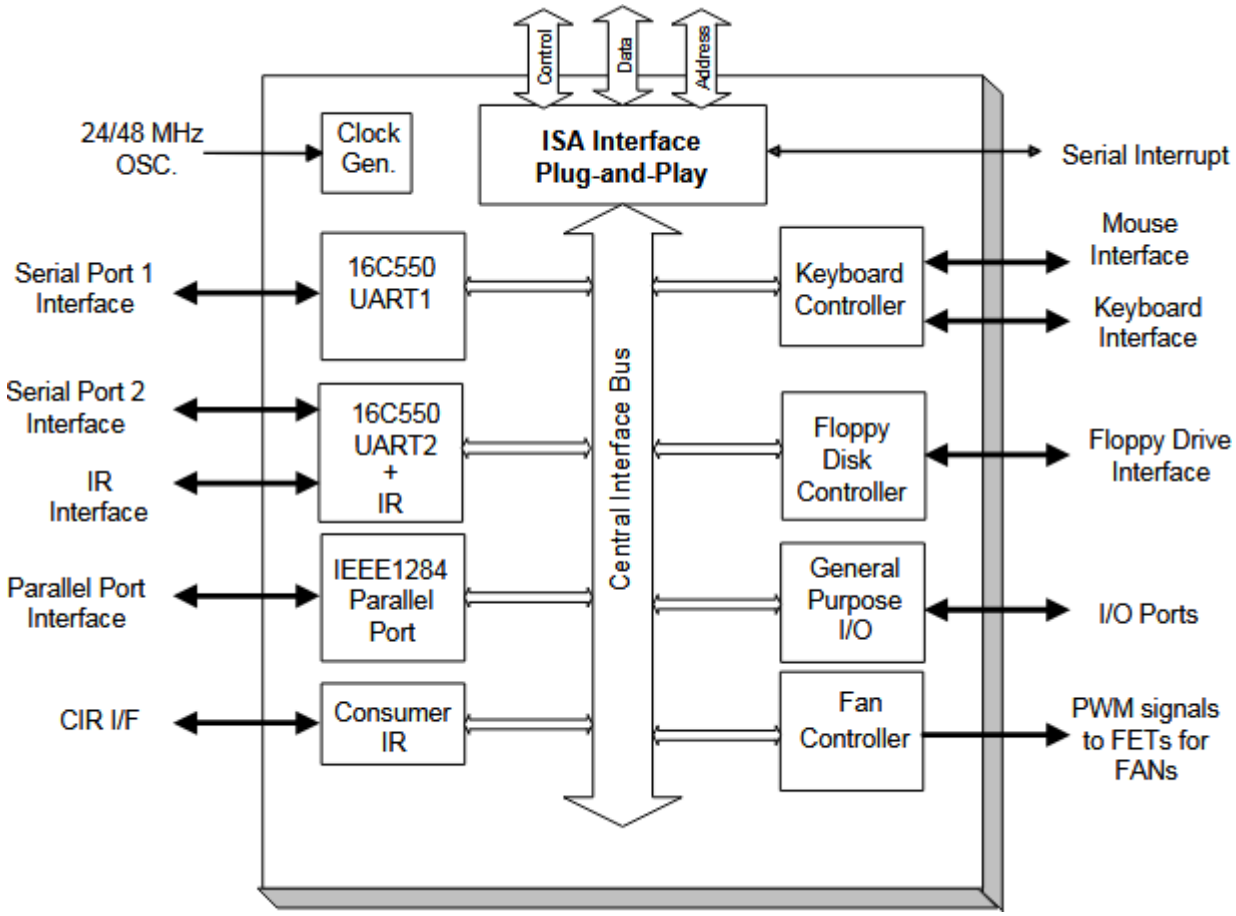


Рис. 2.14. Структурная схема контроллера ввода-вывода

Как видно из рис. 2.14, контроллер клавиатуры реализует также и интерфейс с мышью.

Микросхема ROM BIOS (Flash BIOS) (см. рис. 2.13) также подключена через интерфейс ISA и связана с контроллером ввода-вывода шиной X-Bus. Через ту же шину подключена микросхема LM79, монитор аппаратного обеспечения (Hardware Monitor), которая контролирует состояние вентиляторов и т. п.

В заключение этого параграфа еще раз отметим такую важную особенность структуры компьютера, как ее иерархичность (см. рис. 1.14). Она так же четко проявляется и на рис. 2.13 — одни ин-

терфейсы ЭВМ подключаются через другие (USB, IDE, ISA через PCI; интерфесы, реализуемые контроллером ввода-вывода — через ISA и PCI), а многие устройства ввода-вывода и даже запоминающие устройства, в частности, ROM BIOS, связаны с процессором через цепочку интерфейсов.

Развитие архитектуры ПЭВМ

Популярный предшественник i440BX чипсет i82430FX (Triton), поддерживающий процессор Pentium, включает четыре микросхемы [45]:

− 82437FX, системный контроллер (TSC, Triton System Controller), вырабатывает управляющие сигналы для кэш-памяти второго уровня и оперативной памяти, в том числе сигналы мультиплексирования адреса;

− 82438FX, шинный формирователь (TDP, Triton Data Path device), 2 шт., осуществляет буферирование шины данных, памяти и ввода-вывода, под управлением 82437FX — переключение между байтовым и двухбайтовым режимами;

− 82371FB, мост или ускоритель (PIIX, PCI ISA/IDE Accelerator bridge chip), предоставляет интерфейс между шинами PCI и ISA, двухканальный интерфейс IDE, способный поддерживать до четырех устройств. Включает в себя семь 32-разрядных DMA-каналов, пять 16-разрядных таймеров, два 8-канальных контроллера прерываний, схему отображения прерываний PCI-AT, NMI-логику, схему регенерации адресов ISA, схему арбитража шин ISA-PCI.

Помимо микросхем чипсета, как было отмечено выше, материнская плата содержит контроллер ввода-вывода. Для чипсета Triton это микросхема PC87306, в состав которой входят следующие компоненты:

− два COM-порта UART;

− параллельный порт;

− контроллер дисковода гибких дисков;

− часы реального времени (RTC, Real Time Clock) и CMOS RAM емкостью 128 байт (14 байт для часов, 114 — для установок программы SETUP BIOS);

− 8042-совместимый контроллер клавиатуры.

В чипсете i440BX, как уже было сказано, число основных микросхем уменьшилось до двух. Сформировалась архитектура северного и южного моста, которая сохранялась в основных чертах на протяжении долгого времени. Ее развитие происходило в направлении улучшения характеристик контроллеров, интерфейсов с внешними устройствами и новыми типами оперативной памяти, разработки новых интерфейсов. Было выпущено множество чипсетов для поддержки процессора Pentium III с соответствующими улучшенными характеристиками. Но Pentium III, как и Pentium II, имеет микроархитектуру P6 (см. п. 3.2). Более существенные изменения начали происходить с выпуском процессора Pentium 4 с микроархитектурой NetBurst. Однако и здесь архитектура северного и южного моста сохранялась еще долгое время. Для поддержки Pentium 4 был разработан чипсет i850 с двумя микросхемами:

1. 82850 — MCH, контроллер-концентратор памяти [46].
2. 82801BA — ICH, контроллер-концентратор ввода-вывода [47].

Основные характеристики чипсета i850:

− системная шина с эффективной частотой передачи данных 400 МГц;

− поддержка двухканальной памяти RDRAM (RAMBUS DRAM) с пропускной способностью 3,2 Гбайт/с;

− AGP 4X;

− два контроллера и 4 порта USB 1.1;

− интерфейс IDE — Ultra ATA/100.

Pentium 4 имеет так называемую шину с учетверенной накачкой (Quad Pumped Bus), которая позволяет за один такт осуществлять четыре транзакции и при частоте генератора 100 МГц передавать данные с эффективной частотой 400 МГц (см. п. 3.2). Под эту характеристику процессора была разработана память RDRAM и чипсет с ее

поддержкой. Однако память RDRAM из-за дороговизны не получила распространения, и в дальнейшем были разработаны чипсеты i845 — i875 с поддержкой других типов ОЗУ. Их основные модификации и особенности:

− i845, i845S — память SDRAM;

− i845E — частота шины 133 (533) МГц;

− i845D — память DDR;

− i845PE — память PC2700 (DDR333);

− i845G — северный мост со встроенным видеоконтроллером;

− i865, i875 — шина процессора 200 (800) МГц, двухканальная память DDR400 (ширина канала 128 бит или 16 байтов), AGP 8x, Serial ATA, RAID.

Память DDR оказалась наиболее оптимальным вариантом, золотой серединой между SDRAM и RDRAM, и в дальнейшем именно она получила развитие. Цифры в обозначении памяти имеют следующее значение: DDR333 — частота передачи, МГц; PC2700 — пропускная способность, Мбайт/с: 333 МГц * 8 (ширина канала в байтах) ≈ 2 700 Мбайт/с. DDR400 соответствует PC3200 на один канал.

В компьютере появился еще один последовательный интерфейс — SATA (Serial ATA). Он является развитием параллельного интерфейса ATA (IDE) и, соответственно, используется для подключения накопителей. Сейчас существует уже третья версия этого интерфейса. Скорость передачи SATA:

− SATA 1.x: 1,5 Гбит/с;

− SATA 2.x: 3 Гбит/с;

− SATA 3.x: 6 Гбит/с.

Надо отметить, что после появления интерфейса SATA предыдущий интерфейс ATA стали называть PATA (Parallel ATA).

В более производительных системах для подключения накопителей используется последовательный интерфейс SAS (Serial Attached SCSI):

− SAS-1: 3 Гбит/с (2004 г.);

− SAS-2: 6 Гбит/с (2009 г.);

– SAS-3: 12 Гбит/с (2013 г.);

– SAS-4 (24G): 22,5 Гбит/с (2017 г.);

– SAS-5: 45 Гбит/с (разрабатывается).

Он является развитием параллельного интерфейса SCSI.

Также в персональных компьютерах появляется технология RAID, разработанная уже задолго до этого для серверных систем. RAID (Redundant Array of Inexpensive Disks, избыточный массив недорогих дисков, или позднее Redundant Array of Independent Disks, избыточный массив независимых дисков) — технология хранения данных, которая объединяет несколько компонентов физических дисков в одну или несколько логических единиц с целью повышения избыточности и, соответственно, надежности хранения данных или повышения производительности. Это название было выбрано для контраста с предыдущей концепцией высоконадежных дисковых накопителей для мейнфреймов — SLED (Single Large Expensive Disk, один большой дорогой диск). Накопители с технологией RAID называются RAID-массивами. Данные в них распределяются по дискам одним из нескольких способов, называемых схемами или уровнями RAID, в зависимости от требуемой избыточности, емкости и производительности. Уровни RAID выше RAID 0, обеспечивают защиту от неисправимых ошибок секторов, а также сбоев целых физических дисков.

Состав чипсета i875P (рис. 2.15) [48]:

1. 82875P — MCH.
2. 82801EB (ICH5) или 82801ER (ICH5R).

MCH содержит:

− контроллер двухканальной памяти DDR;

− интерфейс AGP 8x/4x;

− интерфейс CSA (Communication Streaming Architecture) для подключения гигабитного сетевого адаптера.

Интерфейс между микросхемами чипсета — HI (Hub Interface).

ICH реализует интерфейсы:

− USB 2.0;

− GPIO (General Purpose Input Output), интерфейс ввода-вывода общего назначения, его линии не специализированы и могут быть использованы производителями плат для реализации дополнительной функциональности;

− Serial ATA (SATA), последовательный ATA;

− параллельный ATA (IDE);

− AC '97 (Audio Codec '97) для подключения микросхемы аудиокодека;

− LPC (Low Pin Count) для подключения Flash-ROM с BIOS и контроллера ввода-вывода (SIO, Super I/O);

− PCI;

− SMBus (специализация двухпроводного интерфейса I2C).

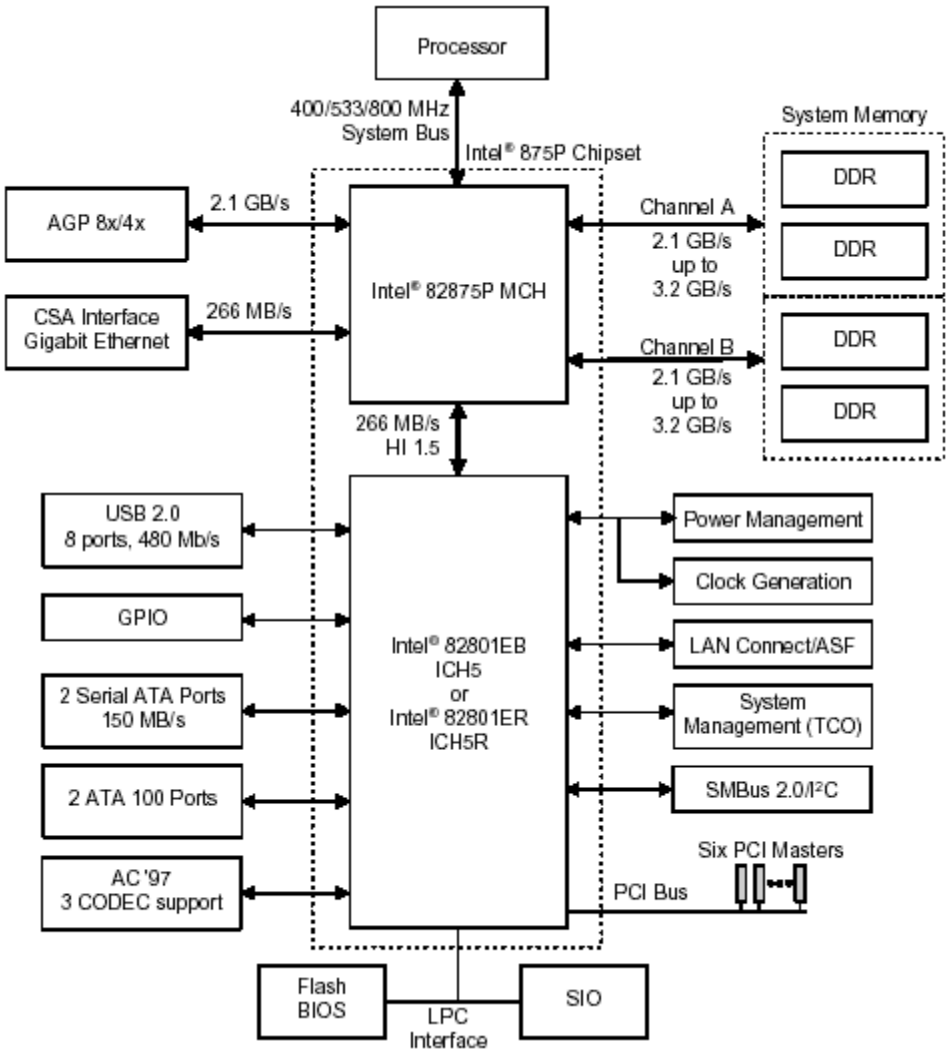


Рис. 2.15. Структурная схема системы с чипсетом i875P

AC '97 — спецификация интерфейса для подключения аудиокодека (AC), модемного кодека (MC) и/или аудио/модемного кодека (AMC) к цифровому контроллеру, находящемуся в ICH, посредством линий связи, называемых AC-link. Она была разработана Intel в 1997 году и стала одним из первых широко распространенных решений в области интегрированного звука. Наличие этого интерфейса и соответствующих аппаратных элементов позволяет отказаться от использования в компьютере отдельной (дискретной) звуковой платы.

Скорость передачи различных каналов системы на чипсете i875P приведена в табл. 2.2.

Таблица 2.2

Скорость передачи интерфейсов чипсета i875P

Интерфейс	**Тактовая частота, МГц**	**Число транзакций за такт**	**Размер шины данных, бит**	**Скорость передачи**
System Bus	100/133/200	4	64	3,2 / 4,3 / 6,4 Гбайт/с
DDR	133/166/200	2	64	2,1 / 2,7 / 3,2 Гбайт/с (на канал)
AGP 8x	66	8	32	2,1 Гбайт/с
CSA	66	4	8	266 Мбайт/с
HI	66	4	8	266 Мбайт/с
PCI 2.3	33	1	32	133 Мбайт/с
ATA/100	до 44,444 запись до 50 чтение	1	16	до 88,9 Мбайт/с (запись) до 100 Мбайт/с (чтение)
SATA	750	2	1	1,5 Гбит/с
USB 2.0 High-speed	до 240	до 2	1	480 Мбит/с

В последующих чипсетах появляются новые интерфейсы (рис. 2.16) [49]:

– PCI Express (PCIe) для подключения видеоадаптера и других плат;

– DMI (Direct Media Interface) для соединения MCH и ICH.

Оба интерфейса разработаны Intel и являются последовательными.

Чипсет i955X Express был разработан для поддержки процессора Pentium Extreme Edition, с архитектурой Intel 64. В его состав входят:

1. 82955X: MCH, корпус FCBGA-1202 (Flip Chip Ball Grid Array).
2. 82801GB/82801GR: ICH7, корпус MBGA-652 (Metal Ball Grid Array).

Между собой они соединены интерфейсом DMI.

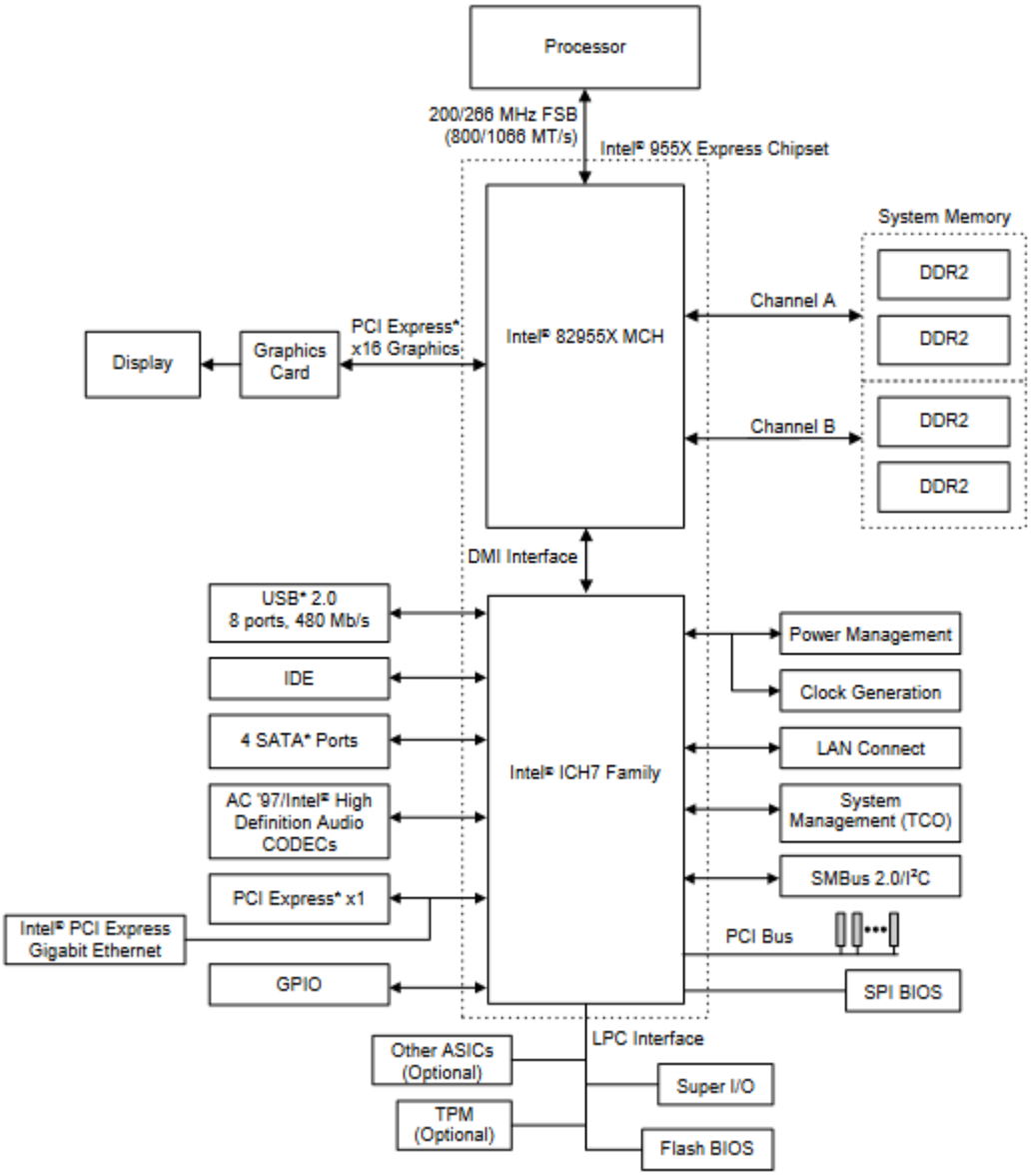


Рис. 2.16. Структурная схема системы
с чипсетом i955X Express

Большинство элементов, отраженных на схеме, присутствовали и в предыдущих чипсетах. Но здесь в качестве опциональных приведены два новых элемента:

− ASIC (Application-Specific Integrated Circuit), интегральная схема специального назначения, например, видеокодек;

− TPM (Trusted Platform Module), криптографический процессор.

Кроме того, в архитектуре персонального компьютера появляется новая спецификация аудиосистемы — High Definition Audio (HD Audio) (см. п. 4.3).

Наиболее важным интерфейсом, появившимся на этом этапе, является PCIe. В отличие от интерфейса PCI PCIe является последовательным. Передача данных может осуществляться одновременно в обе стороны, т. е. линия передачи данных PCIe является полнодуплексной (двунаправленной). В большинстве последовательных интерфейсов по одной линии передаются и информационные биты (данные), и служебные (контрольные или управляющие). Поэтому в таких интерфейсах различаются две характеристики: скорость передачи и пропускная способность. Пропускная способность меньше скорости передачи. Скорость передачи (bitrate, transfer rate) — количество любых передаваемых битов, т.е. транзакций, в секунду (Т/с). Пропускная способность (bandwidth, throughput) — количество передаваемых битов или байтов данных в секунду (бит/с или Б/с). Но надо заметить, что иногда в документации и скорость передачи указывается в битах в секунду. В таких случаях имеются в виду все биты, и информационные, и служебные. В PCIe 1.0 в одной последовательной посылке 10 битов, из них 8 — биты данных (кодирование 8b/10b). Скорость передачи — 2,5 ГТ/с, соответственно, пропускная способность в одну сторону — 2 Гбит/с. У последующих версий скорость передачи и пропускная способность увеличивались примерно вдвое.

Характеристики интерфейса DMI примерно такие же, как и PCIe.

Затем в составе северного моста появился графический процессор (рис. 2.17) [50]. Такие чипсеты были разработаны уже для процессоров следующего поколения с микроархитектурой Core, хотя и поддерживали некоторое время процессоры Pentium.

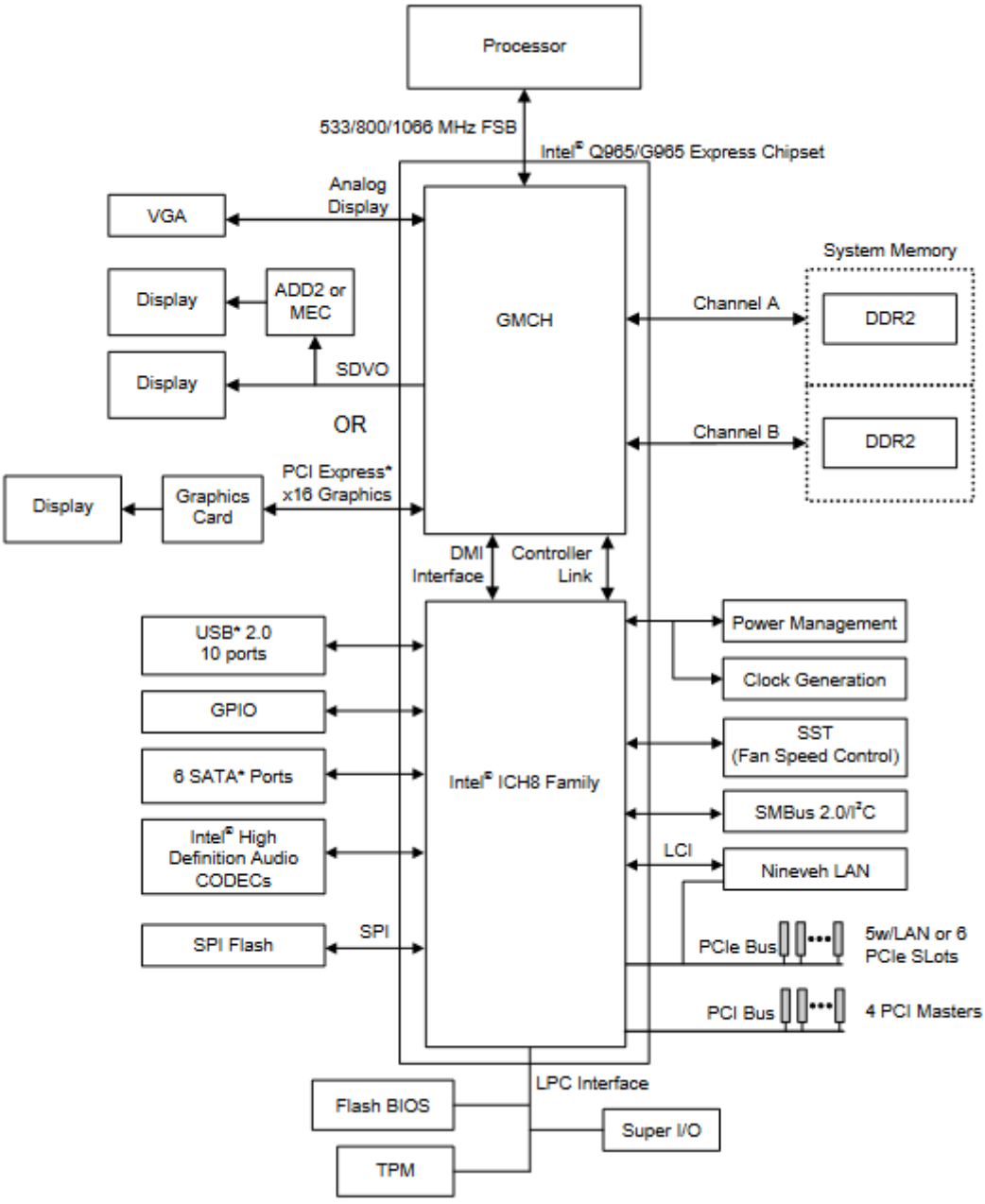


Рис. 2.17. Структурная схема системы
с чипсетом Q965/G965 Express

Микросхема северного моста с интегрированным графическим процессором называется **GMCH** (Graphics Memory Controller/Hub) — контроллер-концентратор памяти и графической подсистемы.

Конечно, по-прежнему можно использовать монитор с видеоадаптером, подключаемым в слот PCIe x16. Но имеются и варианты подключения монитора к интегрированному видеоадаптеру:

− через обычный аналоговый разъем VGA (Video Graphics Array) (другое название D-sub);

− с использованием SDVO (Serial Digital Video Out) непосредственно или через плату ADD2 (Advanced Digital Display Card) или MEC (Media Expansion Card).

SDVO — это технология, разработанная Intel, позволяющая использовать PCIe x16 для добавления дополнительных интерфейсов видеосигналов, таких, как выходы мониторов VGA и DVI (Digital Visual Interface), телевизионные выходы SDTV (Standard-Definition Television) и HDTV (High-Definition Television) или входы ТВ-тюнера, на системную плату, содержащую встроенный графический процессор Intel.

Для подключения микросхемы Flash-ROM BIOS в чипсете Q965/G965 Express используется интерфейс SPI (Serial Peripheral Interface).

Контроллер-концентратор платформы (PCH)

В дальнейшем в процессор (Core i7 с микроархитектурой Intel Nehalem) интегрируется контроллер памяти, и необходимость в MCH отпадает (рис. 2.18).

Здесь вместо MCH появляется IOH (Input-Output Hub). Процессор связан с ним интерфейсом QPI (QuickPath Interconnect). QPI (ранее Common System Interface, CSI) – последовательный интерфейс типа «точка-точка», разработанный Intel для соединения процессоров в многопроцессорных системах между собой, а также процессора с главной микросхемой чипсета.

Затем в процессор (Core i3 и Core i5 с той же микроархитектурой Intel Nehalem) интегрируются графический процессор и интерфейс PCIe, и в составе чипсета остается только одна микросхема —

контроллер-концентратор платформы (PCH, Platform Controller-Hub) (рис. 2.19).

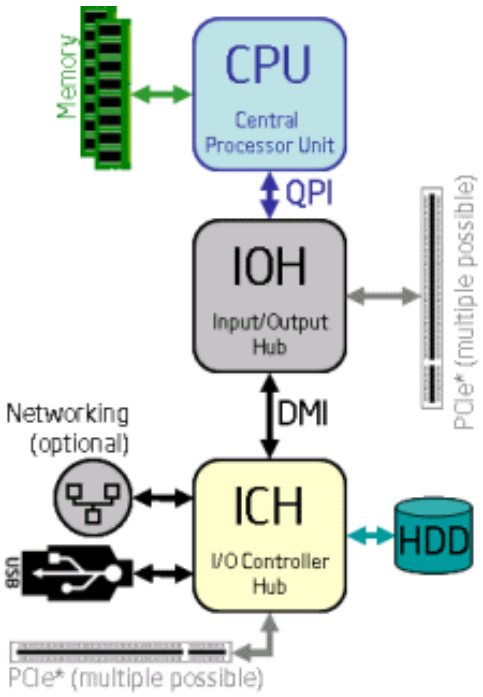


Рис. 2.18. Структура ПЭВМ на базе процессора
с интегрированным контроллером памяти

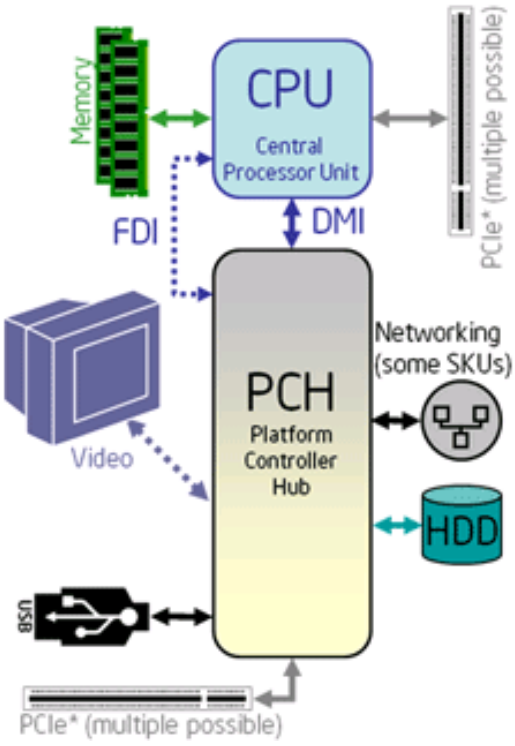


Рис. 2.19. Структура ПЭВМ с PCH

Функциональность северного моста распределилась между процессором и южным мостом. Контроллер памяти и интерфейс PCIe вошли в состав процессора, а PCH реализует все остальное.

Процессор с интегрированным GPU (Intel HD Graphics) дополнительно имеет интерфейс FDI (Flexible Display Interface). Для использования функций интегрированного GPU интерфейс FDI должен иметь и PCH. FDI специально разработан Intel для связи интегрированного графического процессора с PCH, к которому через соответствующие разъемы подключается монитор.

В настольных системах в настоящее время используется так называемая двухчиповая платформа (рис. 2.20) [51].

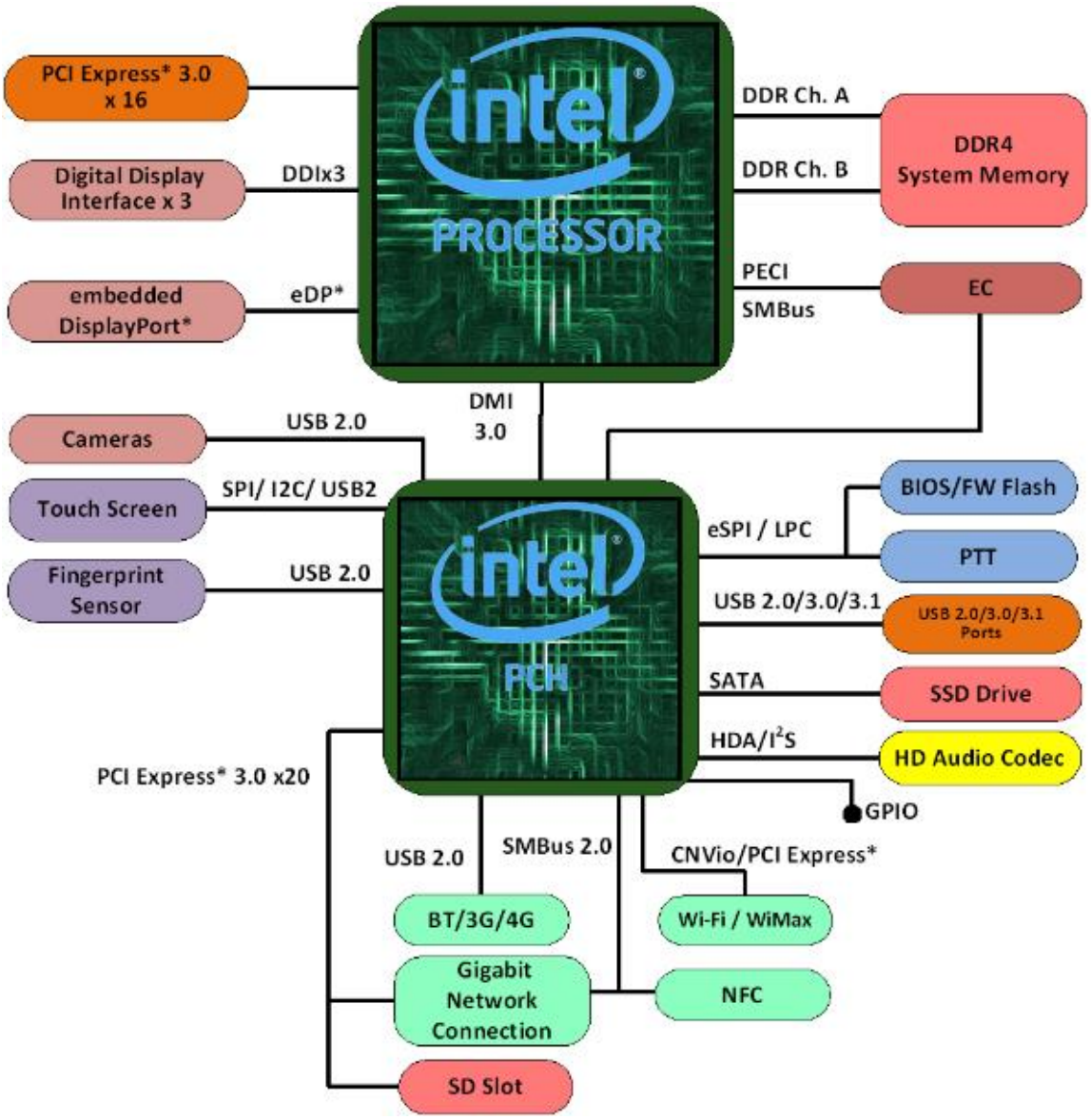


Рис. 2.20. Двухчиповая платформа: CPU и PCH

В двухчиповой платформе две основные микросхемы: процессор H- или S-типа и микросхема PCH (Platform Controller-Hub). H- и S-процессоры (для 8-го поколения процессоров Core i) имеют энергопотребление от 35 до 95 Вт.

В мобильных системах используется одночиповая платформа (рис. 2.21) [51].

В одночиповой платформе и процессор, и контроллер-концентратор платформы находятся в одном корпусе. В ней используется процессор U-типа (ультрамобильный), он имеет более низкое энергопотребление и большее количество выводов.

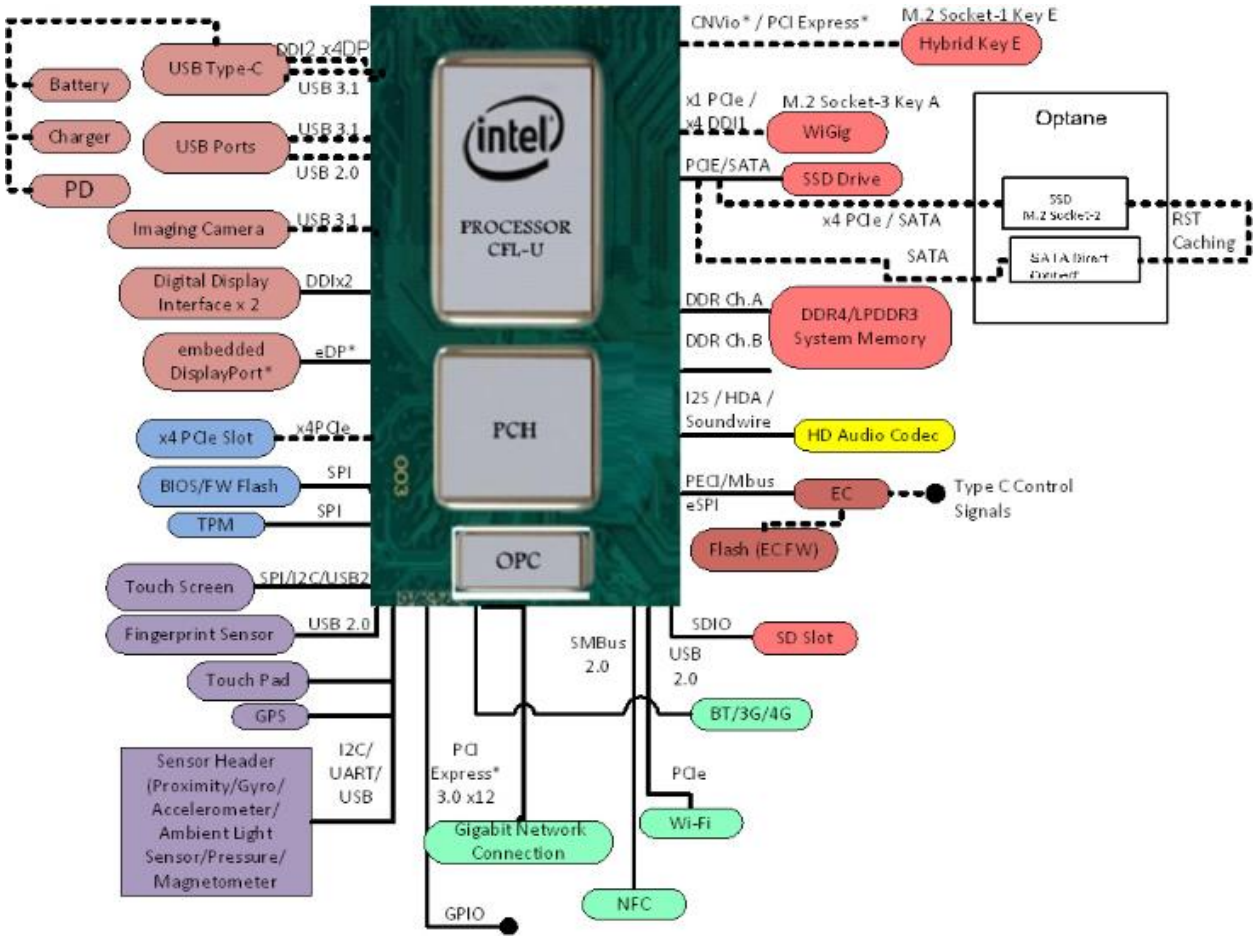


Рис. 2.21. Одночиповая платформа

Также в одном корпусе с процессором и PCH находится кэш-память OPC (On-Package Cache).

И в той, и в другой платформе используется двухканальная память DDR. Интерфейс с нею реализует интегрированный в состав процессора контроллер памяти (Integrated Memory Controller, IMC) (см. п. 4.2).

Интерфейс с памятью был и остается параллельным, но из интерфейсов ввода-вывода остались только последовательные, которые в настоящее время являются основными в компьютерных системах: PCIe, SATA, USB, SPI, I2C. Через них подключается множество устройств и других интерфейсов, в том числе, сетевых:

− Wi-Fi;

− Bluetooth (BT);

− 3G/4G;

− NFC;

− Gigabit Network.

Wi-Fi — технология радиосвязи для подключения устройств к локальной компьютерной сети и доступа в Интернет с использованием маршрутизаторов и точек доступа, основанная на семействе стандартов IEEE 802.11. Максимальная скорость передачи в соответствии с последним стандартом 802.11be (2023 г.) — до 30 Гбит/с. Wi-Fi — это наиболее распространенная технология, она используется повсеместно: в домашних сетях, на предприятиях, в организациях и общественных местах.

Bluetooth — технология радиосвязи ближнего действия с использованием передатчиков небольшой мощности (до 2,5 мВт), используемая для обмена данными между устройствами на небольших расстояниях (до 10 метров). В основном используется для обмена файлами между находящимися поблизости мобильными устройствами, подключения беспроводных наушников к смартфонам и плеерам и т. п.

NFC (Near-Field Communication) — технология индуктивной связи между двумя электронными устройствами на расстоянии до 10 см. NFC-соединение происходит автоматически, когда два устройства оказываются в радиусе действия друг друга. Устройство с NFC может использоваться как платежная карта. Два устройства с поддерж-

кой NFC могут обмениваться файлами и другой информацией, например, параметрами настройки соединения Bluetooth или Wi-Fi и др.

3G, 4G — интерфейсы мобильной сотовой связи, обозначаемые номерами поколений (Generation). Каждое из поколений мобильной связи характеризуется используемыми стандартами, технологиями и основными показателями:

– 1G: аналоговая мобильная связь;

– 2G: цифровая связь, основной стандарт — GSM (Global System for Mobile Communications), в котором используется технология множественного доступа с разделением каналов по времени (TDMA, Time Division Multiple Access) и частоте (FDMA, Frequency Division Multiple Access);

– 2.5G: основной стандарт — GPRS (General Packet Radio Service) — надстройка над GSM, представляющая собой технологию пакетной передачи данных, позволяющую производить обмен данными с другими устройствами в сети GSM и с внешними сетями, в том числе Интернетом;

– 2.75G: основной стандарт — EDGE (Enhanced Data rates for GSM Evolution) — развитие GPRS, отличающееся способом кодирования данных, позволяющим за один промежуток времени передавать больший объем данных;

– 3G: основные стандарты UMTS (Universal Mobile Telecommunications System), W-CDMA (Wideband Code Division Multiple Access) — также надстройка над GSM, технология, использующая широкополосный множественный доступ с кодовым разделением каналов, в отличие от TDMA и FDMA, с максимальной скоростью приема (абонентом) 42 Мбит/с и передачи 11 Мбит/с (при использовании HSPA+ (High Speed Packet data Access), 3.75G);

– 4G: основной стандарт — LTE (Long-Term Evolution) — развитие технологий GSM/EDGE и UMTS/HSPA для более высокоскоростного обмена данными, максимальная скорость приема — 3 Гбит/с.

В настоящее время идет внедрение технологии 5G. Максимальная скорость передачи с использованием этой технологии оценивает-

ся в 20 Гбит/с. При этом поддерживается до 1 миллиона устройств на 1 км^2, что важно для Интернета вещей.

Gigabit Network — проводной интерфейс, посредством которого компьютеры подключаются к локальной сети Ethernet. Ethernet — это семейство технологий для соединения компьютеров в компьютерные сети с использованием пакетной передачи данных. Ethernet первоначально использовал шинную топологию с использованием коаксиальных кабелей RG-58. Информация, передаваемая одним узлом, одновременно принималась всеми остальными узлами, что было сходно с радиовещанием. Отсюда и название Ethernet («Эфирная сеть»). Сейчас сети Ethernet имеют многоуровневую звездообразную топологию. Компьютеры соединяются с использованием таких типов устройств, как маршрутизатор (router), коммутатор (switch), концентратор (hub). В качестве соединителей используются витая пара и оптоволокно.

2.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 2

Задание 1. Конфигурация ПЭВМ

Подробную информацию о составе аппаратных средств персонального компьютера можно получить с помощью специализированных программных средств — информационно-диагностических утилит.

Выделяется три типа информации об аппаратном обеспечении компьютера:

− конфигурация: перечень и характеристики аппаратных средств;

− данные мониторинга — показания датчиков: температуры CPU и GPU, тактовой частоты CPU, GPU и памяти, скорости вращения вентиляторов, потребляемой мощности и т. д.;

− данные тестов производительности (технико-эксплуатационные показатели): производительность процессора, подсистемы памяти, компьютера в целом и т. п.

Некоторые утилиты предоставляют информацию только одного типа, другие — нескольких типов. Они различаются объемом предоставляемой информации, формой ее представления, множеством тестов производительности.

Популярные свободно распространяемые информационно-диагностические программы:

– CPU-Z: конфигурация, тесты производительности (benchmark) (рис. 2.22);

– GPU-Z: конфигурация графической подсистемы, мониторинг (рис. 2.23);

– HWiNFO: подробная информация о конфигурации, мониторинг (рис. 2.24);

– Speccy: конфигурация, мониторинг (рис. 2.25).

Популярные платные программы:

– AIDA64: конфигурация, мониторинг, тесты производительности (рис. 2.26);

– SiSoftware Sandra (System Analyzer, Diagnostic and Reporting Assistant): конфигурация, мониторинг, тесты производительности (рис. 2.27);

– SysInfo Detector: конфигурация, мониторинг (рис. 2.28).

Имеются бесплатные демонстрационные версии программ AIDA64, SiSoftware Sandra — SiSoftware Sandra Lite, SysInfo Detector. Они имеют определенные ограничения, но вполне могут быть использованы для выполнения практических заданий этого курса.

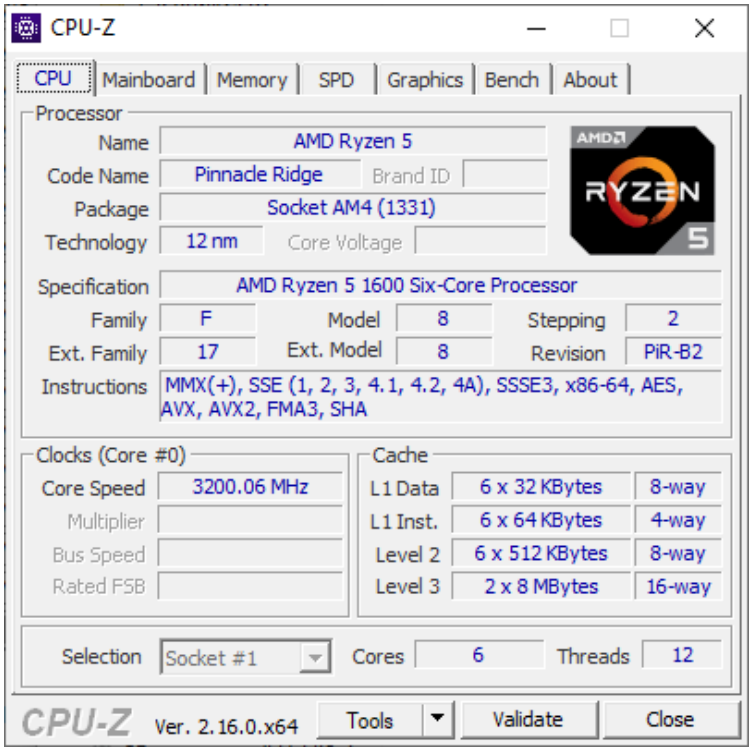


Рис. 2.22. Окно программы CPU-Z

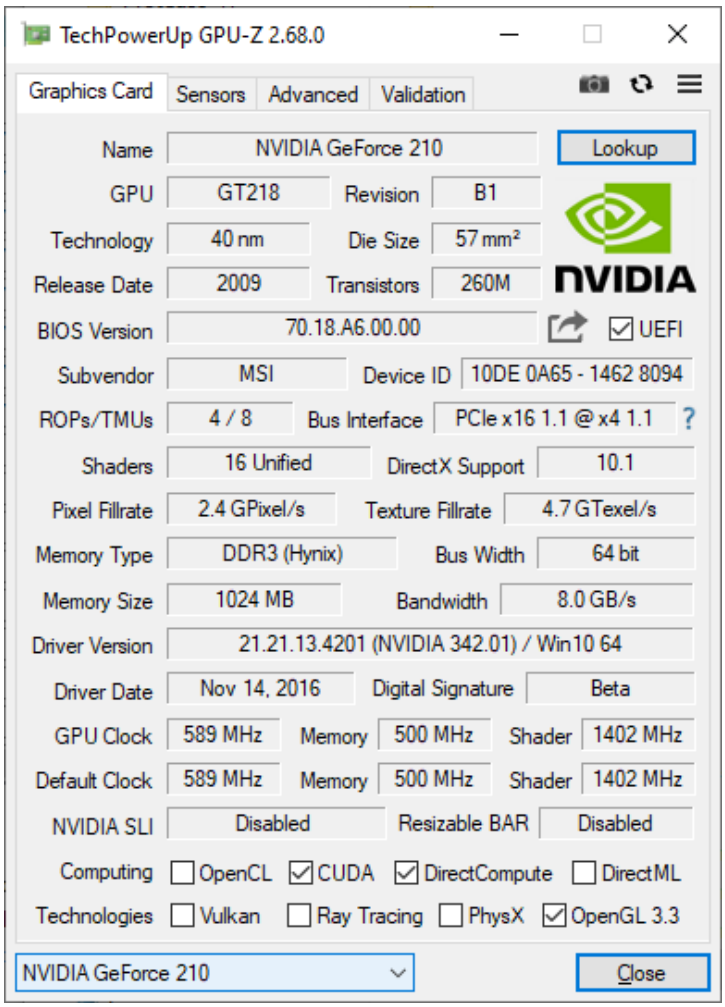


Рис. 2.23. Окно программы GPU-Z

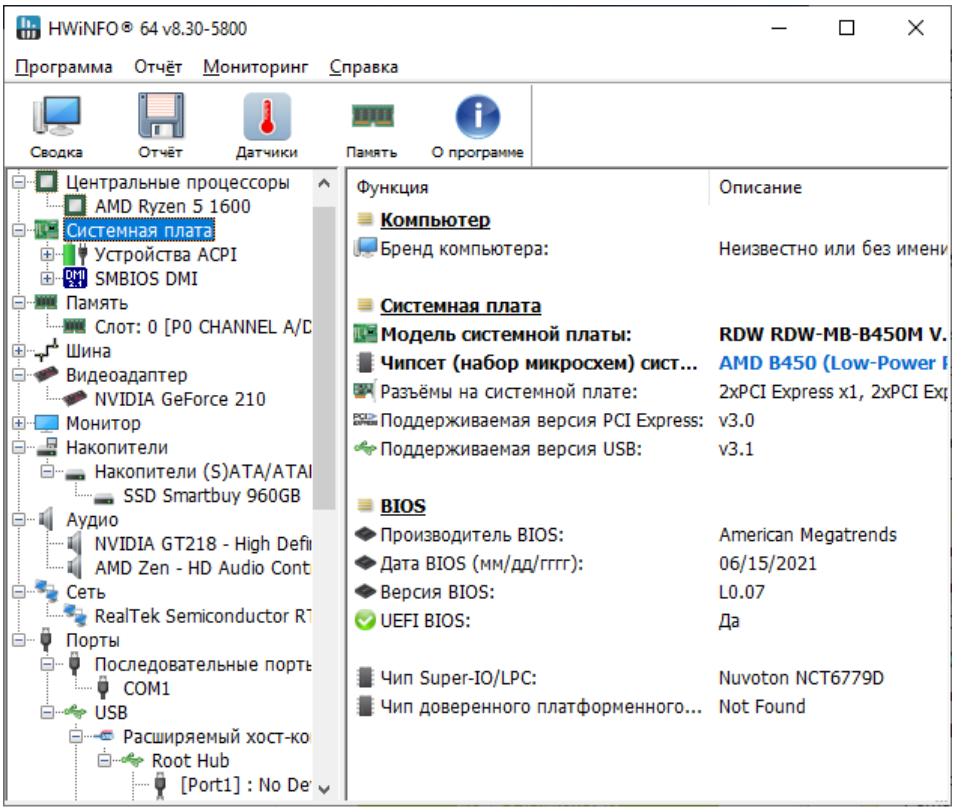


Рис. 2.24. Главное окно HWiNFO

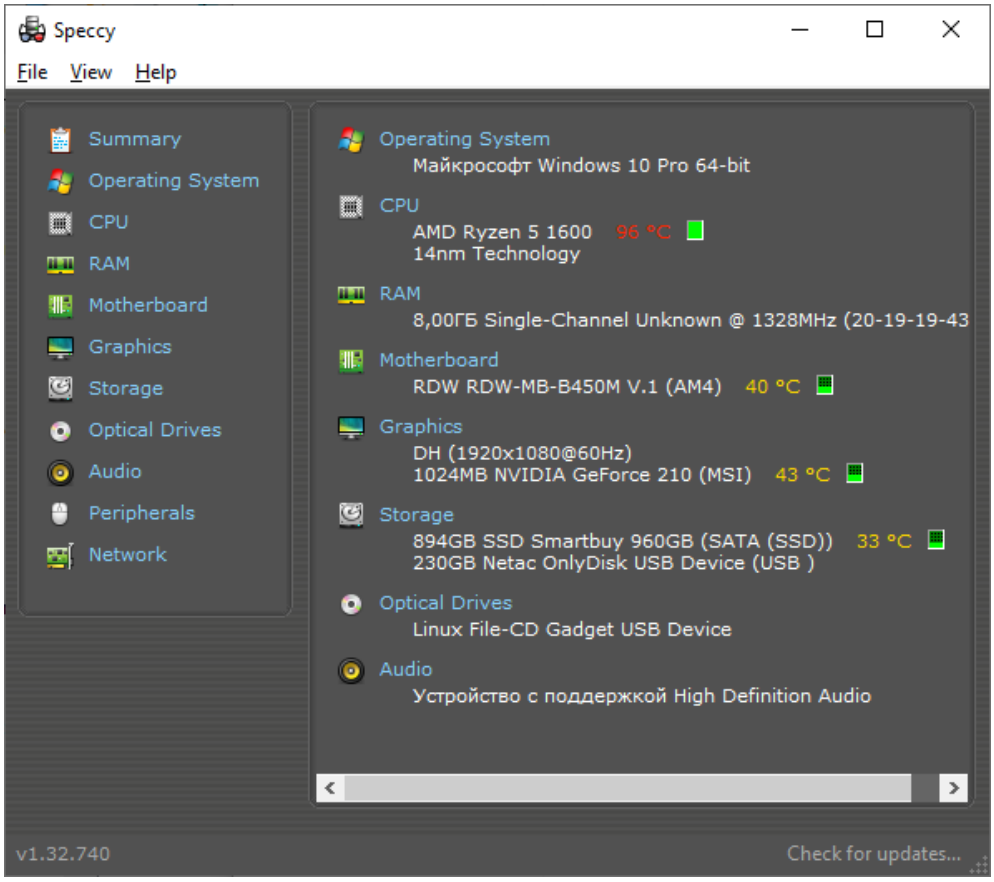


Рис. 2.25. Главное окно Speccy

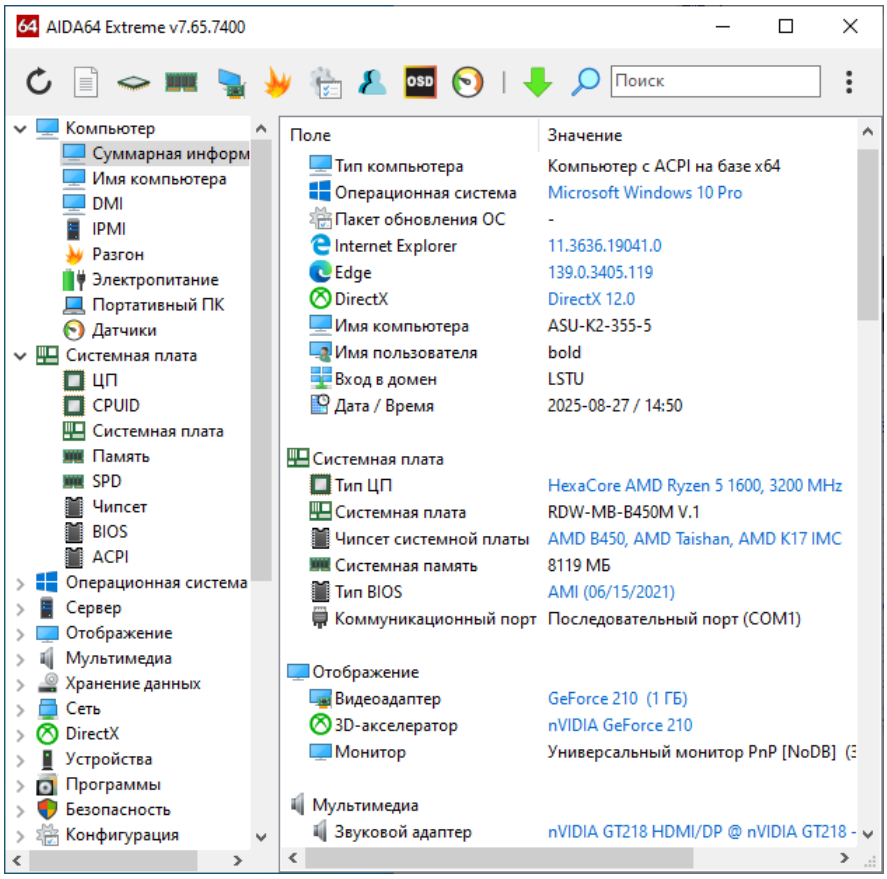


Рис. 2.26. Главное окно AIDA64

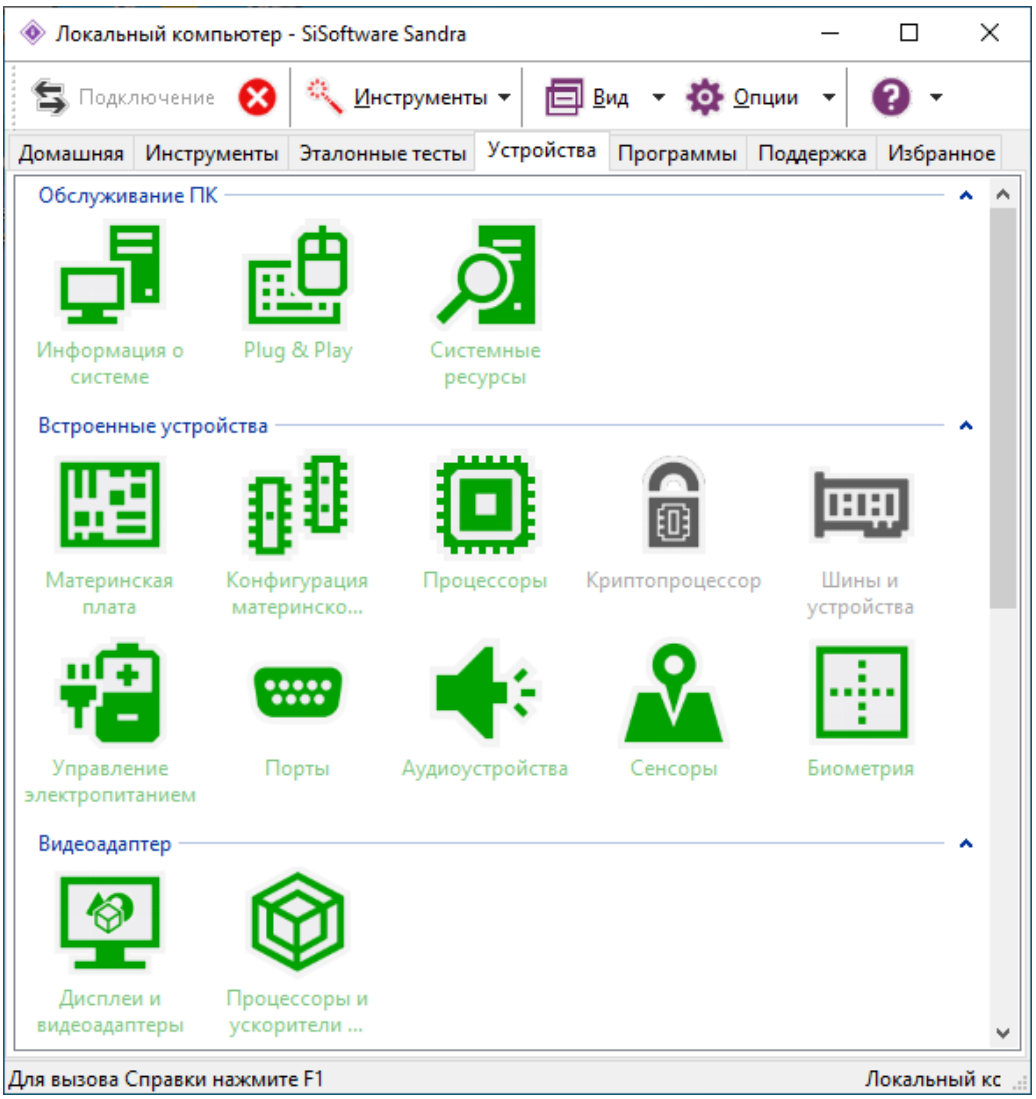


Рис. 2.27. Главное окно SiSoftware Sandra Lite
(активна вкладка «Устройства»)

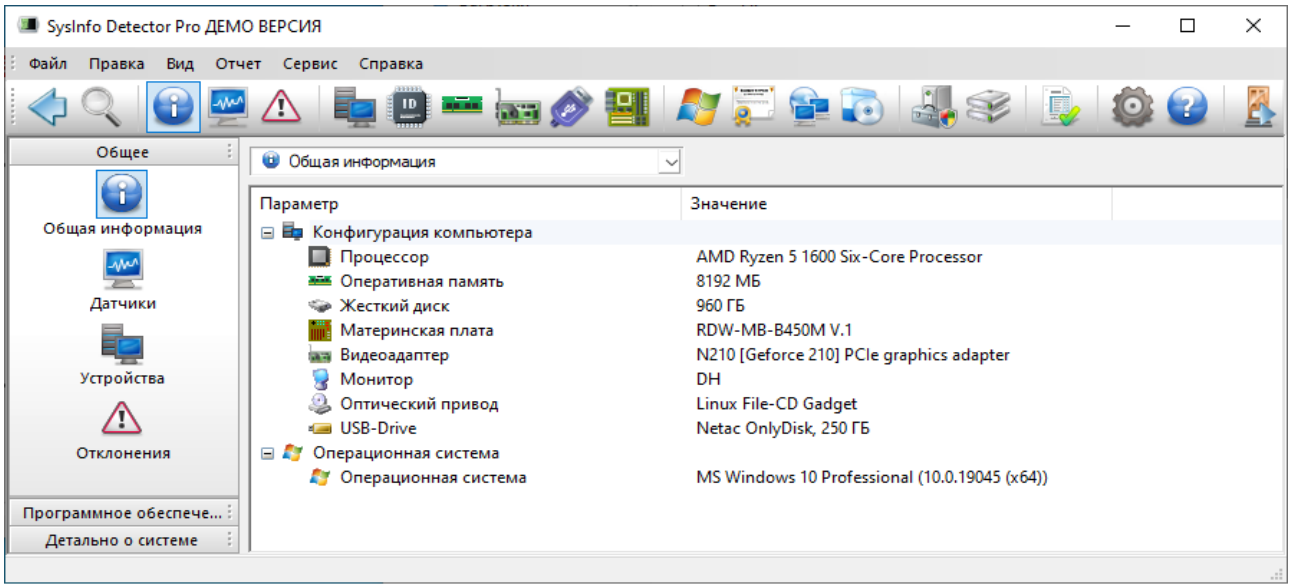


Рис. 2.28. Главное окно SysInfo Detector

В этом задании с помощью одной или нескольких из перечисленных программ необходимо получить общую информацию о конфигурации компьютера:

- процессор: модель, основные характеристики;
- чипсет: модель, состав, характеристики;
- оперативная память: тип, объем, характеристики;
- постоянная память (ROM BIOS): объем;
- устройства внешней памяти: объем, тип, характеристики;
- устройства ввода-вывода: видеоадаптер и монитор, звуковые устройства и др.

При этом нужно сделать оценку полученных численных показателей: пропускной способности системной шины и канала оперативной памяти. Для этого необходимо рассчитать их значения по тактовой частоте и ширине канала и сравнить с выданными используемой программой.

В HWiNFO для получения общей информации о конфигурации служит кнопка «Сводка» в панели инструментов (см. рис. 2.24). При ее нажатии открывается окно «Сводка системы» (рис. 2.29).

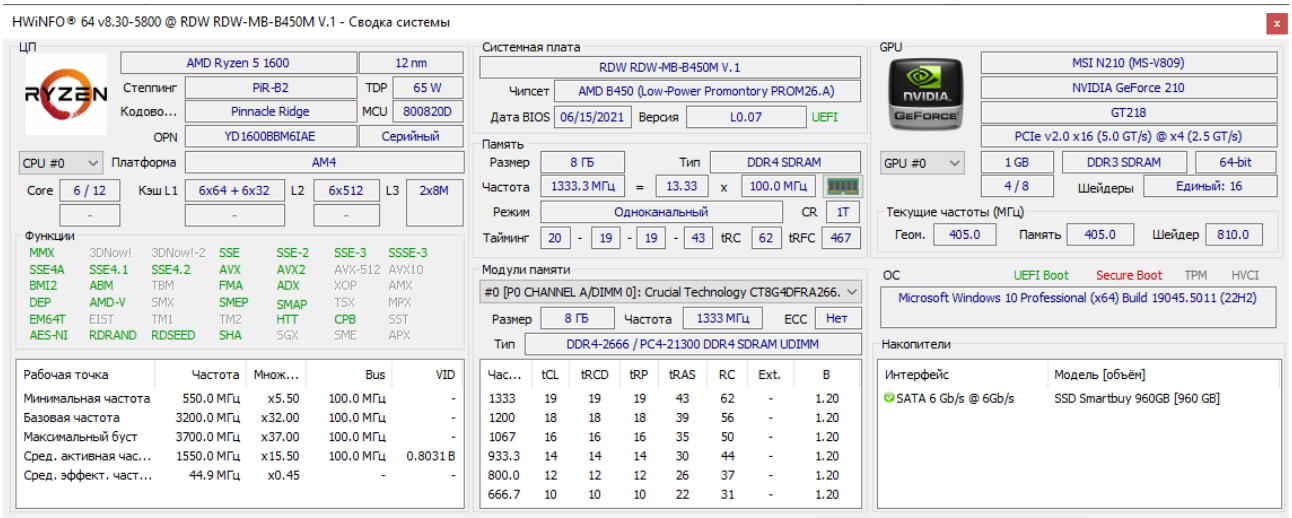


Рис. 2.29. Сводная информация о компьютере в HWiNFO

В SiSoftware Sandra Lite для получения сводной информации о конфигурации во вкладке «Устройства» выбираем «Информация о системе», после чего открывается соответствующее окно (рис. 2.30).

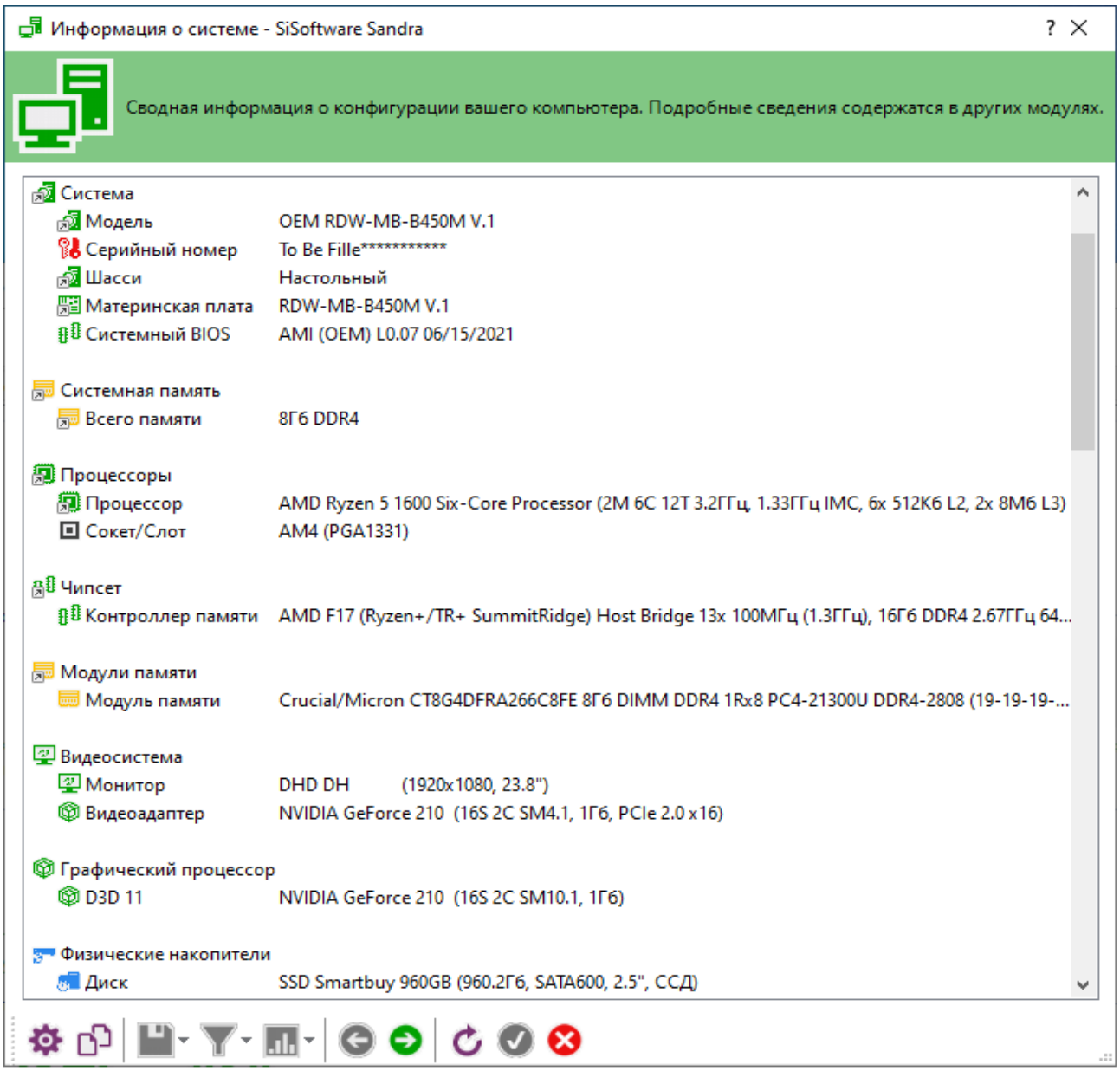


Рис. 2.30. Информация о системе

Для получения информации о чипсете и оперативной памяти выбираем «Устройства/Материнская плата» (рис. 2.31).

Отметим характеристики чипсета:

- модель AMD B450 (см. рис. 2.29);
- системная шина: AMD Infinity Fabric (IF);
- частота (скорость): 1,3 ГГц;
- ширина: 256 бит;
- максимальная пропускная способность 40,62 Гбайт/с.

Надо иметь в виду, что некоторые утилиты, например, AIDA64, используют терминологию «северный мост» — «южный мост», даже

если в системе имеется только одна микросхема чипсета — контроллер-концентратор платформы PCH. В этом случае они PCH обозначают как южный мост, а интегрированный в процессор контроллер памяти IMC — как северный мост. Другие утилиты, например, Sandra, просто относят IMC к чипсету (см. рис. 2.30, 2.31).

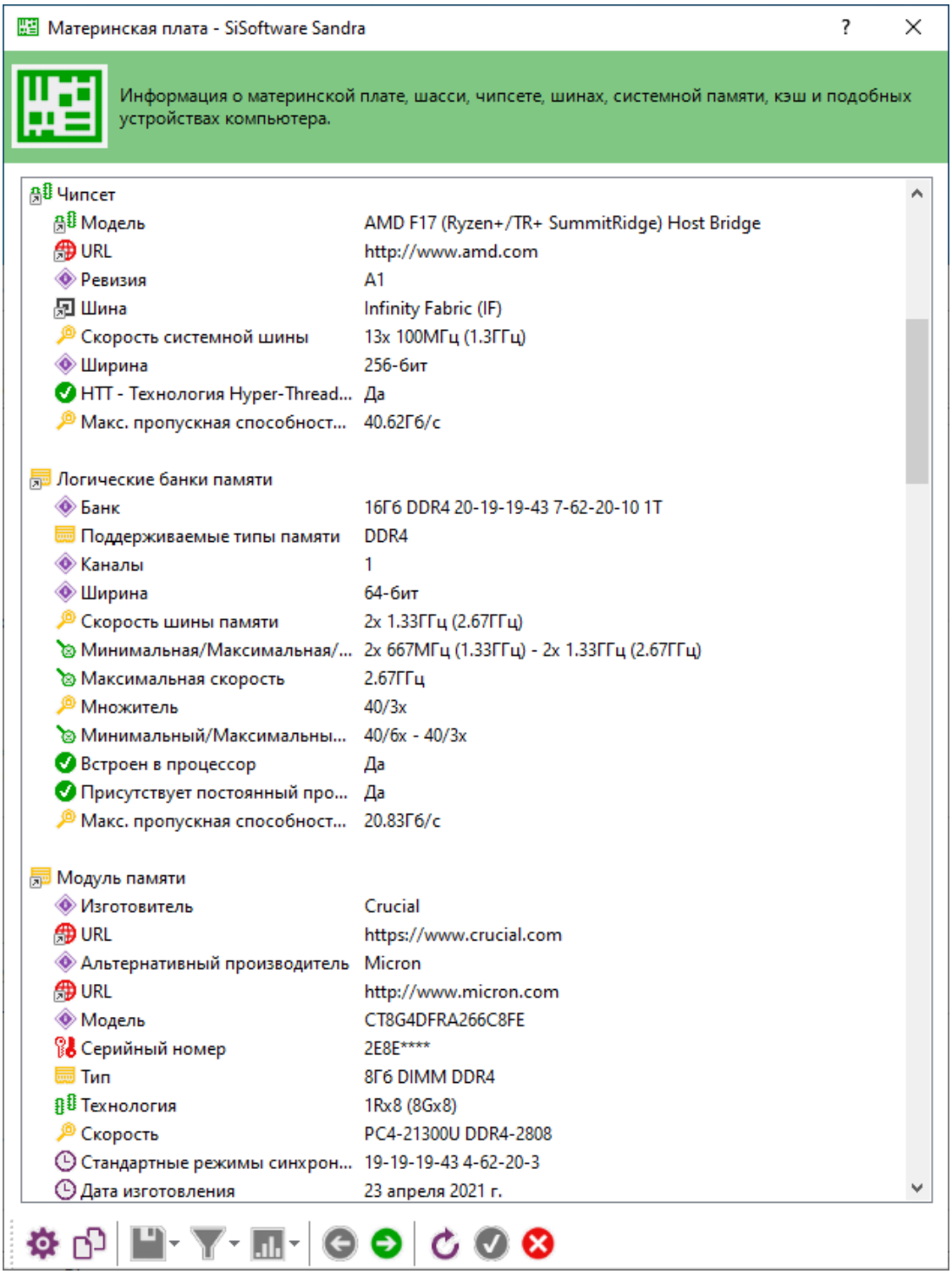


Рис. 2.31. Информация о системной плате: чипсете и ОЗУ

Рассчитаем скорость передачи по шине IF:

1,3 ГГц · 256 бит = 332,8 Гбит/с (ГТ/с).

Это вполне соответствует выданному программой значению максимальной пропускной способности:

40,62 Гбайт/с · 8 бит = 324,96 Гбит/с.

Характеристики подсистемы памяти:

- ширина канала: 64 бита (8 байт);
- частота (скорость): 1333 МГц;
- максимальная пропускная способность 20,83 Гбайт/с;
- один модуль памяти DDR4-2666 (Double Data Rate SDRAM, четвертое поколение синхронной динамической памяти с удвоенной скоростью передачи, 2666 МТ/с (1333 · 2)) или PC-21300 (Personal Computer 21300 Мбайт/с), объем 8 Гбайт, тайминги 20-19-19-43.

Рассчитаем пропускную способность канала памяти:

2666 МГц · 8 байт = 21328 Мбайт/с.

Полученное значение соответствует указанному 20,83 Гбайт/с.

По результатам выполнения задания составить таблицы или списки аппаратных средств с их характеристиками и схему структуры персонального компьютера по примеру рисунков 2.13, 2.15-2.17. Порядок выполнения задания проиллюстрировать скриншотами использованных программ.

Задание 2. Оценка производительности ПЭВМ

Как было сказано в п. 2.1, основным тестом рейтинга Top500 является HPL (High Performance Linpack), основанный на решении систем линейных уравнений. Имеются реализации Linpack и для персональных компьютеров, например, программа LinpackXtreme (рис. 2.32).

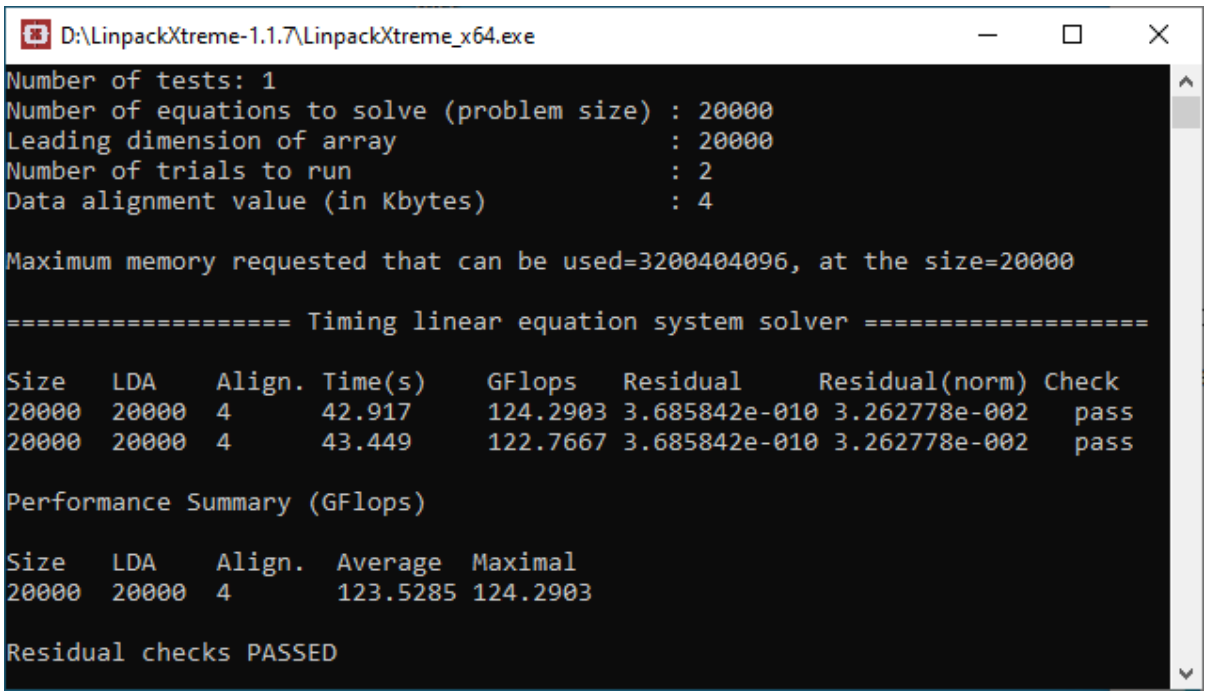


Рис. 2.32. Информация о системной плате: чипсете и ОЗУ

В данном примере решалась система из 20000 уравнений. В итоге получено значение 123,5285 ГФлопс. На одно ядро приходится:

123,5285 / 6 ≈ 20,6 ГФлопс.

За один такт выполняется:

20,6 ГФлопс / 3,2 ГГц ≈ 6,4 операций с плавающей точкой.

Имеется множество тестов производительности в AIDA64 и SiSoftware Sandra (рис. 2.32).

Для оценки общей производительности компьютера можно выбрать «Индекс производительности». Sandra визуализирует результаты в виде лепестковой диаграммы, представляющей собой многоугольник, каждый угол которого соответствует определенной оценке производительности тестируемого компьютера в сравнении с эталонной. В качестве эталонной можно выбрать одну или несколько систем из предлагаемого списка. Каждая система на диаграмме отображается своим цветом. Число углов диаграммы варьируется в зависимости от теста. В «Индексе производительности» она имеет четыре угла. Левый угол — общая производительность системы (рис. 2.33), верхний — производительность процессора, правый — подсистемы памяти (рис. 2.34), нижний — графической подсистемы.

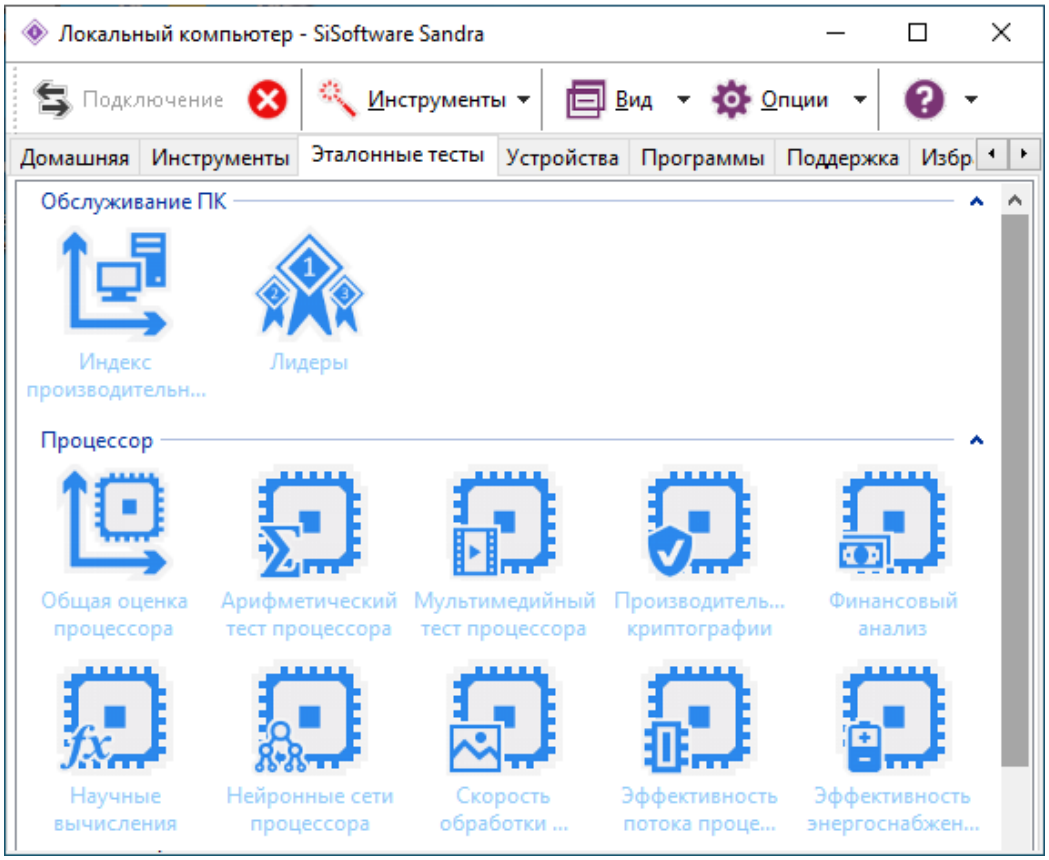


Рис. 2.32. Тесты производительности компьютера в SiSoftware Sandra Lite

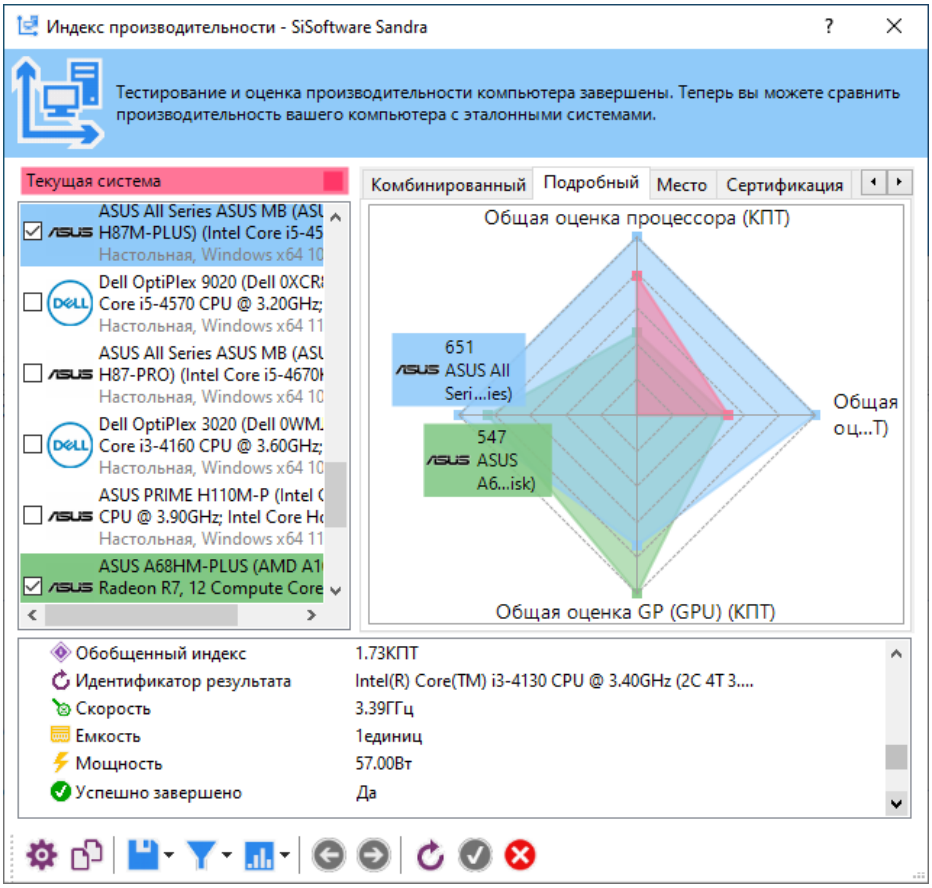


Рис. 2.33. Индекс производительности в SiSoftware Sandra Lite

КПТ (kPT) в программе Sandra — единица измерения производительности, придуманная компанией SiSoftware. Первая буква k — кило, PT — Performance Test (тест производительности). Значение индекса производительности в этих единицах формируется на основе нескольких тестов. Оно мало о чем говорит, но лепестковая диаграмма позволяет сделать относительную оценку производительности тестируемой системы в сравнении с другими (эталонными) системами.

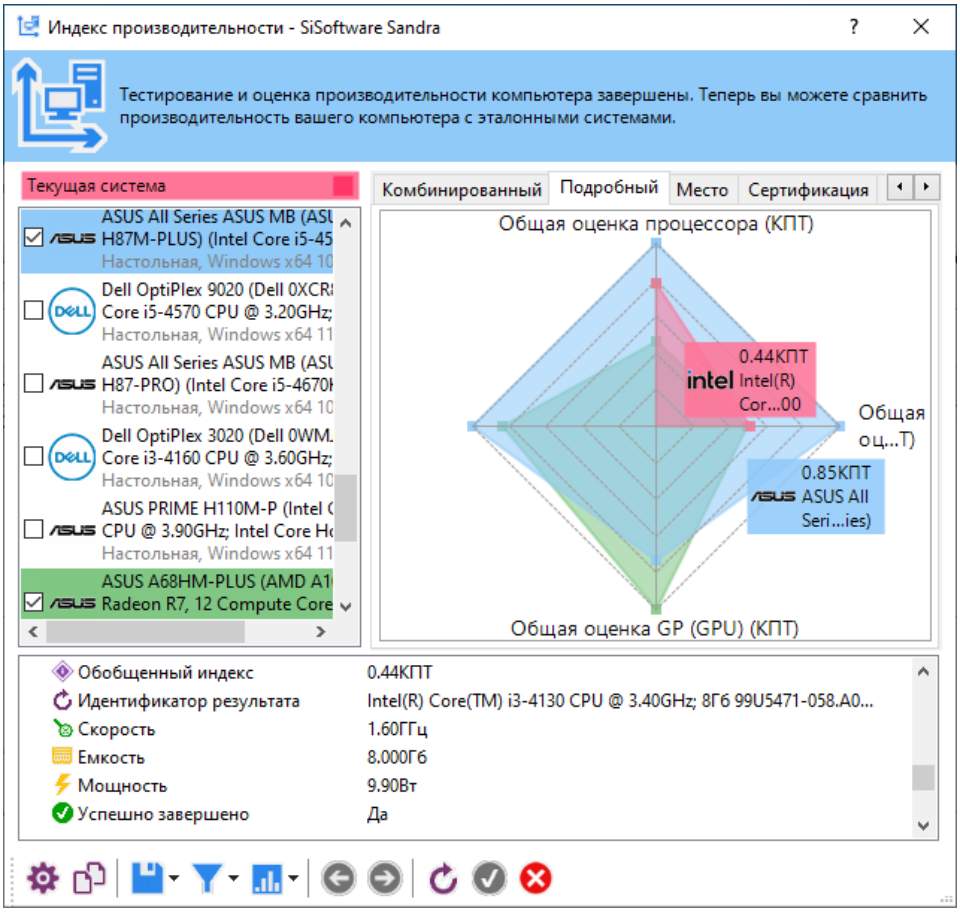


Рис. 2.34. Индекс производительности в SiSoftware Sandra Lite.
Производительность подсистемы памяти

AIDA64 в тесте производительности подсистемы основной и кэш-памяти выдает дифференцированные оценки пропускной способности для чтения, записи и копирования (рис. 2.35).

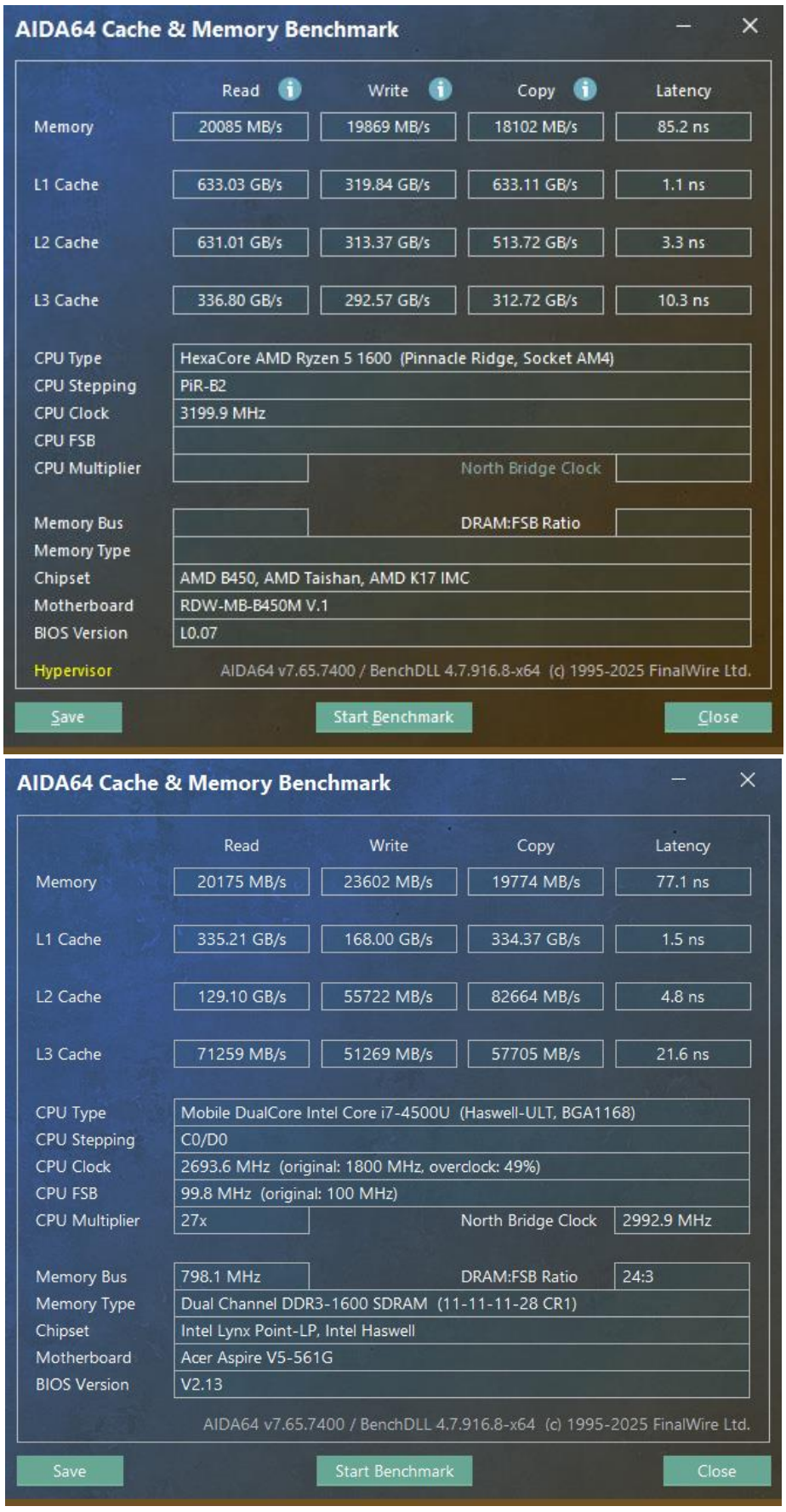


Рис. 2.35. Результаты теста производительности подсистемы памяти в AIDA64

В этом задании необходимо с использованием информационно-диагностических утилит провести тестирование общей производительности компьютера. По результатам выполнения задания составить таблицы с полученными оценками технико-эксплуатационных показателей. Порядок выполнения задания проиллюстрировать скриншотами.

3. МИКРОПРОЦЕССОР. БАЗОВАЯ СТРУКТУРА И СОВРЕМЕННЫЕ ТЕХНОЛОГИИ

3.1. ХАРАКТЕРИСТИКИ И КЛАССИФИКАЦИЯ ПРОЦЕССОРОВ

Определение и характеристики микропроцессоров

Процессор (центральный процессор) — основной элемент компьютера. В однопроцессорных компьютерных системах он один, в многопроцессорных — множество. До появления микропроцессора процессор конструктивно реализовывался на печатной плате и включал множество интегральных схем и дискретных электронных элементов.

Первый микропроцессор Intel 4004 был разработан в 1971 г. в компании Intel. Основными разработчиками были Маршиан Эдвард Хофф (Marcian Edward Hoff), Федерико Фаджин (Federico Faggin) и Стэнли Мейзор (Stanley Mazor).

Микропроцессор — это универсальное программно-управляемое устройство цифровой электроники, осуществляющее прием, обработку и выдачу цифровых двоичных данных в соответствии с программой, хранящейся в запоминающем устройстве, и реализованное в виде одной или нескольких интегральных схем. Сейчас все процессоры — это микропроцессоры в виде одной интегральной схемы, и более того, помимо собственно процессора на одном кристалле с ним находится множество других устройств: математический сопроцессор (FPU), векторный сопроцессор (устройство MMX), графический процессор (GPU), кэш-память различных уровней, контроллер памяти, контроллер прерываний и т. д. Поэтому понятия процессор и микропроцессор в настоящее время являются синонимами. Для наиболее сложных процессоров, использующихся, например, в смартфонах,

принят также и такой термин, как «система на кристалле», System on Chip (SoC).

Для лучшего понимания определения микропроцессора сравним его с другими типами цифровых микросхем в терминах одного из основных понятий теории управления — черного ящика. Элемент цифровой электроники представляет для пользователя как раз такой черный ящик, который принимает цифровые двоичные сигналы на входе, осуществляет их преобразование и выдает результат на выходе.

Первую группу цифровых микросхем составляют логические элементы (НЕ, И, И-НЕ (НЕ-И), ИЛИ, ИЛИ-НЕ (НЕ-ИЛИ), Исключительное ИЛИ), шифраторы, дешифраторы, мультиплексоры и демультиплексоры. Входные сигналы, представляющие собой некоторые данные, преобразуются в соответствии с определенной логической функцией в выходные данные. В каждый момент времени выход зависит только от комбинации текущих логических значений входов. Какая-либо зависимость от предшествующих значений входов отсутствует. Поэтому элементы этой группы называются элементами комбинационной логики.

В составе группы элементов комбинационной логики выделяются шифраторы, дешифраторы, мультиплексоры и демультиплексоры. Отличие их от логических элементов состоит в нарушении однородности входных данных. У мультиплексоров и демультиплексоров есть собственно данные и есть управляющая информация, представляющая собой номер подключаемого на входе или выходе канала. У шифраторов и дешифраторов также имеются управляющие сигналы. Вообще, эти устройства можно объединить в одном изделии. Одна из популярных микросхем К155ИД4 (74155) представляет собой сдвоенный дешифратор-демультиплексор.

Однако у всех элементов комбинационной логики отсутствует то, что называется внутренним состоянием или памятью.

Элементы, обладающие внутренним состоянием (памятью), относятся к классу последовательностной логики. В нее входят триггеры, счетчики, регистры и др. Триггер можно реализовать из логиче-

ских элементов, например, из двух элементов ИЛИ-НЕ или И-НЕ, если выходы каждого из них подключить к одному из входов другого элемента (см. п. 1.2). Поскольку у таких устройств появляется внутреннее состояние, то появляется и новый тип входов — сброс или предустановка (установка в определенное начальное состояние). Другой тип входа, который появляется у этих устройств, — тактовый сигнал. Он необходим для синхронизации изменения состояния устройства с некоторыми дискретными моментами времени, которые задает генератор тактовых сигналов.

В группе последовательностной логики содержится множество все более усложняющихся типов элементов. **Триггер** — самый простой из них, «атомарный» элемент с памятью. Он может хранить один бит информации. Синхронный триггер можно рассматривать как счетчик до двух. **Регистр** — это совокупность триггеров. Регистры могут иметь емкость 8, 16, 32, 64 бита и т. д. Микросхемы памяти — еще более емкие запоминающие устройства. Еще одна подгруппа — программируемые интегральные схемы: различные контроллеры прерываний, прямого доступа к памяти, клавиатуры и т. д. и интерфейсы или порты, параллельные и последовательные различного вида. Запоминающих элементов у них немного, но сложность их состоит во множестве управляющих сигналов, параметров настройки, режимов работы, которые задаются и устанавливаются программно.

Наконец, самым сложным элементом в ряду цифровых интегральных схем является микропроцессор. Он также осуществляет преобразование входных данных в выходные. Но это преобразование может быть любым и сколь угодно сложным. В целом оно определяется выполняемой программой, а в каждом цикле — текущей командой программы. Микропроцессор, собственно, и был создан как альтернатива жесткой логике. Микропроцессор осуществляет чтение команд из памяти, их дешифрацию и выполнение. При этом входная информация для выполнения команд не находится на входных выводах, как у логических элементов, а располагается во внутренних реги-

страх процессора, будучи туда предварительно введенной из памяти или внешних устройств, причем в общем случае не за один цикл. И конечно, управляющие сигналы процессора имеют самый широкий спектр.

Среди других типов цифровых интегральных схем для лучшего понимания сущности микропроцессора рассмотрим еще один — программируемые логические интегральные схемы (ПЛИС) или программируемые логические матрицы (ПЛМ). Они определенным образом сочетают в себе свойства всех вышеперечисленных устройств. ПЛИС имеют в своем составе память (перепрограммируемую постоянную или статическую), логические элементы и ключи, их соединяющие. Создание системы на основе ПЛИС заключается в построении битовой последовательности, записываемой в память ПЛИС. Эта битовая последовательность задает конфигурацию логических элементов замыканием-размыканием определенных ключей. Создание битовой последовательности для ПЛИС имеет аналогию с созданием программ для микропроцессоров. Необходимая конфигурация ПЛИС создается в системе автоматизированного проектирования с использованием средств, аналогичным языкам программирования высокого уровня. Затем с помощью компилятора она переводится в битовую последовательность. Таким образом, ПЛИС сочетают в себе преимущества жесткой логики и универсального процессора. Процессор, как уже было отмечено, — это алгоритмическое устройство. А в природе алгоритм вторичен, т. е. алгоритм поведения системы определяется ее структурой, является сущностью, производной от структуры системы. Сложные естественные системы являются синергетическими (самоорганизующимися). На этом же принципе должен быть основан новый тип вычислительных устройств. Это может быть что-то вроде самоорганизующихся ПЛИС, в которых битовая последовательность создается автоматически в процессе обучения. Пока же средства создания битовых последовательностей являются программами для микропроцессора. А микропроцессор в этом смысле замкнут или са-

модостаточен: программы для него создаются с использованием средств, также являющихся его программами.

Основными характеристиками процессора являются:

1. Технологический процесс.
2. Количество транзисторов на кристалле.
3. Тактовая частота (внутренняя и внешняя).
4. Количество ядер.
5. Потребляемая (рассеиваемая) мощность.
6. Тип корпуса (разъема), количество выводов.
7. Архитектура.

Технологический процесс — это числовой параметр, определяемый размером электронного элемента, создаваемого на кристалле. Чем меньше размер элемента, тем, соответственно, большее их число можно разместить на кристалле одной и той же площади. Основным электронным элементом интегральной схемы является транзистор, чаще всего униполярный (полевой). Поэтому за число, обозначающее технологический процесс, чаще всего принимается размер затвора транзистора на кристалле. Впрочем, здесь нет единого стандарта, каждый производитель самостоятельно трактует понятие технологического процесса, руководствуясь маркетинговыми соображениями. Особенно это характерно для текущей ситуации, когда технологический процесс составляет около 7 нм, что сопоставимо с размером атомов, и фактически достигнут предел значения этой величины. Но это касается только самых высокопроизводительных процессоров для настольных и мобильных систем. Выпускается множество процессоров для встроенных и специализированных компьютерных систем с большим значением технологического процесса — 65, 45, 32, 28, 22, 14 нм.

С технологическим процессом связано число транзисторов на кристалле. Именно этот параметр указывался для первых микропроцессоров. Для современных процессоров значение этой характеристики составляет несколько миллиардов. Например, процессор AMD

Ryzen Threadripper 3990X содержит 64 ядра и 7,3 миллиарда транзисторов на кристалле.

Тактовая частота — номинальное значение частоты тактовых импульсов, на которой работают внутренние узлы процессора, т. е. максимальное значение внутренней частоты, при котором процессор работает надежно. Это одна из двух величин в формуле (1.2), впрямую влияющая на производительность процессора:

$$E = I / T = (I / Clock) \cdot (Clock / T) = IPC \cdot F.$$

Внешняя тактовая частота, частота системной шины, формируемая генератором, меньше внутренней. Они связаны множителем, называемым коэффициентом умножения. Внутренняя частота также может быть разной — номинальной для большей части процессора и удвоенной для исполнительных блоков. Тактовая частота может изменяться в зависимости от нагрузки.

Количество ядер процессора влияет на значение второй величины в формуле (1.2), *IPC*. Оно равно произведению числа ядер процессора на максимальное количество команд, выполняемых за один такт одним ядром.

Тактовая частота процессора оказывает прямое влияние на следующую его характеристику — потребляемую (рассеиваемую) мощность:

$$P = C_d \cdot U^2 \cdot F,$$

где C_d — динамическая емкость;

U — напряжение питания;

F — тактовая частота.

Конечно, на потребляемую мощность также влияет количество ядер и, вообще, количество одновременно работающих в процессоре устройств. При работе процессора его загруженность изменяется, она зависит от того, какие программы выполняются в текущий момент

времени. Соответственно, изменяется и текущая потребляемая мощность. В спецификациях процессора для оценки потребляемой мощности указывается такая величина, как TDP (Thermal Design Power, расчетная тепловая мощность). TDP — это мощность процессора в ваттах при определенной рабочей нагрузке. Разные производители при расчете TDP определяют эту нагрузку по-разному. Одни берут среднюю нагрузку, другие — максимальную.

В процессе развития микропроцессорной техники растут тактовая частота, количество транзисторов и сложность процессоров (число ядер и различных устройств в процессоре).

В 1965 г. один из основателей Intel Гордон Мур (Gordon Moore) выявил эмпирическую закономерность, впоследствии названную законом Мура, согласно которой количество транзисторов, размещаемых на кристалле интегральной схемы, за год увеличивается примерно вдвое. По его прогнозу за десять лет количество транзисторов должно было увеличиться с 2^6 до 2^{16}. До 1971 г. он выполнялся неплохо, затем хуже:

1971 г. — Intel 4004 — 2 300 транзисторов;

1972 г. — Intel 8008 — 3 500 транзисторов;

1973 г. — Intel 8080 — 4 800 транзисторов;

1975 г. — Zilog Z80 — 8 400 транзисторов;

1978 г. — Intel 8086 — 29 000 транзисторов.

В 1975 г. он сделал корректировку своего закона: количество транзисторов на кристалле удваивается каждые 24 месяца. Но сейчас эта величина почти уже не изменяется, количество транзисторов на кристалле процессора достигло нескольких миллиардов в конце нулевых годов и сейчас остается примерно на этом же уровне. Тактовая частота также остается на уровне около 3–4 ГГц уже многие годы. На основе закона Мура был сделан прогноз, согласно которому производительность процессора удваивается каждые 18 месяцев. Если тактовая частота не растет, то производительность процессора и, соответственно, компьютера можно увеличить только за счет *IPC*. Производительность компьютеров продолжает расти за счет увеличения в них

числа процессоров и общего числа ядер. По статистике рейтинга Top500 зависимость производительности от времени суперкомпьютеров рейтинга на протяжении 20 лет была практически экспоненциальной, а сейчас также наблюдается некоторое насыщение (рис. 3.1) [20].

На графике ось ординат — логарифмическая, т. е. равномерная по десятичному логарифму производительности. Цена деления вертикальной шкалы — единица логарифма, десятикратное увеличение производительности. Логарифм производительности растет линейно, производительность — экспоненциально.

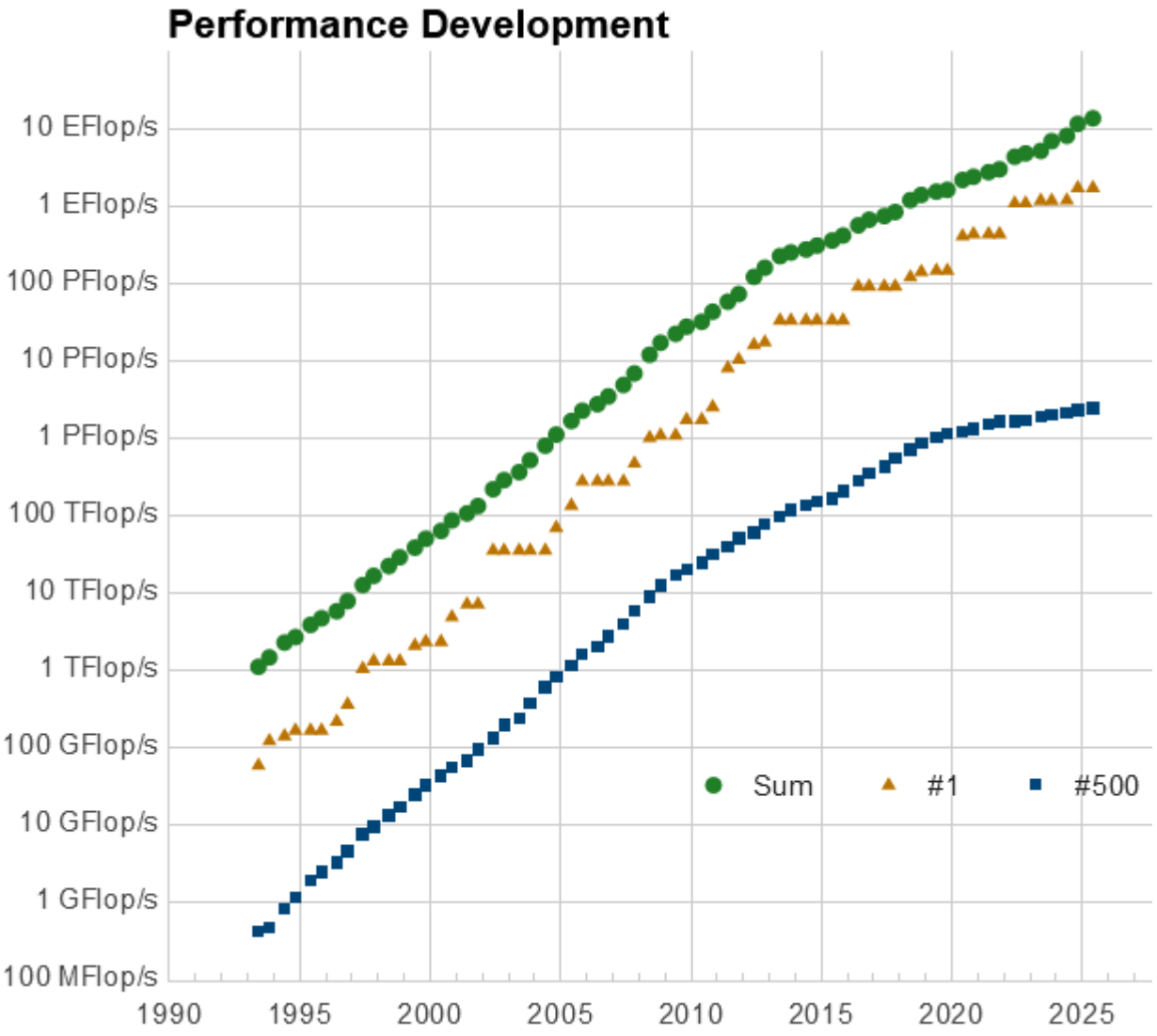


Рис. 3.1. Рост производительности компьютеров рейтинга Top500

Чтобы при росте производительности потребляемая мощность оставалась в разумных пределах, совершенствуются полупроводниковые технологии. Снижается также напряжение питания, которое в соответствии с формулой (1.2) тоже оказывает прямое влияние на энергопотребление. Рассеиваемая мощность первых процессоров была невелика, и они не требовали специального охлаждения. Оно требуется, начиная с 4 Вт рассеиваемой мощности. Снижение температуры процессора на 10 градусов ведет к удвоению времени его безотказной работы. Более того, нагрев кристалла выше 70 градусов может привести к необратимому повреждению процессора. Наиболее распространенные средства охлаждения — радиаторы и вентиляторы. В среднем радиатор снижает температуру на 20 градусов, радиатор вместе с вентилятором — на 40 градусов. Во многих современных радиаторах используется жидкостное охлаждение.

Основные типы корпуса процессоров:

– DIP (Dual In Line Package) — корпус с двухрядным расположением выводов — прямоугольный с изогнутыми выводами, выходящими из двух боковых сторон перпендикулярно им;

– QFP (Quad Flatpack Package) — прямоугольный с четырехрядным расположением прямых выводов, выходящих из боковых сторон параллельно плоскости основания;

– PGA (Pin Grid Array) — прямоугольный с прямыми (штырьковыми) выводами, перпендикулярными плоскости основания, расположенными в виде матрицы по нижней поверхности корпуса;

– BGA (Ball Grid Array) — прямоугольный с матрицей полушарообразных или столбиковых выводов на плоскости основания.

Существует множество разновидностей корпусов PGA и BGA.

Корпусу PGA соответствует разъем Socket с соответствующим числом контактов. Первые процессоры выпускались в корпусе DIP с 40 выводами (DIP-40). Некоторые модели микроконтроллеров и сейчас выпускаются в таких корпусах. Современные высокопроизводительные процессоры для персональных компьютеров, рабочих станций и серверов выпускаются в корпусах PGA различного типа, но

уже не со штырьковыми, а выводами в виде плоских контактных площадок, с количеством выводов более чем 2 000.

Классификация процессоров по архитектуре

Последняя из приведенных характеристик процессора — архитектура — является самой сложной. Она включает в себя большое количество отдельных архитектурных признаков:

− множество и разрядность шин (адреса, данных, управления, внутренних и внешних), множество управляющих сигналов;

− количество ядер и других параллельно работающих блоков;

− множество внутренних устройств (устройство с плавающей точкой, кэш-память и т. д.);

− множество и разрядность регистров, группы регистров;

− типы обрабатываемых данных;

− система команд (группы команд, порядок выполнения команд, способы адресации).

По различным архитектурным признакам осуществляются следующие четыре основных деления процессоров на классы:

− RISC (Reduced (Restricted) Instruction Set Computer, компьютер (процессор) с сокращенным набором команд) и CISC (Complex (Complete) Instruction Set Computer, компьютер (процессор) с полным (сложным) набором команд);

− гарвардской и принстонской архитектуры;

− со скрытым (implicit) параллелизмом (суперскалярные процессоры) и явным (explicit) параллелизмом (EPIC-процессоры);

− процессоры с фиксированной и модифицируемой системой команд.

По особенностям системы команд, а также совокупности других, связанных с этим архитектурных признаков выделяются RISC и CISC процесоры. ***RISC-процессоры имеют следующие основные признаки:***

− небольшая несложная система команд;

− команды простые, короткие, одной длины (два или четыре байта);

– большинство команд выполняется за один такт;

– большое число регистров общего назначения с минимальной специализацией.

В RISC-процессорах можно использовать любые регистры общего назначения в любых командах с любыми способами адресации. Это свойство называется ортогональностью, ортогональностью системы команд или ортогональностью регистров. (Конечно, на сто процентов это никогда не достигается.)

Из-за простоты системы команд программа для RISC-процессоров включает, естественно, большее число команд, чем аналогичная для CISC-процессора. Примеры RISC-процессоров — Power PC компании IBM и SPARC компании Sun Microsystems.

У ***CISC-процессоров характеристики противоположные:***

– система команд большая и сложная;

– команды различаются по сложности и длине, имеются сложные и длинные команды;

– команды выполняются более чем за один такт;

– небольшое число регистров общего назначения с выраженной специализацией.

В CISC-процессорах полностью отсутствует ортогональность. Каждый регистр общего назначения имеет определенную специализацию.

Пример CISC-процессоров — семейство x86.

Если говорить строго, то это классы теоретические, две крайние точки зрения на основные принципы устройства процессора. Любая реальная модель процессора находится где-то в промежутке между этими полюсами, имеет черты обоих классов, но находится ближе к тому или иному полюсу. Поэтому каждый конкретный процессор относят к одному из этих классов. Тем не менее множество современных процессоров на разных уровнях сочетает в себе свойства RISC- и CISC-архитектур, поскольку каждая из них имеет определенные преимущества.

Простые команды RISC-процессоров позволяют эффективно использовать для своего исполнения конвейер. Поэтому в некоторых CISC-процессорах, например, семейства x86, имеется микропрограммная память — внутреннее ПЗУ, содержащее набор микропрограмм, состоящих из микрокоманд. Сложные CISC-команды отображаются на эти микропрограммы, т. е. в процессе выполнения они распадаются на еще более мелкие функциональные единицы — микрокоманды. Этот дополнительный уровень детализации команд в CISC-процессорах позволяет использовать конвейер и в этом типе процессоров.

В RISC-процессорах оптимизация исполняемого кода в основном выполняется компилятором. В CISC-процессорах с элементами RISC в полной мере это сделать невозможно по причине того, что RISC-уровень находится внутри и никак не проявляется программно. И хотя компилятор также выполняет определенную оптимизацию, основная ее часть осуществляется во время исполнения программы самим процессором путем распределения микрокоманд по параллельным блокам с возможным их переупорядочиванием. Несмотря на то, что эти процессоры с точки зрения программиста и с точки зрения проектировщика различаются очень сильно, это можно считать их преимуществом, поскольку программа, скомпилированная для конкретной модели процессора, будет эффективно выполняться и на других моделях.

Гарвардская архитектура характеризуется раздельной памятью команд и данных. Она была впервые реализована в Гарвардском университете в компьютере Mark I.

Принстонская архитектура характеризуется объединенной (смешанной) памятью команд и данных. Она ассоциируется с архитектурой фон Неймана, поскольку он работал в принстонском Институте перспективных исследований.

Несмотря на то, что эти два типа архитектур выделяются по принципу организации запоминающего устройства, выделяются и со-

ответствующие классы процессоров, поскольку основные характеристики компьютера определяются характеристиками его главного элемента — процессора.

Многие современные процессоры сочетают также и эти типы архитектур. Например, процессоры x86 используют смешанные основную память и кэш второго уровня, но раздельный кэш первого уровня.

Суперскалярные и EPIC-процессоры выделяются по признаку отражения в системе команд и, соответственно, в программе внутреннего параллелизма процессора:

− суперскалярный процессор — со скрытым параллелизмом;

− EPIC-процессор — с явным параллелизмом.

И тот, и другой класс процессоров содержит не менее двух конвейеров, которые работают параллельно, и реализует параллелизм на уровне команд (ILP, Instruction-Level Parallelism).

В суперскалярном процессоре распараллеливание (распределение команд по конвейерам) происходит на аппаратном уровне по внутренним алгоритмам процессора. Команды суперскалярных процессоров не содержат в явном виде информации, касающейся параллельной обработки внутри процессора. Соответственно, компилятор не может оптимизировать программу для параллельной обработки. Пример суперскалярного процессора — Pentium.

Процессоры с явным (explicit) параллелизмом реализуют технологию вычислений под названием EPIC (Explicitly Parallel Instruction Computing). Распараллеливание происходит уже на уровне компилятора, поскольку команды в явном виде содержат информацию, касающуюся параллельной обработки внутри процессора. Примером EPIC-процессора является Intel Itanium.

Подклассом EPIC-процессоров являются процессоры с длинным командным словом (VLIW, Very Long Instruction Word). Команды VLIW-процессора имеют специальные поля, содержащие информацию для каждого из параллельных обрабатывающих устройств процессора, предписывающую им определенные действия.

Также иногда выделяется такой тип архитектуры, как MISC (Minimal Instruction Set Computer, компьютер с минимальным набором команд). В нем используется сокращенный набор команд, при этом несколько простых команд укладывается в одно большое слово. Порядок выполнения команд определяется таким образом, чтобы в максимальной степени загрузить все узлы процессора. Таким образом, MISC сочетает в себе RISC-, VLIW- и суперскалярную архитектуру.

По возможности изменения системы команд выделяют два типа процессоров:

− процессор с фиксированной системой команд (с произвольной логикой);

− процессор с модифицируемой системой команд (с возможностью микропрограммирования).

Микропрограммируемые процессоры оснащаются управляющим запоминающим устройством, в котором хранится микропрограмма — встроенные программы, определяющие набор выполняемых команд. Каждой команде соответствует своя микропрограмма. Микропрограммное управление дает возможность корректировать систему команд и детали выполнения отдельных команд. Однако, это было актуально только в начале развития микропроцессорной техники. Сейчас системы команд стабилизировались, и большинство процессоров имеет фиксированную систему команд. Тем не менее, многие современные процессоры имеют микропрограммную память. Такие процессоры, конечно, имеют конвейер, что в сочетании с микропрограммным управлением дает возможность увеличения производительности. Примером процессора с микропрограммной памятью является Pentium и все последующие процессоры x86. Начиная с процессора Pentium 4 (микроархитектуры NetBurst), кэш-память команд первого уровня хранит уже не команды, а микрокоманды.

Наиболее известными в настоящее время архитектурами и семействами процессоров являются:

1. Intel x86, AMD x86 — CISC с элементами RISC.

2. IBM Power — RISC.

4. ARM (Advanced RISC-Machine) — RISC.

Еще несколько лет назад достаточно распространенными были RISC-процессоры SPARC компании Sun Microsystems, в последнее время производимые компанией Fujitsu, но сейчас их ипользование практически прекратилось. Также пришла в упадок еще одна популярная в свое время архитектура IBM Power. Зато появились и интенсивно развиваются другие архитектуры: NVIDIA Grace (см. следующий пункт) и RISC-V. В аппаратном обеспечении вычислительных систем RISC-V является примерно тем же, чем является Linux в программном обеспечении, т. е. это свободно лицензируемая архитектура. Проекты и стандарты архитектурных описаний RISC-V доступны без ограничений для любого использования, включая коммерческие реализации в виде готовых микросхем. Ее концепция была разработана в Калифорнийском университете в Беркли и представлена в 2010 г. Сейчас в развитии RISC-V и выпуске микросхем с этой архитектурой участвует множество стран и ведущих компаний, в том числе Intel. В нашей стране микроконтроллеры на базе RISC-V выпускают компании «Микрон», «Миландр» и «Байкал Электроникс».

Объективную картину популярности, преимуществ современных архитектур и семейств процессоров, причем в динамике, дает рейтинг Top500 (табл. 3.1).

Таблица 3.1

Доля различных классов процессоров в суперкомпьютерных системах рейтинга Top500

№ рейтинга	Дата	Intel x86	AMD x86	IBM Power	SPARC	ARM	NVIDIA Grace	∑
45	2015.06	433	23	36	7	0	0	499
46	2015.11	445	21	26	7	0	0	499
47	2016.06	455	13	23	7	0	0	498
48	2016.11	462	7	22	7	0	0	498
49	2017.06	464	6	21	7	0	0	498
50	2017.11	471	5	15	7	0	0	498

№ рейтинга	Дата	Intel x86	AMD x86	IBM Power	SPARC	ARM	NVIDIA Grace	∑
51	2018.06	475	2	15	6	0	0	498
52	2018.11	476	2	13	6	1	0	498
53	2019.06	478	2	13	4	1	0	498
54	2019.11	474	5	14	3	2	0	498
55	2020.06	471	10	12	1	4	0	498
56	2020.11	459	21	10	1	5	0	496
57	2021.06	432	48	8	0	6	0	494
58	2021.11	408	73	7	0	6	0	494
59	2022.06	388	93	7	0	6	0	494
60	2022.11	379	101	7	0	5	0	492
61	2023.06	360	121	7	0	6	0	494
62	2023.11	338	140	7	0	8	0	493
63	2024.06	315	156	6	0	9	7	493
64	2024.11	309	162	4	0	9	9	493
65	2025.06	294	173	4	0	9	13	493
66	2025.11	285	178	4	0	9	18	494

По данным рейтинга Top500 более 90 % самых высокопроизводительных систем мира в настоящее время используют процессоры архитектуры x86 производства Intel и AMD. Поэтому в данном учебном пособии основное внимание уделяется именно этой архитектуре, разработанной Intel. Число систем с IBM Power меделенно снижается. Число систем с процессорами ARM, появившихся в рейтинге в 2018 году, медленно растет. Сначала это была система Astra — Apollo 70 с процессорами Marvell ThunderX2, затем появились компьютеры на процессорах Fujitsu A64FX, самым известным из которых является Fugaku, занявший первое место в 55-м рейтинге (июнь 2020 г.). Сейчас их число дополнили системы с процессорами NVIDIA Grace, которые также имеют ядра архитектуры ARM. Надо отметить, что архитектура ARM в настоящее время доминирует в смартфонах. Также надо учесть, что во многих суперкомпьютерных системах рейтинга Top500 в качестве ускорителей используются специализированные процессоры. В большинстве случаев — это графические процессоры производства NVIDIA. Они будут рассмотрены в следующем параграфе.

Специализированные процессоры

Важный архитектурный признак процессора — тип обрабатываемых данных. Это основная единица информации, которой манипулируют команды процессора.

По размеру это могут быть бит, полубайт (тетрада), байт, слово, двойное слово, учетверенное слово, двойное учетверенное слово *(double quadword)* и т. д. Здесь под словом подразумевается информационная единица длиной шестнадцать бит. Однако существует и другие его определения, в соответствии с которыми длина слова определяется разрядностью внешней шины данных или размером той самой базовой информационной единицы, которую и называют типом обрабатываемых данных.

По формату (способу кодирования) данные могут представлять собой целые числа в дополнительном коде, вещественные числа с плавающей точкой, массивы чисел (векторы) и т. д.

По типу обрабатываемых данных, особенностям системы команд, а также совокупности других архитектурных признаков выделяются универсальные и специализированные процессоры. Универсальные процессоры используются в качестве центральных процессоров (CPU, Central Processing Unit). Специализированные процессоры чаще всего используются вместе с универсальным центральным процессором как сопроцессоры-ускорители или центральные элементы соответствующей подсистемы ЭВМ, но могут и сами выступать в качестве центрального процессора в специализированных вычислительных системах. Например, сетевой процессор является центральным процессором в маршрутизаторах. Правда, для выполнения этой функции они все равно имеют основное ядро, являющееся универсальным процессором. Все специализированные процессоры, как и универсальные, имеют свою систему команд, выполняют программы, но команды их ориентированы на определенную область применения. Специализированные процессоры могут различаться степенью специализации. Бывает, что специализированные

процессоры, использующиеся в качестве сопроцессоров-ускорителей, дорабатываются для использования их в качестве центральных процессоров. В качестве такого примера можно привести процессор Intel Xeon Phi.

Основные типы специализированных процессоров:

– математический процессор или устройство обработки вещественных чисел с плавающей точкой (Floating Point Unit, FPU);

– физический процессор (Physics Processing Unit, PPU);

– сигнальный процессор (Digital Signal Processor, DSP);

– медийный процессор (media processor);

– графический процессор (Graphics Processing Unit, GPU);

– тензорный процессор (Tensor Processing Unit, TPU);

– сетевой процессор (Network Processor, NP или ENP, Ethernet NP);

– криптографический процессор (Trusted Platform Module, TPM, или Platform Security Processor, PSP);

– процессор обработки данных (Data Processing Unit, DPU).

Математический процессор (сопроцессор) выполняет операции над числами с плавающей точкой. Раньше он выпускался в виде отдельной микросхемы, а сейчас входит в состав большинства универсальных процессоров. Для работы вместе с процессором Intel 8086 был выпущен сопроцессор Intel 8087 (1980 г.). Он имеет 80-битные регистры данных и работает с несколькими типами данных:

– целое слово 16 бит;

– короткое целое 32 бита;

– длинное целое 64 бита;

– упакованное десятичное 80 бит;

– короткое вещественное 32 бита;

– длинное вещественное 64 бита;

– расширенное вещественное 80 бит.

Таким образом, уже в те годы в программе можно было использовать 64-битные целые числа, правда, только в командах математического сопроцессора. На одном кристалле с универсальным цен-

тральным процессором математический сопроцессор был размещен в процессоре i486 в виде FPU.

Физический процессор специализирован для выполнения вычислений, связанных с различными физическими процессами: движением твердых тел и жидкостей, столкновением частиц и т. п., — что требуется в основном в играх. Физический процессор в виде специализированного микропроцессора был реализован в 2005 г. в микросхеме Ageia PhysX компании Ageia Technologies. В 2008 г. она была поглощена компанией NVIDIA, и в дальнейшем функциональность физического процессора была включена в графический процессор.

Сигнальный процессор предназначен для выполнения в реальном времени различных функций цифровой обработки сигналов:

- фильтрации сигнала;
- свертки (смешения) двух сигналов;
- вычисления корреляционных функций сигналов;
- усиления, ограничения, трансформации сигналов;
- прямого и обратного преобразования Фурье сигнала.

Для их выполнения они на аппаратном уровне поддерживают базовые операции цифровой обработки сигналов, к которым относятся:

- умножение с накоплением ($C = A \cdot B + C$);
- модульная адресная арифметика;
- нормировка результатов арифметических операций и т. п.

Для выполнения этих операций, в свою очередь, сигнальные процессоры имеют аппаратный умножитель, позволяющий выполнять умножение двух чисел за один такт. Также в них реализована аппаратная поддержка программных циклов и кольцевых буферов.

Сигналы существуют самых разных типов. Соответственно, сигнальные процессоры могут иметь ту или иную специализацию. DSP — универсальный сигнальный процессор. Специализированные сигнальные процессоры: медийные, звуковые и графические процессоры.

Медийные процессоры предназначены для обработки так называемой мультимедийной информации (аудиосигналов, графики),

а также для выполнения коммуникационных функций. Сюда относится кодирование-декодирование аудио и видео в формате MPEG, табличный синтез звука, ускорение трехмерной графики и т. п. Они также могут реализовываться в виде отдельных микросхем или в составе центрального процессора, как например, в Pentium MMX.

Графические процессоры изначально предназначались для формирования изображения на экране монитора. Однако в дальнейшем они стали использоваться в качестве ускорителей для выполнения общих научно-технических расчетов в рамках направления, называемого GPGPU (General-Purpose computing on GPU). Современные графические процессоры имеют большое количество ядер и зачастую превосходят по вычислительной мощности центральные (универсальные) процессоры.

Тензорный процессор — более специализированный графический процессор, предназначенный для интенсивных вычислений с пониженной точностью, например, FP8 (Floating Point 8 бит), включающих в основном матричное умножение и свертку, при более высокой энергоэффективности (производительности на ватт). В отличие от графического процессора TPU не имеет блоков, связанных с формированием изображений, например, растеризации и текстурирования. TPU позиционируются как ускорители искусственного интеллекта, поскольку операции умножения на веса с последующим добавлением смещения, так называемого плавного умножения-сложения (FMA, Fused Multiply-Add), для выполнения которых они и разработаны, широко используются в искусственных нейронных сетях. Графические процессоры NVIDIA, начиная с микроархитектуры NVIDIA Volta, впервые реализованной в линейке Tesla V100, включают тензорные ядра для работы с типами данных FP16, INT8, INT4 и INT1. Каждое ядро может выполнять 1024 INT1, 256 INT4, 128 INT8 и 64 FP16 операций за такт, а один графический процессор может иметь несколько сотен тензорных ядер.

Не так давно появилось понятие **языкового процессора** (Language Processing Unit, LPU). LPU — это процессор для обработки больших языковых моделей (Large Language Model, LLM). LPU значительно ускоряет работу искусственных нейронных сетей с генеративным искусственным интеллектом, использующих LLM. Один из первых LPU разработан и выпущен компанией Groq. Вначале Groq выбрала для такого типа процессора название Tensor Streaming Processor (TSP), а позже заменила его на LPU. Фактически LPU — это еще более специализированный тензорный процессор. LPU компании Groq выпускается по техпроцессу 14 нм. Он включает множество обрабатывающих блоков SIMD, конвейеров, которые Groq в документации называет конвейерными лентами (conveyor belts) и встроенную оперативную память с пропускной способностью 80 Тбайт/с [52].

Сетевой процессор — это процессор, предназначенный для использования в сетевых устройствах. Его основной задачей является обработка пакетов (packet processing). В состав сетевого процессора входит одно центральное ядро и несколько ядер процессоров обработки пакетов (пакетные процессоры — packet processor, PP), а также запоминающие устройства и интерфейсы ввода-вывода.

Центральное ядро выполняется на базе универсального процессора. На нем исполняется все программное обеспечение, в том числе операционная система.

Пакетные процессоры работают параллельно, выполняя функции обработки пакетов. Они реализуются как RISC-процессоры или DSP-процессоры.

Криптографический процессор выполняет функции защиты информации: кодирование-декодирование, управление паролями, проверку прав доступа, ограничение доступа и т. п. Это также могут быть отдельные микросхемы или специальные блоки в универсальных процессорах. Например, в процессорах x86 на аппаратном уровне реализована поддержка симметричного алгоритма шифрования AES (Advanced Encryption Standard) и имеются специальные команды, которые входят в состав расширения AES-NI.

Процессор обработки данных DPU предназначен для параллельной обработки неструктурированных больших данных и, соответственно, используется в вычислительных системах data-центров (центров обработки данных). Они разгружают центральный процессор, распределяя нагрузку с одного CPU на множество DPU. DPU содержит процессор, контроллер сетевого интерфейса (NIC) и программируемые движки данных — встроенные хранилища «ключ-значение» (KVS, Key-Value Store), которые сортируют и индексируют данные. Пример DPU — NVIDIA BlueField.

В центрах обработки данных основная вычислительная нагрузка в настоящее время — генеративный искусственный интеллект. Это искуственные нейронные сети, использующие генеративные модели, в том числе так называемые большие языковые модели (Large Language Model, LLM), и создающие на их основе новые информационные сущности: изображения, текст, программный код и т. д. Генеративный искусственный интеллект требует огромных вычислительных мощностей. При этом, энергопотребление должно находиться в разумных пределах для обеспечения приемлемой энергоэффективности. Для реализации такого рода систем и задач компания NVIDIA разработала процессор (суперчип, как она его называет) Grace Hopper — гетерогенное вычислительное устройство, которое объединяет два процессора: универсальный Grace CPU с архитектурой ARM и графический Hopper GPU с тензорными ядрами (рис. 3.2) [53].

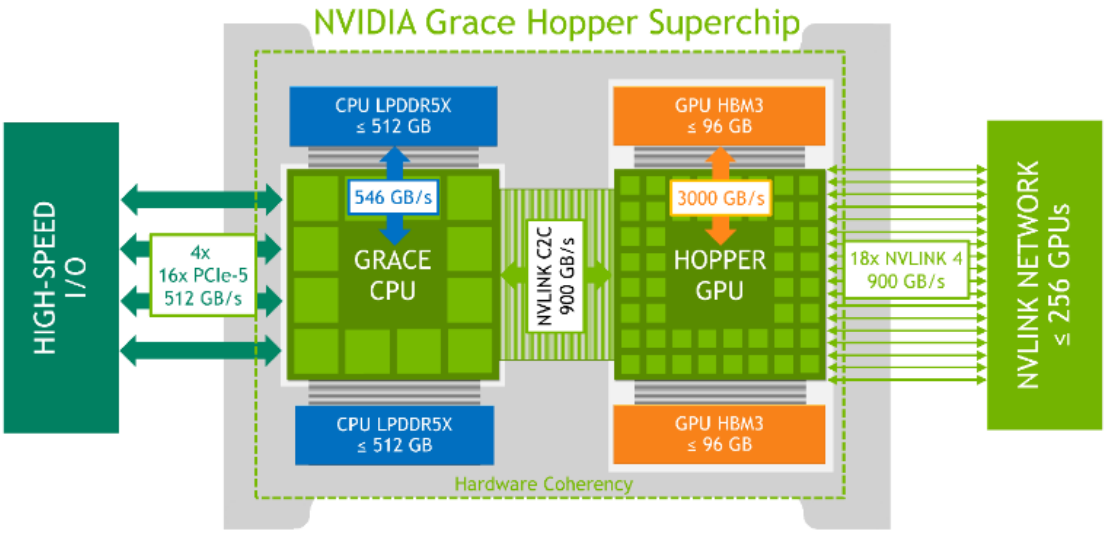


Рис. 3.2. Структура процессора NVIDIA Grace Hopper

Основные характеристики процессора Grace Hopper:

− Grace CPU — 72 ядра ARM Neoverse V2;

− Hopper GPU — H100 Tensor Core;

− канал между CPU и GPU — NVLink-C2C с пропускной способностью 900 GB/s в обоих направлениях и энергоэффективностью 1,3 пикоджоуля на бит;

− память CPU — LPDDR5X (Low Power DDR) до 512 GB с пропускной способностью 546 GB/s на один CPU;

− память GPU — HBM3 (High Bandwidth Memory) до 96 GB с пропускной способностью 3000 GB/s (3 TB/s);

− адресное пространство GPU — 150 TB;

− интерфейс ввода-вывода — 4 канала 16x PCIe Gen 5 с пропускной способностью 512 GB/s;

− потребляемая мощность — 1000 W (1 КВт).

GPU H100 для различных типов данных имеет следующие значения пиковой производительности:

− FP64 Tensor Core — 66,9 Tflop/s;

− FP32 Tensor Core — 494,7 Tflop/s;

− FP16 Tensor Core — 989,4 Tflop/s;

− FP8 Tensor Core — 1978,9 Tflop/s.

На основе процессора Grace Hopper можно создавать мощные масштабируемые вычислительные системы (рис. 3.3 и 3.4).

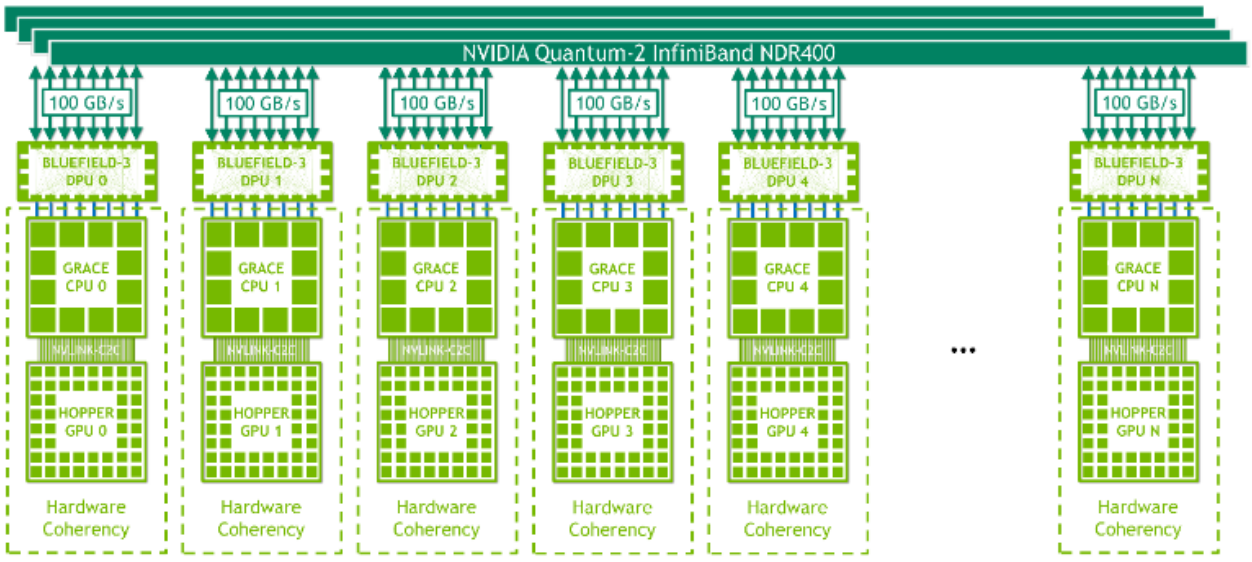


Рис. 3.3. Система NVIDIA HGX Grace Hopper с сетью InfiniBand

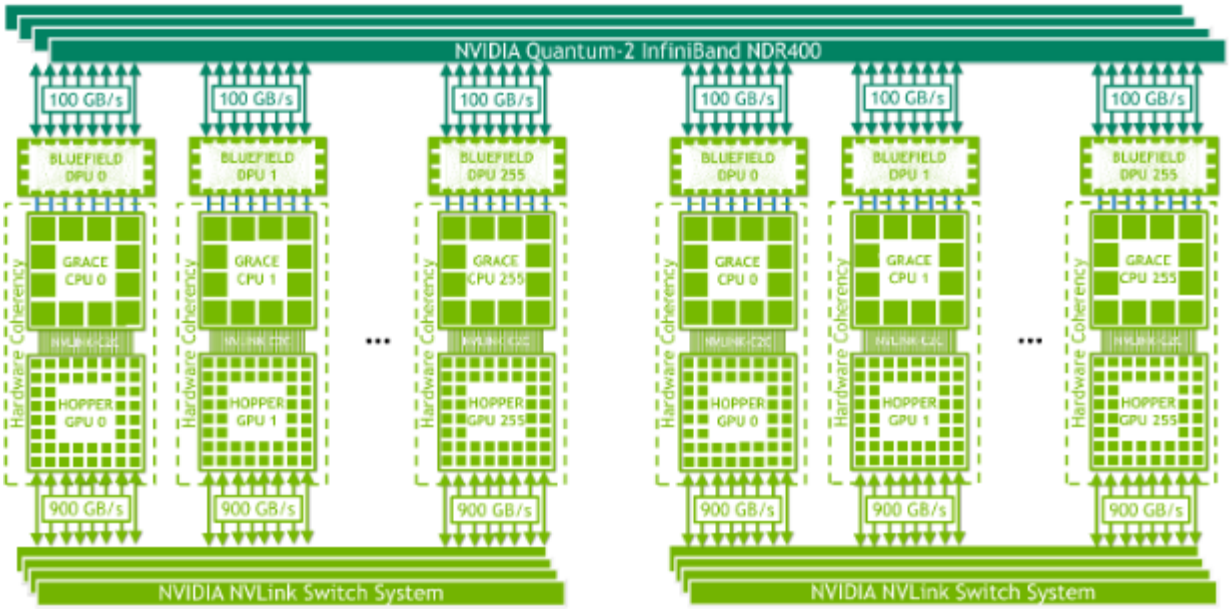


Рис. 3.4. Система NVIDIA HGX Grace Hopper с интерфейсом NVLink

Для этого со стороны CPU используется сетевое соединение InfiniBand NDR400 (см. п. 5.2) и DPU BlueField-3. Процессоры обработки данных BlueField обеспечивают высокоскоростное сетевое подключение, а также выполняют следующие функции:

− управление системой;

− защиту данных;

− изоляцию арендаторов;

− сбор телеметрии;

− виртуализацию хранилища.

Со стороны GPU используется интерфейс NVIDIA NVLink Switch System, который в сочетании с функцией EGM (Extended GPU Memory) позволяет каждому графическому процессору адресовать до 150 TB памяти GPU, предоставляемой всеми суперчипами. Также с помощью EGM любой поток любого GPU через NVLink Switch System или через NVLink C2C может получать доступ ко всей памяти как LPDDR5X CPU, так и HBM3 GPU (рис. 3.5).

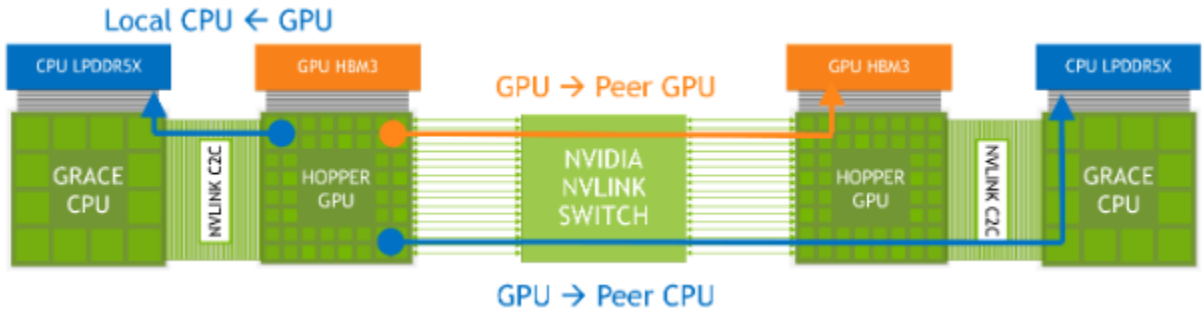


Рис. 3.5. Доступ к памяти в системе
с процессорами Grace Hopper

Другие важные функциональные характеристики Grace Hopper:

− MPAM (Memory Resource Partitioning and Monitoring, разделение и мониторинг ресурсов памяти);

− MIG (Multi-Instance GPU);

− PMU (Performance Monitoring Units, модули мониторинга производительности).

MPAM позволяет распределять доступную полосу пропускания памяти LPDDR5X и использование кэша процессора. MIG позволяет разделить графический процессор на более мелкие экземпляры. Совместно MPAM и MIG могут использоваться для разделения системных ресурсов с целью улучшения качества обслуживания. Модули мониторинга производительности PMU используются для определения и сбора различных показателей производительности CPU, GPU, интерфейсов NVLink C2C, PCIe и памяти.

GPU имеет также устройства асинхронного выполнения (Asynchronous Execution Engines), позволяющие обеспечивать загруженность всех исполнительных блоков.

На основе процессоров Grace Hopper создан действующий суперкомпьютер NVIDIA DGX GH200, выполняющий нейросетевые алгоритмы с языковыми моделями с триллионами параметров. ***Основные характеристики DGX GH200:***

− 256 процессоров Grace Hopper с общим числом ядер CPU — 18 432;

− 144 ТБ разделяемой памяти GPU;

− производительность 1 эксафлопс в вычислениях с 8-разрядными вещественными числами с плавающей точкой (FP8).

В марте 2024 года NVIDIA представила новый графический процессор Blackwell [54]. Он содержит 208 млрд. транзисторов, что в 2,5 раза больше, чем у Hopper. При этом его энергоэффективность в 30 раз выше. В архитектуре Blackwell введен новый тип данных — FP4, 4-разрядное вещественное число с плавающей точкой. Максимальное значение производительности одного процессора Blackwell — 20 PFlop/s. На его основе также разработано множество высокопроизводительных систем, например:

− NVIDIA DGX B200, включающая два CPU Intel Xeon Platinum 8570 и восемь GPU Blackwell;

− NVIDIA GB300 NVL72, включающая 36 CPU Grace и 72 GPU Blackwell Ultra.

3.2. ХАРАКТЕРИСТИКИ, НАЗНАЧЕНИЕ ВЫВОДОВ И СТРУКТУРА УНИВЕРСАЛЬНОГО МИКРОПРОЦЕССОРА

Классический микропроцессор i8080

Микропроцессор i8080 был выпущен в 1973 г. Это был первый по-настоящему универсальный микропроцессор, пригодный для создания универсальных ЭВМ, на основе которого был создан первый популярный персональный компьютер Altair 8800 (см. п. 1.2). Он является классикой микропроцессорной техники, поскольку в нем были воплощены основные решения, которые и сейчас сохраняются в виде определенного подмножества во множестве характеристик современных процессоров: и в множестве сигналов, и в системе команд, и в множестве регистров, и в целом в архитектуре. Поэтому важно отметить, что информация, приведенная в этом пункте, относится не только к процессору i8080, но и в целом к основам микропроцессорной техники, т. е. носит концептуальный характер. Популярная книга

Роджера Токхайма (Roger Tokheim) «Микропроцессоры: Курс и упражнения» посвящена именно этому процессору, однако рассмотрение основ микропроцессорной техники в ней начинается с еще более простого, «гипотетического» процессора [55]. В нашем учебном пособии такой базой является сам процессор i8080.

Процессор i8080 имеет следующие характеристики:

- технология изготовления n-MOS;
- количество транзисторов — 4800;
- тактовая частота — 2 МГц;
- корпус — DIP-40;
- разрядность шины данных — 8;
- разрядность шины адреса — 16;
- объем адресного пространства — 64 Кбайт (используются все 16 адресных линий);
- количество портов ввода-вывода — 256 (используются 8 адресных линий, старшая и младшая половина дублируют друг друга);
- количество команд — 75.

Рассмотрим его внешнюю и внутреннюю стороны — множество выводов и структуру.

Все микросхемы, другие элементы и устройства ЭВМ связаны между собой посредством выводов и проводников, по которым передаются и принимаются сигналы. Проводники и передаваемые по ним сигналы делятся на три группы: шину адреса, шину данных и шину управления (см. п. 2.2). Большинство сигналов ЭВМ формируется на основе сигналов процессора.

Выводы процессора и других цифровых микросхем можно классифицировать следующим образом:

1. Информация (сигналы) или энергия (питание).
2. Для сигналов: данные, адрес, управление.
3. По направлению передачи информации (так же только для сигналов): входы, выходы, двунаправленные.
4. Для выходных сигналов по количеству состояний — два (логический 0 или 1) или три (0, 1, Z (высокое сопротивление)).

Состояние высокого сопротивления означает просто отключение вывода от шины, что требуется для исключения возможности конфликта уровней, поскольку, например, как уже говорилось, все устройства ЭВМ связаны между собой по линиям данных.

Тип корпуса процессора i8080 — DIP-40 (рис. 3.6).

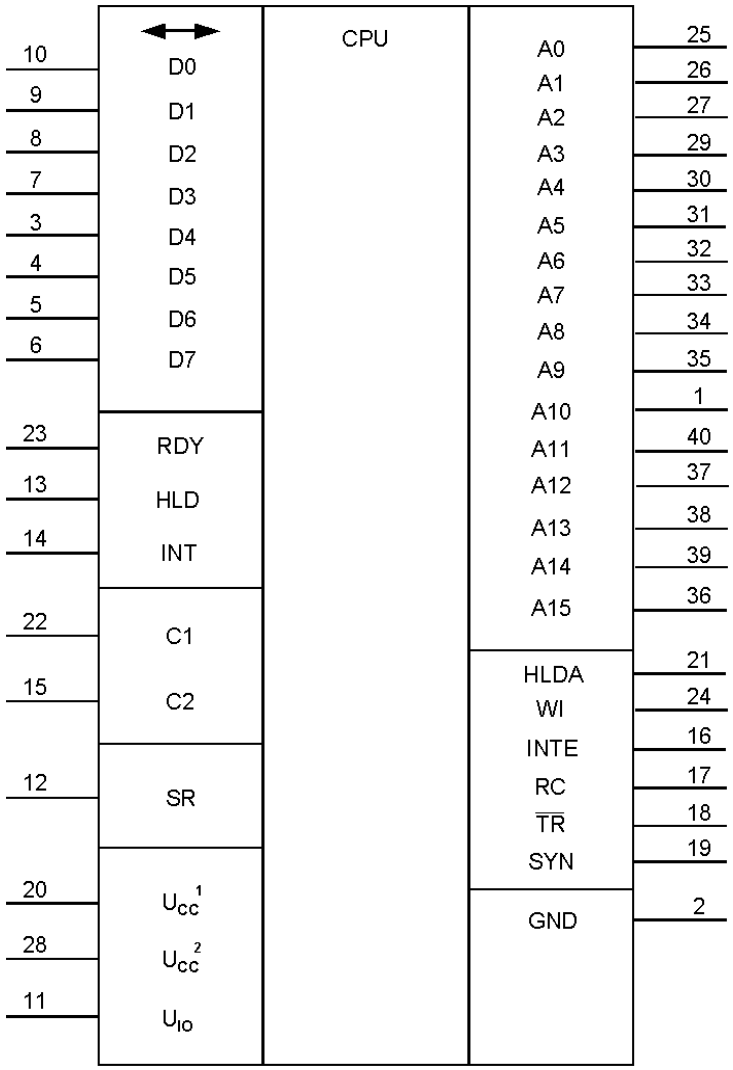


Рис. 3.6. Условное графическое обозначение процессора i8080

Назначение выводов процессора i8080 таково:

− *питание:*

  − U_{CC1}, U_{CC2}, U_{IO}: +5В, +12В, —5В соответственно;

  − GND (Ground) — земля или общая шина.

  − входы-выходы: D7–D0 — данные;

− *входы:*

− RDY (Ready) — готовность, сбрасывается внешним устройством в случае неготовности;

− HLD (Hold) — захват, устанавливается внешним устройством для захвата управления шиной;

− INT (Interrupt) — запрос прерывания;

− C1, C2 — тактовые импульсы;

− SR (Set/Reset) — сброс (установка в начальное состояние);

− *выходы:*

− A15–A0 — адрес;

− HLDA (Hold Available) — подтверждение захвата, устанавливается в ответ на сигнал HLD;

− WI (Waiting) — ожидание, устанавливается при переходе RDY на низкий (неактивный) уровень;

− INTE (Interrupt Enable) — разрешение прерывания;

− RC (Receiving) — прием (другие обозначения этого сигнала: RD, R (Reading), DBIN (Data Bus Input));

− TR# (Transmission) — передача (другие обозначения: WR#, W# (Writing));

− SYN — синхронизация, выдается в начале каждого машинного цикла.

У всех сигналов, кроме TR#, активный уровень высокий, т. е. единичка. У сигнала TR# активный уровень низкий, о чем говорит знак решетки или черта сверху в обозначении сигнала.

По входам C1, C2 процессор принимает две последовательности тактовых импульсов, формируемые генератором.

Сигнал SR устанавливает процессор в начальное состояние, которое строго определено: прерывания запрещены, процессор начинает машинный цикл M1 — чтение команды из памяти по начальному (в данном случае нулевому) адресу.

Сигнал синхронизации SYN выдается в начале каждого машинного цикла и свидетельствует, что процессор приступил к выполнению очередного машинного цикла.

Сигналы приема и передачи (чтения, записи) уже нам знакомы, они были рассмотрены в п. 2.2. Их значение определяет направление

обмена информацией к процессору или от процессора. Они могут обозначаться по-разному даже для одной и той же микросхемы:

− прием или чтение: RC (Receiving), R, RD (Reading), DBIN (Data Bus Input);

− передача или запись: TR# (Transmission), WR#, W#(Writing).

Процессор устанавливает высокий уровень сигнала RC, если необходимо провести чтение данных из памяти или ввод данных из внешнего устройства.

Процессор устанавливает низкий уровень сигнала TR# для записи данных в память или для вывода данных во внешнее устройство.

Оставшиеся три входных и три выходных управляющих сигнала тесно связаны между собой попарно:

− RDY — WI;

− HLD — HLDA;

− INTE — INT.

Сигналы RDY — WI используются для синхронизации работы процессора с медленными устройствами ввода-вывода. Внешнее устройство в случае неготовности подает на вход RDY логический ноль, т. е. переводит сигнал в неактивное состояние. Процессор переходит в состояние ожидания готовности внешнего устройства и информирует об этом окружение установкой сигнала WI в активное состояние. Эти сигналы можно использовать для организации пошагового режима работы процессора при выполнении программы, когда после каждой команды или машинного цикла процессор переходит в состояние ожидания и можно посмотреть значения, находящиеся на шинах данных, адреса и управления с помощью средств измерения или индикации.

Сигналы HLD — HLDA используются для передачи функций по управлению шиной от процессора к внешнему устройству. Внешнее устройство для захвата шины формирует логическую единицу на входе HLD, процессор по завершении текущих действий отключается от шины, информируя об этом окружение выдачей сигнала HLDA.

После этого формирование и выдачу адресов, сигналов записи и чтения осуществляет внешнее устройство. Это называется прямым доступом к памяти.

Сигналы INTE — INT используются для реализации возможности реагирования процессора на внешние аппаратные прерывания. Процессор после включения компьютера постоянно выполняет цикл чтения, дешифрации и исполнения команд. Конечно, при этом он может переходить в состояния ожидания, захвата или останова. В любом случае он должен реагировать на различные события, которые могут происходить при работе вычислительной системы: нажатия на клавиши, появление сигнала на выходе таймера, приход данных от внешнего устройства и т. п. Такого рода события приводят к формированию сигнала INT на входе процессора. Если прерывания разрешены, а это определяется значением сигнала INTE (INTE = 1 — прерывания разрешены, INTE = 0 — прерывания запрещены), то процессор заканчивает выполнение текущей команды и переходит к выполнению обработчика прерывания — подпрограммы, алгоритм которой определяется смыслом данного прерывания. Например, если прерывание сгенерировано портом ввода, процессор должен прочитать из него данные. Процессор i8080 для перехода к обработчику прерывания должен принять команду вызова подпрограммы. Основные принципы фон-неймановской архитектуры: в запоминающем устройстве находится программа (принцип хранимой программы), устройство управления (процессор) читает команды программы и выполняет их. Но при возникновении аппаратного прерывания команда вызова подпрограммы не может быть прочитана из памяти, ее там просто нет и не может быть. Аппаратное прерывание — асинхронное событие, которое в общем случае может произойти в любой момент и заранее не может быть предсказано. Поэтому команда вызова подпрограммы обработчика аппаратного прерывания формиру-

ется аппаратно-универсальными микросхемами, такими как триггеры, регистры и логические элементы, или специализированными, такими как контроллер прерываний. Для работы с процессором i8080 предназначена микросхема i8259 — программируемый контроллер прерываний. Аппаратура выдает эту команду процессору в ответ на формируемый им выходной сигнал INTA (Interrupt Available), которому не соответствует отдельный вывод через корпус процессора. Сигнал INTA входит в состав байта состояния процессора (см. далее). Контроллер прерываний i8259 имеет восемь входных линий IRQ (Interrupt Request). Номер входа IRQ, к которому подключено данное устройство, — это один из видов аппаратных ресурсов системы (см. п. 2.2). В IBM-совместимых персональных компьютерах используются контроллеры прерываний, совместимые с i8259A, которые расширяют по функциональности i8259, но количество входов запроса прерывания IRQ у них такое же — восемь.

Конечно, в работе вычислительной системы есть периоды, в течение которых прерывания должны быть запрещены. Например, после включения питания компьютера или при получении сигнала сброса прерывания запрещены, поскольку вначале должны проработать программы проверки и инициализации (настройки) аппаратуры, в том числе контроллера прерываний для правильной их обработки. Прерывания запрещаются и разрешаются программно с помощью соответствующих команд. Для процессора i8080 это команды DI (Disable Interrupts) и EI (Enable Interrupts).

Сигналы A15–A0 — шина адреса процессора. Число адресных линий — 16, таким образом, размер адресного пространства — 64 Кбайта. При адресации портов ввода-вывода адрес 8-битный. Он дублируется на старшей и младшей половине шины адреса, т. е. A15–A8 = A7–A0. Поэтому размер пространства ввода-вывода — 256 портов. По шине адреса в первом такте каждого машинного цикла про-

цессор выдает адрес ячейки памяти или порта ввода-вывода, по которому он будет записывать или читать информацию по шине данных.

Шина данных процессора — линии D7–D0. Однако в первом такте каждого машинного цикла процессор выводит по линиям данных управляющие сигналы, которые в совокупности представляют собой так называемый байт состояния, значение которого определяет тип машинного цикла. Таким образом, линии D7–D0 не только шина данных, но и шина управления. Здесь можно отметить, что ограничение на количество выводов в корпусе процессора является существенным фактором. Поэтому многие выводы имеют альтернативное назначение, когда одному выводу соответствует более одного сигнала. Это имеет место и в других процессорах.

Формат байта состояния процессора:

- D7 — MR (Memory Read), чтение памяти;
- D6 — INP (Input), ввод;
- D5 — M1, машинный цикл номер 1, прием первого байта команды;
- D4 — OUT (Output), вывод;
- D3 — HLTA (Halt Available), подтверждение останова;
- D2 — STACK, стек;
- D1 — WO# (Write-Output), запись-вывод;
- D0 — INTA (Interrupt Available), подтверждение прерывания.

Поскольку в третьем такте машинного цикла по шине данных происходит обмен информацией между процессором и его окружением, байт состояния должен быть записан (защелкнут) в специальную или универсальную микросхему — системный контроллер или регистр общего назначения — для того, чтобы управляющие сигналы сохраняли необходимые значения в течение всего машинного цикла.

Процессор i8080 имеет десять типов машинных циклов, которым соответствуют определенные значения байта состояния (табл. 3.2).

Таблица 3.2

Типы машинного цикла процессора i8080

Тип машинного цикла	Байт состояния							
	D7 MR	D6 INP	D5 M1	D4 OUT	D3 HLTA	D2 STACK	D1 WO#	D0 INTA
Выбор команды	1	0	1	0	0	0	1	0
Чтение памяти	1	0	0	0	0	0	1	0
Запись в память	0	0	0	0	0	0	0	0
Чтение стека	1	0	0	0	0	1	1	0
Запись в стек	0	0	0	0	0	1	0	0
Ввод	0	1	0	0	0	0	1	0
Вывод	0	0	0	1	0	0	0	0
Прерывание	0	0	1	0	0	0	1	1
Останов	1	0	0	0	1	0	1	0
Прерывание при останове	0	0	1	0	1	0	1	1

После выдачи байта состояния процессор по линиям данных обменивается информацией с памятью или устройствами ввода-вывода, принимает или передает данные. Передаваемые данные — значения, записываемые в стек, в переменные в памяти или в порты вывода. Принимаемые данные — это команды, значения, читаемые из переменных или вводимые из портов. Они интерпретируются процессором в зависимости от типа машинного цикла и кода операции выполняемой команды. В первом машинном цикле любой команды данные интерпретируются как код операции команды. Данные, принимаемые в следующих циклах, воспринимаются как операнды или адреса операндов в памяти или внешних устройствах. При этом надо учесть, что операнды в силу специфики действия, выполняемого командой, тоже

могут быть адресами. Такими командами являются, например, команды передачи управления (перехода).

Принимаемые данные записываются во внутренние запоминающие элементы процессора — регистры. Код операции команды помещается в регистр команд, операнды и адреса операндов — в регистры, определяемые командой, т. е. ее кодом операции.

Внутреннее устройство процессора i8080 представлено его структурной схемой (рис. 3.7).

Основными устройствами любого процессора являются устройство управления и арифметико-логическое устройство (АЛУ).

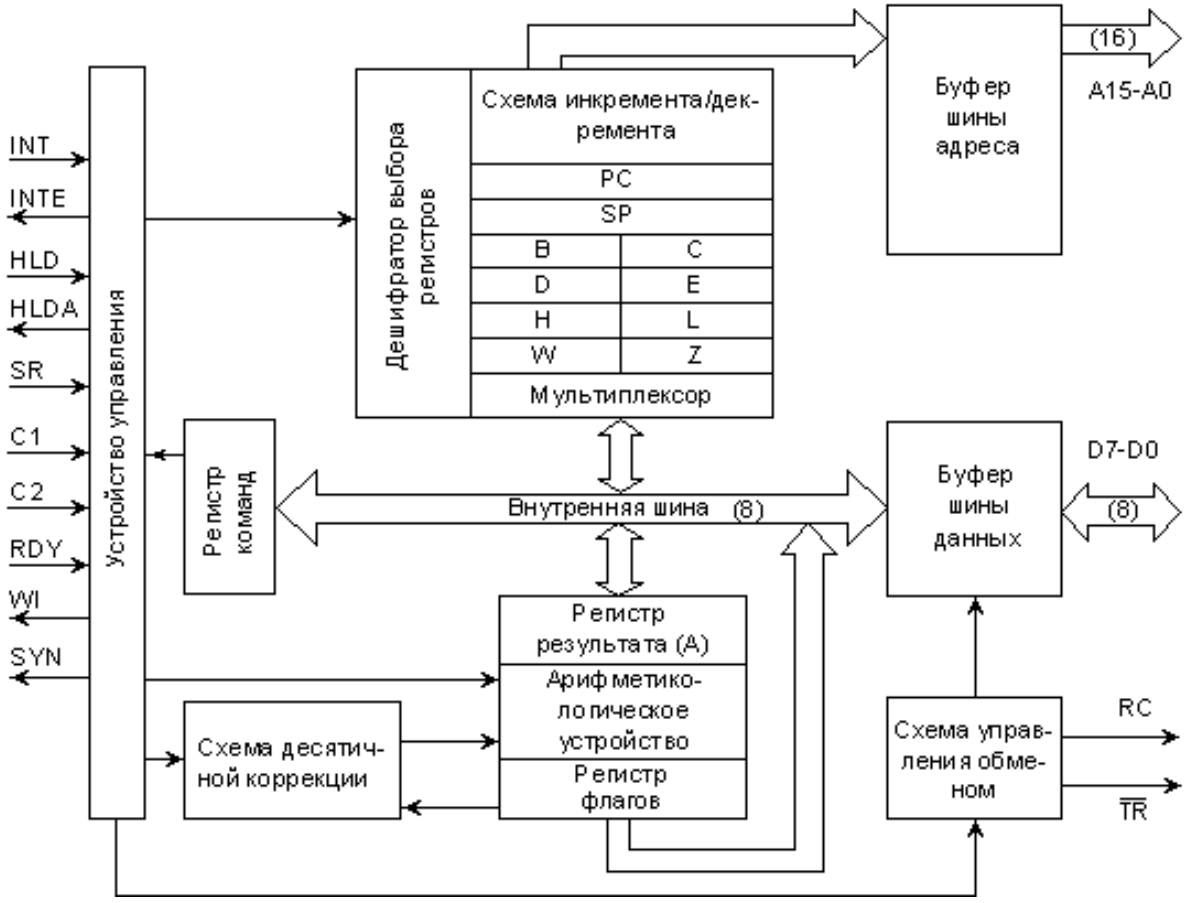


Рис. 3.7. Структурная схема процессора i8080

АЛУ выполняет арифметические и логические операции, а устройство управления организует выполнение команд в целом. Устройство управления в первом цикле команды извлекает код опе-

рации (КОП, COP) команды и помещает его в регистр команд. Дешифратор команд (на рисунке не показан) декодирует его, т. е. распознает, какая это команда, и по результатам декодирования и текущим значениям входных управляющих сигналов устройство управления формирует необходимые выходные и внутренние управляющие сигналы, которые организуют дальнейший процесс выполнения команды.

Схема управления обменом является частью устройства управления и вместе с ним формирует сигналы приема и передачи, а также переключает буфер шины данных, который связывает внутреннюю шину процессора с внешними линиями данных, на прием или передачу.

АЛУ наиболее тесно связано с аккумулятором (регистром результата) и регистром флагов. В большинстве арифметических и логических команд по умолчанию используется аккумулятор как источник одного из операндов и как приемник результата. Основные признаки результата (ноль или не ноль, положительный или отрицательный и т. п.) должны быть зафиксированы. Для этого служит регистр флагов.

Другие регистры процессора изображены в отдельном блоке. Регистры, которые используются в выполняемой команде, выбираются дешифратором, связанным с устройством управления, и подключаются мультиплексором.

Схема инкремента-декремента инкрементирует указатель команд (регистр PC) на линейных участках программы и указатель стека (регистр SP) при извлечении данных из стека, декрементирует указатель стека при помещении данных в стек. Адрес из PC, SP или других регистров, которые могут быть источником адреса, выдается через буфер шины адреса.

С АЛУ и регистром флагов связана схема десятичной коррекции. Она осуществляет преобразование результата арифметических операций над двоично-десятичными числами в двоично-десятичный код. Тип данных процессора i8080 — целые двоичные числа в дополнительном коде. Исторически в финансово-экономических областях чаще использовались двоично-десятичные числа. Двоично-

десятичный код — форма записи десятичных чисел, когда каждая цифра десятичного числа записывается двоичным числом. Двоично-десятичные числа бывают упакованными или неупакованными. В упакованных десятичная цифра представлена четырьмя битами, в неупакованных — восемью, т. е. байтом. Нужно различать тип данных программы и тип данных процессора. Процессор, выполняя арифметические команды, считает операнды числами в дополнительном коде. Если в программе они считаются двоично-десятичными числами, то результат получится неправильный, и его нужно скорректировать. Это делается по команде DAA.

Пример. Сложение десятичных чисел 38 и 26, представленных в двоично-десятичном коде:

38	00111000
+26	+00100110
64	01011110 – 5E ≠ 64
Дес. кор.	+00000110 – 6 (15 – 9 = 6)
	01100100 – 64

Важнейшая характеристика процессора — множество программно-доступных регистров. Процессор i8080 имеет десять таких регистров:

− PC (Program Counter) — счетчик программы;

− SP (Stack Pointer) — указатель стека;

− шесть регистров общего назначения (РОН или GPRs, General Purpose Registers): B, C, D, E, H, L;

− A — аккумулятор;

− F — регистр флагов.

Счетчик программы PC содержит адрес очередного байта команды. Другое название этого регистра — указатель команд (IP, Instruction Pointer). На линейных участках программы он инкрементируется схемой инкремента-декремента в каждом машинном цикле.

При передаче управления его содержимое заменяется адресом перехода.

Указатель стека SP содержит адрес верхушки стека — последней заполненной ячейки стека. Стек — это специальная область памяти, предназначенная для хранения временной информации: адресов возврата из подпрограмм, параметров, переданных подпрограмме, локальных данных подпрограммы и т. п. Порядок работы со стеком следующий. При помещении в стек SP декрементируется, и в новую верхушку записывается операнд. При извлечении из стека операнд читается по текущему значению SP, и после этого SP инкрементируется. Таким образом, стек представляет собой буфер, организованный по принципу LIFO (Last In, First Out, последним вошел, первым вышел) или FILO (First In, Last Out, первым вошел, последним вышел). Стек в документации чаще всего изображают таким образом, что внизу находятся младшие адреса, а вверху старшие. И говорят, что стек «растет сверху вниз». Но чтобы не было путаницы и зависимости от порядка расположения адресов, нужно запомнить, что стек растет от старших адресов к младшим. В командах работы со стеком размер операнда — два байта. Шина данных процессора 8-разрядная, и помещение или извлечение из стека осуществляется за два машинных цикла, не считая цикла чтения кода операции. В системах на основе процессоров Intel принят так называемый обратный порядок хранения слов (little endian order в противоположность big endian order — прямому порядку хранения слов) в памяти, при котором по порядку возрастания адресов сначала хранится младший байт, затем старший. В соответствии с ним сначала в стек помещается старший байт, затем младший. Извлечение из стека происходит в обратном порядке.

Регистры общего назначения B, C, D, E, H, L используются в арифметических и логических командах как источники одного из входных операндов, в командах передачи данных как операнды-источники или операнды-приемники. Также они могут использоваться парами как три 16-разрядных регистра BC, DE, HL. В этом каче-

стве они имеют специализацию указателей данных (содержат адрес операнда): HL — в командах перемещения данных (MOV), арифметических и логических командах с операндом в памяти данных; BC, DE — в командах загрузки или размещения аккумулятора.

Аккумулятор также относится к регистрам общего назначения. Однако в процессоре i8080 он имеет максимальную специализацию. По умолчанию (неявно) аккумулятор используется в качестве источника и приемника во многих командах. По этой причине он сильнее других регистров связан с АЛУ.

В регистр флагов F АЛУ записывает признаки результата арифметической или логической операции. Другого типа флагов в процессоре i8080 нет. ***Формат регистра F таков:***

7	6	5	4	3	2	1	0
S	Z		AC		P		CY

S (Sign) — флаг знака, устанавливается, если результат арифметической или логической команды — отрицательное число, т. е. повторяет старший бит результата;

Z (Zero) — флаг нуля, устанавливается, если результат арифметической или логической команды равен нулю;

AC (Auxiliary Carry) — флаг дополнительного переноса, устанавливается, если был перенос из 3-го разряда в 4-й или же заем из 4-го разряда в 3-й при выполнении арифметических команд;

P (Parity) — флаг паритета (четности), устанавливается, если число единичек в результате арифметической или логической команды четно;

CY (Carry) — флаг переноса, устанавливается в следующих случаях:

− если был перенос из старшего разряда результата операции сложения или заем в старший разряд уменьшаемого при выполнении команды вычитания;

− если был сдвиг единички из крайнего разряда аккумулятора при выполнении команд сдвига;

– если выполнилась команда установки признака переноса или его инвертирования в случае первоначального нулевого значения.

В противном случае флаги сбрасываются, это важно иметь в виду: флаги не только устанавливаются, но и сбрасываются по результату арифметической или логической команды.

Значения флагов влияют на процесс ветвления программы (передачу управления, поток управления). Ветвления: переходы, вызовы подпрограмм и возвраты из подпрограмм — бывают безусловными и условными. Последние осуществляются при выполнении какого-либо условия. Количество возможных условий и, соответственно, количества условных переходов, вызовов и возвратов связаны с количеством флагов следующим образом:

$$NC=2(NF-1),$$

где NC — количество условий;

NF — количество флагов.

Единственным флагом, который не порождает никаких условий, является флаг дополнительного переноса. Он используется процессором при выполнении команды десятичной коррекции аккумулятора DAA. Оставшиеся четыре флага соответствуют следующим условиям:

- перенос (CY = 1);
- нет переноса (CY = 0);
- чет (P = 1);
- нечет (P = 0);
- ноль (Z = 1);
- не ноль (Z = 0);
- минус (S = 1);
- плюс (S = 0).

По этой причине регистр флагов называется также регистром условий.

Признаки результата предыдущей команды определенным образом характеризуют состояние процессора и программы. Поэтому регистр флагов называют еще и регистром состояния. А совокупность аккумулятора и регистра флагов — словом состояния процессора или программы (PSW, Processor (Program) Status Word). Однако нужно иметь в виду, что термин состояние используется также и в другом смысле, как тип машинного цикла, характеризуемый байтом состояния.

Хотя аккумулятор и называется по-другому регистром результата, не всегда результат выполнения арифметической команды оказывается в нем. Поэтому надо отметить, что значения флагов устанавливаются именно по результату команды.

Есть также программно-недоступные регистры:

− WZ — регистр временного хранения;

− регистр команд.

Имя WZ условное, оно никак не используется в командах в качестве операнда. Размер регистра WZ — 16 бит. Он используется при обменах, например, в команде XTHL (eXchange Top with HL) — обмене пары HL с верхушкой стека.

У регистра команд вообще нет имени. Можно также условно для краткости обозначить его как IR (Instruction Register).

Счетчик программы PC и регистр флагов F занимают промежуточное положение. Они считаются программно-доступными регистрами.

У каждого символьного имени есть числовой эквивалент. Как уже говорилось, программа, записанная на каком-либо языке программирования, преобразуется компилятором в машинный код — двоичные числа. Код операции — число, в котором есть множество полей: основное — информация о действии (операции), и дополнительные — информация об операндах. Именам регистров соответствуют числа:

000 — B;

001 — C;

010 — D;
011 — E;
100 — H;
101 — L;
110 — M (memory, ячейка памяти);
111 — A;
00 — BC;
01 — DE;
10 — HL;
11 — SP.

Имени переменной также соответствует число — ее адрес в памяти.

Регистры PC и F не кодируются. Они используются в командах неявно. PC изменяется при выполнении любых команд, не только команд передачи управления. Правда, в системе команд i8080 есть команда PCHL — помещения в регистр PC значения HL — указателя данных в указатель команд. В ней имя PC используется явно, но оно присутствует в мнемонике команды, а не в качестве операнда. И вообще, в целом в программировании такие действия не приветствуются, так что в последующих процессорах Intel семейства x86 таких команд нет.

Регистр F изменяется при выполнении арифметических и логических команд. Однако у процессора также есть команда, которая явно влияет на регистр флагов — POP PSW — извлечение из стека слова состояния программы, младшим байтом которого является регистр флагов. Есть команды, которые явно влияют на флаг переноса: STC (Stand Carry, установить флаг переноса), CMC (Compliment Carry, инвертировать флаг переноса). В других процессорах для каждого флага могут быть команды установки и сброса.

Единственным типом данных процессора i8080 являются двоичные числа в дополнительном коде. То есть при выполнении арифметических операций процессор рассматривает операнды как числа в дополнительном коде. Положительные числа в дополнительном коде

представляются так же, как и в прямом коде, т. е. как обычные двоичные числа. Отрицательные числа в дополнительном коде представляются следующим образом:

1. В двоичной системе счисления записывается абсолютное значение (модуль) данного отрицательного числа. Количество разрядов соответствует размеру операнда.
2. Разряды инвертируются.
3. К полученному числу прибавляется единица.

Пример. Представление числа –9 в дополнительном коде:

1. $9_{10} = 00001001_2$;
2. Побитная инверсия дает 11110110;
3. Прибавляем единичку и получаем 11110111 = F7h.

Проверка: F7 + 9 = 100.

Можно и сразу так получить дополнительный код: 100 – 9 = F7.

Видно, что у отрицательных чисел всегда в старшем разряде единица. Поэтому флаг знака S и повторяет старший бит результата.

Вычитание происходит путем сложения уменьшаемого с вычитаемым в дополнительном коде. Во флаг CY при этом записывается инвертированное значение переноса:

```
        . . . . .
_ 9   + 0 0 0 0 1 0 0 1
  6       1 1 1 1 1 0 1 0
  3     1 0 0 0 0 0 0 1 1
        ↙ – инверсия
      CY = 0
```

```
              . .
_ 6   + 0 0 0 0 0 1 1 0
  9       1 1 1 1 0 1 1 1
– 3     0 1 1 1 1 1 1 0 1
        ↙ – инверсия
      CY = 1
```

Флаг AC командами вычитания не инвертируется. В этом флаге при выполнении команд вычитания оказывается реальный перенос из третьего разряда аккумулятора в четвертый, возникающий при сложении уменьшаемого и взятого с обратным знаком вычитаемого. Рас-

смотрим в качестве примера команду SUB B (A = A – B) при условии, что A = 02h, а B = 01h:

```
          . . . . . . .
_ 2   + 0 0 0 0 0 0 1 0
  1       1 1 1 1 1 1 1 1
  1   1 0 0 0 0 0 0 0 1
      ↙ – инверсия
    CY = 0
```

После выполнения этой команды A = 01h, F = 12h, т. е. AC = 1, CY = 0. Видно, что перенос происходит как из третьего, так и из седьмого разряда. Перенос из третьего разряда не инвертируется (AC = 1), а из седьмого — инвертируется (CY = 0).

Следующим после i8080 был процессор i8085. Он в основном имел те же характеристики и структуру, но были и нововведения: встроенный генератор тактовых импульсов, мультиплексированная шина адреса/данных, выводы для приема и передачи последовательных данных, усовершенствованная система прерываний, две дополнительные команды для работы с прерываниями и последовательными данными, тактовая частота 3 МГц и одно напряжение питания +5В.

Микропроцессор i8086. Семейство x86

Процессор Intel8086 (1978 г.) — первый процессор семейства x86. ***Он имеет следующие характеристики:***

- разрядность шины данных — 16;
- разрядность шины адреса — 20;
- количество команд — 135;
- количество транзисторов — 29 000;
- тактовая частота 2–5 МГц;
- корпус — DIP-40.

Таким образом, размер адресного пространства i8086 — 1 Мбайт. При адресации внешних устройств используются только

16 адресных линий, поэтому размер пространства ввода-вывода — 65 536 портов.

Годом позже появился процессор i8088. Его единственное отличие от i8086 — 8-разрядная внешняя шина данных. Это дало возможность использования развитой аппаратной базы i8080 для создания вычислительных систем на его основе. Благодаря наличию очереди команд, производительность i8088 в среднем лишь на 20 % ниже, чем у i8086.

В дальнейшем в последующих процессорах семейства x86 расширялась функциональность, увеличивалась разрядность и, конечно, производительность, но в целом сохранялась программная модель, так что все процессоры этого семейства программно совместимы снизу вверх, т. е. программа, написанная для младших моделей, будет работать и на старших, за некоторыми исключениями.

Для совместной работы с процессором i8086 был разработан и выпущен математический сопроцессор i8087, который также развивался параллельно с центральными процессорами, а начиная с процессора i486 входит в его состав в виде FPU, который обозначается в документации как FPU x87. Использование математического сопроцессора позволило создавать на основе i8086 мощные функциональные вычислительные системы. Именно на процессорах этого семейства компания IBM разработала свои первые персональные компьютеры.

Несмотря на то, что, как и i8080, процессор i8086 был выпущен в корпусе DIP-40, благодаря альтернативному назначению выводов и наличию двух режимов работы, минимального и максимального, множество сигналов существенно расширилось (рис. 3.8).

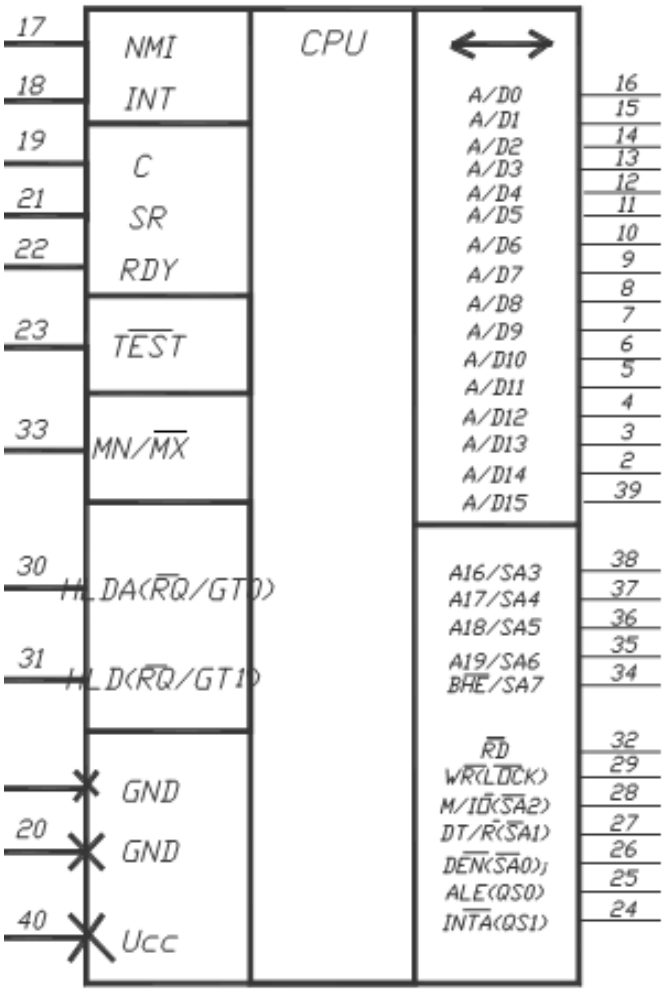


Рис. 3.8. Условное графическое обозначение процессора i8086

Режим работы процессора выбирается при проектировании вычислительной системы: минимальный, если создается не очень сложная система, максимальный, если более сложная, с системным контроллером. Режим задается аппаратно с помощью вывода MN/MX#. Процессор использует только одно напряжение питания — +5 В. Если вывод MN/MX# подключается к шине питания (Ucc), то процессор будет работать в минимальном режиме, если к общей шине, земле (GND), — то в максимальном.

В обозначении некоторых выводов на рис. 3.8 есть скобки. Вне скобок приведены сигналы, соответствующие данным выводам в минимальном режиме, в скобках — в максимальном.

Сигналы процессора i8086:

- *входы-выходы*:
  - A/D15–A/D0 — мультиплексированная шина адреса/дан-ных;
  - RQ#/GT1, RQ#/GT0 — запрос-подтверждение (Request/ Grant) на управление шиной;
- *входы*:
  - NMI (Non Maskable Interrupt) — немаскируемый запрос прерывания;
  - INT — маскируемый запрос прерывания;
  - C — тактовый сигнал;
  - SR — установка-сброс;
  - RDY — готовность;
  - TEST# — проверка;
  - MN/MX# — минимальный или максимальный режим;
  - HLD — захват;
- *выходы*:
  - HLDA — подтверждение захвата;
  - A19–A16 — старшие четыре линии канала адреса;
  - BHE# — разрешение старшего байта;
  - RD# — чтение;
  - WR# — запись;
  - M/IO# — память/устройство ввода-вывода;
  - DT/R# — передача-прием данных;
  - DEN# — разрешение передачи-приема данных;
  - ALE (Address Lock Enable) — стробирующий сигнал адреса;
  - INTA# — подтверждение прерывания;
  - SA7–SA3, SA2#–SA0# — состояние:
    - SA5 — прерывания разрешены или запрещены;
    - SA4–SA3 — сегментный регистр, по которому формируется физический адрес;
    - SA2#–SA0# — тип цикла шины в максимальном режиме;
  - LOCK# — блокировка шины;
  - QS1, QS0 (Queue State) — состояние очереди команд.

В процессоре i8080 были совмещены шина данных и часть шины управления. В i8086 — шина данных и младшие 16 линий шины адреса. Вначале по выводам A/D15–A/D0 выдается адрес. Это подтверждает активный уровень сигнала ALE, по которому адрес защелкивается в формирователе шины адреса. Затем по этим линиям происходит обмен данными. Сигнал BHE# означает, что 8-разрядные данные передаются по старшей половине шины адреса-данных A/D15–A/D8.

Сигналы RQ#/GT1, RQ#/GT0 используются в максимальном режиме вместо сигналов HLD и HLDA. Каждый из выводов 30 и 31 является в этом случае двунаправленным. Вначале по нему принимается сигнал запроса на управления шиной, а затем выдается сигнал подтверждения. Эти сигналы, как и HLD и HLDA, используются для организации режима прямого доступа к памяти.

Сигнал NMI — немаскируемый запрос прерывания. Обработка немаскируемого прерывания не зависит от того, разрешены прерывания или нет. Как уже говорилось, на определенных этапах работы процессора прерывания должны быть запрещены, чтобы процессор выполнил определенную последовательность команд непрерывно. Но если происходят критические события, например, ошибка памяти и т. п., дальнейшее выполнение текущих команд не имеет смысла. Для реакции на такого рода события и предназначен сигнал NMI. При этом сигнал INTA не вырабатывается. Он вырабатывается как и прежде по сигналу INT, но в отличие от i8080, по сигналу INTA i8086 получает от контроллера прерываний i8259A не всю команду вызова подпрограммы, а только номер прерывания, называемый также вектором, по которому определяет адрес обработчика через таблицу векторов прерываний.

TEST# — сигнал, используемый совместно с командой WAIT. Выполняя ее, процессор проверяет уровень этого сигнала. Если он активный, то процессор переходит к выполнению следующей коман-

ды. В противном случае он вводит холостые такты и пребывает в состоянии ожидания. Это используется для синхронизации работы процессора и других устройств, например, математического сопроцессора.

Сигнал блокировки шины LOCK# используется в максимальном режиме. Он вырабатывается по команде (префиксу) LOCK и информирует другие устройства о том, что они не должны пытаться запрашивать шину.

Для информирования окружения о своем текущем состоянии и выработки необходимых системных управляющих сигналов в процессоре i8086 имеется восемь сигналов состояния, тоже байт, но формат его сильно отличается от такового у процессора i8080. Для обозначения текущего цикла шины используются три младших сигнала состояния (табл. 3.3).

Таблица 3.3

Типы цикла шины процессора i8086

SA2#	SA1#	SA0#	Цикл шины
0	0	0	подтверждение прерывания
0	0	1	ввод
0	1	0	вывод
0	1	1	останов
1	0	0	извлечение кода операции команды
1	0	1	чтение
1	1	0	запись
1	1	1	пассивный (цикла шины нет)

Теперь используется термин цикл шины, а не машинный цикл, поскольку в структуре процессора появился новый уровень (рис. 3.9, 3.10).

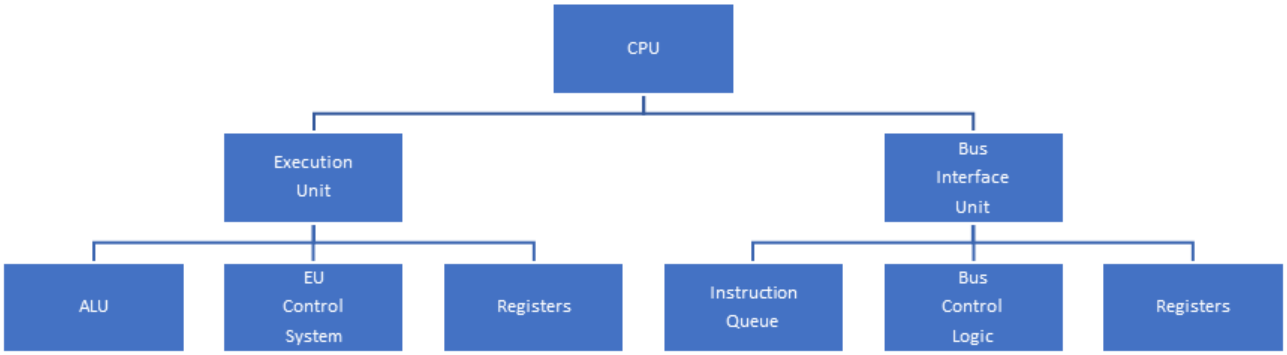


Рис. 3.9. Иерархическая структура процессора i8086

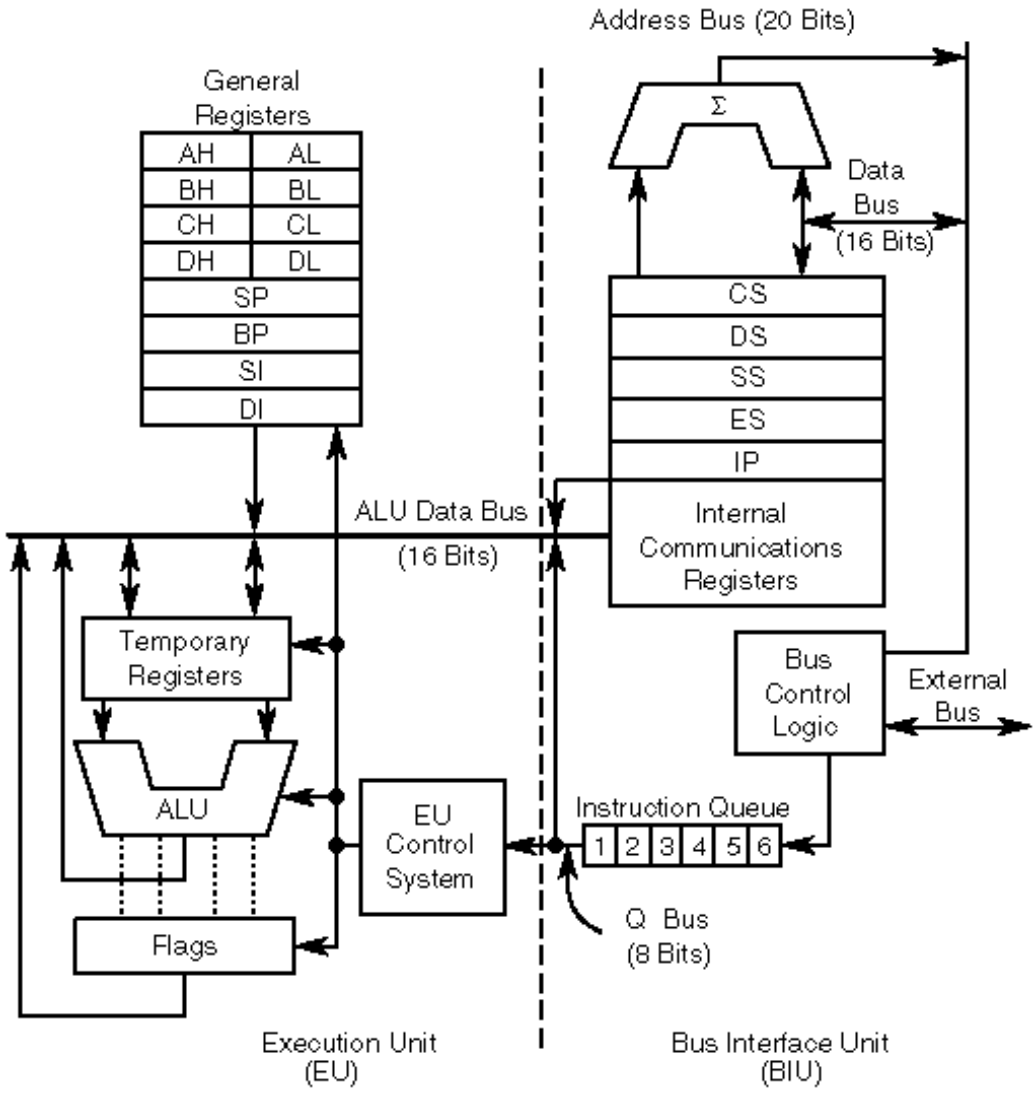


Рис. 3.10. Структурная схема процессора i8086

Процессор i8086 на верхнем уровне делится на две части: устройство исполнения (EU, Execution Unit) и устройство шинного

интерфейса (BIU, Bus Interface Unit). АЛУ входит в состав устройства исполнения вместе с регистрами общего назначения, временными регистрами и регистром флагов, а устройство управления распределилось по этим двум частям.

Введение в состав процессора устройства шинного интерфейса в качестве самостоятельного модуля — принципиальный момент. Разделение интерфейса и реализации, блоков системы, отвечающих за взаимодействие с внешним миром, и блоков, обеспечивающих ее функциональность — это общий принцип, который используется в компьютерной технике и на аппаратном уровне, и на программном. Программы на языке С состоят из интерфейсных, заголовочных файлов и файлов реализации. В объектно-ориентированных программах основная структурная единица, класс, также имеет интерфейсную и внутреннюю составляющие. Отделение интерфейса от реализации предоставляет много преимуществ: повышение эффективности работы системы, большую защищенность, упрощение и ускорение развития и модернизации и т. п.

В устройство шинного интерфейса входят:

- очередь команд (прообраз кэш-памяти команд);
- устройство управления шиной;
- группа сегментных регистров вместе с указателем команд;
- сумматор, формирующий физический адрес из логического.

Благодаря очереди команд внутренние и внешние действия процессора не связаны жестко, достигается относительная независимость работы устройства исполнения и устройства шинного интерфейса. Процессор выполняет ранее выбранные команды и уже читает новые. Размер очереди команд — 6 байтов, так что в нее может поместиться одна 6-байтная команда (это максимальный размер команды у процессора i8086), две 3-байтные и т. д. Состояние очереди команд характеризуют выходные сигналы процессора QS1, QS0:

- 00 — в последнем такте ничего не выбрано;
- 01 — выбран последний байт;
- 10 — очередь реинициализирована;

− 11 — выбран текущий байт.

Общее число программно-доступных регистров процессора i8086 — 14, их размер — 16 бит. Всего можно выделить четыре группы регистров:

− регистры общего назначения (AX, CX, DX, BX, SP, BP, SI, DI — 8 шт.);

− сегментные регистры (CS, DS, ES, SS — 4 шт.);

− указатель команд (IP);

− регистр флагов (Flags).

Особенностью регистров общего назначения является то, что их можно использовать в большинстве арифметических и логических команд и как источник операнда, и как приемник результата. Аккумулятор теперь не имеет такой привилегии и входит в группу регистров общего назначения. Однако он все равно выделяется из этой группы, так как его специализация как регистра результата остается в командах умножения и деления, а также ввода-вывода. Многие другие команды с использованием аккумулятора короче, чем с другими регистрами. Но и другие регистры общего назначения также имеют свою специализацию:

− CX (Counter, счетчик), используется как счетчик цикла в команде LOOP и командах работы со строками (массивами) с префиксом повторения;

− DX (Data, данные), используется в командах умножения и деления слов, косвенного ввод-вывода;

− BX (Base, база), является аналогом регистра HL i8080, используется как источник базового адреса в командах с косвенной адресацией: MOV AX, [BX], MOV DX, VAR[BX] и т. п.

Квадратные скобки у имени регистра означают, что в регистре находится не операнд, а адрес операнда. То есть по команде MOV AX, [BX] в регистр AX помещается содержимое ячейки памяти, исполнительный адрес которой (смещение внутри сегмента) содержится в регистре BX. По команде MOV DX, VAR[BX] в регистр DX помещается содержимое ячейки памяти, исполнительный адрес которой

определяется как сумма смещения, соответствующего переменной VAR, и значения, содержащегося в регистре BX.

Буква X в названиях этих регистров от слова eXtended (расширенный). Это традиционное использование буквы X в различных названиях. В процессорах i8080 и i8085 были 8-битные регистры A, B, C и D, а здесь они расширены до 16 бит.

Регистры AX, BX, CX и DX можно выделить в качестве отдельной подгруппы. Ее особенность в том, что каждый из этих регистров состоит из двух частей, старшего и младшего байтов, обращаться к которым можно отдельно по именам: AH, AL, BH, BL, CH, CL, DH, DL.

Оставшиеся четыре регистра также выделяются в отдельную подгруппу указательных и индексных регистров:

– SI (Source Index, индекс источника), используется в строковых командах в качестве указателя на операнд-источник;

– DI (Destination Index, индекс приемника), используется в строковых командах в качестве указателя на операнд-приемник;

– BP (Base Pointer, указатель базы), используется в командах подпрограмм для доступа к внутренним элементам стека при передаче параметров подпрограмме через стек;

– SP (Stack Pointer, указатель стека), как и в i8080, указывает на верхушку стека, используется в командах помещения в стек, извлечения из стека, вызова подпрограмм и возврата из подпрограмм.

Данные регистры хранят внутрисегментные адреса (смещения). По умолчанию регистр SI сегментируется по DS, DI — по ES; BP и SP — по SS.

По другому признаку могут быть выделены еще две подгруппы регистров-указателей — базовых (BX, BP) и индексных (SI, DI). Они могут быть использованы как модификаторы адреса. В этом заключается основная специализация данных регистров. Модификация может осуществляться как по одному, так и по двум регистрам из разных подгрупп: первый регистр из пары модификаторов должен быть BX или BP, а второй — SI или DI.

Модификация адреса — одно из новшеств архитектуры i8086. Адрес операнда команды в памяти может вычисляться сложным образом, например: MOV DX, VAR[BX][SI–4]. Здесь из адреса переменной VAR компилятор сразу вычтет число 4 и поместит полученное значение в машинный код команды в качестве прямого смещения. Процессор при выполнении данной команды прочитает это смещение и сложит его со значениями регистров BX и SI. Полученная сумма называется исполнительным (эффективным) адресом. Это окончательное смещение от начала сегмента, по которому процессор рассчитает физический адрес и выдаст его по шине адреса, чтобы прочитать операнд-источник. Выгода от такого способа адресации состоит в том, что, меняя содержимое регистра-модификаторов, можно обращаться к различным ячейкам памяти, не меняя команды. Это удобно, например, при работе с массивами. Замена прямого адреса (адреса переменной), указанного в команде, на исполнительный называется модификацией адреса, а используемые при этом регистры — модификаторами. Конечно, адреса переменной в команде может и не быть: MOV AX, [BX][SI] или, что то же самое, MOV AX, [BX+SI].

Регистр SP в процессоре i8086 относится к группе регистров общего назначения. Он может использоваться в качестве операнда в любых арифметических и логических командах, но, конечно, его основное назначение, его специализация — указатель на верхушку стека.

В семействе x86 стек реализуется в виде отдельного сегмента, базу которого содержит регистр сегмента стека SS. SP содержит смещение (адрес от начала сегмента) последней заполненной ячейки стека (верхушки стека). Если в стек на данный момент ничего не помещалось, то SP указывает на неопределенные данные, не имеющие значения для программы.

Для работы со стеком предназначены команды:

– PUSH, помещение в стек;

– POP, извлечение из стека.

(Это еще один нюанс терминологии: запись в стек называется помещением, чтение — извлечением.)

Также на стек влияют команды вызова подпрограммы, возврата из подпрограммы и некоторые другие. При вызове подпрограммы в стек помещается адрес возврата, при возврате из подпрограммы он, соответственно, извлекается из стека. При переходе к обработчику аппаратного прерывания в стеке, помимо адреса возврата, сохраняется регистр флагов.

Порядок изменения SP такой же, как и в i8080: при помещении в стек SP декрементируется и затем по данному адресу записывается операнд. Извлечение из стека происходит в обратном порядке: операнд извлекается из верхушки стека, а затем SP инкрементируется. В i8086 шина данных 16-разрядная и одно слово (два байта) помещается или извлекается из стека за один цикл шины. Соответственно, величина декремента и инкремента указателя стека равна двум.

Принципиально новая группа регистров процессора i8086 — сегментные регистры:

- CS (Code Segment) — сегментный регистр сегмента команд;
- DS (Data Segment) — сегментный регистр сегмента данных;
- ES (Extra Segment) — сегментный регистр дополнительного сегмента данных;
- SS (Stack Segment) — сегментный регистр сегмента стека.

У процессора i8080 адресное пространство непрерывное (плоское). У i8086 оно является дискретным, сегментированным, т. е. разделенным на сегменты, 64-килобайтные области памяти. Принстонская архитектура, к которой относятся процессоры i8080, i8086 и последующие семейства x86 характеризуется смешанной памятью команд и данных в отличие от гарвардской архитектуры, где память команд и данных раздельная. Но в i8086 появляется возможность выделения в этой общей смешанной памяти отдельных областей с разной информацией: командами, исходными данными, результатами, временными промежуточными данными. И хотя канал с памятью по-прежнему один, это дает множество преимуществ. Само по себе структурирование (технических объектов, программы, памяти и т. п.) — вещь необходимая и полезная. Кроме того, сегментирование дает следующие преимущества:

– отсутствие необходимости иметь 20-разрядные регистры для хранения и выдачи адресов;

– уменьшение размера программы (в командах содержатся не 20-битные физические адреса, а 16-битные смещения);

– возможность создания динамически перемещаемых программных модулей при использовании позиционно-независимых команд передачи управления, т. е. содержащих не абсолютные, а относительные адреса перехода.

После появления защищенного режима, начиная с процессора i80286, появилась возможность защиты сегментов — ограничения и разграничения доступа и т. п.

Сегментные регистры хранят так называемую базу сегмента, по которой процессор определяет адрес начала сегмента, умножая ее на 16 (или сдвигая влево на один шестнадцатеричный разряд или четыре двоичных), а затем, прибавляя к нему смещение, получает физический адрес, который и выдает на шину адреса. Можно сказать, сегментные регистры являются указателями на сегменты. Какой сегментный регистр используется для формирования физического адреса в текущем цикле шины, определяют сигналы состояния SA4–SA3:

– 00 — ES, дополнительный сегмент данных;

– 01 — SS, сегмент стека;

– 10 — CS, сегмент команд;

– 11 — DS, сегмент данных.

Чтобы программа была правильно скомпилирована, необходимо указать соответствие между сегментными регистрами и сегментами:

```
assume cs:code, ds:indat, es:outdat
```

а затем их инициализировать:

```
mov dx, indat
mov ds, dx
```

В каждой команде используемые сегментные регистры предполагаются по умолчанию. Например, как уже было сказано, для реги-

стров-указателей это соответствие таково: SI — DS, DI — ES; BP и SP — SS.

Указатель команд IP содержит смещение следующего байта команды от начала текущего сегмента кода, который будет выбирать устройство шинного интерфейса в очередном цикле шины. IP всегда сегментируется по CS.

Если нужно изменить соответствие сегментных регистров по умолчанию, используются специальные элементы команды — префиксы замены сегмента.

Из-за наличия сегментирования в процессоре i8086 существует понятие логического адреса. Логический адрес — это пара 16-битных чисел сегмент:смещение (база:смещение). Например, логический адрес очередного байта команды представляет собой значение пары CS:IP.

В логическом адресе смещение — это исполнительный или эффективный адрес (EA, Effective Address), «окончательное» смещение. Как уже говорилось, оно может формироваться достаточно сложным образом с использованием различных способов адресации.

Адрес, который процессор выдает по шине адреса, называется физическим. Его размер у i8086 равен двадцати битам. Адрес начала сегмента — это база, умноженная на 16. Таким образом, адрес начала сегмента всегда кратен шестнадцати. Благодаря этому выделяется такая единица памяти, как параграф. Параграф — область памяти размером в 16 байт, начинающаяся с адреса, кратного шестнадцати. Поэтому начало сегмента соответствует границе параграфа.

Поскольку смещение имеет разрядность 16 бит, то максимальный размер сегмента: 2^{16} = 64 Кбайта.

Соответствие между физическими и логическими адресами не является взаимно-однозначным. Одному значению логического адреса соответствует одно значение физического адреса. Но одному значению физического адреса может соответствовать множество значений логического адреса. Кроме того, существует множество значений логического адреса, которое дает значение физического адреса, пре-

вышающее максимально допустимое для i8086 — FFFFFh. В таких случаях по шине адреса выдаются двадцать младших битов получившегося значения. Например, логическому адресу FFFFh:0010h будет соответствовать физический адрес 00000h.

У процессора i80286 и последующих процессоров семейства x86 разрядность физической шины адреса больше 20 бит. В них также появился защищенный режим, а обычный режим, режим процессора i8086, стал называться реальным. После включения питания компьютера процессор оказывается в реальном режиме. При работе в реальном режиме любой процессор семейства x86 выглядит как i8086, и в нем используются только младшие двадцать адресных линий, A19–A0, остальные адресные линии блокируются. Однако есть возможность снять блокировку с адресной линии A20. Для этого используется специальный сигнал A20M# (см. далее). При этом в реальном режиме появляется еще один сегмент памяти, лежащий выше границы первого мегабайта. Этот сегмент называется областью верхней памяти (High Memory Area, HMA). Ему соответствует диапазон логических адресов от FFFFh:0010h до FFFFh:FFFFh. Размер HMA составляет 64 килобайта минус 16 байт. Операционная система MS-DOS может использовать область верхней памяти, располагая там свое ядро. Для этого необходимо подключить драйвер HIMEM.SYS и поместить в файл CONFIG.SYS строку:

```
DOS = HIGH
```

Наличие сегментирования накладывает дополнительные детали на использование подпрограмм. Подпрограмма может находиться в одном сегменте с программой, из которой она вызывается. Такая подпрограмма называется ближней. При переходе к ней в стек помещается только значение регистра IP. Если подпрограмма находится в другом сегменте, она называется дальней, и при ее вызове в стек помещается полный логический адрес возврата — CS:IP.

Регистр флагов (Flags) i8086, как и другие регистры, является 16-разрядным. В старшем байте появилось четыре новых флага: один признака результата и три управляющих:

– OF (Overflow Flag, бит 11) — флаг переполнения, устанавливается, если результат операции со знаковыми числами выходит за пределы разрядной сетки, т. е. за пределы допустимого диапазона;

– DF (Direction Flag, бит 10) — флаг направления, определяет порядок сканирования строк (0 — от меньших адресов к большим (регистры DI/SI увеличиваются), 1 — от больших к меньшим (регистры DI/SI уменьшаются));

– IF (Interrupt Flag, бит 9) — флаг прерывания, определяет реакцию процессора на запросы внешних прерываний по входу INT (0 — прерывания запрещены, 1 — разрешены);

– TF (Trace Flag, бит 8) — флаг трассировки, переводит процессор в пошаговый (покомандный) режим работы, в котором процессор генерирует внутреннее прерывание INT1 (исключение #DB, Debug) после выполнения каждой команды, что используется в отладчиках.

Младший байт регистра Flags — это в точности регистр флагов i8080. Только в документации они обозначаются несколько иначе — SF, ZF, AF, PF и CF. Однако, надо заметить, что флаг PF отражает четность числа единиц только в младшем байте результата. Остальные флаги являются признаками всего результата, который может быть байтом, словом или двойным словом (в командах умножения).

Тип данных процессора i8086 — целые двоичные числа, представленные в дополнительном коде. Вообще говоря, это имеет значение только для арифметических команд процессора. Другим командам, например, командам передачи данных, все равно, что это за последовательность нулей и единиц, содержащаяся в операнде. Как они интерпретируются: беззнаковые целые, знаковые целые, дробные с фиксированной точкой, символы, логические значения, числа в каком-либо коде — определяется алгоритмом программы, который, в свою очередь, определяется программистом. Процессор же рассматривает операнды как целые в дополнительном коде и только при

выполнении арифметических команд. Дополнительный код удобен тем, что позволяет использовать одни и те же арифметические команды для выполнения операций и над знаковыми, и над беззнаковыми числами, поскольку если знаковые целые представлены в дополнительном коде, то складывать и вычитать их можно по алгоритмам для беззнаковых целых. Более того, хотя имеются отдельные команды для сложения и вычитания, вычитание процессор осуществляет по тому же алгоритму, что и сложение. Он прибавляет к уменьшаемому число, равное вычитаемому со знаком минус, и инвертирует перенос для правильного формирования флага переноса (см. п. 3.2).

Так как материальные носители информации в вычислительных системах, регистры или ячейки памяти, имеют ограниченный размер, то сложение и вычитание производится по так называемому модулю 2^k, где k — размер регистров, в которых содержатся операнды (величина разрядной сетки):

$$sum(x,y) = (x+y)\,mod\ 2^k = \begin{cases} x+y & , \text{ если } x+y < 2^k.\ CF = 0. \\ x+y-2^k, & \text{если } x+y \geq 2^k.\ CF = 1. \end{cases}$$

$$difference(x,y) = (x-y)\,mod\ 2^k = \begin{cases} x-y & , \text{ если } x \geq y.\ CF = 0. \\ 2^k + x - y, & \text{если } x < y.\ CF = 1. \end{cases}$$

Здесь CF — флаг переноса.

Единичка, выходящая при сложении за пределы разрядной сетки, оказывается во флаге переноса. Потеря этой единички в регистре соответствует вычитанию 2^k. При вычитании единичка в этом флаге свидетельствует о заеме в старший разряд уменьшаемого, что соответствует прибавлению 2^k. Поскольку процессор вычитает числа по алгоритму сложения, то заем — это инвертированный перенос. То есть команда вычитания, работающая по алгоритму сложения, записывает во флаг переноса значение, полученное инвертированием реального переноса из старшего разряда. Так, команды сложения числа 3 с числом (−8) и вычитания из числа 3 числа 8 дают один и тот же результат, что совершенно естественно, однако их влияние на регистр

флагов различается. Команда сложения дает CF = 0, а команда вычитания — CF = 1. Влияние на флаг вспомогательного переноса в процессорах i8080 и i8086 различно. В процессоре i8086 флаг AF инвертируется так же, как и CF.

Для знаковых целых в дополнительном коде старший разряд имеет смысл знака числа. Оставшиеся разряды — это модуль (цифровая часть). Их число на единицу меньше общего числа разрядов, т. е. равняется k–1, однако все равно арифметические операции над ними осуществляются по модулю 2^k (используются те же команды, те же алгоритмы). При этом старшие цифры, представляющие собой знаки, складываются как обычные цифры. Фактически знаковые числа складываются как беззнаковые. Только исходные данные и результаты интерпретируются как знаковые. Однако, выигрывая в одном, проигрываем в другом. Использование тех же алгоритмов имеет следующий побочный эффект. Рассмотрим сложение знаковых чисел +127 и +2 при условии, что k = 8. Складывая их как беззнаковые, получим 129. Интерпретируя 129 как число со знаком, получим –127. А это является неправильным результатом. 129 не умещается в семи разрядах, отведенных под модуль. Получилось, что модуль «залез» в знаковый разряд. Истинный знак при этом был потерян. Если бы был реализован отдельный алгоритм для знаковых чисел, то сложение и вычитание в нем осуществлялось бы по модулю 2^k–1. Накрывание знака мантиссой не происходило бы, однако пришлось бы отдельно оперировать со знаками. Поэтому разработчиками i8086 был выбран другой способ решения этой проблемы. Был введен новый флаг OF, флаг переполнения, который фиксирует эту ситуацию. Переполнение происходит при сложении чисел одного знака или, что то же самое, при вычитании чисел с разными знаками, например (+127)–(–2), когда результат оказывается вне диапазона представимых знаковых чисел.

Для k = 8 диапазон знаковых чисел: –128 ÷ +127:

–128 10000000

–127 10000001

…

–2	11111110
–1	11111111
0	00000000
1	00000001
…	
127	01111111

Для выполнения операций умножения и деления знаковых и беззнаковых чисел в процессорах x86 существуют отдельные команды.

Процессоры архитектуры IA-32 и Intel 64

Основной технико-эксплуатационный показатель компьютера — производительность (см. п. 2.1). Развитие процессоров идет в направлении роста производительности при ограничении на потребляемую мощность. В процессе этого развития увеличивается разрядность, растет тактовая частота, расширяются возможности отладки, мониторинга производительности и функционирования, управления питанием, построения многопроцессорных систем, синхронизации работы различных устройств ЭВМ, совершенствуется система команд, появляются новые архитектурные особенности и элементы: внутренняя кэш-память, встроенный математический сопроцессор (FPU), контроллер прерываний (APIC, Advanced Programmable Interrupt Controller), контроллер памяти и т. д.

Важным показателем, влияющим на производительность, является разрядность процессора. Эта величина кратна степени двойки, поэтому растет дискретно поэтапно, но при этом практически линейно (рис. 3.11):

- 4 бита — i4004, 1971 г.;
- 8 бит — i8008, 1972 г.;
- 16 бит — i8086, 1978 г.;
- 32 бита — i80386, 1985 г.;
- 64 бита — Pentium 4 Extreme Edition, 2005 г.

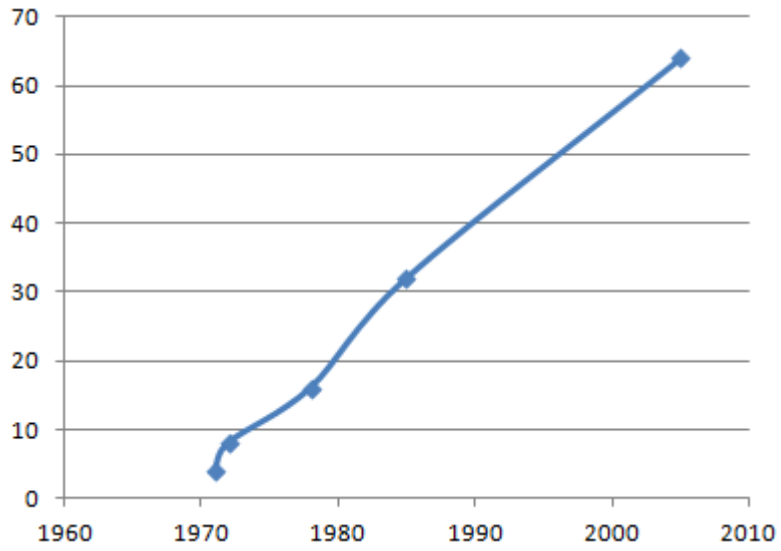


Рис. 3.11. Рост разрядности процесссоров Intel по годам

У других компаний-производителей сроки примерно такие же. Все здесь идет более-менее синхронно. 128 бит будет в 2045 г. Это, конечно, шутка. Разрядность процессора — комплексное свойство. Оно включает множество разрядностей: внешних и внутренних шин, регистров общего назначения, других регистров, операндов команд, адресов операндов в командах и регистрах. Разрядность регистров FPU с самого начала — 80 бит. В современных процессорах x86 есть регистры размером 128, 256 и 512 бит. Но главная разрядность процессора — разрядность регистров общего назначения. Именно она подразумевается под разрядностью процессора. В семействе x86 выделяются подсемейства 32-разрядных процессоров — IA-32 (Intel Architecture 32-bit) — и 64-разрядных — Intel 64. Наиболее характерные представители подсемейства IA-32 — процессоры Pentium — Pentium III, Intel 64 — Pentium 4, Core. В семействе x86 имеется линейка процессоров Xeon. Они были разработаны компанией Intel для использования в серверах и высокопроизводительных рабочих станциях. Процессоры Xeon имеют более продвинутые характеристики и свойства, в том числе средства поддержки создания многопроцессорных систем на их основе. Первые процессоры Xeon имеют архитектуру IA-32, последние, как и все современные процессоры, — Intel 64.

В процессе развития процессоров семейства x86 увеличивалось и количество выводов. Так, например, процессор Pentium имеет 296 выводов: 106 — питание, 177 — сигналы, которые распределяются по 29 группам (табл. 3.4).

Таблица 3.4

Группы сигналов процессора Pentium

Функция	Сигналы
Синхронизация (Clock)	CLK
Инициализация (Initialization)	RESET, INIT, BF1-BF0
Шина адреса (Address Bus)	A31–A3, BE7#-BE0#
Маска адреса (Address Mask)	A20M#
Шина данных (Data Bus)	D63–D0
Четность адреса (Address Parity)	AP, APCHK#
Поддержка APIC (APIC Support)	PICCLK, PICD0-1
Четность данных (Data Parity)	DP7–DP0, PCHK#, PEN#
Внутренняя ошибка четности (Internal Parity Error)	IERR#
Системная ошибка (System Error)	BUSCHK#
Определение цикла шины (Bus Cycle Definition)	M/IO#, D/C#, W/R#, CACHE#, SCYC, LOCK#
Управление шиной (Bus Control)	ADS#, ADSC#, BRDY#, BRDYC#, NA#
Кэширование страниц (Page Cacheability)	PCD, PWT
Управление кэшированием (Cache Control)	KEN#, WB/WT#
Обращение к кэшу / сохранение кэша (Cache Snooping/Consistency)	AHOLD, EADS#, HIT#, HITM#, INV
Обновление кэша (Cache Flush)	FLUSH#
Упорядочение записи (Write Ordering)	EWBE#
Арбитраж шины (Bus Arbitration)	BOFF#, BREQ, HOLD, HLDA
Частное управление шиной в двухпроцессорных системах (Dual Processing Private Bus Control)	PBGNT#, PBREQ#, PHIT#, PHITM#

Функция	Сигналы
Прерывания (Interrupts)	INTR, NMI
Сообщения об ошибках операций с плавающей точкой (Floating Point Error Reporting)	FERR#, IGNNE#
Режим управления системой (System Management Mode)	SMI#, SMIACT#
Проверка функционального резерва (Functional Redundancy Checking)	FRCMC#, (IERR#)
Порт доступа к тестированию (Test Access Port, TAP)	TCK, TMS, TDI, TDO, TRST#
Мониторинг контрольных точек и производительности (Breakpoint/Performance Monitoring)	PM0/BP0, PM1/BP1, BP3-2
Управление питанием (Power Management)	STPCLK#
Вспомогательные функции для двухпроцессорных систем (Miscellaneous Dual Processing)	CPUTYP, D/P#
Отладка (Debugging)	R/S#, PRDY
Детектирование напряжения (Voltage Detection)	VCC2DET#

Pentium относится к подсемейству IA-32, т. е. является 32-разрядным процессором, но шина данных у него 64-разрядная. Поэтому в шине адреса отсутствуют сигналы A2–A0. 64 бита — это 8 байт. $8 = 2^3$ — для адресации 8-байтных порций данных не нужны три младших адресных сигнала, они адресуют более мелкие информационные единицы. Однако возможность побайтной адресации памяти сохраняется, для этого используются сигналы BE7#–BE0# (Byte Enable), которые определяют размер передаваемых данных и также относятся к группе адресных сигналов. Так что в системной шине адреса компьютера сигналы A2–A0 имеются. Они формируются контроллером памяти на основе сигналов BE7#–BE0# (рис. 3.12).

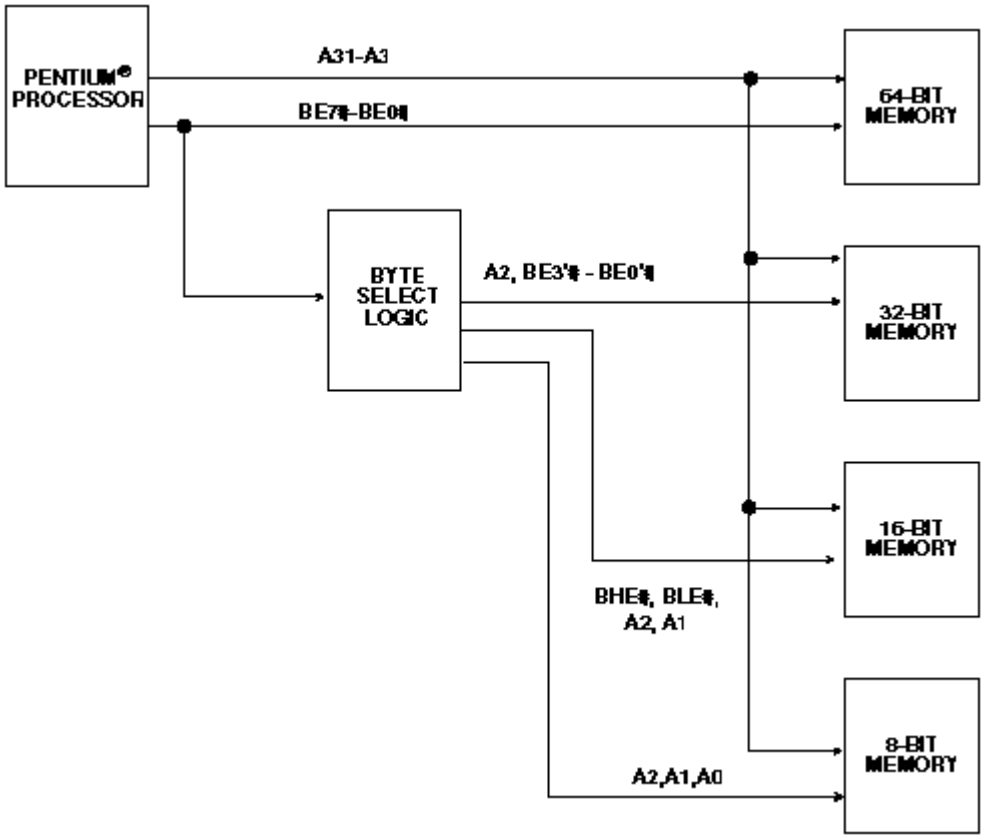


Рис. 3.12. Адресация памяти процессором Pentium

Для повышения надежности обмена данными имеется аппаратный контроль четности адреса и данных.

В группе сигналов прерываний отсутствует знакомый сигнал INTA. Теперь он является внутренним, поступает от ядра процессора внутреннему контроллеру прерываний APIC.

От модели к модели у процессоров растет тактовая частота, а, следовательно, и рассеиваемая мощность. Поэтому появляется необходимость уменьшения напряжения питания и обеспечения источником питания требуемой мощности.

У процессоров Pentium II, Pentium III появляются сигналы VID (Voltage Identification, идентификация напряжения), которые автоматически программируют регулятор напряжения на требуемый уровень. Выводы питания VCC должны обеспечивать процессор напряжением, определяемым сигналами VID.

Pentium 4 в корпусе FCPGA-478 имеет 111 выводов питания VCC, 112 выводов VSS (земля) и 5 сигналов идентификации напря-

жения VID[4:0]. Всего возможны 32 комбинации. Первые 31 из них задают определенный уровень напряжения: от 1,850 В при VID[4:0] = 00000 до 1,100 В при VID[4:0] = 11110 с шагом в 0,025 В. Последняя комбинация, VID[4:0] = 11111, отключает регулятор напряжения.

Другие важные моменты:

− множество сигналов во многом расширено сигналами, обеспечивающими надежность функционирования компьютера, а также экономию питания;

− множество сигналов становится все более абстрактным, например, отсутствуют знакомые сигналы приема-передачи, а тип транзакции определяют сигналы адресной шины и сигналы REQ[4:0]#;

− сигналы адресной шины являются двунаправленными и многофункциональными.

Адресные сигналы процессора Pentium 4 — A[35:3]#. Соответственно, размер адресного пространства процессора — 2^{36} = 16 Гбайт. В первой подфазе адресной фазы по этим выводам передается адрес транзакции, во второй — информация о типе транзакции. По заднему фронту сигнала RESET#, т. е. при переходе его от активного уровня к неактивному, процессор проверяет подмножество этих сигналов для определения конфигурации по включению питания (power-on configuration).

Появились интересные сигналы DBI[3:0]# (Data Bus Inversion, инвертирование шины данных). Они определяют полярность сигналов D[63:0]#. DBI[3:0]# активизируются, когда данные на шине данных инвертированы. Агент шины инвертирует сигналы шины данных, если больше чем половина битов в соответствующей группе изменит свое значение в следующем цикле. Таким образом, реально изменит свое значение меньше половины битов. Это снижает энергопотребление.

В процессорах Intel Core 2 в корпусе LGA-775 сигналы контроля четности адреса и данных отсутствуют, поскольку процессор и так

работает достаточно надежно, но добавилось множество сигналов для регулирования напряжения питания.

В дальнейшем после появления в составе процессора контроллера памяти и других устройств на выводах уже присутствуют не только сигналы собственно процессора, но и системные сигналы.

Основные особенности архитектуры процессоров IA-32 и Intel 64 удобно рассмотреть, начиная с процессора Pentium (в том числе с технологией MMX) (рис. 3.13) [56].

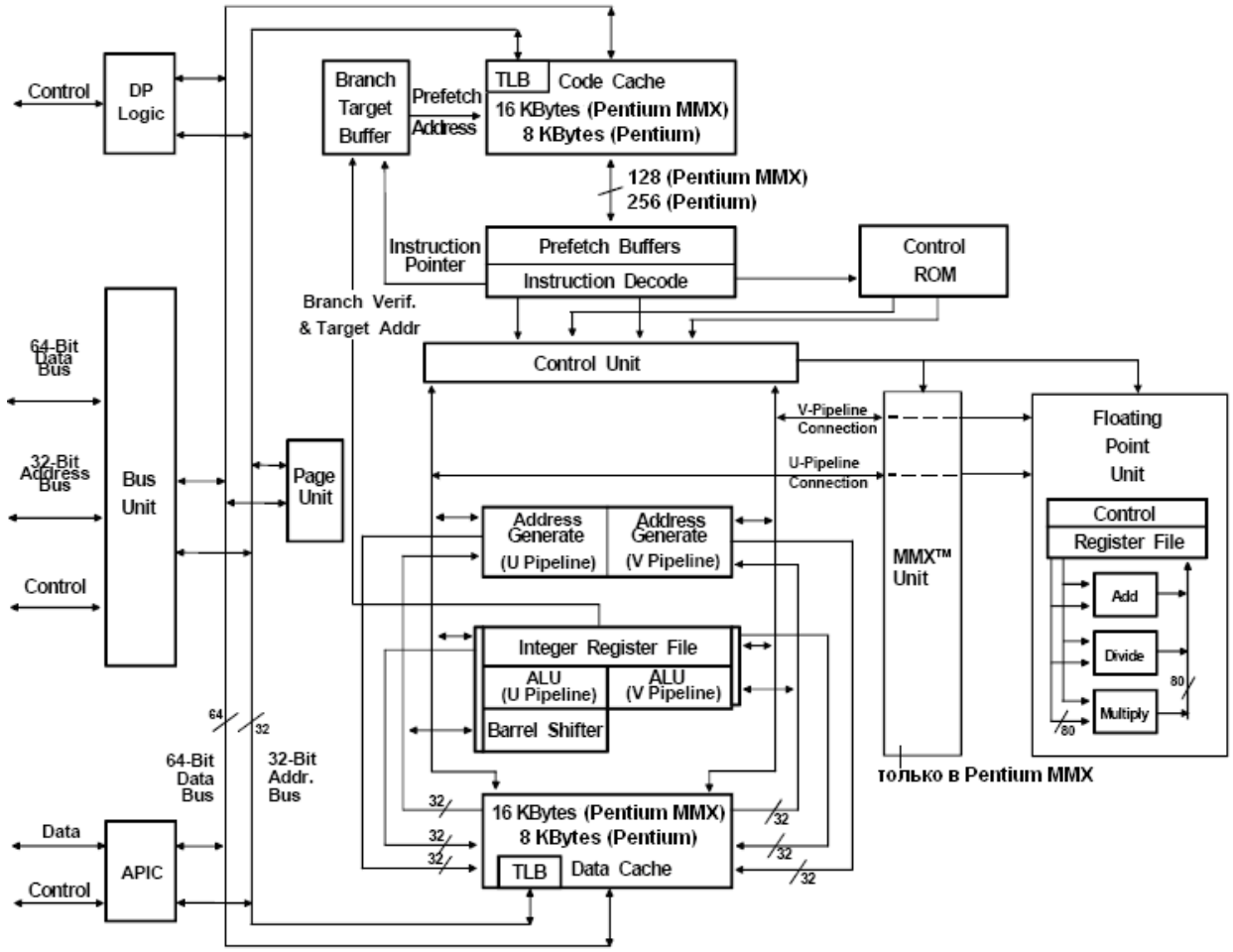


Рис. 3.13. Структурная схема процессоров Pentium и Pentium MMX

В дальнейшем в документации Intel уже нет подобных картинок, поскольку в основном все эти архитектурные элементы сохраняются. Развитие происходит на уровне более мелких деталей, которые в целом называются микроархитектурой.

Процессор Pentium поддерживает все особенности архитектуры предшествующих процессоров Intel и имеет новые важные характеристики, в число которых входят:

– суперскалярность (наличие двух конвейеров);

– конвейеризованное устройство с плавающей точкой (Pipelined FPU);

– раздельная кэш-память первого уровня для команд и данных (Code Cache, Data Cache), 8 Кбайт каждая;

– динамическое предсказание переходов (Dynamic Branch Prediction);

– отслеживание исполнения команд (Execution Tracing);

– 64-разрядная шина данных с поддержкой пакетных циклов чтения-записи и конвейеризацией цикла шины, позволяющей выполняться одновременно двум циклам;

– контроль по четности адреса и внутренний контроль по четности;

– проверка на основе функциональной избыточности (Functional Redundancy Checking) (не поддерживается Pentium MMX);

– контроль производительности (Performance Monitoring);

– тестирование и отладка в соответствии со стандартом IEEE 1149.1 (IEEE 1149.1 Boundary Scan);

– режим управления системой (System Management Mode);

– поддержка двухпроцессорной системы (Dual Processing Support);

– команды сериализации для поддержки многопроцессорности;

– встроенный расширенный контроллер прерываний (APIC, Advanced Programmable Interrupt Controller).

Процессор Pentium MMX имеет следующие дополнительные характеристики:

– технология MMX;

– четыре буфера предвыборки;

− кэш-память команд и данных по 16 Кбайт каждая;

− дополнительная стадия конвейера — стадия выборки (Fetch) между стадиями предвыборки (Prefetch) и декодирования команды.

Устройство MMX тесно связано с FPU: оно использует его регистры, но реализует совершенно другой тип вычислений. По сути, оно является векторным процессором, который по классификации Флинна относится к классу SIMD (Single Instruction, Multiple Data — одиночная команда, множественные данные) (см. п. 5.1). Состав команд Pentium MMX включает четыре новых типа данных и 57 новых команд для увеличения производительности мультимедийных и коммуникационных программ.

Слева на рис. 3.13 мы видим внешние связи процессора в виде уже знакомых шин адреса, данных и управления, которые реализует устройство шинного интерфейса (Bus Unit). В дополнение к нему на этом внешнем уровне присутствуют средства реализации многопроцессорных вычислительных систем:

− DP Logic (Dual Processing Logic) — устройство обеспечения двухпроцессорной обработки, при которой два процессора работают с одинаковым уровнем приоритета (симметрично), разделяя кэш второго уровня, который реализуется в виде отдельных микросхем, расположенных на плате между процессором и основной памятью;

− расширенный контроллер прерываний APIC.

APIC является важнейшим звеном организации работы многопроцессорной компьютерной системы. Помимо обработки внутренних и внешних прерываний от различных устройств он принимает и посылает межпроцессорные прерывания (InterProcessor Interrupt, IPI). Именно APIC содержит в одном из своих регистров идентификатор процессора, в состав которого он входит. APIC совместим с контроллером прерываний i8259A, но имеет большую функциональность.

Помимо сегментации памяти в IA-32 и Intel 64 присутствует еще один способ ее структурирования — страничная адресация. На рисунке это отражено в виде блока Page Unit.

Центральная часть процессора — два конвейера команд (U-pipeline и V-pipeline), два арифметико-логических устройства (АЛУ) с общим набором регистров (Integer Register File), FPU и устройство MMX. Каждый конвейер при отсутствии зависимостей между командами выполняет команду за один такт. Процессор в целом способен выполнять за один такт две целочисленные команды и одну команду с плавающей точкой (при некоторых условиях — две). Таким образом, его пиковая производительность при частоте 133 МГц равна 133 MFLOPS при выполнении вычислений с плавающей точкой и 266 MIPS при выполнении целочисленных вычислений.

U-конвейер может выполнять все команды целочисленной арифметики и арифметики с плавающей точкой. V-конвейер — простые команды целочисленной арифметики и команду арифметики с плавающей точкой FXCH.

Конвейер процессора Pentium имеет пять этапов (стадий), Pentium MMX — шесть:

- PF (Prefetch) — предвыборка;
- F (Fetch) — выборка (только для Pentium MMX);
- D1 (Instruction Decode) — декодирование команды;
- D2 (Address Generate) — генерация адреса;
- EX (Execute) — исполнение;
- WB (Writeback) — обратная запись.

Для поддержки предсказания переходов в процессоре имеется буфер BTB (Branch Target Buffer, буфер адресов переходов или буфер предсказания переходов) и два 32-байтных буфера предвыборки (Prefetch Buffers). Один осуществляет предвыборку в линейном режиме, другой — в соответствии с буфером BTB. Таким образом, почти всегда нужный код предварительно выбран перед тем, как возникает необходимость его выполнения. В Pentium MMX четыре 16-байтных буфера предвыборки и улучшенный алгоритм предсказания переходов.

Процессоры Pentium и Pentium MMX имеют раздельный кэш первого уровня (L1) — кэш данных и кэш кода, что является призна-

ком гарвардской архитектуры. Они могут быть разрешены или запрещены аппаратно или программно. Кэш L1 процессора Pentium — двухпроходной частично ассоциативный (2-way set-associative), размером по 8 килобайт для команд и данных, Pentium MMX — четырехпроходной (4-way set-associative), размером по 16 килобайт для команд и данных. Каждый кэш имеет буфер трансляции адресов (TLB, Translation Lookaside Buffer) для перевода линейных адресов в физические.

Кэш данных соответствует протоколу MESI (Modified, Exclusive, Shared, Invalid). Он может быть сконфигурирован как кэш с обратной записью (writeback) или со сквозной записью (writethrough) на построчной основе. Кэш данных имеет два порта — по одному на каждый конвейер. Теги кэша данных имеют три порта для поддержки передачи данных двум конвейерам и одновременно цикла запроса в одном такте.

Режим кэширования определяется битами CD (Cache Disable) и NW (Not Writethrough) управляющего регистра CR0.

Кэш кода, буфер BTB и буферы предвыборки вместе отвечают за доставку последовательности команд исполнительным устройствам. Ширина внутренней шины команд между кэшем кода и буферами предвыборки у Pentium — 256 бит, Pentium MMX — 128 бит.

Команды выбираются из кэша кода или поступают с внешней шины. Адреса переходов запоминаются в BTB. Устройство декодирования содержит два параллельных декодера, которые декодируют и посылают команды в конвейеры.

Управляющее ПЗУ (Control ROM) содержит микрокод, контролирующий последовательность операций, выполняемых процессором.

Управляющее устройство (Control Unit) осуществляет прямое управление обоими конвейерами.

В процессоре Pentium существенно расширены средства повышения надежности — проверки целостности данных и обнаружения

ошибок. Проверка четности данных поддерживается на уровне байтов. Добавлены контроль по четности адреса и внутренний контроль по четности наряду с новым исключением — исключением машинного контроля (machine check exception) (см. п. 4.1). Для повышения надежности обнаружения ошибок используется контроль на основе функциональной избыточности (functional redundancy checking). В двухпроцессорной системе второй процессор, «контролер», синхронно (в режиме Lock Step) осуществляет выборку значений с выходов главного процессора и сравнивает их со значениями, рассчитанными им самим. В случае несоответствия он вырабатывает сигнал ошибки. Pentium MMX контроль функциональной избыточности не поддерживает.

Поскольку количество функций, выполняемых процессором, растет, то растет и сложность тестирования. Pentium повышает возможности тестирования и отладки, реализуя IEEE Boundary Scan (Standard 1149.1).

В последующих процессорах эта архитектурная база сохраняется, развитие идет в направлении совершенствования устройства и работы ядра процессора — конвейерных обрабатывающих блоков, что в совокупности входит в понятие микроархитектуры. Последовательность появления новых микроархитектур такова:

− P6 — 1995 г., Pentium Pro;

− NetBurst — 2000 г., Pentium 4;

− Core — 2006 г., Xeon 5100;

− Core с различными кодовыми наименованиями: Nehalem (2008 г., Core i7) и т. д.

Процессоры семейства IA-32 Pentium Pro — Pentium III, в том числе Pentium II Xeon и Pentium III Xeon, имеют в своей основе микроархитектуру P6 (рис. 3.14).

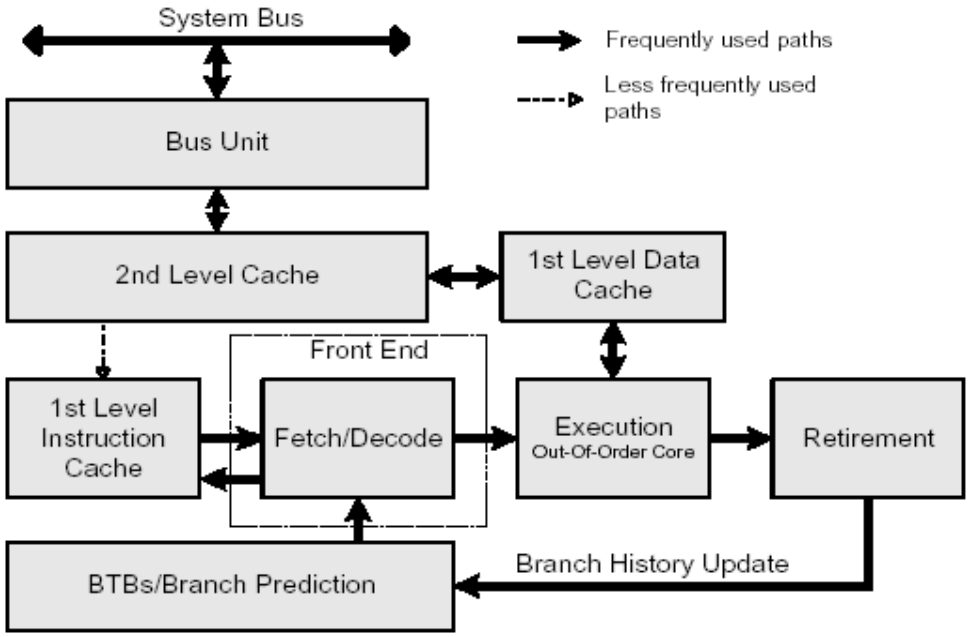


Рис. 3.14. Микроархитектура P6

Основные особенности микроархитектуры P6:

− динамическое исполнение (переупорядочивание команд, out-of-order):

− предсказание переходов;

− динамический анализ потока данных через процессор для определения зависимостей и выяснения целесообразности внеочередного выполнения команд;

− выполнение команд с опережением;

− двухуровневый кэш: раздельный L1 — по 8 Кбайт для команд и данных; общий L2 — 256 Кбайт, 512 Кбайт или 1 Мбайт;

− 12-стадийный конвейер.

На стадии завершения (Retirement) происходит сохранение результатов и обновление состояния. Здесь результаты выполненных микроопераций обрабатываются так, что состояние процессора обновляется в соответствии с первоначальным порядком программы.

В Pentium III появилось расширение технологии MMX — SSE (Streaming SIMD Extensions) — отдельные 128-битные регистры XMM.

В процессоре Pentium 4 была введена новая микроархитектура Intel NetBurst (рис. 3.15).

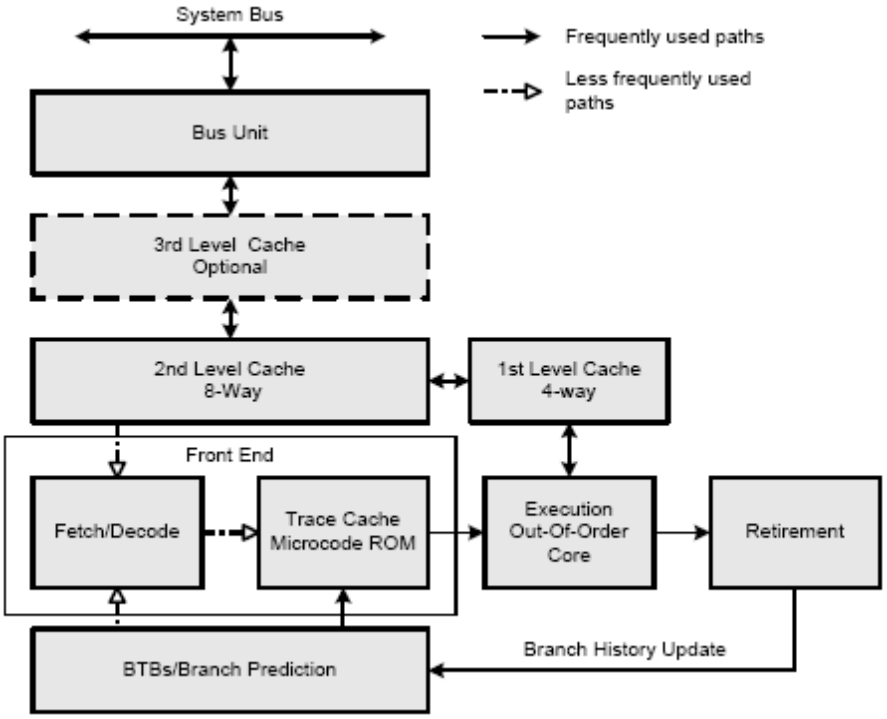


Рис. 3.15. Микроархитектура NetBurst

Позже появились и процессоры Xeon с этой микроархитектурой.

Основные черты микроархитектуры NetBurst:

– 20-стадийный гиперконвейер;

– системная шина с учетверенной накачкой (Quad Pumped Bus) с пропускной способностью 3,2 ГБ/с;

– кэш второго уровня с улучшенной передачей данных (Advanced Transfer Cache, ATC);

– кэш команд первого уровня (Trace Cache) хранит микрокоманды;

– блок быстрого исполнения команд (Rapid Execution Engine) работает с удвоенной тактовой частотой;

– улучшенная технология динамического исполнения;

– расширение SSE2;

– улучшенные средства тестирования и мониторинга.

Системная шина Pentium 4 использует схему передачи сигналов с буферизацией на физическом уровне, так что за один такт данные передаются четыре раза, и соответственно, по 100-МГц шине обеспечивается передача данных с частотой 400 МГц.

Максимальная производительность процессора Pentium 4 достигается при работе с памятью типа RDRAM (Rambus DRAM). Именно этот тип памяти поддерживал первый чипсет для Pentium 4 — i850. Однако из-за ее дороговизны вскоре были выпущены чипсеты, обеспечивающие работу Pentium 4 с памятью SDRAM, а впоследствии — DDR.

Главной особенностью микроархитектуры NetBurst является то, что кэш-память команд первого уровня хранит до 12 000 декодированных микроопераций в порядке их выполнения. Это увеличивает производительность за счет исключения декодера из основного исполнительного устройства и делает более эффективным использование кэш-памяти за счет уменьшения количества обменов данными между ней и основной памятью из-за наличия команд ветвления и возможности неправильного предсказания переходов. В результате удается передать больший объем команд в исполнительные блоки процессора и уменьшить общее время, требуемое на возврат из тех ветвлений, которые были неправильно предсказаны.

Блок быстрого исполнения команд (Rapid Execution Engine) включает два АЛУ, которые работают с частотой, превышающей тактовую частоту процессора в два раза. Это позволяет выполнять основные целочисленные команды с двойной тактовой частотой. Например, блок быстрого исполнения команд процессора Pentium 4 с тактовой частотой 1,5 ГГц работает с частотой 3 ГГц.

Кэш-память L2 с улучшенной передачей данных (Advanced Transfer Cache, ATC) работает на частоте ядра процессора и имеет канал шириной 256 бит (32 байта). Таким образом, его пропускная способность при частоте 1,5 ГГц составляет 48 ГБ/с.

Улучшенная технология динамического исполнения состоит в том, что процессор может просматривать на лету 126 команд для определения целесообразности их переупорядочивания, осуществлять до 48 загрузок из конвейера и 24 размещений в конвейере. Он также имеет усовершенствованный алгоритм предсказания переходов, обеспечивающий уменьшение числа ложных предсказаний при-

мерно на 33 % по сравнению с аналогичным показателем процессоров с микроархитектурой P6.

Расширение SSE2 — это 144 новые команды, включающие 128-разрядные целочисленные арифметические операции и операции с плавающей точкой. Они ускоряют выполнение программ обработки видео, звука, изображений, шифрования, а также различных научно-технических расчетов.

В Pentium 4 Prescott (2004 г.) появилось расширение SSE3, включающее 13 новых команд.

Средства тестирования и мониторинга системы:

− встроенный алгоритм самотестирования (Built-In Self Test, BIST), осуществляющий поиск ошибки в микрокоде и в больших логических матрицах, тестирование кэш-памяти команд, кэш-памяти данных, буферов ассоциативной трансляции (Translation Lookaside Buffers, TLB) и постоянной памяти (ROM);

− порт доступа к тестированию IEEE 1149.1 (Test Access Port, TAP);

− внутренние счетчики, контролирующие работу процессора и осуществляющие подсчет событий;

− мониторинг термического состояния, позволяющий получить реальную величину потребляемой энергии и конструировать более экономичные системные платы.

Именно в рамках микроархитектуры NetBurst в процессоре Pentium 4 Northwood появилась технология HyperThreading (Intel HT Technology) (см. п. 3.4).

Микроархитектура Intel Core (рис. 3.16) была введена в процессоре Xeon 5100, а затем было выпущено множество процессоров с этой микроархитектурой и соответствующим названием: Core 2 Duo, Core 2 Quad и т. д.

Основные черты микроархитектуры Intel Core:

− Wide Dynamic Execution (внеочередное исполнение микроопераций, до 4 команд за такт на ядро):

− 14-стадийный конвейер;

– 3 АЛУ;

– 4 декодера для декодирования до 5 команд за один такт;

– улучшенное предсказание переходов;

– Advanced Smart Cache L2 размером до 4 Мбайт и с 256-битной внутренней шиной;

– SSE4 (новые команды SIMD).

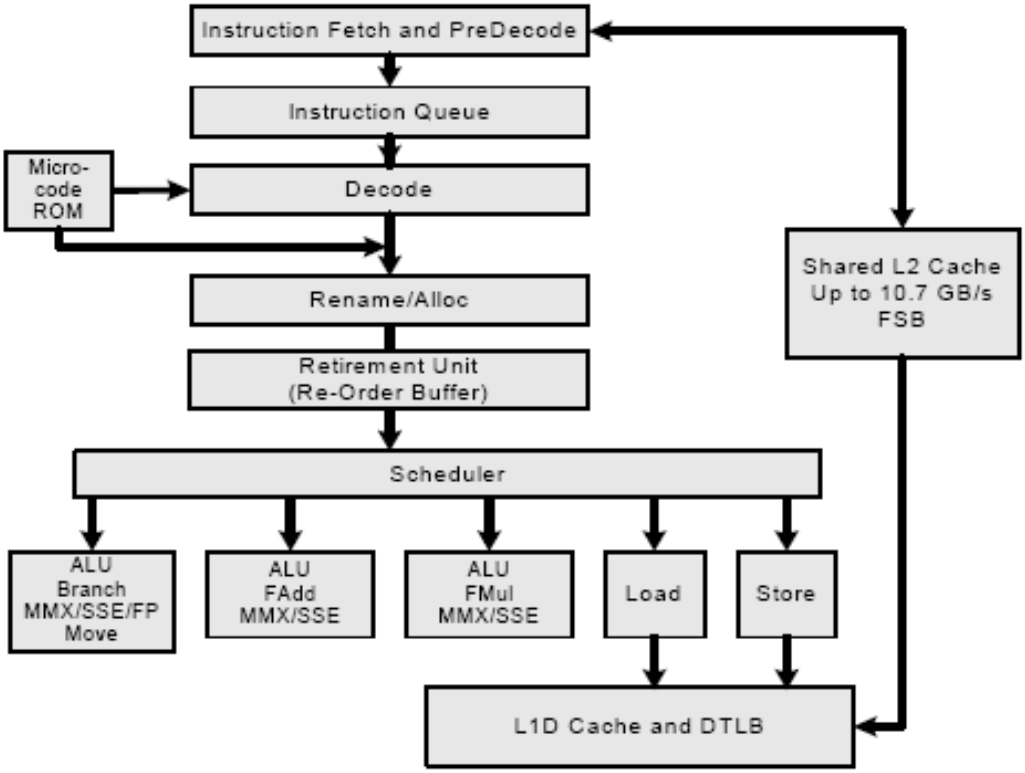


Рис. 3.16. Микроархитектура Intel Core

Вначале в ней отсутствовала технология Hyper-Threading. За счет этого, а также уменьшения количества стадий конвейера и снижения тактовой частоты энергопотребление процессоров с микроархитектурой Intel Core было существенно снижено. Последние процессоры Pentium 4 были уже слишком горячими. Но позже технология Hyper-Threading была добавлена в микроархитектуру Core в процессорах Core i.

С микроархитектурой Intel Core были выпущены первые многоядерные процессоры: Core 2 Duo, Core 2 Quad, Core 2 Extreme, Xeon и др. (рис. 3.17, 3.18).

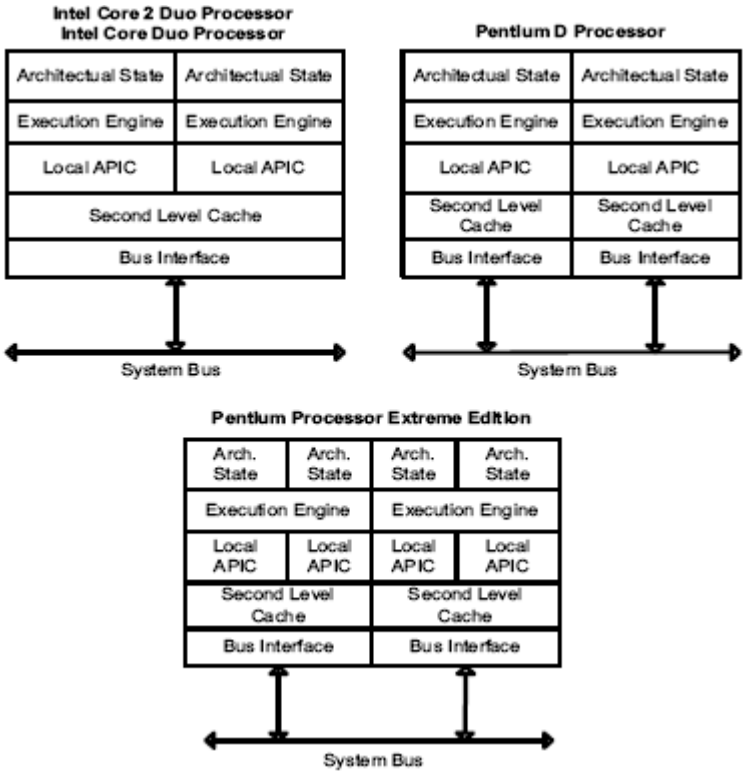


Рис. 3.17. Архитектура двухъядерных процессоров

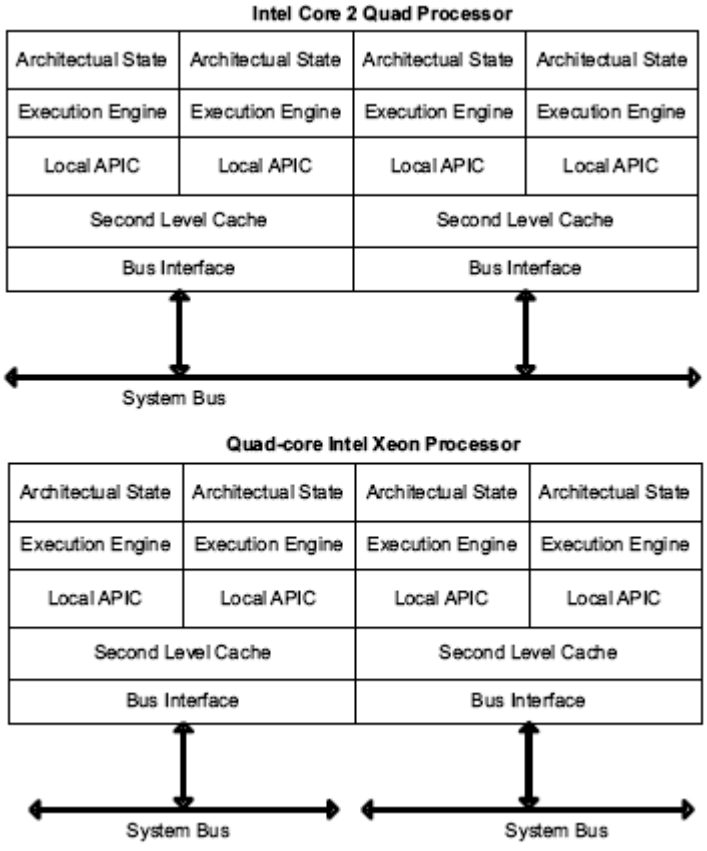


Рис. 3.18. Архитектура четырехъядерных процессоров

В современных процессорах x86 микроархитектура Core в целом остается, новых картинок не появляется, но происходит ее дальнейшее развитие в различных деталях. Каждое новое поколение процессоров имеет микроархитектуру под собственным кодовым наименованием — Nehalem, Westmere, Sandy Bridge и т. д.

С микроархитектурой Nehalem были выпущены первые процессоры Intel Core i: Core i7, Core i5, Core i3. Эти названия сохраняются до настоящего времени, меняются только поколения. В них вновь появилась технология Hyper-Threading.

Множество регистров в процессорах IA-32 и Intel 64 существенно расширено. С введением защищенного режима появилась принципиально новая группа регистров — системные регистры, предназначенные для переключения между режимами и организации работы процессора в защищенном режиме. Это регистры управления процессором, регистры управления памятью, отладочные регистры и машинно-специфичные регистры.

Множество обычных несистемных регистров также претерпело изменения. ***Несистемные регистры процессоров IA-32 таковы:***

- 8 32-битных регистров общего назначения: EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI;
- 6 16-битных сегментных регистров: ES, CS, SS, DS, FS, GS;
- 32-битный указатель команд EIP;
- 32-битный регистр флагов EFLAGS;
- регистры FPU:
  - 8 80-разрядных регистров данных;
  - 16-битный управляющий регистр;
  - 16-битный регистр состояния;
  - 16-битный регистр тегов;
  - 11-битный регистр кода операции;
  - 48-битный указатель команд;
  - 48-битный указатель данных (операндов).
  - 8 64-битных регистров MMX;
  - 8 128-битных регистров XMM;

– 32-битный регистр MXCSR (MMX Control and Status Register) — регистр управления и состояния устройства MMX.

Регистры общего назначения i8086 стали 32-битными, в их названии появилась буква E (extended). Их специализация понизилась (см. п. 3.3). Что касается разрядности сегментных регистров, то здесь появился дополнительный нюанс: различаются разрядность с точки зрения программы и внутренняя аппаратная разрядность. Их программная разрядность, 16 бит, не изменилась. Но теперь они имеют скрытую часть, которая в защищенном режиме содержит дескриптор соответствующего сегмента. Также увеличилось их число до шести.

Регистр флагов EFLAGS помимо признаков результата и управляющих флагов содержит новый тип флагов — системные флаги. Они связаны с защищенным режимом и многозадачностью.

После появления в составе процессора устройств FPU и MMX, добавились и их регистры. Регистры MMX физически представляют собой 64-битную часть регистров данных FPU. Регистры XMM появились в Pentium III в рамках расширения SSE. Это уже физически отдельные регистры.

В Intel 64 также произошло большое расширение множества регистров.

Регистры процессоров Intel 64:

– 16 64-битных регистров общего назначения: RAX, RCX, RDX, RBX, RSP, RBP, RSI, RDI, R8-R15;

– 6 16-битных сегментных регистров: ES, CS, SS, DS, FS, GS;

– 64-битный указатель команд RIP;

– 64-битный регистр флагов RFLAGS, старшие 32 бита зарезервированы, младшие равны EFLAGS;

– регистры FPU:

  – 8 80-разрядных регистров данных;

  – 16-битный управляющий регистр;

  – 16-битный регистр состояния;

  – 16-битный регистр тегов;

  – 11-битный регистр кода операции;

- 64-битный указатель команд;
- 64-битный указатель данных (операндов);
- 8 64-битных регистров MMX;
- 16 128-битных регистров XMM;
- 32-битный регистр MXCSR (MMX Control and Status Register) — регистр управления и состояния устройства MMX.

Системные регистры в Intel 64, конечно, также были расширены. Но основные изменения коснулись регистров общего назначения. Теперь их число — 16. Первые восемь расширены до 64 бит: RAX, RCX, RDX, RBX, RSP, RBP, RSI, RDI. Плюс еще восемь — R8-R15, без такой специализации, как у первых восьми. Еще один шаг в RISC-направлении.

Позже в некоторых моделях процессоров Intel 64 появились 256-битные регистры YMM (расширение AVX, Advanced Vector Extensions) и 512-битные регистры ZMM (расширение AVX-512).

В расширении MPX (Memory Protection Extensions), введенном в процессорах Xeon Scalable и Core i 6-го поколения и просуществовавшем в архитектуре Intel 64 до 2019 г., имеется группа регистров Bounds Registers, которые используются для проверки выхода значения индекса элемента массива за допустимые пределы:

- BND0–BND3: четыре 128-битных регистра, каждый из которых содержит пару 64-битных значений нижней (LB, Lower Bound) и верхней (UB, Upper Bound) границ буфера в памяти;
- BNDCFGU: 64-битный регистр конфигурации для пользовательского режима (CPL = 3);
- IA32_BNDCFGS: 64-битный регистр конфигурации для системного режима (supervisor mode, CPL < 3);
- BNDSTATUS: 64-битный регистр состояния.

Фундаментальные типы данных процессоров IA-32 и Intel 64 таковы:

- Byte (байт, 8 бит);
- Word (слово, 16 бит);
- Double Word (двойное слово, 32 бита);

– Quad Word (учетверенное слово, 64 бита);

– Double Quad Word (двойное учетверенное слово, 128 бит).

Большинство команд оперируют этими данными без какой-либо дополнительной их типизации. Учетверенное слово было введено в i486, двойное учетверенное — в Pentium III с расширением SSE. В памяти операнды хранятся побайтно. Младшему байту операнда соответствует младший адрес (little endian order). Он является адресом и всего операнда. Выравнивание операндов в памяти (расположение слов по четным адресам, двойных слов — по адресам, кратным четырем и т. д.) повышает эффективность программы, так как уменьшается необходимое число обращений процессора к памяти. При этом слова, расположенные по нечетным адресам, но не пересекающие 4-байтную границу, также считаются выравненными и доступны за одно обращение. Некоторые команды, оперирующие двойными учетверенными словами, генерируют исключение общей защиты (#GP, General Protection), если операнды не выравнены.

Кроме размера некоторые команды требуют дополнительных признаков классификации операндов. Арифметические команды оперируют числами, которые, в свою очередь, могут быть следующих типов:

– Byte Unsigned Integer (8-битное беззнаковое целое);

– Word Unsigned Integer (16-битное беззнаковое целое);

– Doubleword Unsigned Integer (32-битное беззнаковое целое);

– Quadword Unsigned Integer (64-битное беззнаковое целое);

– Byte Signed Integer (8-битное знаковое целое);

– Word Signed Integer (16-битное знаковое целое);

– Doubleword Signed Integer (32-битное знаковое целое);

– Quadword Signed Integer (64-битное знаковое целое);

– Single Precision Floating Point (32-битное вещественное (одинарной точности);

– Double Precision Floating Point (64-биттное вещественное (двойной точности));

– Double Extended Precision Floating Point (80-битное вещественное (расширенной точности)).

Кроме того, имеется огромное количество различных типов данных MMX и XMM.

3.3. СИСТЕМА КОМАНД ПРОЦЕССОРА

Программная модель процессора. Характеристики команды

Часто программной моделью процессора называют его программно-доступные регистры. Однако эта модель неполна. Наиболее полной программной моделью процессора является его система команд. Она включает в себя и программно-доступные регистры, и команды, в которых они могут быть задействованы.

Команда процессора имеет следующие характеристики:

- тип (группа);
- формат (в том числе формат кода операции);
- тип операндов;
- способ адресации;
- влияние на регистр флагов;
- количество циклов и тактов, требуемых для ее выполнения.

Во всех процессорах в соответствии с основными типами действий, которые необходимо выполнять в самых различных программах, имеются следующие группы команд (см. п. 1.3):

1. Передачи данных.
2. Передачи управления (перехода, ветвления).
3. Арифметических и логических операций.
4. Машинного управления (управления процессором).

Этот список можно расширить, разбивая группы на подгруппы, выделяя их по другим признакам и т. д. Для каждой конкретной модели процессора разработчик в документации приводит свой пере-

чень групп команд. Но любая команда любого процессора относится к одному из этих четырех типов. Иногда в качестве отдельной группы выделяются команды работы с битами (команды битовых операций). Их можно отнести или к командам передачи данных на уровне отдельных битов, или к командам машинного управления, если это команды установки или сброса управляющих флагов, или к арифметико-логическим командам. Например, сдвиг на один бит влево — это умножение на два, вправо — целочисленное деление на два.

Существует два основных типа информации, содержащейся в команде:

1. Информация о действии, которое нужно выполнить.

2. Информация, которая требуется для выполнения этого действия.

Первый тип информации называется типом операции. Информация этого типа содержится в коде операции команды.

Второй тип может в себя включать:

− данные, над которыми будет осуществлено действие (операция) — входные непосредственные операнды;

− информацию о местонахождении входных и выходных операндов (результатов) (способ адресации).

Информация второго типа, если говорить о процессоре вообще, может содержаться в коде операции или дополнительных байтах команды.

Формат команды — это ее структура, форма представления вышеуказанной информации в команде, способ ее распределения по байтам и битам, способ толкования процессором байтов, групп битов (полей) и отдельных битов. Подчеркнем, что речь здесь идет о представлении команды на языке процессора — машинном языке.

Частью свойства формата команды является ее размер (длина). Он задается в байтах, зависит от способа адресации, типов операндов и других факторов. Максимальная длина команды процессора i8086 составляет шесть байтов, IA-32 и Intel 64 — 15. В первом байте всегда содержится код операции. Он может оказаться единственным

байтом команды. В остальных байтах содержится информация второго типа: для процессора i8086 во втором байте (байте ModR/M) — способ адресации, в остальных — смещение и непосредственные данные (табл. 3.5).

Таблица 3.5

Наиболее полный формат команды i8086

6 бит	1	1	2 бита	3 бита	3 бита	1 или 2 байта	1 или 2 байта
COP	D	W	MOD	REG	R/M	DISP8/16	DATA8/16

Команда i8086 содержит следующие поля:

1. COP — код операции.
2. D (Direction) —направление передачи:

D = 0 — первый (левый) операнд — источник, второй (правый) — приемник;

D = 1 — второй (правый) операнд — источник, первый (левый) — приемник.

3. W (Word) — размер операнда:

W = 0 — байт;

W = 1 — слово.

4. MOD (Mode) — способ (режим) адресации.
5. REG — регистр, первый операнд.
6. R/M — регистр или память, второй операнд.
7. DISP8/16 — смещение операнда 8 или 16 бит.
8. DATA8/16 — непосредственный операнд 8 или 16 бит.

Кодировки команд представляют собой подмножества формата, приведенного в табл. 3.5. Минимальный размер команды — один байт, и он может состоять только из кода операции. Также перед командой могут находиться так называемые префиксы повторения, указания сегментного регистра и т. д. (см. п. 3.3).

Цикл команды состоит из нескольких циклов шины, которые состоят из нескольких тактов. Таким образом, любая команда процес-

сора i8086 выполняется более, чем за один такт. В процессорах IA-32 и Intel 64 за счет использования конвейеров и всего, что с ними связано, за один такт могут быть выполнены две целочисленные команды и одна команда FPU.

Тип операнда определяют следующие свойства:

1. Размер.
2. Кодировка.
3. Беззнаковый или знаковый.
4. Целый или дробный.
5. Скалярный или векторный.

Скалярный операнд представляет собой число, векторный — последовательность (массив) чисел. По этому признаку команды делятся на скалярные и векторные. Векторная команда выполняется быстрее по сравнению с последовательностью аналогичных скалярных команд за счет того, что декодируется один раз. Векторным процессором в x86 является устройство MMX со всеми последующими расширениями.

У процессора i8086 единственный тип данных — скалярные целые числа в дополнительном коде размером один или два байта. (Результат команд умножения может быть 4-байтным числом, размещаемым в двух регистрах.) После вхождения в состав процессора устройств FPU и MMX появилось большое количество новых типов данных — скалярных вещественных и векторных. Адресация операндов осуществляется с использованием пяти базовых типов в различных комбинациях, что рассматривается в следующем параграфе.

По отношению к регистру флагов выделяются следующие группы команд:

1. Влияющие на флаги.
2. Проверяющие флаги (команды условных переходов, циклов).
3. Использующие флаги в качестве операндов при выполнении действий (команды сложения с переносом и т. п.).
4. Не имеющие отношения к регистру флагов.

Влияние команд на флаги бывает следующих типов:

1. В соответствии с результатом операции.
2. Установка (в единицу).
3. Сброс (установка в ноль).
4. Восстановление предыдущих значений, например, IRET.
5. Неопределенное.

Основные правила определения влияния команды на флаги таковы:

1. Не влияют на регистр флагов команды передачи данных и передачи управления.
2. Команды арифметических и логических операций влияют на регистр флагов.

Из этих правил имеются исключения. Они определяются назначением (спецификой) команд и вполне очевидны. Это, например, команды SAHF (Store AH in Flags, разместить AH в регистре флагов) и POPF (извлечь из стека регистр флагов).

Среди команд управления процессором имеются такие, которые не влияют на регистр флагов, например, HLT (останов), и такие, которые влияют на него в соответствии со своим назначением, например, STI (Stand Interrupt).

Способы адресации

Способ адресации — это способ задания местонахождения операндов команды, как входных, так и выходных.

Поскольку в вычислительной системе нет ничего, кроме процессора, памяти и устройств (портов) ввода-вывода, то физически операнды могут находиться в каком-либо из этих элементов.

С программной точки зрения память делится на память команд, память данных и стек. Поэтому варианты местонахождения операндов таковы:

1. Регистр процессора.

2. Команда (память команд).
3. Ячейка памяти данных.
4. Ячейка стека.
5. Порт ввода-вывода.

Адресация операндов в регистре процессора. Неявная и регистровая адресации

Если операнд находится в регистре процессора, то возможны два способа задания его местоположения. Если по логике команды операнд должен располагаться в определенном регистре в соответствии с его специализацией, то указывать имя этого регистра в команде нет необходимости. Это, например, команды загрузки регистра AH флагами (LAHF) и размещения AH в регистре флагов (SAHF), строковые команды загрузки и размещения аккумулятора: LODS (Load String) и STOS (Store String). В арифметических и логических командах многих процессоров подразумевается, что один из операндов и результат находятся в аккумуляторе (откуда и происходит название «аккумулятор» или другое его название «регистр результата»). В командах ввода-вывода также неявно подразумевается, что источником в случае вывода и приемником в случае ввода является аккумулятор. Такая адресация называется неявной.

Если операнд команды может находиться в различных регистрах, то в команде необходимо указать имя требуемого регистра. Такая адресация называется регистровой. Это, например, команды передачи данных из регистра в регистр общего назначения MOV reg1, reg2.

Для i8086 формат этих команд таков:

1000 10dw 11reg1reg2

Возможны два значения машинного кода команды, выполняющей одно и то же действие:

MOV reg1, reg2	:	1000 100w 11reg2reg1	:	reg1 ← reg2
		1000 101w 11reg1reg2	:	reg1 ← reg2
MOV reg2, reg1	:	1000 100w 11reg1reg2	:	reg2 ← reg1
		1000 101w 11reg2reg1	:	reg2 ← reg1

Они различаются значением бита направления и расположением значений имен регистров в команде. Возьмем конкретные регистры AX (поле reg = 000) и BX (reg = 011):

MOV AX, BX	:	1000 1011 11000011	(8B C3)	:	AX ← BX
MOV AX, BX	:	1000 1001 11011000	(89 D8)	:	AX ← BX
MOV BX, AX	:	1000 1011 11011000	(8B D8)	:	BX ← AX
MOV BX, AX	:	1000 1001 11000011	(89 C3)	:	BX ← AX
MOV AL, BL	:	1000 1010 11000011	(8A C3)	:	AL ← BL
MOV AL, BL	:	1000 1000 11011000	(88 D8)	:	AL ← BL
MOV BL, AL	:	1000 1010 11011000	(8A D8)	:	BL ← AL
MOV BL, AL	:	1000 1000 11000011	(88 C3)	:	BL ← AL

В ассемблерной записи всегда первый операнд — приемник, второй — источник. Компилятор tasm выбирает тот вариант машинного кода, который соответствует ассемблерному коду, т. е. когда d = 1, и слева расположен регистр-приемник, а справа — регистр-источник. Можно провести эксперимент. Напишем программу:

```
segment data
        a dw 0Fh
        b dw 0F0h
ends data
assume cs:code, ds:data
segment code
```

```
begin: mov dx, data
       mov ds, dx
       mov ax, a
       mov bx, b
       mov ax, bx
       mov ax, 4C00h
       int 21h
ends code
end begin
```

Скомпилируем ее, скомпонуем и откроем в отладчике. Посмотрим машинный код, выполним команду mov ax, bx, посмотрим результат (рис. 3.19).

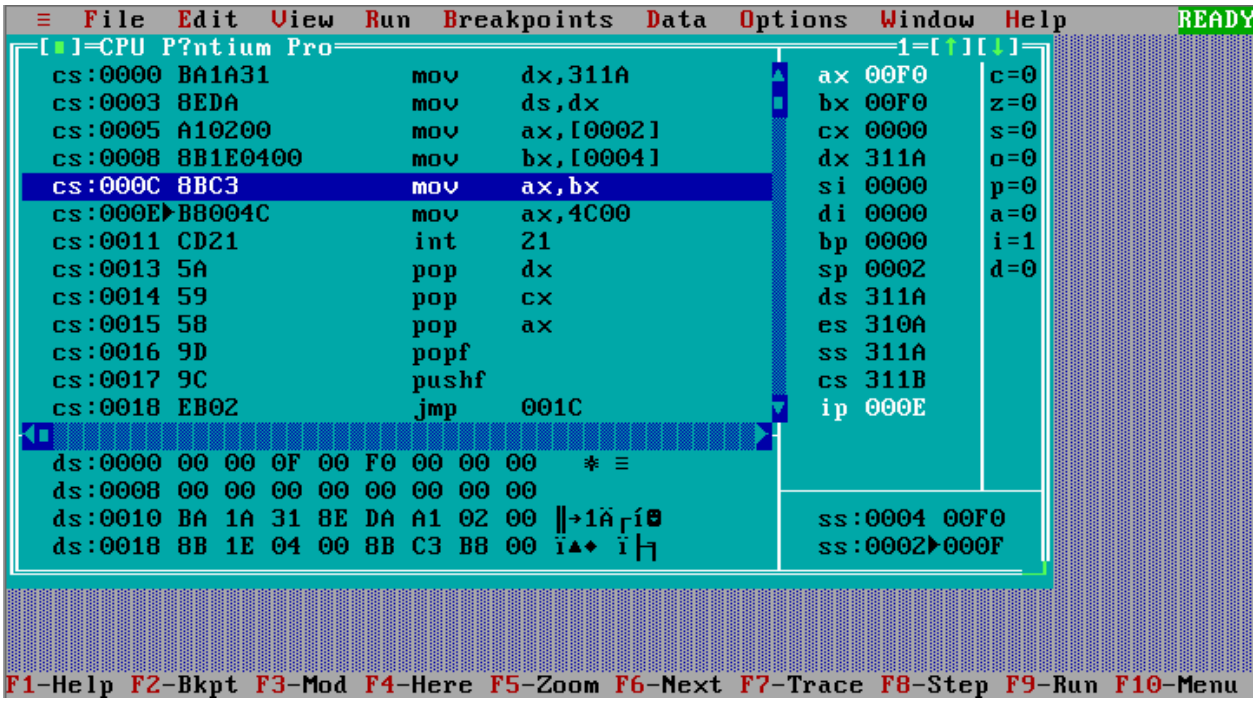


Рис. 3.19. Машинный код команды mov ax, bx — 8BC3

Исправим команду в шестнадцатеричном редакторе или в самом отладчике на 89D8, выполним команду и убедимся, что результат получился тот же (рис. 3.20).

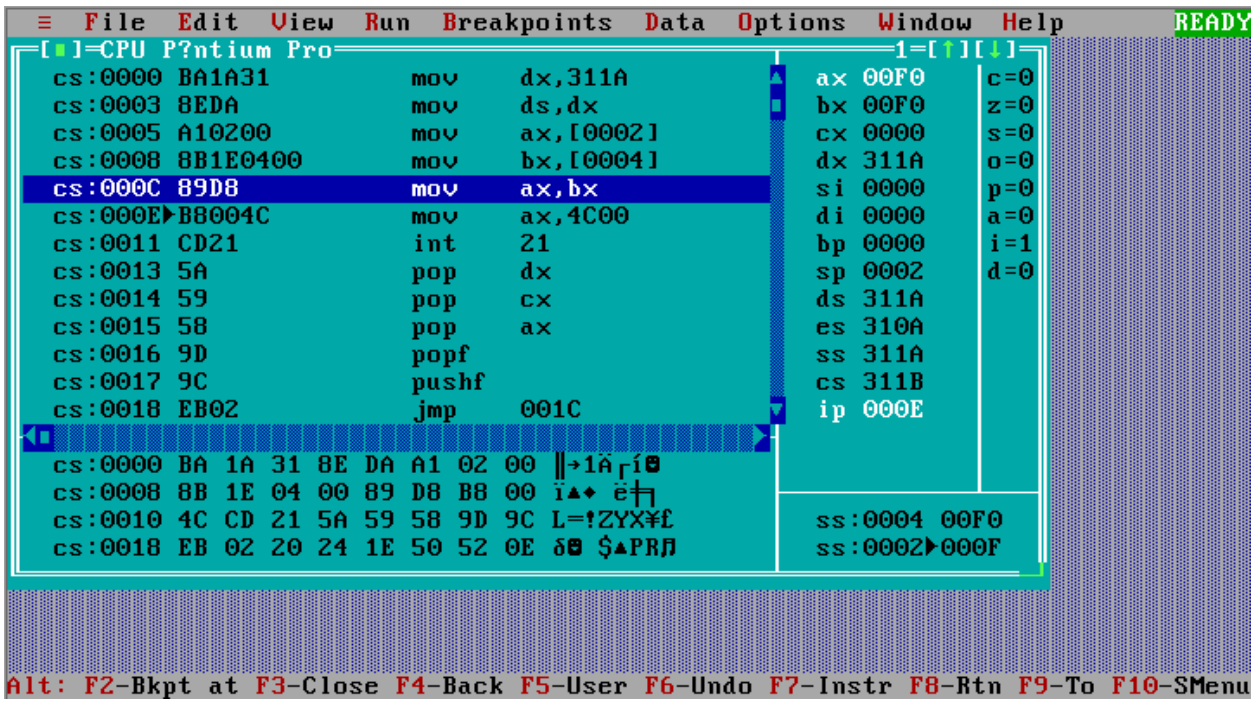


Рис. 3.20. Машинный код команды mov ax, bx — 89D8

Адресация операндов в команде. Непосредственная адресация

Если операнд находится в команде, то он следует непосредственно за кодом операции. Такая адресация называется непосредственной.

Примеры:

команды передачи непосредственных данных в регистр:

MOV reg, data8/16 : 1011w reg data8/16

MOV CX, 0FF0 : 10111001 11110000 00001111 (B9 F0 0F)

команды непосредственного сложения с аккумулятором (ACC = AX или AL):

ADD ACC, data8/16 : 0000010w data8/16

ADD AX, 0102 : 00000101 00000010 00000001 (05 02 01)

ADD AL, 02 : 00000100 00000010 (04 02)

Адресация операндов в памяти данных. Прямая и косвенная адресации

Если операнд находится в памяти данных, то указать его местоположение можно двумя способами. Первый — поместить адрес операнда в команду. Такая адресация называется прямой.

Пример:

команда передачи значения переменной в аккумулятор:

MOV ACC, disp8/16	:	1010000w disp8/16	
MOV AX, VAR_W	:	10100001 disp16	(A1 disp16)
MOV AL, VAR_B	:	10100000 disp16	(A0 disp16)

Здесь:
VAR_W — переменная, объявленная как слово: VAR_W dw value16;
VAR_B — переменная, объявленная как байт: VAR_B db value8;
Но и у той, и у другой переменной адрес 16-разрядный — disp16.

Второй способ указания местоположения операнда — поместить его адрес в регистр процессора. Такая адресация называется косвенной или косвенно-регистровой.

Примеры:

команда передачи в аккумулятор значения в памяти по адресу, находящемуся в регистре BX:

MOV AX, [BX]	:	10001011 00000111	(8B 07)	— здесь d = 1.

или наоборот:

MOV [BX], AX	:	10001001 00000111	(89 07)	— здесь d = 0.

Видно, что косвенная адресация тесно связана с понятием указателя. В названии некоторых регистров, которые могут содержать адрес операнда, присутствует слово «указатель» (pointer): SP (Stack Pointer), BP (Base Pointer). Но любой регистр, содержащий адрес, является указателем.

Благодаря наличию такой особенности архитектуры x86, как модификация адреса (см. п. 3.2), прямая и косвенная адресация могут комбинироваться:

```
MOV AX, VAR[BX]          :    10001011 10000111 disp16 (8B 87 disp16)
MOV VAR[BX][DI], DX      :    10001001 10010001 disp16 (89 91 disp16)
```

Адресация операндов в стеке

Если операнд находится в стеке, то возможны два варианта его адресации (табл. 3.6):

− косвенная с использованием указателя стека SP;

− комбинация косвенной и прямой адресации с использованием указателя базы BP.

Таблица 3.6

Адресация операндов в стеке

Регистр-указатель	**Адресация**	**Кодировка регистров-указателей в команде**	**Команды**	**Доступная область стека**
SP	косвенная	неявная	PUSH, POP; CALL, RET, IRET	верхушка
BP	косвенная плюс прямая	явная с использованием поля R/M байта MOD	MOV, ADD, ADC, SUB, SBB, CMP, AND, OR, XOR и др.	весь стек

Регистр SP, как уже говорилось, указывает на верхушку стека, соответственно он используется для работы с нею. Извлекается из стека значение верхушки, затем SP увеличивается и указывает на новую верхушку. При помещении в стек SP уменьшается, и в новую верхушку записывается значение.

Регистр BP используется для доступа к внутренним элементам стека, например, к параметрам, переданным подпрограмме, которые,

естественно, помещаются в стек перед ее вызовом. После вызова подпрограммы в верхушке стека располагается адрес возврата, а параметры оказывается в глубине под ним. BP инициализируется таким образом:

```
MOV BP, SP
```

Так что после инициализации он указывает так же, как и SP, на верхушку стека. А дальше для доступа к параметрам к нему может быть прибавлено необходимое значение, например:

```
ADD BP, 4
```

или значение смещения указывается в самой команде доступа к стеку:

```
MOV reg, [BP+disp8/16]        :        1000101w mod reg110 disp8/16
```

Здесь бит направления d = 1 — это значит, что первый операнд (reg) — приемник, второй (ячейка стека) — источник.

```
MOV [BP+disp8/16], reg        :        1000100w mod reg110 disp8/16
```

Здесь d = 0 — первый операнд (reg) — источник, второй (стек) — приемник.

Если mod = 10, то смещение имеет размер 2 байта (disp16);
Если mod = 01, то смещение имеет размер 1 байт (disp8);
R/M = 110 в сочетании с mod, равным 01 или 10, означает, что адрес второго операнда BP+disp8/16.

Для арифметических и логических команд справедливо то же самое.

Примеры:

```
MOV AX, [BP+2]:     10001011 01000110 00000010     (8B 46 02)
MOV AX, [BP]   :    10001011 01000110 00000000     (8B 46 00)
MOV BX, [BP]   :    10001011 01011110 00000000     (8B 5E 00)
ADD BX, [BP]   :    00000011 01011110 00000000     (03 5E 00)
ADC BX, [BP]   :    00010011 01011110 00000000     (13 5E 00)
SUB BX, [BP]   :    00101011 01011110 00000000     (2B 5E 00)
SBB BX, [BP]   :    00011011 01011110 00000000     (1B 5E 00)
AND BX, [BP]   :    00100011 01011110 00000000     (23 5E 00)
OR BX, [BP]    :    00001011 01011110 00000000     (0B 5E 00)
XOR BX, [BP]   :    00110011 01011110 00000000     (33 5E 00)
```

Здесь:

w = 1 — размер операнда — слово;

d = 1 — первый операнд (reg) — приемник, второй (стек) — источник;

mod = 01 — смещение 1 байт (disp8);

reg = 000 — регистр AX;

reg = 011 — регистр BX.

Надо обратить внимание на два момента. Если даже в ассемблерной записи disp8/16 = 0, в машинном коде это нулевое смещение присутствует. Если операнд-приемник в стеке, то бит направления d равен 0, т. е. поля REG и R/M местами не меняются:

```
MOV [BP], AX   :    10001001 01000110 00000000 (89 46 00)
ADD [BP], BX   :    00000001 01011110 00000000 (01 5E 00)
```

Это касается и других команд с косвенной адресацией.

Вместе с регистром BP может использоваться индексный регистр SI или DI:

```
MOV AX, [BP+SI+2]      :    10001011 01000010 00000010 (8B 42 02)
MOV [BP+DI+4], AX      :    10001001 01000011 00000100 (89 43 04)
```

В команде возврата из подпрограммы RET может присутствовать непосредственный операнд, который процессор прибавит к указателю стека SP после извлечения из стека адреса возврата. Это делается для очистки стека от параметров, переданных подпрограмме.

Адресация операндов в устройствах ввода-вывода

Если операнд находится в устройстве ввода-вывода, то возможны прямая и косвенная адресации.

Примеры:

прямая адресация, в команде указан номер (адрес) порта:

```
IN AL, 60h      :    11100100 01100000    (E4 60)
OUT 61h, AL     :    11100110 01100001    (E6 61)
```

косвенная адресация, адрес порта содержится в DX:

```
IN AL, DX       :    11101100    (EC)
IN AX, DX       :    11101101    (ED)
OUT DX, AL      :    11101110    (EE)
OUT DX, AX      :    11101111    (EF)
```

Имя регистра DX в машинном коде команды не присутствует, так как только регистр DX в соответствии со своей специализацией может использоваться в качестве указателя порта ввода-вывода. Имя аккумулятора также не присутствует в машинном коде. Таким образом, и аккумулятор, и регистр данных в этих командах подразумеваются неявно.

Приведенные примеры относятся к случаю так называемого изолированного ввода-вывода (см. п. 2.2), т. е. когда внешнее устройство не имеет такого ресурса, как диапазон памяти в адресном пространстве и для обмена данными с ним можно использовать только команды IN и OUT. В случае наличия такого ресурса используется ввод-вывод по принципу доступа к памяти (пример — видеопамять),

и можно использовать любые команды. С программной точки зрения операнд находится как бы в памяти данных, и здесь также возможны прямая и косвенная адресация или их комбинация.

Таким образом, по разным признакам выделяется пять базовых типов адресации (табл. 3.7):

1. Неявная.
2. Регистровая.
3. Непосредственная.
4. Прямая.
5. Косвенная.

Таблица 3.7

Типы адресации

Объект / **Расположение**	**Операнд**	**Адрес операнда**
Регистр	1–2. Неявная или регистровая	5. Косвенная
Команда	3. Непосредственная	4. Прямая

Строка таблицы — это местоположение информационного объекта (регистр или команда**). Столбец** — тип объекта (собственно операнд или его адрес). Пересечение дает способ адресации. Соответствие способа адресации и местонахождения операнда таково:

- 1–2 — регистр;
- 3 — память команд;
- 4 — память данных или внешнее устройство;
- 5 — память данных, стек или внешнее устройство.

Надо отметить, что явно или неявно в команде может быть указано имя регистра как с самим операндом, так и с его адресом. Однако, название «неявная адресация» используется только применительно к первому случаю.

Еще раз отметим, что прямая и косвенная адресация могут комбинироваться в разных вариантах (табл. 3.8). Все эти комбинации предоставляют удобства при работе с массивами.

Таблица 3.8

Общий формат команд с комбинацией прямой и косвенной адресации для одного из операндов, другой операнд — регистр

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
COP						d	w	mod			reg			R/M		disp16																
COP						d	w	mod			reg			R/M		disp8																

Если w = 0, то reg = reg8 — 8-разрядный регистр, если w = 1, то reg = reg16 — 16-разрядный регистр (табл. 3.9).

Таблица 3.9

Значения имен регистров в команде

reg	w = 0	w = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Если d = 0, то регистр является источником: M ← reg.

Если d = 1, то регистр является приемником: reg ← M.

Используемая комбинация базовых и индексных регистров и смещения задается полем ModR/M (табл. 3.10).

Таблица 3.10

Значения кодировки поля ModR/M

R/M	Mod = 00	Mod = 01	Mod = 10
000	(BX) + (SI)	(BX) + (SI) + DISP8	(BX) + (SI) + DISP16
001	(BX) + (DI)	(BX) + (DI) + DISP8	(BX) + (DI) + DISP16
010	(BP) + (SI)	(BP) + (SI) + DISP8	(BP) + (SI) + DISP16
011	(BP) + (DI)	(BP) + (DI) + DISP8	(BP) + (DI) + DISP16
100	(SI)	(SI) + DISP8	(SI) + DISP16
101	(DI)	(DI) + DISP8	(DI) + DISP16
110	DISP 16	(BP) + DISP8	(BP) + DISP16
111	(BX)	(BX) + DISP8	(BX) + DISP16

Примеры:

```
8B 00              :   MOV AX, [BX+SI]
8B 01              :   MOV AX, [BX+DI]
8B 02              :   MOV AX, [BP+SI]
8B 03              :   MOV AX, [BP+DI]
8B 04              :   MOV AX, [SI]
8B 05              :   MOV AX, [DI]
8B 46 00           :   MOV AX, [BP]
8B 07              :   MOV AX, [BX]

8B 40 disp8=02     :   MOV AX, [BX+SI+2]
8B 41 disp8=02     :   MOV AX, [BX+DI+2]
8B 42 disp8=02     :   MOV AX, [BP+SI+2]
8B 43 disp8=02     :   MOV AX, [BP+DI+2]
8B 44 disp8=02     :   MOV AX, [SI+2]
8B 45 disp8=02     :   MOV AX, [DI+2]
8B 46 disp8=02     :   MOV AX, [BP+2]
8B 47 disp8=02     :   MOV AX, [BX+2]

8B 80 disp16=VAR+2 :   MOV AX, VAR[BX+SI+2]
8B 81 disp16=VAR+2 :   MOV AX, VAR[BX+DI+2]
8B 82 02 01        :   MOV AX, [BP+SI+0102h]
```

```
8B 83 02 01              :    MOV AX, [BP+DI+0102h]
8B 84 disp16=VAR+2       :    MOV AX, VAR[SI+2]
8B 85 disp16=VAR+2       :    MOV AX, VAR[DI+2]
8B 86 02 01              :    MOV AX, [BP+0102h]
8B 87 disp16=VAR+2       :    MOV AX, VAR[BX+2]
```

Итак, один операнд может адресоваться одним из базовых способов или комбинацией прямой и косвенной адресации. Но в большинстве команд несколько операндов. В командах передачи данных их два: один входной (операнд-источник) и один выходной (операнд-приемник); в логических и арифметических командах — три: два входных и один выходной (результат); в командах передачи управления также несколько операндов. И каждый из операндов может адресоваться по-разному. Таким образом, в команде может присутствовать множество различных способов адресации. Например, в команде CALL ptr16:16 присутствуют три способа адресации:

− косвенная: адрес операнда-приемника дополнительного действия, помещения в стек адреса возврата, содержится в регистре SP: SS:[SP]← CS:IP;

− непосредственная: операнд-источник основного действия, передачи управления, адрес вызываемой подпрограммы (ptr16:16) содержится в команде непосредственно за кодом операции: CS:IP←ptr16:16;

− неявная: операнд-приемник основного действия и операнд-источник дополнительного действия — пара регистров CS:IP, которая определяется неявно типом операции и своей специализацией.

Конечно, не все сочетания способов адресации в одной команде возможны. Во всех приведенных выше примерах помимо непосредственной, прямой или косвенной присутствует регистровая или неявная адресация. Регистровая и неявная могут использоваться с любой адресацией. Непосредственная адресация не может использоваться с непосредственной, т. е. непосредственный операнд может быть

только один, а со всеми другими способами она сочетается. Например, непосредственная и косвенная адресация:

```
MOV [BX], 1      :      C7 07 01 00
```

Непосредственная и прямая:

```
MOV VAR_W, Value16          :    C7 06 disp16 data16
MOV word ptr ds:[0002], 1   :    C7 06 0200 0100
MOV VAR_B, Value8           :    C6 06 disp16 data8
MOV byte ptr data:[0002], 1 :    C6 06 0200 01
```

Здесь в ассемблерной записи команд при задании числового значения смещения переменной (в квадратных скобках) используется имя сегментного регистра или сегмента для указания, в каком сегменте находится переменная, а также ключевое слово ptr (pointer) для указания типа (размера) переменной — слово или байт.

Непосредственная адресация и комбинация прямой и косвенной:

```
MOV VAR_W[BX], Value16      :    C7 87 disp16 data16
MOV VAR_B[BX], Value8       :    C6 87 disp16 data8
```

Сочетание прямой и косвенной адресации для разных операндов возможно только в командах PUSH и POP, т. е. можно поместить в стек или извлечь из стека 16-битную переменную с использованием различных комбинаций ModR/M:

```
PUSH VAR_W          ⎫
PUSH VAR_W[BX]      ⎬    :    11111111 mod110R/M disp8/16
PUSH [BX+4]         ⎭
PUSH word ptr [0004+BX]  :    FF 36 0400
PUSH [BX+4]              :    FF 77 04
```

Косвенная адресация для разных операндов возможна только в строковых командах:

CMPS	:	1010011W
MOVS	:	1010010W

В них два массива, источник и назначение, адресуются с использованием индексных регистров SI и DI.

Существуют команды, которые не используют никаких способов адресации, например, HLT, WAIT. Это те команды, которые не требуют для выполнения действия никакой дополнительной информации, т. е. информация второго типа в них просто отсутствует.

Система команд процессоров IA-32 и Intel 64

В процессорах IA-32 и Intel 64 расширились существующие регистры и появились новые. В системе команд также произошли соответствующие изменения. Общий формат команды процессоров IA-32 и Intel 64 представлен на рис. 3.21 [57].

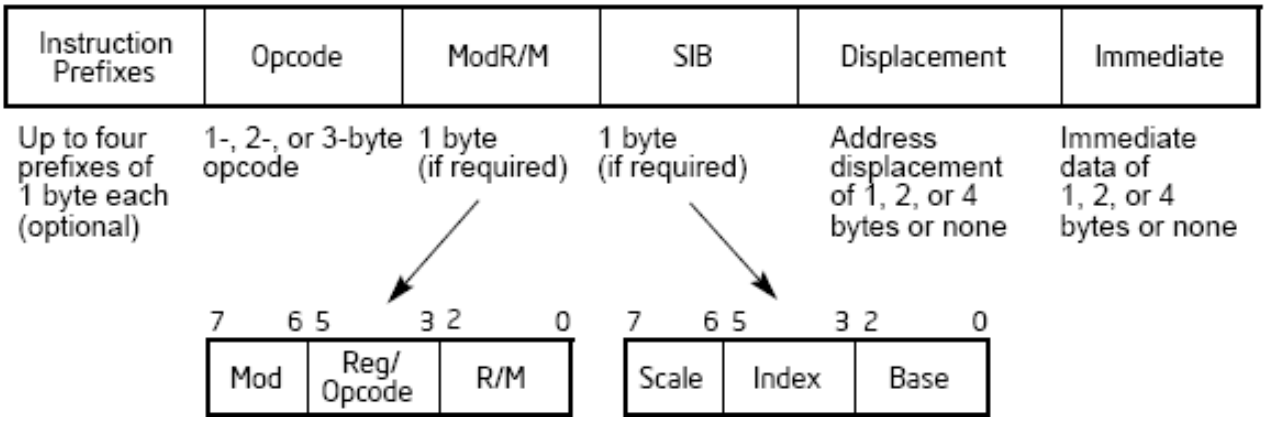


Рис. 3.21. Формат команды процессоров IA-32 и Intel 64

Максимальная длина команды — 15 байтов. Возможно наличие еще одного байта — SIB (Scale, Index, Base), в котором указываются

коэффициент масштабирования при работе с массивами (размер элементов массива в байтах), базовый и индексный регистры.

Префиксы были и в системе команд процессора i8086, но там они рассматривались как отдельные специфические команды управления процессором, действие которых распространяется только на следующую за ними команду. Сейчас в документации Intel они рассматриваются как часть команды. Префиксы делятся на четыре группы:

1. Блокировки и повторения.
2. Замены сегмента и указания ветвления.
3. Замены размера операнда, машинный код 66h.
4. Замены размера адреса, машинный код 67h.

В каждой команде может быть использовано по одному префиксу из каждой группы в любом порядке. Действие избыточных префиксов (более одного из группы) является неопределенным и меняется от процессора к процессору.

В расширениях SSE и AVX используются свои префиксы. В SSE они имеют те же машинные коды, что и обычные префиксы, но трактуются иначе. В AVX используется отдельный префикс VEX, который может иметь размер два или три байта. В 64-битном режиме используется также префикс REX.

Префиксы блокировки и повторения:

F0h — LOCK;

F2h — REPNE/REPNZ (Repeat while Not Equal/Zero);

F3h — REP/REPE/REPZ (Repeat while Equal/Zero).

Префикс блокировки LOCK переводит одноименный сигнал в активный уровень на время выполнения команды, снабженной данным префиксом.

Префиксы повторения используются только со строковыми командами. Они приводят к тому, что команда выполняется множество раз для заданного количества элементов строки. Фактически они реализуют циклы при работе с массивами. Счетчиком цикла является регистр (E)CX (RCX). К тому же при использовании условных префик-

сов дополнительно проверяется условие, и цикл прекращается, когда условие перестает выполняться. Соответственно, условные префиксы применяются с арифметическими строковыми командами (CMPS, SCAS), а безусловный — с командами передачи данных (INS, OUTS, MOVS, LODS, STOS). Например:

```
MOVSD:        ES:[EDI] ← DS:[ESI];   ESI=ESI+4, EDI=EDI+4 при DF=0
                                     ESI=ESI–4, EDI=EDI–4 при DF=1
REP MOVSB:    {ES:[EDI] ← DS:[ESI]; ESI=ESI+1, EDI=EDI+1 при DF=0
                                     ESI=ESI–1, EDI=EDI–1 при DF=1}
              — эти действия будут выполнены ECX раз.
```

Указателями в строковых командах являются индексные регистры (E)SI (RSI) и (E)DI (RDI). На каждом шаге они изменяются на величину элемента массива в байтах: 1 — байт (B), 2 — слово (W), 4 — двойное слово (D), 8 — учетверенное слово (Q).

Префиксы замены сегмента:

26h — ES;
2Eh — CS;
36h — SS;
3Eh — DS;
64h — FS;
65h — GS.

Два из них также используются как префиксы указания ветвления:

2Eh — ветвление не принимается;

3Eh — ветвление принимается.

Префиксы замены сегмента могут быть использованы с любыми командами, префиксы указания ветвления используются только с командами условных переходов. Они позволяют программе давать указание процессору о наиболее вероятном пути хода программы. Они введены в Pentium 4 как часть расширения SSE2.

Префиксы замены размера операнда 66h и замены размера адреса 67h используются в следующих случаях. Начиная с архитектуры IA-32, сегменты могут быть 16-разрядными (смещение 16 бит, максимальный размер сегмента 64 Кбайт) и 32-разрядными (смещение 32 бита, максимальный размер сегмента 4 Гбайт). Если в 16-разрядном сегменте используются 32-разрядные операнды или наоборот в 32-разрядном сегменте используются 16-разрядные операнды, то используется префикс 66h.

Например:

MOV DX, CX
в 16-разрядном сегменте: 8B D1
в 32-разрядном сегменте: 66 8B D1
MOV EDX, ECX
в 16-разрядном сегменте: 66 8B D1
в 32-разрядном сегменте: 8B D1

То же самое касается и адресов операндов. Например, если 32-разрядная переменная объявлена в 32-разрядном сегменте, то в 16-разрядном сегменте команда записи в нее значения EDX будет выглядеть так:

MOV VAR, EDX : 66 67 89 15 disp32

В некоторых командах префиксы повторения и префикс замены размера операнда 66h используются в качестве обязательных префиксов для выражения различных функциональных возможностей. В таких командах, в частности командах SIMD-расширений, они предшествуют так называемому escape-байту 0Fh (escape opcode byte 0Fh). У процессора i8086 была команда ESC (Escape), которая имела несколько машинных кодов (D8-DF) и использовалась для передачи математическому сопроцессору тех команд программы, которые относились к его набору. Сейчас все команды FPU x87 относятся к классу так называемых escape-команд. Они имеют общий формат кода опе-

рации, в котором первым байтом является одно из значений от D8h до DFh. Такая же система у команд устройства MMX — SIMD-команд со всеми последующими расширениями. У них в коде операции присутствует escape-байт 0Fh.

Примеры:

Команда преобразования двух упакованных знаковых двойных слов из регистра XMM или памяти в два упакованных числа с плавающей точкой удвоенной точности в регистре XMM:

CVTDQ2PD — F3 0F E6.

Здесь первый байт — обязательный префикс команд SSE/SSE2/SSE3, а не префикс повторения, второй байт — escape-байт, третий — собственно код операции.

Команда горизонтального сложения 16-разрядных знаковых целых с упаковкой в регистр XMM:

PHADDW — 66 0F 38 01.

Здесь первый байт — обязательный префикс команд SSE/SSE2/SSE3, а не префикс замены размера операнда, второй байт — escape-байт, третий и четвертый байты — собственно код операции.

Примечание. В расширениях SSE различаются команды горизонтальных и вертикальных вычислений (см. п. 5.1). Горизонтальное сложение (horizontal add) выполняется над попарными соседними элементами вектора.

Формат кода операции зависит от типа команды. В общем случае он содержит поля, которые определяют:

- тип операции (COP);
- направление операции, т. е. куда записывается результат — в первый операнд или второй (d);
- размер данных (w) — байт или полный размер (2 или 4 байта);
- имя используемого регистра (reg, sreg2, sreg3, eee (CR или DR));
- условие (cond (tttn));
- расширение со знаком непосредственных данных (s).

Многие команды, операнды которых находятся в памяти, имеют байт способа адресации ModR/M. Этот байт содержит три поля:

– поле Mod (биты 7–6) в комбинации с полем R/M (биты 2–0) дает 32 возможных значения: 8 регистров и 24 способа адресации;

– поле Reg/Opcode (R/O, биты 5–3) содержит имя регистра или дополнительную часть кода операции в зависимости от значения первичного кода операции;

– поле R/M определяет используемый регистр или в комбинации с полем Mod — способ адресации.

Определенные значения байта ModR/M для полного определения способа адресации требуют байта SIB. Он содержит следующие поля:

– Scale (биты 7–6) — коэффициент масштабирования;

– Index (биты 5–3) — номер индексного регистра;

– Base (биты 2–0) — номер регистра базы.

В архитектуре IA-32 и Intel 64 существует возможность масштабирования с коэффициентами 1, 2, 4, 8. Масштабирование предоставляет удобство при работе с массивами элементов различного размера. В качестве регистров-указателей могут использоваться любые 32-разрядные регистры общего назначения (специализация снижается — движение в сторону RISC-архитектуры), за одним исключением — в качестве индексного регистра не может использоваться ESP. Индексный регистр в случае использования масштабирования при переходе к следующему элементу массива изменяется на значение, равное коэффициенту масштабирования (рис. 3.22).

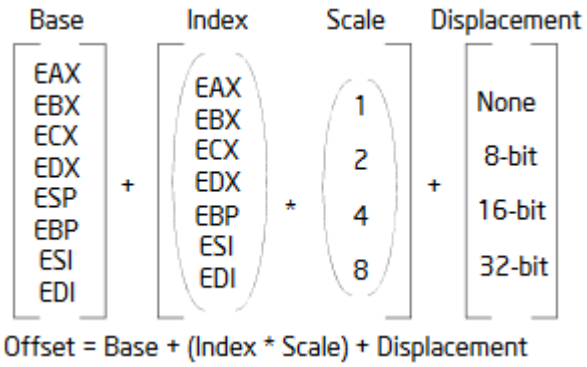


Рис. 3.22. Формирование эффективного адреса в IA-32 и Intel 64

В Intel 64 в 64-битном режиме значение базы и индекса может находиться в любом из 16 регистров общего назначения. Также в 64-битном режиме в команде может быть использован еще один префикс — REX (коды 40-4Fh), который должен стоять непосредственно перед кодом операции (рис. 3.23).

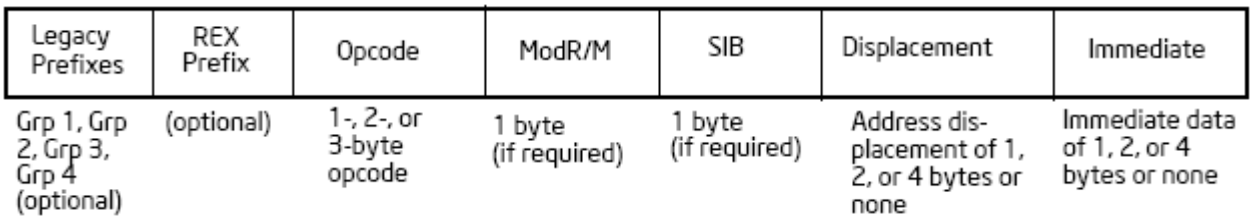


Рис. 3.23. Формат команды в 64-битном режиме

При этом ограничение на длину команды в 15 байтов остается в силе. Однако непосредственный операнд и смещение могут быть 8-байтными:

MOV RCX,1122334455667788h : 48 B9 88 77 66 55 44 33 22 11
MOV RAX, QWORD PTR [RBX + 0123456789ABCDEFh] :
48 8B 83 EF CD AB 89 67 45 23 01

Префикс REX имеет собственный формат: 0100WRXB (рис. 3.24):

- W: 0 — размер операнда определяется разрядностью сегмента;
  1 — размер операнда 64 бита;
- R — расширение поля reg байта ModR/M;
- X — расширение поля index байта SIB;
- B — расширение поля r/m байта ModR/M или поля base байта SIB или поля reg байта Opcode.

Надо отметить, что в соответствии с обратным порядком хранения слов в команде располагаются только байты смещения и непосредственного операнда. Байты кода операции, байт ModR/M и байт SIB располагаются в соответствии с прямым порядком хранения слов.

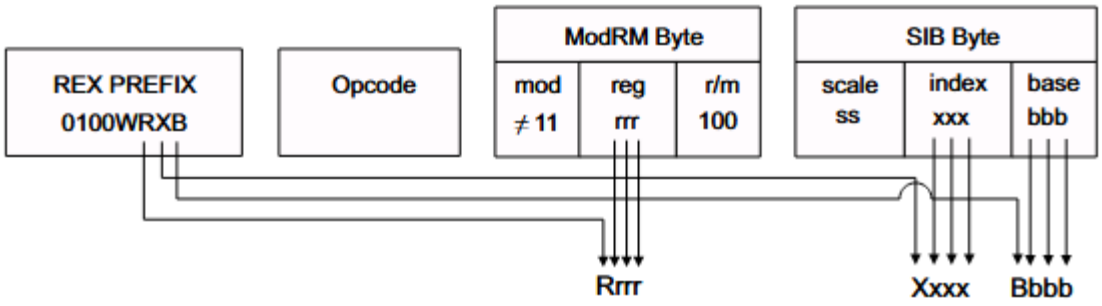


Рис. 3.24. Использование префикса REX при наличии байта SIB

Основные группы команды процессоров IA-32 и Intel 64 таковы:

– команды общего назначения (general purpose instructions);

– x87 FPU;

– MMX;

– SSE, SSE2, SSE3, SSSE3, SSE4;

– AVX, AVX2, AVX-512;

– системные;

– 64-битного режима;

– VMX (Virtual-Machine Extensions) — поддержка технологии Intel VT (Virtualization Technology);

– MPX (Memory Protection Extensions), предоставляют возможности надежной проверки границ при обращении к памяти.

Имеются также и другие группы: SMX (Safer Mode Extensions, расширений безопасного режима), SGX (Software Guard Extensions) и т. д.

Команды общего назначения делятся на следующие подгруппы:

1. Команды передачи данных.
2. Команды целочисленной двоичной арифметики.
3. Команды десятичной арифметики.
4. Команды логических операций.
5. Команды сдвига.
6. Команды битовых и байтовых операций.
7. Команды передачи управления.
8. Команды строковых операций.
9. Команды ввода-вывода.

10. Команды управления флагами.
11. Команды операций над сегментными регистрами.
12. Другие команды.

Полностью описать систему команд CISC-процессора, даже кратко, в таком учебном пособии невозможно. В технической документации Intel [57] описание системы команд составляет около двух тысяч страниц. Поэтому приведем здесь лишь мнемоники основных команд.

Команды передачи данных:

− MOV: передача данных между регистрами или регистрами и памятью;

− CMOVE/CMOVZ: условная передача, если равно / если ноль;

− CMOVNE/CMOVNZ: условная передача, если не равно / не ноль;

− CMOVA/CMOVNBE: условная передача, если выше / не ниже или равно;

− CMOVAE/CMOVNB: условная передача, если выше или равно / не ниже;

− CMOVB/CMOVNAE: условная передача, если ниже / не выше или не равно;

− CMOVBE/CMOVNA: условная передача, если ниже или равно / не выше;

− CMOVG/CMOVNLE: условная передача, если больше / не меньше или равно;

− CMOVGE/CMOVNL: условная передача, если больше или равно / не меньше;

− CMOVL/CMOVNGE: условная передача, если меньше / не больше или равно;

− CMOVLE/CMOVNG: условная передача, если меньше или равно / не больше;

− CMOVC: условная передача, если перенос;

− CMOVNC: условная передача, если нет переноса;

− CMOVO: условная передача, если переполнение;

− CMOVNO: условная передача, если нет переполнения;
− CMOVS: условная передача, если результат отрицательный;
− CMOVNS: условная передача, если результат неотрицательный;
− CMOVP/CMOVPE: условная передача, если четно;
− CMOVNP/CMOVPO: условная передача, если нечетно;
− XCHG: обмен между регистрами или регистром и переменной;
− BSWAP: обмен байтами;
− XADD: обмен и сложение;
− CMPXCHG: сравнение и обмен;
− CMPXCHG8B: сравнение и обмен 8 байтами;
− PUSH: помещение в стек;
− POP: извлечение из стека;
− PUSHA/PUSHAD: помещение в стек всех регистров общего назначения;
− POPA/POPAD: извлечение из стека всех регистров общего назначения;
− CWD/CDQ: преобразование слова в двойное слово / двойного слова в учетверенное слово;
− CBW/CWDE: преобразование байта в слово / слова в двойное слово в регистре EAX;
− MOVSX: передача и расширение знаком;
− MOVZX: передача и расширение нулем.

Команды целочисленной двоичной арифметики:

− ADCX: сложение беззнаковых чисел с переносом;
− ADOX: сложение беззнаковых чисел с переполнением;
− ADD: сложение;
− ADC: сложение с переносом;
− SUB: вычитание;
− SBB: вычитание с заемом;
− IMUL: знаковое умножение;
− MUL: беззнаковое умножение;
− IDIV: знаковое деление;
− DIV: беззнаковое деление;

− INC: инкремент;
− DEC: декремент;
− NEG: изменение знака (дополнение до двух);
− CMP: сравнение.

Команды десятичной арифметики:

− DAA: десятичная коррекция после сложения;
− DAS: десятичная коррекция после вычитания;
− AAA: ASCII коррекция после сложения;
− AAS: ASCII коррекция после вычитания;
− AAM: ASCII коррекция после умножения;
− AAD: ASCII коррекция перед делением.

Логические команды:

− AND: побитовое логическое И;
− OR: побитовое логическое ИЛИ;
− XOR: побитовое логическое Исключительное ИЛИ;
− NOT: побитовое логическое НЕ.

Команды сдвига:

− SAR: арифметический сдвиг вправо;
− SHR: логический сдвиг вправо;
− SAL/SHL: арифметический / логический сдвиг влево;
− SHRD: двойной сдвиг вправо;
− SHLD: двойной сдвиг влево;
− ROR: циклический сдвиг вправо;
− ROL: циклический сдвиг влево;
− RCR: циклический сдвиг через перенос вправо;
− RCL: циклический сдвиг через перенос влево.

Команды битовых и байтовых операций:

− BT: проверка бита;
− BTS: проверка и установка бита;
− BTR: проверка и сброс бита;
− BTC: проверка и инвертирование бита;
− BSF: сканирование битов вперед;
− BSR: сканирование битов назад;

− SETE/SETZ: установка байта, если равно / если ноль;

− SETNE/SETNZ: установка байта, если не равно / если не ноль;

− SETA/SETNBE: установка байта, если выше / если не ниже или равно;

− SETAE/SETNB/SETNC: установка байта, если выше или равно / если не ниже / если нет переноса;

− SETB/SETNAE/SETC: установка байта, если ниже / если не выше или равно / если перенос;

− SETBE/SETNA: установка байта, если ниже или равно / если не выше;

− SETG/SETNLE: установка байта, если больше / если не меньше или равно;

− SETGE/SETNL: установка байта, если больше или равно / если не меньше;

− SETL/SETNGE: установка байта, если меньше / если не больше или равно;

− SETLE/SETNG: установка байта, если меньше или равно / если не больше;

− SETS: установка байта, если знак минус;

− SETNS: установка байта, если знак плюс;

− SETO: установка байта, если переполнение;

− SETNO: установка байта, если нет переполнения;

− SETPE/SETP: установка байта, если четно;

− SETPO/SETNP: установка байта, если нечетно;

− TEST: логическое сравнение (логическое И без сохранения результата);

− CRC32: расчет контрольной суммы для проверки целостности данных;

− POPCNT: подсчет количества битов, установленных в 1.

Команды передачи управления:

− JMP: безусловный переход;

− JE/JZ: переход, если равно / если ноль;

− JNE/JNZ: переход, если не равно / если не ноль;

− JA/JNBE: переход, если выше / если не ниже или равно;

− JAE/JNB: переход, если выше или равно / если не ниже;

− JB/JNAE: переход, если ниже / если не выше или равно;

− JBE/JNA: переход, если ниже или равно / если не выше;

− JG/JNLE: переход, если больше / если не меньше или равно;

− JGE/JNL: переход, если больше или равно / если не меньше;

− JL/JNGE: переход, если меньше / если не больше или равно;

− JLE/JNG: переход, если меньше или равно / если не больше;

− JC: переход, если перенос;

− JNC: переход, если нет переноса;

− JO: переход, если переполнение;

− JNO: переход, если нет переполнения;

− JS: переход, если минус;

− JNS: переход, если плюс;

− JPO/JNP: переход, если нечетно;

− JPE/JP: переход, если четно;

− JCXZ/JECXZ: переход, если регистр CX / ECX равен нулю;

− LOOP: команда цикла со счетчиком ECX;

− LOOPZ/LOOPE: цикл со счетчиком ECX, пока ноль / пока равно;

− LOOPNZ/LOOPNE: цикл со счетчиком ECX, пока не ноль / пока не равно;

− CALL: вызов подпрограммы;

− RET: возврат из подпрограммы;

− IRET: возврат из обработчика прерывания;

− INT n: программное прерывание с указанным номером;

− INT1: отладочное прерывание (#DB, debug exception);

− INT3: прерывание по точкам останова (#BP, breakpoint exception);

− INTO: прерывание при переполнении (#OF, overflow exception);

− BOUND: проверка выхода за границы массива (#BR, BOUND range exceeded exception);

− ENTER: вход в подпрограмму с созданием стекового кадра заданного размера;

− LEAVE: выход из подпрограммы с освобождением стекового кадра.

Надо заметить, что из-за наличия сегментирования в процессорах x86 переходы и вызовы подпрограмм могут ближними или дальними. Переходы в пределах одного сегмента называются ближними, межсегментные — дальними.

Команды строковых операций:

− MOVS/MOVSB: передача строки / строки байтов из одного места в памяти в другое;

− MOVS/MOVSW: передача строки / строки слов из одного места в памяти в другое;

− MOVS/MOVSD: передача строки / строки двойных слов из одного места в памяти в другое;

− CMPS/CMPSB: сравнение двух строк / строк байтов;

− CMPS/CMPSW: сравнение двух строк / строк слов;

− CMPS/CMPSD: сравнение двух строк / строк двойных слов;

− SCAS/SCASB: сканирование строки / строки байтов;

− SCAS/SCASW: сканирование строки / строки слов;

− SCAS/SCASD: сканирование строки / строки двойных слов;

− LODS/LODSB: загрузка очередного элемента строки / строки байтов в аккумулятор;

− LODS/LODSW: загрузка очередного элемента строки / строки слов в аккумулятор;

− LODS/LODSD: загрузка очередного элемента строки / строки двойных слов в аккумулятор;

− STOS/STOSB: размещение аккумулятора в очередном элементе строки / строки байтов;

− STOS/STOSW: размещение аккумулятора в очередном элементе строки / строки слов;

− STOS/STOSD: размещение аккумулятора в очередном элементе строки / строки двойных слов;

− REP: повторение, пока ECX не равен нулю;

− REPE/REPZ: повторение, пока равно / пока ноль и ECX не равен нулю;

− REPNE/REPNZ: повторение, пока не равно / пока не ноль и ECX не равен нулю.

Префиксы повторения относятся к строковым командам, так как используются только вместе с ними.

Команды ввода-вывода:

− IN: ввод из порта;

− OUT: вывод в порт;

− INS/INSB: ввод очередного элемента строки / строки байтов из порта;

− INS/INSW: ввод очередного элемента строки / строки слов из порта;

− INS/INSD: ввод очередного элемента строки / строки двойных слов из порта;

− OUTS/OUTSB: вывод очередного элемента строки / строки байтов в порт;

− OUTS/OUTSW: вывод очередного элемента строки / строки слов в порт;

− OUTS/OUTSD: вывод очередного элемента строки / строки двойных слов в порт.

Команды управления флагами:

− STC: установка флага переноса;

− CLC: сброс флага переноса;

− CMC: инвертирование флага переноса;

− CLD: сброс флага направления;

− STD: установка флага направления;

− LAHF: загрузка младшего байта регистра флагов в регистр AH;

− SAHF: размещение регистра AH в младшем байте регистра флагов;

− PUSHF/PUSHFD: помещение FLAGS/ EFLAGS в стек;

− POPF/POPFD: извлечение FLAGS/ EFLAGS из стека;

− STI: установка флага прерывания;

− CLI: сброс флага прерывания.

Команды операций над сегментными регистрами:

− LDS: загрузка регистра DS;

− LES: загрузка регистра ES;

− LFS: загрузка регистра FS;

− LGS: загрузка регистра GS;

− LSS: загрузка регистра SS.

Во всех этих командах вместе с сегментным регистром загружается также регистр общего назначения, используемый с ним в качестве указателя, из переменной соответствующего размера. Регистр общего назначения и переменная указываются в командах в качестве аргументов.

Другие команды:

− LEA: загрузка эффективного (исполнительного) адреса (окончательного смещения) в регистр общего назначения, используемый в качестве указателя (это та же команда MOV);

− NOP: нет операции;

− UD: неопределенная команда (генерирует соответствующее исключение);

− XLAT/XLATB: табличное преобразование;

− CPUID: идентификация процессора;

− MOVBE: передача данных после перестановки байтов (преобразование данных с little endian order в big endian order или наоборот);

− RDRAND (Read Random Number): чтение в регистр аппаратно сгенерированного случайного числа.

Более подробную информацию о командах процессора можно получить в технической документации Intel [57] или книгах по ассемблеру. Хорошее описание основных команд процессоров x86 с примерами программ дано в книге Зубкова Assembler [19].

3.4. СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В МИКРОПРОЦЕССОРАХ

Конвейерная обработка

Процессор выполняет команды за несколько этапов, которые можно выделять с различной степенью детализации: выборка, исполнение; выборка, дешифрация, исполнение и т. д. Во временном аспекте цикл команды состоит из машинных циклов (циклов шины, транзакций), а машинный цикл — из тактов. Сущностью конвейерной обработки является то, что эти этапы выполняются одновременно, но для разных команд. Для этого необходимо, чтобы на каждом этапе были задействованы свои аппаратные средства, работающие в определенной степени независимо друг от друга. Благодаря этому за один такт может выполняться одна команда, а при наличии нескольких конвейеров — и более, как например, в процессоре Pentium и последующих.

Конвейерная обработка (pipeline processing) в ЭВМ — это выполнение команд за несколько этапов (стадий, ступеней), на каждом из которых используются разные узлы процессора, так что в течение одного и того же такта может выполняться несколько команд, находящихся на разных этапах. При завершении первого этапа команда переходит на второй этап, на котором используются другие аппаратные средства. Средства, используемые на первом этапе, оказываются свободными для начала выполнения следующей команды. Таким образом, как только очередная команда переходит на следующий этап, начинает выполняться новая команда. Одновременно выполняется столько команд, сколько стадий в конвейере. Это, конечно, требует дополнительного управляющего устройства. Конвейерная архитектура присуща RISC-процессорам. Стадий конвейера может быть разное количество. Они выделяются в соответствии с действиями, необходимыми для выполнения команды. Минимальное количество стадий — две: выборка и исполнение. Число стадий в конвейе-

рах процессоров x86 сначала росло, но при переходе к микроархитектуре Core снизилось:

− i486 — 5;

− Pentium — 5;

− Pentium MMX — 6;

− микроархитектура P6 — 12;

− микроархитектура NetBurst — 20;

− микроархитектура Core — 14.

При увеличении количества стадий, конечно, увеличивается энергопотребление, поэтому его надо снижать за счет других факторов.

В семействе x86 есть специфическое семейство процессоров Atom с одноименной микроархитектурой. Они разрабатывались как раз по критерию минимального энергопотребления для использования в нетбуках (маленьких ноутбуках), планшетах, смартфонах и встраиваемых системах, но в мобильном сегменте не получили широкого распространения, так как по производительности они проигрывали процессорам ARM, которые и доминируют в мобильном секторе. Процессоры Atom являются 64-разрядными, т. е. относятся к Intel 64. Конвейер Atom имеет 16 стадий, но он без переупорядочивания команд, поэтому ниже и энергопотребление, и производительность.

Как уже говорилось выше (см. п. 3.2), команды процессора Pentium проходят при своем выполнении пять, а Pentium MMX — шесть стадий конвейера (рис. 3.25):

− PF (Prefetch) — предвыборка;

− F (Fetch) — выборка команды (только для Pentium MMX);

− D1 (Decode) —декодирование команды;

− D2 — генерация адресов операндов;

− EX (Execute) — исполнение, обращение к АЛУ и кэшу данных;

− WB (Writeback) — обратная запись, сохранение результатов и обновление состояния.

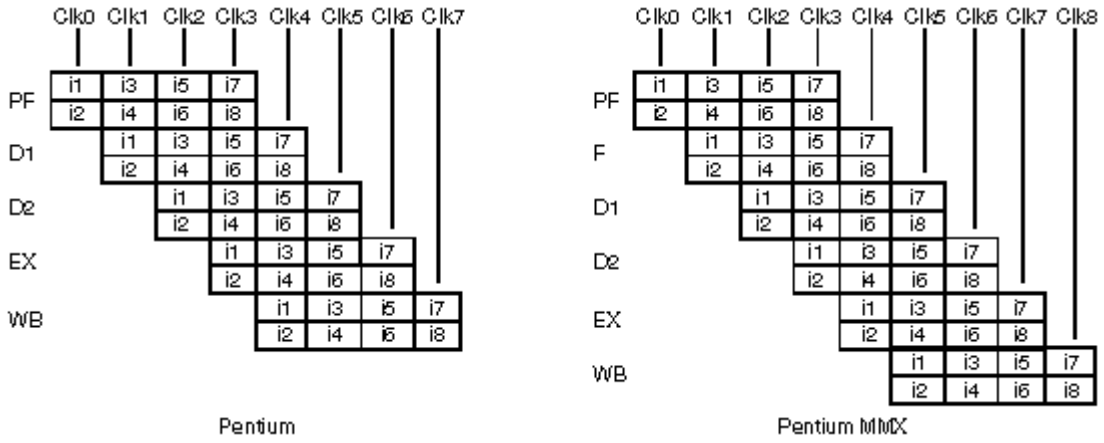


Рис. 3.25. Конвейерная обработка в процессоре Pentium

На одной стадии в двух конвейерах одновременно (в одном такте) находятся две целочисленные команды, а в целом в конвейерах на всех стадиях 10 целочисленных команд у процессора Pentium и 12 — у процессора Pentium MMX. Таким образом, за один такт оба процессора выполняют по две целочисленные команды. На стадии предвыборки команды выбираются из кэша кода или из памяти. На этой стадии участвуют два буфера предвыборки вместе с буфером адресов переходов BTB. Каждый буфер предвыборки имеет размер строки кэша 32 байта. Это позволяет одному буферу предвыборки выбирать команды последовательно, а другому — в соответствии с предсказаниями BTB. Причем эти роли закреплены не жестко, а могут чередоваться. Команды выбираются первым буфером последовательно до тех пор, пока не будет выбрана команда перехода. После этого BTB предсказывает, произойдет переход или нет. Если предсказывается отсутствие перехода, выборка продолжает осуществляться последовательно (линейно). Если предсказывается переход, активизируется второй буфер и предвыборка начинает осуществляться им, как будто переход произошел. Если переход предсказан неверно, то после обнаружения этого конвейеры сбрасываются и выборка начинается заново.

В Pentium MMX имеются четыре 16-байтных буфера предвыборки, которые могут генерировать четыре потока команд, основанных на предсказаниях BTB.

На стадии выборки осуществляется декодирование префиксов и длины команды.

Pentium MMX имеет буфер команд типа FIFO (First In, First Out) между стадиями F и D1. Этот буфер является прозрачным и не вносит задержки в случае, если он пустой. В каждом такте в него могут быть помещены две команды. Когда средняя скорость выполнения меньше, чем две команды за такт, буфер команд обычно заполнен.

На стадии D1 два параллельных декодера декодируют две очередные команды. В Pentium требуется дополнительный такт для декодирования префиксов команды.

На стадии D2 вычисляются (генерируются) адреса операндов, находящихся в памяти.

На стадии выполнения в работу вступает АЛУ. При необходимости осуществляется доступ к кэшу данных, что требует более одного такта для прохождения этой стадии. На этой стадии все команды u-конвейера и все команды v-конвейера, за исключением команд условного перехода, верифицируются для правильного предсказания переходов.

На заключительной стадии (WB) команда может изменить состояние процессора и завершить выполнение. На этой стадии верифицируются команды условного перехода v-конвейера.

При своем прохождении через конвейер команда может останавливаться до выполнения некоторых условий. Оба конвейера начинают и заканчивают стадии D1 и D2 одновременно. Когда команда в одном конвейере останавливается, то команда в другом конвейере останавливается на той же стадии. Таким образом, в стадию выполнения оба конвейера также входят одновременно.

При распределении команд по конвейерам возникает ряд проблем из-за наличия информационных зависимостей между командами по входным-выходным операндам. Существует несколько типов зависимостей и способов их уменьшения (табл. 3.11):

1. Запись после чтения. Первая команда читает данные из регистра или памяти, вторая пишет в этот же регистр или ячейку памяти.

2. Чтение после записи. Обратная ситуация: первая команда записывает результат в регистр или переменную, вторая его читает.

3. Чтение после чтения: две команды имеют в качестве операнда-источника одну и ту же переменную или регистр.

Таблица 3.11

Типы зависимостей между командами и способы их уменьшения

Тип зависимости между командами	**Способ уменьшения зависимости**

Запись после чтения: первая команда читает регистр или переменную, вторая пишет в этот же регистр или переменную:
MOV AX, BX
MOV BX, CX
	Переименование регистров (**register renaming**):
	1. MOV BX, CX : Reg0 ← CX,
	2. MOV AX, BX : AX ← BX,
	3. BX ← Reg0

Чтение после записи: первая команда записывает результат в регистр или переменную, вторая его читает:
MOV Var, BX
MOV AX, Var
	Запоминание промежуточных результатов для дальнейшего использования (**Data bypassing**): записываемый результат передается сразу на второй конвейер

Чтение после чтения: две команды имеют в качестве операнда-источника одну и ту же переменную или регистр:
MOV AX, Var
MOV BX, Var
	Одновременное использование одних данных в обоих конвейерах (**Data forwarding**): процессор выполняет операцию чтения и посылает результат на оба конвейера

Число физических регистров всегда больше, чем программно-доступных. Еще в процессоре i8080 был регистр временного хранения WZ. Пусть имеется зависимость «запись после чтения», т. е. в программе команды идут в такой последовательности:

MOV AX, BX

MOV BX, CX

Пусть эти команды распределены по разным конвейерам и вторая команда выполняется раньше первой. Значение CX записывается во временный регистр Reg0 вместо регистра BX. После того как выполнится первая команда и в регистре AX окажется «правильное» значение, в регистр BX записывается значение из регистра Reg0.

Переименование регистров не может снять зависимость второго типа — чтение после записи:

```
MOV VAR, BX
MOV AX, VAR
```

Полностью ее снять в принципе невозможно. Итак, вторая команда начинает выполняться раньше первой, но ее выполнение обязательно должно приостановиться до появления в переменной VAR нужного значения в результате выполнения первой команды. Однако можно уменьшить время простаивания второго конвейера использованием способа Data bypassing, который заключается в том, что значение, записываемое в переменную VAR первой командой, которая выполняется позже, передается сразу на второй конвейер, что позволяет избежать операции чтения из памяти.

Data forwarding уменьшает зависимость третьего типа — чтение после чтения. Здесь любая команда может выполниться первой без простаивания другой, но для исключения лишней транзакции чтения памяти процессор выполняет операцию чтения и посылает результат на оба конвейера.

При выполнении команд перехода до получения нового адреса конвейер простаивает. Поэтому все адреса переходов, встречающиеся в программе, запоминаются в специальном буфере адресов переходов (Branch Target Buffer, BTB). Процессор с достаточно высокой вероятностью правильного предсказания решает, будет производиться переход или нет, и начинает выполнять команды с предсказанного адреса. Это называется опережающим выполнением (Speculative Execution).

В микроархитектуре P6 два 12-стадийных конвейера. Исполнительная часть содержит четыре обрабатывающих блока: выборки-декодирования (Fetch/Decode), диспетчеризации-выполнения (Dispatch/Execute), завершения (Retire) и пула команд (Instruction Pool) (рис. 3.26).

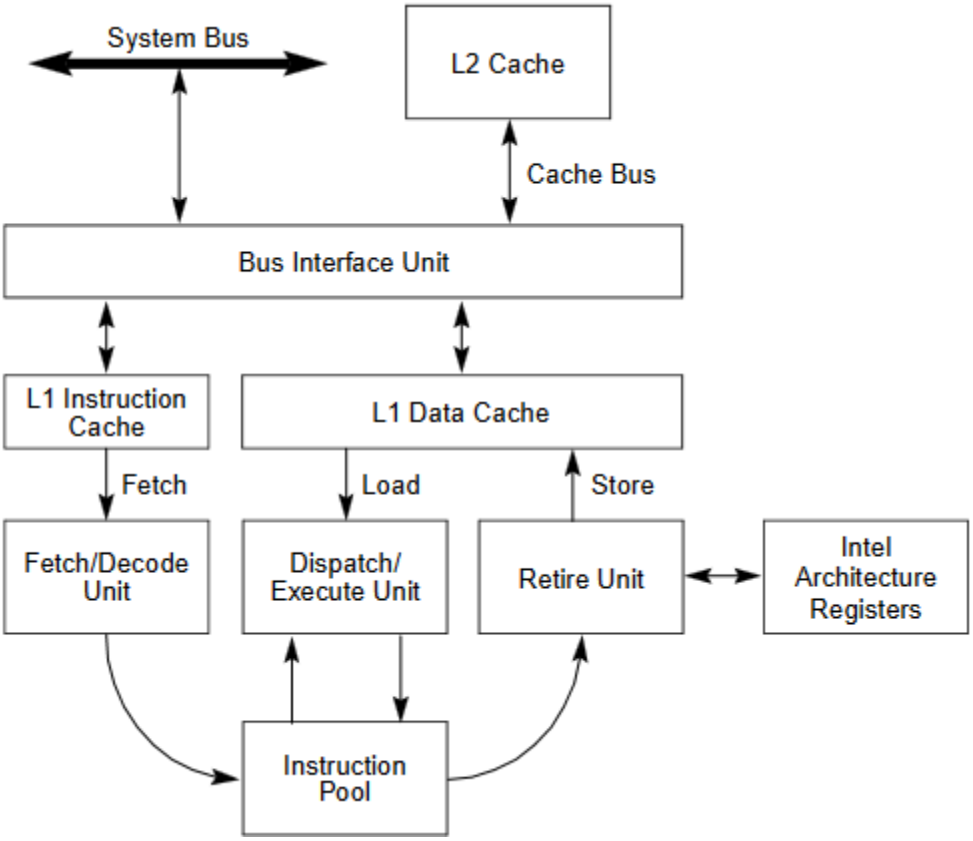


Рис. 3.26. Конвейерная обработка в микроархитектуре P6

Все команды в накопителе имеют определенный статус — «может быть выполнена», «ожидает данные», «выполнена» и т. д.

Гиперконвейер в микроархитектуре NetBurst (см. рис. 3.15) имеет 20 стадий и состоит из следующих блоков:

− Front-end:
  − блок выборки-декодирования;
  − кэш трассировки исполнения с ПЗУ микрокоманд;
− ядро внеочередного исполнения;
− блок завершения.

Основное отличие от конвейера P6 состоит в том, что теперь кэш команд 1-го уровня хранит не команды, а микрокоманды. Блок выборки-декодирования извлекает команды из кэша 2-го уровня, декодирует их и встраивает в последовательности микрокоманд (μops), называемых трассировками (traces), которые помещает в кэш 1-го уровня, который соответственно, называется кэшем трассировки (Trace Cache). В любой момент времени в кэше трассировки содер-

жится несколько трассировок, представляющих предварительно выбранные ветви. В кэше трассировки выполняется поиск команды, которая следует за активной ветвью. Если команда отображается как первая в предварительно выбранной ветви, выборка и декодирование команд из памяти прекращается и предварительно выбранная ветвь становится новым источником команд.

Ядро внеочередного исполнения может переупорядочивать команды таким образом, что если одна микрокоманда задерживается для ожидания данных или разделяемого ресурса, другие микрокоманды, которые должны быть выполнены позже, могут выполняться в обход нее. В процессоре имеется несколько буферов для сглаживания потока микрокоманд. Когда одна часть конвейера испытывает задержку, эта задержка может быть покрыта другими командами, выполняемыми параллельно, или выполнением микрокоманд, которые ранее были помещены в очередь в буфере. В каждом такте ядро исполнения может отправлять до шести микрокоманд на один или несколько из четырех портов. Команды распределяются по портам в зависимости от своего типа (рис. 3.27).

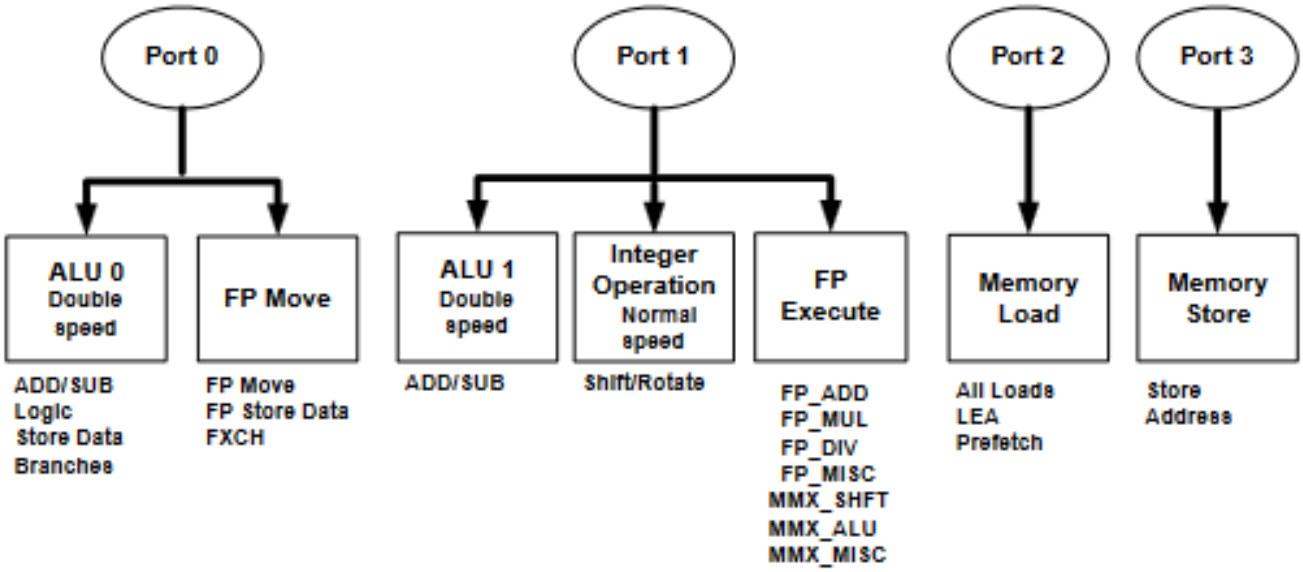


Рис. 3.27. Распределение микроопераций ядром внеочередного исполнения по выходным портам

Блок завершения получает результаты выполненных микрокоманд от ядра исполнения и обрабатывает результаты таким образом, что состояние системы обновляется в соответствии с первоначальным порядком выполнения программы. В процессоре имеется буфер переупорядочения (ROB, Reorder Buffer) — блок, который буферизует завершенные микрооперации и обновляет архитектурное состояние. Когда микрокоманда завершается и записывается ее результат, она удаляется. Блок завершения также отслеживает ветви и отправляет обновленную информацию в BTB. После этого BTB удаляет предварительно выбранные трассировки, которые больше не нужны.

В микроархитектуре Core число стадий конвейера снизилось до 14, блоки остались те же самые, но в их устройстве и работе имеется множество улучшений. Выходных портов ядра внеочередного исполнения — шесть (порты 0–5) (рис. 3.28).

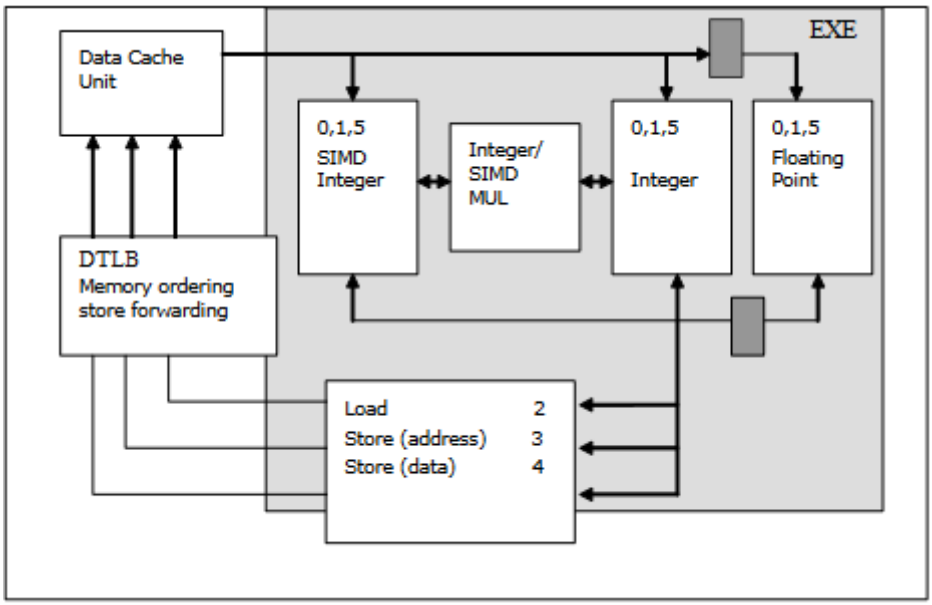


Рис. 3.28. Ядро внеочередного исполнения в микроархитектуре Core

Кроме того, в нем имеется буфер упорядочения памяти (MOB, Memory Ordering Buffer), который позволяет выдавать загрузки и размещения в порты с опережением (спекулятивно) и не по порядку. При этом он гарантирует, что завершенные загрузки и размещения будут иметь правильные данные.

Технология Hyper-Threading

Технология Hyper-Threading (HTT) является реализацией концепции одновременной многопоточности (Simultaneous Multithreading, SMT), которая была предложена еще в прошлом веке. HTT была внедрена в микроархитектуру NetBurst в процессоре Xeon MP Foster и чуть позже появилась в процессоре Pentium 4 Northwood (2002 г.). В микроархитектуре Core она вначале отсутствовала, а позже появилась вновь в процессорах Core i Nehalem. SMT используется также и в других процессорах, например, IBM POWER, начиная с POWER 5 (2004 г.) и AMD x86 с микроархитектурой Zen (2017 г.) и последующих.

Процессор с HTT способен одновременно (параллельно) выполнять два различных потока команд (две нити — threads), т. е. HTT — одна из форм аппаратной поддержки многопоточности.

Архитектурно процессор с технологией HTT представляет собой два логических процессора, каждый из которых имеет свое собственное архитектурное состояние (AS, Architectural State) и контроллер прерываний APIC (рис. 3.29).

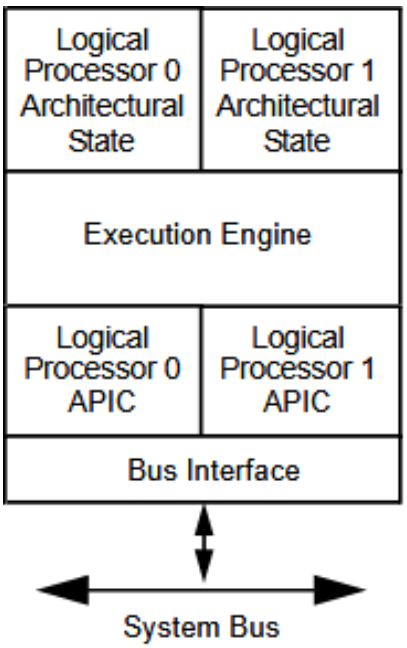


Рис. 3.29. Процессор с технологией HTT

AS — это значения регистров данных, сегментных, управляющих, отладочных и большинства машинно-специфичных регистров. Но в отличие от двухпроцессорной системы логические процессоры разделяют общие ресурсы ядра процессора, которые включают операционное устройство, кэш, устройство шинного интерфейса и др. После включения питания и инициализации каждый логический процессор может быть отдельно остановлен, прерван или запущен для выполнения собственной нити независимо от других логических процессоров. Когда команда HLT выполняется на процессоре с технологией Hyper-Threading, останавливается только тот логический процессор, который выполняет эту команду. Другие логические процессоры продолжают работать, если только каждый из них по отдельности не будет остановлен выполнением команды HLT.

В отличие от обычной многопроцессорной системы (MP), в которой используются два или более отдельных физических процессора, логические процессоры в процессоре с технологией HTT совместно используют основные ресурсы физического процессора: обрабатывающие блоки и интерфейс системной шины (рис. 3.30).

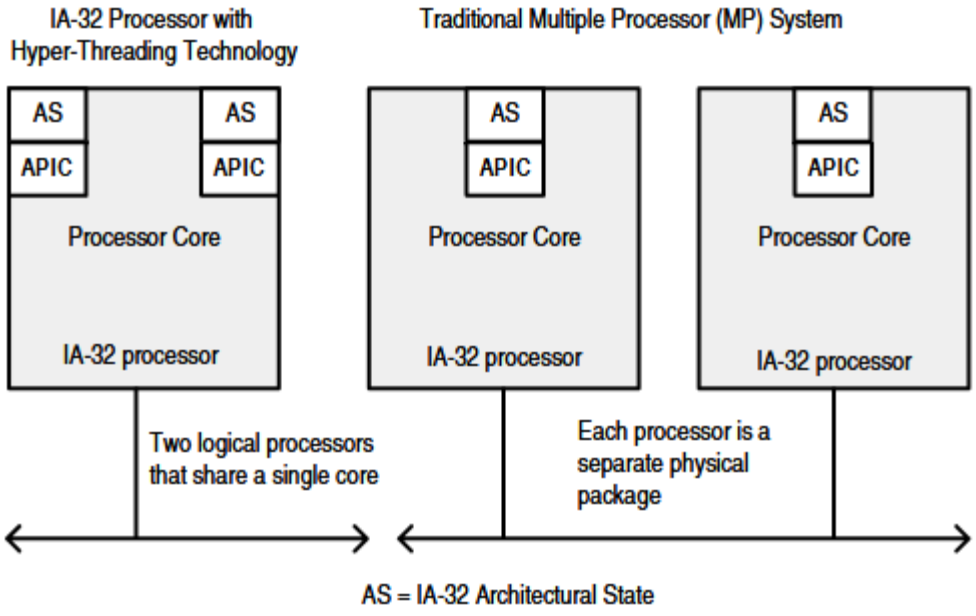


Рис. 3.30. Сравнительная схема двухпроцессорной системы и системы на одном процессоре с технологией HTT

По сравнению с двухпроцессорной системой это дает следующее преимущество: HTT совместно с механизмом переупорядочивания команд позволяет уменьшить число циклов ожидания (простаивания) обрабатывающих блоков конвейеров, соответственно, увеличить их загруженность и в итоге повысить производительность компьютера при использовании многозадачного многопоточного программного обеспечения (рис. 3.31).

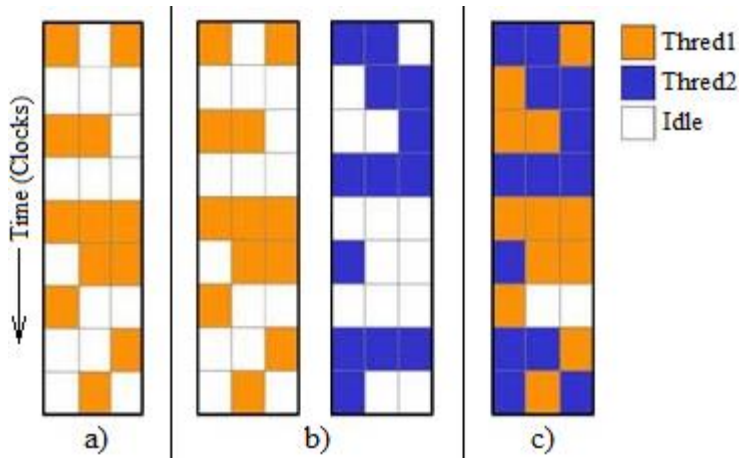


Рис. 3.31. Иллюстрация повышения загруженности обрабатывающих блоков конвейеров в процессорах с HTT:
a — однопроцессорная система без HTT;
b — двухпроцессорная система без HTT;
c — однопроцессорная система с HTT

Элементарная прямоугольная ячейка на рисунке символизирует загрузку одного операционного ресурса в течение одного такта. Если ячейка закрашена, то ресурс загружен. Если нет, то простаивает. Один горизонтальный блок из трех ячеек соответствует одному такту. Три ячейки в одном горизонтальном блоке означают три операционных ресурса, работающих параллельно. Циклы простаивания образуются по причине отсутствия необходимых данных в кэше, ошибочных предсказаний переходов, командных зависимостей в пределах каждой нити. В процессоре с HTT в таких случаях ядро исполнения не будет бездействовать, а передаст управление другому логическому процессору для выполнения его потока команд. Таким образом, пока

один логический процессор вынужденно простаивает, общие вычислительные ресурсы используются вторым логическим процессором.

Технология HTT в многозадачных приложениях дает повышение производительности до 25 %. Величина прироста эффективности, конечно, зависит от того, как в приложении реализована многопоточность. Для использования возможностей технологии HT необходимо, чтобы ее кроме процессора поддерживали чипсет, BIOS и операционная система. Каждое ядро процессора с HTT выглядит для операционной системы и другого программного обеспечения как два процессора.

Инициализация, распределение потоков команд и прерываний по логическим процессорам в системе с процессором с HTT осуществляется так же, как и в обычной многопроцессорной системе.

При инициализации системы один из логических процессоров выбирается в качестве загрузочного (BSP, BootStrap Processor), а остальные логические процессоры обозначаются как прикладные (AP, Application Processor). Каждому логическому процессору присваивается локальный идентификатор — APIC_ID, который записывается в регистр APIC ID логического процессора. При этом, если процессоры с технологией HTT используются в многопроцессорной системе, каждому логическому процессору присваивается уникальный локальный идентификатор APIC_ID благодаря использованию для него иерархической структуры с несколькими полями.

После завершения загрузки операционной системы загрузочный процессор (BSP) продолжает выполнять код операционной системы, в то время как другие логические процессоры переводятся в состояние останова. Чтобы выполнить поток кода (thread) на одном из остановленных логических процессоров, операционная система должна выдать ему межпроцессорное прерывание (IPI, InterProcessor Interrupt). В ответ на IPI остановленный логический процессор выходит из состояния останова и начинает выполнять поток, идентифицированный вектором прерывания, который он получил как часть IPI. Когда все логические процессоры активны, ядро исполнения

(execution engine) выполняет их потоки команд одновременно, и общие ресурсы выделяются активным логическим процессорам по мере необходимости. Для управления выполнением нескольких потоков на активных логических процессорах операционная система может использовать обычные методы симметричной многопроцессорной обработки (SMP, Symmetric MultiProcessing). Например, операционная система может использовать заданный временной интервал или другой механизм балансировки нагрузки для периодического прерывания работы каждого из активных логических процессоров. Операционная система проверяет свою очередь выполнения на наличие потока, ожидающего выполнения, прерывает работу выбранного логического процессора и отправляет ему ожидающий поток.

Внешние прерывания принимаются контроллером прерываний чипсета, который распределяет их по логическим процессорам (рис. 3.32). Каждый логический процессор также может отправлять IPI другим логическим процессорам путем записи необходимых значений в регистр ICR (Interrupt Command Register) своего локального APIC.

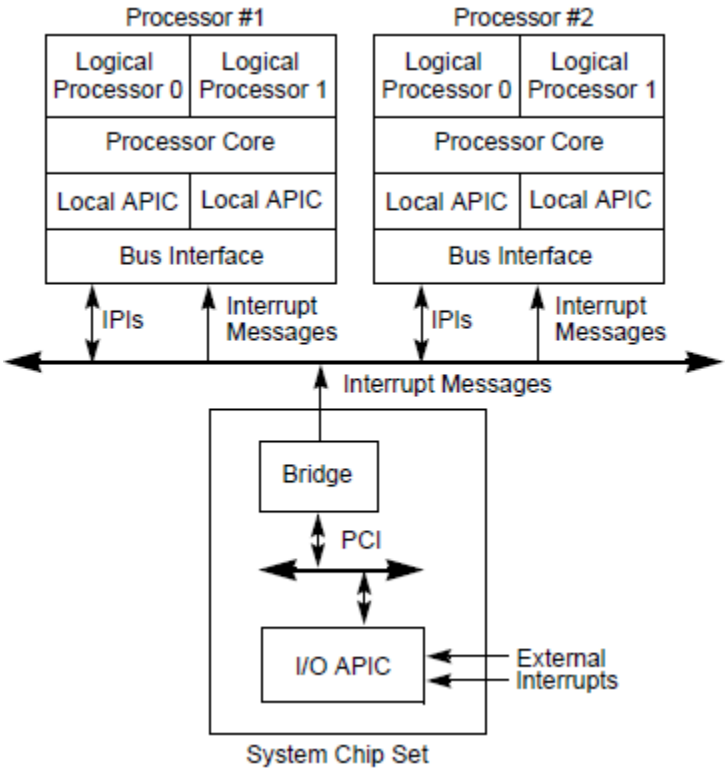


Рис. 3.32. Распределение прерываний в многопроцессорных системах с HTT

Технология виртуализации

Идея виртуализации сама по себе не нова. Первоначально она заключалась в том, чтобы на одной физической вычислительной системе для отдельных пользователей или для разных целей реализовать несколько отдельных вычислительных сред (систем), которые называются виртуальными машинами (рис. 3.33) [12].

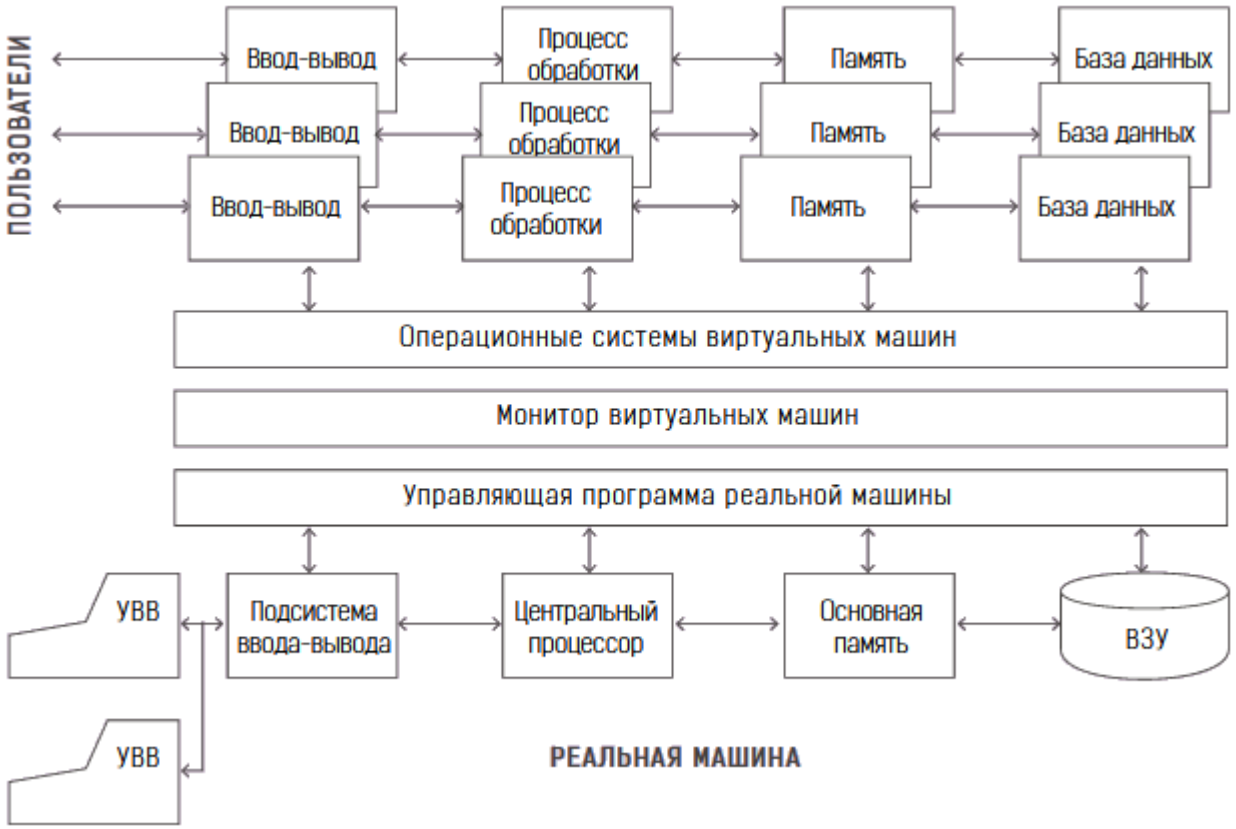


Рис. 3.33. Вычислительная система с технологией виртуализации

Она была реализована еще в начале 70-х гг. прошлого века в операционной системе VM (Virtual Machine) для мейнфреймов компании IBM, а позже в ее отечественном аналоге — СВМ (системе виртуальных машин), операционной системе для ЕС ЭВМ.

Виртуализация в СВМ была последовательной и полной: на виртуальной машине можно было запустить другую копию системы СВМ и т. д. Главным модулем СВМ был монитор виртуальных машин (МВМ). В ОС IBM VM для аналогичного модуля использовалось

название CP (Control Program). MBM управлял аппаратурой реальной ЭВМ и реализовывал набор виртуальных машин с заданной конфигурацией. Теоретически MBM позволял поддерживать до 10 000 виртуальных машин на одной реальной системе, но на практике число одновременно работающих виртуальных машин ограничивалось производительностью ЭВМ и могло достигать нескольких десятков. Позже в ЕС ЭВМ были реализованы средства микропрограммной поддержки СВМ.

В середине 1990-х гг. появился один из самых популярных в настоящее время языков программирования — Java. Программы, написанные на этом языке, компилируются не в машинный код физического процессора, а в так называемый байт-код Java — машинный язык виртуальной машины Java (JVM, Java Virtual Machine). JVM в режиме интерпретации исполняет байт-код, т. е. переводит его уже в машинный код, который выполняется процессором. JVM имеет свои типы данных, регистры и систему команд. Это еще одна разновидность виртуализации. Чуть позже появилась среда исполнения CLR (Common Language Runtime) — виртуальная машина платформы Microsoft.NET Framework. Работа в ней осуществляется таким же образом: программа компилируется в так называемый управляемый код (managed code) на промежуточном языке (MSIL, Microsoft Intermediate Language), а затем CLR преобразует его в машинный код. Есть и другие виртуальные машины подобного типа.

В более широком смысле виртуализация — это предоставление набора вычислительных ресурсов, абстрагированное от аппаратной реализации. Например, виртуальная память — предоставление такого вида аппаратных ресурсов, как диапазон памяти в адресном пространстве процессора, но реально располагающегося не в основной оперативной, а внешней памяти. Реализуется виртуальная память с использованием механизма страничной адресации. В настоящее время виртуализация — очень актуальная тема и популярное современное направление и технология, несмотря на долгую историю. Существует множество информационных источников на эту тему, и тер-

минология в них весьма варьируется. Поэтому необходимо руководствоваться самыми авторитетными источниками: технической документацией и стандартами.

В наиболее полном смысле виртуализация — это группа технологий, основанных на преобразовании формата или параметров программных или сетевых запросов к компьютерным ресурсам с целью обеспечения независимости процессов обработки информации от программной или аппаратной платформы информационной системы [59].

Осуществляется виртуализация:

− программ;

− вычислительных систем;

− вычислительных сетей;

− систем хранения данных;

− памяти.

Виртуализация программного обеспечения — это технология создания изолированной программной среды (контейнера) со специфическим набором компонентов имитируемой операционной системы, обеспечивающим работу отдельных программ.

Виртуализация аппаратного обеспечения или виртуализация вычислительных систем — это технология создания изолированной программной среды (контейнера) со специфическим набором компонентов имитируемого микропрограммного и аппаратного обеспечения, обеспечивающим работу отдельных операционных систем.

Виртуальная машина (ВМ) — это программно реализованная виртуальная вычислительная система, которая состоит из виртуальных устройств обработки, хранения и передачи данных и которая дополнительно может содержать программное обеспечение и пользовательские данные. Виртуальная машина скрывает настоящую реализацию находящихся в ней процессов и объектов от процессов, запущенных вне виртуальной машины.

Виртуализация вычислительных сетей (виртуализация каналов передачи данных) — это технология объединения аппаратных и программных сетевых ресурсов и сетевых функций в едином программно

администрируемом объекте для реализации их логического взаимодействия через дополнительные виртуальные сетевые ресурсы и функции.

Виртуальная вычислительная сеть (виртуальный канал передачи данных) — это вычислительная сеть, содержащая один или более виртуальных сетевых ресурсов и/или функций.

Гипервизор (монитор виртуальных машин) — программа, создающая среду функционирования других программ, в том числе других гипервизоров, путем имитации аппаратных средств вычислительной системы и управления данными средствами. Различают гипервизоры 1-го и 2-го типа. Гипервизор 1-го типа — гипервизор, устанавливаемый непосредственно на аппаратное обеспечение. Гипервизор 2-го типа — гипервизор, устанавливаемый на хостовую операционную систему. Хостовая операционная система — операционная система, установленная на физической вычислительной системе. Физическая вычислительная система называется хостом. Гостевая операционная система — операционная система, установленная на виртуальной машине. Каждая виртуальная машина имеет собственную конфигурацию виртуальных комплектующих (процессор, оперативная память, диски, сетевые адаптеры и т. д.), в которой и происходит загрузка и работа гостевой ОС. Стандартизованные виртуальные комплектующие, которые видит гостевая ОС, обеспечивают совместимость и работоспособность виртуальной машины независимо от комплектующих хоста.

Сервер виртуализации (СВ) — физический сервер, на котором установлен гипервизор (рис. 3.34) [60].

Помимо вышеперечисленных программных средств СВ содержит подсистему управления — компонент, который позволяет настраивать другие компоненты СВ.

Физически виртуальная машина вместе со своим программным обеспечением представляет собой набор файлов:

− дисков виртуальной машины;

– настроек виртуальной машины относительно программного обеспечения (ПО) виртуализации.

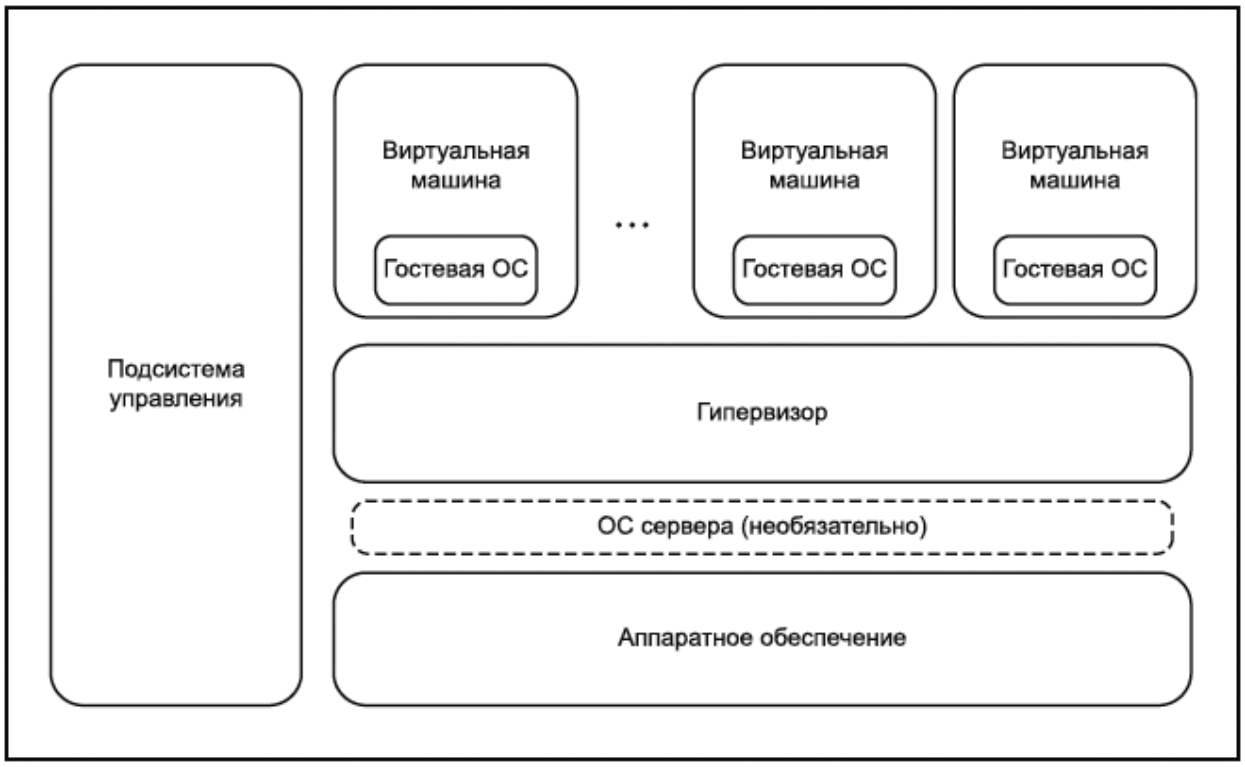


Рис. 3.34. Программно-функциональные компоненты СВ

Это позволяет создавать шаблоны виртуальных машин, перемещать и копировать виртуальные машины. Программные серверы могут перемещаться с физических серверов в виртуальные машины и обратно.

Виртуальные машины полностью изолированы друг от друга и от хостовой ОС. Если внутри виртуальной машины произойдет сбой, это не отразится на работоспособности других виртуальных машин и хоста. Виртуальные машины, как и реальные, могут общаться только через сконфигурированные сетевые соединения.

Различаются полная виртуализация и паравиртуализация. При полной виртуализации гостевая ОС не имеет данных о том, что она запущена в виртуальной машине. Гипервизор предоставляет интерфейс аппаратного обеспечения вычислительного устройства, которое физически доступно для виртуальной машины и для которого в гос-

тевой ОС имеются драйверы, а также полностью эмулирует поведение этого устройства. При паравиртуализации интерфейс аппаратного устройства не предоставляется, гостевые операционные системы программно модифицируются для исполнения в виртуализированной среде и взаимодействуют с гипервизором через предоставляемый им API (Application Programming Interface).

Аппаратная поддержка виртуализации — технология, реализованная в процессорах и чипсетах компьютеров в виде соответствующих архитектурных элементов, в том числе команд, служащая для улучшения технических характеристик информационных систем, построенных с использованием технологий виртуализации, а также повышения безопасности объектов защиты таких систем.

В составе микропроцессора аппаратная поддержка виртуализации впервые была реализована компанией Intel в процессорах Pentium 4 672 и 662 (2005 г.) под названием Intel Virtualization Technology (VT). Аппаратная поддержка виртуализации в рамках VT осуществляется посредством расширений архитектуры процессора, которые называются расширениями виртуальной машины (VMX, Virtual Machine Extensions). Система с процессором с VMX может функционировать как несколько виртуальных систем (виртуальных машин). На каждой виртуальной машине может быть установлено свое программное обеспечение: операционная системы и приложения. В связи с этим вводится понятие платформы. Физическая вычислительная система — это уже не просто компьютер, а база, платформа, на которой работают виртуальные компьютеры.

Расширения VMX поддерживают два основных класса программного обеспечения системы с виртуализацией:

− мониторы виртуальной машины (VMM, Virtual Machine Monitor);

− гостевое программное обеспечение (Guest software).

VMM управляет работой виртуальных машин. Он работает как хост и имеет полный контроль над процессорами и другим оборудо-

ванием платформы: физической памятью, подсистемами прерываний и ввода-вывода.

Гостевое программное обеспечение — это виртуальные машины. Виртуальная машина поддерживает стек, состоящий из операционной системы и прикладного программного обеспечения. Каждая из них работает независимо от других виртуальных машин и использует один и тот же интерфейс для процессора, памяти и ввода-вывода, предоставляемый физической платформой. Программный стек работает так, как если бы он был запущен на платформе без VMM. Программное обеспечение, выполняемое на виртуальной машине, должно работать с пониженными привилегиями, чтобы VMM мог сохранять контроль над ресурсами платформы.

Поддержка виртуализации обеспечивается такой формой работы процессора, как режим VMX. Существует два вида режима VMX: корневой VMX-режим (root-VMX) и некорневой VMX-режим (non-root-VMX). VMM функционирует в режиме root-VMX, а гостевое программное обеспечение — в режиме non-root-VMX. Переходы между этими режимами называются VMX-переходами. Направление перехода определяется относительно гостевого ПО. Переход в режим non-root-VMX называется VM-входом (VM-Entry), переход из режима non-root-VMX в root-VMX — VM-выходом (VM-Exit) (рис. 3.35).

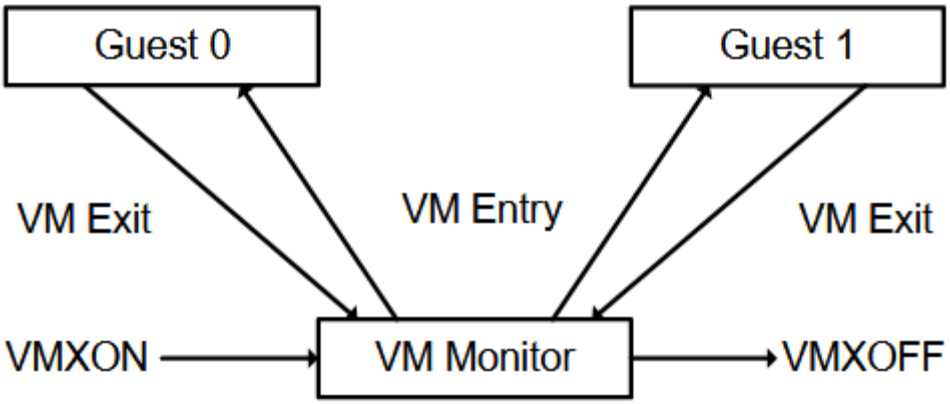


Рис. 3.35. Переходы между VMX-режимами

Вход в режим VMX (root-VMX) осуществляется в следующей последовательности:

− проверка наличия VMX в процессоре с помощью команды CPUID;

− разрешение VMX установкой бита 13 управляющего регистра CR4 VMXE (VMX Enable);

− выделение выравненной 4-килобайтной (или более) области памяти (VMXON-region), которую логический процессор использует для работы в VMX-режиме;

− выполнение команды VMXON с непосредственным операндом, являющимся VMXON-указателем — 64-битным физическим адресом VMXON-region.

Команда VMXON также контролируется машинно-специфичным регистром IA32_FEATURE_CONTROL.

В первых моделях процессоров с VMX вход в VMX-режим мог быть осуществлен только в защищенном режиме с разрешенной страничной адресацией. Впоследствии это ограничение было снято.

После входа в VMX-режим VMM может вводить гостей в виртуальные машины (по одной за раз) посредством VM-входов с использованием команд VMLAUNCH и VMRESUME. VMM восстанавливает управление посредством VM-выхода из виртуальной машины. VM-выход передает управление в точку входа, указанную VMM. VMM может предпринять действия, соответствующие причине выхода из виртуальной машины, а затем вернуться к виртуальной машине, используя VM-вход. VMM может выключить себя, тем самым осуществив выход из режима VMX, выполнив команду VMXOFF. Бит CR4.VMXE может быть сброшен только после выполнения команды VMXOFF, т. е. вне VMX-режима.

Поведение процессора в режиме root-VMX во многом такое же, как и вне режима VMX, за исключением того, что доступна новая группа команд (команды VMX) и множество значений, которые могут быть загружены в определенные управляющие регистры, ограничено.

Поведение процессора в режиме non-root-VMX изменено. В этом режиме некоторые команды, например, VMCALL, и события вызывают VM-выходы. Таким образом, функциональность программного обеспечения в режиме non-root-VMX ограничена. Именно это ограничение и позволяет VMM сохранять контроль над ресурсами процессора.

Не существует видимого программным обеспечением бита, значение которого указывает, находится ли логический процессор в режиме non-root-VMX. Это позволяет VMM предотвратить определение гостевым программным обеспечением того, что оно работает на виртуальной машине.

Поскольку режим VMX накладывает ограничения даже на программное обеспечение, работающее с наивысшим (нулевым) текущим уровнем привилегий, гостевое программное обеспечение может работать с тем уровнем привилегий, для которого оно изначально было разработано. Эта возможность упрощает разработку VMM.

Работа в режиме non-root-VMX и VMX-переходы управляются структурой данных, называемой структурой управления виртуальной машиной (VMCS, Virtual-Machine Control Structure). Доступ к VMCS осуществляется с помощью указателя VMCS, значение которого представляет собой 64-битный адрес VMCS. Указатель VMCS является элементом архитектурного состояния процессора, таким образом, у каждого логического процессора свой указатель VMCS. Указатель VMCS читается и записывается с использованием команд VMPTRST (VM Pointer Store) и VMPTRLD (VM Pointer Load). VMM конфигурирует VMCS с помощью команд VMREAD, VMWRITE и VMCLEAR. VMM может использовать разные VMCS для каждой виртуальной машины, которые он создает и поддерживает. Более того, для виртуальной машины с несколькими логическими процессорами (виртуальными процессорами) VMM может использовать разные VMCS для каждого виртуального процессора.

Технология VT постоянно совершенствуется, в VMX появляются дополнительные функции, доступность которых сообщается про-

граммному обеспечению с помощью соответствующего набора машинно-специфичных регистров. Например, в процессоре Core i7 имеется расширенная таблица страниц (EPT, Extended Page Table) для снижения числа VMX-переходов. При использовании EPT определенные физические адреса обрабатываются как гостевые физические адреса и не используются для прямого доступа к памяти. Гостевые физические адреса преобразуются путем обхода набора структур EPT для получения физических адресов, которые используются для доступа к памяти.

В целом технология виртуализации позволяет более эффективно использовать имеющиеся программные и аппаратные ресурсы, что предоставляет следующие возможности:

− создание сложных вычислительных систем, состоящих из множества физических и виртуальных машин с разной конфигурацией и различными операционными системами и разработанным для них программным обеспечением;

− создание полностью изолированной среды для каждого пользователя;

− создание необходимой среды для разработки, отладки и тестирования программного обеспечения;

− резервирование основного сервера виртуальным;

− быстрый перенос сервера или рабочего места с одного компьютера на другой;

− загрузка виртуальных машин по локальной сети с сервера;

− перемещение виртуальных машин одного физического хоста на другой без необходимости реконфигурации, что дает гибкость, быстроту и удобство при создании и развертывании новых серверов как на имеющихся, так и вновь приобретаемых аппаратных средствах;

− сохранение и загрузка состояния виртуальных машин, откат к предыдущим состояниям, в том числе первоначальному, «чистому».

Внимательный читатель мог заметить, что на рис. 2.12 (см. п. 2.3) диспетчер устройств показывает конфигурацию компьютера с

чипсетом Intel 440BX и процессором Xeon, хотя данный чипсет поддерживает только процессоры Pentium II, Pentium III и Celeron. При этом на рисунке приведен реальный скриншот. Такое сочетание оказалось возможным, потому что это была виртуальная машина.

В настоящее время существует множество программных средств виртуализации разных компаний. Наиболее популярным является ПО виртуализации компании VMware: VMware Server, VMware Workstation и т. д. Они используют технологию Intel VT.

Bochs — это тоже средство виртуализации. Bochs не использует аппаратную поддержку виртуализации в процессорах, поэтому работает медленнее, чем другие средства.

Использование виртуализации очень полезно, а зачастую и незаменимо в учебных целях. Можно использовать множество виртуальных машин с различными операционными системами и необходимыми программными средствами на одном физическом компьютере. Для выполнения практических заданий данного учебного пособия, связанных с разработкой простых программ под DOS, можно использовать Bochs, VMware или другие средства виртуализации с виртуальными машинами с необходимой средой: FreeDOS, TASM и т. п.

3.5. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 3

Задание 1. Характеристики процессора. Оценка производительности процессора

Подробную информацию о характеристиках и производительности процессора можно получить с помощью информационно-диагностических утилит, которые были описаны в п. 2.4.

CPU-Z выдает подробную информацию о процессоре и позволяет оценить его производительность в условных единицах в процентном или абсолютном отношении (рис. 3.36–3.38).

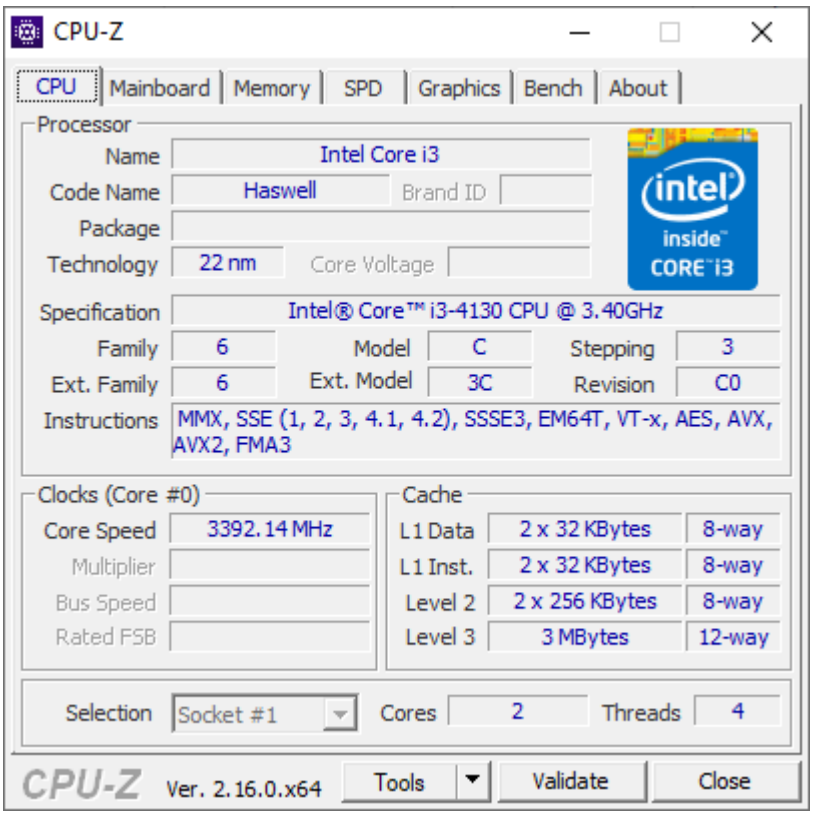


Рис. 3.36. Характеристики процессора в программе CPU-Z

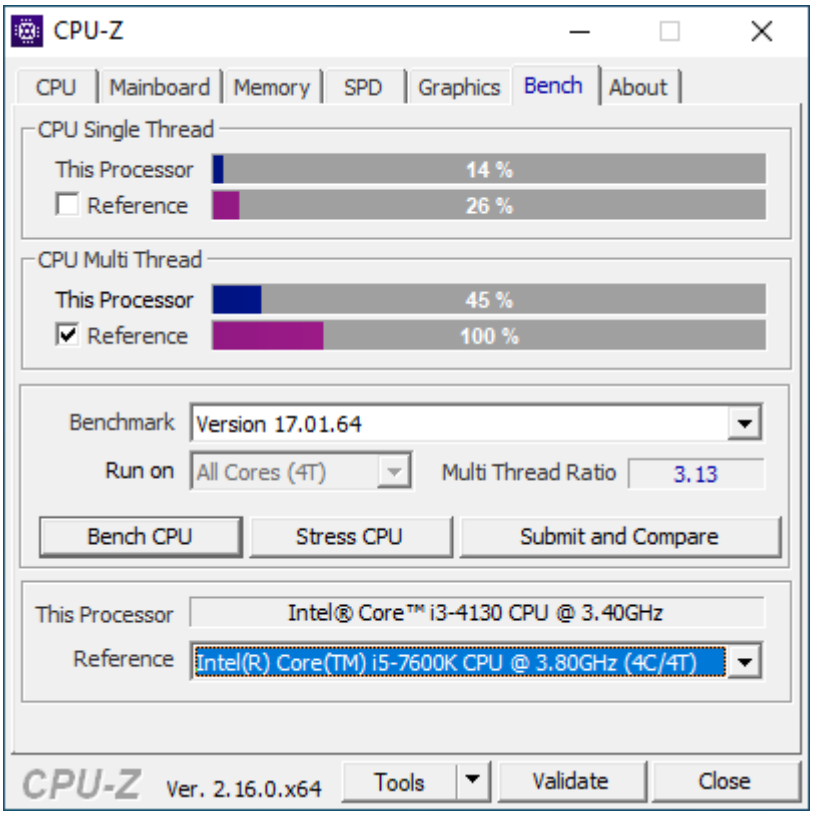


Рис. 3.37. Оценка производительности процессора в сравнении с эталонным в процентном отношении в программе CPU-Z

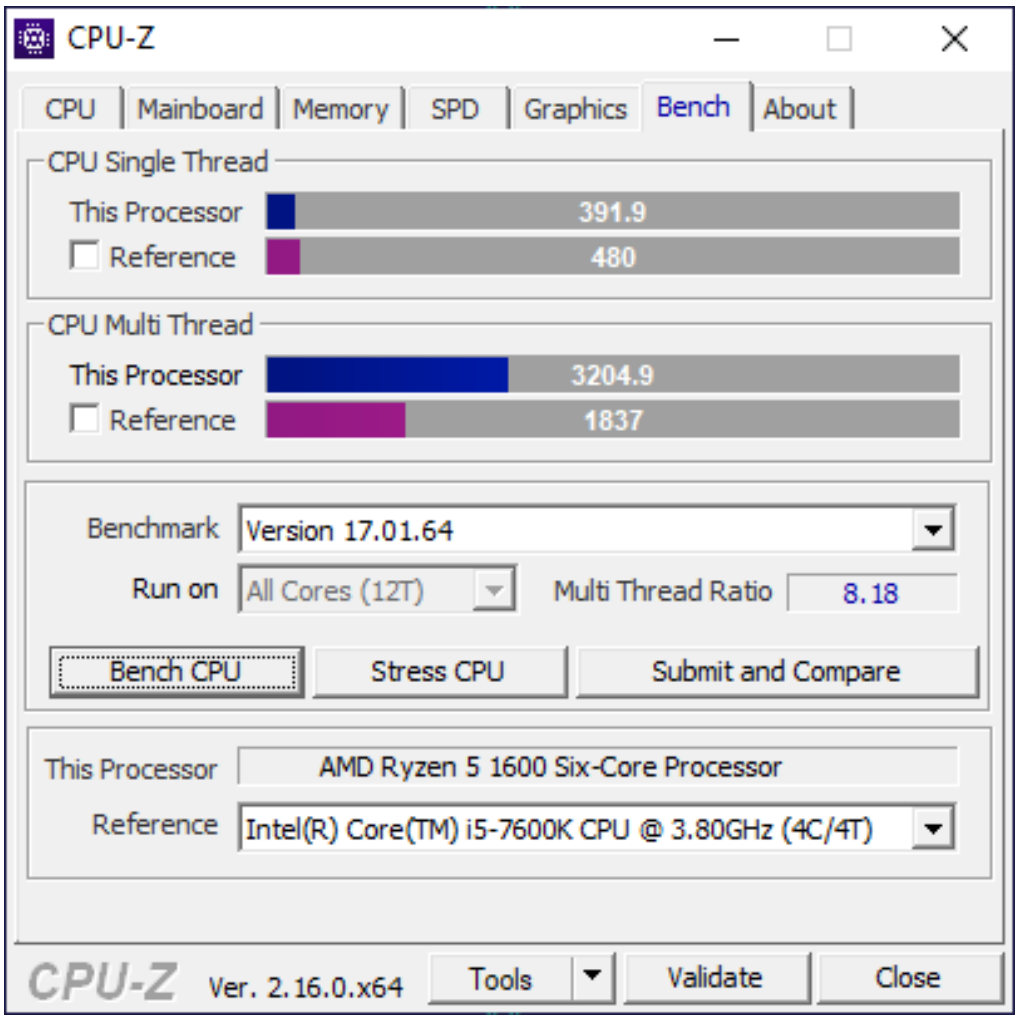


Рис. 3.38. Оценка производительности процессора в сравнении с эталонным в абсолютных единицах в программе CPU-Z

Здесь видно, что производительность процессора AMD Ryzen 5 1600 для одного потока ниже, чем эталонного процессора Intel Core i5-7600K. Но для нескольких потоков она выше, поскольку число логических процессоров у AMD Ryzen 5 1600 больше.

AIDA64 тоже позволяет получить детальную информацию о процессоре, но, в отличие от CPU-Z, имеет значительно большую функциональность по оценке производительности, причем не только центрального, но и графического процессора (рис. 3.39, 3.40).



Рис. 3.39. Оценка производительности
центрального процессора в AIDA64

Как видно из рисунков 3.39 и 3.40, производительность процессора оценивается по ряду различных показателей. При этом очень удобно сравнить производительность CPU и GPU по одинаковым показателям.

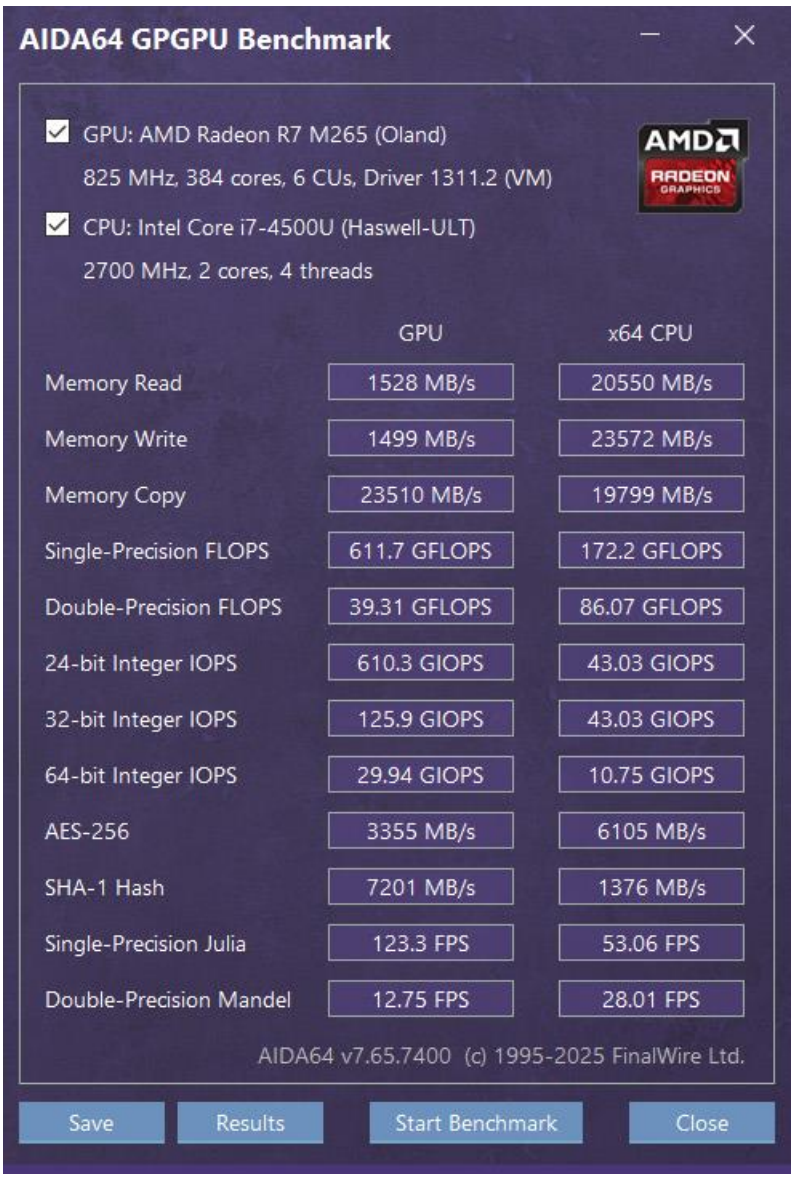


Рис. 3.40. Оценка производительности центрального и графического процессоров в AIDA64

В программе SiSoftware Sandra Lite основные характеристики процессора (модель, количество ядер и потоков, кэш-память, разъем и корпус) выводятся в окне «Информация о системе» (см. рис. 2.30). Здесь нужно обратить внимание на количество ядер (6C — шесть ядер (C — Core)) и количество потоков (12T – двенадцать потоков (T — Thread)). Если потоков в два раза больше, чем ядер, то у процессора имеется технология гиперпоточности (многопоточности) (HTT, Hyper-Treading Technology), которая позволяет одному ядру выполнять два потока команд за счет того, что в нем имеется два ло-

гических процессора. Также видим, что ядра распределены по двум модулям (2M), кэш L2 отдельный для каждого ядра (6x 512 Кб L2), а кэш L3 отдельный для каждого модуля (2x 8 Мб L3).

Для получения дальнейшей информации о процессоре во вкладке «Устройства» выбираем «Процессоры» (рис. 3.41).

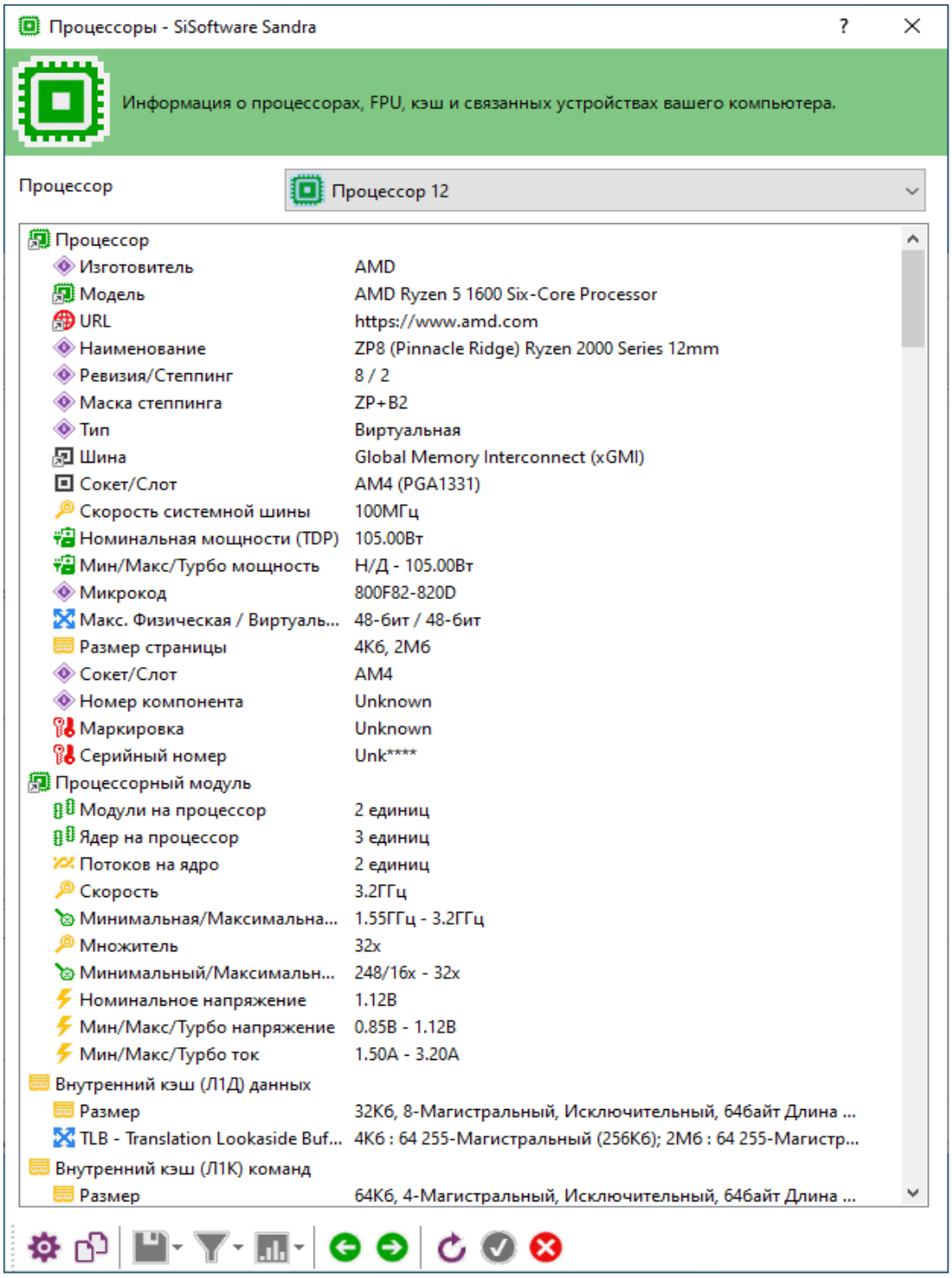


Рис. 3.41. Информация о процессоре

Составляем список основных характеристик процессора:

- модель: AMD Ryzen 5 1600;
- разъем AM4, корпус PGA1331;
- техпроцесс: 12 нм;
- максимальная тактовая частота: 3,2 ГГц;
- номинальная потребляемая мощность (TDP): 105 Вт;
- количество модулей: 2 (2M);
- количество ядер: 6 (6C);
- количество логических процессоров (потоков): 12 (12T);
- размер кэш-памяти данных L1 на одно ядро: 32 Кбайта;
- размер кэш-памяти команд L1 на одно ядро: 64 Кбайта;
- размер кэш-памяти L2 на одно ядро: 512 Кбайт;
- размер кэш-памяти L3 на один модуль: 8 Мбайт.

Для оценки производительности процессора во вкладке «Эталонные тесты» SiSoftware Sandra имеется множество тестов (см. рис. 2.29). Например, для оценки производительности процессора при обычных математических вычислениях с целыми и дробными числами выбираем «Арифметический тест процессора» и получаем следующий результат (рис. 3.42).

Этот пример приведен уже для другого компьютера. Общую производительность процессора по арифметическому тесту Sandra оценивает в 47,96 ГОПС (гигаопераций в секунду). Для сравнения представлены значения производительности эталонного процессора, в качестве которого был выбран Intel Core i7-6700K. Рассчитаем число операций, выполняемых процессором за один такт, для этого поделим число операций в секунду на тактовую частоту процессора:

47,96 ГОПС/3,4 ГГц ≈ 14,1 ОПТ,

где ОПТ – операций за такт.

Процессор двухядерный, таким образом, одно ядро выполняет более 7 операций за один такт. Для процессоров с микроархитектурой Core в соответствии с документацией Intel теоретический максимум — четыре команды за один такт на ядро [57]. Это говорит о том, что в данном случае понятия «команда» и «операция» не являются синонимами. Об этом также свидетельствует присутствие векторных

расширений AVX2 в названии тестов. Понятие операции трактуется с учетом разрядности операндов и числа компонентов вектора, т.е количества скалярных операндов данной разрядности в векторной команде.

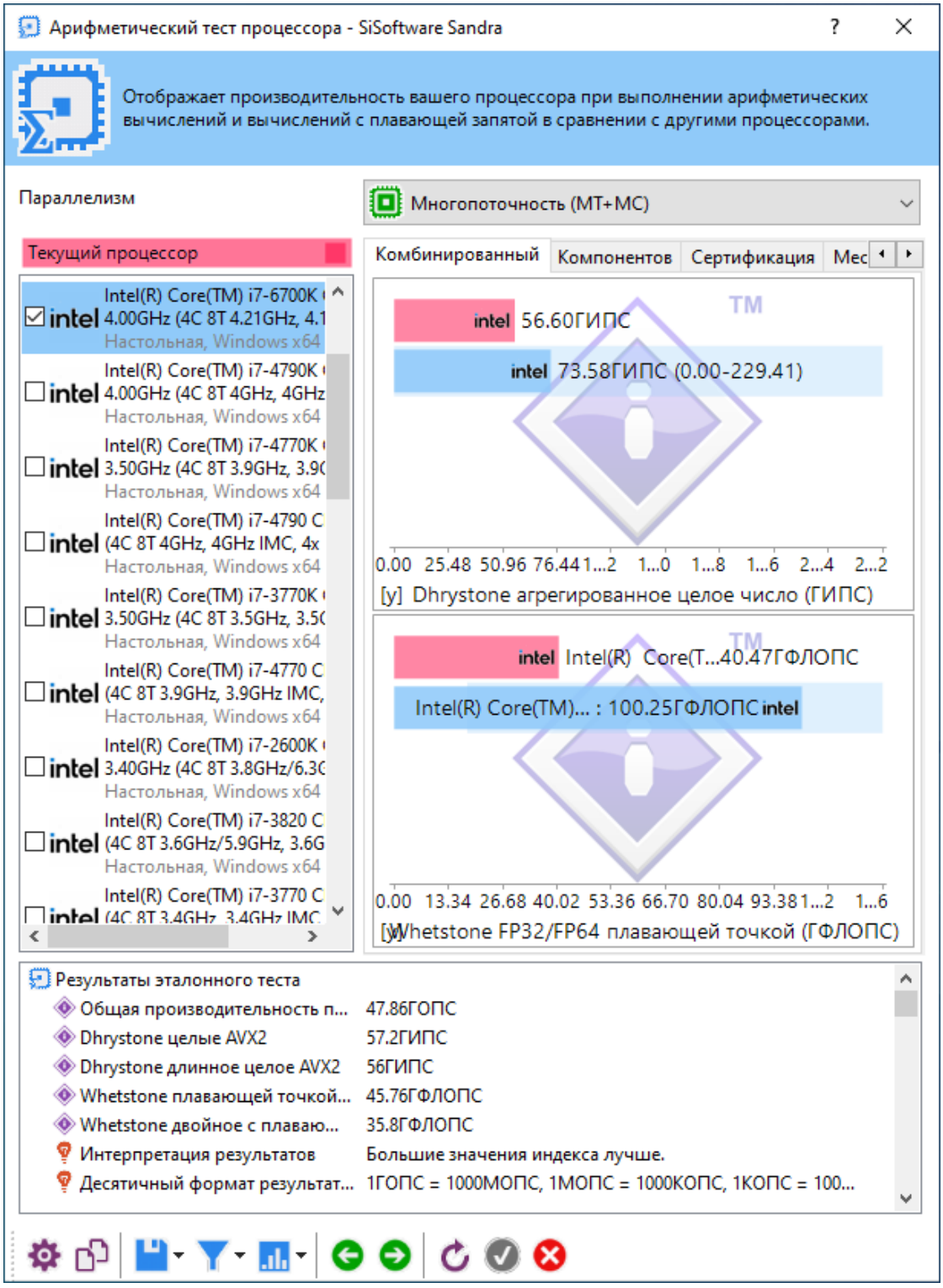


Рис. 3.42. Арифметический тест процессора

Для комплексной оценки производительности процессора используется «Общая оценка процессора» (рис. 3.43).

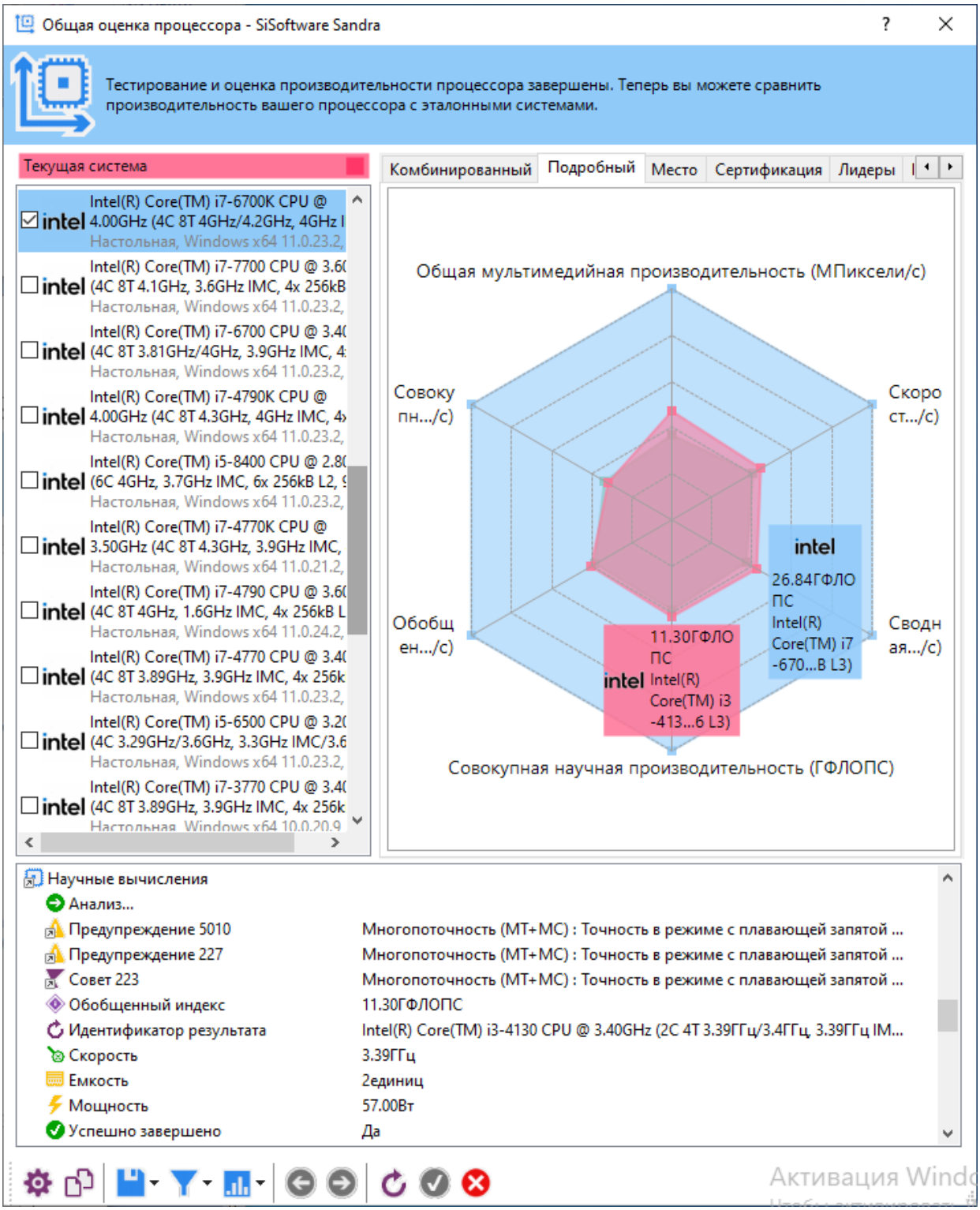


Рис. 3.43. Общая оценка процессора

Здесь так же для визуализации результатов сравнения с эталонными процессорами используется лепестковая диаграмма (см. задание 2 раздела 2). Значение обобщенного индекса производительности составляет 11,3 ГФЛОПС. Число операций с плавающей точкой, выполняемых за один такт:

11,3 ГФЛОПС /3,4 ГГц ≈ 3,3 ОПТ.

В данном задании необходимо с использованием одной или нескольких информационно-диагностических утилит получить подробную информацию о характеристиках процессора и провести тестирование его производительности. Провести оценку полученных значений технико-эксплуатационных показателей посредством расчетов по вышеприведенным примерам. Порядок выполнения задания проиллюстрировать скриншотами.

Задание 2. Система команд процессора. Подпрограммы и стек. Способы адресации и флаги

На языке ассемблера написать программу для вычисления значения заданного выражения с несколькими переменными. Команды вычисления значения выражения должны быть оформлены в виде подпрограммы. Переменные с исходными данными и результатом должны быть массивами из нескольких значений. Основная программа в цикле передает значения переменных подпрограмме через стек. Подпрограмма возвращает результат в регистре. Основная программа записывает его из регистра в результирующую переменную.

Исследовать выполнение программы в отладчике. Акцентировать внимание на командах вызова подпрограммы и возврата из подпрограммы (CALL и RET), помещения в стек (PUSH) и извлечения из стека (POP).

Определить способы адресации, использующиеся в командах программы. Результаты оформить в виде таблицы (табл. 3.12). Подобные команды в таблице повторять не следует.

Таблица 3.12

Способы адресации

№	Команда	Действия	Операнды	Адресация

Определить влияние команд на регистр флагов. Результаты оформить в виде таблицы (табл. 3.13).

Таблица 3.13

Влияние команд на флаги

№	Команда	Значения входных операндов	Значение результата	Значения флагов

Пример выполнения задания

Программа вычисляет значение суммы значений двух переменных A и B.

```
;сумма S=A+B вычисляется в подпрограмме
stck segment stack        ;начало сегмента стека
        dw 10h dup (?)    ;стек имеет размер 16 слов
stck ends                 ;конец сегмента стека
;в сегменте indat – исходные данные: переменные (массивы) A и B
indat segment             ;начало сегмента indat
        dw ?
        A dw 0FFFh, 7FFFh, 8000h
        B dw 1h, 2h, 8001h
indat ends                ;конец сегмента indat
;в сегменте outdat – результат – переменная (массив) S
outdat segment            ;начало сегмента outdat
        S dw 3 dup (?)
outdat ends               ;конец сегмента outdat
code segment              ;начало сегмента code
;сегментные регистры будут указывать на соответствующие сегменты:
        assume cs:code, ds:indat, es:outdat, ss:stck
;подпрограмма
```

```
calc proc               ;объявление начала подпрограммы
        push bp         ;регистры, изменяемые подпрограммой,
                        ;нужно сохранять (в стеке)
        mov bp, sp      ;направление указателя базы на верхушку стека
        add bp, 6       ;направление указателя базы на A в стеке
        mov ax, [bp]    ;передача в AX переменной A из стека
        add ax, [bp-2]  ;AX=A+B
        pop bp          ;восстановление значения BP
        ret 2*2         ;возврат из подпрограммы с очисткой стека
calc endp               ;конец подпрограммы
;основная программа
begin: mov dx, indat    ;точка входа в программу, инициализация
        mov ds, dx      ;сегментного регистра ds
        mov dx, outdat  ;инициализация сегментного
        mov es, dx      ;регистра es
        mov cx, 3       ;инициализация счетчика цикла (в цикле 3 витка)
        xor si, si      ;инициализация si – указателя элементов массива
lp:     push A[si]      ;начало цикла, в стек помещается очередное A:
                        ;SP=SP-2, SS:[SP]=A[SI]
        push B[si]      ;в стек помещается очередное B
                        ;SP=SP-2, SS:[SP]=B[SI]
        call calc       ;вызывается подпрограмма
        mov S[si], ax   ;результат сложения записывается в S
        inc si          ;указатель элементов массива увеличивается на 2
        inc si          ;для перехода к следующему элементу массива
        loop lp         ;команда цикла, выполнит 3 витка цикла
        mov ax, 4c00h   ;запись в ax номера функции завершения программы
        int 21h         ;вызов прерывания DOS для завершения программы
code ends               ;конец сегмента code
        end begin   ;конец программы (команда с меткой begin – точка входа)
```

Скомпилируем программу с параметром генерации листинга l. Листинг программы:

```
 1 0000                                 stck segment stack
 2 0000      10*(????)                      dw 10h dup (?)
 3 0020                                 stck ends
 4 0000                                 indat segment
 5 0000      ????                           dw ?
 6 0002      0FFF 7FFF 8000                 A dw 0FFFh, 7FFFh, 8000h
```

```
 7 0008    0001 0002 8001          B dw 1h, 2h, 8001h
 8 000E                        indat ends
 9 0000                        outdat segment
10 0000    03*(????)               S dw 3 dup (?)
11 0006                        outdat ends
12 0000                        code segment
13                             assume cs:code, ds:indat, es:outdat, ss:stck
14 0000                        calc proc
15 0000    55                      push bp
16 0001    8B EC                   mov bp, sp
17 0003    83 C5 06                add bp, 6
18 0006    8B 46 00                mov ax, [bp]
19 0009    03 46 FE                add ax, [bp-2]
20 000C    5D                      pop bp
21 000D    C2 0004                 ret 2*2
22 0010                        calc endp
23 0010    BA 0000s            begin:  mov dx, indat
24 0013    8E DA                   mov ds, dx
25 0015    BA 0000s                mov dx, outdat
26 0018    8E C2                   mov es, dx
27 001A    B9 0003                 mov cx, 3
28 001D    33 F6                   xor si, si
29 001F    FF B4 0002r         lp: push A[si]
30 0023    FF B4 0008r             push B[si]
31 0027    E8 FFD6                 call calc
32 002A    26: 89 84 0000r         mov S[si], ax
33 002F    46                      inc si
34 0030    46                      inc si
35 0031    E2 EC                   loop lp
36 0033    B8 4C00                 mov ax, 4c00h
37 0036    CD 21                   int 21h
38 0038                        code ends
39                                 end begin
```

Для более удобного восприятия комментарии из листинга удалены. Указатель базы BP сохранять перед использованием подпрограммой не обязательно, т. к. при стандартном его использовании любая подпрограмма инициализирует BP текущим после вызова под-

программы значением указателя стека: mov bp, sp. Поэтому предыдущее его значение значения не имеет (получился каламбур)). Но в этом примере использование команд push bp и pop bp полезно для пояснения следующих важных моментов. Обратим внимание, что команды push bp, pop bp (так же, как и другие команды помещения и извлечения из стека регистров общего назначения) имеют длину один байт. В ассемблерной записи в них присутствует один аргумент — регистр. Код его имени содержится в единственном байте кода операции. Но на самом деле обе эти команды имеют два операнда. В любой команде передачи данных имеется два операнда: операнд-приемник (первый операнд) и операнд-источник (второй операнд). Основное действие, выполняемое процессором по команде push bp:

ss:[sp] ← bp.

Значение регистра bp помещается в стек по адресу ss:sp (указатель стека sp перед очередным помещением декрементируется на величину помещаемого значения в байтах).

По команде pop bp выполняется обратная операция — извлечение из стека значения в регистр bp:

bp ← ss:[sp].

Указатель стека инкрементируется после извлечения.

Таким образом, в любых командах работы со стеком присутствует косвенная адресация. В командах push bp и pop bp помимо косвенной присутствует регистровая адресация. В командах помещения в стек переменных A и B присутствует прямая адресация.

Проанализировав подобным образом команды программы с точки зрения их операндов и способов их указания, заполним таблицу «Способы адресации», начиная с точки входа в программу (табл. 3.14).

Таблица 3.14

Способы адресации

№	Команда	Действия	Операнды	Адресация
1	mov dx, indat BA 0000	DX ← indat	DX	регистровая
			indat (0000)	непосредственная
2	mov ds, dx 8E DA	DS ← DX	DS	регистровая
			DX	регистровая
3	mov cx, 3 B9 0003	CX ← 0003	CX	регистровая
			0003	непосредственная
4	xor si, si 33 F6	SI = SI ⊕ SI = 0	SI	регистровая
			SI	регистровая
5	push A[si] FF B4 0002	SP = SP – 2 SS:[SP] ← [A + SI]	SS:[SP]	косвенная
			[A + SI]	A (0002) - прямая
				SI - косвенная
6	call calc E8 FFD6	SP = SP – 2 SS:[SP] ← IP IP = IP + FFD6 = 2A + FFD6 = 1| 0000 = 0000 (calc)	SS:[SP]	косвенная
			IP	неявная
			FFD6	непосредственная
7	push bp 55	SP = SP – 2 SS:[SP] ← BP	SS:[SP]	косвенная
			BP	регистровая
8	mov bp, sp 8B EC	BP ← SP	BP	регистровая
			SP	регистровая
9	add bp, 6 83 C5 06	BP = BP + 6	BP	регистровая
			06	непосредственная
10	mov ax, [bp] 8B 46 00	AX ← SS:[BP + 0]	AX	регистровая
			SS:[BP + 0]	BP - косвенная
				00 - прямая
11	add ax, [bp-2] 03 46 FE	AX = AX + SS:[BP – 2]	AX	регистровая
			SS:[BP – 2]	BP - косвенная
				FE (–2) - прямая
12	pop bp 5D	BP ← SS:[SP] SP = SP + 2	BP	регистровая
			SS:[SP]	косвенная
13	ret 2*2 C2 0004	IP ← SS:[SP] SP = SP + 2 SP = SP + 4	IP	неявная
			SS:[SP]	косвенная
			0004	непосредственная
14	mov S[si], ax 26 89 84 0000	[S + SI] ← AX	S (0000)	прямая
			[SI]	косвенная
			AX	регистровая
15	inc si 46	SI = SI + 1	SI	регистровая
16	loop lp	CX = CX – 1	IP	неявная

	E2 EC	IF CX ≠ 0 IP = IP + EC = = 31 + EC = 1| 1D (lp)	EC	непосредственная
17	mov ax, 4C00h B8 4C00	AX ← 4C00	AX	регистровая
			4C00	непосредственная
18	int 21h CD 21	SP = SP – 2 SS:[SP] ← Flags SP = SP – 2 SS:[SP] ← CS SP = SP – 2 SS:[SP] ← IP CS:IP ← [21 * 4]	SS:[SP]	косвенная
			Flags, CS, IP	неявная
			21	непосредственная

Машинные коды команд в таблице представлены как в листинге, т. е. с прямым порядком расположения байтов в словах.

В команде add bp, 6 шестерка — непосредственный операнд (immediate). В команде add ax, [bp-2] адрес второго операнда (второго слагаемого) складывается из двух частей: одна часть находится в регистре BP, а вторая (FE = –2) — в команде. Но поскольку FE — это часть адреса, то его нельзя рассматривать как непосредственный операнд, а надо считать смещением (displacement), что соответствует понятию прямой адресации.

Указатель команд IP изменяется при чтении каждой команды, увеличиваясь на длину команды в байтах. Но в таблице отражено его изменение только командами передачи управления call, loop и int.

Последние две строки можно не заполнять, поскольку эти команды не имеют отношения собственно к алгоритму программы, а предназначены для завершения ее работы и возврата в операционную систему.

Далее определяем влияние команд на регистр флагов. На флаги влияют арифметические и логические команды. Выполняем программу в отладчике, фиксируем и анализируем значения флагов после выполнения команд (табл. 3.15).

Таблица 3.15

Влияние команд на флаги

№	Команда	Значения входных операндов	Действие и значение результата	Значения флагов					
				C	Z	S	O	P	A
1	xor si, si	SI = 0000 SI = 0000	SI = 0000 ⊕ 0000 = = 0000	0	1	0	0	1	0
2	add bp, 6	BP = 001A immediate = 6	BP = 001A + 6 = 0020	0	0	0	0	0	1
3	add ax, [bp-2]	AX = 0FFF [BP – 2] = 0001	AX = 0FFF + 0001 = = 1000	0	0	0	0	1	1
4	inc si	SI = 0000	SI = 0000 + 1 = 0001	0	0	0	0	0	0
5	inc si	SI = 0001	SI = 0001 + 1 = 0002	0	0	0	0	0	0
6	add ax, [bp-2]	AX = 7FFF [BP – 2] = 0002	AX = 7FFF + 0002 = = 8001	0	0	1	1	0	1
7	inc si	SI = 0002	SI = 0002 + 1 = 0003	0	0	0	0	1	0
8	inc si	SI = 0003	SI = 0003 + 1 = 0004	0	0	0	0	0	0
9	add ax, [bp-2]	AX = 8000 [BP – 2] = 8001	AX = 8000 + 8001 = = 0001	1	0	0	1	0	0
10	inc si	SI = 0004	SI = 0004 + 1 = 0005	1	0	0	0	1	0
11	inc si	SI = 0005	SI = 0005 + 1 = 0006	1	0	0	0	1	0

Команда xor si, si обнуляет регистр SI. Поэтому значение флага Z равно единице. В результате ноль единичек, ноль считается четным числом, поэтому значение флага четности P так же равно единице.

При выполнении команды add bp, 6 произошел перенос из младшей тетрады младшего байта в старшую тетраду. Поэтому установился флаг дополнительного переноса A. Остальные флаги сбросились.

При сложении чисел 0FFF и 0001 по команде add ax, [bp-2] (строка 3) в результате получилось число 1000. Число единичек в результате нечетное, но флаг четности P установился. Флаг P устанавливается или сбрасывается по значению только младшего байта результата, а оно в данном случае равно нулю.

При сложении чисел 7FFF и 0002 (строка 6) в результате получилось число 8001. Если рассматривать эти числа как беззнаковые, то результат правильный. Но процессор при выполнении команд сложения считает операнды знаковыми числами в дополнительном коде.

Для знаковых чисел это уже неправильный результат, поскольку 8001 — отрицательное число. Поэтому установились флаги знака S и переполнения O.

При сложении отрицательных чисел 8000 и 8001 (строка 9) в результате получилось положительное число 0001. Старшая единичка в двухбайтном регистре AX не поместилась, произошел перенос из старшего разряда. Поэтому установились флаги переноса C и переполнения O.

Обратим внимание, что после выполнения последующей команды инкремента регистра SI (строка 10) оказался установленным флаг C, хотя переноса не было. Это произошло потому, что команда инкремента (и декремента) на флаг переноса не влияет. В соответствии с результатом установились или сбросились все флаги, кроме флага C. Выполнение команд инкремента SI в последнем витке цикла является лишним, поскольку SI после них указывает уже за пределы массивов исходных данных. Можно было бы реализовать более эффективный алгоритм, привязав SI к счетчику цикла CX в команде loop. Но для достижения целей данного практического задания представленный алгоритм подходит лучше.

4. ФУНКЦИОНИРОВАНИЕ И ОБМЕН ДАННЫМИ В ЭВМ

4.1. ФУНКЦИОНИРОВАНИЕ ЭВМ

Последовательность действий процессора при выполнении программы

Процессор при своем функционировании постоянно выполняет следующий цикл: чтение команды, ее дешифрацию и выполнение (цикл «извлечь — выполнить»). При выполнении команды процессор может обращаться к переменным в памяти или портам ввода-вывода. Таким образом, принимая информацию по шине данных, процессор интерпретирует ее в зависимости от типа транзакции (см. п. 3.2). Команды RISC-процессора простые и, в идеале, одной длины. Поэтому RISC-процессор читает команду целиком. Команды CISC-процессора разной длины, читаются побайтно, число и смысл принимаемых байтов зависят от первого байта команды, кода операции. Для некоторых команд он является единственным. Другие команды могут содержать:

1. Дополнительные байты, необходимые для определения местоположения операндов (способа адресации): ModR/M (для процессоров x86), SIB (для процессоров IA-32 и Intel 64).
2. Адрес ячейки памяти или номер порта ввода-вывода, если в команде присутствует прямая адресация.
3. Собственно данные, предназначенные для обработки (непосредственный операнд), если в команде присутствует непосредственная адресация.

Рассмотрим последовательность действий процессора на простом примере команды непосредственной передачи данных в аккумулятор MVI A, data процессора i8080:

1. Адрес первого байта команды (Addr), кода операции, передается из счетчика команд, регистра PC, через буфер адреса (ABuf) на шину адреса (ABus):

$$[PC] \xrightarrow{Addr} [ABuf] \xrightarrow{Addr} [ABus].$$

2. Дешифратор адреса (DC) декодирует адрес (старшую часть адресных линий) — на определенном выходе дешифратора (O, Output) появляется 0, который подается на вход CS# микросхемы памяти (MEM, Memory):

$$[O_{DC}] \xrightarrow{0} [\overline{CS}_{MEM}].$$

3. Активизируется выход RC процессора (CPU), активный уровень сигнала RC подается на вход R/W# микросхемы памяти:

$$[RC_{CPU}] \xrightarrow{1} [R/\overline{W}_{MEM}].$$

4. Счетчик команд инкрементируется: PC = PC + 1 — теперь в PC содержится значение Addr+1.

5. Память из ячейки с указанным адресом Addr выдает КОП команды (Opcode) на шину данных (DBus):

$$[Addr] \xrightarrow{Opcode} [DBus].$$

6. КОП с шины данных через буфер данных и внутреннюю шину поступает в регистр команд (IR):

$$[DBus] \xrightarrow{Opcode} [DBuf] \xrightarrow{Opcode} [IBus] \xrightarrow{Opcode} [IR].$$

7. КОП, находящийся в регистре команд, декодируется дешифратором команд, и формируются управляющие сигналы, необходимые для дальнейшего выполнения команды.

8. Адрес второго байта команды (Addr+1), т. е. непосредственного операнда (data), передается из счетчика команд через буфер адреса на шину адреса:

$$[PC] \xrightarrow{Addr+1} [ABuf] \xrightarrow{Addr+1} [ABus].$$

9. Дешифратор адреса декодирует адрес и формирует сигнал активного уровня на входе CS# микросхемы памяти:

$$[O_{DC}] \xrightarrow{0} [\overline{CS}_{MEM}].$$

10. Активизируется сигнал RC процессора, поступающий на вход R/W# микросхемы памяти:

$$[RC_{CPU}] \xrightarrow{1} [R/\overline{W}_{MEM}].$$

11. Счетчик команд инкрементируется: [PC]=[PC]+1 — теперь в PC содержится значение Addr+2 (адрес первого байта следующей команды).

12. Память из ячейки с адресом Addr+1 выдает непосредственный операнд на шину данных:

$$[Addr + 1] \xrightarrow{data} [DBus].$$

13. Непосредственный операнд с шины данных через буфер данных и внутреннюю шину поступает в аккумулятор (A):

$$[DBus] \xrightarrow{data} [DBuf] \xrightarrow{data} [IBus] \xrightarrow{data} [A].$$

Также надо добавить, что процессор i8080 в первом такте очередного машинного цикла вместе с выдачей адреса осуществляет вы-

дачу на шину данных байта состояния, по значению которого формируются многие системные управляющие сигналы.

Графически последовательность действий процессора при выполнении команд иллюстрируют временные диаграммы (рис. 4.1).

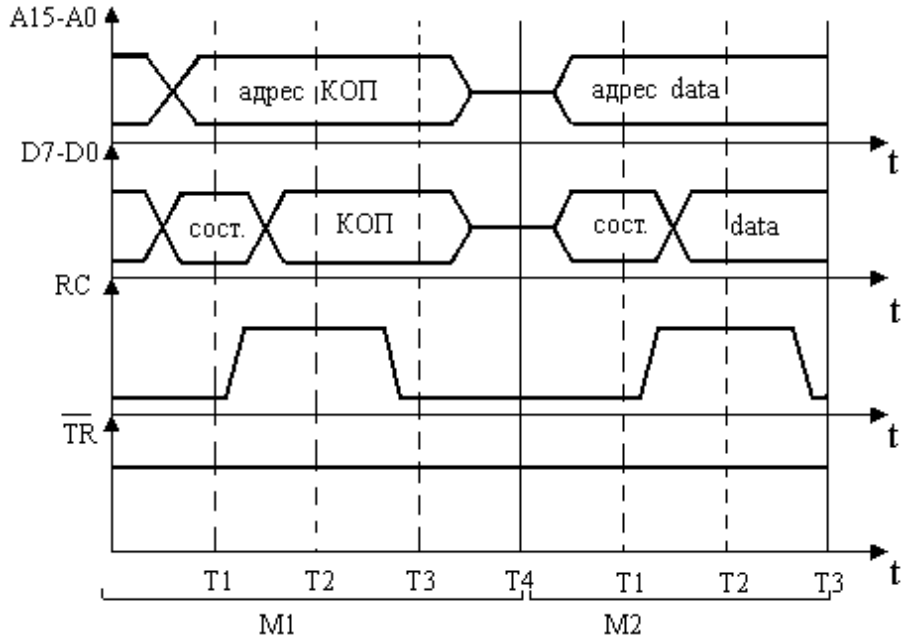


Рис. 4.1. Временная диаграмма выполнения команды MVI A, data

Команда MVI A, data выполняется за два машинных цикла. Первый цикл содержит четыре машинных такта, второй — три.

Машинный такт — это наименьший временной интервал вычислительной системы, один период тактовой частоты.

Машинный цикл (цикл шины, транзакция) — следующий по величине временной интервал системы. Для процессора i8080 машинный цикл может содержать от трех до пяти тактов. Цикл шины i8086 содержит, как минимум, четыре такта.

У процессора i8080 циклы шины, относящиеся к одной команде, следуют друг за другом и соответствуют действиям, выполняемым АЛУ, устройством управления и другими внутренними элементами процессора. Поэтому для цикла шины i8080 используется название машинный цикл. В процессоре i8086 циклы шины, относящиеся к одной команде, благодаря наличию устройства шинного интерфейса

и очереди команд могут и не следовать друг за другом, а перемежаться с циклами шины других команд. В последующих процессорах из-за наличия конвейеров, конвейеризации цикла шины, кэш-памяти нескольких уровней эта картина еще более усложняется, но сама вышеописанная логика действий процессора сохраняется.

Цикл команды — это временной интервал, за который выполняется она команда. Для i8080 он может содержать от одного до пяти машинных циклов. Их количество в цикле команды определяется числом обращений к памяти или внешним устройствам по шине данных. Это число в свою очередь определяется логикой команды и зависит от способа адресации. При побайтном обращении к памяти число машинных циклов в команде не может быть меньше ее длины в байтах. Исключение составляют команды условных переходов. При невыполнении условия, т. е. когда переход не будет выполнен, адрес перехода в некоторых процессорах не читается.

RISC-процессор выполняет большинство команд за один такт, т. е. для этого типа процессоров понятия цикла команды, машинного цикла и такта совпадают. Команда читается целиком и помещается в регистр команд. Код операции занимает в команде определенные поля. После его декодирования команда выполняется, и все эти действия от чтения команды до ее выполнения занимают по времени один такт.

Состояния и режимы работы процессора

С математической точки зрения процессор — это универсальный алгоритм (см. п. 1.3). При выполнении своего алгоритма процессор переходит из состояния в состояние. Поэтому с другой, тоже математической, точки зрения процессор — это конечный автомат. У него есть предопределенное множество состояний, в котором выделяется начальное состояние. В каждом состоянии процессор принимает входную информацию (сигналы, команды и данные) и в зави-

симости от ее значения и текущего состояния переходит в другое состояние, формируя определенные значения выходных сигналов (выходной информации). Элементарное состояние процессора характеризуется значениями всех его регистров и выходных управляющих сигналов. Оно изменяется в каждом такте. Элементарные состояния можно разделить на множества, т. е. выделить укрупненные состояния, различающиеся не только количественно, значениями регистров и сигналов, но и качественно, действиями процессора и его реакцией на входные сигналы. Среди них основным является обычное состояние, в котором процессор выполняет цикл «извлечь — выполнить». Кроме него, выделяются следующие состояния:

− ожидание;

− захват;

− останов.

Их длительность определяется внешними управляющими сигналами.

Кроме того, у современных процессоров имеются состояния пониженного энергопотребления.

Процессор при подаче на него питания или сигнала сброса устанавливается в начальное состояние. Для каждой модели процессора существует ограничение на длительность сигнала сброса — минимальное число тактов, за которое этот сигнал распознается. Начальным состоянием любого процессора является чтение первого байта команды из памяти. В начальном состоянии процессора прерывания запрещены, регистры имеют определенное значение: во всех разрядах нули или единицы. Это касается и регистров, по которым формируется адрес. Таким образом, после окончания действия сигнала сброса процессор производит первое обращение за чтением команды к ячейке памяти по предопределенному адресу.

Для процессора i8086 в начальном состоянии регистр флагов Flags, указатель команд IP, сегментные регистры имеют значения, указанные в табл. 4.1, выводы шины адреса/данных находятся в вы-

сокоомном состоянии, выводы сигналов управления — в высокоомном или пассивном состоянии, пошаговые и маскируемые прерывания запрещены.

Таблица 4.1

Начальное значение регистров процессора i8086

Регистр	Значение регистра
Flags	$F002_{16}$
IP	0000_{16}
CS	$FFFF_{16}$
DS	0000_{16}
ES	0000_{16}
SS	0000_{16}

Таким образом, по окончании сигнала сброса первая команда начинает выбираться из ячейки памяти с физическим адресом FFFF0h. Обычно первой командой является команда межсегментного перехода.

Для процессоров IA-32 и Intel 64 в начальном состоянии CS.Base = FFFF0000h, EIP = 0000FFF0h, и процессор читает первую команду по адресу FFFFFFF0h.

В зависимости от выполняемой команды, значений управляющих сигналов и внутреннего состояния процессор осуществляет то или иное действие, переходит в то или иное состояние, формируя необходимые управляющие сигналы. Управляющие сигналы могут формироваться аппаратной логикой процессора или микропрограммно. В последнем случае каждой команде процессора соответствует программа (подпрограмма), располагающаяся в специальной памяти процессора — микропрограммной памяти. Микропрограмма состоит из микрокоманд. Микрокоманда представляет собой совокупность битов, соответствующих определенным управляющим сигналам процессора. Значение бита определяет значение сигнала в данном такте.

В состояние ожидания процессор переходит, когда на входе готовности (RDY, DRDY# и т. п.) появляется сигнал неактивного уровня. О переходе в это состояние он сигнализирует активизацией соответствующего выходного сигнала (Waiting и т. п.). Сигнал готовности может быть использован для согласования работы процессора с работой медленных устройств, а также для организации пошагового выполнения программы (по командам или по машинным циклам).

Переход в состояние захвата происходит при появлении сигнала активного уровня на соответствующем входе (HLD, RQ# и т. п.). О переходе в это состояние процессор информирует окружение активным уровнем сигнала подтверждения захвата (HLDA, GT# и т. п). В этом состоянии буферные схемы шин адреса и данных переходят в высокоомное состояние, т. е. выводы адреса и данных процессора отключаются от шины. Это состояние позволяет организовать прямой доступ к памяти для любого внешнего устройства, способного сформировать необходимые управляющие и адресные сигналы и осуществить обмен данными с памятью.

В состояние останова процессор переходит при выполнении команды останова (HLT, HALT и т. п.). Переход в это состояние может быть подтвержден соответствующим выходным сигналом (HLTA и т. п.). При этом буферные схемы адреса и данных так же переводятся в высокоомное состояние. Процессор выходит из состояния останова, когда на его вход поступает один из внешних сигналов:

1. *Сброс.* Процессор переходит в начальное состояние и начинает выполнять цикл «извлечь — выполнить».

2. *Захват.* Процессор переходит в состояние захвата и возвращается из него при дезактивации сигнала захвата в состояние останова.

3. *Прерывание.* Процессор переходит к обработке прерывания, если прерывания разрешены, иначе остается в состоянии останова. После обработки прерывания в состояние останова процессор не возвращается.

При переходе к обработчику прерывания процессор выдает в первом такте сигнал подтверждения прерывания, который используется контроллером прерываний или другой аппаратной логикой для выдачи на шину данных команды вызова подпрограммы или номера прерывания.

Помимо различных состояний, в которые процессор может переходить при своем функционировании, он имеет различные режимы работы. Режим можно трактовать как еще более укрупненное состояние. Режимы выделяются по множеству доступных ресурсов процессора: входных и выходных сигналов, регистров и команд. Процессор i8086 имеет два режима работы: минимальный и максимальный (см. п. 3.2). Выбор режима осуществляется на этапе разработки вычислительной системы. Режим задается аппаратно подключением вывода MN/MX# к общей точке (логический 0) или к шине питания +5 В (логическая 1). Режимы различаются множествами управляющих сигналов и, соответственно, назначением некоторых выводов.

Начиная с i80286 в процессорах x86 появляется новый режим работы — защищенный. Прежний, обычный режим работы стал называться реальным.

Процессоры семейства IA-32 имеют следующие режимы работы:

− реальный (Real Mode) — режим процессора i8086, любой процессор семейства x86, в том числе IA-32 и Intel64, оказывается в этом режиме при включении питания или при сбросе;

− защищенный (Protected Mode), предоставляет все возможности архитектуры процессора;

− режим виртуального 8086 (Virtual-8086 Mode) — реальный режим в защищенной многозадачной среде;

− режим управления системой (SMM, System Management Mode) — предоставляет возможность управления питанием или безопасностью системы.

Процессоры Intel 64 имеют все режимы IA-32 и дополнительный — IA-32e, предоставляющий 64-битную программную среду. IA-32e имеет два подрежима:

− 64-битный режим (64-bit Mode), поддерживающий 64-битные операционные системы и приложения;

− режим совместимости (Compatibility Mode), позволяющий работать любому программному обеспечению в 64-битной среде.

В защищенном режиме существует множество режимов страничной адресации, а также два режима расширений виртуальной машины: корневой (root-VMX) и некорневой (non-root-VMX) (см. п. 3.4).

Защищенный режим неразрывно связан с понятием системной архитектуры, которая включает множество системных регистров, структур данных и команд. В рамках системной архитектуры в защищенном режиме осуществляется:

− управление памятью;

− защита программных модулей;

− реализация виртуальной памяти с использованием механизма страничной адресации;

− реализация многозадачности;

− реализация многопроцессорности;

− обработка исключений и прерываний;

− управление кэш-памятью;

− управление питанием;

− управление аппаратными средствами;

− отладка и мониторинг производительности.

Для защиты программных модулей при реализации многозадачности в защищенном режиме существует четыре уровня привилегий, от 0 до 3. Ядро операционной системы работает с наивысшим, нулевым, текущим уровнем привилегий (Current Privilege Level, CPL). Пользовательские программы — с самым низким, третьим. Таким образом, можно говорить о различных режимах с точки зрения уровня

привилегий — системном (supervisor mode), для которого CPL<3, и пользовательском (user mode) — CPL=3.

Переключение процессора из реального режима в защищенный осуществляется программно, установкой бита 0 управляющего регистра CR0. Но до этого необходимо выполнить ряд подготовительных действий: создать необходимые системные структуры данных (таблицы дескрипторов, каталоги и таблицы страниц и т. п.), разрешить при необходимости полное адресное пространство включением адресной линии A20, загрузить системные регистры. Переключение из защищенного режима в реальный осуществляется сбросом бита 0 регистра CR0.

Особенности работы процессора в защищенном режиме и системной архитектуры в целом подробно рассматриваются в дисциплине «Архитектура вычислительных систем» в программах ИТ-направлений и специальностей высшего образования [58].

Прерывания и исключения

По способу синхронизации обмена данными выделяется два способа взаимодействия процессора с внешними устройствами или два типа интерфейсов ввода-вывода:

1. Ввод-вывод по опросу — синхронный ввод-вывод, обмен инициирует процессор.
2. Ввод-вывод по прерыванию — асинхронный ввод-вывод, обмен инициирует внешнее устройство.

Ввод-вывод по опросу занимает часть процессорного времени. При этом способе процессор постоянно обращается к внешнему устройству для определения его готовности к обмену данными, с определенной периодичностью определяет его текущее состояние и вводит или выводит информацию. Он может использоваться в том случае, если не вносит существенного замедления в работу системы.

Прерывания — очень важный механизм, используемый во всех вычислительных системах. Он позволяет более эффективно исполь-

зовать процессорное время. При ипользовании прерываний процессору не нужно проверять в цикле состояние устройств ввода-вывода. При необходимости обмена само устройство обращается к процессору, формируя сигнал прерывания с помощью контроллера прерываний или других аппаратных средств.

Прерывания и опрос могут сочетаться. Например, в клавиатуре персональных компьютеров имеется микроконтроллер, процессор которого взаимодействует с матрицей клавиш по опросу, осуществляет ее постоянное сканирование и при обнаружении фактов нажатия и отпускания клавиш отправляет соответствующую информацию в формате так называемого скан-кода контроллеру клавиатуры (см. п. 4.3). Тот размещает его в определенном регистре и формирует сигнал прерывания, который через контроллер прерываний поступает процессору.

Сигнал запроса прерывания от внешнего устройства поступает на один из входов IRQ (Interrupt ReQuest) контроллера прерываний. Номер входа запроса прерывания IRQ — один из видов аппаратных ресурсов компьютера (см. п. 2.2) (рис. 4.2).

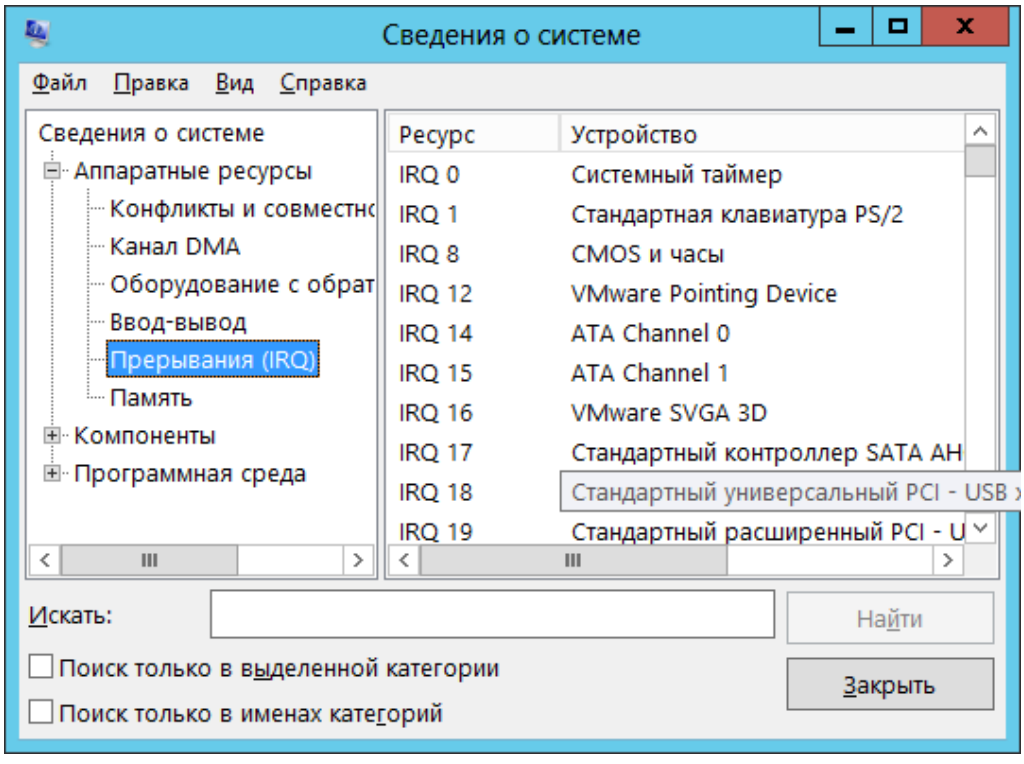


Рис. 4.2. Номера запросов прерывания различных устройств компьютера

Контроллер прерываний в соответствии с текущими установленными разрешениями и приоритетами подает сигнал прерывания на вход маскируемых запросов прерывания INTR (INT) процессора. Если прерывания разрешены, процессор выдает сигнал подтверждения прерывания INTA#, по которому контроллер прерываний выдает на шину данных соответствующий номеру IPQ номер прерывания INT, называемый также вектором прерывания. Данный цикл шины называется циклом подтверждения прерывания. Размер номера прерывания — один байт. Таким образом, возможны 256 номеров прерываний — от 0 до 255. Некоторые из номеров предопределены аппаратно, другие определяются программно. По номеру прерывания процессор определяет адрес обработчика прерывания. Обработчик прерывания (ISR, Interrupt Service Routine) — это подпрограмма, которая и осуществляет обмен информацией между процессором и внешним устройством или другие необходимые действия, соответствующие смыслу прерывания. Адрес подпрограммы — это адрес ее первой команды. При переходе к обработчику прерывания в стеке сохраняется не только адрес возврата, но и регистр флагов. Для определения адресов обработчиков и перехода к ним используется три типа таблиц:

– таблица адресов прерываний реального режима;

– IDT (Interrupt Descriptor Table) — таблица дескрипторов прерываний, используется в защищенном режиме;

– LVT (Local Vector Table) — локальная таблица векторов прерываний, своя у каждого APIC.

Первые две таблицы располагаются в памяти, LVT — это регистры APIC.

В реальном режиме адреса обработчиков прерываний находятся в таблице, которая располагается в самом начале адресного пространства, с нулевого адреса, в оперативной памяти. Адрес обработчика прерывания — четырехбайтный логический адрес: два байта — база сегмента, и два байта — смещение от начала сегмента. Таким образом, адрес обработчика прерывания в таблице — это номер прерыва-

ния, умноженный на четыре. Соответственно, размер таблицы векторов прерываний — 1 Кбайт.

В защищенном режиме адреса обработчиков определяются по дескрипторам (шлюзам) прерываний, расположенным в IDT. Номер прерывания — это номер шлюза в IDT.

Имеются и прерывания без номера. Это, например, внутренние прерывания и прерывание SMI. Информация, необходимая для обработки прерываний и, в частности, внутренних прерываний также содержится и в LVT.

В архитектуре Intel 64 имеется еще один тип таблиц, связанных с прерываниями, — IST (Interrupt Stack Table). Они используются для задания местоположения стека в 64-битном режиме.

Вообще, система прерываний в архитектуре IA-32 и Intel 64 сложна и многообразна. По разным признакам выделяются следующие типы прерываний:

- программные и аппаратные;
- синхронные и асинхронные;
- внешние и внутренние;
- маскируемые и немаскируемые;
- с номером и без номера.

Программные прерывания являются синхронными. Они вызываются командами INT n, INTO, INT1, INT3 и BOUND. Все остальные прерывания — аппаратные. Тот тип прерываний, о которых шла речь неоднократно ранее, — внешние прерывания от устройств ввода-вывода. Все они являются асинхронными и имеют номера, в том числе немаскируемое прерывание. Немаскируемое прерывание не зависит от флага IF, маскируемые могут быть запрещены или разрешены сбросом или установкой этого флага. Немаскируемое прерывание принимается по входу LINT1 (NMI), маскируемое — по входу LINT0 (INTR). В компьютере имеется два вида APIC — процессора (local APIC) и APIC ввода-вывода (I/O APIC), который располагается в чипсете. Внешние прерывания принимаются I/O APIC и отправляются APIC процессора.

Кроме того, имеются межпроцессорные прерывания (Interprocessor Interrupts, IPI), которые посредством APIC передаются от одного процессора другому в многопроцессорной системе или в многоядерном (многопоточном) процессоре.

Внутренние прерывания — это прерывания от счетчиков мониторинга производительности, датчика температуры и т. п. Они не имеют номера и поступают непосредственно локальному APIC.

Также надо иметь в виду, что прервать работу процессора могут и другие внешние входные сигналы: RESET#, FLUSH#, STPCLK#, SMI#, R/S# и INIT#. Каждый из них обрабатывается по-своему.

И наконец, имеется еще один тип прерывания — исключение. Исключение определяется как синхронное событие, которое генерируется, когда процессор обнаруживает одно или несколько предопределенных условий во время выполнения команды. Однако в командах программных прерываний INT1, INT3 переход к обработчику соответствующего исключения осуществляется безусловно.

В зависимости от источника возникновения и от того, может ли выполнение программы или задачи быть продолжено без потери непрерывности, выделяется три типа исключений: ловушка (trap), ошибка (fault) и прекращение (abort).

Ловушка — это исключение, которое происходит по одной из следующих команд программных прерываний:

- INT1: #DB (Debug);
- INT3: #BP (Breakpoint);
- INTO: #OF (Overflow).

Ловушка #DB также может происходить при обращении по адресам, указанным в качестве точек останова в отладочных регистрах. В некоторых случаях #DB имеет тип не ловушки, а ошибки. Ловушки позволяют продолжить выполнение программы или задачи без потери непрерывности. Адрес возврата (CS:EIP) указывает на команду, которая должна выполниться после команды ловушки.

Ошибка — это исключение, которое может быть исправлено и которое после исправления позволяет продолжить выполнение программы так же без потери непрерывности. Процессор восстанавлива-

ет состояние, которое было до начала выполнения команды, вызвавшей ошибку. Адрес возврата в стеке указывает на команду, вызвавшую ошибку. При возникновении некоторых исключений типа ошибки или прекращения в стеке также сохраняется код ошибки.

Примеры исключений типа ошибки:

– #DE (Divide Error), происходит при выполнении команд DIV и IDIV в случае деления на ноль;

– #BR (BOUND Range Exceeded), вызывается командой BOUND при выходе за пределы границ массива;

– #UD (Undefined Opcode), вызывается командой UD или недействительным кодом операции;

– #NM (No Math Coprocessor), вызывается командой FPU при отсутствии FPU;

– #GP (General Protection), происходит при выполнении любой команды обращения к памяти при нарушении прав доступа или в результате других проверок;

– #PF (Page Fault), ошибка страницы;

– #VE (Virtualization Exception), нарушение при обращении к EPT (Extended Page Table).

Так как для исключения типа ошибки адрес возврата указывает на команду, вызвавшую ошибку, то после возврата из обработчика она выполняется повторно. Например, исключение ошибки страницы #PF возникает, когда в программе или задаче происходит обращение к операнду, расположенному в странице, которой нет в памяти. Обработчик этого исключения загружает страницу в память и возвращает управление той же команде обращения к данной странице, т. е. продолжается нормальное выполнение программы.

Прекращение — это исключение, которое не позволяет продолжить непрерывное выполнение программы или задачи по причине неисправимости ошибки или невозможности точного определения команды, вызвавшей исключение. Исключения типа прекращения вызывают аппаратные ошибки и несогласованные или недопустимые значения в системных таблицах. Обработчик этого типа исключений

должен выполнить сбор диагностической информации о состоянии процессора в момент возникновения исключения, а затем максимально корректно завершить работу программы и системы.

Примеры исключений типа прекращения:

– #DF (Double Fault), происходит, когда процессор обнаруживает второе исключение при вызове обработчика исключений для предыдущего исключения;

– #MC (Machine Check), происходит, когда процессор обнаруживает внутреннюю машинную ошибку или ошибку шины.

В IDT могут присутствовать как шлюзы прерываний, так и шлюзы ловушек. В шлюзах обоих этих типов содержится следующая информация:

– права доступа (уровень привилегий);

– селектор сегмента кода, содержащего обработчик;

– смещение в сегменте кода первой команды обработчика.

Различие между этими двумя типами шлюзов состоит в том, что при переходе к обработчику через шлюз прерывания флаг IF сбрасывается (прерывания запрещаются), а при переходе через шлюз ловушки флаг IF не изменяется. Возврат из обработчика прерывания или исключения осуществляется по команде IRET. Она аналогична команде возврата из дальней подпрограммы за исключением того, что она также восстанавливает из стека значение регистра флагов. От кода ошибки команда IRET стек не освобождает. Это должен сделать обработчик до выполнения возврата.

В настоящее время широко используется еще один механизм прерываний — MSI (Message Signaled Interrupt — прерывание, инициируемое сообщением). Он был введен в спецификации интерфейса PCI версии 2.2, а затем в спецификации PCI 3.0 появилась его расширенная версия — MSI-X. Этот механизм применяется во многих современных последовательных интерфейсах: PCIe, SATA, USB (см. п. 4.2).

4.2. ИНТЕРФЕЙСЫ ПАМЯТИ И ВВОДА-ВЫВОДА В ЭВМ

Классификация интерфейсов ЭВМ

В этом пункте подытожим и дополним все вышесказанное об интерфейсах, использующихся в компьютерных системах.

В полном смысле интерфейс в ЭВМ — это совокупность средств взаимодействия элементов ЭВМ между собой. В нем выделяются три части:

– аппаратная: микросхемы, соединения, разъемы и т. п.;

– информационная: протокол — порядок и последовательность обмена, формат передаваемых данных, пакетов, кадров и т. п.;

– программная: код, выполняемый процессорами для реализации протокола и выполнения обмена.

Важными характеристиками интерфейса являются скорость передачи и пропускная способность. Скорость передачи определяется как произведение количества линий данных (одновременно передаваемых битов) на тактовую частоту или количество передач (транзакций) в секунду, если за один такт данные могут быть переданы несколько раз. Пропускная способность в общем случае не больше скорости передачи, так как собственно данные передаются не в каждом такте. В большинстве последовательных интерфейсов в информационных посылках вместе с битами данных содержатся биты синхронизации. Передача данных осуществляется в виде пакетов, которые помимо собственно данных включают адресную и управляющую информацию. Можно сказать, что одна физическая линия последовательного интерфейса содержит шину данных, шину адреса и шину управления. Кроме того, существуют задержки и повторные передачи, так что пропускную способность, как и производительность процессора, можно разделить на пиковую (максимальную или теоретическую) и реальную. Пиковая пропускная способность последовательного интерфейса равна скорости передачи, умноженной на отношение числа битов данных к общему числу передаваемых битов.

Существует огромное количество различных видов интерфейсов. Набор интерфейсов, использующихся в ЭВМ, определяется ее типом. Основные интерфейсы персональных ЭВМ были рассмотрены в общих чертах в п. 2.3. Многие из них используются и в других типах ЭВМ: мобильных, встроенных и высокопроизводительных. Например, интерфейс PCIe реализован в современных процессорах семейства x86, различные модели которых используются в мобильных и настольных системах, а также в суперкомпьютерах.

В соответствии с концепцией «треугольника фон Неймана» на верхнем уровне выделяется два типа интерфейса: процессора с памятью и процессора с устройствами ввода-вывода.

Интерфейс с памятью в современных компьютерах на основе процессоров x86 реализует интегрированный в состав процессора контроллер памяти (Integrated Memory Controller, IMC). Этот интерфейс характеризуется и различается в основном количественными параметрами: количеством каналов, таймингами, тактовой частотой и пропускной способностью.

IMC поддерживает технологию Intel Flex Memory Technology и имеет несколько режимов работы:

− одноканальный (Single-Channel Mode);

− двухканальный (Dual-Channel Mode), «родной» для Intel Flex Memory Technology;

− симметричный двухканальный (Dual-Channel Symmetric Mode (Interleaved Mode, IL)).

Если объем памяти в каналах разный, память делится на симметричную и асимметричную зоны. Доступ к симметричной зоне осуществляется в IL-режиме. Доступ к асимметричной зоне осуществляется в одноканальном режиме. Реальная производительность компьютера для большинства приложений максимальна в IL-режиме. Поэтому для достижения максимальной производительности необходимо устанавливать одинаковый объем памяти в обоих каналах. В этом случае доступ ко всей памяти осуществляется в IL-режиме.

Пропускная способность интерфейса с памятью рассчитывается как произведение скорости передачи на общую ширину каналов в байтах (для одного канала — 8 байт):

− 3200 MT/s · 8 байт = 25600 МБайт/с;

− 5000 MT/s · 8 байт = 40000 МБайт/с;

− 7200 MT/s · 8 байт = 57600 МБайт/с.

Интерфейсы с внешними устройствами намного разнообразнее. По основным признакам их можно классифицировать следующим образом:

1. По количеству линий данных (числу одновременно передаваемых битов):

1.1. Параллельные.

1.2. Последовательные.

2. По направлению передачи информации:

2.1. Односторонние (однонаправленные, симплекс).

2.2. Двусторонние (двунаправленные) с поочередной передачей (полудуплексные).

2.3. Двусторонние (двунаправленные) с одновременной передачей (дуплексные, полнодуплексные).

3. По наличию тактирования:

3.1. Синхронные.

3.2. Асинхронные.

4. По среде передачи:

4.1. Проводные.

4.2. Безпроводные.

5. По множеству типов подключаемых устройств:

5.1. Универсальные (общего назначения).

5.2. Специализированные.

6. По месту расположения контактов или разъемов (внутри или снаружи корпуса компьютера, системного блока):

6.1. Внутренние.

6.2. Внешние.

7. По топологии:

7.1. Многоточечные (магистральные, шина).

7.2. Двухточечные (точка-точка).

7.3. Звездообразные (радиальные, звезда).

Шина — это соединение нескольких устройств между собой одним набором проводников.

«Точка-точка» — соединение двух устройств.

Звезда — усложнение двухточечной топологии, когда в интерфейсе дополнительно используется специальный тип устройств с множеством точек подключения, который в общем случае называется концентратором или хабом (hub). К каждой из этих точек может быть подключено одно устройство.

Многоточечными последовательными интерфейсами, т. е. шинами, являются:

- I2C;
- SMBus (System Management Bus), специализация I2C;
- SPI (Serial Peripheral Interface).

В компьютерных системах промышленной автоматизации используется интерфейс RS-485, также имеющий шинную топологию.

Топологию «точка-точка» имеют:

- PCI Express (PCIe);
- DMI (Direct Media Interface);
- QPI (QuickPath Interconnect);
- SATA (Serial ATA);
- SAS (Serial Attached SCSI);
- UART (Universal Asynchronous Reciver-Transmitter), USART (Universal Synchronous-Asynchronous Reciver-Transmitter).

Интерфейс USB имеет многоуровневую звездообразную топологию, другими словами, иерархическую или древовидную структуру.

Интерфейс с памятью всегда был и остается параллельным, а интерфейсы с внешними устройствами практически все уже в настоящее время являются последовательными. В большинстве последовательных интерфейсов, в том числе PCIe, SAS, QPI, DMI, для передачи

данных в одном направлении используется так называемая дифференциальная пара — два проводника, по которым передается один сигнал, представляющий собой разность сигналов этих двух проводников. Одна линия передачи состоит из двух дифференциальных пар, так что данные одновременно передаются в обоих направлениях, т. е. в полнодуплексном режиме.

Большинство интерфейсов являются синхронными, но не всегда тактовый сигнал передается по отдельной линии. Он может быть встроен в последовательную посылку данных. Некоторые интерфейсы, например USART, имеют два режима работы: синхронный и асинхронный. В синхронном режиме для передачи тактового сигнала используется отдельная линия. В асинхронном режиме тактовый сигнал и соответствующая линия не используются, но в начале посылки данных вставляется старт-бит, а в конце — один или два стоп-бита.

Также большинство интерфейсов являются в той или иной степени универсальными, т. е. используются для подключения различных типов устройств. Специализированные интерфейсы предназначены для подключения устройств только одного типа. Они более редки. В качестве примеров можно привести AGP (Accelerated Graphics Port) — специализированный интерфейс для подключения видеоадаптера и ATA (PATA, SATA) — для подключения устройств внешней памяти (дисков, накопителей).

Разъемы внутренних интерфейсов расположены внутри корпуса компьютера или системного блока. Они предназначены для подключения графических, звуковых, сетевых плат, накопителей. Конечно, есть внутренние интерфейсы и без разъемов, например, DMI и QPI. Внешние интерфейсы могут быть проводными или безпроводными. Разъемы проводных внешних интерфейсов могут быть расположены на различных панелях корпуса: задней, боковой, передней и даже нижней (у моноблоков). Внешние интерфейсы, в свою очередь, можно разделить на два подтипа:

– для подключения «наиболее» периферийных устройств: клавиатуры, мыши, принтера, флешки, внешнего диска;

– сетевые, для соединения компьютеров или устройств со встроенными компьютерами, например, IP-камер, в компьютерную сеть.

К сетевым интерфейсам относятся (см. п. 2.3):

– цифровые интерфейсы мобильной сотовой связи, поддерживающие стандарт GPRS и более поздние;

– Wi-Fi;

– Bluetooth;

– NFC (Near-Field Communication);

– Ethernet.

Некоторые внешние интерфейсы могут использоваться как в качестве сетевых, так и для подключения периферии. Например, по Bluetooth можно подключить наушники к смартфону, а можно соединить два смартфона для передачи файла. USB можно использовать для подключения к персональному компьютеру смартфона или различных периферийных устройств: мыши, клавиатуры, флешки, принтера и т. д.

В суперкомпьютерах для соединения отдельных узлов между собой часто используется высокоскоростной Ethernet, а также специализированные сети, такие как InfiniBand, Slingshot, Omni-Path (см. п. 5.2).

Многие современные интерфейсы ввода-вывода имеют многоуровневую организацию и могут работать в различных режимах (см. далее). В зависимости от уровня и режима один и тот же интерфейс может относиться к различным типам. Например, PCIe на нижнем, физическом уровне — «звезда», на верхнем уровне — «точка-точка». USB в режиме SuperSpeed (SS) является полнодуплексным, в других режимах — полудуплексным. SATA на физическом уровне полнодуплексный, на верхнем — полудуплексный.

Интерфейс PCIe

PCIe (PCI (Peripheral Component Interconnect) Express) — высокоскоростной двунаправленный (полнодуплексный) синхронный последовательный интерфейс общего назначения. В настоящее время он широко используется во многих типах компьютерных систем для подключения самых различных устройств внутри системного блока: видеоадаптеров, SSD-накопителей, сетевых и звуковых карт. PCIe имеет множество модификаций и форм-факторов для различных типов систем и устройств.

Основные преимущества PCIe:

- высокая масштабируемость пропускной способности;
- низкие задержки;
- поддержка горячего подключения;
- улучшенная система управления питанием;
- детализированный контроль ошибок.

Интерфейс PCIe на физическом уровне имеет топологию «звезда», его центральным элементом является коммутатор (Switch), с которым каждое устройство связано выделенным соединением «точка-точка», что исключает необходимость арбитража шины и возможность возникновения конфликтов (рис. 4.3) [61].

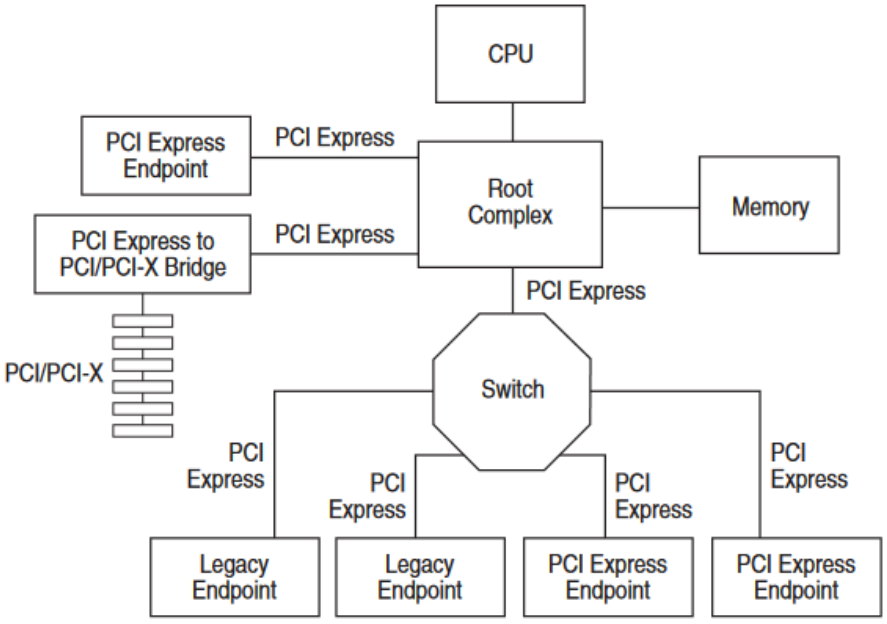


Рис. 4.3. Структура интерфейса PCIe в составе компьютера

Корневой комплекс (Root Complex, RC) — элемент, соединяющий подсистему CPU/Memory с подсистемой ввода-вывода. Он может входить в состав микросхемы чипсета (см. рис. 2.16) или самого процессора (см. рис. 2.19–2.21). Мост PCI Express - PCI/PCI-X обеспечивает подключение устройств с интерфейсом PCI/PCI-X. Функционирование интерфейса обеспечивает множество регистров, которые имеются в составе корневого комплекса, коммутатора и самих подключаемых устройств (Endpoint) (рис. 4.4).

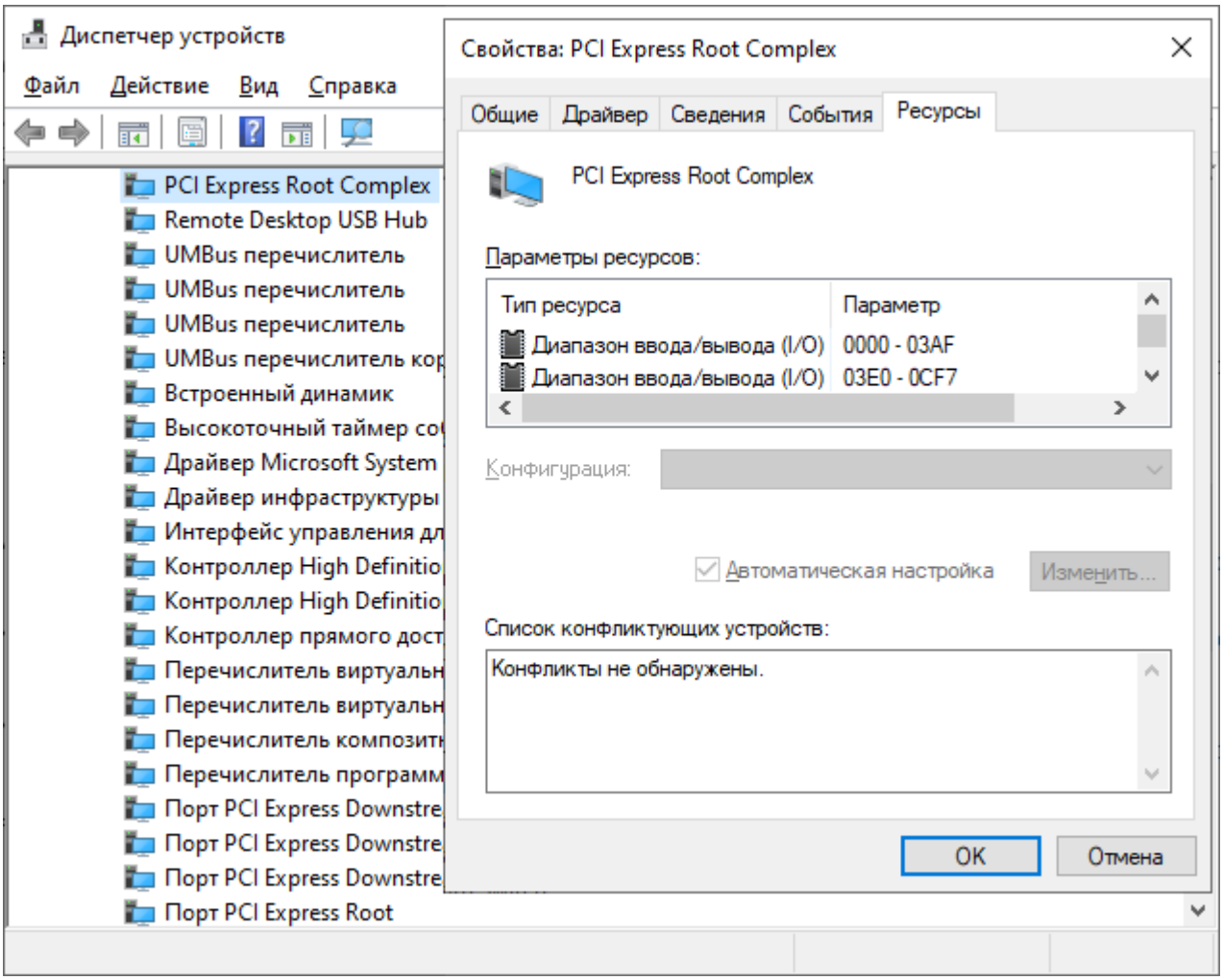


Рис. 4.4. Ресурсы корневого комплекса включают адреса портов (регистров)

В множестве регистров подключаемых устройств выделяются такие части, как конфигурационное пространство (Configuration Space) и «Capability Structures». Конфигурационное пространство содержит информацию об устройстве и используется для управления

им. Capability Structures предоставляют расширенные функции для управления прерываниями, питанием и производительностью, а также обработки ошибок. С помощью этих регистров система обнаруживает устройство, настраивает его и обеспечивает дальнейшую работу.

PCIe имеет трехуровневую архитектуру (рис. 4.5). В нем выделяются:

1. Транзакционный уровень (Transaction Layer).
2. Канальный уровень (Data Link Layer).
3. Физический уровень (Physical Layer).

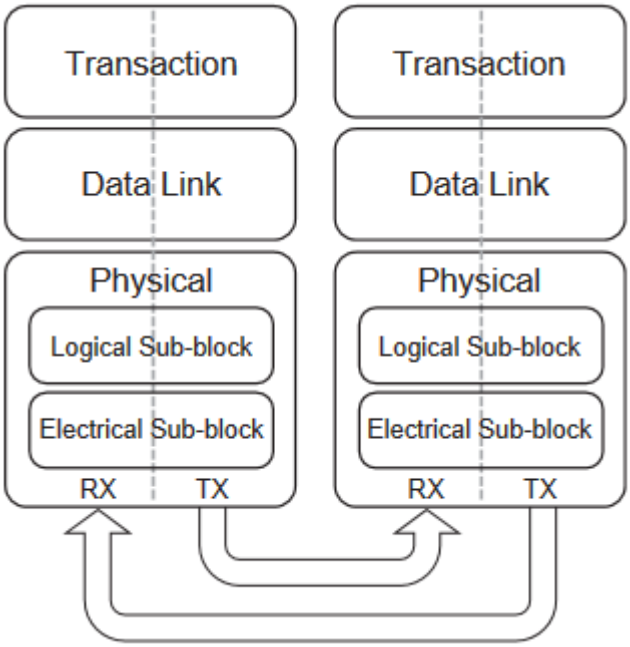


Рис. 4.5. Трехуровневая архитектура PCIe

Данные в PCIe передаются в форме пакетов. Пакеты имеют сложную структуру, включающую множество полей, содержащих всю необходимую информацию: адресную, управляющую и собственно данные. Каждый пакет имеет, в том числе, метку класса трафика (Traffic Class, TC) — трехбитное поле TC[2:0], которое содержится в заголовке пакета TLP. Метки TC позволяют дифференцировать трафик по уровням приоритета обслуживания. Например, если TC[2:0]=000_2, то это метка TC0 — Best Effort service class (Лучший

класс обслуживания). На транзакционном уровне осуществляется формирование (в передающем устройстве) или разбор (в принимающем устройстве) пакетов TLP (Transaction Layer Packets), управление потоком данных и виртуальными каналами (см. далее), поддержка QoS (Quality of Service). ***Имеются следующие типы транзакций:***

- чтение или запись памяти (Memory Read/Write);
- ввод-вывод (I/O Read/Write);
- конфигурирование (Configuration Read/Write);
- передача сообщений (Message).

На уровне канала передачи данных осуществляется обеспечение целостности данных, включая обнаружение и исправление ошибок. Передающая сторона на канальном уровне добавляет в TLP порядковый номер, контрольный код и отправляет на физический уровень для передачи принимающей стороне. Принимающая сторона на канальном уровне осуществляет проверку целостности полученных TLP и отправку их на транзакционный уровень для дальнейшей обработки. При обнаружении ошибки на канальном уровне выполняется запрос повторной передачи TLP до тех пор, пока информация не будет получена корректно или пока не будет установлено, что связь прервана. На канальном уровне также генерируются пакеты DLLP (Data Link Layer Packets), которые используются для управления каналом.

Физический уровень включает в себя все аппаратные схемы интерфейса: буферы, схемы преобразования данных из параллельного формата в последовательный и обратно, линии (lanes) для передачи данных и т. п. На физическом уровне осуществляется инициализация канала, кодирование-декодирование данных (8b/10b, 128b/130b, 242b/256b), синхронизация. PCIe в зависимости от конфигурации (x1, x2, x4, x8, x16, x32) включает одну или множество линий. Одна линия (lane) состоит из двух дифференциальных пар проводников: для передачи (TX) и для приема (RX). Пропускная способность масштабируется линейно с количеством линий (табл. 4.2).

Таблица 4.2

Основные характеристики интерфейса PCIe

Версия PCIe	Год	Кодирование	Скорость передачи, ГТ/с	Пропускная способность, Гбит/с	
				x1	x16
PCIe 1.0	2002	8b/10b	2,5	2	32
PCIe 2.0	2007	8b/10b	5	4	64
PCIe 3.0	2010	128b/130b	8	7,877	126,031
PCIe 4.0	2017	128b/130b	16	15,754	252,062
PCIe 5.0	2019	128b/130b	32	31,508	504,123
PCIe 6.0	2022	242b/256b	64	60,5	968
PCIe 7.0	2025	242b/256b	128	121	1936

В интерфейсе PCIe используется виртуализация в форме т. н. механизма виртуального канала (VC, Virtual Channel) (рис. 4.6) [60].

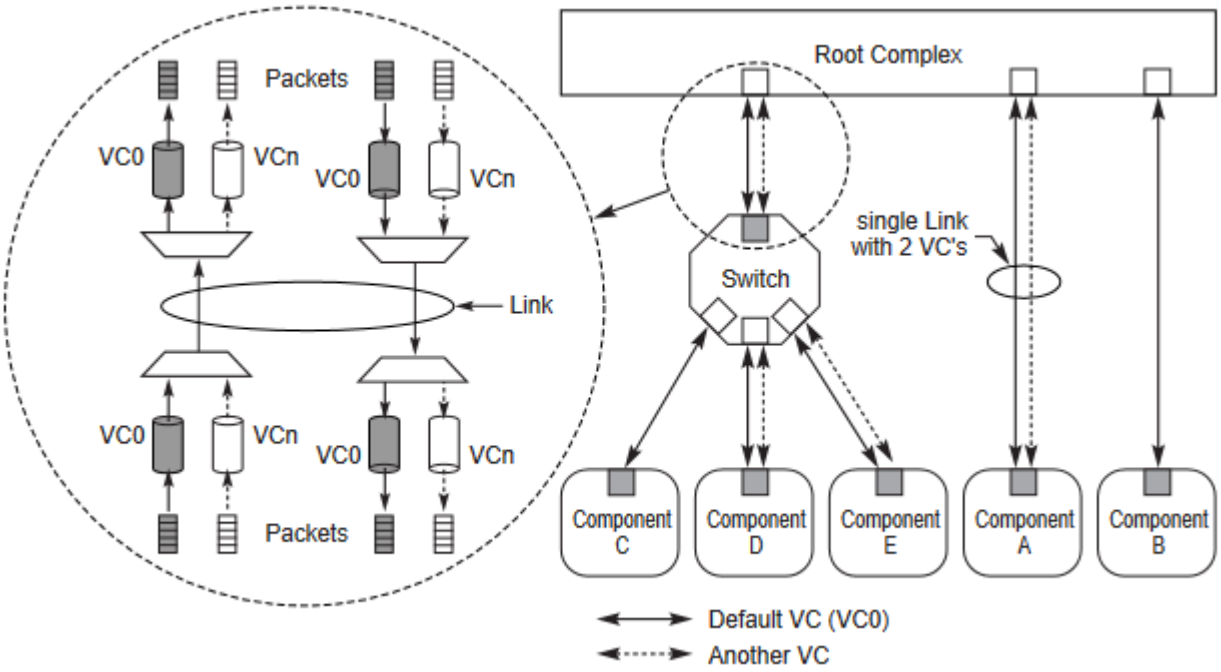


Рис. 4.6. Механизм виртуального канала

Основой механизма VC являются независимые ресурсы: очереди, буферы и связанная с ними логика управления. Эти ресурсы ис-

пользуются для перемещения информации по физическим каналам связи с полностью независимым управлением потоками между различными виртуальными каналами. Концепция механизма VC состоит в том, что трафик, проходящий через виртуальные каналы, мультиплексируется на общий физический канальный ресурс на передающей стороне (рис. 4.7) и демультиплексируется на отдельные каналы VC на принимающей стороне [60].

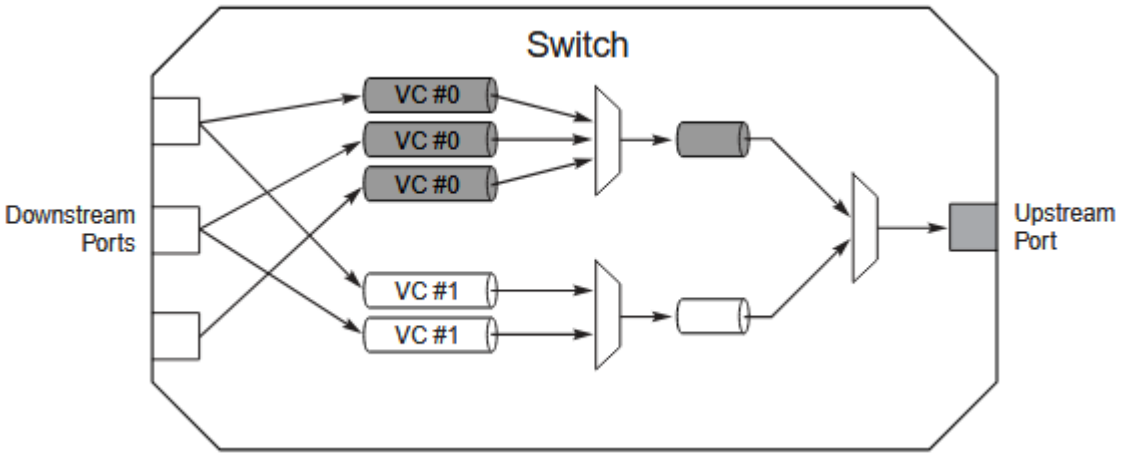


Рис. 4.7. Мультиплексирование виртуальных каналов в коммутаторе в восходящем потоке данных (Upstream Flow)

В интерейсе PCIe восходящий поток (Upstream Flow) — поток данных в сторону корневого комплекса, нисходящий поток (Downstream Flow) — от корневого комплекса. Соответственно, Upstream Port — порт со стороны корневого комплекса, Downstream Port — порт с обратной стороны. У корневого комплекса все порты — нисходящие.

Механизм VC позволяет оптимизировать трафик и решить проблему блокировки, когда один поток трафика создает узкое место для всего трафика в системе. Пакеты распределяются по виртуальным каналам в соответствии со значением их меток TC. Для этого осуществляется отображение (mapping) каждого поддерживаемого класса трафика на один из виртуальных каналов, т. е. одному VC может быть назначено несколько TC, но один TC не может быть назначен нескольким VC. Идентификатор виртуального канала, VC ID, содер-

жится не в пакете TLP, а в пакете DLLP. Таким образом, сопоставление пакетов TLP с виртуальными каналами по их VC ID выполняется в каждом порту на основе отображения TC в VC.

В интерфейсе PCIe используется механизм прерываний, называемый MSI (Message Signaled Interrupt — прерывания, инициируемые сообщениями), который появился в спецификации PCI 2.2. В режиме эмуляции поддерживается также обычный механизм — INTx.

Прерывание MSI инициирует запись устройством PCIe сообщения по определенному адресу памяти, на который отображаются регистры системного (I/O APIC) или локального (local APIC) контроллера прерываний. Запись сообщения генерирует прерывание с номером (вектором), содержащимся в сообщении. При этом номер прерывания может быть отрицательным числом. Расширение MSI-X позволяет задавать одному устройству несколько номеров прерываний для разных событий. Номер прерывания MSI имеет размер 16 бит, MSI-X — 32 бита (рис. 4.8).

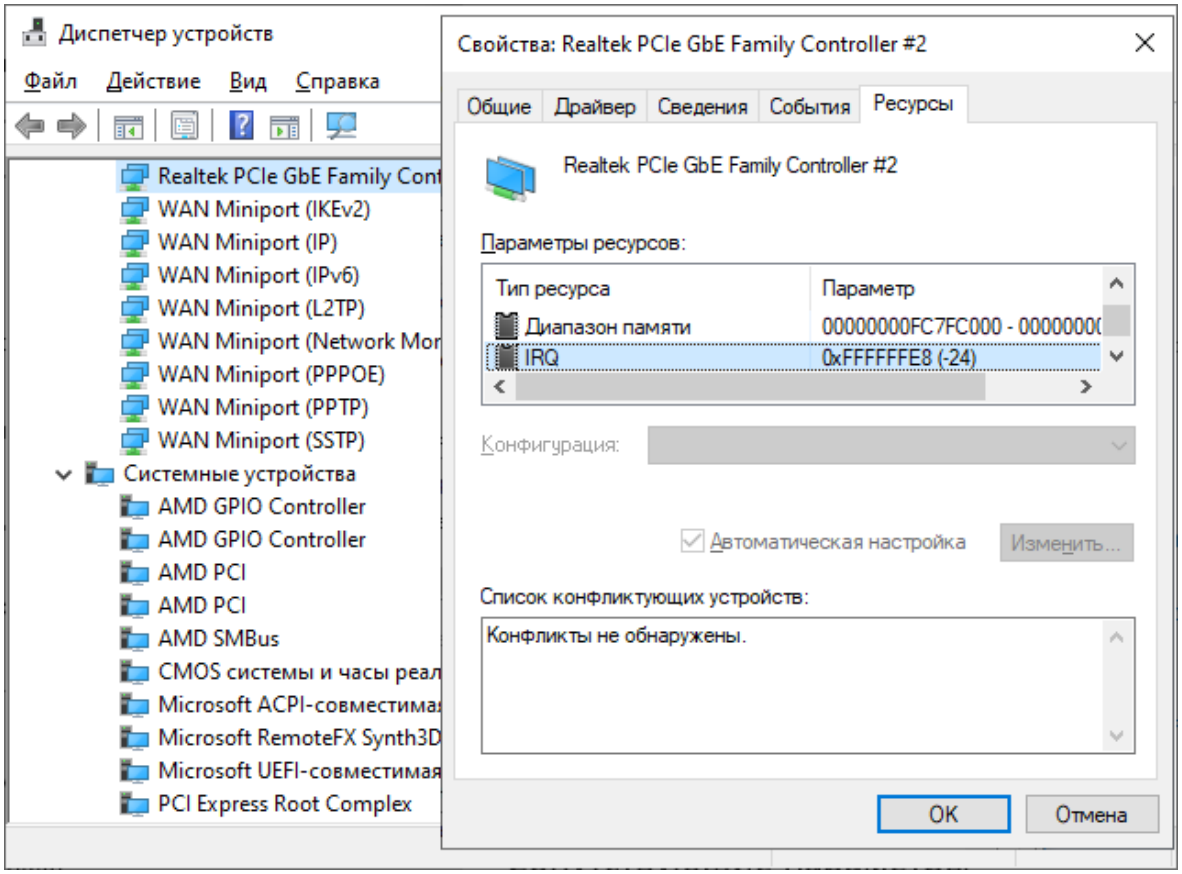


Рис. 4.8. Номер прерывания сетевого адаптера — FFFFFFE8 (–24)

Механизм MSI особенно эффективен в многопроцессорных или многоядерных компьютерных системах. Устройства, использующие этот механизм, могут на аппаратном уровне выбирать процессор (ядро) для передачи ему конкретного прерывания для обработки.

Интерфейс USB

USB (Universal Serial Bus) — стандартный интерфейс для подключения периферийных устройств к компьютеру или другому электронному устройству. Также с помощью USB можно подключить к компьютеру смартфон или модем для мобильного интернета (4G-модем). Как уже было сказано выше, USB имеет многоуровневую звездообразную, т. е. иерархическую или древовидную, топологию (рис. 4.9) [62].

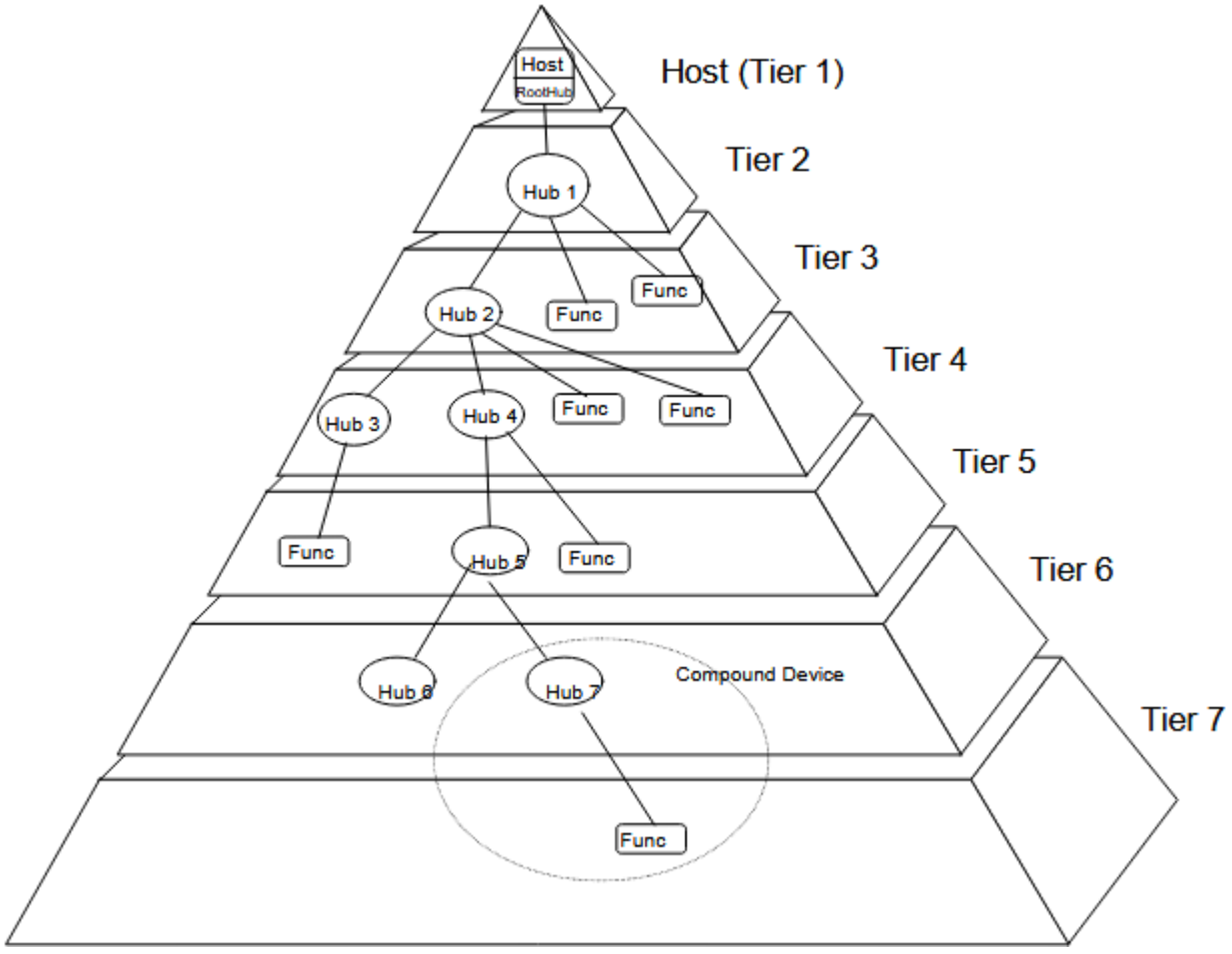


Рис. 4.9. Иерархическая топология интерфейса USB

В этой иерархии имеется два типа устройств: концентратор (хаб, hub) и функция (оконечное устройство). Концентратор является центром каждой звезды (рис. 4.10).

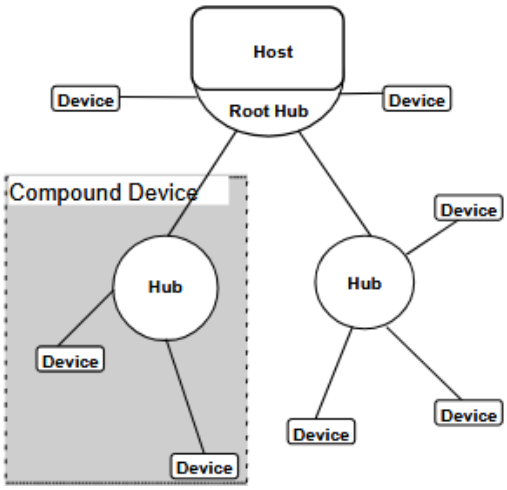


Рис. 4.10. Интерфейс USB — совокупность звезд

Хост (Host) — компьютерная система, в которой установлен хост-контроллер USB [62]. Хост-контроллер входит в состав микросхемы чипсета IOCH или PCH (см. п. 2.3).

В компьютере может быть несколько шин USB и, соответственно, несколько хост-контроллеров, каждый из которых управляет отдельной шиной (рис. 4.11).

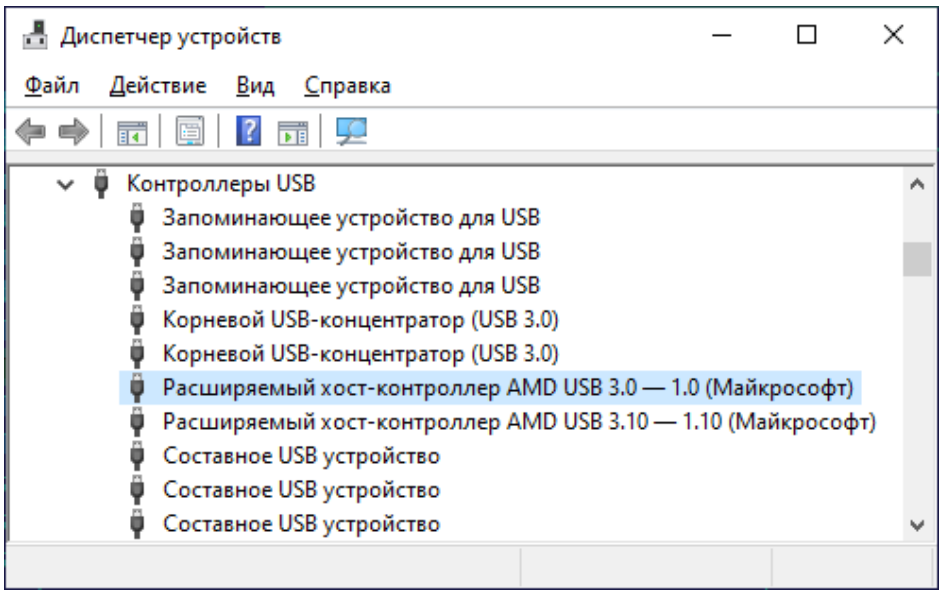


Рис. 4.11. Список устройств USB с двумя хост-контроллерами

В современных компьютерных системах контроллеры USB, как и другие устройства, подключаются через интерфейс PCIe, т. е. являются PCIe-устройствами (рис. 4.12).

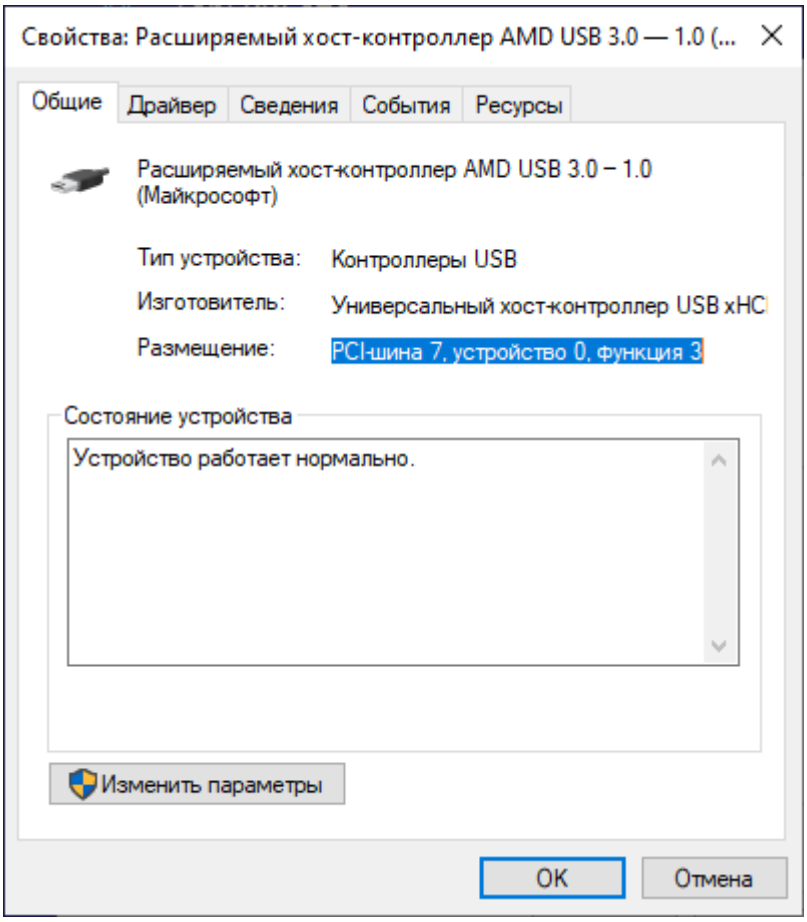


Рис. 4.12. Хост-контроллер USB подключен к шине PCIe

Это отражает тенденцию развития архитектуры компьютерных систем от специализированных шин к унифицированной PCIe-инфраструктуре для всех периферийных устройств, включая контроллеры USB, SATA (см. далее) и др.

Как PCIe-устройство, хост-контроллер USB использует механизм MSI, а точнее — его расширение MSI-X. Его ресурсы включают несколько номеров прерываний, которые являются отрицательными 32-битными числами (рис. 4.13).

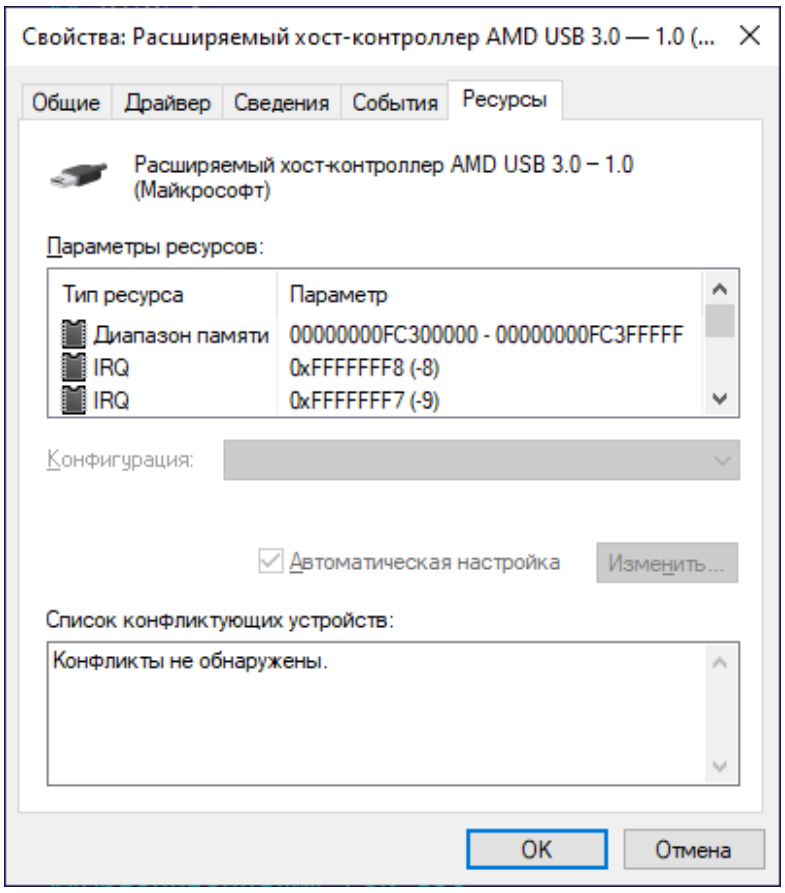


Рис. 4.13. Ресурсы хост-контроллера USB

Корневой концентратор (Root Hub) — концентратор, подключенный непосредственно к хост-контроллеру (рис. 4.14).

Максимальное количество уровней (tiers) в иерархической структуре интерфейса USB — семь (включая корневой уровень). Максимальное число устройств — 127. На всех уровнях в канале связи между хост-контроллером и любым устройством может быть максимум пять некорневых концентраторов. Бывают составные устройства (Compound Device), включающие концентратор и функцию. Это, например, USB-клавиатура с разъемами USB для подключения других устройств. Поскольку составное устройство занимает два уровня, оно не может быть подключено на седьмом уровне. На седьмом уровне могут находиться только функции.

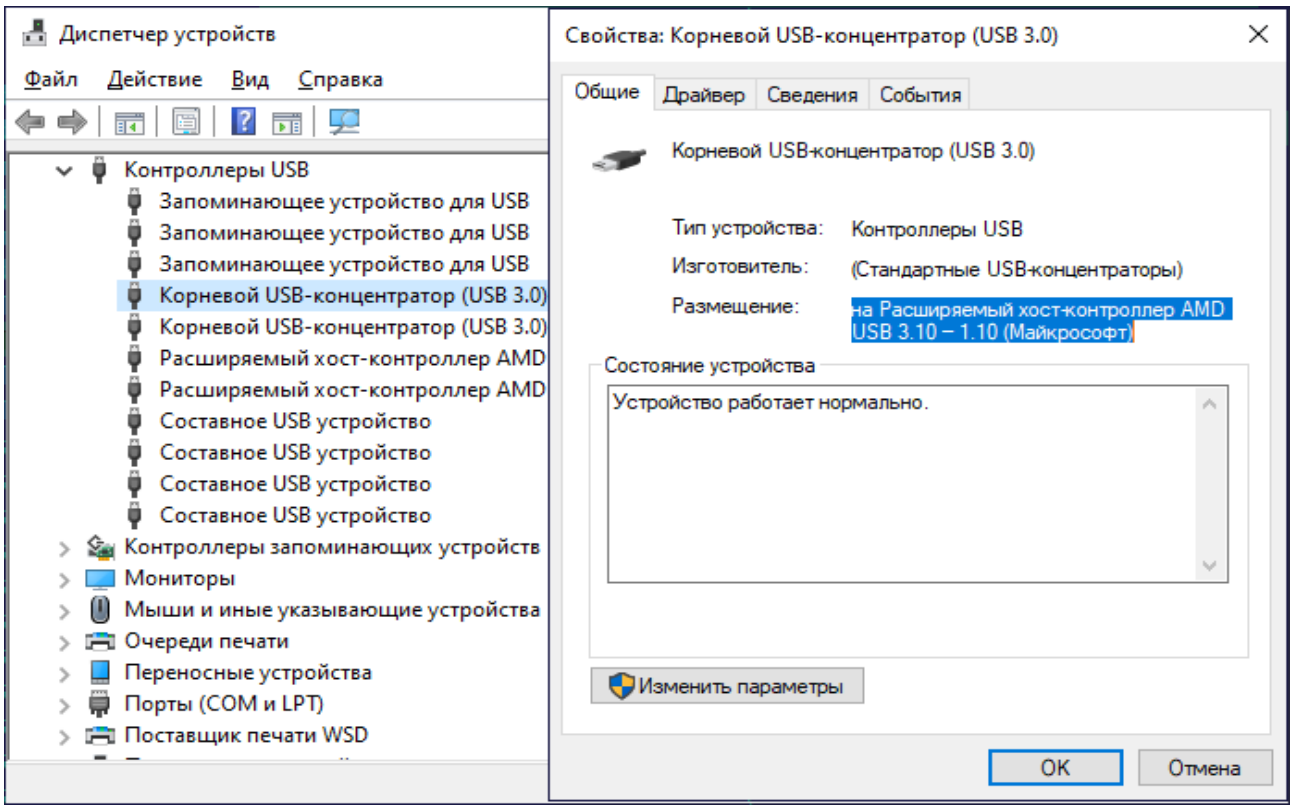


Рис. 4.14. Корневой концентратор подключен к хост-контроллеру

Пример подключения различных устройств по интерфейсу USB в компьютерной системе представлен на рисунке 4.15 [62].

В первых двух версиях интерфейса USB для подключения внешних устройств используется четырехпроводный кабель, при этом два провода (V_{BUS} и GND) используются для питания подключаемого устройства, а два другие (D+ и D-) представляют собой дифференциальную пару, как и в большинстве других современных последовательных интерфейсов (рис 4.16) [62].

Максимальная сила тока в цепи V_{BUS}, GND — 500 мА, для более мощных устройств используется отдельный источник питания.

Как видно из рисунка 4.16, данные в четырехпроводном кабеле USB передаются по одной дифференциальной паре. Таким образом, интерфейс USB версий 1.x и 2.0 работает в полудуплексном режиме.

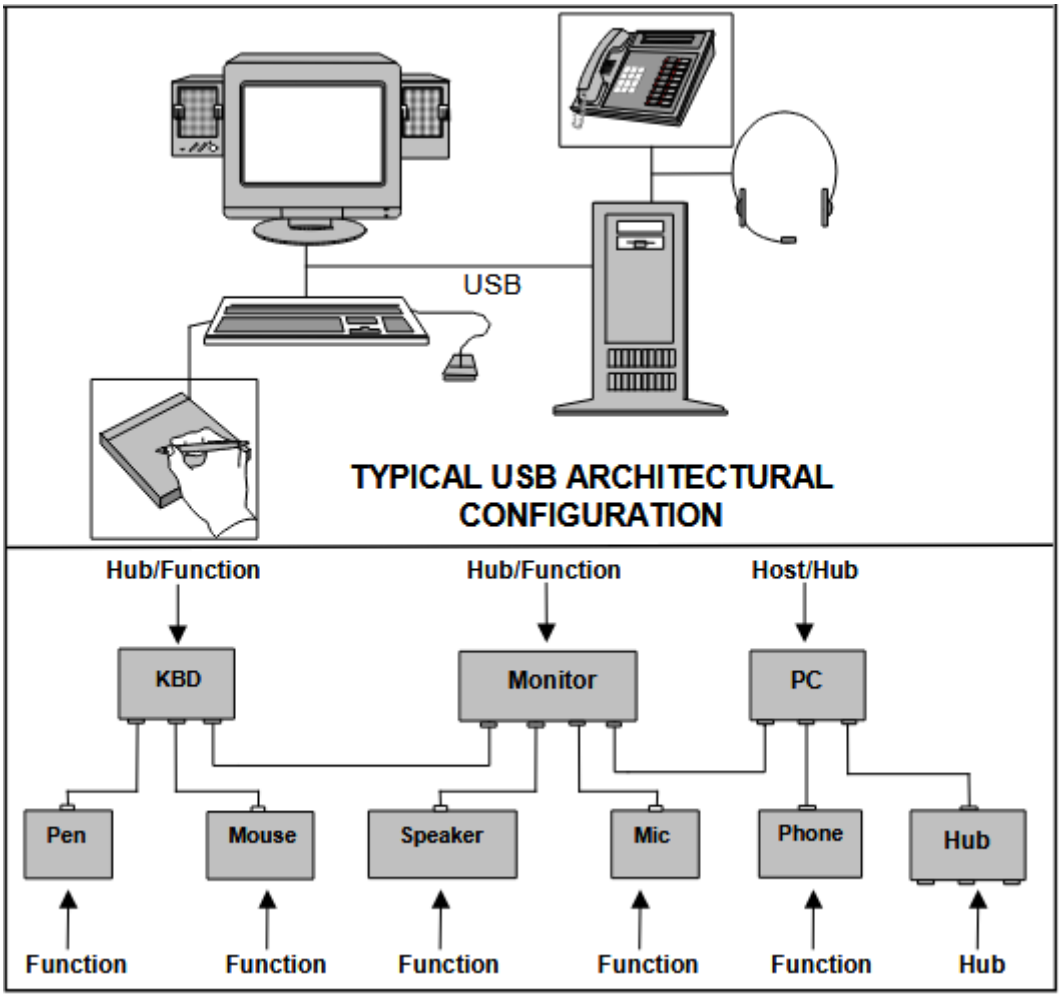


Рис. 4.15. Пример подключения периферийных устройств по интерфейсу USB

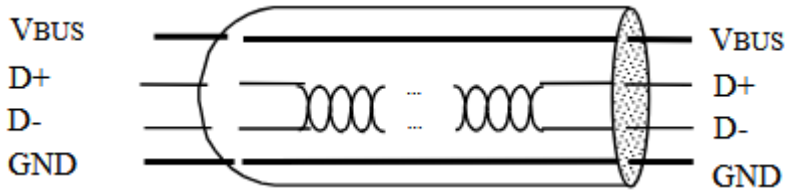


Рис. 4.16. Четырехпроводный кабель USB

Передача данных, как и в интерфейсе PCIe, осуществляется в виде пакетов. Структура пакета USB проще, чем PCIe. Но он так же, помимо собственно данных, включает адресную и управляющую информацию, в том числе сигнал синхронизации.

Существует множество режимов работы интерфейса USB, различающихся скоростью передачи:

– Low-speed (LS): 1,5 Мбит/с, USB 1.0 (1996 г.);

– Full-speed (FS): 12 Мбит/с, USB 1.1 (1998 г.);
– High-speed (HS): 480 Мбит/с, USB 2.0 (2000 г.);
– SuperSpeed (SS): 5 Гбит/с, USB 3.0 (2008 г.);
– SuperSpeed+ (SS+): 10 Гбит/с, USB 3.1 (2013 г.).

Никаких специальных названий после «Super» уже не было, но скорость передачи продолжала расти:

– USB 3.2 (2017 г.): 20 Гбит/с;
– USB4 (2019 г.): 40 Гбит/с;
– USB4 2.0 (2022 г.): 80 Гбит/с (в симметричном режиме).

В кабеле USB 3.0 — восемь линий, для передачи данных в режиме SuperSpeed используются две дополнительные дифференциальные пары: SSTX+, SSTX– и SSRX+, SSRX– (рис. 4.17) [63].

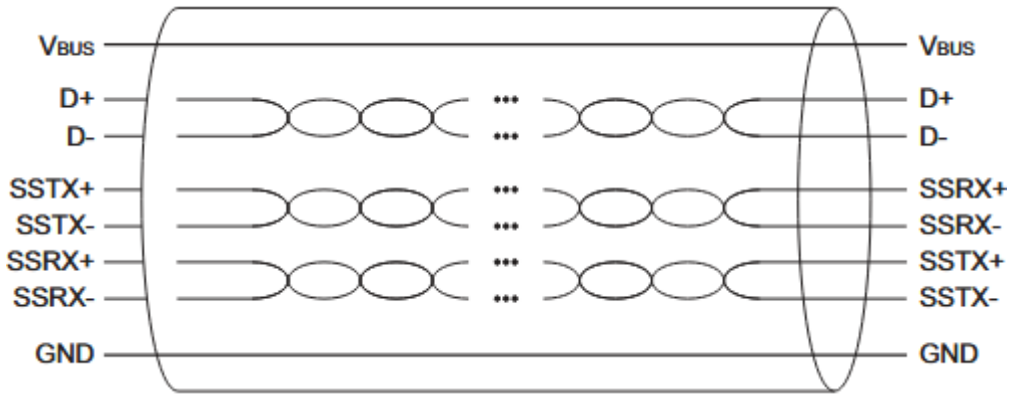


Рис. 4.17. Восьмипроводный кабель USB

Таким образом, начиная с версии 3.0, в режиме SuperSpeed интерфейс USB работает в полнодуплексном режиме. Еще одно изменение — максимальная сила тока в цепи питания увеличена до 900 мА.

С точки зрения взаимодействия компонентов интерфейса USB в нем так же, как и в PCIe, выделяется множество уровней. В режиме SuperSpeed это протокольный уровень (Protocol Layer), канальный уровень (Link Layer) и физический уровень (Physical Layer) [62].

В одной системе могут присутствовать устройства, поддерживающие различные версии интерфейса, и работающие, соответственно, в различных скоростных режимах (рис. 4.18, 4.19) [62, 63].

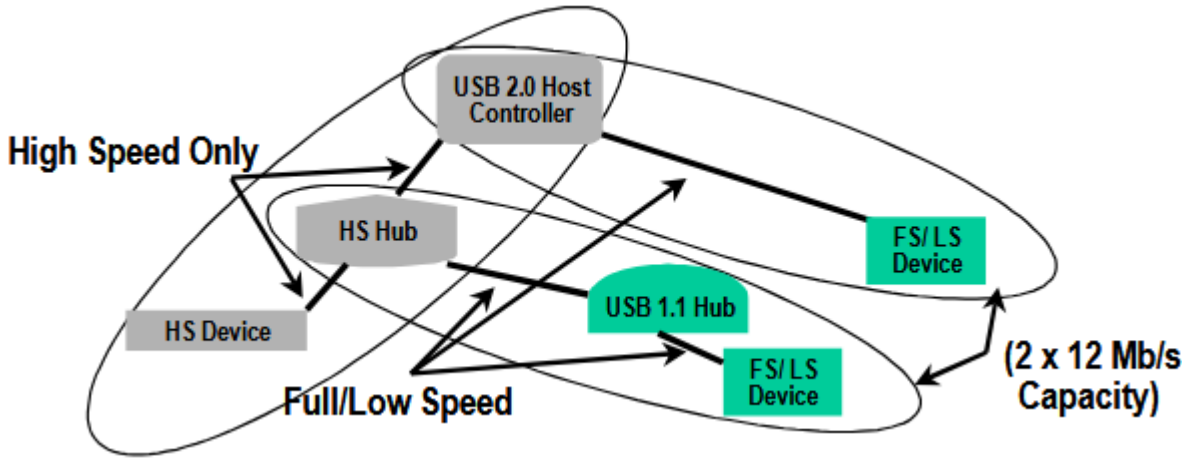


Рис. 4.18. FS- и LS-устройства в HS-системе

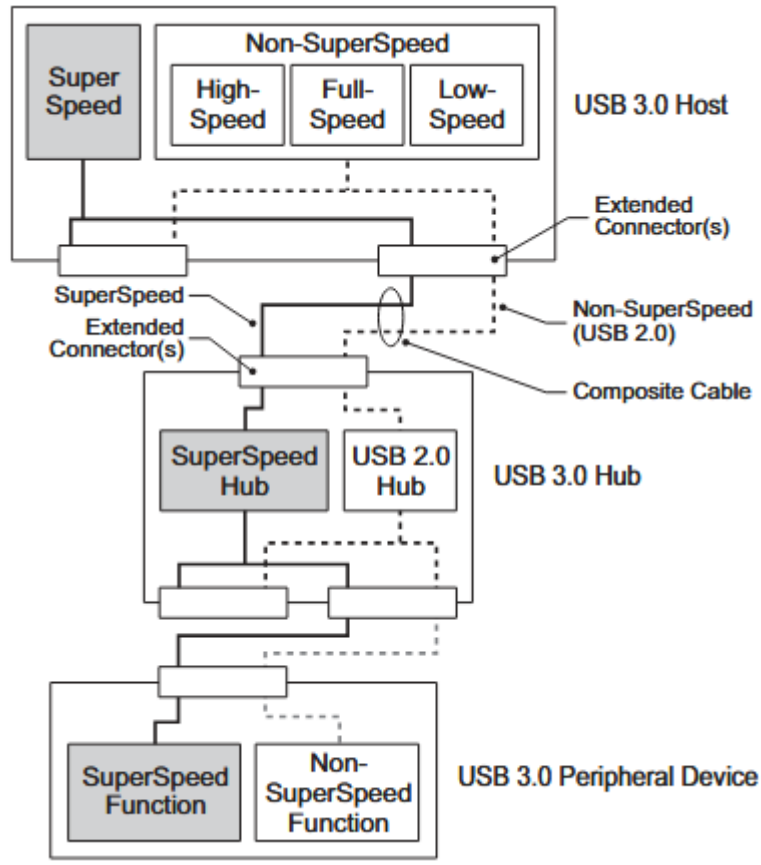


Рис. 4.19. Использование различных версий и режимов USB в одной системе

SATA (Serial ATA) — последовательный интерфейс для подключения накопителей (дисков), т. е. устройств внешней памяти. SATA разработан для замены параллельного интерфейса ATA (AT (Advanced Technology) Attachment), который после появления SATA стал называться PATA (Parallel ATA). Спецификация первой версии интерфейса SATA была выпущена в 2001 году. В отличие от PATA, в котором одним кабелем (шлейфом) могло быть подключено два диска (рис. 4.20), SATA для каждого устройства использует отдельный кабель (рис. 4.21) [64, 65].

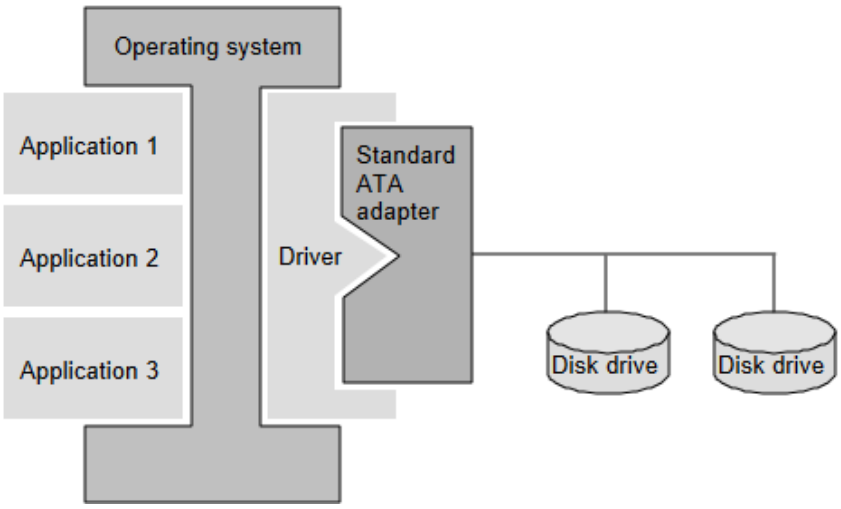


Рис. 4.20. Подключение дисков с помощью интерфейса PATA

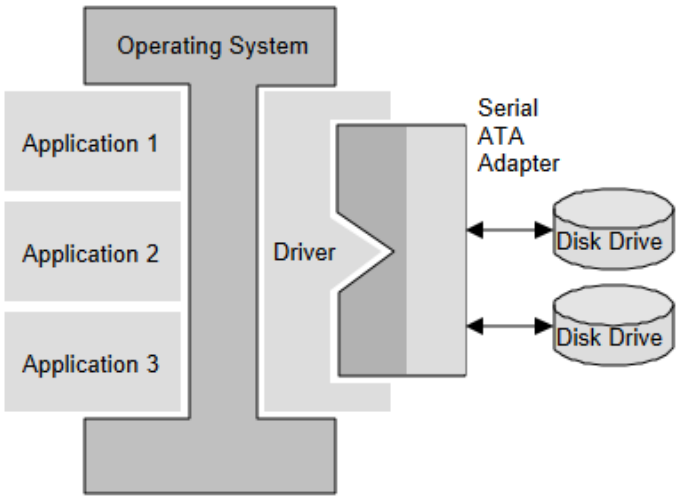


Рис. 4.21. Подключение дисков с помощью интерфейса SATA

В сигнальных кабелях SATA используются семиконктакные разъемы: три земли (G, Ground) и две дифференциальные пары (A+, A- и B+, B-) для передачи данных одновременно в обоих направлениях, т. е. в полнодуплексном режиме (рис. 4.22) [64, 65].

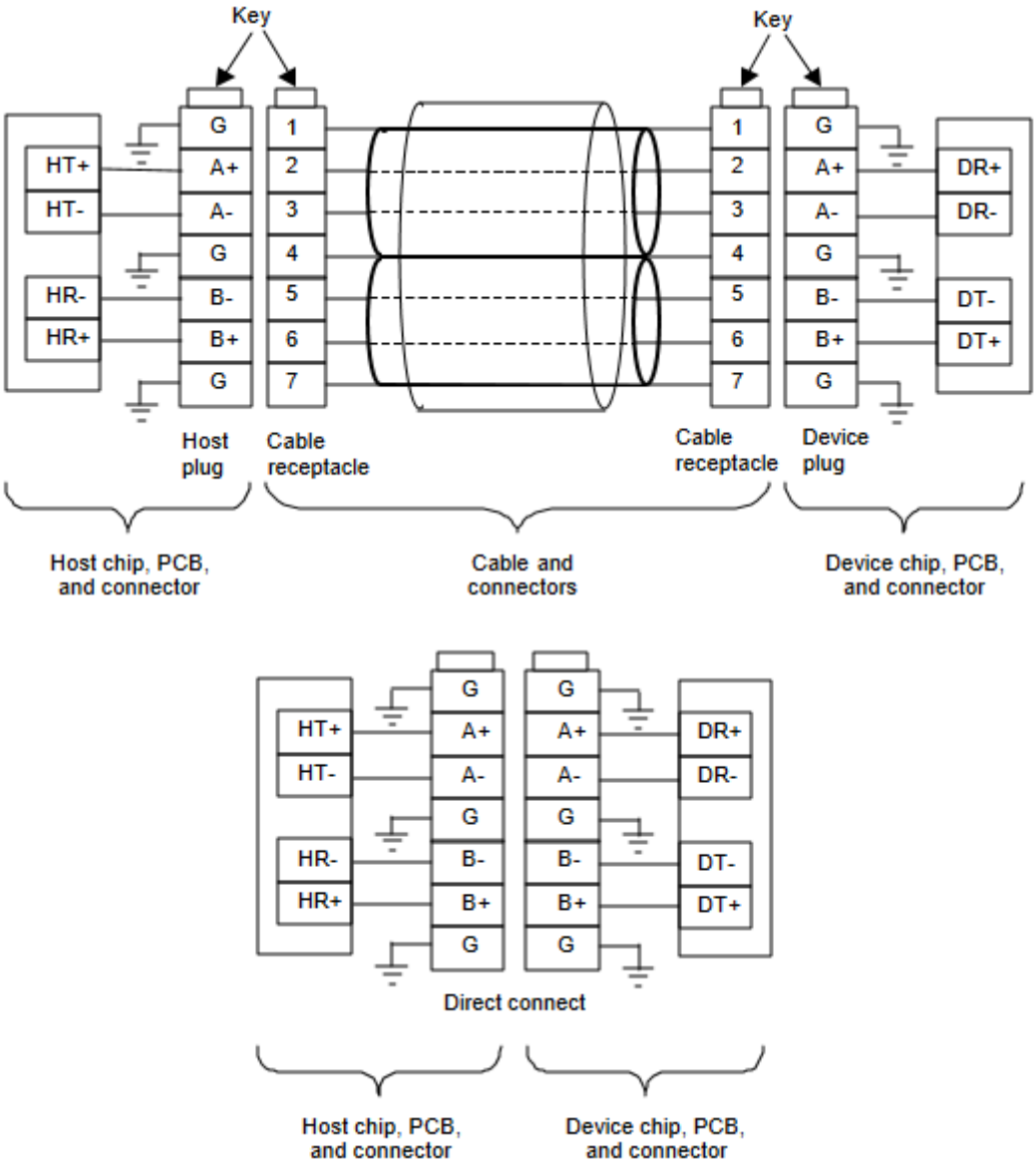


Рис. 4.22. Сигнальные линии интерфейса SATA

Питание подается по отдельному кабелю.

В структуре взаимодействия устройств выделяется четыре уровня:

- прикладной (Application layer);
- транспортный (Transport layer);
- канальный (Link layer);

– физический (Physical layer).

Как уже было сказано выше, в современных системах контроллер SATA, как и множество других контроллеров, подключается по интерфейсу PCIe (рис. 4.23).

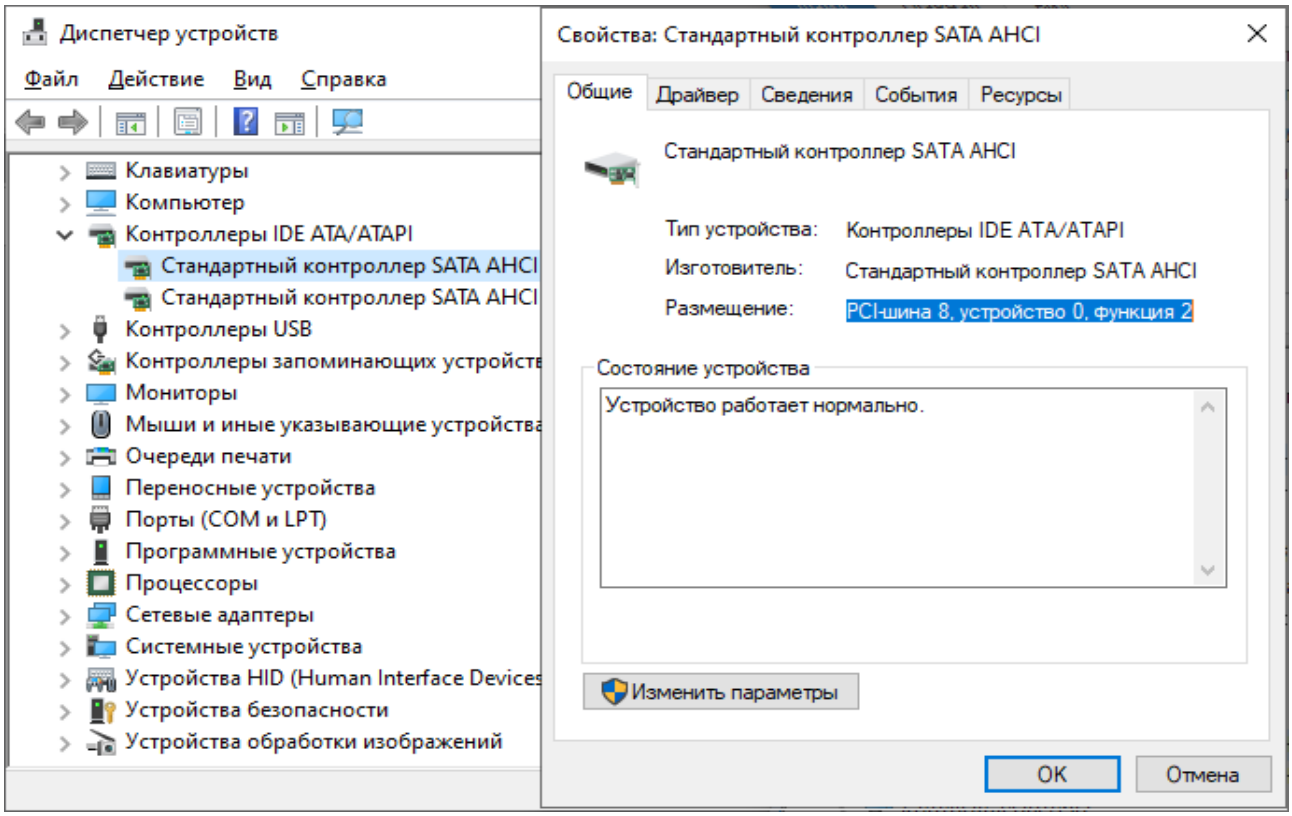


Рис. 4.23. Контроллер SATA подключен к шине PCIe

Контроллер SATA чаще всего входит в состав чипсета, но может быть и дискретным, в виде отдельной платы. Как и все PCIe-устройства, он использует механизм прерываний MSI/MSI-X (рис. 4.24).

AHCI (Advanced Host Controller Interface) — стандарт интерфейса хост-контроллера SATA, разработанный Intel.

В SATA, как и в других последовательных интерфейсах, используется пакетная передача данных, что отражено в названии еще одного стандарта интерфейса для подключения устройств хранения данных — ATAPI (AT Attachment Packet Interface) (см. рис. 4.23).

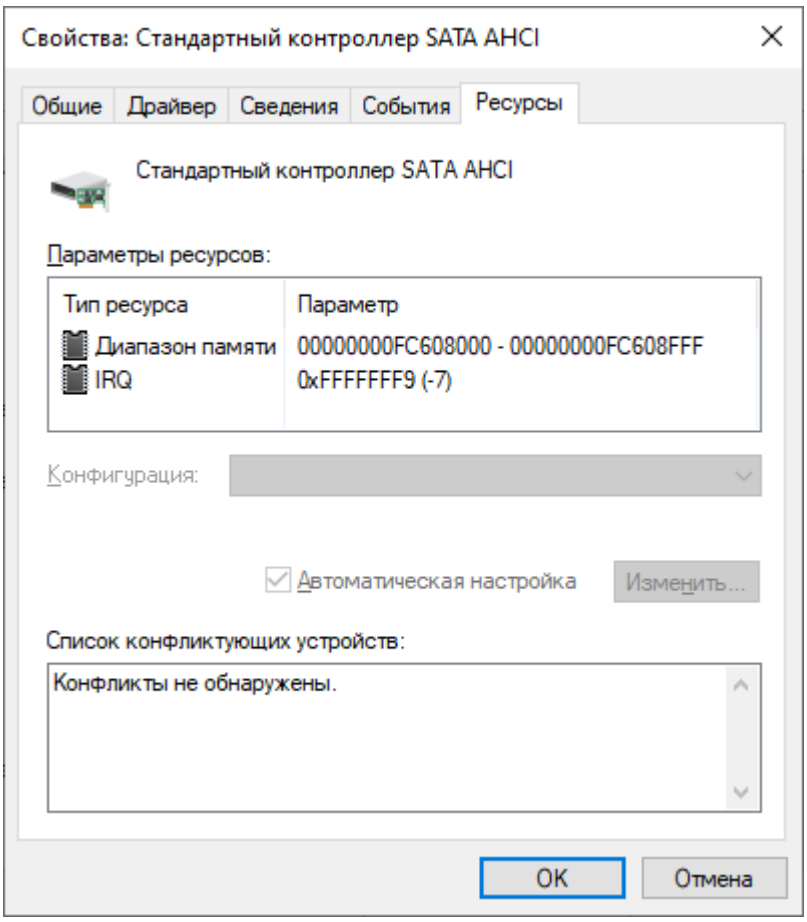


Рис. 4.24. Ресурсы контроллера SATA AHCI

Хотя на физическом уровне SATA — полнодуплексный интерфейс, из-за особенностей протокола он работает в полудуплексном режиме со следующими значениями скорости передачи:

- SATA 1.x (с 2003 г.): 1,5 Гбит/с;
- SATA 2.x (с 2004 г.): 3 Гбит/с;
- SATA 3.x (с 2009 г.): 6 Гбит/с.

В SATA используется кодирование 8b/10b. Соответственно, максимальная пропускная способность в одну сторону:

- SATA 1.x: 1,5 Гбит/с * 8/10 = 1,2 Гбит/с = 150 МБ/с;
- SATA 2.x: 3 Гбит/с * 8/10 = 2,4 Гбит/с = 300 МБ/с;
- SATA 3.x: 6 Гбит/с * 8/10 = 4,8 Гбит/с = 600 МБ/с.

Имеются разновидности интерфейса SATA:

- External SATA (eSATA) — интерфейс для подключения внешних накопителей;

– SATA Express (SATAe) — интерфейс для подключения накопителей двух типов: SATA и PCIe.

SATAe поддерживает два протокола:

– AHCI — для режима SATA;

– NVMe (Non-Volatile Memory Express) — для режима PCIe.

Частью протоколов SATA является технология S.M.A.R.T. (см. п. 2.2) (рис. 4.25).

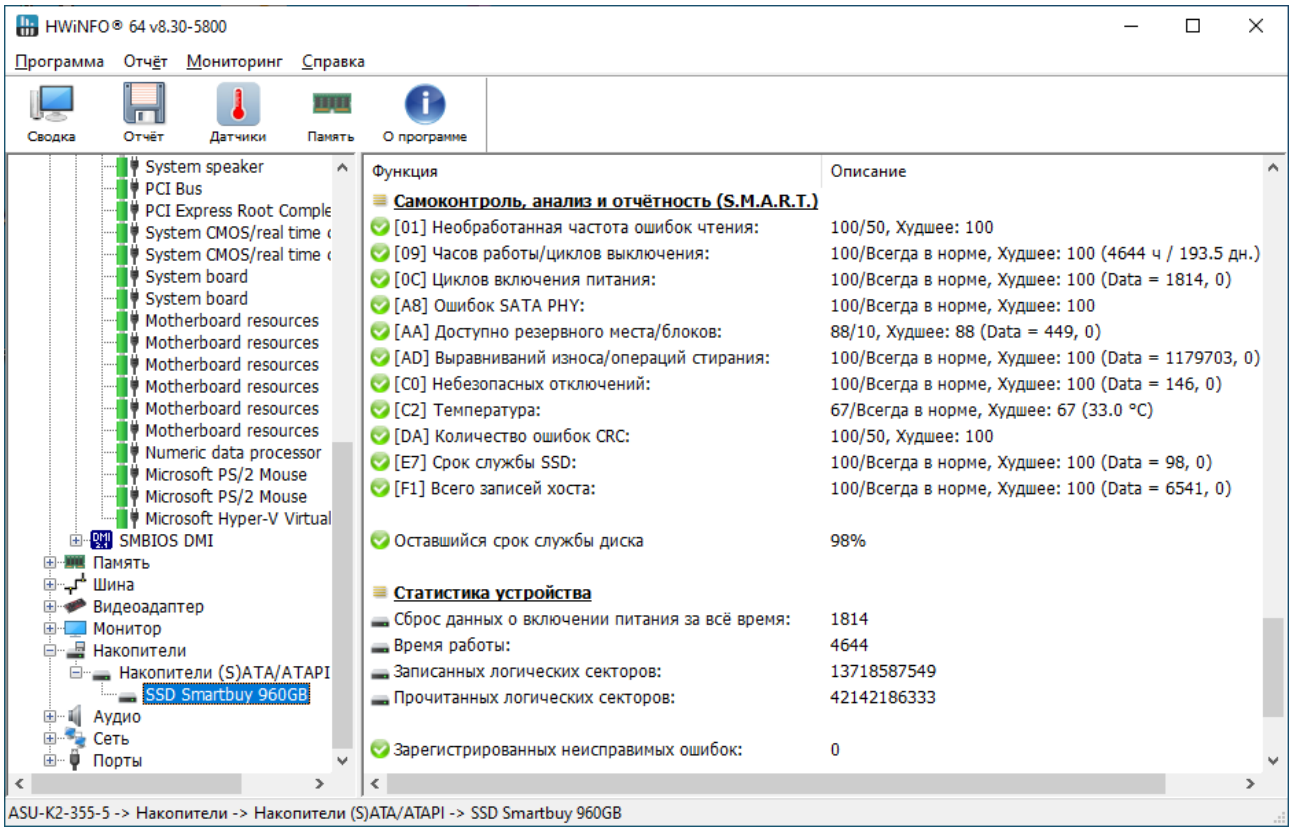


Рис. 4.25. Данные S.M.A.R.T. в программе HWiNFO

SAS (Serial Attached SCSI) — более производительный по сравнению с SATA последовательный интерфейс для подключения накопителей, использующийся в серверных системах. Он был разработан для замены параллельного интерфейса SCSI (Small Computer System Interface) и основывается на терминологии и наборах команд SCSI.

SAS работает в полнодуплексном режиме на всех уровнях и характеризуется следующими значениями скорости передачи:

- SAS-1 (2004 г.): 3 Гбит/с;
- SAS-2 (2009 г.): 6 Гбит/с;
- SAS-3 (2013 г.): 12 Гбит/с;
- SAS-4 (24G) (2017 г.): 22,5 Гбит/с;
- SAS-5 (разрабатывается): 45 Гбит/с.

HBA-контроллер интерфейса SAS (Host Bus Adapter, адаптер шины хоста), как и другие контроллеры, может быть интегрированным или дискретным. В любом случае он подключается по интерфейсу PCIe и, соответственно, использует механизм MSI/MSI-X (рис. 4.26).

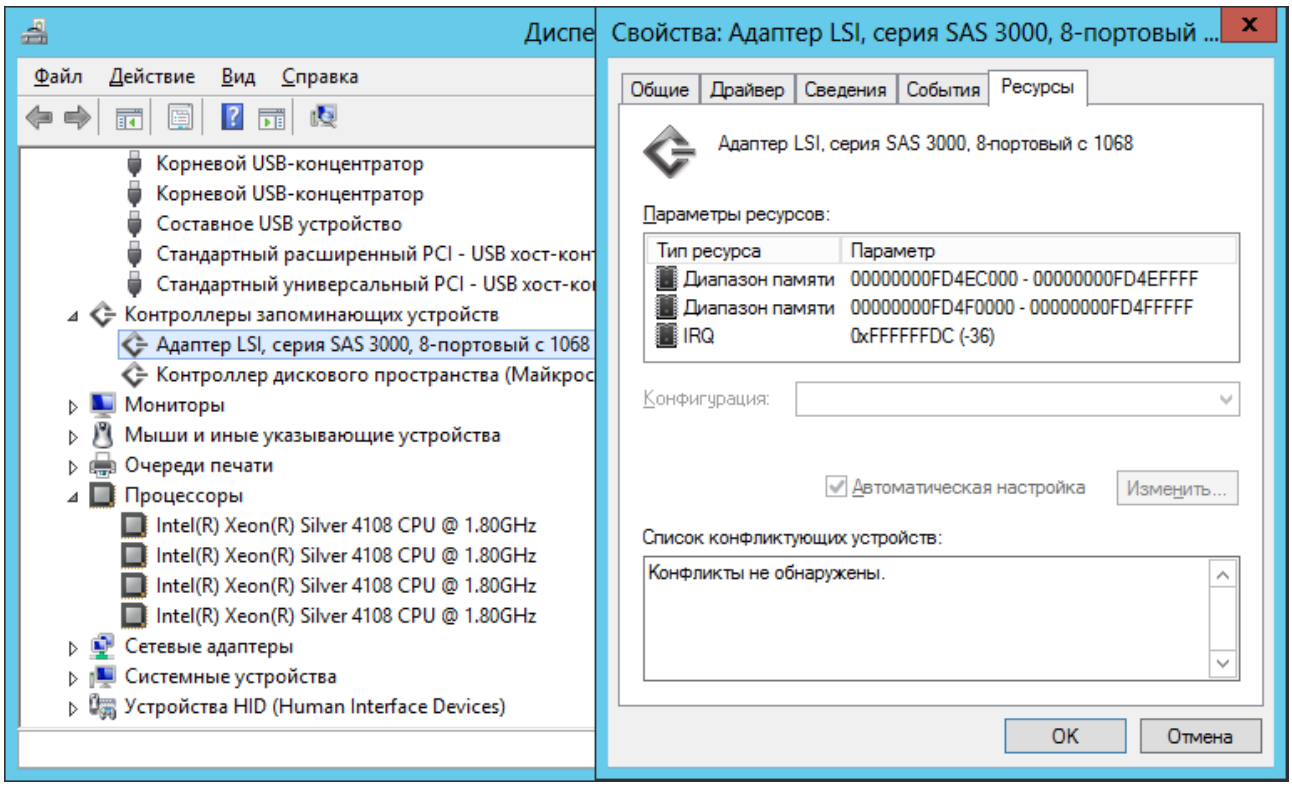


Рис. 4.26. Ресурсы HBA-контроллера SAS

SAS обратно совместим с интерфейсом SATA: устройства SATA 2.x и SATA 3.x могут быть подключены к контроллеру SAS, но не наоборот.

Как и большинство других современных последовательных интерфейсов, SATA и SAS имеют топологию «точка-точка», но в них

могут быть использованы коммутационные устройства, которые в SAS называются расширителями (Expander) (рис. 4.27) [66].

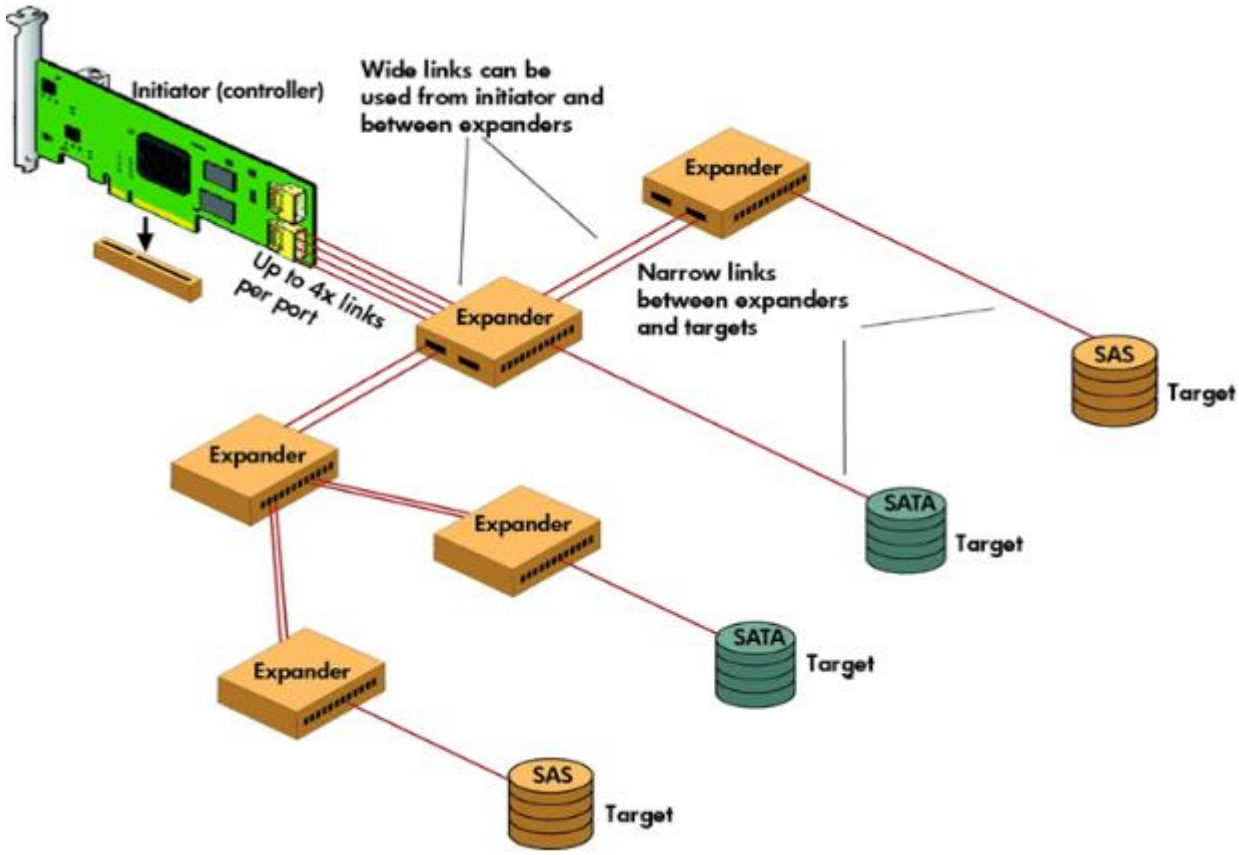


Рис. 4.27. Пример соединения устройств (Target) SATA/SAS с контроллером (Initiator) через расширители (Expander)

NVMe (Non-Volatile Memory Express) — интерфейс, разработанный специально для подключения SSD-накопителей непосредственно через шину PCIe. Спецификация NVMe 1.0 появилась в 2011 г. Интерфейс NVM Express (NVMe) позволяет программному обеспечению хоста взаимодействовать с подсистемой энергонезависимой памяти NVM [67].

SSD-накопители можно подключать по интерфейсам SATA или SAS, но эти интерфейсы изначально разрабатывались для HDD-накопителей, и они ограничивают пропускную способность, которая может быть достигнута с использованием преимуществ флеш-памяти. Интерфейс NVMe позволяет накопителям напрямую взаимодействовать с центральным процессором через высокоскоростную шину

PCIe, что снижает задержки и увеличивает пропускную способность за счет использования высокого внутреннего параллелизма твердотельных накопителей. Для этого в NVMe применяется новый набор команд и механизм очередей, оптимизированный для работы с многоядерными процессорами. Логика NVMe физически реализуется контроллером NVMe, подключенным к шине PCIe (рис. 4.28).

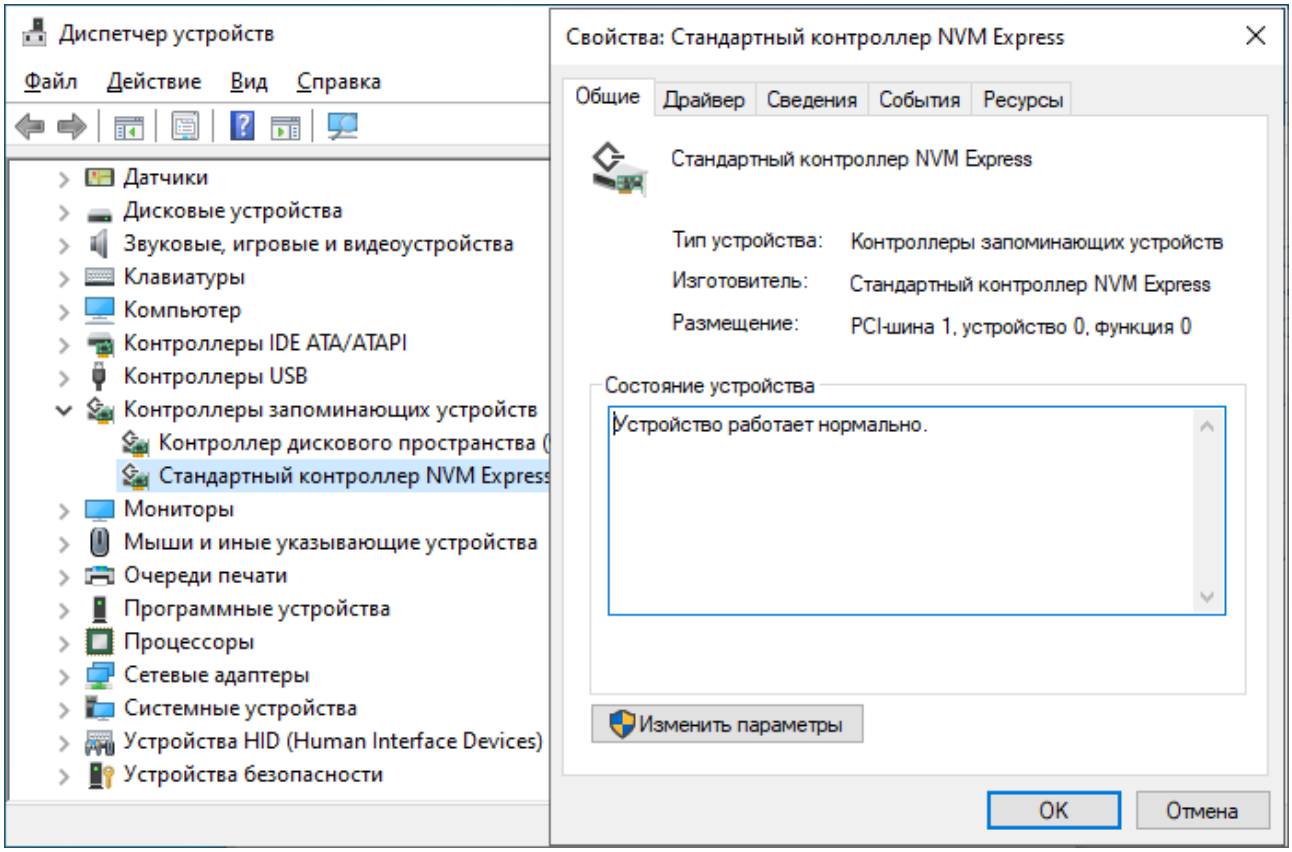


Рис. 4.28. Контроллер NVMe подключен к шине PCIe

В основе протокола NVMe лежат очереди команд. Контроллер управляет командами и планирует их выполнение, оптимизируя использование внутренних ячеек флеш-памяти SSD-накопителя. Имеется два вида очередей: очереди отправки (Submission Queues, SQ) и завершения (Completion Queues, CQ). Контроллер подает накопителю команды на чтение или запись в SQ, а накопитель отправляет отчет об их выполнении через CQ. В NVMe может быть использовано до 64К очередей и до 64К команд в каждой очереди. Очереди команд

могут быть распределены по разным ядрам процессора, что существенно повышает производительность в многозадачных системах.

В дальнейшем интерфейс NVMe был расширен для всех решений хранения данных, подключаемых с использованием различных интерфейсов и сетей передачи данных, включая Ethernet, InfiniBand и Fibre Channel (рис. 4.29) [67].

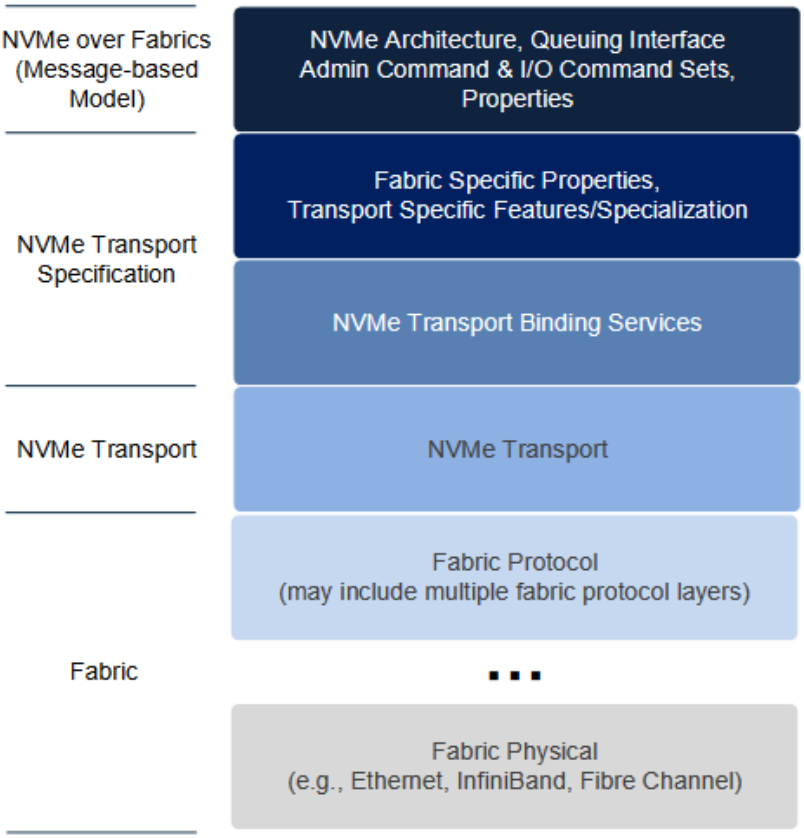


Рис. 4.29. Многоуровневая структура NVMe-oF

Для этого используется протокол NVMe over Fabrics (NVMe-oF), который расширяет возможности интерфейса NVMe на всю сеть (Fabric), позволяя каждому узлу (серверу) обращаться к удаленным SSD-накопителям так, как если бы они были подключены локально по шине PCIe.

Пропускная способность NVMe зависит от характеристик используемой среды, версии интерфейса PCIe, но в целом она на порядок выше, чем у интерфейсов SATA или SAS.

Сетевые интерфейсы. Ethernet

Сетевой интерфейс — это средства подключения компьютера (или устройства) к компьютерной сети. Существует множество типов сетей и сетевых интерфейсов. Практически во всех компьютерах имеется адаптер Ethernet (сетевая карта, сетевая плата, Network Interface Card, NIC), и он подключен к шине PCIe (см. рис. 4.8).

Ethernet — это тип локальной компьютерной сети (Local Area Network, LAN), разработанный совместно корпорациями DEC (Digital Equipment Corporation), Intel и Xerox (рис. 4.30) [68].

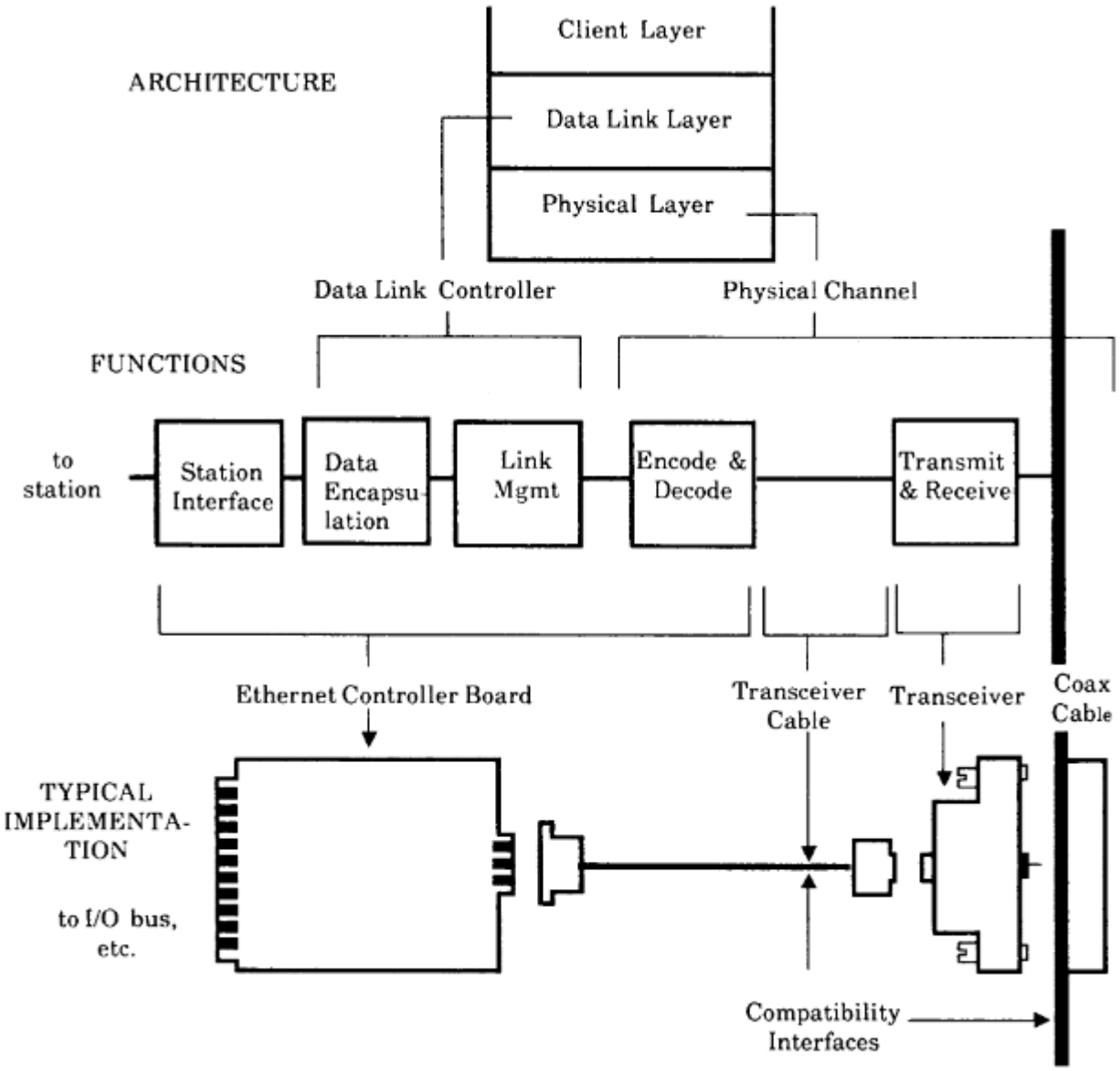


Рис. 4.30. Архитектура Ethernet. Концепция и пример реализации

В обычных сетевых платах основной микросхемой является специализированный контроллер (MAC-контроллер). В сетевых адаптерах для высокопроизводительных систем используются процессоры обработки данных (DPU) или микросхемы, включающие и контроллер, и процессор. Например, основой платы NVIDIA ConnectX-6 является микросхема Mellanox ConnectX-6 — гибридное устройство категории SmartNIC/DPU, программируемый сетевой контроллер, который содержит как специализированные аппаратные блоки, так и процессорные ядра DPU [69].

Технологию Ethernet регламентирует семейство стандартов, определяющих физический (Physical Layer, L1) и канальный (Data Link Layer, L2) уровни модели OSI (Open Systems Interconnection). OSI — это концептуальная модель взаимодействия устройств в локальных и глобальных компьютерных сетях, разработанная международной организацией по стандартизации (ISO) в 1984 году. На модели OSI базируются многие сетевые протоколы и технологии, в том числе стек протоколов TCP/IP (Transmission Control Protocol/Internet Protocol), который используется для передачи данных в Интернете и других компьютерных сетях.

Данные между узлами сети Ethernet передаются в виде пакетов, средой передачи может быть витая пара, оптоволокно или радиоканал. Ранее использовались и коаксикальные кабели (см. рис. 4.30). Логическая топология сети Ethernet — «шина». При использовании коаксиальных кабелей и на физическом уровне Ethernet имел шинную топологию. В настоящее время на физическом уровне Ethernet имеет звездообразную коммутируемую архитектуру (Switched Fabric), в которой узлы соединяются между собой через коммутаторы (Switches).

Переход к звездообразной топологии начался с использованием концентраторов или хабов (Hubs). Хабы работают на физическом уровне OSI, они просто передают сигнал на все порты. Таким образом, среда является разделяемой, с возможностью возникновения коллизий. Изначально в основе работы сети Ethernet на канальном уровне лежала технология CSMA/CD (Carrier Sense Multiple Access

with Collision Detection, множественный доступ с прослушиванием несущей частоты и обнаружением коллизий). CSMA/CD заключается в том, что устройства, подключенные к сети, прослушивают среду передачи, проверяя наличие сигнала. Если среда свободна, устройство начинает передачу. Если одновременно начинают передавать два устройства, происходит коллизия. Устройства обнаруживают коллизию и прекращают передачу, затем они ожидают некоторое случайное время и повторяют передачу. Таким образом, в CSMA/CD реализуется полудуплексный режим.

В современных сетях Ethernet используются полнодуплексные соединения. Например, в витой паре передача осуществляется по одной дифференциальной паре проводов, прием — по другой. Использование коммутаторов практически исключает вероятность возникновения конфликтов. В этом случае технология CSMA/CD не нужна, но по-прежнему поддерживается в полудуплексных соединениях и для обеспечения обратной совместимости. Коммутаторы работают на канальном уровне OSI, они используют таблицу MAC-адресов при передаче сигналов, обеспечивая полный дуплекс на каждом порту и, соответственно, безколлизионную среду.

MAC-адрес (Media Access Control) или физический адрес — это уникальный идентификатор устройства в сети. Он имеет размер шесть байт, присваивается каждому устройству на этапе производства и записывается в его постоянную память. В Windows MAC-адрес можно посмотреть через «Параметры», «Сеть и Интернет» или в командной строке с использованием команды ipconfig /all (рис. 4.31).

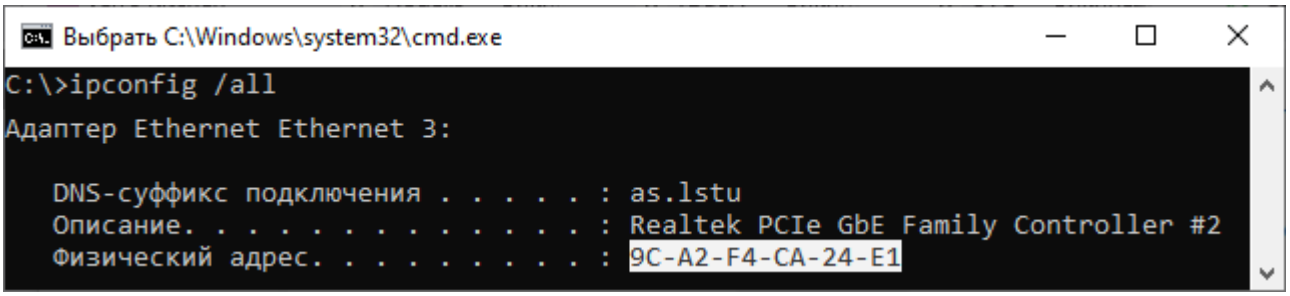


Рис. 4.31. MAC-адрес адаптера Ethernet

Коммутаторы уровня 3 (L3-коммутаторы) сочетают функции коммутатора второго уровня (L2) и маршрутизатора. Они работают на сетевом уровне (Network Layer, L3) модели OSI и осуществляют передачу данных на основе IP-адресов, но в пределах локальной сети. Для соединения с глобальными сетями, Интернетом необходимо такое устройство, как маршрутизатор.

Маршрутизатор (Router) соединяет разные типы сетей: локальные и глобальные, домашние и внешние. Например, в домашней сети он связывает все устройства в доме с интернет-провайдером. Маршрутизатор, как и L3-коммутатор, работает на сетевом уровне модели OSI. Он принимает пакеты от устройств внутри сети или от других маршрутизаторов, проверяет IP-адреса источника и назначения, затем на основе таблиц маршрутизации определяет наиболее эффективный путь и пересылает пакеты далее в соответствии с выбранным маршрутом на пути к пункту назначения. IP-адрес (от Internet Protocol) — это уникальный числовой идентификатор устройства в компьютерной сети, работающей по протоколу IP. В версии протокола IPv4 IP-адрес имеет длину 4 байта, а в версии протокола IPv6 — 16 байт. IP-адрес может назначаться компьютеру автоматически (динамически) с помощью DHCP-сервера (Dynamic Host Configuration Protocol) или вручную (статически). Его можно посмотреть с помощью той же команды ipconfig (рис. 4.32).

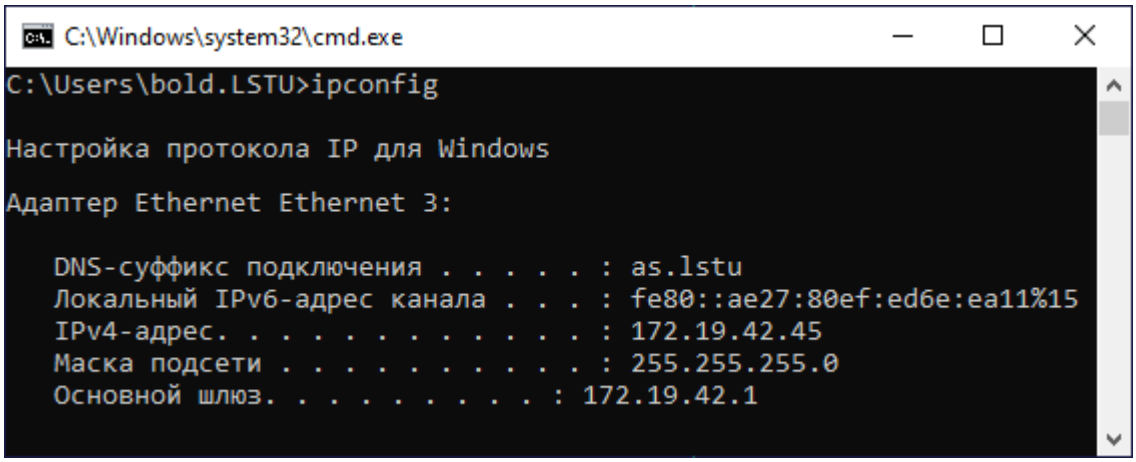


Рис. 4.32. IP-адрес адаптера Ethernet

Характеристики некоторых стандартов Ethernet представлены в табл. 4.3.

Таблица 4.3

Стандарты Ethernet и их характеристики

Стандарт	Скорость передачи	Год	Кабель	Максимальное расстояние
10BASE5	10 Мбит/с	1980	толстый коаксиал (RG-8)	500 м
10BASE2	10 Мбит/с	1985	тонкий коаксиал (RG-58)	185 м
10BASE-T	10 Мбит/с	1990	UTP Cat3	100 м
100BASE-TX	100 Мбит/с	1995	UTP Cat5	100 м
1000BASE-T	1 Гбит/с	1999	UTP Cat5e	100 м
10GBASE-T	10 Гбит/с	2006	UTP Cat6a	100 м
40GBASE-T	40 Гбит/с	2016	UTP Cat8	30 м

UTP (Unshielded Twisted Pair) — неэкранированная витая пара. Cat — категория.

В настоящее время существует множество реализаций 100-гигабитного Ethernet, и разрабатывается терабитный Ethernet.

4.3. ОСНОВНЫЕ ПОДСИСТЕМЫ ВВОДА-ВЫВОДА ПЭВМ

Ввод с клавиатуры

Основным элементом подсистемы клавиатуры является контроллер клавиатуры. Ранее контроллер клавиатуры выполнялся на отдельной микросхеме универсального периферийного интерфейса (UPI, Universal Peripheral Interface) 8042 (8742). Сейчас он входит в состав микросхем более высокой степени интеграции, которые имеют общее название контроллера ввода-вывода, вместе с последовательным и параллельным портом и другими интерфейсами (см. рис. 2.13). Как уже было отмечено выше, контроллер клавиатуры реализует также интерфейс с мышью. Кроме того, он используется для

управления адресной линией A20 (сигнал GATEA20) для снятия ограничения на размер адресного пространства процессора в 1 Мбайт (1 М = 2^{20}) (рис. 4.33) [44].

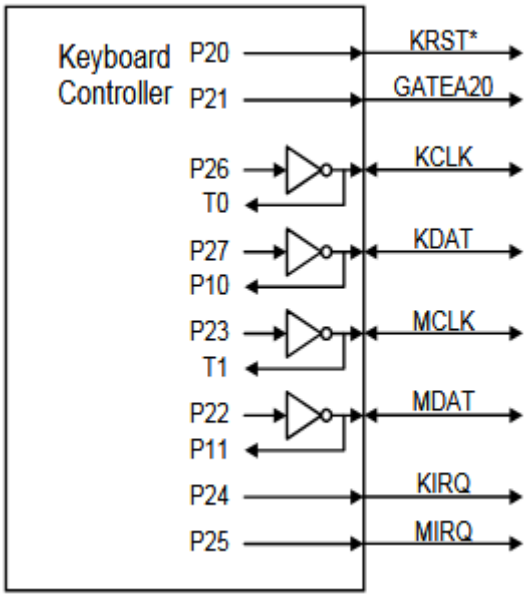


Рис. 4.33. Линии контроллера клавиатуры в составе контроллера ввода-вывода

И клавиатура, и мышь подключаются к контроллеру клавиатуры по последовательному интерфейсу. Сейчас это в основном интерфейс USB. Ранее использовались клавиатуры PS/2 с разъемом MiniDIN-6 и AT с разъемом DIN-5. По одной из линий интерфейса передаются данные (KDAT для клавиатуры, MDAT для мыши), по другой — тактовые импульсы (KCLK, MCLK).

Контроллер клавиатуры генерирует сигналы прерывания клавиатуры и мыши (KIRQ, MIRQ), которые поступают на контроллер прерываний. Номер прерывания клавиатуры, как уже было сказано выше, — IRQ 1 (рис. 4.34).

В состав самой клавиатуры входят (рис. 4.35) [70]:

− микро-ЭВМ (микроконтроллер);

− наборное поле (матрица) клавиш;

− светодиодные индикаторы.

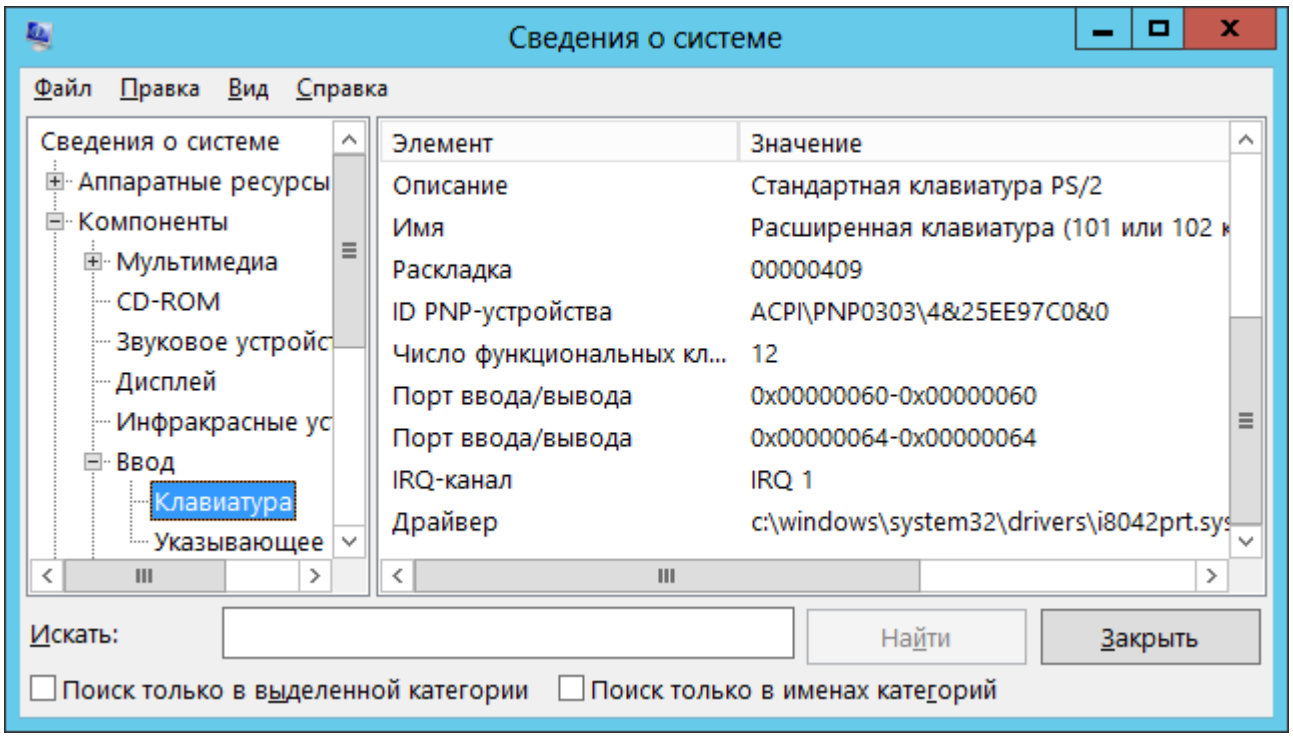


Рис. 4.34. Свойства клавиатуры в программе «Сведения о системе» (msinfo32)

Микроконтроллер клавиатуры иногда называют контроллером клавиатуры. Не надо путать его с контроллером клавиатуры, находящимся в системном блоке в составе контроллера ввода-вывода (см. рис. 2.13, 2.14). В состав микроконтроллера клавиатуры входят (рис. 4.36) [70]:

– микропроцессор (CPU);
– постоянное запоминающее устройство (ROM);
– оперативное запоминающее устройство (SRAM);
– тактовый генератор;
– порты ввода-вывода.

В состав микроконтроллера клавиатуры могут входить также таймеры, контроллер прерываний и другие устройства, в зависимости от производителя и модели микроконтроллера.

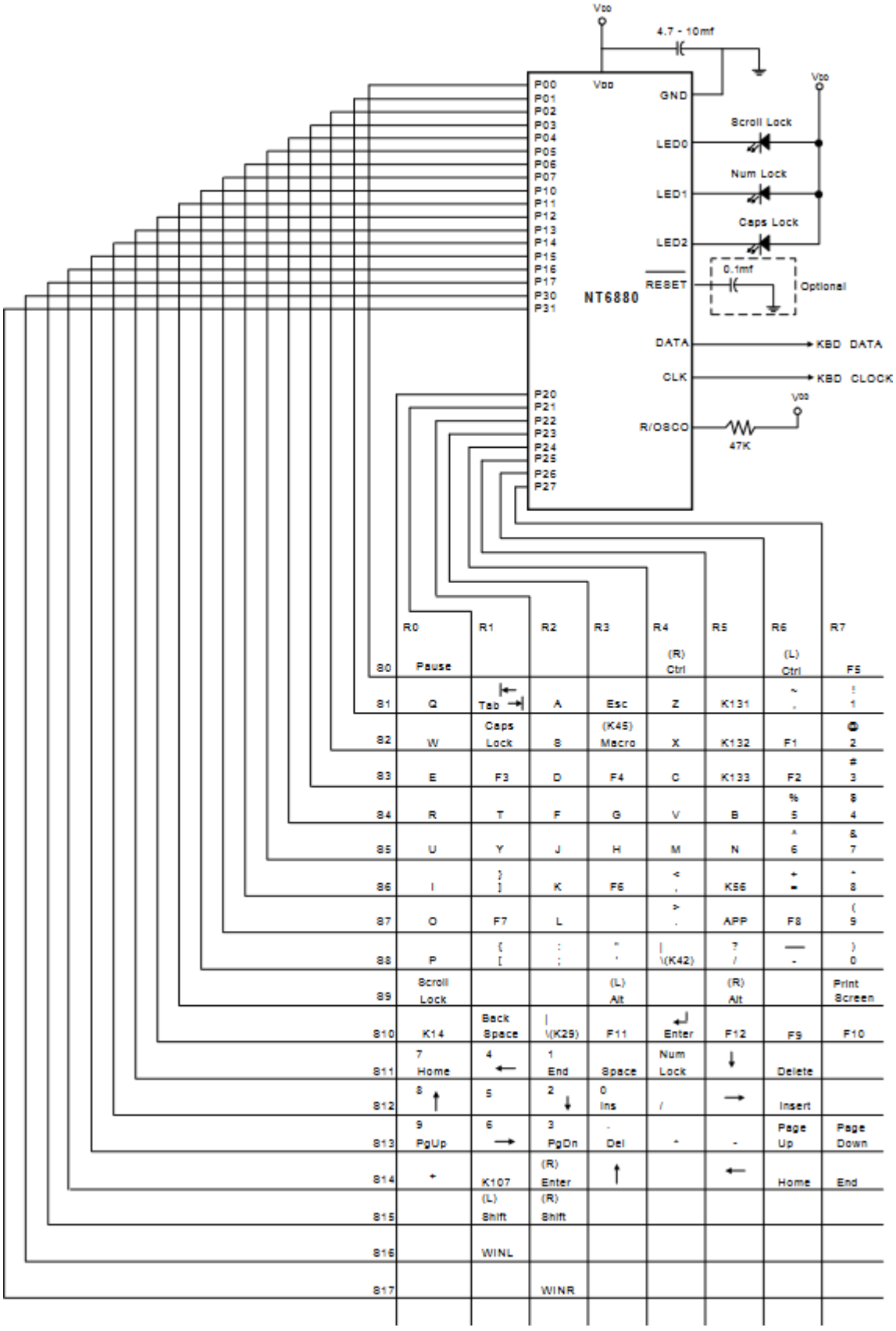


Рис. 4.35. Схема клавиатуры

Порты ввода-вывода реализуют внутренний интерфейс с матрицей клавиш и индикаторами, а также внешний интерфейс с контроллером клавиатуры (сигналы CLK и DATA, которые соответствуют сигналам KCLK и KDAT на рис. 4.33).

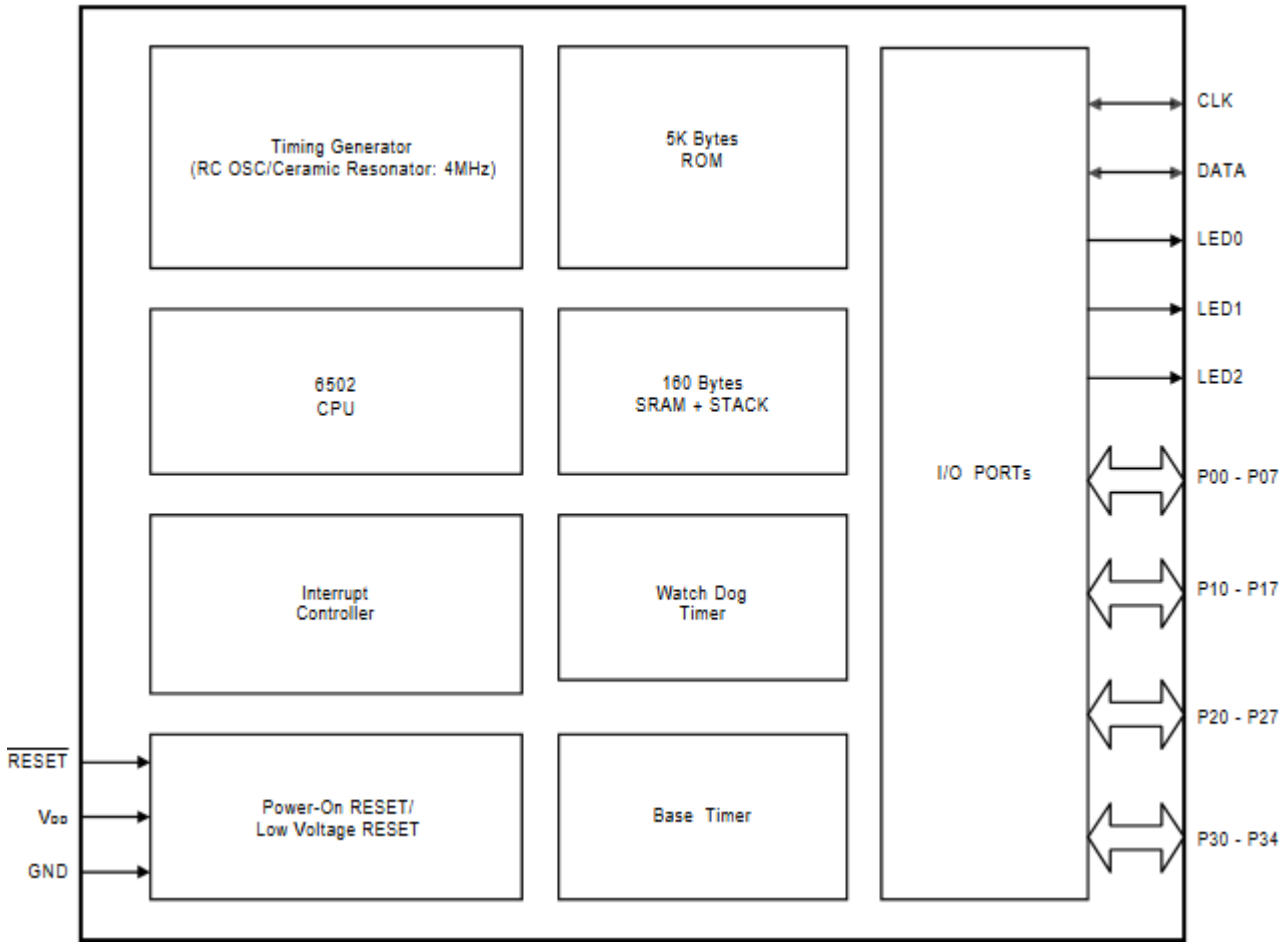


Рис. 4.36. Структура микроконтроллера клавиатуры

Микроконтроллер клавиатуры через соответствующие порты ввода-вывода сканирует матрицу клавиш, фиксирует нажатия и отпускания клавиш в форме так называемого скан-кода и отправляет его контроллеру клавиатуры. Для большинства клавиш размер скан-кода — один байт. При поступлении скан-кода контроллер клавиатуры помещает его в порт 60h и генерирует сигнал аппаратного прерывания, поступающий на вход IRQ 1 контроллера прерываний, который запрограммирован BIOS таким образом, что прерыванию на входе IRQ 1 соответствует номер прерывания INT 9. Контроллер

прерываний подает сигнал прерывания на вход INTR процессора и затем выдает ему номер прерывания 9 по сигналу INTA. Процессор по номеру прерывания определяет адрес обработчика и начинает его выполнять. Обработчик прерывания клавиатуры является частью драйвера клавиатуры, который выполняет следующие функции:

− чтение скан-кода из порта 60h;

− преобразование скан-кода в расширенный код ASCII;

− размещение расширенного кода ASCII в буфере клавиатуры, находящемся в оперативной памяти (начальный адрес — 041Eh);

− слежение за состоянием клавиш сдвига и клавиш переключателей (скан-коды этих клавиш не преобразуются в расширенный код ASCII, а просто их состояние, т. е. нажатие или отпускание, фиксируется в определенной области памяти);

− детектирование специальных комбинаций клавиш, например Ctrl-Alt-Del, и др.

Порт 60h работает не только на чтение, но и на запись. В него можно записывать команды и данные для их исполнения. В этом случае он выполняет функцию управления клавиатурой. С его помощью можно установить величину задержки и скорости повтора удерживаемой клавиши (Typematic Rate Delay), программно включить или выключить режимы Caps Lock, Scroll Lock и Num Lock. Также для работы с клавиатурой используется порт 64h, чтение которого позволяет определить состояние клавиатуры. Ранее использовался также порт 61h для разрешения работы клавиатуры.

В качестве примера рассмотрим программу ввода скан-кодов клавиш:

```
;компоновка с параметром 3, так как используется 32-битный регистр:
;tlink /3 scancode.obj
        .386                    ;директива для использования 32-битных расширений
data    segment
        dw ?
        SC db 0FFFDh dup(?)     ;массив для хранения скан-кодов
data    ends
```

```
code  segment use16
      assume cs:code, ds:data
main: mov dx, data
      mov ds, dx
      xor ecx, ecx        ;ecx будет считать каждый ввод через порт 60h
      xor si, si          ;регистр si будет хранить индекс массива скан-кодов
inloop: in al, 60h        ;ввод скан-кода через порт 60h
      inc ecx             ;подсчитан очередной ввод
      cmp al, bl          ;сравнение введенного скан-кода с предыдущим
      je inloop           ;если они равны, снова ввести скан-код
      mov bl, al          ;если не равны, сохранить введенный скан-код
      cmp al, 01h         ;1 — скан-код нажатия клавиши Escape
      jne NotEsc          ;если не Escape, продолжить ввод
      jmp quit            ;если Escape, завершить программу
NotEsc: mov SC[si], al    ;записать введенный скан-код в массив
      inc si              ;увеличить индекс массива
      jmp inloop          ;начать следующий виток цикла ввода
quit: mov ax, 4C00h       ; завершить программу
      int 21h
code  ends
      end main
```

Если имя регистра заключено в квадратные скобки, то это означает, что в регистре находится адрес (или его часть), т. е. по команде mov SC[si], al значение al записывается в память по адресу SC+SI в сегменте данных data.

В данном примере ввод осуществляется по опросу, т.е. без использования прерывания IRQ 1/ INT 9. Для иллюстрации того, сколько напрасных действий выполняет процессор при опросе, в программе осуществляется подсчет каждого ввода через порт 60h. В качестве счетчика выступает 32-битный регистр ECX. 16-битного регистра будет мало. После открытия программы в отладчике была поставлена точка останова (F2) на команде mov ax, 4C00h, затем программа была запущена (F9) и были последовательно нажаты цифровые клавиши основного блока клавиатуры от 1 до 0, а затем нажата клавиша

Escape. В результате, в массив были записаны скан-коды нажатия и отпускания клавиш (рис. 4.37).

В начале массива стоят скан-коды нажатия и отпускания клавиши F9 (43 и C3), которой в отладчике запускается выполнение программы до точки останова. Затем идут скан-коды цифровых клавиш: 1 — 02, 82; 2 — 03, 83; ...; 0 — 0B, 8B. Видно, что скан-коды отпускания равны сумме скан-кода нажатия и числа 80h, т. е. они различаются старшим битом.

При выполнении программы всего было нажато и отпущено 12 клавиш. При этом счетчик каждого ввода ECX равен 03A665F9h = 61 236 729$_{10}$ (!).

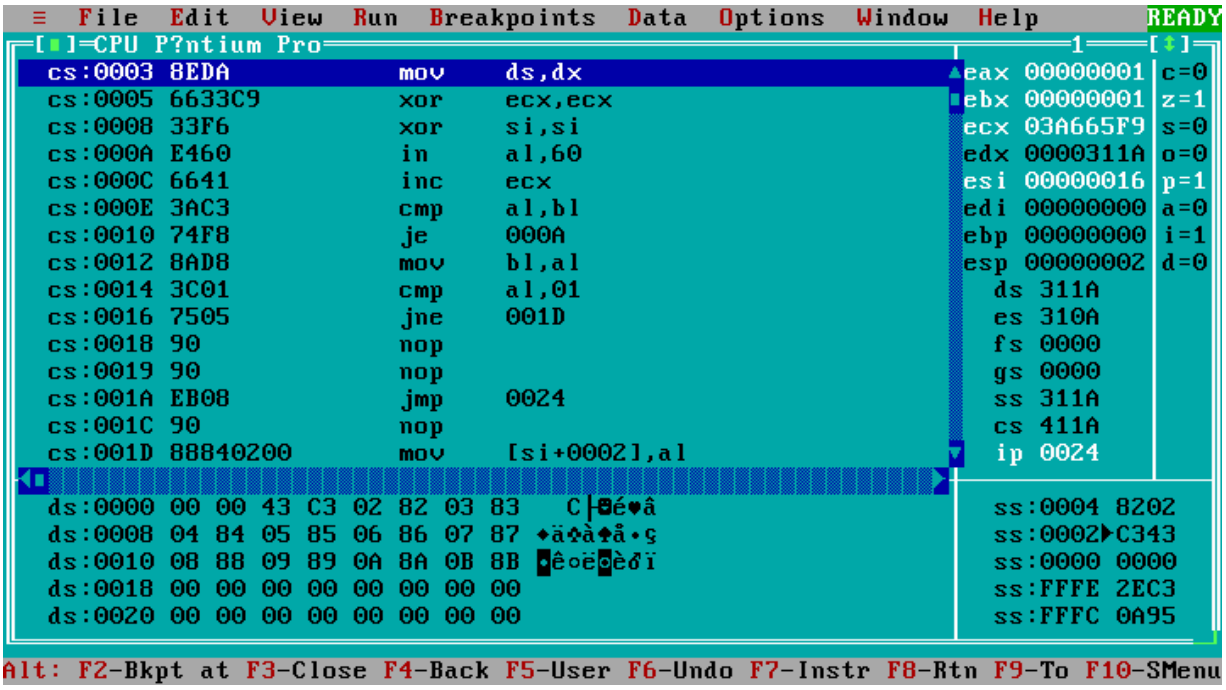


Рис. 4.37. Скриншот отладчика со скан-кодами цифровых клавиш основного блока клавиатуры

Видеосистема

Видеосистема (графическая система) включает видеоадаптер и монитор. Видеоадаптер (видеокарта, графическая карта, graphics card) с конструктивной точки зрения может быть дискретным, т. е. в виде

отдельной платы, или интегрированным, когда его основные элементы входят в состав материнской платы в виде отдельных микросхем, в состав чипсета в виде части микросхемы северного моста или в состав центрального процессора в виде графического процессора (GPU). Почти все современные процессоры x86 имеют в настоящее время в своем составе интегрированный GPU. Дискретная видеокарта в настоящее время подключается по интерфейсу PCIe. Ранее для ее подключения использовались интерфейсы AGP, PCI, ISA. Монитор, в свою очередь, может подключаться к видеоадаптеру с помощью других интерфейсов и разъемов. Большинство современных видеоадаптеров содержат только цифровые разъемы: DisplayPort (DP), HDMI (High-Definition Multimedia Interface), DVI (Digital Visual Interface). Но выпускаются еще видеоадаптеры и с аналоговым разъемом VGA (D-sub).

В зависимости от производителя и модели структура видеоадаптера может различаться, но общем случае в его состав входят следующие основные части:

1. Графический процессор (GPU).
2. Контроллеры (графический контроллер или контроллер дисплея, контроллер видеопамяти и др.).
3. Оперативная память (видеопамять, Video RAM, VRAM).
4. Постоянная память (видео-ПЗУ, Video ROM).

Видеоадаптер, как и другие устройства, использует множество ресурсов компьютера (рис. 4.38). Видеопамять занимает определенные диапазоны в адресном пространстве, регистры, входящие в состав контроллеров, — диапазоны номеров портов ввода-вывода.

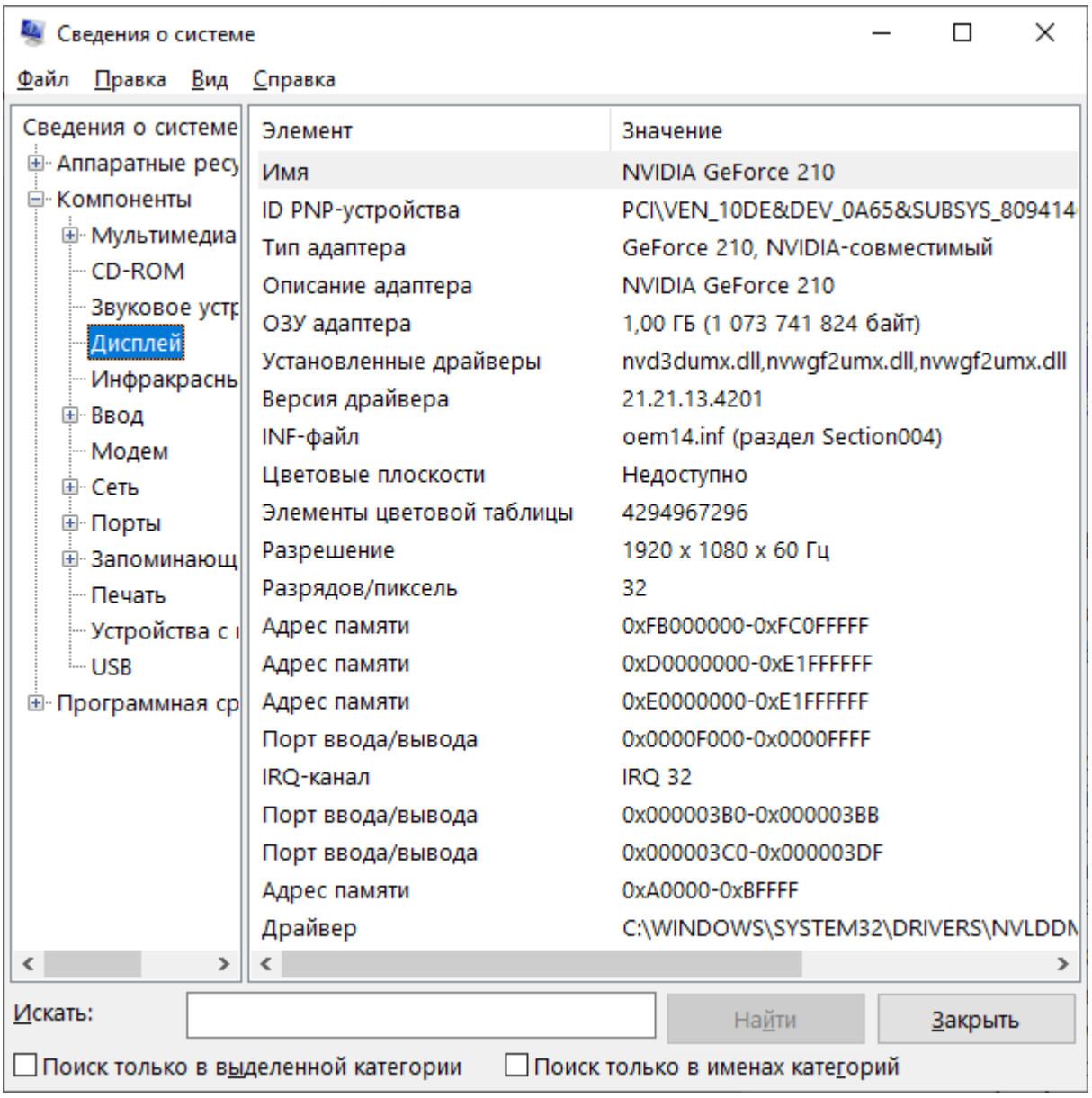


Рис. 4.38. Свойства видеокарты в программе «Сведения о системе» (msinfo32)

Изображение на экране монитора (дисплее) формируется на основе данных в видеопамяти. Порядок формирования изображения зависит от режима работы видеоадаптера (видеорежима), который настраивается с помощью определенных регистров. Видеопамять отображается на основное адресное пространство процессора. Диапазон видеопамяти в адресном пространстве зависит от видеорежима.

Видео-ПЗУ содержит BIOS видеоадаптера, экранные шрифты, служебные таблицы и другие данные. С его помощью осуществляется

инициализация и обеспечивается работа видеоадаптера до загрузки операционной системы и драйверов.

Существует множество стандартных видеорежимов, которые устанавливаются посредством функции 0 прерывания BIOS видеоадаптера INT 10h. Перед вызовом прерывания номер функции записывается в регистр процессора AH (AH = 0), а номер режима — в регистр AL. Можно установить и нестандартный режим непосредственным программированием регистров видеоадаптера.

Графические контроллеры различных производителей, как и видеоадаптеры в целом, имеют свои особенности, но в любом случае они содержат множество обрабатывающих, запоминающих и интерфейсных элементов, которые, в конечном итоге, обеспечивают формирование изображения на экране монитора. Например, в состав графического контроллера Winbond W9970CF входят следующие основные блоки: графический движок (Graphics Engine), видеоускоритель (Video Accelerator), контроллер памяти (DRAM Controller) и цифро-аналоговый преобразователь (RAMDAC, Random Access Memory Digital to Analogue Converter) (рис. 4.39) [71].

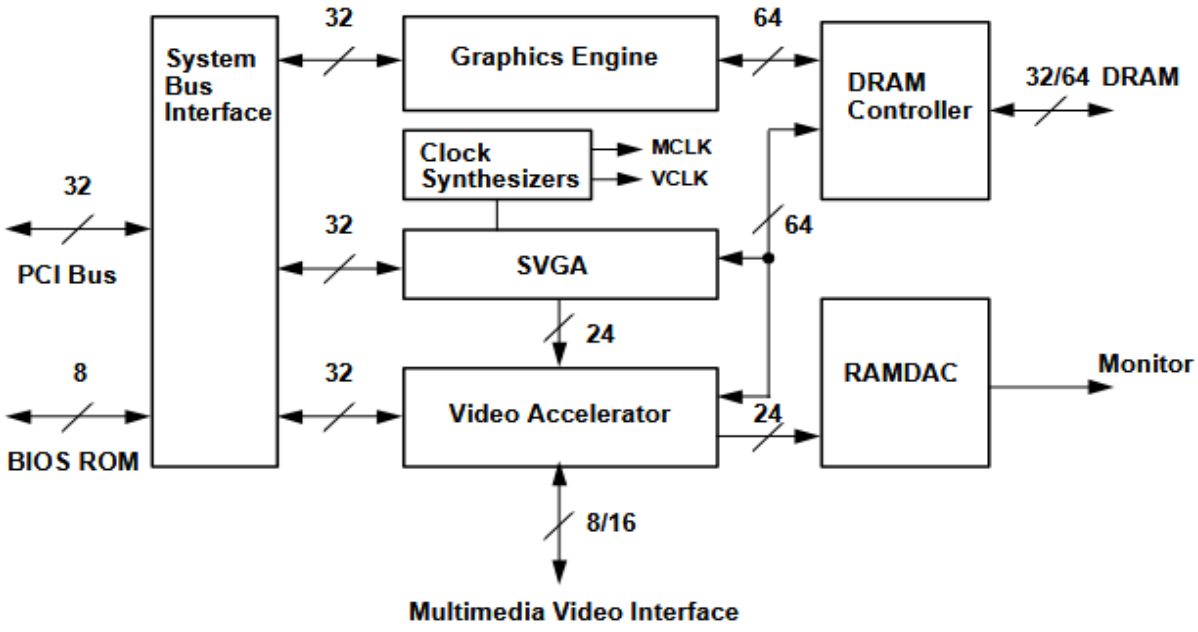


Рис. 4.39. Структура графического контроллера Winbond W9970CF

Графический движок (Graphics Engine) повышает производительность приложений с графическим интерфейсом пользователя за счет ускорения функций отрисовки изображений и аппаратного управления курсором.

RAMDAC включает статическую оперативную память и цифроаналоговый преобразователь (ЦАП, DAC). Он преобразует цифровой код цвета пикселов из видеопамяти в аналоговые сигналы для трех компонентов цвета R (Red), G (Green), B (Blue), поступающие на разъем VGA (D-sub) и далее на монитор.

Как уже было сказано, в настоящее время для подключения монитора используются в основном цифровые интерфейсы, но имеются в продаже и видеоадаптеры с аналоговым разъемом VGA и, соответственно, с цифроаналоговым преобразователем. Существует разновидность интерфейса DVI — DVI-I, в которой имеются те же аналоговые сигналы для трех компонентов цвета, что и в интерфейсе VGA.

Видеоускоритель предназначен для повышания качества воспроизведения видео. Он содержит множество буферов и обрабатывающих блоков (обрезка и фильтрация, масштабирование, интерполяция, преобразование цветового пространства, смешивание и наложение и т. д.) (рис. 4.40) [71].

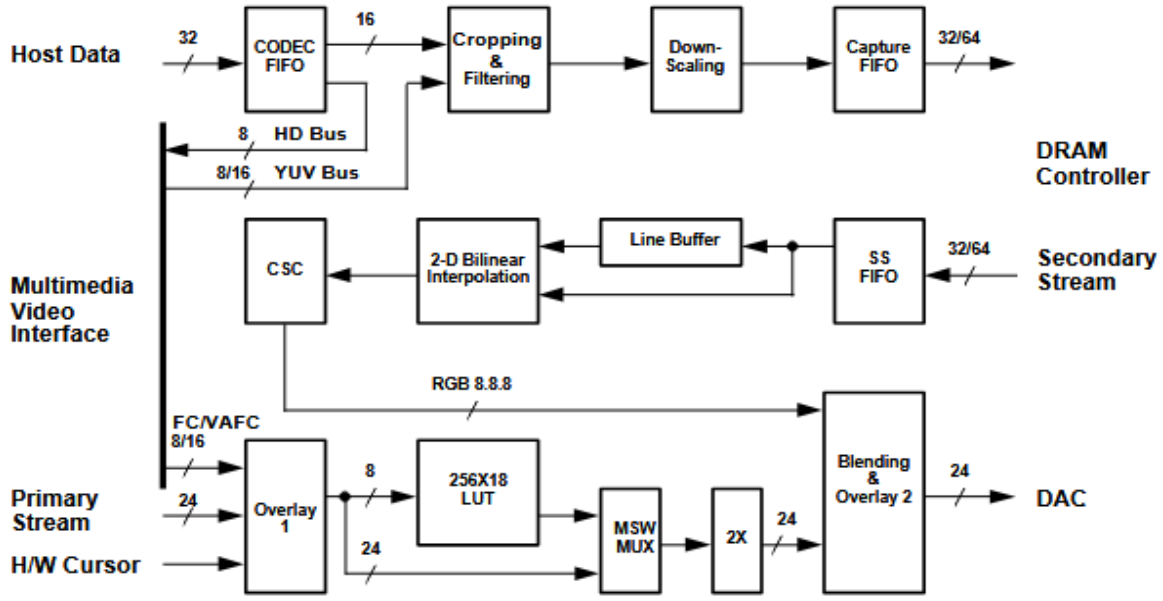


Рис. 4.40. Структура видеоускорителя графического контроллера W9970CF

Видеоускоритель также реализует интерфейс (Multimedia Video Interface) с аппаратными видеодекодерами MPEG (Moving Picture Experts Group) и телевизионными декодерами (рис. 4.41) [72]. (MPEG — стандарты сжатия видео- и аудиоинформации, разрабатываемые соответствующей группой международной организации по стандартизации (ISO, International Organization for Standardization), определяющие формат видеофайлов, свойства декодеров и т. п.)

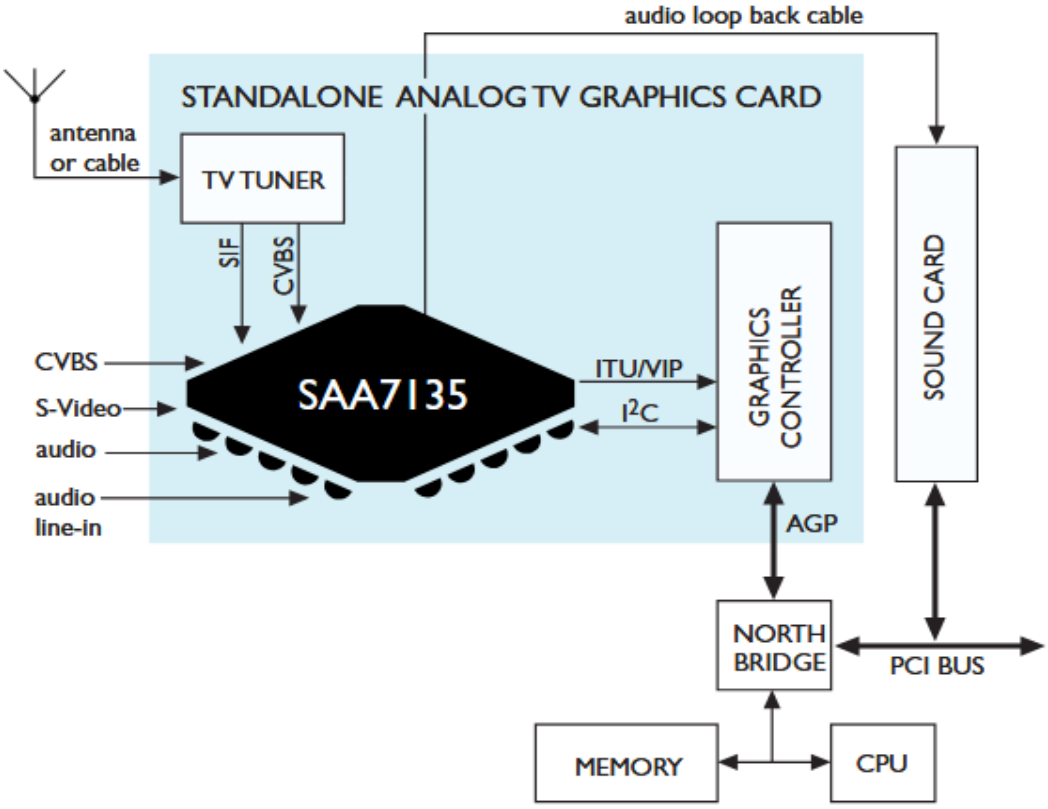


Рис. 4.41. Пример структуры компьютера с видеокартой, включающей видеодекодер (SAA7135) и телевизионный приемник (TV tuner)

Графический контроллер значительно ускоряет вывод информации на экран, разгружая центральный процессор выполнением своих операций. В еще большей степени это делает графический процессор (GPU). GPU, как и любой процессор, имеет собственную систему команд. Производительность многих современных GPU очень высока, что позволяет использовать их не только для формирования картинки на мониторе, но и выполнения общих научно-технических расчетов (GPGPU, General-Purpose computing on GPU), т. е. в качестве математического сопроцессора (ускорителя). Например, в 66-м рейтинге Top500 (ноябрь 2025 г.) 253 суперкомпьютера используют сопроцес-

соры (ускорители), 252 из них использует в качестве сопроцессоров графические процессоры, 219 — графические процессоры NVIDIA [20].

Современные графические процессоры — высокопараллельные вычислительные системы, имеющие сложную иерархическую структуру (рис. 4.42) [73].

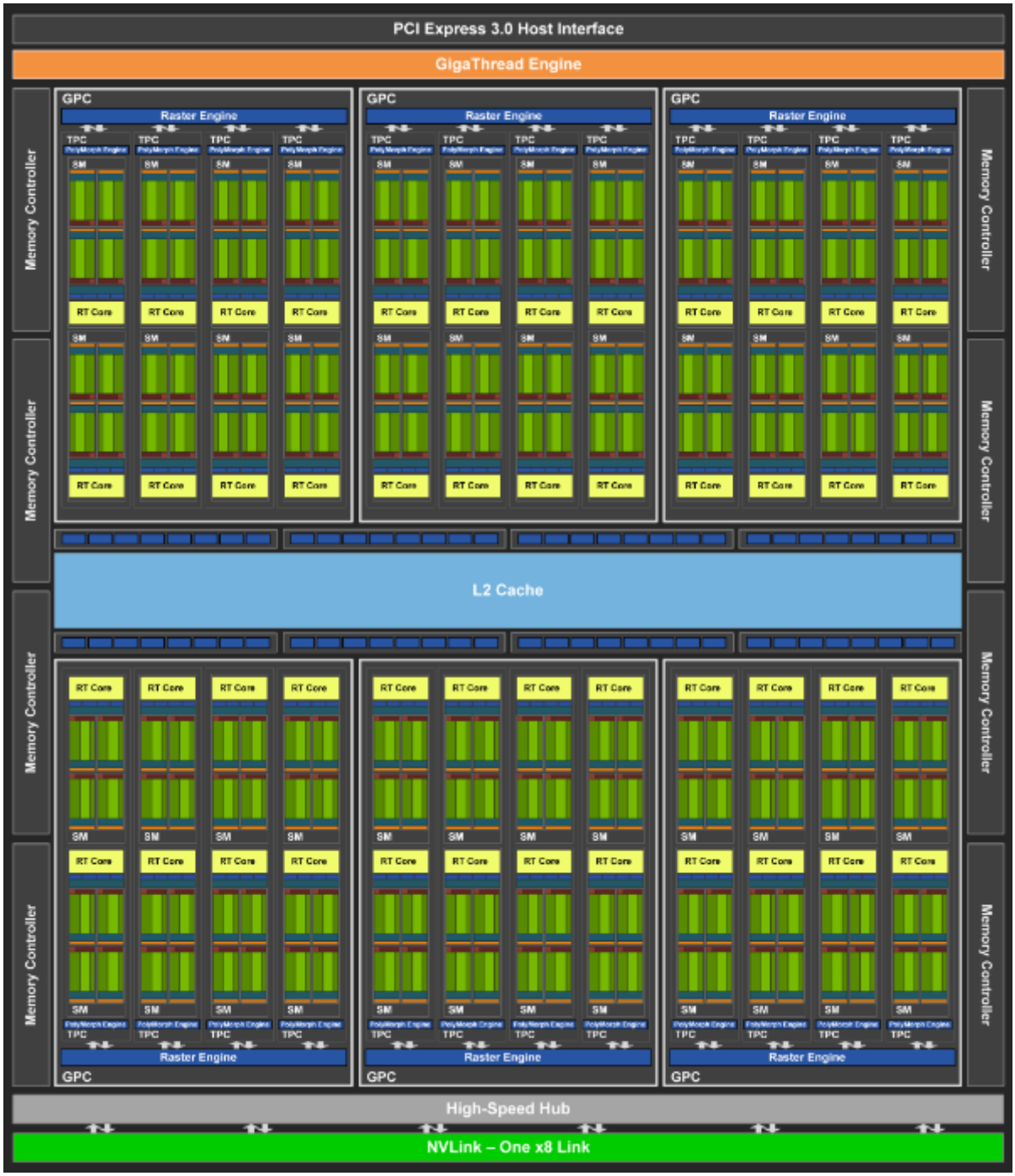


Рис. 4.42. Структура GPU на примере чипа NVIDIA TU104

GPU включает несколько графических кластеров (Graphics Processing Cluster, GPC), контроллеров памяти, кэш L2, интерфейс PCIe для взаимодействия с CPU и интерфейс NVLink для взаимодействия с другими GPU компьютерной системы.

GPC включает несколько текстурных кластеров (Texture Processing Cluster, TPC) и блок растеризации (Raster Engine).

TPC выполняют следующие основные функции:

− шейдинг вершин — преобразование 3D-координат вершин в 2D-координаты экрана;

− текстурирование — наложение текстур на 3D-модели.

Вершина (Vertex) в 3D-графике — это базовая единица, представляющая собой точку в трехмерном пространстве с заданными координатами. Вершины соединяются в ребра, которые, в свою очередь, формируют полигоны (многоугольники). Основа трехмерного компьютерного изображения (3D-модели) — сетка из полигонов. Чаще всего используется сетка, состоящая из треугольников. Затем на нее накладывается текстура — растровое изображение, которое придает поверхности модели нужный вид (цвет, рельеф и т. п.).

Блок растеризации преобразует 3D-модель в растровое (двумерное) изображение (пиксели).

Каждый TPC включает один или два потоковых мультипроцессора (Streaming Multiprocessor, SM). Каждый SM содержит множество ядер CUDA (INT32, FP32), тензорных ядер, регистровых файлов, текстурных модулей, кэш данных L1 и ядро RT (Ray Tracing, трассировка лучей) (рис. 4.43) [73]. (CUDA (Compute Unified Device Architecture) — программно-аппаратная архитектура параллельных вычислений, разработанная NVIDIA для своих графических процессоров.)

Число GPC, TPC, SM и других элементов в различных моделях GPU NVIDIA различается. В GPU других производителей аналогичные элементы имеют другое название.

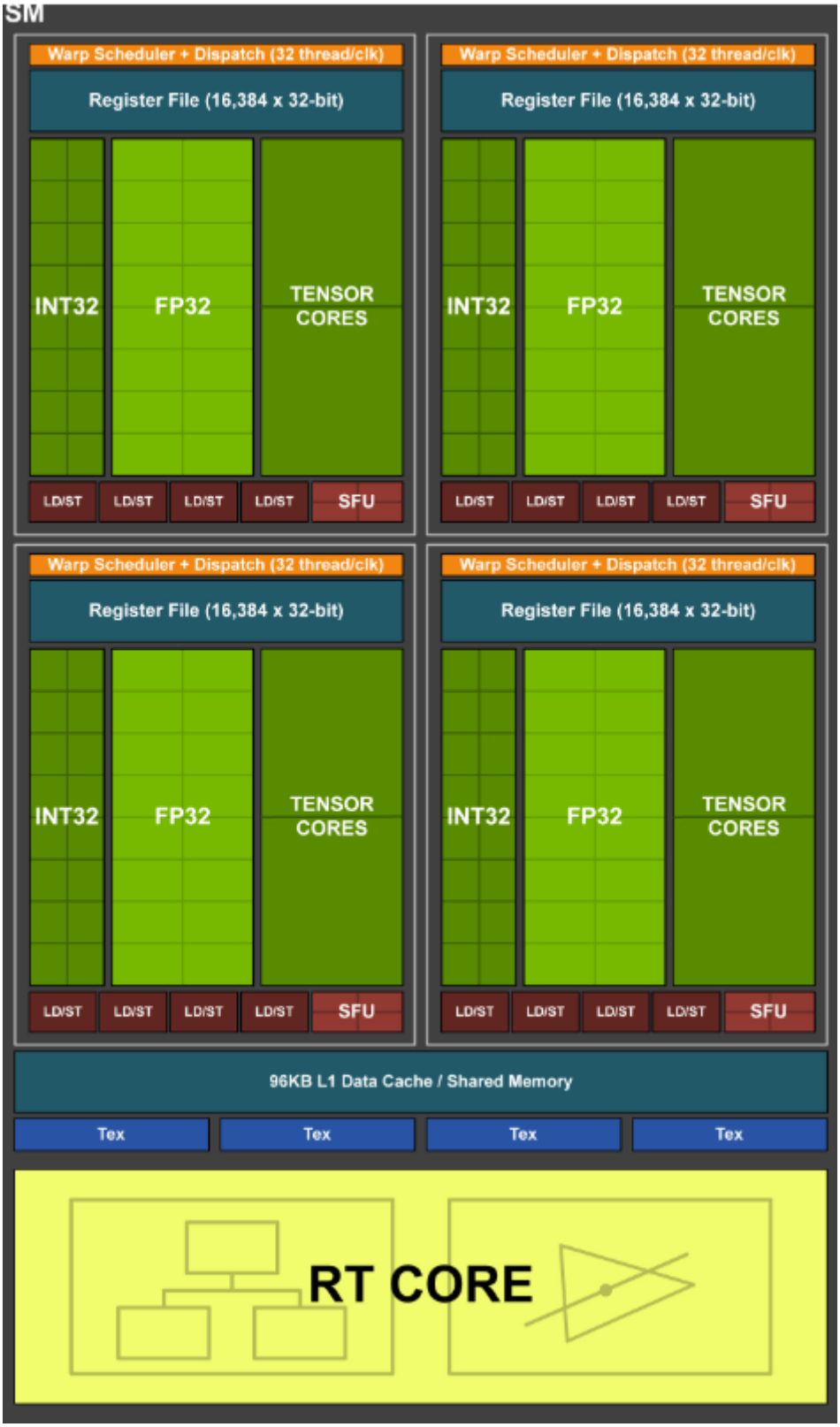


Рис. 4.43. Структура потокового мультипроцессора чипа NVIDIA TU104

Компьютерная система может включать множество GPU и CPU. Для взаимодействия GPU между собой используется высокоскорост-

ной интерфейс NVLink, который позволяет каждому GPU напрямую получать доступ к памяти других GPU, а также объединять память нескольких GPU для размещения больших наборов данных и более быстрых вычислений (рис. 4.44) [74].

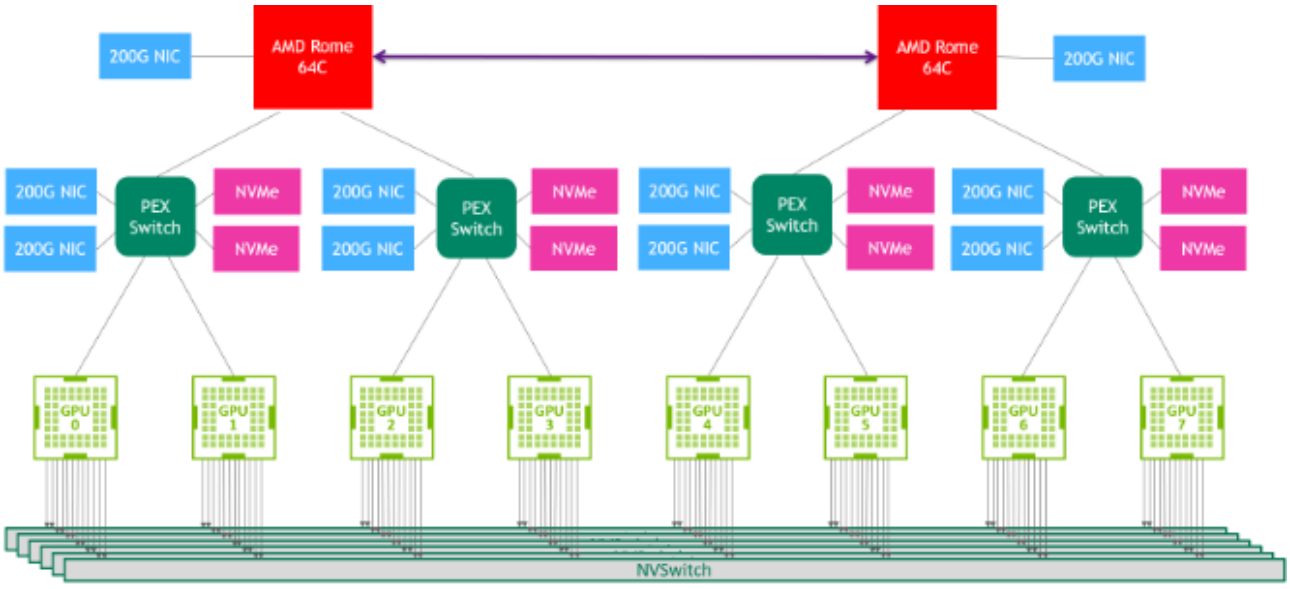


Рис. 4.44. Пример многопроцессорной компьютерной системы

GPU взаимодействуют с CPU (AMD Rome на рис. 4.44) с помощь интерфейса PCIe (PEX Switch на рис. 4.44).

Аудиосистема

Аудиосистема (звуковая система) включает звуковую карту (sound card), микрофон и колонки или наушники. По аналогии с видеокартой, звуковая карта может быть дискретной или интегрированной. Современные дискретные звуковые карты имеют интерфейс PCI Express или USB. Ранее они выпускались с интерфейсом PCI или ISA. Одной из первых широко распространенных звуковых карт стала плата Sound Blaster компании Creative Technology. Поэтому название Sound Blaster стало нарицательным и долгое время использовалось как синоним звуковой карты.

Звуковая карта включает следующие элементы:

– аудиоконтроллер (аудиопроцессор, звуковой процессор), в состав которого входит процессор цифровой обработки сигналов (DSP, Digital Signal Processor), цифроаналоговые преобразователи (ЦАП, DAC, Digital-to-Analog Converter) и аналого-цифровые преобразователи (АЦП, ADC, Analog-to-Digital Converter);

– аудиокодек, так же включающий ЦАП и АЦП, усилители для наушников и микрофона, аналоговые коммутаторы для переключения входов-выходов.

Также в состав звуковой карты могут входить и другие микросхемы, например, MIDI-синтезатор, используемый для генерирования музыки в играх по стандартному протоколу MIDI (Musical Instruments Digital Interface).

Современные звуковые карты Creative Technology, например, серии Sound Blaster Z, в качестве центрального элемента содержат высокопроизводительный аудиопроцессор Sound Core3D. Основные характеристики Sound Core3D [75]:

– 4 DSP-ядра;
– 6-канальный 24-битный ЦАП;
– 4-канальный 24-битный АЦП;
– усилитель для наушников;
– интерфейс для цифрового микрофона;
– интерфейс S/PDIF;
– интерфейс GPIO.

Об интерфейсе GPIO было сказано выше.

S/PDIF (Sony/Philips Digital Interface) — цифровой интерфейс для подключения различных компонентов аудиосистемы, разработанный компаниями Sony и Philips в середине 1980-х годов. Может использовать различные типы кабелей: электрический коаксиальный или волоконно-оптический.

Выделение в составе звуковой системы в качестве двух основных элементов аудиоконтроллера (цифровой части) и аудиокодека (аналоговой части) было введено в спецификации Audio Codec '97

(AC '97), которая, как уже говорилось ранее, появилась в 1997 году и была разработана Intel в сотрудничестве с основными производителями звуковых кодеков (рис. 4.45) [76].

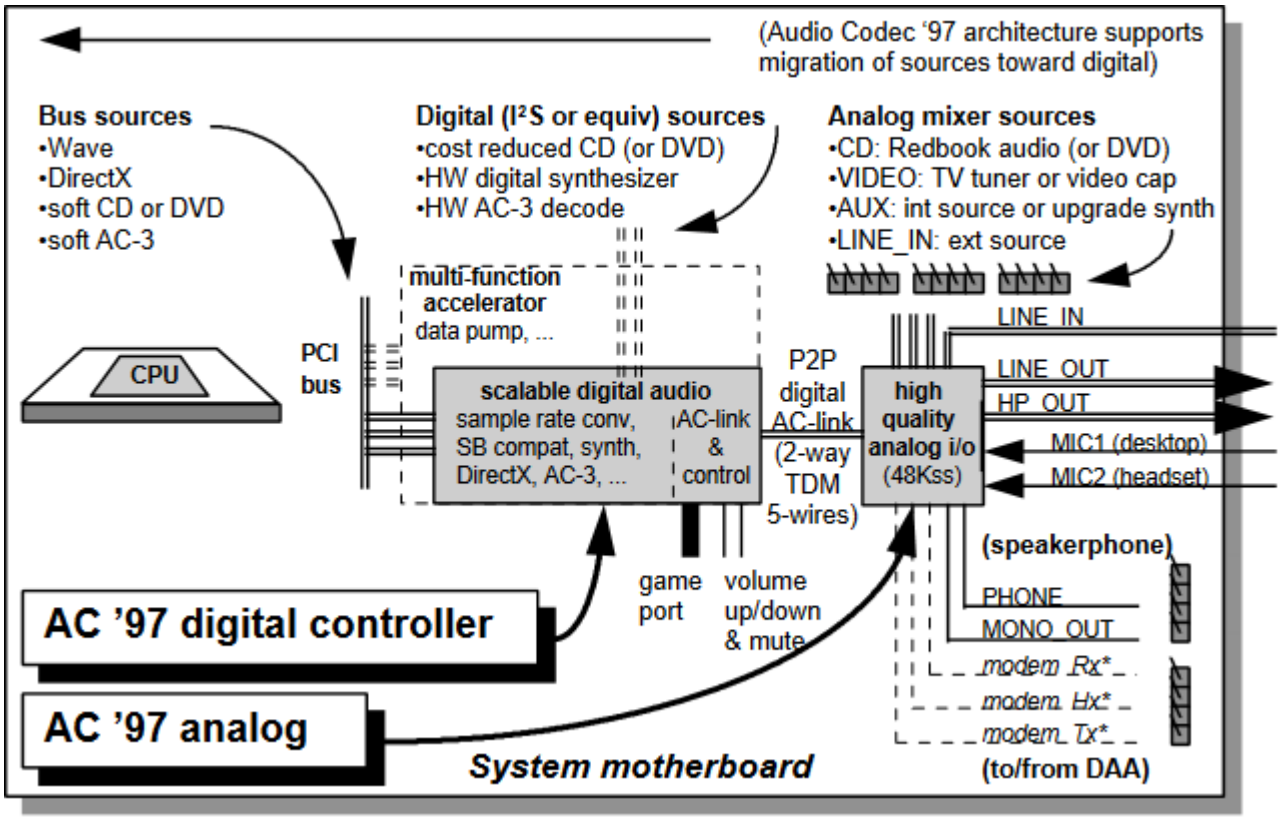


Рис. 4.45. Структура аудиосистемы в соответствии со спецификацией AC '97

Аудиоконтроллер (хост-контроллер, цифровой контроллер DC '97 (Digital Controller)) интегрирован в микросхему южного моста, а аудиокодек — в материнскую плату в виде отдельной микросхемы. Связь между ними осуществляется посредством интерфейса AC-link, который является 5-проводным двунаправленным последовательным двухточечным одноранговым, т.е. децентрализованным или пиринговым (P2P, Peer-to-Peer, равный к равному) интерфейсом с фиксированной скоростью передачи.

По интерфейсу AC-link одновременно передается множество входных и выходных аудиопотоков в цифровом формате PCM (Pulse Code Modulation, импульсно-кодовая модуляция) с использованием схемы мультиплексирования с временным разделением (TDM, Time

Division Multiplexed). Тактовая частота AC-link составляет 12,288 МГц, соответственно, скорость передачи — 12,288 МТ/с. Объединенный цифровой поток состоит из 256-битных кадров. Частота кадров составляет 12,288 МГц / 256 = 48 КГц. Каждый кадр включает 13 слотов. Слот 0 (Tag) имеет длину 16 бит и содержит флаги допустимости для остальных слотов. Остальные 12 слотов являются 20-битными слотами данных. Таким образом, по AC-link одновременно передаеется 12 входных и 12 выходных потоков данных.

Назначение пяти линий AC-link таково:

− SYNC — сигнал синхронизации кадра 48 КГц, выходной для контроллера и входной для кодека;

− BIT_ CLK — тактовый сигнал 12,288 МГц, входной для контроллера или вторичных кодеков и выходной для первичного кодека;

− SDATA_OUT — выходные данные для контроллера и входные для кодека;

− SDATA_IN — входные данные для контроллера и выходные для кодека;

− RESET# — сигнал сброса, выходной для контроллера и входной для кодека.

В состав аудиокодека входят следующие элементы (рис. 4.46) [76]:

− интерфейс с цифровым контроллером (digital interface), включающий 64 16-битных регистра;

− цифроаналоговые и аналого-цифровые преобразователи;

− миксер.

Аудиокодек обеспечивает различные входы-выходы для других элементов аудиосистемы:

− микрофона (MIC1, MIC2);

− наушников (HP-OUT, HeadPhone Output);

− привода компакт-дисков при воспроизведении аудиодисков и др.

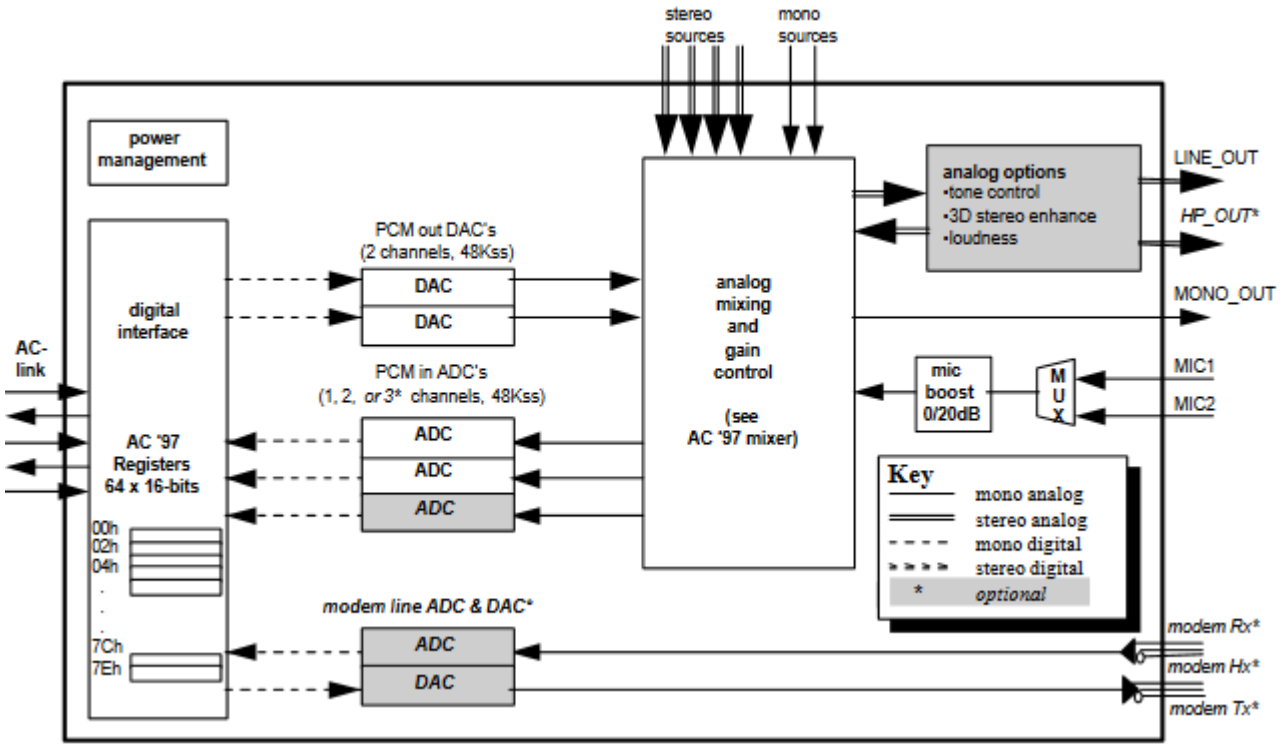


Рис. 4.46. Структура аудиокодека AC '97 2.1

В развитие спецификации AC '97 было выпущено несколько ее версий (revisions), последней из которых была версия 2.3, а в 2004 году появилась новая спецификация Intel High Definition Audio (HD Audio) (рис. 4.47) [77]. HD Audio обеспечивает обработку большего количества потоков с более высоким качеством звука.

Она по-прежнему определяет два типа основных аудиоэлементов: аудиоконтроллер и аудиокодек — но не имеет обратной совместимости с AC '97. Аудиоконтроллер (кодовое наименование Azalia) соединяется с одним или несколькими кодеками с помощью интерфейса High Definition Audio Link (HDA-link). Аудиоконтроллер содержит один или несколько DMA-модулей, каждый из которых может быть настроен для передачи одного аудиопотока в память из кодека или из памяти в кодек (рис. 4.48) [77].

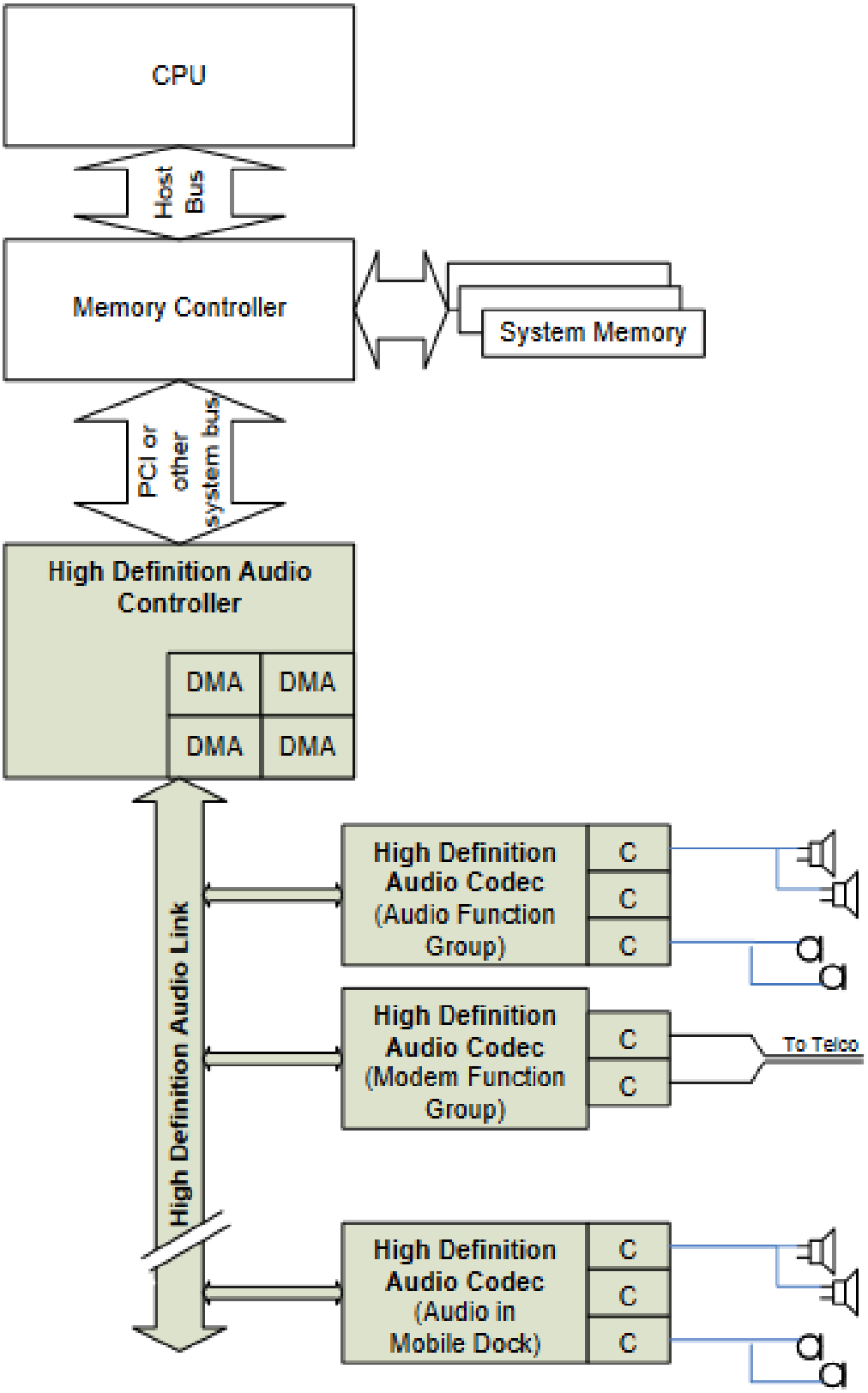


Рис. 4.47. Общая структура ПЭВМ со спецификацией HD Audio

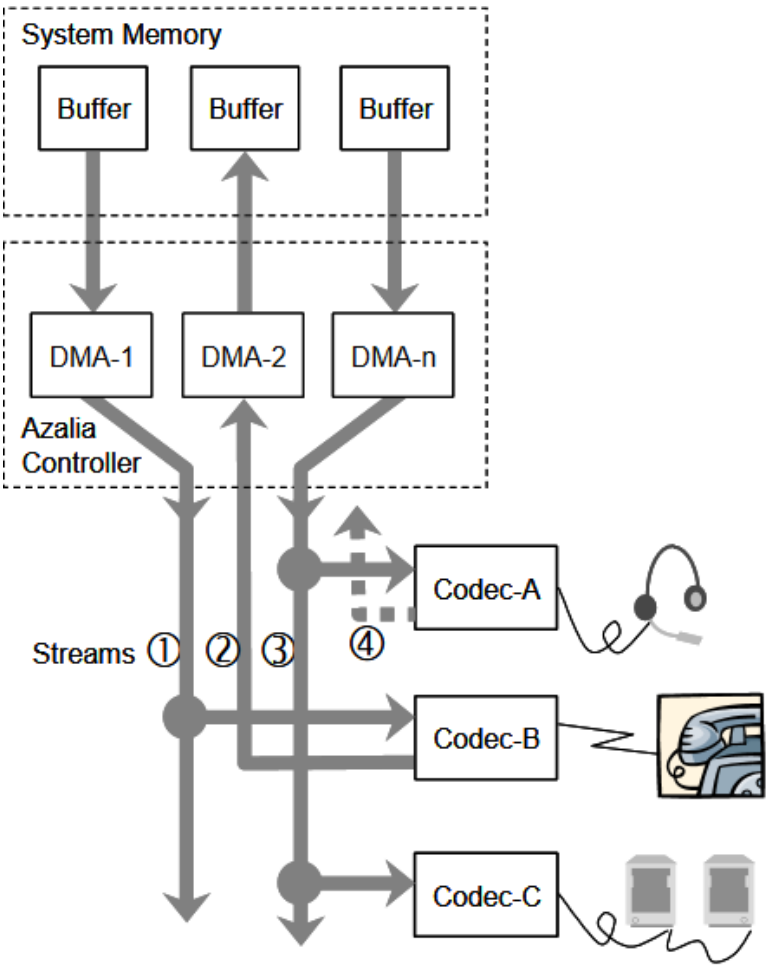


Рис. 4.48. Взаимодействие кодеков с памятью через DMA-модули контроллера

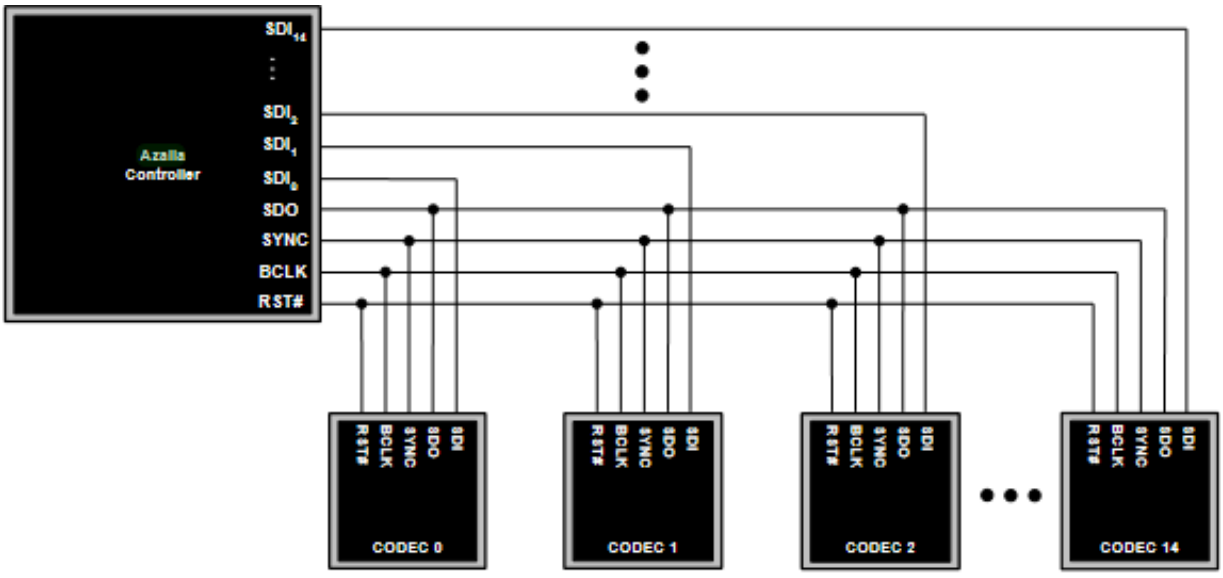


Рис. 4.49. Интерфейс между контроллером и кодеками HDA-link

В звуковой системе на основе спецификации HD Audio может быть использовано до 15 кодеков (рис. 4.49) [77]. После сброса в соответствии с протоколом инициализации каждому кодеку назначается уникальный идентификатор.

В интерфейсе HDA-link входные линии данных отдельные для каждого кодека. Сигнал тактовой частоты BCLK генерируется контроллером и подается на все кодеки.

Имеется множество микросхем аудиокодеков различных производителей, например:

– IDT STAC9752A/9753A компании Integrated Device Technology (рис. 4.50) [78];

– AD1888 компании Analog Devices (рис. 4.51) [79];

– VT1708 компании VIA Technologies (рис. 4.52) [80];

– ALC892 компании Realtek Semiconductor [81].

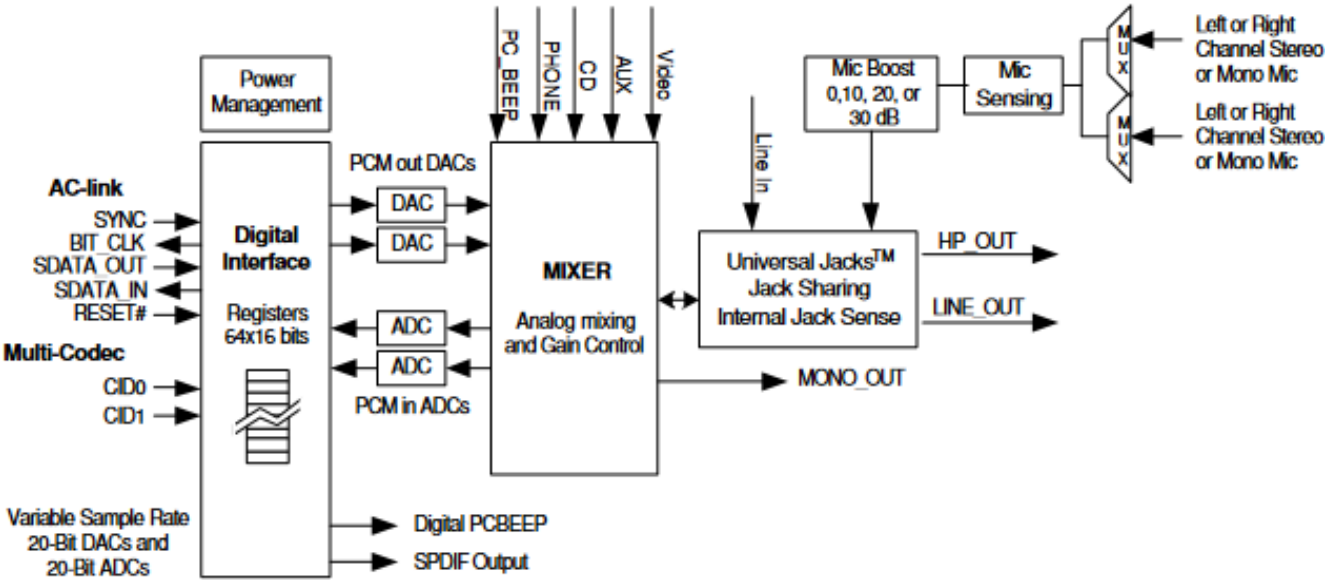


Рис. 4.50. Структура аудиокодека IDT STAC9752A/9753A

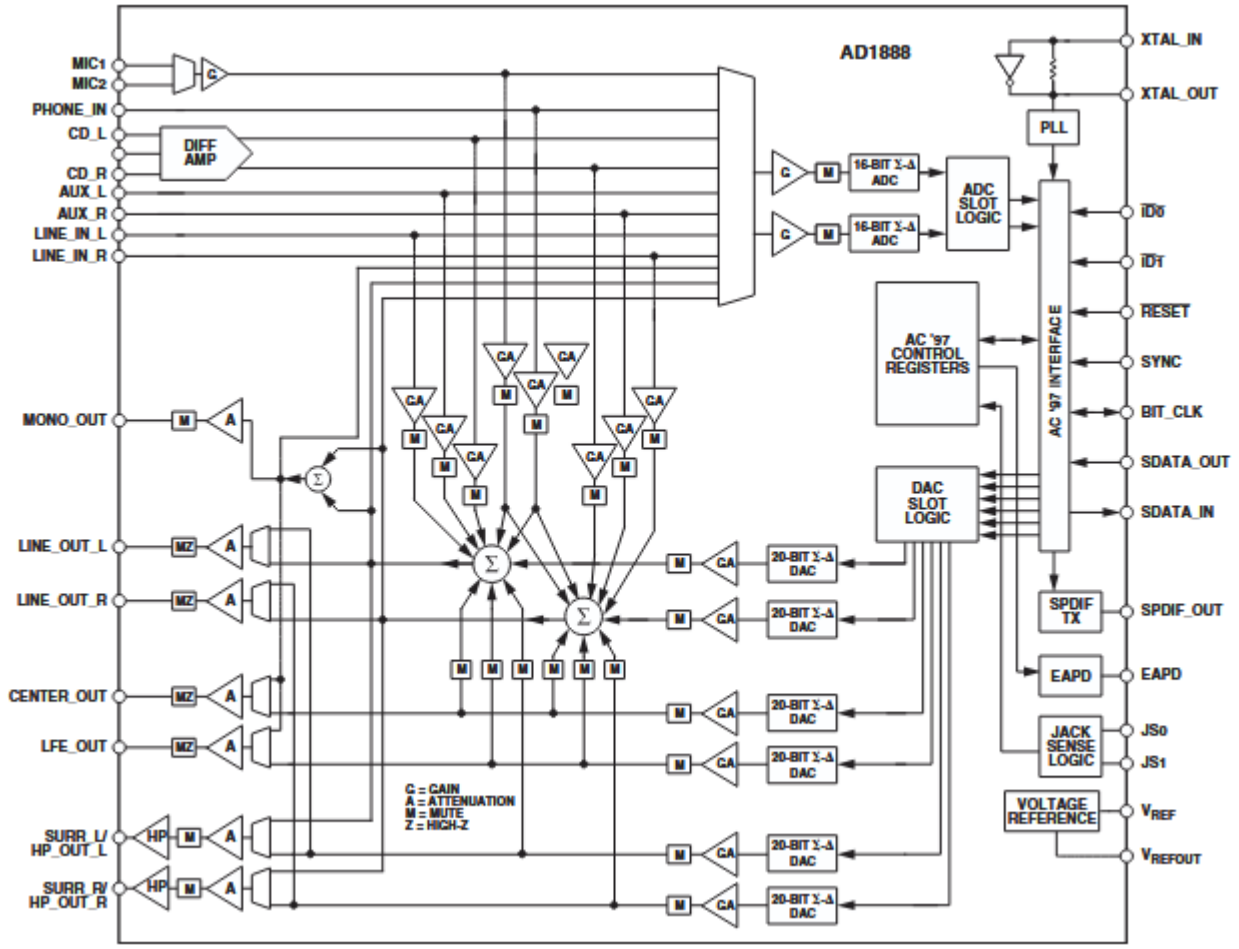


Рис. 4.51. Структура аудиокодека AD1888

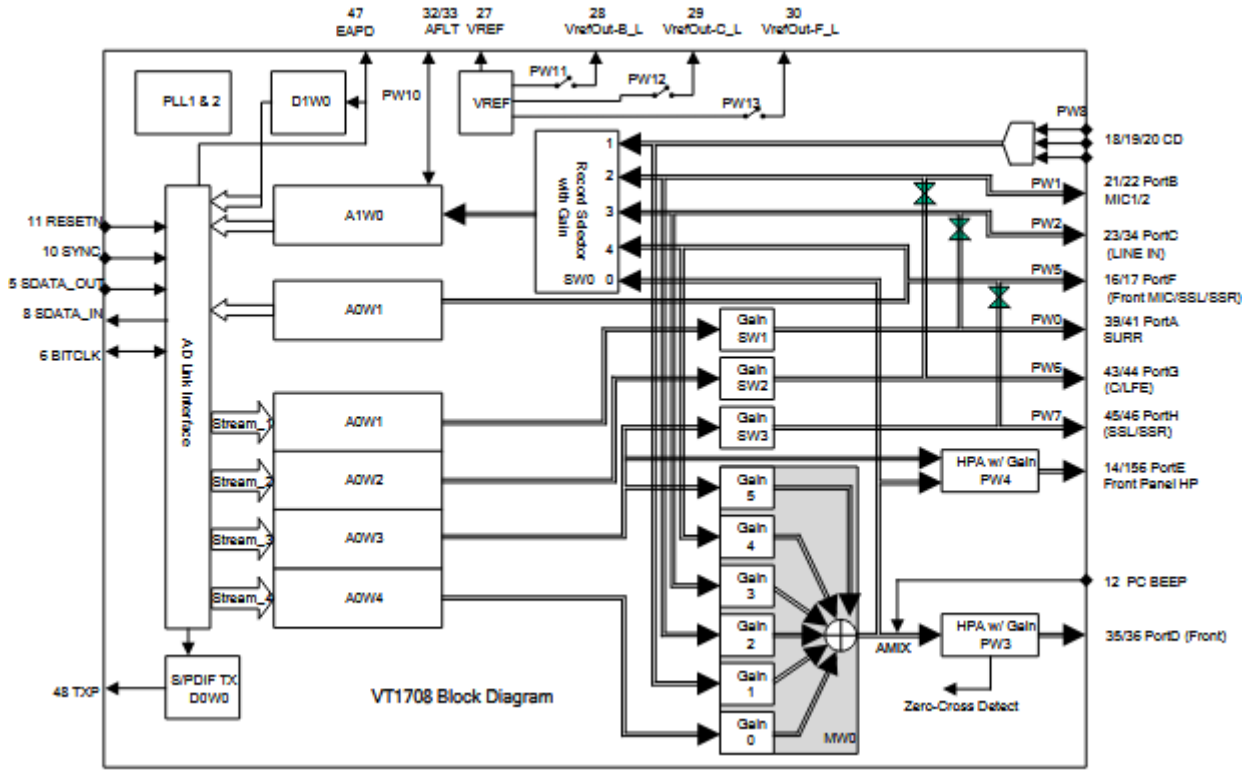


Рис. 4.52. Структура аудиокодека VT1708

Каждый производитель разрабатывает документацию на свои изделия с той или иной степенью детализации. Схема кодека IDT STAC9752A/9753A на рис. 4.50 почти полностью повторяет общую схему кодека по спецификации AC '97 (см. рис. 4.46). Схема кодека AD1888 имеет больше подробностей. Можно отметить, что внутренний генератор тактовых импульсов кодека AD1888 (в правом верхнем углу схемы) состоит всего лишь из двух элементов: инвертора и резистора, обеспечивающего обратную связь (см. п. 1.2). Генератор стабилизируется кварцем 24,576 МГц, подключаемым к выводам XTAL_IN и XTAL_OUT. Выходной сигнал генератора через блок фазовой автоподстройки частоты (PLL, Phase-Locked Loop) поступает на интерфейс AC-link.

Кодеки STAC9752A/9753A и AD1888 разработаны в соответствии с версией 2.3 спецификации AC '97, которая допускает использование в аудиосистеме нескольких кодеков. У них имеются входы идентификации (CID_0, CID_1 у STAC9752A/9753A, ID0#, ID1# у AD1888), которые используются для выбора одного из нескольких кодеков в текущий момент времени.

В портативных и мобильных устройствах: ультрабуках, планшетах, смартфонах — применяются другие типы кодеков с другими интерфейсами: I2C, MIPI SoundWire (Mobile Industry Processor Interface) и др. Это, например, микросхема CS42L42 компании Cirrus Logic [82].

4.4. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 4

Задание 1. Интерфейсы ПЭВМ

Информацию о характеристиках и показателях интерфейсов компьютера можно получить с помощью тех же средств операцион-

ной системы (диспетчера устройств и др.) и специализированных информационно-диагностических утилит, которые использовались в предыдущих разделах. При анализе конфигурации компьютера в разделе 2, определении характеристик процессора в разделе 3 была получена, в том числе, и основная информация об интерфейсах (см. рис. 2.23–2.31, 3.41). В данном задании необходимо собрать подробную информацию об интерфейсах персонального компьютера.

CPU-Z выдает информацию об основном интерфейсе и интерфейсе графического адаптера (рис. 4.53), а также о некоторых характеристиках интерфейса с памятью (рис. 4.54).

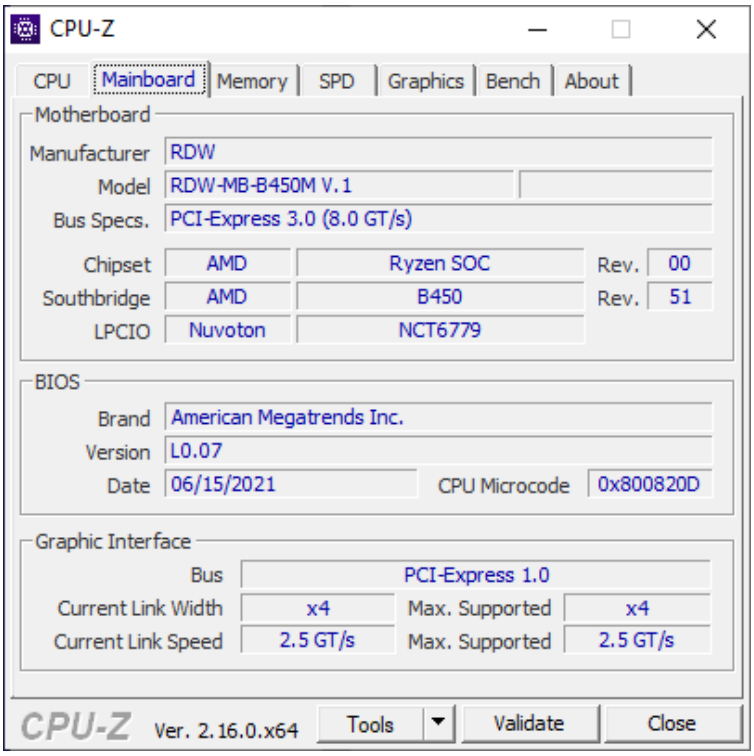


Рис. 4.53. Характеристики интерфейсов в программе CPU-Z

AIDA64 выдает информацию об интерфейсах, реализуемых северным и южным мостами чипсета (рис. 4.55, 4.56), а также об устройствах и контроллерах других интерфейсов, подключенных к шине PCIe (рис. 4.57).

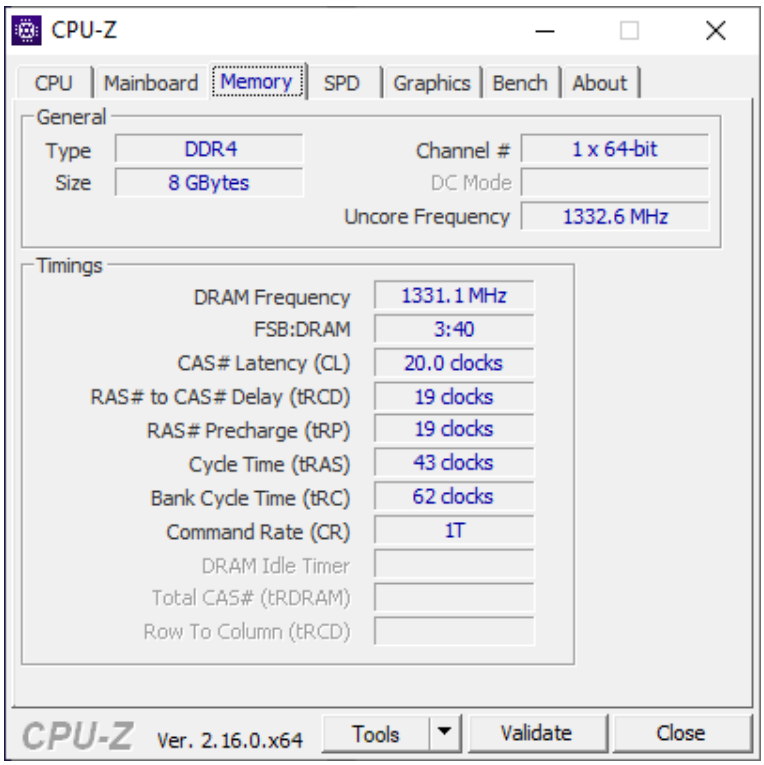


Рис. 4.54. Характеристики интерфейса с памятью в программе CPU-Z

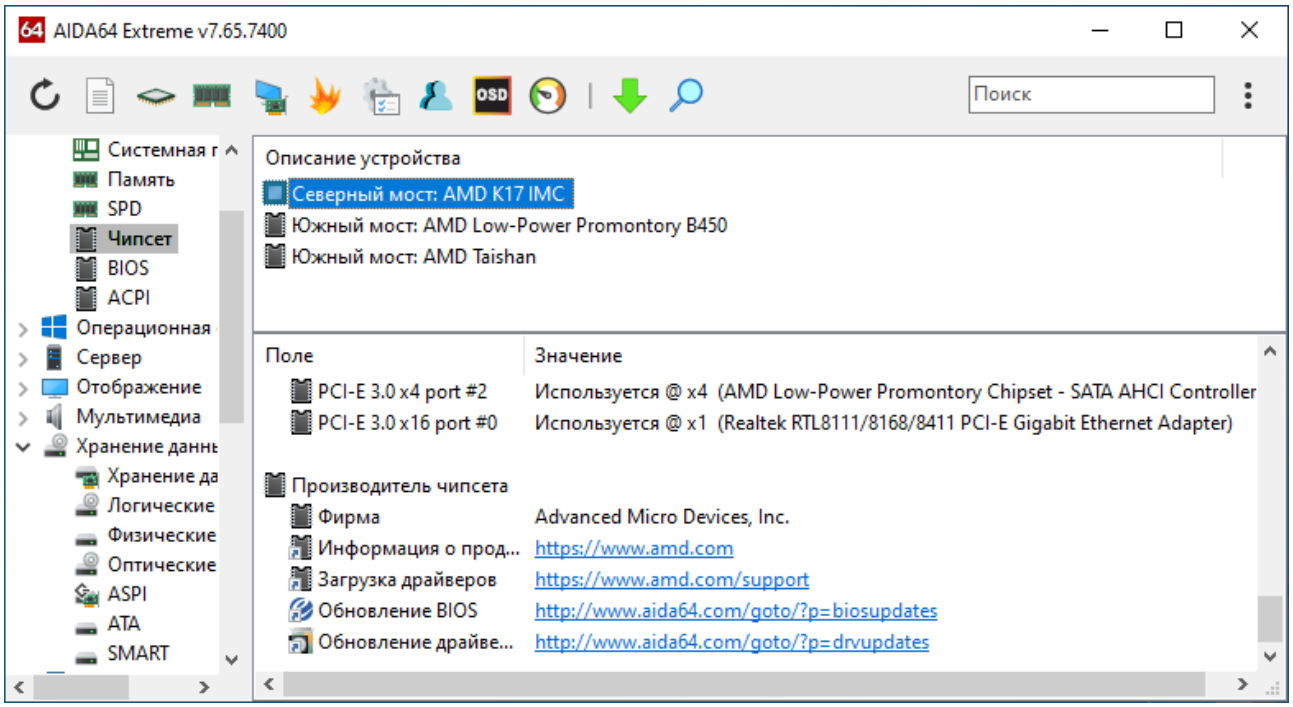


Рис. 4.55. Интерфейсы, реализуемые северным мостом, в программе AIDA64

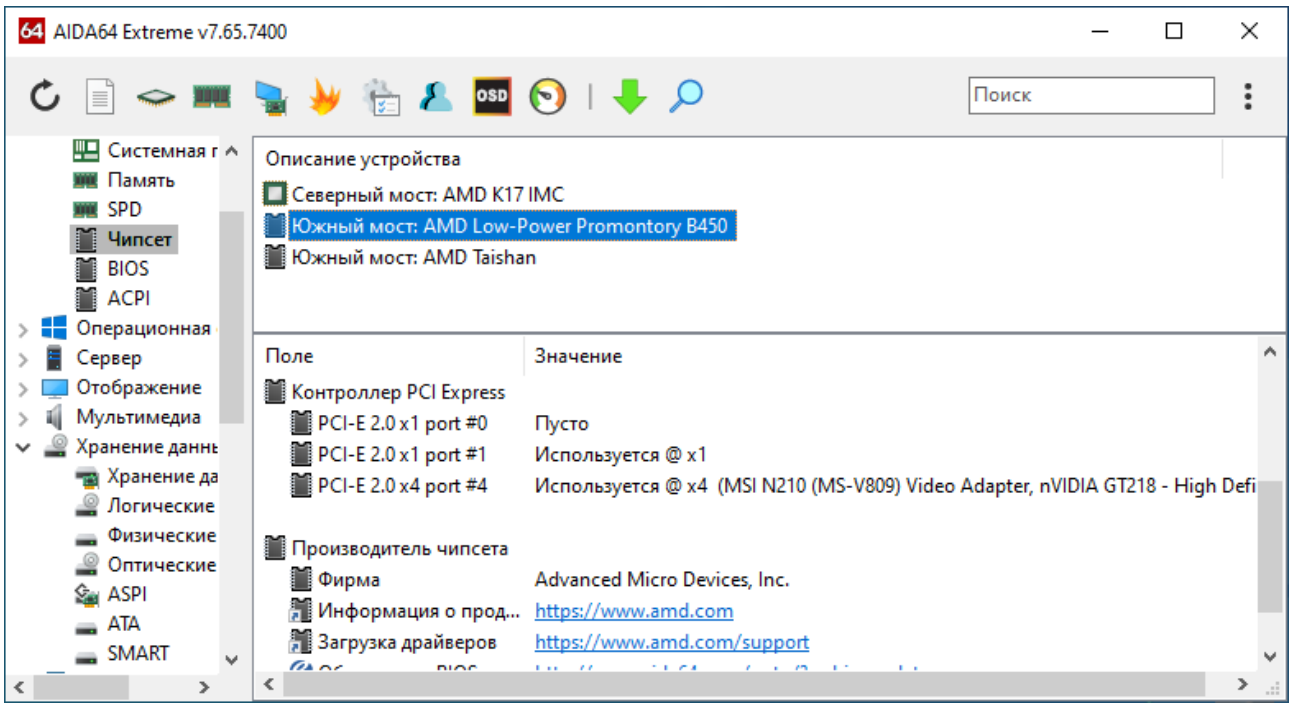


Рис. 4.56. Интерфейсы, реализуемые южным мостом, в программе AIDA64

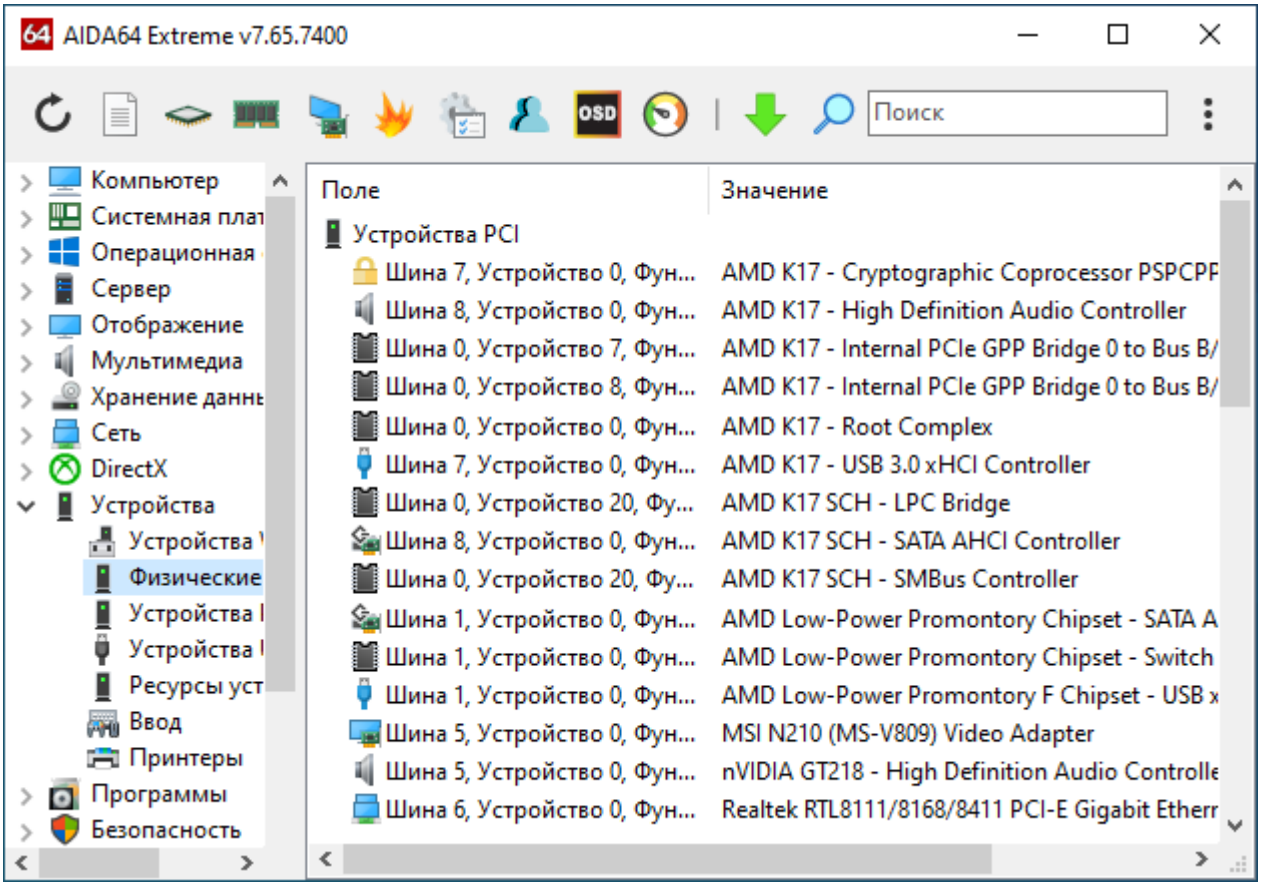


Рис. 4.57. Устройства, подключенные к шине PCIe, в программе AIDA64

В данном задании необходимо с использованием средств операционной системы и информационно-диагностических утилит получить подробную информацию об интерфейсах компьютера, их характеристиках и показателях производительности. Порядок выполнения задания проиллюстрировать скриншотами. Результаты оформить в виде списков или таблиц.

Задание 2. Ввод данных с клавиатуры

На языке ассемблера необходимо написать программу ввода скан-кодов клавиш клавиатуры.

Как было сказано выше, микроконтроллер клавиатуры, находящийся в самой клавиатуре, отправляет скан-коды нажатия и отпускания клавиш контроллеру клавиатуры, находящемуся в системном блоке. Одной клавише соответствует два скан-кода: нажатия и отпускания. При этом справедлива формула:

скан-код отпускания = скан-код нажатия + 80h

или

скан-код отпускания = скан-код нажатия OR 80h,

т.е. различие между скан-кодами отпускания и нажатия заключается в старшем бите, в скан-коде нажатия он равен нулю, в скан-коде отпускания – единице.

Пример программы:

```
data   segment
       dw ?
       SC db 0FFFDh dup(?)     ;массив для хранения скан-кодов
data   ends
code   segment
       assume cs:code, ds:data
main:  mov dx, data
       mov ds, dx
       xor si, si              ;регистр si будет хранить индекс массива скан-
кодов
```

```
;начало цикла ввода:
inloop: in al, 60h          ;ввод скан-кода через порт 60h
        cmp al, bl          ;сравнение введенного скан-кода с предыдущим
        je inloop           ;если они равны, снова ввести скан-код
        mov bl, al          ;если не равны, сохранить введенный скан-код
        cmp al, 01h         ;1 – скан-код нажатия клавиши Escape
        jne NotEsc          ;если не Escape, продолжить ввод
        jmp quit            ;если Escape, завершить программу
NotEsc: mov SC[si], al      ;записать введенный скан-код в массив
        inc si              ;увеличить индекс массива
        jmp inloop          ;начать следующий виток цикла ввода
quit:   mov ax, 4C00h       ; завершить программу
        int 21h
code    ends
        end main
```

Исследовать работу программы в отладчике с использованием контрольных точек (точек останова, breakpoints):

1. Открываем программу в отладчике.
2. Выделяем (стрелочкой вниз) команду, на которой нужно остановиться. В данном случае это первая команда блока завершения программы — mov ax, 4C00h.
3. Ставим на ней контрольную точку — F2 (Bkpt, Breakpoint).
4. Запускаем программу — F9 (Run). Появится экран с ожиданием ввода.
5. Набираем на клавиатуре свою фамилию, а затем клавишу Esc (в приведенном примере выход из программы происходит по нажатию клавиши Esc). Программа останавливается на команде с точкой останова.
6. Смотрим, что находится в массиве скан-кодов — переходим в поле сегмента данных (Shift + Tab), нажимаем Ctrl + G (Goto), переходим в начало сегмента данных (DS:0000). В нем можно увидеть скан-коды нажатия и отпускания клавиши F9, которую мы нажимали для запуска программы после установки контрольной точки, и скан-коды остальных нажатых клавиш.

Выписать нажатые клавиши и их скан-коды нажатия и отпускания. Оформить это в виде списка или таблицы. Привести скриншот отладчика со значениями введенных скан-кодов (см. рис. 4.37).

Задание 3. Вывод информации на дисплей

Операционная система DOS (MS-DOS или FreeDOS) использует видеорежим номер 3 — текстовый режим 16 цветов, 80×25. Экран представляет собой матрицу знакомест (символов) размером 25×80: 25 строк по 80 символов. Каждому знакоместу соответствует одно слово в видеопамяти следующего формата:

− младший байт (четный адрес) — ASCII-код символа;

− старший байт (нечетный адрес) — атрибут символа: младшая тетрада — цвет символа, старшая тетрада — цвет фона и мерцание символа:

7	6	5	4	3	2	1	0
яркость/ мерцание	цвет фона			цвет символа			
	R	G	B	яркость	R	G	B

В каждой тетраде три младших бита соответствуют трем базовым цветам: красному, зеленому и синему. Назначение бита 7 зависит от значения бита 3 регистра управления режимом контроллера атрибутов видеоадаптера. Если он равен 0, то бит 7 байта атрибутов определяет яркость цвета фона, если равен 1 — мерцание символа.

Адрес начала видеопамяти в этом режиме — B8000h. Он соответствует верхнему левому углу экрана. Видеопамять удобно сегментировать по регистру ES. В этом случае для вывода на экран можно использовать строковые команды, в которых на операнд-приемник указывает пара регистров ES:DI (см. п. 3.2, 3.3).

Рассмотрим в качестве примера программу вывода текстового сообщения на экран:

```
cseg   segment
       assume cs:cseg, ds:cseg
;текст сообщения с атрибутами, которое будет выведено на экран
message db 'H',7,'e',7,'l',7,'l',7,'o',7,',',7,' ',7,'W',7,'o',7,'r',7,'l',7,'d',7,'!',7
message_l = $-message                ;длина сообщения в байтах
rest_scr = (80*25*2-message_l)/2     ;длина оставшейся части экрана в словах
start:  mov dx, 0B800h               ;база сегмента видеопамяти
        mov es, dx                   ;ES = B800h
        mov dx, cseg                 ;ds будет указывать на cseg,
        mov ds, dx                   ;так как сообщение расположено в нем
        mov si, offset message       ;DS:[SI] — указатель на сообщение
        xor di, di                   ;ES:[DI] — указатель на видеопамять
        mov cx, message_l            ;CX — длина сообщения (счетчик цикла)
        rep movsb            ;цикл вывода на экран: ES:[DI] ← DS:[SI], CX=CX-1
        mov ax, 0720h                ;пробел (20h) серый на черном фоне (07h)
        mov cx, rest_scr             ;CX — остаток экрана
        rep stosw                    ;очистка остатка экрана: ES:[DI] ← AX, CX=CX-1
pause: in al, 60h                    ;цикл, пауза для просмотра результата
        cmp al, 01h                  ;пауза заканчивается по нажатию Esc
        jne pause
        mov ax, 4C00h
        int 21h
cseg   ends
        end start
```

В директиве определения переменной message в апострофах указаны символы сообщения. Компилятор по соответствующим адресам расположит ASCII-коды этих символов. Байт атрибутов у всех них один и тот же: 07h — серый (неярко-белый) символ на черном фоне. Символ $ для компилятора — текущее значение смещения. REP — это префикс повторения, он заставляет процессор выполнить предваряемую им строковую команду CX раз. Таким образом реализуется цикл, счетчиком которого является регистр CX. В каждом витке цикла CX декрементируется, и цикл заканчивается, когда он ста-

новится равным нулю. После запуска программы с левого края верхней строки экрана будет выведено сообщение:

Hello, World!

Еще один пример — вывод символа со всеми возможными атрибутами режима 3:

```
code  segment
      assume cs:code
main: mov dx, 0B800h   ;база видеопамяти
      mov es, dx       ;дополнительный сегмент — видеопамять
      xor di, di       ;DI — смещение в видеосегменте
;очистка экрана перед выводом (белым цветом):
      mov ax, 0FF20h
      mov cx, 80*25
      rep stosw
;инициализация регистров для вывода:
      mov ax, 0030h    ;30h — ASCII-код нуля
      xor di, di
      mov cx, 16
;внешний цикл, вывод строк:
c1:   mov bx, cx  ;сохранение CX в начале нового витка внешнего цикла
      mov cx, 16
c2:   mov es:[di], ax    ;внутренний цикл, вывод символов в строке
      inc di             ;перемещение указателя видеопамяти
      inc di             ;на следующую позицию
      inc ah             ;новый атрибут
      loop c2            ;следующий виток внутреннего цикла
      add di, 160-32     ;перевод указателя на следующую строку
      mov cx, bx  ;восстановление CX перед новым витком внешнего цикла
      loop c1            ;следующий виток внешнего цикла
pause: in al, 60h        ;пауза для просмотра результата
      cmp al, 01h        ;пауза заканчивается по нажатию Esc
      jne pause
```

```
;очистка экрана перед выходом из программы (черным цветом):
        xor di, di
        mov cx, 80*25
        mov ax, 0720h
        rep stosw
        mov ax, 4C00h
        int 21h
code    ends
        end main
```

В результате выполнения программы, начиная с верхнего левого угла, на экране будет выведена матрица нулей с различными атрибутами размером 16×16. Если бит 7 байта атрибутов определяет яркость цвета фона (бит 3 регистра управления режимом контроллера атрибутов равен 0), по строкам меняется цвет фона от черного до белого, по столбцам — цвет символа (рис. 4.58).

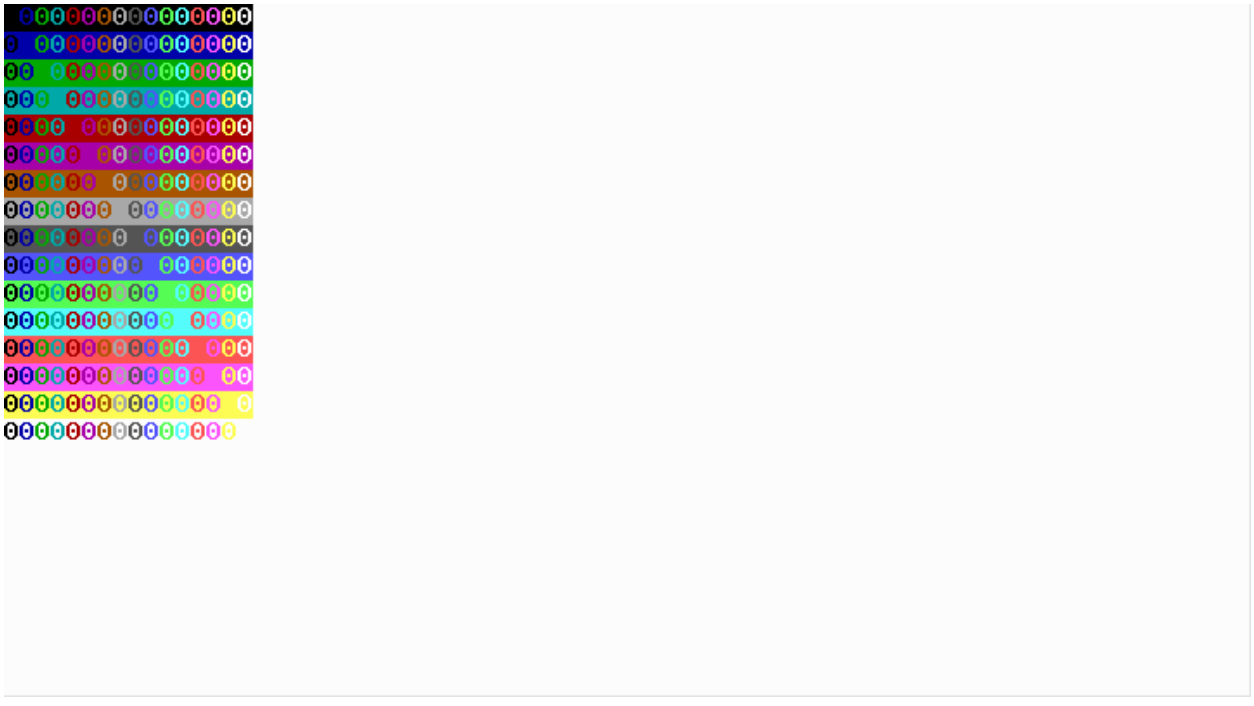


Рис. 4.58. Атрибуты символов в видеорежиме номер 3

В этом задании необходимо на языке ассемблера написать программу вывода своей фамилии на экран в определенном месте с указанными атрибутами.

5. ПАРАЛЛЕЛИЗМ И МНОГОЗАДАЧНОСТЬ. КОМПЛЕКСНЫЕ ЭВМ

5.1. ТЕРМИНОЛОГИЯ И КЛАССИФИКАЦИЯ В ОБЛАСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Виды и уровни параллелизма

Параллелизм является важным фактором увеличения производительности ЭВМ:

$$E = I/T = (I/Clock) \cdot (Clock/T) = IPC \cdot F,$$

где I — количество выполненных команд;

T — время;

Clock — количество тактов;

F — тактовая частота;

IPC — Instructions Per Clock, количество команд, выполняемых за такт.

Количество команд, выполняемых за один такт, тем больше, чем больше степень параллелизма на уровне команд. Но, помимо этого, существуют и другие виды параллелизма. Вычислительные системы — сложные системы. Они включают множество аппаратных и программных элементов разного типа, уровней и подсистем. Поэтому можно ввести различные виды и уровни параллелизма. Их может быть сколько угодно, но обычно выделяются следующие виды:

− параллелизм на уровне битов (BLP, Bit-Level Parallelism);

− параллелизм на уровне данных (DLP, Data-Level Parallelism);

− параллелизм на уровне команд (ILP, Instruction-Level Parallelism);

− параллелизм на уровне задач (TLP, Task-Level Parallelism, Task parallelism, Function parallelism, Control parallelism).

Параллелизм на уровне битов означает, что размер операнда команды процессора больше одного бита. Он реализуется во всех процессорах и, соответственно, компьютерах в большинстве команд. Но есть и команды работы с отдельными битами. Они часто выделяются в отдельную группу. Если говорить не об обработке, а о передаче данных, то по этому признаку выделяются параллельные и последовательные интерфейсы.

Параллелизм на уровне данных распространен в задачах обработки сигналов. Его реализует векторный (матричный) процессор. Векторный процессор — это процессор, операндами команд которого являются упорядоченные массивы данных — векторы. В такой терминологии обычный процессор называется скалярным. В большинстве современных универсальных процессоров имеются векторные обрабатывающие устройства. В процессорах семейства x86 это устройство MMX и его расширения. Многие специализированные процессоры: сигнальные, графические, физические и др. — также являются векторными.

Параллелизм на уровне команд означает, что в один момент времени выполняется несколько команд процессора. Он реализуется, например, при наличии хотя бы одного конвейера. В этом случае одновременно выполняется столько команд, сколько стадий у конвейера, но каждая — на своей стадии. Это достигается благодаря тому, что на каждой стадии конвейера задействованы свои аппаратные средства. Если конвейеров несколько, то на каждой стадии выполняется несколько команд, и такой процессор называется суперскалярным. Для суперскалярного процессора *IPC* больше единицы.

Параллелизм на уровне задач состоит в том, что общая вычислительная задача разбивается на несколько относительно самостоятельных подзадач, каждая из которых выполняется на отдельном исполнительном устройстве или в отдельный промежуток времени. Таким образом, на уровне задач можно говорить о распараллеливании в пространстве (по различным обрабатывающим блокам) или во времени (time-sharing), когда один обрабатывающий блок по очереди

с определенным интервалом выполняет несколько задач. В этом случае каждой задаче выделяется такой ресурс, как процессорное время. Понятие задачи (task) является для процессора одним из основных. В процессорах имеются аппаратные и программные средства для реализации многозадачности (multitasking) запуска, выполнения и переключения между задачами. К ним относятся такие элементы системной архитектуры, как регистр задач (TR, Task Register), сегмент состояния задачи (TSS, Task State Segment), шлюз задачи и др. [58]. Операционная система на основе понятия задачи строит такие элементы, как процессы (processes) и потоки (threads).

Конечно, все виды параллелизма: BLP, DLP, ILP и TLP — комбинируются для одного и того же компьютера или процессора. В частности, все они присутствуют в современных процессорах семейства x86. Но основным видом параллелизма является параллелизм на уровне задач, и поэтому под параллельной обработкой информации понимается способ реализации вычислений, при котором в каждый момент времени в активном состоянии находится сразу несколько процессов. Таким образом, обязательным условием параллельной обработки является наличие нескольких (больше одного) обрабатывающих (исполнительных) устройств. Такими устройствами могут быть:

− отдельные компьютеры (конечно, связанные между собой);

− отдельные компьютерные узлы с процессором, памятью и интерфейсами для взаимодействия между собой;

− отдельные физические процессоры одного компьютера;

− отдельные ядра одного физического процессора;

− отдельные логические процессоры одного ядра физического процессора.

Основная задача при параллельной обработке — синхронизация (координация) процессов, т. е. обеспечение их правильного взаимодействия.

Надо отметить, что как синонимы термина “multitasking” используются “concurrent processing” («параллельная обработка»)

и “concurrent computing” («параллельные вычисления»). Однако, как уже было сказано выше, многозадачность может быть реализована с разделением по времени на скалярном одноядерном процессоре. Под терминами “concurrent processing” или “concurrent computing” понимается именно многозадачность с разделением по времени. Таким образом, на многопроцессорных компьютерах или многоярерных процессорах многозадачность позволяет выполнять больше задач, чем имеется процессоров или ядер.

Классификация ЭВМ по признаку параллелизма

Существует множество различных способов классификации вычислительных систем по признаку параллелизма. Самой известной и логичной является классификация Флинна. Компьютерная программа — это совокупность команд и данных. В классификации Флинна используются, соответственно, два признака — параллелизм команд и параллелизм данных (табл. 5.1):

− SISD (Single Instruction stream, Single Data stream), одиночный поток команд, одиночный поток данных;

− SIMD (Single Instruction stream, Multiple Data stream), одиночный поток команд, множественный поток данных;

− MISD (Multiple Instruction stream, Single Data stream), множественный поток команд, одиночный поток данных;

− MIMD (Multiple Instruction stream, Multiple Data stream), множественный поток команд, множественный поток данных.

Таблица 5.1

Классификация Флинна

		Параллелизм данных	
		–	+
Параллелизм команд	–	SISD	SIMD
	+	MISD	MIMD

Параллелизм означает одновременное выполнение нескольких потоков команд или данных. При этом надо иметь в виду, что количество потоков может различаться на разных уровнях.

SISD — это компьютер с одним процессором без каких-либо элементов параллелизма: отдельных физических ядер, логических процессоров, конвейеров, устройств векторной обработки и т. д. SISD-процессор выполняет за один такт не более одной команды над скалярными операндами. Таким образом, в математической терминологии SISD-процессор называется скалярным. Суперскалярным называется процессор, способный выполнить за один такт более одной команды. В нем имеются однотипные узлы, одновременно обрабатывающие разные команды на одной стадии. Это еще не векторный процессор, но уже и не скалярный. В семействе x86 первым суперскалярным процессором был Pentium (1993 г.). В нем было реализовано два целочисленных конвейера и конвейеризированное устройство обработки вещественных чисел с плавающей точкой. Соответственно, он мог выполнять до трех команд за один такт. Снаружи поток команд один, но внутри он разделяется по разным конвейерам. При этом существенным моментом является то, что это распределение выполняется чисто аппаратно по внутренним алгоритмам процессора и программно никак не проявляется.

SIMD — это векторный процессор. Операнды команд такого процессора представляют собой векторы. Одна команда одновременно обрабатывает все компоненты двух векторов, т. е. сразу векторы в целом (рис. 5.1).

При этом, есть команды горизонтальной обработки и вертикальной обработки. Они позволяют эффективно обрабатывать массивы данных с различной организацией. Вертикальное сложение и/или вычитание выполняется над соответствующими элементами разных векторов. Например, при вертикальном сложении двух XMM регистров размером 128 бит каждый, результатом будет XMM регистр размером 128 бит, в котором первые 16 бит будут содержать сумму первых

элементов из обоих исходных регистров, вторые 16 бит — сумму вторых элементов и т. д. Есть и команды, которые одновременно выполняют и сложение, и вычитание (рис. 5.2).

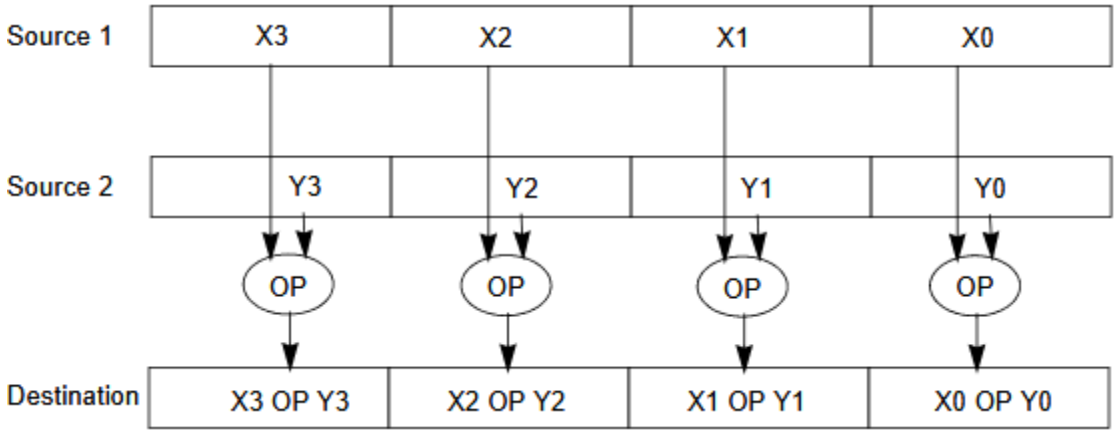


Рис. 5.1. Схема выполнения SIMD-команд

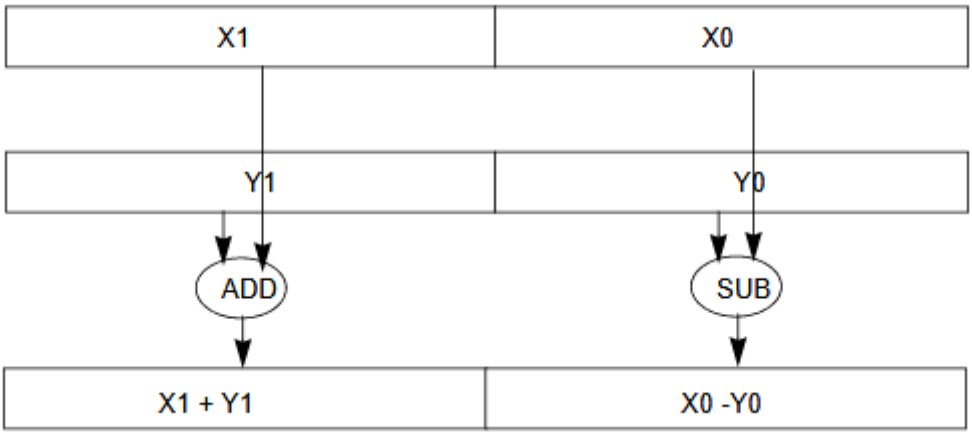


Рис. 5.2. Вертикальное вычисление (команда ADDSUBPD)

Горизонтальные вычисления выполняются над парами соседних элементов вектора. Например, при горизонтальном сложении двух XMM регистров, размером 128 бит каждый, результатом будет XMM регистр размером 128 бит, в котором первые 16 бит будут содержать сумму первых двух 16-битных элементов из обоих исходных регистров, вторые 16 бит — сумму вторых двух 16-битных элементов и т. д. (рис. 5.3).

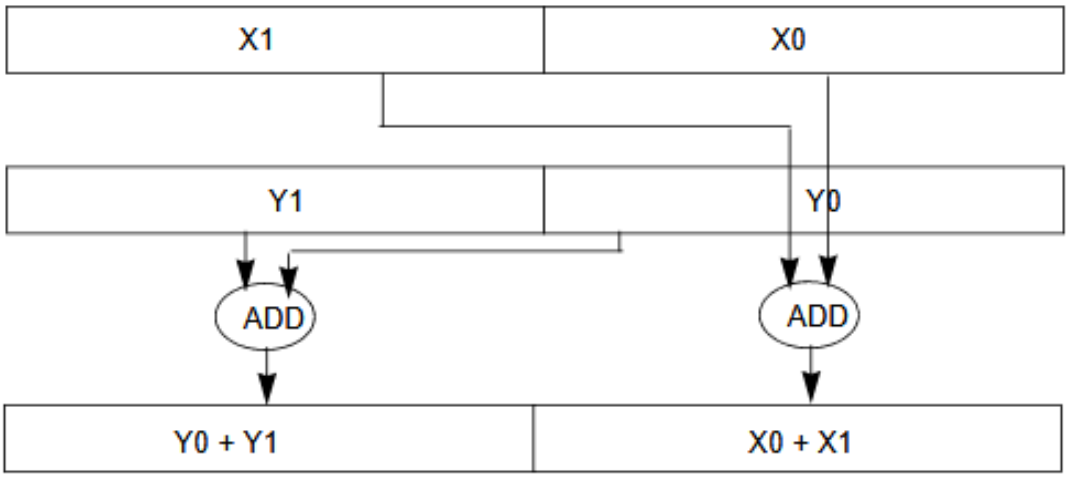


Рис. 5.3. Горизонтальное вычисление (команда HADDPD)

В семействе x86 блок вычислений SIMD появился в процессоре Pentium MMX (1996 г.). В последующих процессорах этого семейства технология SIMD развивалась. Она продолжает развиваться и в настоящее время. Ее развитие вначале носило название SSE (Streaming SIMD Extensions), а затем AVX (Advanced Vector Extensions) (табл. 5.2):

− Pentium III (1999 г.): SSE (128-битные регистры XMM);

− Pentium 4 (2000 г.): SSE2 (144 новые команды);

− Pentium 4 Prescott (2004 г.): SSE3 (13 новых команд: пять операций с комплексными числами, пять потоковых операций над числами с плавающей запятой с использованием формата AOS, две команды для синхронизации потоков и одна команда для кодирования видео);

− Xeon Woodcrest (2006 г.): SSSE3 (Supplemental Streaming SIMD Extensions);

− Core 2 (2006 г.): SSE4 (47 новых команд);

− Core 2 Penryn (2007 г.): SSE4.1 (20 новых команд);

− Core i7 Nehalem (2008 г.): SSE4.2 (11 новых команд);

− Core i Sandy Bridge (второе поколение Core i, 2011 г.): AVX (256-битные регистры YMM, новые команды и новая схема кодирования команд);

− Core i Haswell (четвертое поколение Core i, 2013 г.): AVX2 (расширение большинства команд до 256 бит, добавление новых команд);

− Xeon Phi x200 (Knights Landing) (2016 г.): AVX-512 (512-битные регистры ZMM, новые команды).

Таблица 5.2

Регистры расширений AVX

Биты 511–256	**Биты 255–128**	**Биты 127–0**
ZMM0	YMM0	XMM0
ZMM1	YMM1	XMM1
ZMM2	YMM2	XMM2
ZMM3	YMM3	XMM3
ZMM4	YMM4	XMM4
ZMM5	YMM5	XMM5
ZMM6	YMM6	XMM6
ZMM7	YMM7	XMM7
ZMM8	YMM8	XMM8
ZMM9	YMM9	XMM9
ZMM10	YMM10	XMM10
ZMM11	YMM11	XMM11
ZMM12	YMM12	XMM12
ZMM13	YMM13	XMM13
ZMM14	YMM14	XMM14
ZMM15	YMM15	XMM15
ZMM16	YMM16	XMM16
ZMM17	YMM17	XMM17
ZMM18	YMM18	XMM18
ZMM19	YMM19	XMM19
ZMM20	YMM20	XMM20
ZMM21	YMM21	XMM21
ZMM22	YMM22	XMM22
ZMM23	YMM23	XMM23

Биты 511–256	Биты 255–128	Биты 127–0
ZMM24	YMM24	XMM24
ZMM25	YMM25	XMM25
ZMM26	YMM26	XMM26
ZMM27	YMM27	XMM27
ZMM28	YMM28	XMM28
ZMM29	YMM29	XMM29
ZMM30	YMM30	XMM30
ZMM31	YMM31	XMM31

Регистры YMM в качестве младшей 128-битной части включают регистры XMM. С появлением расширения AVX-512 разрядность регистров была увеличена до 512 бит, а их число — до 32. Регистры XMM и YMM являются младшими частями регистров ZMM.

MISD — это архитектура, в которой несколько исполнительных модулей одновременно выполняют несколько потоков команд над одним и тем же потоком данных. Вычислительная система MISD-архитектуры имеет несколько функциональных блоков, работающих параллельно, каждый из которых имеет на входе один набор данных. Таким образом, MISD-архитектура подразумевает аппаратную избыточность, что позволяет создавать на ее основе надежные отказоустойчивые вычислительные системы. Например, процессор Pentium имеет такое свойство, как проверка на основе функциональной избыточности (Functional Redundancy Checking), правда, оно реализуется не в рамках одного процессора, а с использованием еще одного процессора — контролера. Система с двумя процессорами Pentium, работающими в режиме контроля, имеет MISD-архитектуру.

MIMD — это вычислительные системы, в которых одновременно выполняются различные потоки команд над различными потоками данных. Все суперкомпьютеры из списка Top500 относятся к этой архитектуре. В свою очередь MIMD-компьютеры могут быть

классифицированы по более детальным признакам: особенностям аппаратно-программной конфигурации, организации памяти, интерфейсам и др.

Основная классификация MIMD-систем такова:

− системы с общей разделяемой памятью (shared memory):

− с одной единой памятью и одной шиной (bus-based) для доступа к памяти со стороны каждого из процессоров;

− с иерархической памятью (hierarchical) и иерархической шиной, где каждый из процессоров через иерархию шин имеет доступ к любой части запоминающего устройства;

− системы с распределенной памятью (distributed memory), состоят из узлов (node), включающих процессор и запоминающее устройство, процессор узла непосредственно взаимодействует только с памятью узла:

− узлы расположены в вершинах многомерного куба (hypercube interconnection network);

− узлы расположены форме двумерной решетки (mesh interconnection network).

В системах с разделяемой памятью каждый процессор может иметь собственный кэш, поэтому необходимо обеспечивать их когерентность (согласованность). Типы систем с разделяемой памятью:

− SMP/UMA (Symmetric MultiProcessing / Uniform Memory Access, симметричная мультипроцессорность / однородный доступ к памяти);

− NUMA (Non-Uniform Memory Access, неоднородный доступ к памяти), время доступа к памяти определяется ее расположением по отношению к процессору;

− ASMP (Asymmetric MultiProcessing, асимметричная мультипроцессорность), отдельные процессоры используются для различных задач.

Выделяется также такой класс MIMD-систем с разделяемой памятью, как гибридные системы. Гибридная вычислительная систе-

ма — система с гетерогенной аппаратной вычислительной архитектурой (HSA, Heterogeneous System Architecture), в которой используются процессоры различного типа, например, универсальные центральные процессоры и графические процессоры, соединенные общей шиной, с разделяемой памятью и задачами.

Наиболее распространенными типами MIMD-систем с распределенной памятью являются:

− массивно-параллельная архитектура (MPP, Massive Parallel Processing);

− кластер (cluster).

MPP-система состоит из отдельных узлов, содержащих процессор, локальную оперативную память и коммуникационные средства.

Кластер — это параллельная или распределенная система, которая состоит из нескольких связанных между собой компьютеров и используется как единый, унифицированный компьютерный ресурс.

Имеется также еще один класс MIMD-систем с распределенной памятью — Grid-системы. Grid-система представляет собой совокупность вычислительных ресурсов, расположенных в различных местах, но связанных коммуникационными средствами для выполнения общей задачи. В отличие от кластеров, в Grid-системах узлы более независимы, разнообразны и могут быть достаточно удалены друг от друга географически.

Для систем с распределенной памятью используется также термин SPMD (Single Program, Multiple Data), поскольку в таких системах множество узлов одновременно выполняют разные участки кода одной и той же программы для различных наборов данных. Узлы взаимодействуют друг с другом посредством обмена сообщениями, при этом чаще всего используется API (Application Programming Interface) такого типа, как MPI (Message Passing Interface).

5.2. ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Суперкомпьютеры рейтинга Top500

Обширную интересную информацию о характеристиках параллельных вычислительных систем дает рейтинг Top500 [20]. В нем представлены наиболее высокопроизводительные вычислительные системы мира — суперкомпьютеры, которые являются, конечно, параллельными системами. Анализ их характеристик и архитектуры позволяет составить представление о текущем положении дел и тенденциях развития в области высокопроизводительных вычислений (High-Performance Computing, HPC). Рейтинг существует с 1993 г., и на его сайте накоплен большой материал, позволяющий получать множество различных статистик. Рейтинг выходит два раза в год — в июне и ноябре. Для определения реальной производительности компьютеров в нем используется тест HPL (High Performance Linpack) (см. п. 2.1). Конечно, не все суперкомпьютеры по тем или иным причинам присутствуют в рейтинге Top500. Например, по соображениям секретности в нем не представлен суперкомпьютер Национального центра управления обороной России, являющийся одним из самых мощных военных суперкомпьютеров.

Вначале в рейтинге доминировали MIMD-системы с разделяемой памятью класса SMP (Symmetric MultiProcessing) (рис. 5.4). Но с течением времени картина изменилась. Начиная с июня 2012 г. в рейтинге присутствуют только системы с распределенной памятью: кластеры и MPP-системы. В 66-м рейтинге (ноябрь 2025 г.) доля кластеров составляет 86,3 %, MPP — 13,7 %. Однако по общей производительности их доли более близкие: кластеры — 50,8 %, MPP — 49,2 % [20]. То есть, в среднем производительность MPP-системы больше, чем кластера.

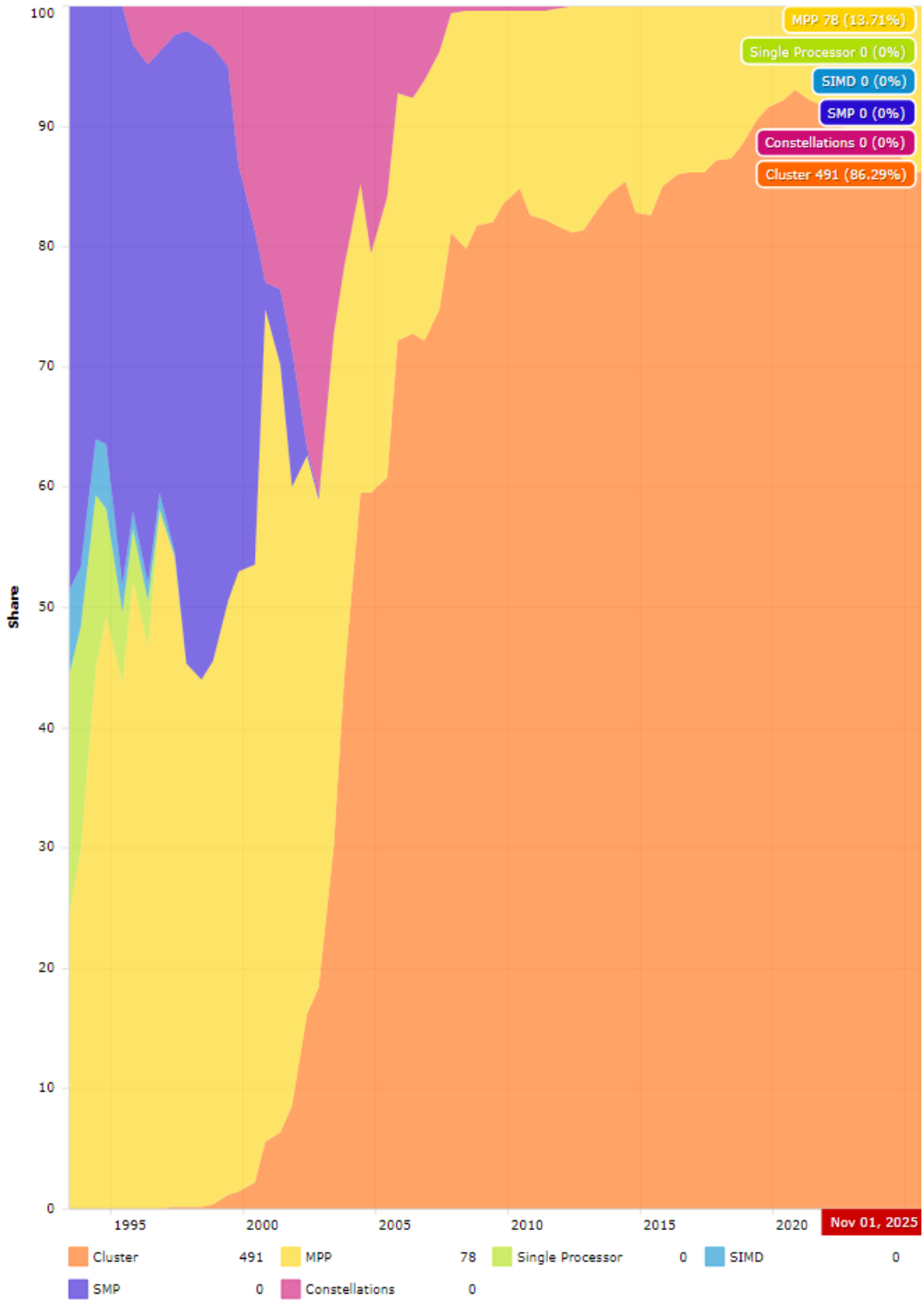


Рис. 5.4. Динамика архитектуры суперкомпьютеров

92,6 % суперкомпьютеров основаны на процессорах архитектуры x86, но более производительных, чем те, которые используются в персональных компьютерах: 285 систем (57 %) используют процессоры Intel Xeon, 178 (35,6 %) — AMD EPYC (микроархитектура Zen) различных поколений. Они содержат десятки ядер и в целом имеют

более продвинутые архитектурные характеристики: большее число каналов с памятью, кэш-память до четырех уровней и т. д. Например, самые производительные процессоры EPYC 4-го поколения (Zen-4) имеют 96 ядер и 192 потока, Xeon Phi серии x200 — 72 ядра и 288 потоков (4 потока на одно ядро). Процессоры Xeon Phi изначально были созданы как сопроцессоры-ускорители, но затем доработаны для использования их в качестве универсальных центральных процессоров (рис. 5.5) [83].

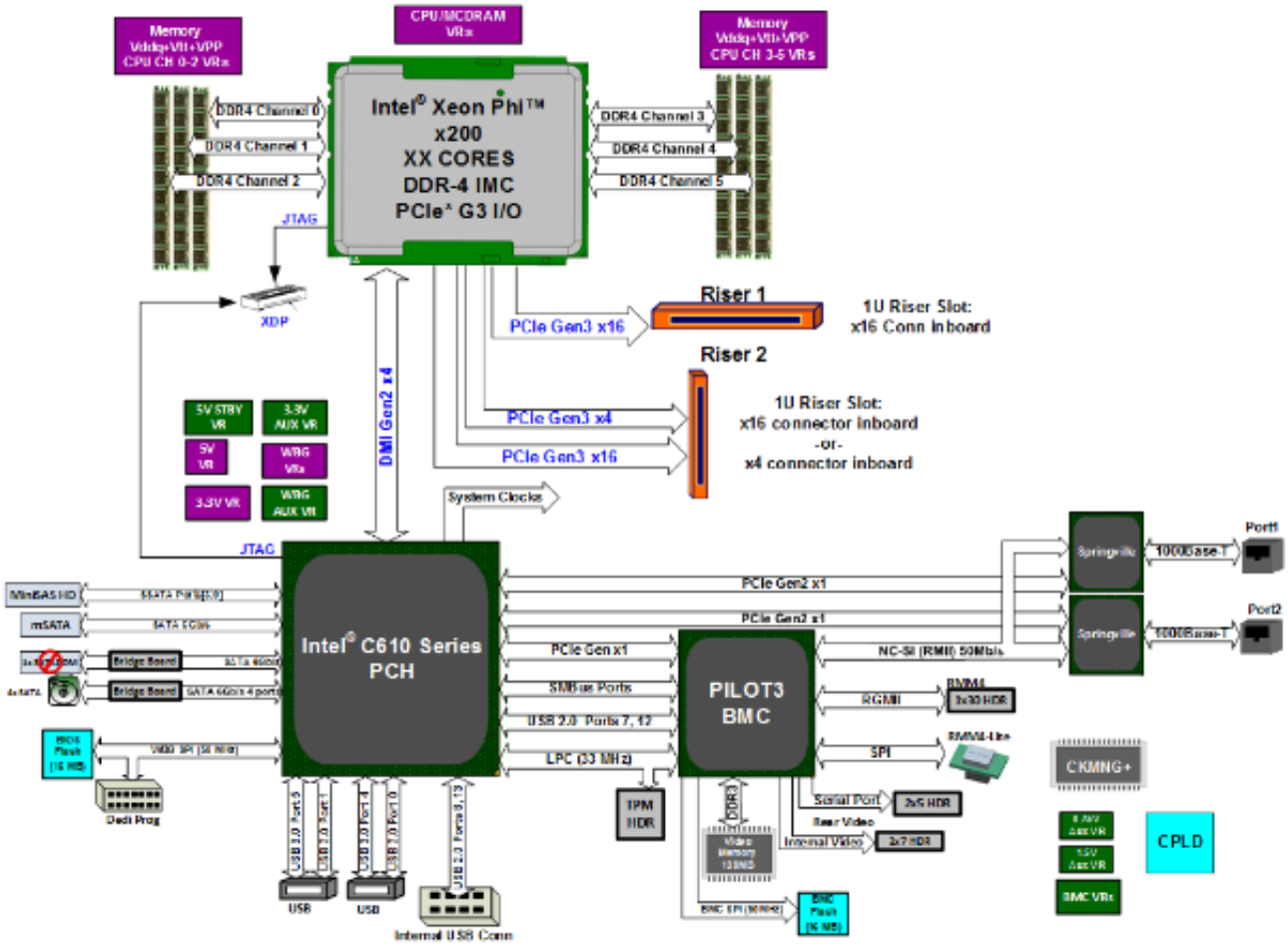


Рис. 5.5. Структура системы на процессоре Xeon Phi серии x200

Доля суперкомпьютеров на основе процессоров архитектуры x86 в настоящее время снижается (рис. 5.6).

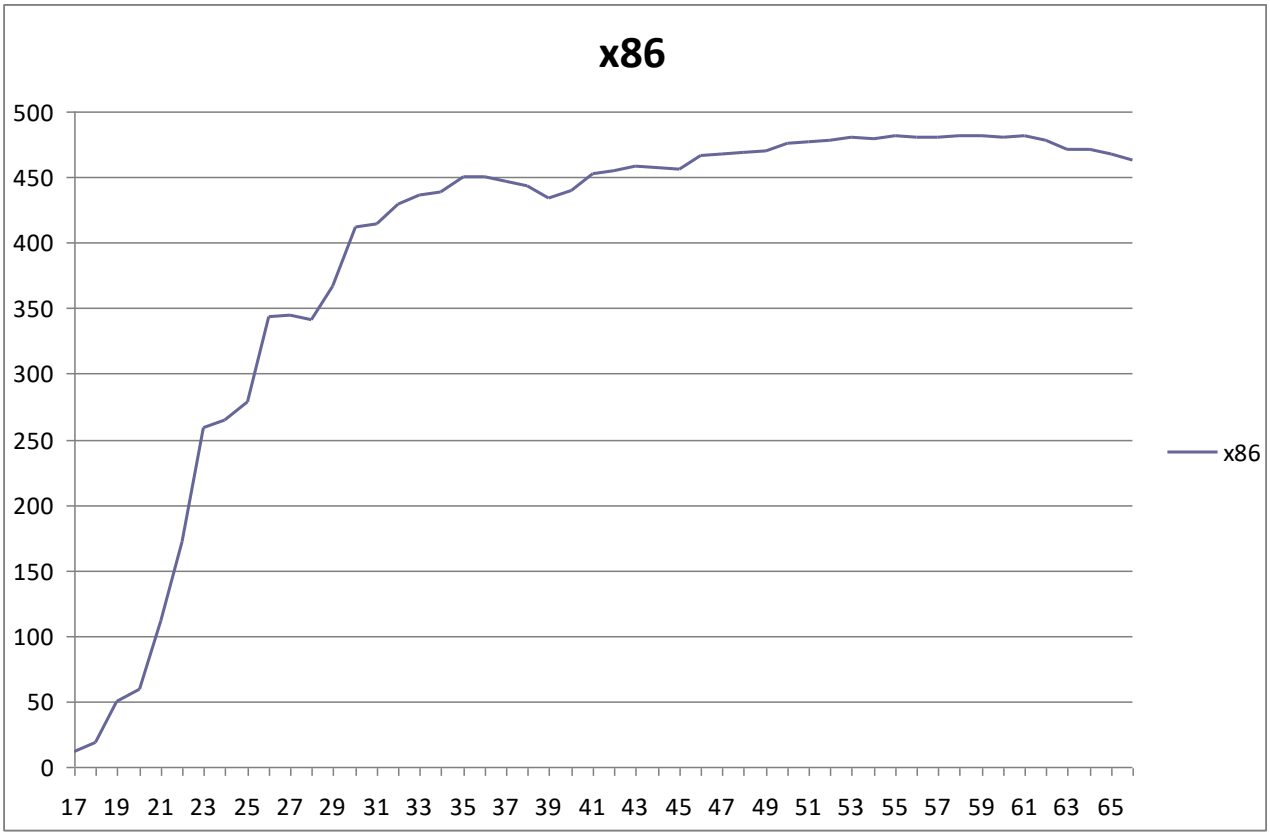


Рис. 5.6. Изменение числа систем на процессорах x86 в Top500

Это происходит за счет увеличения числа систем с процессорами NVIDIA Grace.

Растет число систем с ускорителями, причем, как уже было сказано, в большинстве систем в качестве ускорителей используются графические процессоры NVIDIA.

Для связи узлов суперкомпьютеров между собой используются различные типы сетевых соединений, большинство из которых имеет пропускную способность 100 Гбит/с и выше. Наиболее распространенным из них является 100-гигабитный Ethernet, популярны также Slingshot (см. далее), InfiniBand и Intel Omni-Path (табл. 5.3) [20]. InfiniBand — последовательный интерфейс с коммутируемой звездообразной топологией. Каждый вычислительный узел содержит адаптер HCA (Host Channel Adapter), подключаемый к процессору по интерфейсу PCI Express. Периферийные устройства могут содержать адаптер TCA (Target Channel Adapter). Соединения между адаптерами

и коммутаторами осуществляются линиями дифференциальных пар медных проводов или оптических кабелей. InfiniBand используется, например, в суперкомпьютерных системах на процессорах NVIDIA Grace Hopper (см. рис. 3.3, 3.4). Контроллеры и коммутаторы InfiniBand производятся компаниями Intel и Mellanox (поглощена в 2019 г. компанией NVIDIA).

Таблица 5.3

Десять наиболее популярных сетевых соединений в суперкомпьютерах 66-го рейтинга Top500

№ п/п	Сеть (Interconnect)	Число систем	Доля систем (%)	Rmax (GFlops)	Rpeak (GFlops)	Число ядер
1	100G Ethernet	60	12	216 253 310	448 537 700	6 970 768
2	Slingshot-11	52	10,4	6 949 917 670	10 546 027 101	49 295 392
3	InfiniBand HDR	45	9	448 320 400	660 502 440	4 973 416
4	InfiniBand NDR400	35	7	1 543 559 000	2 214 548 440	5 591 200
5	InfiniBand NDR	29	5,8	527 005 740	779 223 810	3 060 608
6	InfiniBand NDR200	28	5,6	559 910 920	773 474 939	3 546 608
7	Intel Omni-Path	20	4	103 792 368	156 788 893	2 328 508
8	Mellanox HDR InfiniBand	19	3,8	223 319 940	310 828 145	3 037 380
9	InfiniBand HDR100	19	3,8	87 209 640	126 993 772	2 534 800
10	25G Ethernet	18	3,6	58 934 710	115 119 060	1 419 572

Суперкомпьютер Frontier

С июня 2022 г. по июнь 2024 г. лидером рейтинга Top500 являлся суперкомпьютер Frontier. В ноябре 20024 г. его сместил на вторую строчку компьютер El Capitan. Frontier имеет архитектуру Cray EX235a и относится к классу MPP. Его производителем является компания HPE (Hewlett Packard Enterprise). Это был первый компьютер, преодолевший рубеж производительности в один эксафлопс при 64-разрядных вычислениях с плавающей точкой. При пиковой произ-

водительности в 1,67982 Eflop/s его реальная производительность по тесту HPL составляла 1,194 Eflop/s. Потребляемая мощность — 22,703 МВт. Таким образом, энергоэффективность Frontier составляла 52,592 GFlops/watts, и по этому показателю он занимал 8-е место в рейтинге Green500. В дальнейшем его производительность была увеличена за счет добавления дополнительных узлов. Основу Frontier составляют 64-ядерные процессоры AMD EPYC с тактовой частотой 2 ГГц. В качестве ускорителей используются графические процессоры AMD Instinct MI250X. Схема одного узла Frontier представлена на рис. 5.7 [84].

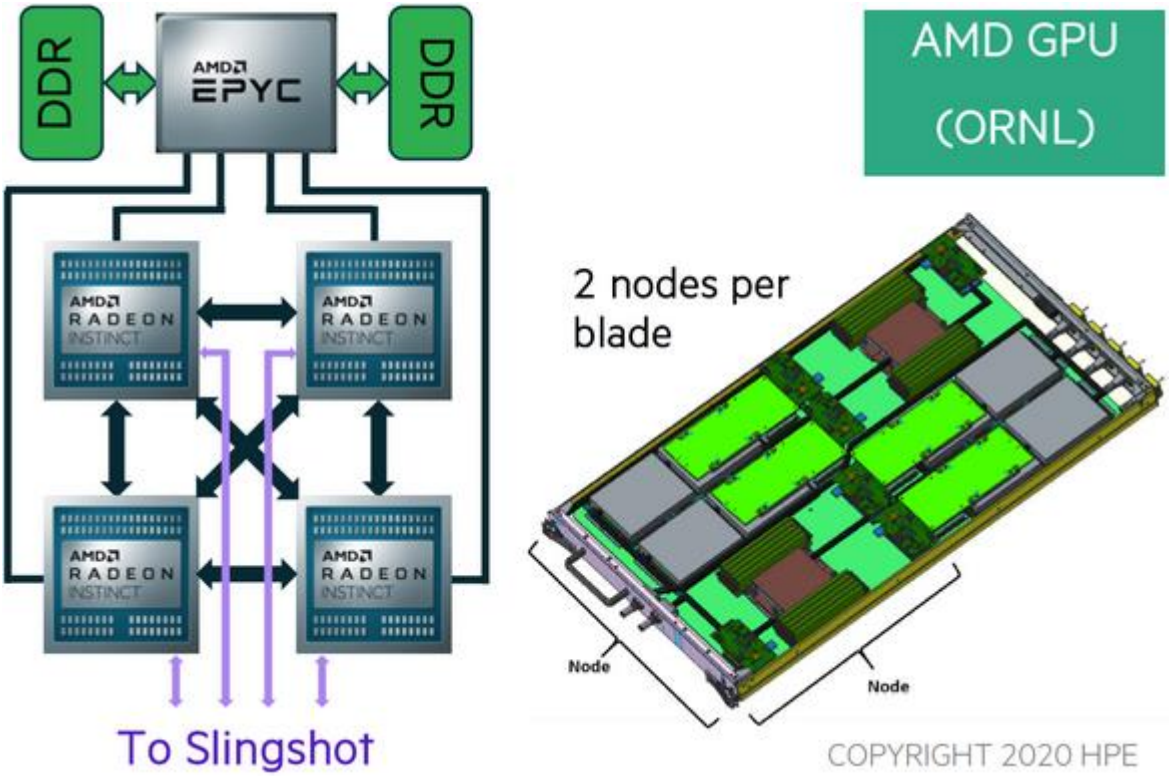


Рис. 5.7. Схема узла Frontier

Для связи узлов между собой используется сетевое соединение (интерконнект) Slingshot-11. Это одна из самых высокопроизводительных сетей, которую используют в настоящее время 52 суперкомпьютера рейтинга Top500. Slingshot-11разработана компанией HPE. Она использует API OpenSHMEM — открытую библиотеку параллельных вычислений для систем класса SPMD и модели памяти

PGAS (Partitioned Global Address Space, разделенное глобальное адресное пространство), содержащую множество необходимых функций для работы и взаимодействия узлов (рис. 5.8) [85].

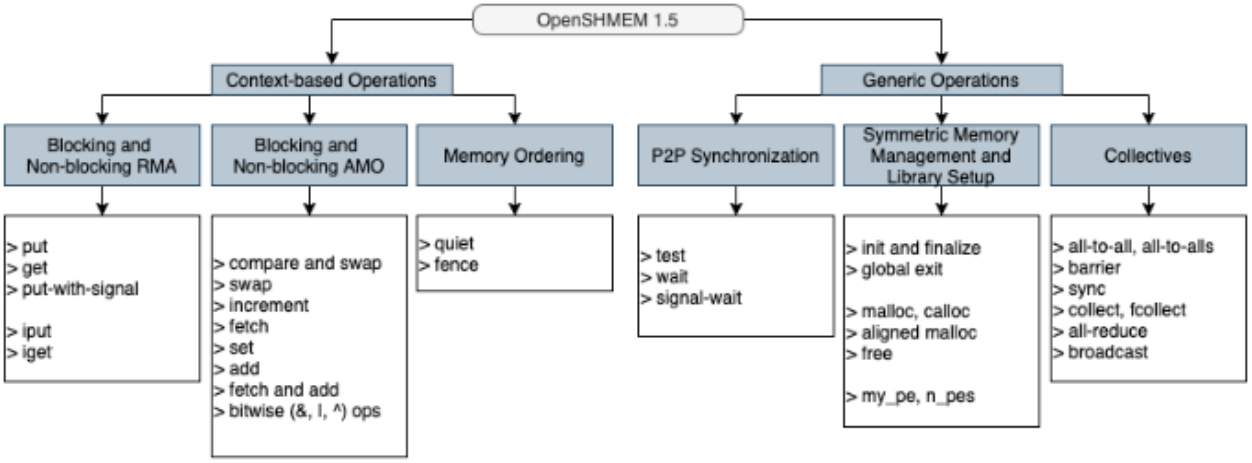


Рис. 5.8. Состав API OpenSHMEM версии 1.5

Модель PGAS имеет определенные преимущества по сравнению с плоской моделью. HPE на основе OpenSHMEM разработала ее специализированную реализацию Cray OpenSHMEMX (рис. 5.9).

Frontier размещен в Окриджской национальной лаборатории министерства энергетики США и используется в рамках проектов EXAALT (Exascale Atomistics for Accuracy, Length and Time) и WDMApp (Whole Device Modeling of Magnetically Confined Fusion Plasmas). В проекте EXAALT осуществляется моделирование различных процессов на атомарном уровне с целью создания новых материалов. Проект WDMApp направлен на разработку единой модели плазмы во всем объеме термоядерного реактора типа токамака (тороидальной камеры с магнитными катушками), включая центральную и граничную области плазмы, что раньше осуществлялось только по отдельности из-за отсутствия необходимых вычислительных мощностей.

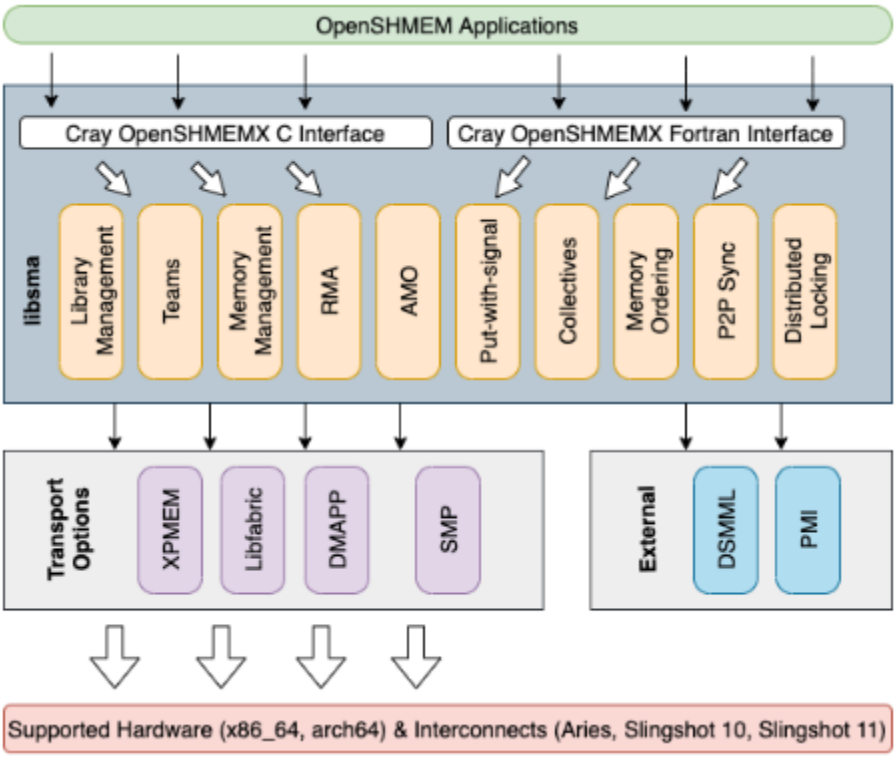


Рис. 5.9. Особенности Cray OpenSHMEMX

Архитектура суперкомпьютера Sunway TaihuLight

Sunway TaihuLight занимал первое место в рейтинге Top500 с июня 2016 г. по ноябрь 2017 г. и только в 62-м ноябрьском рейтинге 2023 г. выбыл из первой десятки, оказавшись на 11-м месте. В настоящее время (66-й рейтинг, ноябрь 2025 г.) он находится на 24 месте. Это MPP-система, разработанная китайским центром NRCPC (National Research Center of Parallel Computer Engineering & Technology). В отличие от большинства других суперкомпьютерных систем рейтинга почти все его элементы: процессор, сетевое соединение, операционная система — собственной разработки. Китай, как и Россия, испытывает политическое и экономическое давление со стороны США. Так, в апреле 2021 г. семь китайских суперкомпьютерных компаний были добавлены в черный список Бюро промышленности и безопасности Министерства торговли США. Поэтому преодоление технологической зависимости от США для Китая также является важной задачей.

Основные характеристики Sunway TaihuLight таковы [86, 87]:

− 40 960 узлов, каждый на основе одного 260-ядерного процессора Sunway (ShenWei) SW26010 (всего 10649600 ядер);

− пиковая производительность 125,44 Пфлопс;

− реальная производительность по тесту Linpack 93 Пфлопс;

− 1 310 720 ГБ оперативной памяти DDR3-2133 (32 ГБ на узел);

− сетевое соединение Sunway;

− операционная система Sunway Raise;

− энергопотребление 15,3 МВт;

− водяное охлаждение;

− энергоэффективность 6 Гфлопс/Вт;

− стоимость 270 млн долларов США;

− занимаемая площадь 605 м^2.

По тесту HPCG (High-Performance Conjugate Gradients) Sunway TaihuLight занял только третье место в том же июньском рейтинге 2016. Его производительность HPCG составляет 0,3 % от пиковой производительности, что говорит о недостаточной пропускной способности памяти и сети. Еще одна характеристика — количество операций с плавающей точкой на один байт передаваемых данных из памяти. Ее значение для процессора SW26010 составляет 22,4 Flops(DP)/Byte (DP — Double Precision, 64 бита). Для сравнения, значение этого показателя Intel Xeon Phi Knights Landing — 7,2 Flops(DP)/Byte. Это говорит о некотором дисбалансе и избыточности SW26010, т. е. для большинства реальных приложений высокая пиковая производительность Sunway TaihuLight просто не понадобится.

Процессор SW26010 разработан в шанхайском центре ICC (Shanghai High Performance Integrated Circuit Design Center). ***Его основные характеристики таковы:***

− корпус FCBGA3832;

− 260 ядер;

− тактовая частота 1,45 ГГц;

− производительность 3,06 Тфлопс;

− RISC-архитектура;

− разрядность 64 бита;

− SIMD 256 бит;

− микроархитектура с внеочередным исполнением команд;

− интерфейсы PCI Express 3.0, Gigabit Ethernet, JTAG.

260 ядер процессора SW26010 разделены на четыре группы (рис. 5.10).

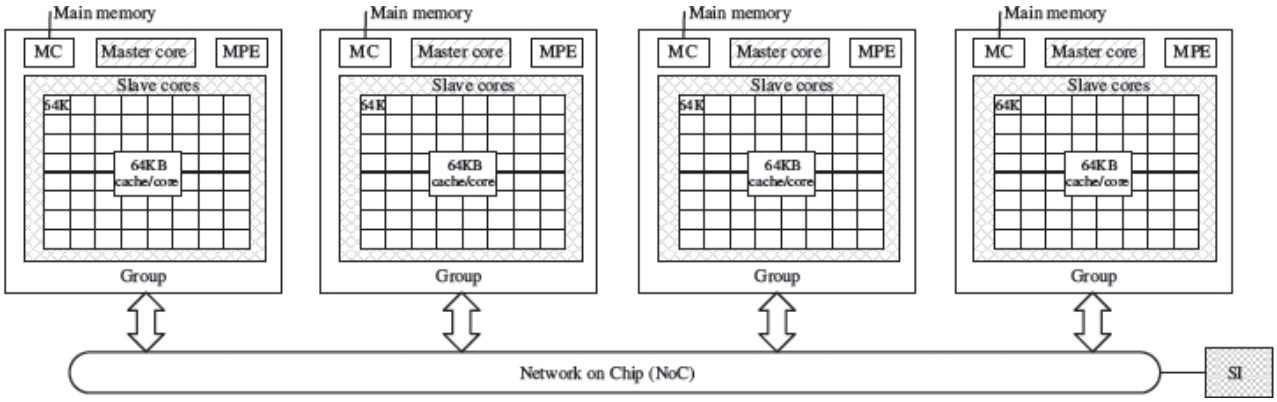


Рис. 5.10. Архитектура процессора SW26010

Каждая группа содержит:

− 64 вычислительных ядра общего назначения (Computing Processing Elements, CPE), организованных в виде двумерной решетки (mesh) 8x8;

− одно управляющее ядро (Management Processing Element, MPE);

− mesh-контроллер, осуществляющий синхронизацию и обработку прерываний;

− контроллер памяти (Memory Controller, MC).

Группа имеет собственную память размером в 8 Гбайт, с которой она связана посредством контроллера памяти 128-битной шиной. Таким образом, для памяти DDR3-2133 пропускная способность составляет 34,1 Гбайт/с на одну группу. Группы связаны между собой сетью на кристалле (Network on Chip, NoC).

Процессоры связаны между собой и с другими устройствами системным интерфейсом (System Interface, SI), который представляет

собой шину PCI Express 3.0 с пропускной способностью в дуплексном режиме 16 Гбайт/с.

Ядра CPE и MPE имеют 64-разрядную RISC-архитектуру с поддержкой 256-разрядных векторных вычислений. Различаются они режимами работы, структурой кэш-памяти и числом конвейеров. ***Характеристики ядра MPE:***

− два режима работы: пользовательский и системный;

− кэш команд L1 32 Кбайта;

− кэш данных L1 32 Кбайта;

− кэш L2 256 Кбайт;

− два конвейера FPU с пиковой производительностью 23,2 Гфлопс при частоте 1,45 ГГц (16 команд за такт).

Характеристики ядра CPE:

− один режим работы — пользовательский;

− кэш команд L1 16 Кбайт;

− сверхоперативная память SPM (Scratch Pad Memory) типа SRAM 64 Кбайта;

− один конвейер FPU с пиковой производительностью 11,6 Гфлопс при частоте 1,45 ГГц (8 команд за такт).

Каждое ядро поддерживает только один поток команд.

Sunway TaihuLight имеет 5-уровневую иерархическую аппаратную структуру (рис. 5.11):

− система — 40 стоек;

− стойка (cabinet) — 4 суперузла;

− суперузел (supernode) — 32 коммутационные панели;

− коммутационная панель (board) — 4 платы;

− плата (card) — 2 узла (node);

− 1 узел содержит один процессор (260 ядер) и имеет пиковую производительность 3,06 Тфлопс;

− 1 суперузел — 256 узлов, 783,97 Тфлопс;

− 1 стойка — 4 суперузла, 3,14 Пфлопс;

− 1 система — 40 стоек, 160 суперузлов, 40 960 узлов, 10 649 600 ядер, 125,44 Пфлопс.

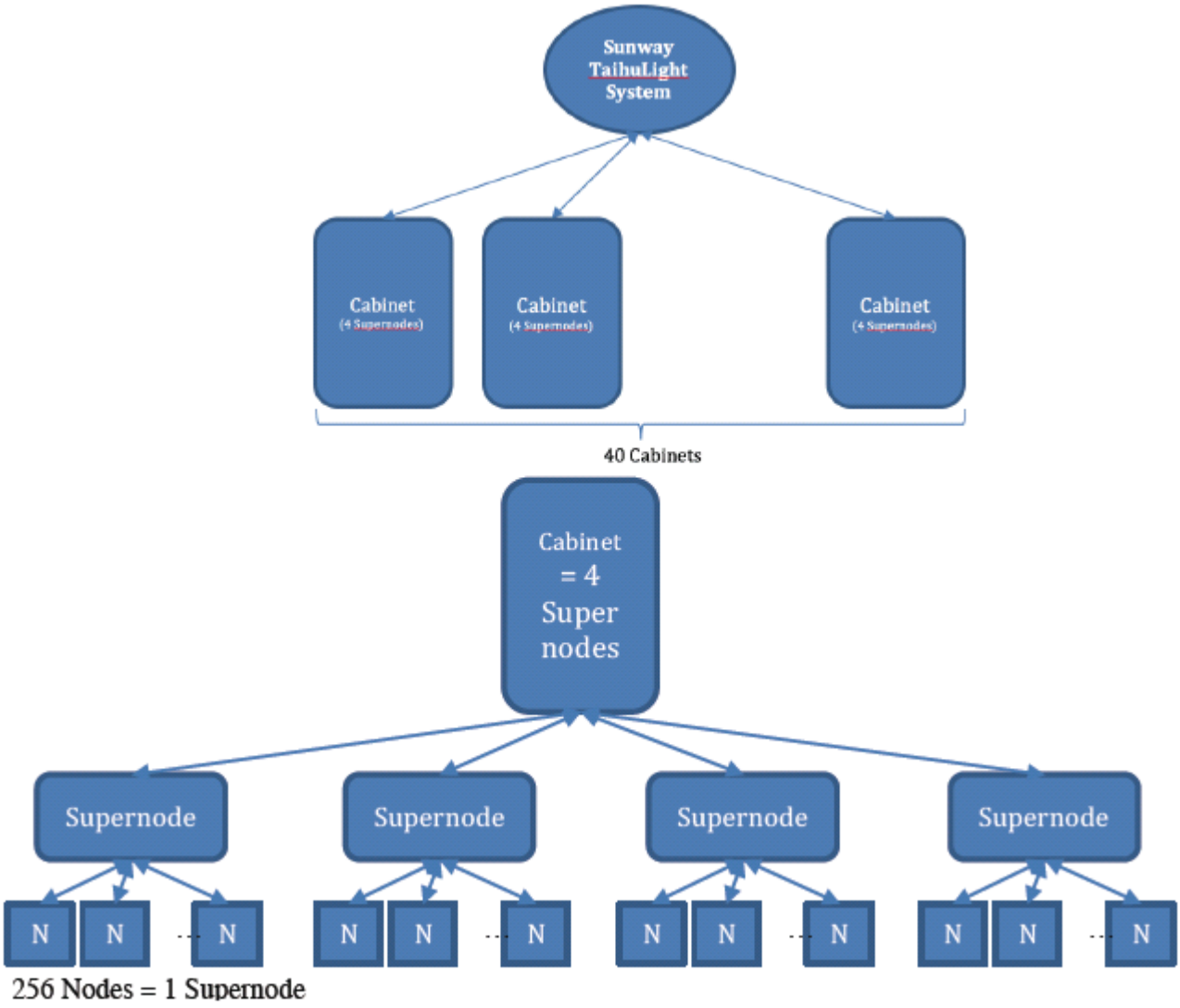


Рис. 5.11. Иерархическая структура Sunway TaihuLight

Общая архитектура Sunway TaihuLight изображена на рис. 5.12.

В составе Sunway TaihuLight выделяются следующие подсистемы:

– вычислительная;
– сетевая;
– периферийная;
– обслуживания и диагностики;
– энергоснабжения и охлаждения.

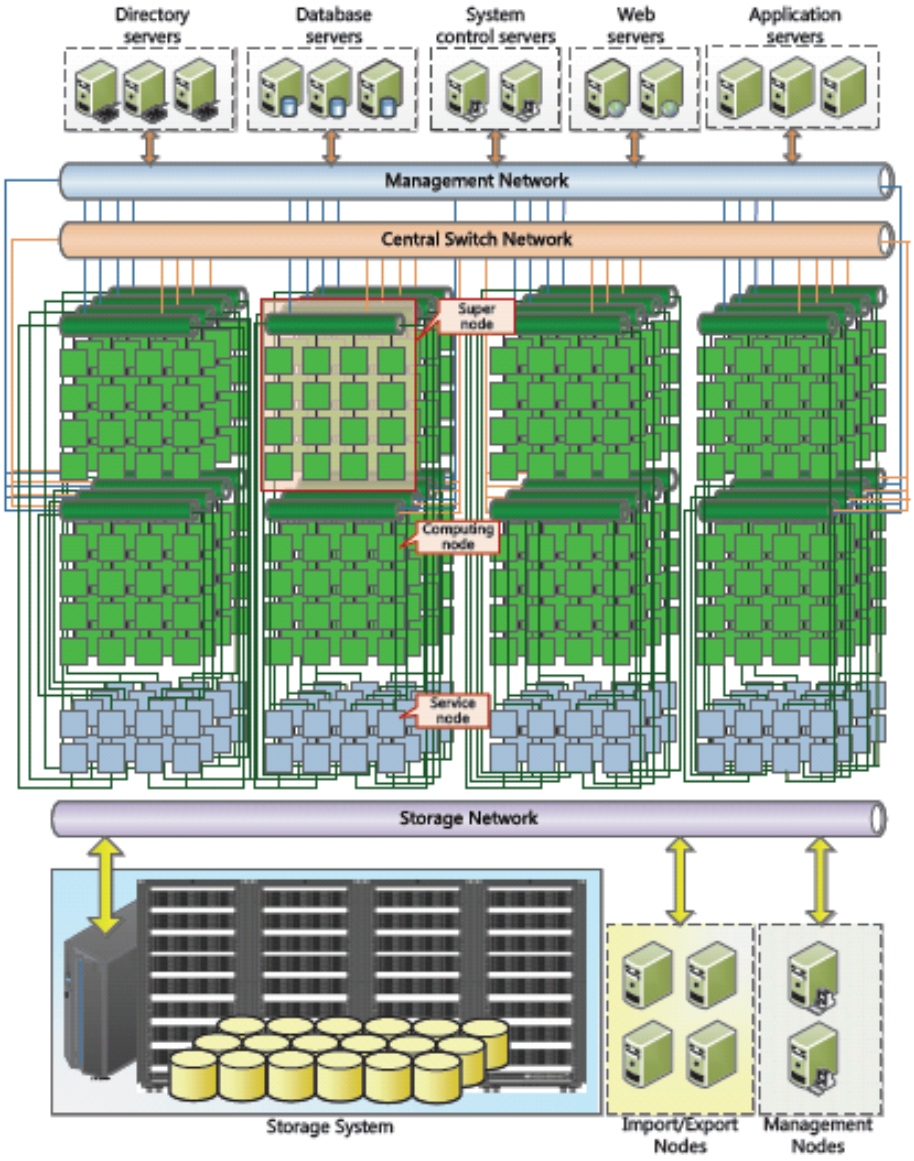


Рис. 5.12. Общая архитектура Sunway TaihuLight

Сетевая система Sunway TaihuLight имеет три уровня:

– верхний — центральная коммутационная сеть (central switching network);

– средний — сеть суперузла (super node network);

– нижний — сеть разделения ресурсов (resource-sharing network).

Периферийная подсистема включает сетевое хранилище информации емкостью 20 Пбайт и периферийную систему управления, включающую системную консоль, управляющий сервер и соответствующую сеть.

Подсистема обслуживания и диагностики содержит встроенные модули для тестирования и обслуживания узлов, суперузлов и всей системы в целом. ***Она выполняет следующие функции:***

– мониторинг окружающей среды;

– обнаружение и запись ошибок;

– обеспечение безопасности;

– мониторинг и визуализацию состояния системы на основе распределенного динамического сбора данных.

Программное обеспечение Sunway TaihuLight включает (рис. 5.13):

– операционную систему Sunway Raise OS 2.0.5, основанную на Linux;

– компиляторы C/C++, Fortran;

– средство автоматической векторизации;

– математические библиотеки;

– среду параллельной разработки и компиляции на основе стандарта OpenACC.

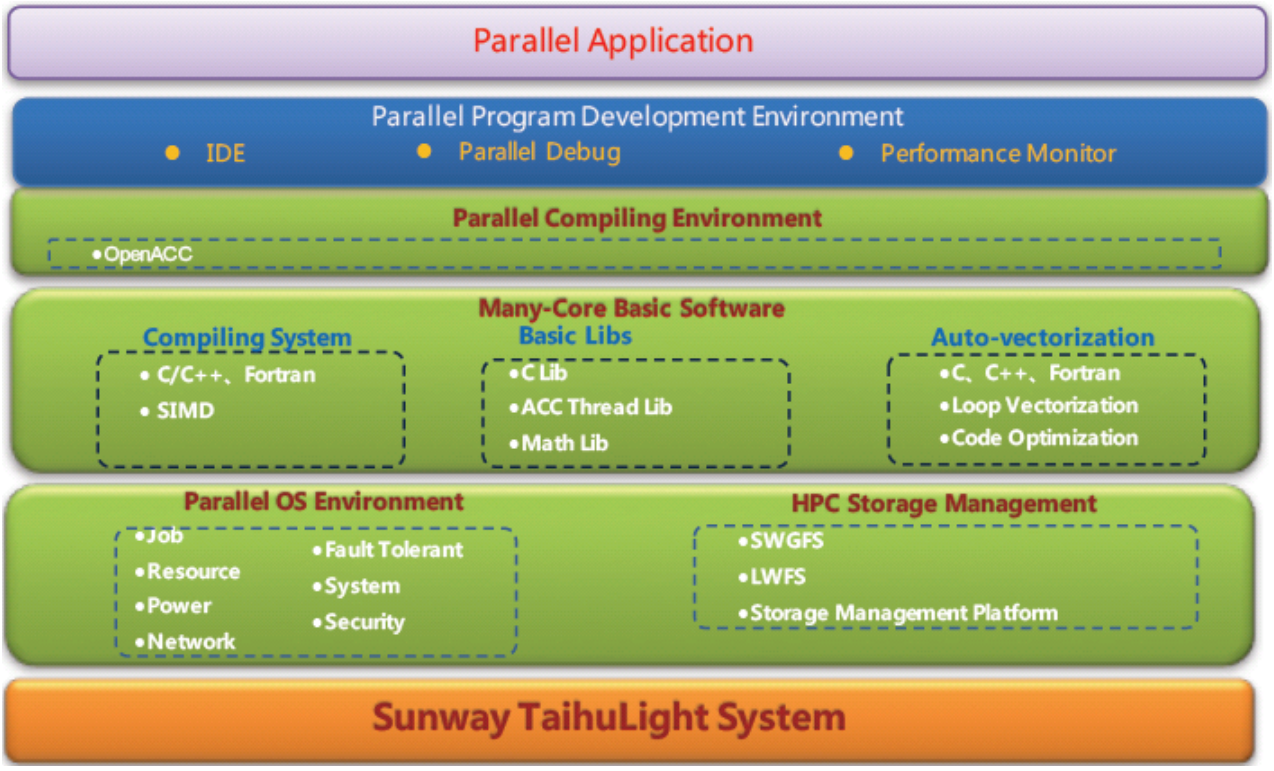


Рис. 5.13. Программное обеспечение Sunway TaihuLight

OpenACC в Sunway TaihuLight применяется для распараллеливания внутри группы ядер. Для распараллеливания на уровне узла используется MPI (Message Passing Interface, интерфейс передачи сообщений) или OpenMP.

На Sunway TaihuLight используется множество приложений для решения следующих задач:

− рефакторинг атмосферной модели CAM (Community Atmospheric Model);

− полностью неявный негидростатический динамический решатель для моделирования формирования облаков в атмосфере (достигнутая производительность 1,5 Пфлопс);

− высокоэффективное численное моделирование глобальных поверхностных волн с высоким разрешением (достигнутая производительность 30,07 Пфлопс);

− петамасштабное атомистическое моделирование кремниевой нанопроволоки (достигнутая производительность 14,7 Пфлопс);

− крупномасштабное фазово-полевое моделирование для огрубленной динамики на основе уравнения Кана — Хилларда с вырожденной мобильностью (достигнутая производительность 39,68 Пфлопс).

5.3. ПРАКТИЧЕСКИЕ ЗАДАНИЯ К РАЗДЕЛУ 5

Задание 1. Динамика рейтинга Top500 по основным техническим характеристикам суперкомпьютерных систем

Сайт Top500.org предоставляет множество полезной и интересной информации по архитектуре высокопроизводительных параллельных компьютерных систем. На ее основе можно оценить текущий уровень развития вычислительной техники, а также отследить динамику по различным показателям и выявить тенденции в этой об-

ласти. Меню STATISTICS позволятт представить эту информацию в удобном табличном и графическом виде (рис. 5.14).

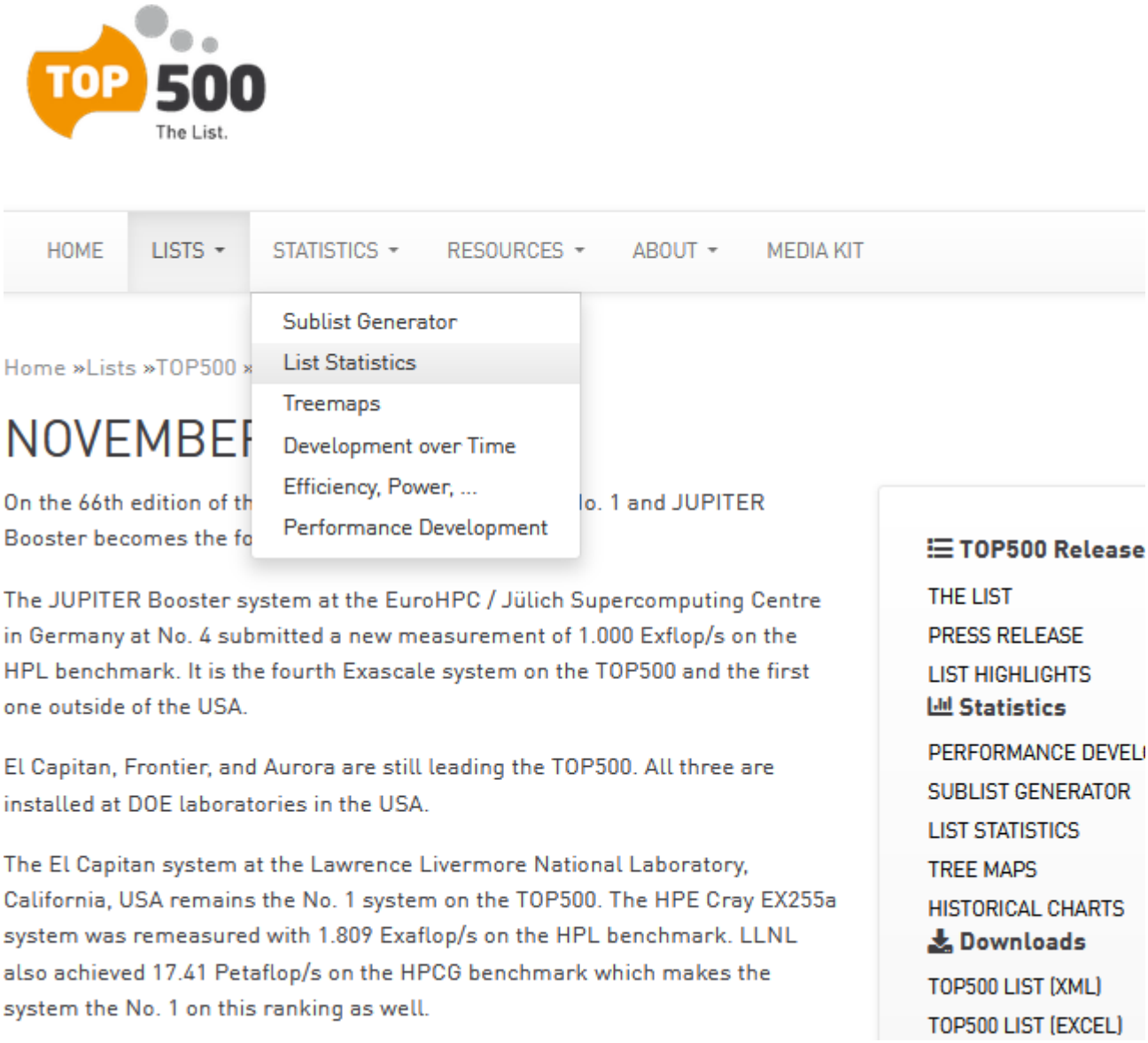
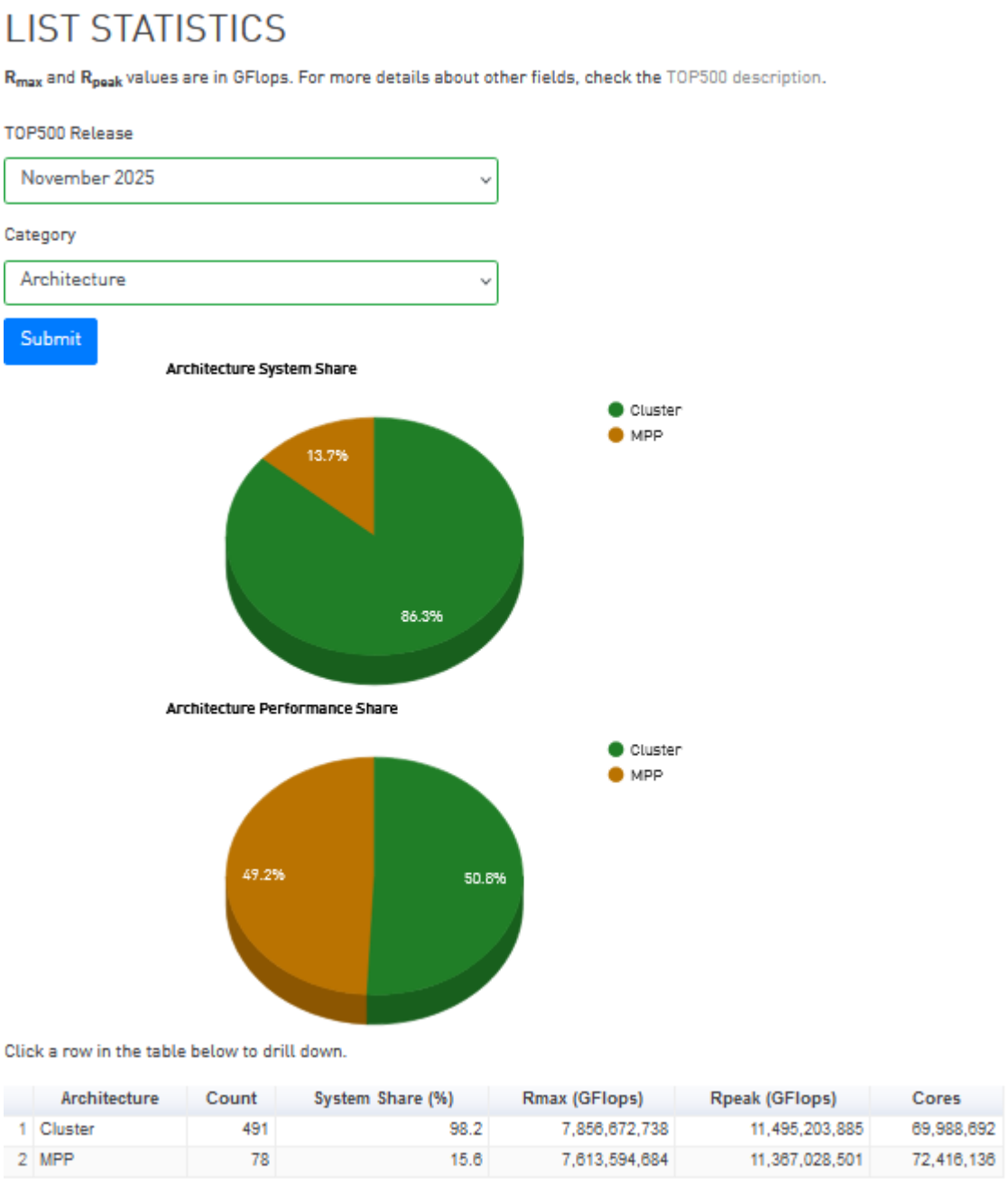


Рис. 5.14. Меню STATISTICS сайта Top500

Например, с помощью опции List Statistics можно построить круговые диаграммы и таблицу, показывающие долю систем в выбранном рейтинге с заданной характеристикой, например, типом архитектуры (рис. 5.14).



	Architecture	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	Cluster	491	98.2	7,856,672,738	11,495,203,885	69,988,692
2	MPP	78	15.6	7,613,594,684	11,367,028,501	72,416,136

Рис. 5.14. Доли систем различной архитектуры в 66-м рейтинге

Далее, можно получить график изменения числа систем, имеющих определенный тип архитектуры (см. п. 5.1) (рис. 5.15).

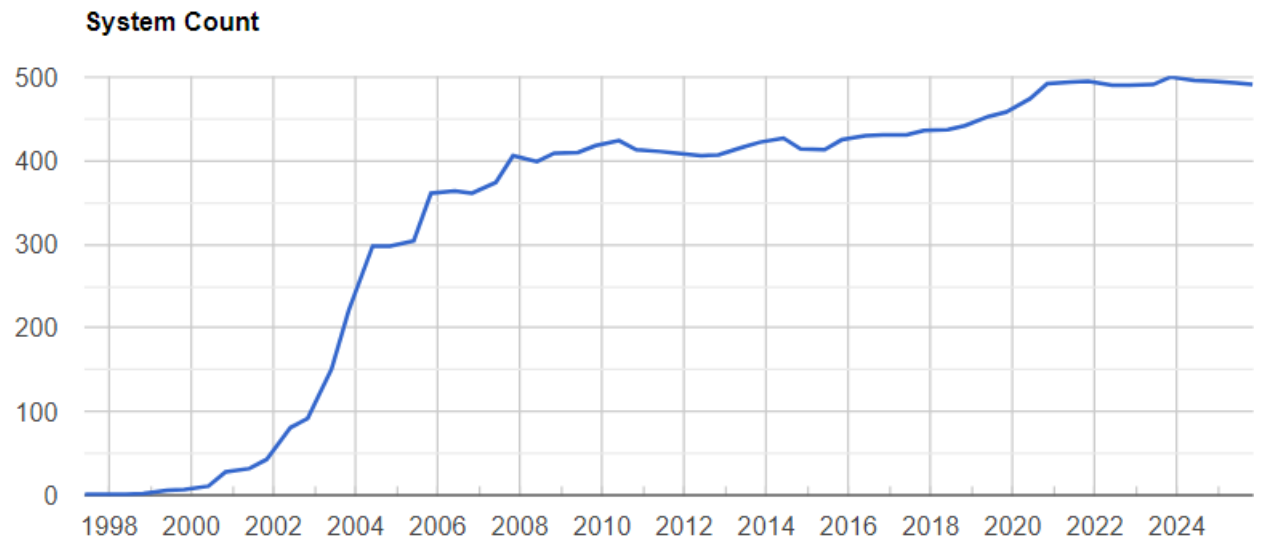


Рис. 5.15. Динамика числа кластеров в рейтинге Top500

Такие же диаграммы, таблицы и графики можно построить для множества различных характеристик. В данном задании необходимо исследовать динамику рейтинга Top500 по заданным характеристикам суперкомпьютеров: типу архитектуры в целом (Cluster, MPP, SMP и т. д.), архитектуре используемых центральных процессоров и сопроцессоров, межсоединениям (сети, интерконнекту), операционной системе, производителю, расположению (регион, страна, учреждение или организация), отрасли (сегменту) и области применения (Application Area).

Полученные данные представить в табличном и графическом виде, отметить тенденцию изменения числа систем в рейтинге с заданной характеристикой.

Задание 2. Получение списка систем с заданными характеристиками

Опция Sublist Generator меню STATISTICS позволяет получить список компьютеров, имеющих заданные характеристики в указанном рейтинге (рис. 5.16).

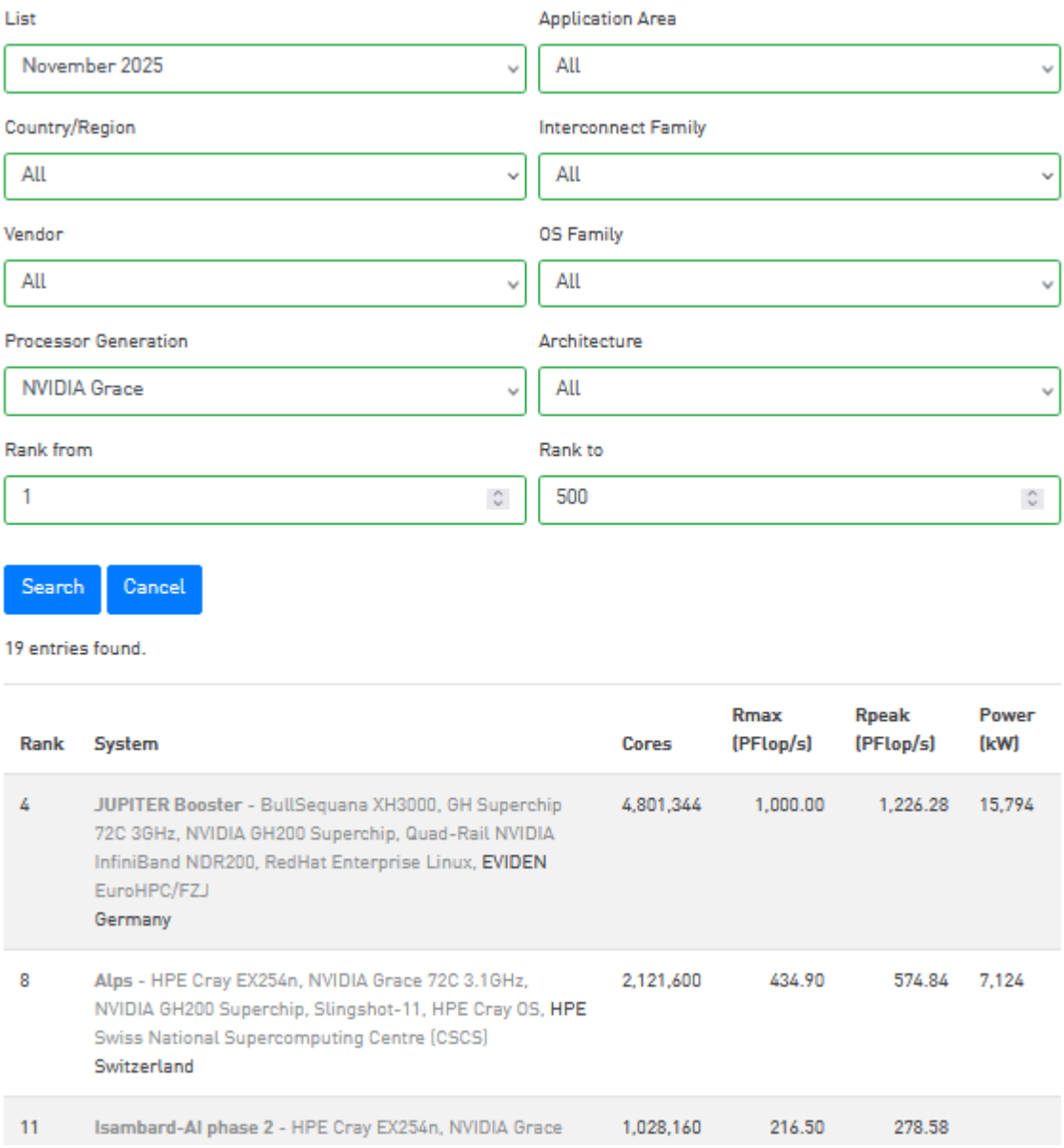


Рис. 5.16. Список систем с процессорами NVIDIA Grace в 66-м рейтинге

В данном задании необходимо получить списки систем в указанном рейтинге с заданными характеристиками.

ЛИТЕРАТУРА

1. Толковый словарь по вычислительным системам / под редакцией В. Иллингуорта [и др.] ; перевод с английского А.К. Белоцкого [и др.] ; под редакцией Е.К. Масловского. — Москва : Машиностроение, 1990. — 560 с.

2. *Винер, Н.* Кибернетика, или управление и связь в животном и машине / Н. Винер. — 2-е изд. — Москва : Наука : Главная редакция изданий для зарубежных стран, 1983. — 344 с.

3. *Wiener, N.* Cybernetics or control and communication in the animal and the machine / N. Wiener. — 2nd ed. — Camb. Mass. (MIT Press), 1985. — 212 p.

4. *Лийв, Э.Х.* Инфодинамика. Обобщенная энтропия и негэнтропия / Э.Х. Лийв. — Таллинн, 1998. — 200 с.

5. *Scott, M. Hitchcock.* Time and Information / M. Hitchcock Scott // Quantum Physics: quant-ph/0012017. — 2001. — Vol. 2. — 4 Feb.

6. *Scott, M. Hitchcock*. Time and Information: The Origins of “Time” from Information Flow In Complex Systems / M. Hitchcock Scott // Quantum Physics: quant-ph/0111025. — 2001. — Vol. 2. — 5 Nov.

7. *Пригожин, И.* Порядок из хаоса. Новый диалог человека с природой / И. Пригожин, И. Стенгерс. — 3-е изд. — Москва : Эдиториал УРСС, 2001. — 312 с.

8. Виртуальный компьютерный музей : [сайт]. — URL: https://www.computer-museum.ru.

9. *Нитусов, А.Ю.* Русский триггер / А.Ю. Нитусов // PC-Week. — 2010. — Апрель.

10. Страницы истории отечественных ИТ. Том 1 / составитель Э.М. Пройдаков. — Москва : Альпина Паблишер, 2015. — 265 с.

11. Страницы истории отечественных ИТ. Том 2 / составитель Э.М. Пройдаков. — Москва : Альпина Паблишер, 2016. — 234 с.

12. Страницы истории отечественных ИТ. Том 3 / составитель Э.М. Пройдаков. — Москва : Альпина Паблишер, 2017. — 240 с.

13. Страницы истории отечественных ИТ. Том 4 / составитель Э.М. Пройдаков. — Москва : Альпина Паблишер, 2018. — 224 с.

14. Страницы истории отечественных ИТ. Том 5 / составитель Э.М. Пройдаков. — Москва : Альпина Паблишер, 2019. — 224 с.

15. *Глушков, В.М.* Введение в кибернетику / В.М. Глушков. — Киев, 1964. — 323 с.

16. *Марков, А.А.* Теория алгорифмов. Том 42 / А.А. Марков : труды математического института им. В.А. Стеклова. — Изд-во АН СССР, 1954.

17. PCB Design and Circuit Simulator Software - Proteus : [website]. — URL: https://www.labcenter.com.

18. bochs: The Open Source IA-32 Emulation Project (Home Page) : [website]. — URL: http://bochs.sourceforge.net.

19. *Зубков, С.В.* Assembler / С.В. Зубков. — Москва : ДМК, 1999. — 640 с.

20. TOP500 : [website]. — URL: https://top500.org.

21. IO500 : [website]. — URL: https://io500.org.

22. Производительность суперкомпьютера «Говорун» в ОИЯИ достигла 2,2 Петафлопс // itWeek : [сайт]. — URL: https://www.itweek.ru/infrastructure/news-company/detail.php?ID=232537.

23. *Кушнеров, А.* Троичная цифровая техника. Ретроспектива и современность / А. Кушнеров. — Беэр-Шева, Израиль : Изд-во Университета им. Бен-Гуриона. — 28.10.2005.

24. *Джонсон, К.* Когнитивный компьютер IBM имитирует работу мозга / К. Джонсон // // itWeek : [сайт]. — URL: https://www.itweek.ru/idea/article/detail.php?ID=133074&ysclid=lxlvzrhr14286244451.

25. *Zasedatelev, A.V.* Single-photon nonlinearity at room temperature / A.V. Zasedatelev, A.V. Baranikov, D. Sannikov [etc.] // Nature. — 2021. — Vol. 597. — P. 493–497.

26. Национальный центр физики и математики : [website]. — URL: https://ncphm.ru/about/territory/laboratoriyu-fotonnykh-vychislitelnykh-ustroystv.php.

27. *Chua, L.O.* Memristor — The Missing Circuit Element / L.O. Chua // IEEE transactions on circuit theory. — 1971. — September. — Vol. ct-18. — № 5. — P. 507–519. — Doi:10.1109/TCT.1971.1083337.

28. *Kuekes, P.J.* Molecular crossbar latch : Patent US 6,586,965 B2 / P.J. Kuekes ; Hewlett Packard Development Company. — 2003. — Jul. 1.

29. *Kuekes, P.J.* The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits / P.J. Kuekes, D.R. Stewart, R.S. Williams // Journal of Applied Physics. — 2005. — Feb. 1. — № 97, 034301. — URL: https://doi.org/10.1063/1.1823026.

30. Memristive FG–PVA Structures Fabricated with the Use of High Energy Xe Ion Irradiation : Materials 2022, 15, 2085 / A.I. Ivanov, I.V. Antonova, N.A. Nebogatikova, A.Olejniczak // MDPI : [website]. — URL: https://doi.org/10.3390/ma15062085.

31. *Bhola, A.* Memristors and crossbar latches / A. Bhola, G. Kanitkar // ICWET '10 : Proceedings of the International Conference and Workshop on Emerging Trends in Technology. — 2010. — February. — P. 915–918. — URL: https://doi.org/10.1145/1741906.1742116.

32. Особенности применения наномемристорной логики для проектирования цифровых систем / Г. Кривуля, Е. Сыревич, И. Власов,О. Павлов // Вестник ХНТУ Инженерные науки. — 2013. — №1(46). — С. 280–286.

33. *Di Ventra, M.* Memcomputing: a computing paradigm to store and process information on the same physical platform / M. Di Ventra1, Yu.V. Pershin // Nature Physics. — 2013. — № 9. — P. 200–202.

34. *Pershin, Yu.V.* Self-organization and solution of shortest-path optimization problems with memristive networks // Yu.V. Pershin, M. Di Ventra // Phys. Rev. — 2013. — July. — № 88, 013305.

35. *Ди Вентра, М.* Просто добавь память / М. Ди Вентра, Ю. Першин // В мире науки. — 2016. — Январь/февраль.

36. Yttria-stabilized zirconia cross-point memristive devices for neuromorphic applications / A.V. Emelyanov, K.E. Nikiruy, V.A. Demin [etc.] // Microelectronic Engineering. — 2019. — Vol. 215. — 15 July, 110988.

37. Multifilamentary Character of Anticorrelated Capacitive and Resistive Switching in Memristive Structures Based on (Co−Fe−B)x (LiNbO3)100−x Nanocomposit / M.N. Martyshov, A.V. Emelyanov, V.A. Demin [etc.] // Phys. Rev. — 2020. —September 8.

38. *Таненбаум, Э.* Архитектура компьютера / Э. Таненбаум. — 5-е изд. — Санкт-Петербург : Питер, 2009. — 844 с.

39. *Денисенко, В.В.* Компьютерное управление технологическим процессом, экспериментом, оборудованием / В.В. Денисенко. — Москва : Горячая линия — Телеком, 2013. — 606 с.

40. Микросхемы MRAM // CROCUS NANO ELECTRONICS : [website]. — URL: http://crocusnano.com/mikrosxemyi-pamyati-mram.

41. Intel 100MHz Pentium II processor/440BX AGPset Uni-Processor Customer Reference Schematics. April 09, 1998.

42. Intel 440BX AGPset: 82443BX Host Bridge/Controller. Order Number: 290633-001. April 1998.

43. PCI-to-ISA / IDE Xcelerator. Order Number: 290562-001. April 1997.

44. IT8673F. Advanced Input / Output (Advanced I/O). Preliminary Specification V0.5. Integrated Technology Express, Inc. 2002.

45. 82430FX PCIset Datasheet. 82437FX System Controller (TSC) and 82438FX Data Path Unit (TDP). Order Number: 290518-002. November 1996.

46. Intel 850 Chipset: 82850 Memory Controller Hub (MCH). Datasheet. Document Number: 290691-001. November 2000.

47. Intel 82801BA I/O Controller Hub 2 (ICH2). Datasheet. Order Number: 290687-001. June 2000.

48. Intel 875P Chipset Platform Design Guide. Document Number: 252527-003. June 2003.

49. Intel 955X Express Chipset. Datasheet. Document Number: 306828-001. April 2005.

50. Intel 965 Express Chipset Family. Datasheet. Document Number: 313053-002. July 2006.

51. 8th Gen Intel Core Processor Families. Datasheet, Volume 1 of 2. Document Number: 337344-001. August 2018.

52. What is a Language Processing Unit?: [website]. — URL: https://groq.com/blog/the-groq-lpu-explained.

53. NVIDIA Grace Hopper Superchip Architecture. Performance and Productivity for Strong-Scaling HPC and Giant AI Workloads : [website]. — URL: https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper.

54. NVIDIA Blackwell Architecture Technical Brief. NVIDIA Corporation. 2025.

55. *Токхайм, Р.* Микропроцессоры: Курс и упражнения / Р. Токхайм ; перевод с английского, под редакцией В.Н. Грасевича. — Москва : Энергоатомиздат, 1988. — 336с.

56. Pentium processor. Order Number 241997-010. June 1997.

57. Intel 64 and IA-32 Architectures Software Developer's Manual. Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4. Order Number: 325462-075US. June 2021.

58. *Болдырихин, О.В*. Архитектура ЭВМ : учебник / О.В. Болдырихин, В.А. Алексеев. — Москва : Ай Пи Ар Медиа, 2024. — 694 с. — ISBN 978-5-4497-3255-2. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/142627.html (дата обращения: 04.09.2024). — Режим доступа: для авторизир. пользователей.

59. ГОСТ Р 56938-2016. Защита информации. Защита информации при использовании технологий виртуализации. Общие положения.

60. ГОСТ Р 59163-2020. Информационные технологии. Методы и средства обеспечения безопасности. Руководство по обеспечению безопасности при внедрении серверов виртуализации.

61. PCI Express Base Specification. Revision 3.0. PCI-SIG. November 10, 2010.

62. Universal Serial Bus Specification. Revision 2.0. April 27, 2000.

63. Universal Serial Bus 3.0 Specification. Revision 1.0. November 12, 2008.

64. Serial ATA: High Speed Serialized AT Attachment. Revision 1.0. 29-August-2001.

65. Serial ATA International Organization: Serial ATA Revision 3.1. 18-July-2011. Gold Revision.

66. Serial ATA technology. Technology brief, 4th edition. Hewlett-Packard Development Company. TC1108815, October 2011.

67. NVM Express Base Specification, Revision 2.0d, January 11th, 2024.

68. The Ethernet. A Local Area Network. Data Link Layer and Physical Layer Specifications. Digital Equipment Corporation, Intel Corporation, Xerox Corporation. Version 2.0. November, 1982.

69. NVIDIA ConnectX-6 Ethernet Adapter Cards User Manual. NVIDIA Corporation, 2025.

70. Novatek NT6880 Keyboard Controller. Datasheet, V1.0.

71. W9970CF Video Graphics Controller. Winbond Electronics Corp. March 1996.

72. SAA7135. Single-chip, analog TV video/stereo decoder for worldwide PCI applications with Dolby Pro Logic. Philips Semiconductors. December 2002.

73. NVIDIA Turing GPU Architecture. WP-09183-001_v01. NVIDIA Corporation, 2018.

74. NVIDIA A100 Tensor Core GPU Architecture. V1.0. NVIDIA Corporation, 2020.

75. Creative Introduces Sound Core3D — the High-Performance Multi-Core Sound and Voice Processor Designed for Embedded PC Applications: [website]. — URL: https://sg.creative.com/corporate/pressroom?id=13233.

76. Audio Codec '97. AC '97 Component Specification. Revision 2.1. May 22, 1998.

77. High Definition Audio Specification. Revision 1.0a. June 17, 2010.

78. IDT STAC9752A/9753A. AC '97 2.3 codecs with stereo microphone & universal jack. Datasheet. V 1.5 1206.

79. AD1888. AC '97 SoundMAX Codec. Analog Devices. 2003.

80. VT1708. High Definition Audio Codec. Datasheet. VIA Technologies, Inc. January 10, 2006.

81. ALC892. 7.1+2 channel HD Audio codec with content protection. Datasheet. Realtek Semiconductor Corp. 08 March 2011.

82. CS42L42. Low-Power Audio Codec with SoundWire–I2S/TDM and Audio Processing. Cirrus Logic. DS1083F708. JUN 2025.

83. Intel Xeon Phi Processor x200 Product Family Datasheet, Volume One: Electrical. Volume 1 of 2. Rev. 001. August. Order No.: 334710-001.

84. Frontier // Oak Ridge National Laboratory : [website]. — URL: https://www.olcf.ornl.gov/frontier/.

85. Early Experience in Supporting OpenSHMEM on HPE Slingshot NIC (Slingshot 11). Naveen Namashivayam. Bob Cernohous. Nathan Wichmann. — URL: https://cug.org/proceedings/cug2022_proceedings/includes/files/pap111s2-file1.pdf.

86. The Sunway TaihuLight supercomputer: system and applications // Science China. Information Sciences. — 2016. — July. — Vol. 59.

87. Report on the Sunway TaihuLight System. Jack Dongarra. University of Tennessee. Oak Ridge National Laboratory. June 24, 2016. University of Tennessee. Department of Electrical Engineering and Computer Science. Tech Report UT-EECS-16-742.