А. П. Жмакин

Архитектура ЭВМ

2-е издание

- Функциональная организация ЭВМ
- Машинная арифметика и синтез устройств
- Программные модели АЛУ
- Архитектура микропроцессорных систем
- Программная модель учебной ЭВМ
- Лабораторный практикум и курсовое проектирование

+CD

bhv

Анатолий Жмакин

Архитектура ЭВМ

2-е издание

Ñàíêò- Ïåòåðáóðã

«ÁÕÂ- Ïåòåðáóðã»

2010

УДК 681.3(075.8)
ББК 32.973-02я73
Ж77

Жмакин А. П.

Ж77 Архитектура ЭВМ: 2-е изд., перераб. и доп.: учеб. пособие. — СПб.: БХВ-Петербург, 2010. — 352 с.: ил. + CD-ROM — (Учебная литература для вузов)

ISBN 978-5-9775-0550-5

Пособие объединяет в одном издании теоретическую часть одноименной дисциплины и лабораторный практикум. Рассмотрены базовые вопросы организации ЭВМ: функциональная организация ЭВМ, системы команд и командный цикл. Большое внимание уделено арифметическим основам ЭВМ, принципам построения различных устройств и их взаимодействию. Обсуждаются вопросы построения микропроцессорных систем. Во втором издании лабораторный практикум дополнен программными моделями арифметико-логического устройства, представленными, наряду с программной моделью ЭВМ, на прилагаемом компакт-диске. Кроме того, пособие содержит материалы для выполнения курсового проектирования.

Для студентов и преподавателей технических вузов

УДК 681.3(075.8)
ББК 32.973-02я73

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Фото	*Кирилла Сергеева*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 03.03.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 28,38.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0550-5

Îãëàâëåíèå

Предисловие

Эта книга создавалась как учебное пособие по архитектуре процессоров и ЭВМ. Материал книги ориентирован на студентов инженерных специальностей, обучающихся в области разработки программного обеспечения и информационных систем, для которых "компьютерное железо" не является основным предметом изучения, но которые хотят (и должны) знать основы построения процессоров, организацию взаимодействия основных устройств ЭВМ, программирование на низком уровне. Книга может быть полезной и студентам педагогических специальностей, в учебных планах которых предусмотрены курсы по изучению архитектуры ЭВМ (физика, математика, информатика).

Читателям необходимо владеть начальными знаниями в области цифровой схемотехники (булева алгебра, логические элементы, триггеры, операционные элементы).

В основу книги положены материалы курсов лекций, читаемых автором на кафедре вычислительной техники Курского государственного технического университета (КГТУ) и кафедре программного обеспечения и администрирования информационных систем КГУ. Книга объединяет в себе теоретический материал, цикл лабораторных работ и задания на курсовое проектирование.

Пособие состоит из трех частей.

В *части I* рассматриваются общие принципы организации ЭВМ, включая их функциональную и структурную организацию. Достаточно подробно рассмотрены арифметические основы ЭВМ, представление чисел в различных кодах и алгоритмы выполнения арифметических операций. Уделено внимание кодированию десятичных чисел и десятичной машинной арифметике, а также системам счисления в остаточных классах. Далее рассматриваются принципы построения устройств ЭВМ — концепция операционного и управляющего автоматов, подходы к их синтезу. Рассмотрены управляющие автоматы с жесткой и программируемой логикой. Отдельная глава посвящена

организации многоуровневой памяти ЭВМ, вопросам взаимодействия устройств памяти разных уровней.

Часть II посвящена обсуждению базовой архитектуры систем на основе микропроцессоров *x*86. Кроме внутренней структуры микропроцессора и его интерфейса рассматривается организация ввода/вывода, прерываний, прямого доступа в память. Кратко рассмотрена эволюция архитектуры процессоров семейства *x*86.

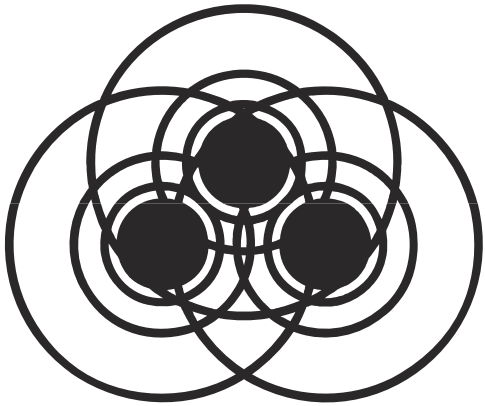
Часть III содержит лабораторный практикум и курсовое проектирование. Лабораторный практикум включает в себя большинство тем *части I*. Для выполнения лабораторных работ предлагается ряд программных моделей, размещенных на компакт-диске. Программные модели арифметико-логических устройств позволяют разрабатывать и отлаживать алгоритмы арифметических и логических преобразований, знакомиться со структурами операционных автоматов, проектировать управляющие автоматы по заданным (разработанным) микропрограммам. На базе программной модели учебной ЭВМ можно познакомиться со структурой ЭВМ, системой команд, программированием на уровне ассемблера, изучить принципы взаимодействия процессора с внешними устройствами, организацию многоуровневой памяти ЭВМ. Для каждой из десяти лабораторных работы сформулирована цель, индивидуальные задания, требования к оформлению отчета и контрольные вопросы; для некоторых заданий приведены примеры выполнения.

Содержанием курсовой работы является разработка арифметико-логического устройства, реализующего заданный набор операций с учетом ограничений на код выполнения операций и способ построения управляющего автомата. Сформулированы индивидуальные задания, определены этапы выполнения работы и содержание пояснительной записки.

В работе над книгой принимал участие канд. тех. наук, доцент А. М. Фрумкин, совместно с которым написаны *разд. 8.1*, *10.1* и *10.2*.

Автор благодарит студентов и выпускников факультета информатики и вычислительной техники КГУ В. Кузьминова, А. Ильина, А. Алферова, руководителя отдела архитектуры корпорации Skyros И. Жмакина, принимавших активное участие в разработке и отладке приведенных в книге программных моделей АЛУ и ЭВМ.

Автор выражает искреннюю признательность заведующему кафедрой системного программирования СпбГУ, профессору, доктору физ.-мат. наук А. Н. Терехову и доценту, канд. физ.-мат. наук В. А. Костину за ценные замечания, сделанные при рецензировании книги.



ЧАСТЬ I

Принципы организации ЭВМ

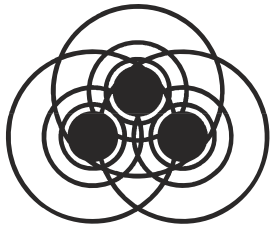
Принято считать, что ЭВМ — "сложная система". Это понятие имеет много трактовок, в т. ч. и такую: *"сложную систему невозможно адекватно описать на одном языке"*. Обычно ЭВМ рассматривают на нескольких уровнях:

- ❒ логические элементы;
- ❒ операционные элементы (узлы);
- ❒ устройства;
- ❒ структура ЭВМ и система команд.

На каждом из уровней используются свои языки описания. И "выше", и "ниже" приведенных элементов списка можно выделить другие уровни, но их рассмотрение лежит за пределами этой книги.

Приступая к изучению вопросов архитектуры ЭВМ, читатель должен иметь представление о логических и операционных элементах цифровой техники (конъюнкторы, инверторы, ..., триггеры, регистры, мультиплексоры, дешифраторы, сумматоры и т. д.).

Центральным в структуре ЭВМ является, несомненно, процессор, а главными устройствами любого процессора можно считать *арифметико-логическое устройство* (АЛУ) и *устройство управления* (УУ). Далее мы подробно рассмотрим принципы и способы организации АЛУ. Поскольку АЛУ разрабатывается для реализации определенных алгоритмов арифметической и логической обработки данных, то неизбежным становится и рассмотрение различных вариантов таких алгоритмов.



Глава 1

Начальные сведения об ЭВМ

1.1. История развития вычислительной техники

С тех пор как человечество осознало понятие количества, разрабатывались и применялись различные приспособления для отображения количественных эквивалентов и операций над величинами. Отбросив рассмотрение "доисторических" с точки зрения вычислительной техники средств (кучки камней, счеты и т. д.), рассмотрим кратко историю развития вычислительных машин.

Пожалуй, первой реально созданной машиной для выполнения арифметических действий в десятичной системе счисления можно считать *счетную машину Паскаля*. В 1642 г. Б. Паскаль продемонстрировал ее работу. Машина выполняла суммирование чисел (восьмиразрядных) с помощью колес, которые при добавлении единицы поворачивались на 36° и приводили в движение следующее по старшинству колесо всякий раз, когда цифра 9 должна была перейти в значение 10. Машина Паскаля получила известность во многих странах, было изготовлено более 50 экземпляров машины.

Впрочем, еще до Паскаля машину, механически выполняющую арифметические операции, изобразил в эскизах Леонардо да Винчи (1452—1519). Суммирующая машина по его эскизам выполнена в наши дни и доказала свою работоспособность.

В средние века (расцвет механики) было предложено и выполнено много различных вариантов арифметических машин: Морлэнд (1625—1695), К. Перро (1613—1688), Якобсон, Чебышев и др. Первую машину, с помощью которой можно было не только складывать, но и умножать и делить, разработал Г. Лейбниц (1646—1716). Однако большинство подобных машин изготавливались авторами в единичных экземплярах. Удачное решение инженера В. Однера, разработавшего колесо с переменным числом зубьев, позволило

почти век серийно выпускать арифмометры (например, "Феликс" Курского завода "Счетмаш"), являвшиеся основным средством вычислений вплоть до эпохи ПЭВМ и калькуляторов.

Все упомянутые выше механизмы обладали одной особенностью — могли автоматически выполнять только отдельные действия над числами, но не могли хранить промежуточные результаты и, следовательно, выполнять последовательность действий.

Первой вычислительной машиной, реализующей автоматическое выполнение последовательности действий, можно считать *разностную машину Ч. Беббеджа* (1792—1871). В 1819 г. он изготовил ее для расчета астрономических и морских таблиц. Машина обеспечивала хранение необходимых промежуточных значений и выполнение последовательности сложений для получения значения функции. В дальнейшем Беббедж предложил так называемую *аналитическую машину*, предназначенную для решения любых вычислительных задач. При желании в аналитической машине Беббеджа можно найти прообразы всех основных устройств современной ЭВМ: арифметическое устройство ("мельница"), память ("склад"), устройство управления (на перфокартах), позволяющее выбирать различные пути решения в зависимости от значений исходных данных и промежуточных результатов. Проект аналитической машины Беббеджа так и не был реализован — из-за несоответствия идеи и элементной базы.

Даже выпускаемые большими сериями электрические *релейные машины Холлерита* (1860—1929) — *табуляторы* — не произвели переворота в средствах обработки информации, хотя и широко использовались для обработки статистической информации вплоть до 70-х годов прошлого века.

Идеи аналитической машины Беббеджа были использованы в релейных машинах, выпускавшихся в 30—40-х годах XX века. Теоретической основой разработки релейно-контактных схем явился аппарат булевой алгебры, который в дальнейшем использовался для синтеза схем ЭВМ. Однако и электрические реле как элементная база вычислительной техники не удовлетворяли потребностям этой техники по всем основным параметрам (быстродействие, надежность, потребляемая мощность, стоимость, габариты и др.).

Только освоение электронных схем в качестве элементной базы положило начало действительно массовому внедрению сначала вычислительной, а потом и информационной техники во все сферы человеческой деятельности. Первые электронные цифровые вычислительные машины (ЭВМ) были разработаны и выпущены на рубеже 40—50-х годов прошлого века в США, Англии и чуть позднее — в СССР.

1.2. Цифровые и аналоговые вычислительные машины

Все приведенные ранее факты относятся к истории так называемой *цифровой* вычислительной техники, в которой информация представлена в дискретной форме (в форме чисел, кодов, знаков). Однако большинство физических величин может принимать значение из непрерывного множества — континуума. Существуют вычислительные устройства, оперирующие непрерывной информацией (пример — логарифмическая линейка, где информация представлена отрезками длины). Существует и целый класс электронных вычислительных машин — так называемые *аналоговые*, информация в которых представляется непрерывными значениями электрического напряжения или тока. Принцип работы таких машин — в построении электрических цепей, процессы в которых описываются теми дифференциальными уравнениями, которые требуется решить.

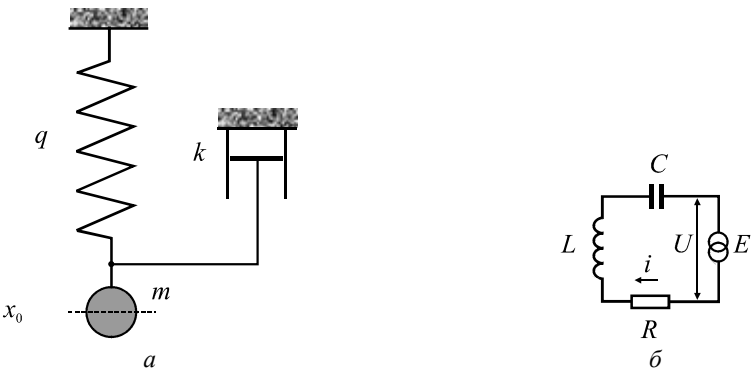


Рис. 1.1. Модель: *а* — механическая система; *б* — "аналогичная" ей электрическая цепь

Классический пример такого подхода показан на рис. 1.1. Например, требуется изучить поведение механической колебательной системы, описываемой дифференциальным уравнением (1.1). Подберем электрическую цепь, процессы в которой описываются тем же дифференциальным уравнением с точностью до обозначений (1.2). Между механическими величинами (рис. 1.1, *а*) и электрическими (рис. 1.1, *б*) существует соответствие (сравните уравнения (1.1) и (1.2)).

$$m\frac{dx}{dt} = mg - \int_0^t x \cdot dt - kx\,. \qquad (1.1)$$

$$L\frac{di}{dt} = U - \int_0^t i \cdot dt - Ri\,. \qquad (1.2)$$

Таким образом, для механического устройства можно подобрать электрическую цепь, процессы в которой описываются аналогичными дифференциальными уравнениями. Или, для заданного дифференциального уравнения (системы) построить электрическую цепь, которая описывается этим уравнением.

Существуют хорошо отработанная методика синтеза таких цепей и наборы функциональных блоков (АВМ), позволяющие собирать и исследовать синтезированные цепи.

Достоинства АВМ: простота подготовки решения, высокая скорость решения.

Недостатки АВМ: неуниверсальность (предназначены только для решения дифференциальных уравнений) и низкая точность решения.

В настоящее время АВМ находят применение лишь в ограниченных областях технического моделирования. Поэтому в дальнейшем будем употреблять термин "ЭВМ", имея в виду только цифровые вычислительные машины, как это принято в современной терминологии.

1.3. Варианты классификации ЭВМ

За свою полувековую историю ЭВМ из единичных экземпляров инструментов ученых превратились в предмет массового потребления. Спектр применения ЭВМ в современном обществе чрезвычайно широк, причем именно область применения накладывает основной отпечаток на характеристики ЭВМ. Поэтому в большинстве подходов к классификации ЭВМ именно область применения является основным параметром классификации.

Изделия современной техники, особенно вычислительной, традиционно принято делить на поколения (табл. 1.1), причем основным признаком поколения ЭВМ считается ее элементная база. Следует помнить, что любая классификация не является абсолютной. Всегда можно отыскать объект классификации, который по одним параметрам относится к одному классу, а по другим — к другому. Это в большой степени относится и к классификации поколений ЭВМ: некоторые авторы выделяют три поколения ЭВМ (дальнейшее развитие ЭВМ идет как бы вне поколений), другие насчитывают целых шесть.

В рамках *первого поколения* ЭВМ не возникала необходимость в классификации, т. к. машин были считанные единицы и использовались они, как правило, для выполнения научно-технических расчетов. Отдельные машины характеризовались быстродействием (числом выполняемых операций в секунду), объемом памяти, стоимостью, надежностью (наработка на отказ), габаритно-весовыми характеристиками, потребляемой мощностью и другими параметрами.

Таблица 1.1. *Поколения ЭВМ*

Поколение	**Элементная база**	**Годы существования**	**Области применения**
Первое	Электронные лампы	50—60	Научно-технические расчеты
Второе	Транзисторы, ферритовые сердечники	60—70	Научно-технические расчеты, планово-экономические расчеты
Третье	Интегральные схемы	70—80	Научно-технические расчеты, планово-экономические расчеты, системы управления
Четвертое	СИС, БИС, СБИС и т. д.	80 и по сей день	Все сферы деятельности

Использование транзисторов в качестве элементной базы *второго поколения* привело к улучшению примерно на порядок каждого из основных параметров ЭВМ. Это, в свою очередь, резко расширило сферу применения ЭВМ, причем в разных областях применения к ЭВМ предъявлялись различные требования. Так называемые "научно-технические расчеты" характеризовались относительно небольшим объемом входной и выходной информации, но очень большим числом сложных операций с высокой точностью над входной информацией, а "планово-экономические расчеты"[1] — наоборот, простейшими операциями (сложение, сравнение) над огромными объемами информации.

Соответственно в рамках второго поколения ЭВМ выделялись:

- ЭВМ для *научно-технических расчетов*, характеризующиеся мощным быстродействующим процессором с развитой системой команд (в т. ч. реализующей арифметику с плавающей запятой) и относительно небольшой внешней памятью и номенклатурой устройств ввода/вывода;
- ЭВМ для *планово-экономических расчетов*, характеризующиеся, прежде всего, большой многоуровневой памятью, развитой номенклатурой устройств ввода/вывода (УВВ), но относительно простым и дешевым процессором, система команд которого включает простые арифметические команды (сложение, вычитание) с фиксированной запятой.

Характерно, что и языки программирования "второго поколения" так же разделялись на "математические" (FORTRAN) и "экономические" (COBOL).

[1] Здесь используется терминология, принятая в годы существования второго поколения ЭВМ.

Однако по мере расширения сферы применения ЭВМ, улучшения их основных характеристик, появления новых задач, границы между выделенными классами стали размываться. Уже в рамках второго поколения стали выделять так называемые ЭВМ *общего назначения*, одинаково хорошо приспособленные для решения разнообразных задач. Такие машины объединяли в себе достоинства "научно-технических" и "планово-экономических" ЭВМ: мощный процессор, большую память, широкую номенклатуру УВВ (в то время это уже можно было себе позволить). Такие машины могли решать задачи, недоступные предыдущим моделям. Но для решения более простых задач их ресурсы являлись избыточными и, следовательно, решение этих задач экономически не оправдано. Поэтому ЭВМ общего назначения (универсальные ЭВМ) стали выпускать различной вычислительной мощности (и, следовательно, стоимости): *большие*, *средние* и *малые*.

В рамках ЭВМ *третьего поколения* стал усиленно развиваться новый класс — *управляющие ЭВМ*. К ЭВМ, работающим в контуре управления объектом или технологическим процессом, предъявляются специфические требования: прежде всего, высокая надежность, способность работать в экстремальных внешних условиях (перепады температуры, давления, питающих напряжений, высокий уровень электромагнитных помех и т. п.), быстрая реакция на изменения состояния внешней среды, малые габариты и вес, простота обслуживания. В то же время к таким характеристикам, как быстродействие процессора, мощность системы команд, объем памяти, часто не предъявлялись слишком высоких требований, зато решающим становился фактор стоимости. Эти особенности привели к появлению класса так называемых *мини-ЭВМ*, а затем и *микроЭВМ*, хотя в дальнейшем и мини- и микроЭВМ использовались не только в качестве управляющих. Иногда эти классы объединяли понятием *проблемно-ориентированные ЭВМ*.

Наряду с упомянутыми классами ЭВМ широкого применения всегда выпускались машины, которые можно было считать *специализированными*. Это, во-первых, так называемые *суперЭВМ*, выпускаемые в единичных экземплярах и предназначенные для решения задач, недоступных для серийной вычислительной техники. Для ряда применений создавались специализированные ЭВМ, архитектура и структура которых оптимизировалась под решение конкретной задачи. Ту же задачу можно было решить и на универсальной ЭВМ подходящего класса, но со значительно более низкими показателями качества. В то же время решение других задач на специализированной ЭВМ было либо невозможно, либо крайне неэффективно. Одна из возможных классификаций ЭВМ на рубеже 3—4 поколений показана на рис. 1.2.

Еще одним важным явлением, проявившимся при развитии третьего поколения ЭВМ, стало появление *семейств ЭВМ*. В рамках одного семейства, объе-

диненного общими архитектурными, структурными, а иногда — и конструктивными решениями, выпускались несколько (иногда — более десятка) классов ЭВМ: малые, средние, "полусредние", большие, очень большие и т. д.

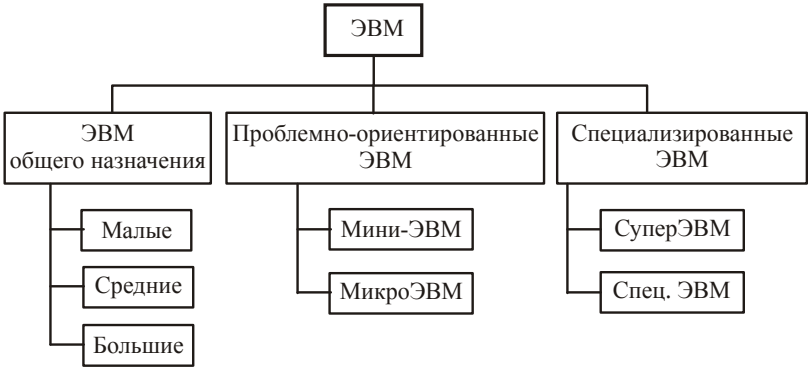


Рис. 1.2. Вариант классификации ЭВМ

Общими для большинства семейств являются:

- ❒ внутренний язык, что позволяет осуществлять совместимость программ на уровне машинных кодов (IBM-360, ЕС ЭВМ) либо системы команд, обладающие совместимостью "снизу вверх" (PDP-11), когда старшие представители семейства реализуют все команды младших моделей плюс еще некоторые команды;
- ❒ форматы данных;
- ❒ форматы записи на внешний носитель;
- ❒ интерфейс, что позволяет иметь единую номенклатуру внешних устройств для всех представителей семейства;
- ❒ преемственность программного обеспечения (как правило, та же совместимость "снизу вверх").

Для решения конкретной задачи пользователь подбирал соответствующий экземпляр семейства, а по мере усложнения задачи осуществлялся переход на более старшие модели семейства, причем уже отлаженные на младших моделях программы, как правило, не требовали доработки.

Наиболее известными примерами семейств ЭВМ могут служить:

- ❒ семейство универсальных ЭВМ третьего поколения IBM-360 и его советский аналог — ЕС ЭВМ, включающее малые машины ЕС-1010 и ЕС-1020, средние ЕС-1022, ЕС-1030, ЕС-1035 и др., большие ЕС-1050, ЕС-1060, ЕС-1065;

- семейство мини-ЭВМ PDP-11 и его советский аналог — СМ ЭВМ (лишь часть представителей семейства — СМ-3, СМ-4, СМ-1420);
- семейство микроЭВМ LXI-11 (Электроника-60 и ее модификации);
- семейство микропроцессоров i80*x*86.

1.4. Классическая архитектура ЭВМ

Считается, что основные идеи построения современных ЭВМ в 1945 г. сформулировал американский математик Дж. фон Нейман, определив их как *принципы программного управления*:

1. Информация кодируется в двоичной форме и разделяется на единицы — слова.
2. Разнотипные по смыслу слова различаются по способу использования, но не по способу кодирования.
3. Слова информации размещаются в ячейках памяти и идентифицируются номерами ячеек — адресами слов.
4. Алгоритм представляется в форме последовательности управляющих слов, называемых *командами*. Команда определяет наименование операции и слова информации, участвующие в ней. Алгоритм, записанный в виде последовательности команд, называется *программой*.
5. Выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определенном программой.

Поэтому классическую архитектуру современных ЭВМ, представленную на рис. 1.3, часто называют "архитектурой фон Неймана".

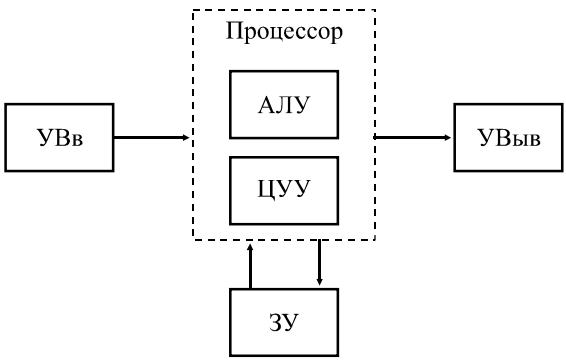


Рис. 1.3. Классическая архитектура ЭВМ

Программа вычислений (обработки информации) составляется в виде последовательности команд и загружается в память машины — *запоминающее устройство* (ЗУ). Там же хранятся исходные данные и промежуточные результаты обработки. *Центральное устройство управления* (ЦУУ) последовательно извлекает из памяти команды программы и организует их выполнение. *Арифметико-логическое устройство* (АЛУ) предназначено для реализации операций преобразования информации. Программа и исходные данные вводятся в память машины через *устройства ввода* (УВв), а результаты обработки предъявляются на *устройства вывода* (УВыв).

Характерной особенностью архитектуры фон Неймана является то, что память представляет собой единое адресное пространство, предназначенное для хранения как программ, так и данных.

Такой подход, с одной стороны, обеспечивает большую гибкость организации вычислений — возможность перераспределения памяти между программой и данными, возможность самомодификации программы в процессе ее выполнения. С другой стороны, без принятия специальных мер защиты снижается надежность выполнения программы, что особенно недопустимо в управляющих системах.

Действительно, поскольку и команды программы, и данные кодируются в ЭВМ двоичными числами, теоретически возможно как разрушение программы (при обращении в область программы как к данным), так и попытка "выполнения" области данных как программы (при ошибочных переходах программы в область данных).

Альтернативной фон-неймановской является так называемая *гарвардская архитектура*. ЭВМ, реализованные по этому принципу, имеют два непересекающихся адресных пространства — для программы и для данных, причем программу нельзя разместить в свободной области памяти данных и наоборот. Гарвардская архитектура применяется главным образом в управляющих ЭВМ.

1.5. Иерархическое описание ЭВМ

ЭВМ как сложная система может быть адекватно описана на нескольких уровнях с применением различных языков описания на каждом из уровней.

Принципы структурного описания предполагают введение следующих понятий:

- *система* — совокупность элементов, объединенных в одно целое для достижения определенных целей. Для полного описания системы следует определить ее функции и структуру;

- *структура системы* — фиксированная совокупность элементов системы и связей между ними;
- *элемент* — неделимая часть системы, структура которого не рассматривается, а определяются только его функции.

Функции системы стремятся описывать в математической форме, иногда — в словесной (содержательной форме). Структура системы может быть задана в виде графа или эквивалентных ему математических форм (матриц). Инженерной формой задания структуры является схема (отличается от графа только формой). Различным уровням представления систем соответствуют различные виды схем.

Свойства системы не являются простой суммой свойств входящих в нее элементов; за счет организации связей между элементами приобретается новое качество, отсутствующее в элементах. Например, радиокомпоненты → логические элементы → сумматор.

Для сложных систем характерно, что функция, реализуемая системой, не может быть представлена как композиция функций, реализуемых наименьшими элементами системы (иначе говоря, функцию сложной системы нельзя адекватно описать на одном языке). Действительно, функционирование ЭВМ нельзя описать лишь на языке электрических процессов, в ней происходящих. Функции ЭВМ как системы выявляются лишь при рассмотрении информационных и логических аспектов ее работы.

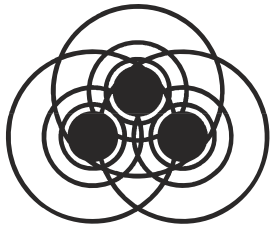
Поэтому в описании сложных систем используют несколько форм описания (языков) функций и структуры — иерархию функций и структуры. Иерархический подход к описанию сложных систем предполагает, что на высшем уровне иерархии система рассматривается как один элемент, имеющий входы и выходы для связи с внешней средой. В этом случае функция не может быть задана подробно и представляется как отображение состояний входов на состояние выходов системы.

Чтобы раскрыть устройство и порядок функционирования системы, глобальная функция и сама система разделяются на части — функции и структурные элементы следующего более низкого уровня иерархии и т. д. до тех пор, пока функции и структура системы не будут раскрыты полностью, с необходимой степенью детализации.

В этом случае элемент — это, прежде всего, удобное понятие, а не физическое свойство, т. к. один и тот же физический объект может рассматриваться как элемент на одном уровне иерархии и как система — на другом (более низком) уровне. В табл. 1.2 представлены основные уровни ЭВМ и языки описания этих уровней.

Таблица 1.2. *Уровни описания ЭВМ*

Уровень описания	**Объект**	**Структурный базис**	**Язык описания**
Электрические схемы	Логические и запоминающие элементы	Электронные и радиокомпоненты — транзисторы, резисторы и др.	Соотношения теории электрических цепей
Логические схемы	Операционные элементы (счетчики, сумматоры, дешифраторы, регистры и т. д.), микропрограммные автоматы	Логические и запоминающие элементы	Булева алгебра, теория конечных автоматов
Операционные схемы	Операционные устройства: (арифметико-логическое устройство, устройство управления, запоминающее устройство и др.)	Операционные элементы, микропрограммные автоматы	Языки описания микроопераций
Структурные схемы	ЭВМ и системы	Операционные устройства	Языки машинных команд, микропрограмм
Программный уровень	Операционные системы, вычислительный процесс	Команды и операторы	Алгоритмические языки



Глава 2

Функциональная организация ЭВМ

Термин *"функциональная организация ЭВМ"* часто используют в качестве синонима (в некотором смысле) более широкого термина — *"архитектура ЭВМ"*, который, в свою очередь, трактуется разными авторами несколько в различных смыслах. Наиболее близким к трактовке автора может служить определение термина "архитектура ЭВМ", данное в [7]. Приведем это определение.

Архитектура ЭВМ — это абстрактное представление ЭВМ, которое отражает ее структурную, схемотехническую и логическую организацию. Понятие архитектуры ЭВМ является комплексным и включает в себя:

- структурную схему ЭВМ;
- средства и способы доступа к элементам структурной схемы;
- организацию и разрядность интерфейсов ЭВМ;
- набор и доступность регистров;
- организацию и способы адресации памяти;
- способы представления и форматы данных ЭВМ;
- набор машинных команд ЭВМ;
- форматы машинных команд;
- обработку нештатных ситуаций (прерываний).

В рамках данной книги мы, в основном, будем рассматривать перечисленные выше вопросы.

2.1. Командный цикл процессора

Командой называется элементарное действие, которое может выполнить процессор без дальнейшей детализации. Последовательность команд, выполнение которых приводит к достижению определенной цели, называется *программой*.

Команды программы кодируются двоичными словами и размещаются в памяти ЭВМ. Вся работа ЭВМ состоит в последовательном выполнении команд программы. Действия по выбору из памяти и выполнению одной команды называются *командным циклом*.

В составе любого процессора имеется специальная ячейка, которая хранит адрес выполняемой команды — *счетчик команд* или *программный счетчик*. После выполнения очередной команды его значение увеличивается на единицу (если код одной команды занимает несколько ячеек памяти, то содержимое счетчика команд увеличивается на длину команды). Таким образом осуществляется выполнение последовательности команд. Существуют специальные команды (передачи управления), которые в процессе своего выполнения модифицируют содержимое программного счетчика, обеспечивая переходы по программе. Сама выполняемая команда помещается в *регистр команд* — специальную ячейку процессора.

Во время выполнения командного цикла процессор реализует такую последовательность действий:

1. Извлечение из памяти содержимого ячейки, адрес которой хранится в программном счетчике, и размещение этого кода в регистре команд (чтение команды).
2. Увеличение содержимого программного счетчика на единицу.
3. Формирование адреса операндов.
4. Извлечение операндов из памяти.
5. Выполнение заданной в команде операции.
6. Размещение результата операции в памяти.
7. Переход к п. 1.

Пункты 1, 2 и 7 обязательно выполняются в каждом командном цикле, остальные могут не выполняться в некоторых командах. Если длина кода команды составляет несколько машинных слов, то пп. 1 и 2 повторяются.

Фактически вся работа процессора заключается в циклическом выполнении пунктов 1—7 командного цикла. При запуске машины в счетчик команд аппаратно помещается фиксированное значение — начальный адрес программы (часто 0 или последний адрес памяти; встречаются и более экзотические способы загрузки начального адреса). В дальнейшем содержимое программного счетчика модифицируется в командном цикле. Прекращение выполнения командных циклов может произойти только при выполнении специальной команды "СТОП".

2.2. Система команд процессора

Разнообразие типов данных, форм их представления и действий, которые необходимы для обработки информации и управления ходом вычислений, порождает необходимость использования различных команд — набора команд. Каждый процессор имеет собственный вполне определенный набор команд, называемый *системой команд процессора*. Система команд должна обладать двумя свойствами — *функциональной полнотой* и *эффективностью*.

Функциональная полнота — это достаточность системы команд для описания любого алгоритма. Требование функциональной полноты не является слишком жестким. Доказано, что свойством функциональной полноты обладает система, включающая всего *три* команды (система Поста): *присвоение 0*, *присвоение 1*, *проверка на 0*. Однако составление программ в такой системе команд крайне неэффективно.

Эффективность системы команд — степень соответствия системы команд назначению ЭВМ, т. е. классу алгоритмов, для выполнения которых предназначается ЭВМ, а также требованиям к производительности ЭВМ. Очевидно, что реализация развитой системы команд связана с большими затратами оборудования и, следовательно, с высокой стоимостью процессора. В то же время ограниченный набор команд приводит к снижению производительности и повышенным требованиям к памяти для размещения программы. Даже простые и дешевые современные микропроцессоры поддерживают систему команд, содержащую несколько десятков (а с модификациями — сотен) команд.

Система команд процессора характеризуется тремя аспектами: форматами, способами адресации и системой операций.

2.2.1. Форматы команд

Под *форматом команды* следует понимать длину команды, количество, размер, положение, назначение и способ кодировки ее полей.

Команды, как и любая информация в ЭВМ, кодируются двоичными словами, которые должны содержать в себе следующие виды информации:

- тип операции, которую следует реализовать в данной команде (КОП);
- место в памяти, откуда следует взять первый операнд (А1);
- место в памяти, откуда следует взять второй операнд (А2);
- место в памяти, куда следует поместить результат (А3).

Каждому из этих видов информации соответствует своя часть двоичного слова — поле, а совокупность полей (их длины, расположение в командном слове, способ кодирования информации) называется форматом команды. В свою очередь, некоторые поля команды могут делиться на подполя. Формат команды, поля которого перечислены выше, называется *трехадресным* (рис. 2.1, *а*).

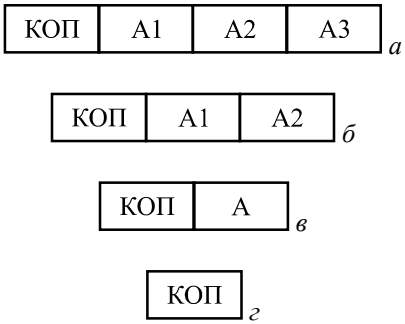


Рис. 2.1. Форматы команд: *а* — трехадресный; *б* — двухадресный; *в* — одноадресный; *г* — безадресный

Команды трехадресного формата занимают много места в памяти, в то же время далеко не всегда поля адресов используются в командах эффективно. Действительно, наряду с двухместными операциями (сложение, деление, конъюнкция и др.) встречаются и одноместные (инверсия, сдвиг, инкремент и др.), для которых третий адрес не нужен. При выполнении цепочки вычислений часто результат предыдущей операции используется в качестве операнда для следующей. Более того, нередко встречаются команды, для которых операнды не определены (СТОП) или подразумеваются самим кодом операций (DAA, десятичная коррекция аккумулятора).

Поэтому в системах команд реальных ЭВМ трехадресные команды встречаются редко. Чаще используются *двухадресные команды* (рис. 2.1, *б*), в этом случае в бинарных операциях результат помещается на место одного из операндов.

Для реализации одноадресных форматов (рис. 2.1, *в*) в процессоре предусматривают специальную ячейку — *аккумулятор*. Первый операнд и результат всегда размещаются в аккумуляторе, а второй операнд адресуется полем А.

Реальная система команд обычно имеет команды нескольких форматов, причем тип формата определяется в поле КОП.

2.2.2. Способы адресации

Способ адресации определяет, каким образом следует использовать информацию, размещенную в поле адреса команды.

Не следует думать, что во всех случаях в поле адреса команды помещается адрес операнда. Существует пять основных способов адресации операндов в командах.

- *Прямая* — в этом случае в адресном поле располагается адрес операнда. Разновидность — *прямая регистровая* адресация, адресующая не ячейку памяти, а РОН. Поле адреса регистра имеет в команде значительно меньшую длину, чем поле адреса памяти.
- *Непосредственная* — в поле адреса команды располагается не адрес операнда, а сам операнд. Такой способ удобно использовать в командах с константами.
- *Косвенная* — в поле адреса команды располагается адрес ячейки памяти, в которой хранится адрес операнда ("адрес адреса"). Такой способ позволяет оперировать адресами как данными, что облегчает организацию циклов, обработку массивов данных и др. Его основной недостаток — потеря времени на двойное обращение к памяти — сначала за адресом, потом — за операндом. Разновидность — *косвенно-регистровая* адресация, при которой в поле команды размещается адрес РОН, хранящего адрес операнда. Этот способ, помимо преимущества обычной косвенной адресации, позволяет обращаться к большой памяти с помощью коротких команд и не требует двойного обращения к памяти (обращение к регистру занимает гораздо меньше времени, чем к памяти).
- *Относительная* — адрес формируется как сумма двух слагаемых: базы, хранящейся в специальном регистре или в одном из РОН, и смещения, извлекаемого из поля адреса команды. Этот способ позволяет сократить длину команды (смещение может быть укороченным, правда в этом случае не вся память доступна в команде) и/или перемещать адресуемые массивы информации по памяти (изменяя базу). Разновидности — *индексная* и *базово-индексная адресации*. Индексная адресация предполагает наличие индексного регистра вместо базового. При каждом обращении содержимое индексного регистра автоматически модифицируется (обычно увеличивается или уменьшается на 1). Базово-индексная адресация формирует адрес операнда как сумму трех слагаемых: базы, индекса и смещения.
- *Безадресная* — поле адреса в команде отсутствует, а адрес операнда или не имеет смысла для данной команды, или подразумевается по умолча-

нию. Часто безадресные команды подразумевают действия над содержимым аккумулятора. Характерно, что безадресные команды нельзя применить к другим регистрам или ячейкам памяти.

Одной из разновидностей безадресного обращения является использование так называемой магазинной памяти или *стека*. Обращение к такой памяти напоминает обращение с магазином стрелкового оружия. Имеется фиксированная ячейка, называемая *верхушкой стека*. При чтении слово извлекается из верхушки, а все остальное содержимое "поднимается вверх" подобно патронам в магазине, так что в верхушке оказывается следующее по порядку слово. Одно слово нельзя прочитать из стека дважды. При записи новое слово помещается в верхушку стека, а все остальное содержимое "опускается вниз" на одну позицию. Таким образом, слово, помещенное в стек первым, будет прочитано последним. Говорят, что стек поддерживает дисциплину LIFO — Last In First Out (последний пришел — первый ушел). Реже используется безадресная память типа *очередь* с дисциплиной FIFO — First In First Out (первый пришел — первый ушел).

2.2.3. Система операций

Все операции, выполняемые в командах ЭВМ, принято делить на пять классов.

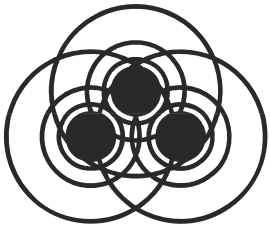
- *Арифметико-логические и специальные* — команды, в которых выполняется собственно преобразование информации. К ним относятся арифметические операции: сложение, вычитание, умножение и деление (с фиксированной и плавающей занятой), команды десятичной арифметики, логические операции конъюнкции, дизъюнкции, инверсии и др., сдвиги, преобразование чисел из одной системы счисления в другую и такие экзотические, как извлечение корня, решение системы уравнений и др. Конечно, очень редко встречаются ЭВМ, система команд которых включает все эти команды.
- *Пересылки* и *загрузки* — обеспечивают передачу информации между процессором и памятью или между различными уровнями памяти (СОЗУ ↔ ОЗУ). Разновидность — *загрузка регистров и ячеек константами*.
- *Ввода/вывода* — обеспечивают передачу информации между процессором и внешними устройствами. По структуре они очень похожи на команды предыдущего класса. В некоторых ЭВМ принципиально отсутствует различие между ячейками памяти и регистрами внешних устройств (единое адресное пространство) и класс команд ввода/вывода не выделяется, все обмены осуществляются в рамках команд пересылки и загрузки.

- *Передачи управления* — команды, которые изменяют естественный порядок выполнения команд программы. Эти команды меняют содержимое программного счетчика, обеспечивая переходы по программе. Существуют команды безусловной и условной передачи управления. В последнем случае передача управления происходит, если выполняется заданное в коде команды условие, иначе выполняется следующая по порядку команда. В качестве условий обычно используются признаки результата предыдущей операции, которые хранятся в специальном регистре признаков (флажков). Чаще всего формируются и проверяются признаки нулевого результата, отрицательного результата, наличия переноса из старшего разряда, четности числа единиц в результате и др. Различают три разновидности команд передачи управления:
  - переходы;
  - вызовы подпрограмм;
  - возвраты из подпрограмм.

  Команды *переходов* помещают в программный счетчик содержимое своего адресного поля — адрес перехода. При этом старое значение программного счетчика теряется. В микроЭВМ часто для экономии длины адресного поля команд условных переходов адрес перехода формируется как сумма текущего значения программного счетчика и относительно короткого знакового смещения, размещаемого в команде. В крайнем случае, в командах условных переходов можно и вовсе обойтись без адресной части — при выполнении условия команда "перепрыгивает" через следующую команду, которой обычно является безусловный переход.

  Команда *вызова* подпрограммы работает подобно команде безусловного перехода, но старое значение программного счетчика предварительно сохраняется в специальном регистре или в стеке. Команда возврата передает содержимое верхушки стека или специального регистра в программный счетчик. Команды *вызова* и *возврата* работают "в паре". Подпрограмма, вызываемая командой вызова, должна заканчиваться командой возврата, что обеспечивает по окончании работы подпрограммы передачу управления в точку вызова. Хранение адресов возврата в стеке обеспечивает возможность реализации вложенных подпрограмм.

- *Системные* — команды, выполняющие управление процессом обработки информации и внутренними ресурсами процессора. К таким командам относятся команды управления подсистемой прерывания, команды установки и изменения параметров защиты памяти, команда останова программы и некоторые другие. В простых процессорах класс системных команд немногочисленный, а в сложных мультипрограммных системах предусматривается большое число системных команд.



Глава 3

Арифметические основы ЭВМ

Безусловно, одним из основных направлений применения компьютеров были и остаются разнообразные вычисления. Обработка числовой информации ведется и при решении задач, на первый взгляд не связанных с какими-то расчетами, например, при использовании компьютерной графики или звука.

В связи с этим встает вопрос о выборе *оптимального представления чисел в компьютере*. Безусловно, можно было бы использовать 8-битное (байтовое) кодирование отдельных цифр, а из них составлять числа. Однако такое кодирование не будет оптимальным, что легко увидеть из простого примера. Пусть имеется двузначное число 13. При 8-битном кодировании отдельных цифр в кодах ASCII его представление выглядит следующим образом: 0011000100110011, т. е. код имеет длину 16 битов; если же определять это число просто в двоичном коде, то получим 4-битную цепочку 1101.

Важно, что представление определяет не только способ записи данных (букв или чисел), но и допустимый набор операций над ними; в частности, буквы могут быть только помещены в некоторую последовательность (или исключены из нее) без изменения их самих; над числами же возможны *операции*, изменяющие само число, например, извлечение корня или сложение с другим числом.

Представление чисел в компьютере имеет две особенности:

- ❒ числа записываются в двоичной системе счисления (в отличие от привычной десятичной);
- ❒ для записи и обработки чисел отводится конечное количество разрядов (в "некомпьютерной" арифметике такое ограничение отсутствует).

Следствия, к которым приводят эти отличия, и рассматриваются в данной главе.

3.1. Системы счисления

Начнем с некоторых общих замечаний относительно понятия *"число"* [10]. Можно считать, что любое число имеет значение (содержание) и форму представления.

Значение числа задает его отношение к значениям других чисел ("больше", "меньше", "равно") и, следовательно, порядок расположения чисел на числовой оси. Форма представления, как следует из названия, определяет порядок записи числа с помощью предназначенных для этого знаков. При этом значение числа является инвариантом, т. е. не зависит от способа его представления. Это означает также, что число с одним и тем же значением может быть записано по-разному, т. е. отсутствует взаимно однозначное соответствие между представлением числа и его значением.

В связи с этим возникают вопросы, во-первых, о формах представления чисел и, во-вторых, о возможности и способах перехода от одной формы к другой.

Способ представления числа определяется *системой счисления*.

> ***Определение***
>
> *Система счисления* — это правило записи чисел с помощью заданного набора специальных знаков — цифр.

Людьми использовались различные способы записи чисел, которые можно объединить в несколько групп: унарная, непозиционные и позиционные.

Унарная — это система счисления, в которой для записи чисел используется только один знак — | (вертикальная черта, палочка). Следующее число получается из предыдущего добавлением новой палочки: их количество (сумма) равно самому числу. Унарная система важна в теоретическом отношении, поскольку в ней число представляется наиболее простым способом и, следовательно, просты операции с ним. Кроме того, именно унарная система определяет значение целого числа количеством содержащихся в нем единиц, которое, как было сказано, не зависит от формы представления.

Из *непозиционных* наиболее распространенной можно считать римскую систему счисления. В ней некоторые базовые числа обозначены заглавными латинскими буквами: 1 — I, 5 — V, 10 — X, 50 — L, 100 — C, 500 — D, 1000 — M. Все другие числа строятся комбинацией базовых в соответствии со следующими правилами:

- если цифра меньшего значения стоит справа от большей цифры, то их значения суммируются; если слева — то меньшее значение вычитается из большего;

- цифры I, X, C и M могут следовать подряд не более трех раз каждая;
- цифры V, L и D могут использоваться в записи числа не более одного раза.

Например, запись XIX соответствует числу 19, MDXLIX — числу 1549. Запись чисел в такой системе громоздка и неудобна, но еще более неудобным оказывается выполнение в ней даже самых простых арифметических операций. Отсутствие нуля и знаков для чисел больше M не позволяют римскими цифрами записать любое число (хотя бы натуральное). По указанным причинам теперь римская система используется лишь для нумерации.

В настоящее время для представления чисел применяют, в основном, *позиционные системы счисления*.

Определение

Позиционными называются системы счисления, в которых значение каждой цифры в изображении числа определяется ее положением (позицией) в ряду других цифр.

Наиболее распространенной и привычной является система счисления, в которой для записи чисел используется 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Число представляет собой краткую запись многочлена, в который входят степени некоторого другого числа — основания системы счисления. Например:

$$575{,}15 = 5\cdot 10^2 + 7\cdot 10^1 + 5\cdot 10^0 + 1\cdot 10^{-1} + 5\cdot 10^{-2}.$$

В данном числе цифра 5 встречается трижды, однако значение этих цифр различно и определяется их положением (позицией) в числе. Количество цифр для построения чисел, очевидно, равно основанию системы счисления. Также очевидно, что максимальная цифра на 1 меньше основания. Причина широкого распространения именно десятичной системы счисления понятна — она происходит от унарной системы с пальцами рук в качестве "палочек".

Однако в истории человечества имеются свидетельства использования и других систем счисления — пятеричной, шестеричной, двенадцатеричной, двадцатеричной и даже шестидесятеричной. Общим для унарной и римской систем счисления является то, что значение числа в них определяется посредством операций сложения и вычитания базисных цифр, из которых составлено число, *независимо от их позиции в числе*. Такие системы получили название *аддитивных*.

В отличие от них позиционное представление следует считать *аддитивно-мультипликативным*, поскольку значение числа определяется операциями