2.1. Что такое информация? [illegible] мы представления информации. 2. Цифровой автомат. 2.4. Как [illegible] работу ЦА? 2.5. Алгоритм и [illegible] ция. 2.6. Алгоритм и [illegible] 2.7. Универсальная машина Тьюрин га — прообраз ЭВМ. [illegible] мное управление в ЦА. [illegible] пути к ЭВМ или принцип хранимой в памяти программы. [illegible] адресации. 2.11. Что делает [illegible] универсальным ЦА?

АВТОМАТ, КОТОРЫЙ УМЕЕТ ВСЕ, ИЛИ ОСНОВЫ ИНФОРМАТИКИ

Человек обрабатывает информацию, руководствуясь не только правилами, но и тем, что мы называем здравым смыслом, интуицией. Машина же действует строго по правилам и «руководствуется» алгоритмом.

Не спешите утверждать, что алгоритм — набор правил, которые точно определяют последовательность действий. Ни рецепты поваренных книг, ни правила уличного движения, ни требования этики алгоритмами не являются. Наставления типа «немного посолить» или «с учетом ширины проезжей части» не удовлетворяют требованиям точности. В этике еще сложнее.

Вплоть до середины нашего века строгое определение алгоритма не смогли дать даже великие математики, пока не выяснилось, что точным во всех отношениях может быть только поведение машин.

По этой причине название главы следует понимать так: ЭВМ умеет делать все то, что умеют делать другие автоматы.

Что такое информация?

Информация — это какие-либо сведения о чем-то, совокупность данных об определенном объекте.

Такое толкование обусловлено происхождением самого термина (от лат. *informatio* — разъяснение, осведомление). Огромная популярность слова информация в наши дни определяется той ролью, которую начинают играть в нашей жизни ЭВМ, но подлинный смысл понятия информации глубже, чем интуитивное пред-

ставление о нем. Научное определение информации позволило подойти к совершенно разноплановым, на первый взгляд, процессам с единой точки зрения; подобным образом самые различные явления в физике объединяет понятие энергии.

Согласно наиболее общим воззрениям, информация — продукт отражения системой окружающей среды, или же продукт самоотражения системы. Под системой понимается материальный объект, помещенный в материальную среду. Если в объекте происходят какие-либо изменения, то он становится источником информации либо об окружающей среде, либо о протекающих в нем процессах. Информация имеет вполне материальные *носители* типа физических полей, нервных импульсов и т. п. Более высокая форма отражения характерна для организмов, обладающих центральной нервной системой, в которых разнообразная информация может складываться в целостное восприятие.

Мышлению человека свойственна высшая форма отражения объективной действительности — сознание. Наше понимание сути явлений и объектов представляет собой высшую форму информации, которая может соответствовать и нематериальным, воображаемым объектам.

Передача информации присуща всем явлениям природы. Однако, являясь пассивным отражением в системах неживой природы, информация становится активной в кибернетических или самоуправляемых системах, к которым можно отнести как живые организмы, так и автоматы.

Как весьма сложную кибернетическую систему можно рассматривать и человеческий мозг.

2.2. Формы представления информации

Информация может быть представлена в *аналоговой* или *дискретной* форме.

Если, например, электрическое напряжение или ток изменяются по тому же закону, что и некоторая другая физическая величина, то их называют электрическими аналогами этой физической величины. Разработано немало преобразователей различных параметров в их электрические аналоги. Подавляющее большинство преобразуемых величин изменяется непрерывно, поэтому и они, и их электрические аналоги принимают, строго говоря, бесконечное множество значений. По этой причине слово «аналоговый» стали отождествлять со словом «непрерывный». Оно употребляется сейчас лишь в этом смысле.

При дискретном или *цифровом* представлении информации используемая в качестве ее носителя физическая величина принимает конечное множество значений. Их удобно обозначать какими-либо символами.

Сравните лестницу и наклонную плоскость. В первом случае имеется строго определенное количество фиксированных высот, равное числу ступенек. Все их можно пронумеровать. Наклонная

плоскость соответствует бесконечному количеству значений высоты.

Лестницу можно приблизить к наклонной плоскости с какой угодно точностью, увеличивая число ступенек.

Подобная ситуация наблюдается и в технике. Если определена допустимая ошибка измерения, то любая аналоговая величина может быть заменена конечным набором своих дискретных значений, а они, в свою очередь, закодированы символами какого-нибудь алфавита, например пронумерованы обычными десятичными цифрами.

Закодированную информацию можно представить двумя принципиально различными способами. В первом из них символы кода в виде сигналов передаются *последовательно* один за другим по единственной линии связи, во втором — более быстром — группа символов передается *параллельно* по нескольким линиям связи одновременно.

2.3. Цифровой автомат

Назовем *цифровым автоматом* (ЦА) всякую искусственную систему обработки информации, представленной в дискретной форме. Такое устройство имеет вход и выход. На них используется параллельное представление информации, а ЦА для обработки последовательной информации выступает как частный случай. Совокупность входных и выходных сигналов ЦА в данный момент времени называют соответственно *входным* и *выходным словом*. Работу любого ЦА можно трактовать как преобразование входных слов в выходные слова того же или другого алфавита (рис. 2.1). Под алфавитом понимают множество всех использованных символов. Входные и выходные слова могут изменяться только в определенные моменты времени, называемые тактовыми. Их удобно обозначать целыми неотрицательными числами.

$$t = 0,\ 1,\ 2,\ \ldots,\ i,\ \ldots .$$

Интервал времени между двумя соседними тактовыми моментами называется *тактом*. Длительность тактов может быть неодинаковой, но в большинстве случаев они равновелики и определяются генератором синхронизации.

На первый взгляд ЦА напоминает словарь, который каждому слову ставит в соответствие иноязычный эквивалент. А вот для того чтобы он напоминал переводчика, необходимо учесть, что смысл слова, как правило, находится в тесной связи с предшеству-

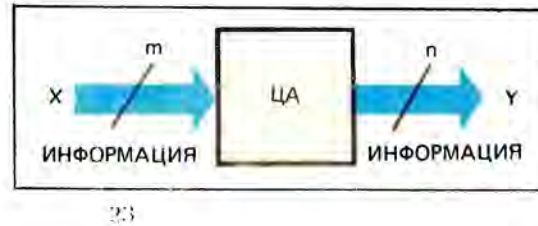


Рис. 2.1. Цифровой автомат можно рассматривать как преобразователь m-разрядных входных слов в n-разрядные выходные слова.

ющим текстом, поэтому в общем случае ЦА должен обладать способностью запоминать поступающую в него информацию. Однако в отличие от переводчика в некоторых случаях ЦА достаточно запомнить не само слово, а только факт его появления, что позволяет существенно экономить объем памяти.

Поясним сказанное примером. Типичный ЦА с памятью — обычные часы. Их показания в каждом такте зависят только от показаний в предыдущем такте. Сами показания являются выходными словами, а входной информацией служат сигналы с периодом в 1 секунду, вырабатываемые механическим или электрическим способом. За сутки на такой автомат поступает $24 \times 60 \times \times 60 = 86\,400$ входных слов и вырабатывается столько же различных выходных слов. Таким образом, часы имеют 86 400 состояний, но для их хранения, в принципе, достаточно пяти элементов памяти, каждый из которых способен запоминать лишь одну десятичную цифру. Обычно таких элементов шесть, и они соответствуют традиционному исчислению времени с помощью единиц и десятков часов, минут и секунд. При хранении же всех входных слов потребовалось бы 86 400 элементов для запоминания каждого входного сигнала.

Если же выходная информация зависит не только от факта появления, но и от значения многих предыдущих слов, объем памяти ЦА возрастает.

Так, цифровому автомату, предназначенному для выполнения арифметической операции сложения последовательно поступающих чисел, достаточно запомнить лишь одно слагаемое до прихода второго. А вот при обработке текста смысл предложения становится ясным только после приема последнего слова, поэтому вплоть до этого момента в памяти должны храниться, по крайней мере, все предыдущие слова предложения. Хранимую в памяти ЦА информацию о «предыстории» его работы принято называть *внутренним состоянием*. Удобно обозначать внутреннее состояние одним словом Q подходящей длины.

2.4. Как описать работу ЦА

Итак, ЦА под действием входного слова переходит из одного состояния в другое, сохраняя принятое состояние, по крайней мере, в течение такта, причем в общем случае выходное слово зависит не только от входного сигнала, но и от состояния автомата.

Любой цифровой автомат может быть построен из устройств двух типов — *комбинационных схем* (КС) и *схем с памятью*, которые сами могут рассматриваться как элементарные ЦА. Выходное слово КС в каждом такте однозначно определяется входным словом, и работу КС можно формально описать функцией

$$Y = \Phi(X).$$

Простейший способ задания $\Phi(X)$ — таблица, в которой всем возможным входным словам X ставятся в соответствие значения Y, например таблица 2.1.

Таблица 2.1

Входное слово X		Выходное слово Y	
x_1	x_2	y_1	y_2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Схемы с памятью, а в общем случае и произвольные ЦА описываются выражениями

$$Y_{t+1} = \Phi(X_t, Q_t),$$
$$Q_{t+1} = F(X_t, Q_t),$$

где индексы $t+1$ и t относятся, соответственно, к новому и предыдущему тактам работы ЦА. Функции Φ и F и в этом случае удобно задавать таблично. Совместное изображение функций Φ и F получило название *таблицы переходов* ЦА. В такой таблице, например таблице 2.2, предыдущее состояние ЦА выступает на правах аргу-

Таблица 2.2

Предыдущее состояние ЦА Q_t	Входное слово X		Выходное слово Y	Новое состояние Q_{t+1}
$q_1(t)$	$x_1(t)$	$x_2(t)$	$y_1(t+1)$	$q_1(t+1)$
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	1	1
1	1	1	0	0

мента, а новое — в качестве функции. В простейших схемах с памятью, которые будут рассматриваться в главе 6, внутреннее состояние обычно отождествляют с выходным словом, а их работа описывается одной функцией

$$\Phi \equiv F.$$

Структура и работа любого ЦА тесно связаны с формой представления информации. Если вся входная информация может быть представлена параллельно, в виде одного слова, то запоминание ее необходимо лишь в тех случаях, когда элементы ЦА не рассчитаны

на обработку длинных слов. В остальных случаях весь ЦА может быть выполнен в виде одной КС, и его быстродействие будет максимальным.

При последовательной форме представления информации неизбежно ее запоминание. По этой причине схемы с памятью иногда называют последовательными.

На практике обычно принимается некоторое компромиссное решение: информация представляется в смешанной параллельно-последовательной форме, чем обеспечивается приемлемое соотношение сложности устройства ЦА и его быстродействия.

2.5. Алгоритм и интуиция

Еще раз отметим различный характер обработки информации в ЦА типа комбинационных схем (КС) и в ЦА с памятью.

В комбинационных схемах выходное слово формируется из входного со скоростью распространения сигнала через схему. Происходит это в одном такте.

В автоматах с памятью имеет место процесс обработки информации, распадающийся на звенья, каждое из которых выполняется в течение одного такта и может быть описано на языке КС. С этим процессом и связано понятием *алгоритма* обработки информации. (Слово «алгоритм» произошло от имени знаменитого математика IX века Аль-Хорезми, сформулировавшего правила арифметических действий над числами.)

В математике получило широкое распространение определение алгоритма как точного предписания конечной последовательности однозначных, понятных действий, направленных на достижение указанной цели, исходя из некоторых начальных данных.

Уточним особенности сформулированного определения:

1. Алгоритмы создаются для исполнителей, которых интересует конечный результат, а не тонкости процесса решения задачи. Даже когда алгоритм создается «для себя», то преследуется цель ускорить и упростить решение, так чтобы это можно было сделать не задумываясь, «автоматически».

2. Отдельный алгоритм зачастую составляет часть другого, более сложного алгоритма. Так, алгоритм решения квадратного уравнения опирается на вспомогательные алгоритмы алгебраических операций над числами. Вообще, решение любой сложной задачи существенно упрощается, если ее можно свести к последовательности подзадач. Это тем более имеет смысл, если подзадачи типичны для широкого круга других задач.

3. Любой алгоритм является дискретным, т. е. распадается на последовательность предписаний, направленных на выполнение некоторых элементарных действий или операций. Эти предписания должны быть понятны исполнителю. Образно говоря, исполнитель должен хорошо владеть языком, на котором описан алгоритм, в противном случае он нуждается в услугах хорошего переводчика.

4. Исполнитель должен уметь правильно производить все элементарные действия, содержащиеся в алгоритме.

5. Алгоритмам обычно свойственна массовость, т. е. применимость к различным исходным данным. Решение любой задачи «в общем виде» по существу всегда представляет собой некоторый алгоритм.

6. Наконец, алгоритм должен быть результативным, т. е. приводить к цели за конечное число шагов. Если же процесс решения завершается безрезультатно или не заканчивается вовсе, то говорят, что алгоритм не применим к взятым исходным данным.

Несмотря на значительные достижения в разработке и распространении всевозможных алгоритмов в математике, попытки научного подхода к алгоритмам вплоть до XX века были малоуспешными. Причина — трудоемкость строгого, формального определения понятия алгоритм. Формулировка, приведенная выше, может быть названа определением лишь в интуитивном смысле. Она не является строгой. В ней нет указаний на то, что может быть объектами алгоритма, а понятия типа «точное предписание», «понятные действия» — расплывчаты.

2.6. Алгоритм и программа

Отсутствие формального определения алгоритма приводило к ряду заблуждений. Так, немецкий математик Готфрид Вильгельм Лейбниц (1646—1716) пытался построить ни более ни менее как общий алгоритм решения любой алгебраической задачи. Аналогичные попытки в несколько более конкретной форме предпринимались даже в начале нашего века. Безуспешность всех этих поисков постепенно привела к выводу о том, что такие задачи *алгоритмически неразрешимы*. Проблема формального определения алгоритма стала еще более острой. Ее решение было найдено лишь в 30-х годах нашего столетия. В его основе лежало осознание того, что любой алгоритм оперирует не с реальными объектами, а с их символическими отображениями. Несколько позднее эту мысль удалось выразить еще более точно: объектами алгоритмов является информация, представленная в дискретной форме. Выяснилось также, что гарантией точности и однозначности алгоритма служит однозначность и точность языка, на котором он описывается. Причина расплывчатости интуитивного определения алгоритма заключается в том, что его исполнителем неявно предполагался человек, получающий предписания на естественном языке, одна из важнейших особенностей которого — многозначность. Она обусловлена наличием омонимов, синонимов, контекстностью, т. е. существенной зависимостью смысла от предшествующего изложения и т. п. Человеческому языку свойственна избыточность: пониманию текста зачастую не препятствуют как оговорки, описки, пропуски букв, так и сокращения целых слов. Все эти особенности способствуют образности, яркости и насыщенности при выраже-

нии самых различных мыслей, эмоций, но затрудняют процесс формализации многих понятий, в том числе и алгоритма.

Не исключено, что подобного рода соображения привели английского ученого А. Тьюринга в 1936 году к идее: предложить схему некоторой машины и назвать алгоритмами все то, что она умеет делать.

Не вдаваясь в детали устройства, отметим, что *машина Тьюринга* (МТ) представляет собой цифровой автомат, оперирующий словами единичной длины (т. е. символами) и снабженный запоминающим устройством в виде бесконечной ленты, на которой записываются символы входного слова, промежуточные результаты, а в итоге работы — и выходное слово.

Алфавит информационных слов, с которыми оперирует машина Тьюринга, может быть произвольным, но конечным; он называется внешним. Помимо него, необходим внутренний алфавит для обозначения внутренних состояний и некоторых особых ситуаций. В результате работа машины Тьюринга допускает символическое описание в виде специальной таблицы переходов. В каждой строке этой таблицы текущему входному символу и текущему состоянию ставится в соответствие выходной символ, новое состояние и указание, слева или справа от текущего нужно взять следующий обрабатываемый символ. Строки таблицы называются *командами*, а таблица в целом — *программой*. Работа МТ начинается с «настройки» на начальное внутреннее состояние и первый символ входного слова. По ним в программе находится нужная команда, а в результате — выходной символ, новое состояние и место, откуда нужно считать следующий входной символ. Так происходит до тех пор, пока не будет достигнуто особое, «конечное» состояние, соответствующее решению задачи.

Основной результат работы Тьюринга заключается не в том, что для любого алгоритма в интуитивном смысле может быть построена соответствующая машина, хотя этот факт и не вызывает сомнений в результате многочисленных проверок. С помощью работы Тьюринга можно доказать, что МТ не может решать задачи, которые интуитивно неразрешимы, в том числе упоминавшуюся проблему Г. Лейбница.

Неразрешимость таких задач на машине Тьюринга не есть следствие ее примитивности. Тьюринг, действительно, искал как можно более простую схему, позволяющую не только формализовать процесс решения, но и обладающую применимостью к любым задачам. Предположение Тьюринга о том, что всякий алгоритм может быть реализован соответствующей машиной, было всего лишь гипотезой. Однако весь последующий опыт позволил возвести этот тезис в ранг формального определения алгоритма. И пожалуй, самым впечатляющим доказательством справедливости идеи Тьюринга явилось установление эквивалентности этого определения другим формальным определениям, данным независимо советскими математиками А. А. Марковым и А. Н. Колмогоровым и американцем Э. Постом.

2.7. Универсальная машина Тьюринга — прообраз ЭВМ

Машина Тьюринга — воображаемая конструкция, построить ее — значит выбрать подходящий для данной задачи алфавит, написать символическую программу и убедиться, будет ли достигнуто в ходе ее выполнения конечное состояние. Это можно сделать путем рассуждений, даже не получив конечного результата.

Работа МТ происходит в нашем воображении. Следовательно, машины Тьюринга — вовсе не машины в обычном смысле, а скорее тексты на некотором языке, что вполне соответствует нашему интуитивному представлению об алгоритме.

А является ли алгоритмом работа любой МТ? В поведении МТ есть много общего: все, что они «умеют», сводится к достаточно простым операциям поиска текущей команды, преобразования входного символа в выходной, изменения внутреннего состояния и нахождения следующего входного символа. Эти операции носят настолько регулярный, «механический» характер, что их вполне можно описать в виде единственной программы для подходящей машины — *универсальной машины Тьюринга* (УМТ).

Идея УМТ заключается в имитации работы любой конкретной МТ. С этой целью в памяти УМТ должны храниться программа работы и образ имитируемой машины, включающий информацию как о содержимом ее памяти с указанием на текущий входной символ, так и о текущем состоянии.

На первый взгляд это кажется невозможным, ведь множество всех имитируемых машин бесконечно. Но тексты на любых языках можно зашифровать (закодировать) с помощью единственного, например цифрового алфавита. Сделать это, очевидно, нужно так, чтобы можно было достаточно просто различать программу от образа машины, символы от состояний и т. п. В этом случае программа работы УМТ сводится к выполнению системы несложных действий.

В образе МТ ищутся коды первого символа и начального состояния. По ним в зашифрованном тексте программы находится нужная команда, несущая информацию о тех изменениях, которые нужно произвести в образе МТ. После их совершения коды нового состояния и следующего символа позволяют отыскать следующую команду и т. д. Этот процесс, называемый *интерпретацией*, продолжается до тех пор, пока не будет достигнуто конечное состояние. В этом случае в памяти УМТ окажется код выходного слова имитируемой МТ.

В воображаемом устройстве УМТ использованы важные идеи, нашедшие позднее воплощение в особенностях конструкции и работы настоящих ЭВМ. В первую очередь, это работа по программе, представляющей собой символическую запись алгоритма на языке, «понятном» для исполнения не очень сложным устройством. Во-вторых, это наличие запоминающего устройства для хранения как исходной, промежуточной и конечной информации, так и самой программы. Память реальных ЭВМ всегда конечна, но это ограни-

чение становится несущественным в современных ЭВМ, обладающих весьма большой памятью. Наконец, это возможность имитации работы любых специализированных устройств обработки информации на одной универсальной конструкции.

2.8. Программное управление в ЦА

Итак, мы вправе считать ЦА устройством, реализующим конкретный алгоритм обработки информации. Его основная деятельность — операции над символами — выполняется комбинационными схемами. Но в нем происходят и другие процессы типа изменения состояния схем с памятью, записи слов в запоминающее устройство. Все они должны быть «привязаны» к определенным тактам, наконец, должно иметься устройство фиксации самих тактовых моментов.

Эта достаточно сложная роль в модели Тьюринга поручена человеку, а в реальных ЦА — устройству управления. В результате любой ЦА можно представить состоящим из *запоминающего устройства* (ЗУ), *операционного устройства* (ОУ) и *устройства управления* (УУ) (рис. 2.2).

В состав ОУ входят комбинационные схемы, общая функция которых — трансформация произвольного символа выбранного алфавита в любой другой символ этого алфавита. В главе 4 будет показано, как это можно сделать с помощью конечного набора элементарных операций. Всякое действие, инициируемое одним управляющим сигналом и выполняемое за один такт, называется *командой*. Таким образом, функция УУ — порождение соответ-

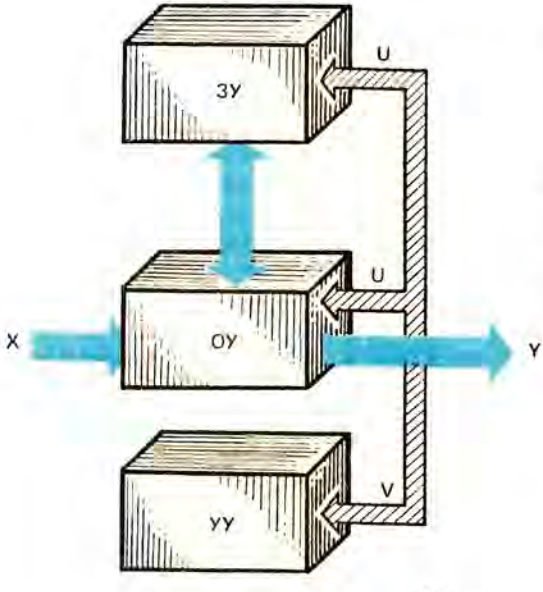


Рис. 2.2.
В процессе работы цифрового автомата ОУ производит обмен информацией с ЗУ и окружающей средой. Выделены сигналы управления, поступающие из УУ. В ответ на них остальные блоки могут выдавать сигналы оповещения *V*, сигнализирующие УУ об окончании выполнения текущей операции, особых ситуациях в ходе ее выполнения и т. п.

Рис. 2.3.
ЦА, использующие принцип программного управления, не могут работать без символической (закодированной) программы, записанной в явном виде на каком-либо носителе информации.

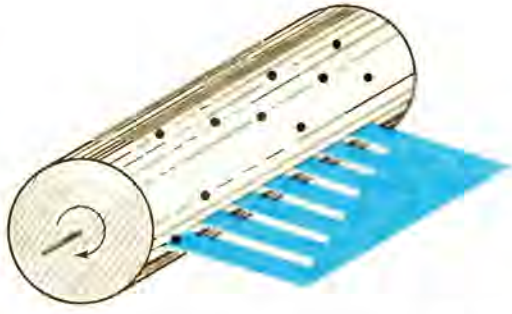


Рис. 2.4.
Работа несложного музыкального автомата — одно из самых древних воплощений принципа программного управления. Штырьки, расположенные на вращающемся барабане, задевают при своем движении вибрирующие пластинки, издающие звуки различной высоты. Продуманное расположение штырьков позволяет воспроизвести некоторую мелодию. Чтобы заставить автомат исполнить новую мелодию, достаточно расположить штырьки иначе.

ствующей выбранному алгоритму последовательности команд, т. е. *программы*.

До сих пор мы говорили о программе как способе символического описания алгоритма. Теперь понятие программы приобретает смысл способа функционирования ЦА, действия которого инициируются последовательностью сигналов УУ. Если устройство управления рассчитано на выполнение единственной программы и работает без внешних воздействий, то говорят, что оно обладает *жесткой или схемной логикой* работы.

При другом подходе УУ выдает управляющие сигналы лишь тогда, когда на его специальном входе присутствует код текущей команды программы. Выполнение каждой команды разбивается на считывание этого кода с внешнего носителя информации и собственно исполнение.

Само УУ играет роль преобразователя закодированных команд программы в управляющие сигналы. Такой подход к организации УУ называется *принципом программного управления* (рис. 2.3). Он позволяет перейти к реализации другого алгоритма без каких бы то ни было переделок УУ. Для этого достаточно записать на носитель информации новую программу работы.

Принцип программного управления известен человечеству давно и реализован в музыкальных автоматах (рис. 2.4), игрушках, многих механизмах. В качестве носителя информации в таких устройствах использовались вращающиеся диски, барабаны с отверстиями или выступами, зубчатые колеса, перфоленты. К сожалению, все они имеют низкую скорость работы.

2.9. На пути к ЭВМ или принцип хранимой в памяти программы

Воображаемая машина Тьюринга подсказывает один из способов построения запоминающих устройств (ЗУ). Длинная лента с символами или отверстиями (перфолента) – не что иное, как *ЗУ с последовательным* доступом; найти нужный символ (или группу символов) на ней можно лишь путем последовательного просмотра всех предшествующих записей. Такие ЗУ удобны для хранения программы, потому что команды программы обычно выполняются последовательно.

Что же касается хранящихся в ЗУ символов, с которыми оперирует программа, то для них последовательный характер доступа — не самый удачный. В общем случае нужная цифровому автомату информация может находиться в произвольных местах (ячейках) ЗУ и возникает проблема, как ее разыскать. Эта проблема может решаться различными путями. Наиболее распространен подход, при котором все ячейки ЗУ нумеруются, а нужная из них разыскивается по номеру, называемому адресом. ЗУ, использующие этот принцип, называются *адресными*. Последовательные ЗУ также можно превратить в адресные, если на носителе рядом с самим символом хранить его адрес.

Последовательный характер работы ЗУ неминуемо снижает быстродействие. Действительно, если после символа, расположенного в одном «конце» ЗУ, понадобилась информация из другого «конца», необходим бесполезный просмотр содержимого всех промежуточных ячеек.

Адресные ЗУ с одинаковым временем доступа ко всем ячейкам получили название *ЗУ с произвольной выборкой* (ЗУПВ). Время доступа к ячейке с любым адресом в наиболее совершенных из них составляет десятые и даже сотые доли микросекунды! Быстродействие же операционных устройств, основу которых составляют комбинационные схемы, может быть еще выше.

А что, если сама программа будет поступать в УУ не с внешнего носителя, а из ЗУПВ?

Принято считать, что эта идея принадлежит американцу венгерского происхождения Дж. фон Нейману, принявшему весьма активное участие в становлении и развитии вычислительной техники. Метод, при котором закодированная программа работы ЦА размещается вместе с обрабатываемой информацией в ЗУ ЦА, получил название *принципа хранимой в памяти программы*, а само ЗУПВ — оперативного запоминающего устройства (ОЗУ). Свое применение он нашел уже в ранних моделях ЭВМ и с тех пор является одной из основных, неотъемлемых особенностей любого компьютера.

Адресный характер ОЗУ приводит к тому, что команды, на которые теоретически распадается алгоритм решения любой задачи, в общем случае содержат следующую информацию:

1) адрес преобразуемого символа;

Рис. 2.5. Четырехадресная команда содержит код операции (КОП), т. е. символическое обозначение выполняемого действия, а также адреса первого и второго операндов, результата действия над ними и следующей команды (соответственно: A1, A2, A3, A4).

2) указание на элементарную операцию, которую нужно совершить над этим символом;
3) адрес ячейки ОЗУ, куда нужно поместить результат;
4) адрес ячейки ОЗУ, где хранится следующая команда программы.

Помимо этого, команда в общем случае должна учитывать внутреннее состояние УУ и при необходимости изменять его.

На практике, кроме операций трансформации одного символа в другой, удобны операции над парами символов. Рассмотрим, например, как выполняет МТ сложение двух чисел, допустим 3 и 4. В первом такте она «видит» цифру 3. Чтобы ее запомнить, есть только один выход — изменить свое состояние. Когда на следующем такте МТ «видит» четверку и пишет вместо нее 7, то это можно трактовать как трансформацию символа 4 в 7 с учетом предыдущего состояния. С таким же успехом можно считать, что символ 7 получается в результате действия (сложения) сразу над двумя символами 3 и 4, а внутреннее состояние не играет роли.

Следовательно, в реальных автоматах могут быть очень удобны команды, оперирующие сразу двумя символами. Будем называть символы, а в общем случае — любые порции информации, с которыми оперирует команда, *операндами*. (Если ЦА использует параллельное представление информации, то операндом может быть и слово.) Тогда любая команда в общем случае должна состоять из составляющих, представленных на рисунке 2.5. Команды такого вида называют четырехадресными. Они могут занимать в ОЗУ сразу несколько последовательных ячеек. Конечно, УУ должно уметь «распознать», сколько ячеек занимает считываемая команда, чтобы случайно не принять в качестве продолжения этой команды начало следующей или посторонний операнд. С этой целью код операции (КОП) всегда располагается в первой из ячеек, занятых командой, и обязательно содержит информацию об их суммарном числе. При этом исключается всякая путаница, ведь A4 указывает на первую ячейку следующей команды с ее КОП, который «подсказывает» УУ, сколько ячеек нужно «прочитать» еще.

Простота составления программы с помощью четырехадресных команд имеет и негативную сторону. Длина команд получается большой, программа занимает много места в ОЗУ, снижается скорость ее выполнения. Существует несколько путей преодоления этих сложностей, называемых проблемой адресации. Рассмотрим некоторые из них.

Если в ЗУ с последовательной выборкой расположить команды в той последовательности, в какой они следуют по программе, то переход от одной команды к следующей будет происходить автоматически, и в адресе А4 нет необходимости.

Для ЗУПВ можно исключить из каждой команды адрес А4, если ввести в состав УУ специальное устройство — *счетчик команд* (СК). В начале работы этот счетчик содержит адрес первой команды программы. Распознав ее КОП, УУ увеличивает содержимое СК на число ячеек, занятых этой командой. Если команды в ОЗУ расположены последовательно, то новое содержимое СК теперь является адресом второй команды. Нетрудно заметить, что в общем случае СК служит указателем адреса команды, следующей за текущей. В результате все команды могут быть трехадресными.

Заметим, что если нужно нарушить естественную последовательность выполнения команд, например из-за изменившегося в результате действия оповещающих сигналов состояния УУ, или при реализации разветвляющегося алгоритма, то для этого необходимо принудительно изменить содержимое СК. С этой целью используют специальные *команды передачи управления* (ПУ). Возможная их структура представлена на рисунке 2.6. *Команды условной передачи управления* удобно использовать для алгоритмов, содержащих ветвление в зависимости от выполнения или невыполнения какого-либо условия.

Команды безусловной передачи управления не содержат кода условия и всегда обеспечивают переход к команде, расположенной по адресу А. Они удобны, если в последовательном расположении команд программы в памяти есть «островки», содержащие операнды или побочные ветви программы.

Наличие команд передачи управления позволяет рассматривать содержимое счетчика команд как аналог текущего состояния МТ.

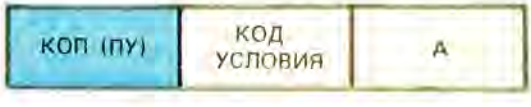
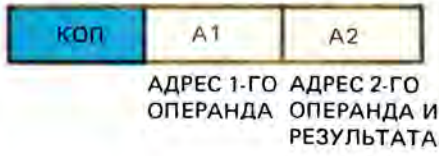


Рис. 2.6. Команда условной передачи управления содержит адрес команды, к выполнению которой должно перейти УУ, если выполнено некоторое условие, например если внутреннее состояние УУ соответствует нужному.

Рис. 2.7
В двухадресной команде адреса результата операции (А3) и следующей команды программы (А4) содержатся неявно: А3 = А2, а А4 определяется посредством счетчика команд.

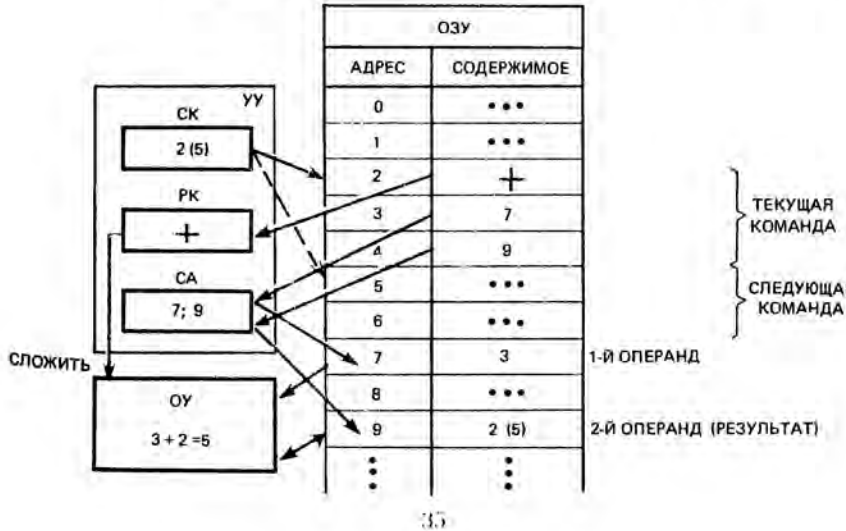


Путь дальнейшего сокращения адресной части команд — размещение в ОЗУ результата выполнения команды на месте одного из операндов, обычно второго. В результате команды ЦА могут быть сделаны двухадресными (рис. 2.7).

Процесс выполнения *двухадресной команды* рассмотрим на примере. Пусть набор символов, с которыми оперирует ЦА, включает десятичные цифры и нас интересует команда сложения чисел 3 и 2, расположенных в ОЗУ по адресам А1 = 7 и А2 = 9. Такая команда может быть закодирована символами +, 7, 9. Предположим, что каждая ячейка ОЗУ рассчитана на хранение одного символа, и символ + находится во второй ячейке. Тогда перед выполнением команды содержимое СК должно равняться 2. Для доступа к указанным в команде операндам УУ использует *систему адресации* (СА), обеспечивающую обращение к ячейкам ОЗУ по их адресам (рис. 2.8).

Еще большее сокращение адресной части команд допускает

Рис. 2.8.
В процессе выполнения «команды» + 7,9 код операции (+) поступает для хранения в специальное устройство — регистр команд (РК) и определяет характер действия, производимого в ОУ. По адресам А1=7 и А2=9 СА находит операнды (3 и 2), передает их в ОУ и после выполнения сложения помещает результат по адресу А2=9. В круглых скобках на рисунке обозначены состояния ячеек памяти и СК, изменившиеся в результате выполнения команды.

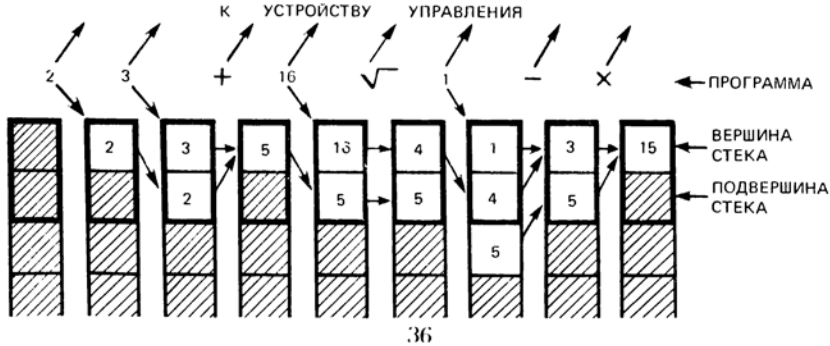


использование так называемого *сверхоперативного запоминающего устройства* (СОЗУ) — небольшого количества быстродействующих элементов памяти, объединенных в *регистры*, устройство которых будет пояснено в 6-й главе. Преимущества *регистровой памяти* обусловлены тем, что часто текущая команда программы использует в качестве одного и даже двух операндов результаты непосредственно предшествующих ей команд. Если они заносятся в СОЗУ, то адресная часть команд не будет занимать много места, т. к. адреса регистров СОЗУ в силу небольшого их числа весьма коротки. Такой метод адресации операндов называется *регистровым*. Если же вместо самих операндов в СОЗУ хранить их адреса A1 и A2, то появляется возможность доступа к операндам всего ОЗУ. Адресная часть команд при этом не увеличивается, так как в ней называются не сами A1 и A2, а короткие адреса регистров. Такая адресация называется *косвенно-регистровой*. С использованием регистров СОЗУ можно организовать и другие методы адресации, поэтому их называют *регистрами общего назначения* (РОН).

Большинство современных ЭВМ рассчитано на работу с двухадресными командами и широким использованием РОН. Как частный случай возможны *одноадресные команды* (например, передачи управления) и *безадресные команды*, не обращающиеся к ЗУ.

В принципе существует возможность построения ЦА, использующих только *нуль-адресные команды*, т. е. команды, не имеющие явного указания ни на один из операндов. Эта возможность основана на представлении алгоритмов в виде т. н. бесскобочной или *польской инверсной записи* (ПОЛИЗ). При этом исключительно удобна организация ЗУ для хранения операндов по принципу «последним пришел — первым вышел» или LIFO (от англ. *Last In-First Out*). ЗУ, использующее этот принцип, называют

Рис. 2.9. Воспринимая программу работы, УУ нульадресной машины должно различать операнды (2, 3, 16, 1) от знаков операций над ними $(+, \sqrt{\ }, -, \times)$. Если очередная «команда» — операнд, то его нужно «протолкнуть» в стек; если это — знак операции, то нужно выполнить данную операцию. Говоря о «проталкивании» и «выталкивании», имеют в виду аналогию между работой стека и магазина для патронов в пистолете-автомате.



стеком: оно запоминает символы в порядке их поступления, а выдает в обратном.

Возможность использования нуль-адресных команд и работу стека поясним на примере вычисления арифметического выражения $(2+3)\times(\sqrt{16}-1)$. В ПОЛИЗ это выражение пишется 2,2, +, 16, $\sqrt{}$, 1, —, $\times$, т. е. так, чтобы знаки операций следовали за соответствующими операндами. Легко заметить, что при этом порядок действий определен совершенно однозначно, хотя скобки не используются. Последнее выражение и представляет собой программу работы УУ, выполнение которой требует, чтобы каждое арифметическое действие производилось над операндами, расположенными в вершине и подвершине стека. При этом результат помещается в вершину, а остальное содержимое стека «выталкивается» на одну позицию вверх. Для одноместных (с одним операндом) действий типа извлечения корня операция производится только над вершиной стека. На рисунке 2.9 изображено содержимое стека после выполнения каждой команды. В начале стек пуст.

Стековые вычисления широко применяются в микрокалькуляторах.

2.11. Что делает ЭВМ универсальным ЦА?

В словарях можно найти примерно следующее определение: ЭВМ — универсальное автоматическое устройство обработки цифровой информации, особенностями которого являются принцип программного управления и принцип хранимой в памяти программы. Очевидное отличие ЭВМ от любого другого ЦА заключается в *универсальности,* под которой, согласно теории алгоритмов, следует понимать возможность интерпретировать работу любого автомата, т. е. реализовать произвольный алгоритм.

Как достигается универсальность?

Подытожим известное.

1. Устройство должно допускать описание своей работы на языке машины Тьюринга. Этому требованию отвечает любой ЦА, в том числе и с жесткой логикой, состоящий из комбинационных схем и схем с памятью. Функции управления им способно осуществлять простейшее устройство, состоящее из *генератора тактов.*

2. Техническое воплощение в УУ принципа программного управления позволяет решать множество задач, для представления которых достаточно одного алфавита. Программа, соответствующая выбранному алгоритму, размещается на внешнем носителе и может быть легко изменена. Нужные изменения внутреннего состояния ЦА могут достигаться действием оповещающих сигналов на элементы памяти УУ. Последовательный во времени характер обработки во многих случаях приводит к использованию ЗУ для хранения входной информации и промежуточных результатов.

3. Реализация принципа хранимой в памяти программы повышает быстродействие УУ до быстродействия запоминающих

устройств. Важную роль при этом приобретает проблема адресации. Каждая хранящаяся в памяти команда программы состоит из кода производимого ею действия и может содержать адресную часть, указывающую УУ на место расположения нужных операндов в ЗУ. Изменения состояния УУ могут отражаться поведением счетчика команд. Возможность его принудительной установки с помощью команд передачи управления позволяет очень удобно учитывать изменения состояния ЦА в самой программе. При трансформации символов удобны двухоперандные (двухместные) операции.

4. При таком полном учете внутренних состояний автомата универсальность как способность решать все задачи, написанные на некотором фиксированном алфавите, достигается при условии, что существует набор команд, позволяющий осуществить произвольное преобразование типа «операнд-операнд» или «пара операндов-операнд». Будем называть такой набор команд трансформации полным.

Для алфавита ЦА наиболее удобны цифровые в буквальном смысле системы, универсальность которых обосновывается в следующей главе. И в этой связи возникает вопрос, образуют ли арифметические действия полный набор для преобразования чисел? Решение этого вопроса не совсем очевидно. (Попробуйте указать арифметические действия над числами 7 и 5, приводящие к числу 3 или дающие из числа 1 число 6.)

А не существует ли способ, допускающий произвольную трансформацию чисел? Такое преобразование может описываться понятием функции. Значения некоторых функций могут быть получены из значений аргумента с помощью известных математических действий. Функцию удобно задавать аналитически, в виде символической записи этих действий, но это возможно не для любой функции. Более универсально табличное задание функций. Любую такую таблицу можно разместить в ЗУ ЭВМ, а для того чтобы ею пользоваться, достаточно единственной команды, позволяющей перемещать операнд из одной ячейки ЗУ (включая РОНы и регистры внешних по отношению к ЭВМ устройств) в другую. Такие команды называются *командами пересылки*. В общем случае они должны содержать два адреса: ячеек источника и приемника операнда.

Представим себе теперь, что значения функции размещены в ячейках ОЗУ с адресами, равными соответствующим значениям аргументов. Тогда для нахождения значения функции достаточно поместить соответствующий аргумент в РОН и использовать его как косвенный адрес операнда-источника в команде пересылки. В результате ее выполнения значение функции окажется в ячейке (или РОН) по указанному в команде второму адресу. Работа с функциями, заданными таблично,— одно из важнейших применений косвенной адресации.

Несмотря на свою простоту, описанный механизм становится малоэффективным при увеличении длины операндов. Так, для

двухместных операций, даже, если операнды состоят из одной десятичной цифры, в ЗУ необходимо отвести 100 (10×10) ячеек для хранения значений функции при всех возможных комбинациях операндов. Естественное стремление к повышению быстродействия приводит к тому, что ЭВМ использует параллельное представление информации в виде слов (чисел), состоящих из нескольких символов (цифр). В этом случае затраты памяти становятся непомерно велики.

К счастью, можно поступить проще. Запись многоразрядных чисел обычно использует те же цифры, что и для одноразрядных. Поэтому произвольное преобразование таких чисел можно производить «цифра за цифрой», т. е. оно сводится к рассмотренному уже преобразованию цифр.

Оказывается, это можно сделать, не прибегая к таблицам, а в «аналитическом» виде с помощью конечного числа операций, называемых *логическими*. Устройства, в которых технически осуществлено выполнение этих операций,— *логические элементы* — являются «кирпичиками», из которых строится любой ЦА, в том числе и ЭВМ. Наряду с передачей управления и пересылками возможность выполнять логические операции в ОУ с помощью специальных команд придает автоматам, использующим принципы программного управления и хранимой в памяти программы, качество универсального ЦА, т. е. ЭВМ.