Доминяк В. И.

ВВЕДЕНИЕ В ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Курс лекций

Санкт-Петербург
1996

Содержание:

1. ЭВМ: цифровые, аналоговые, гибридные

Любая форма человеческой деятельности, любой процесс функционирования технического объекта связаны с передачей и преобразованием информации.

Одно из важнейших положений кибернетики состоит в том, что без информации и её переработки невозможны организованные системы, какими являются живые организмы и искусственные, созданные человеком технические системы.

Информацией называют сведения о тех или иных явлениях природы, событиях в общественной жизни и процессах в технических устройствах. Информация, воплощённая и зафиксированная в некоторой материальной форме, называется *сообщением.* Например, при составлении расписания занятий собираются и обрабатываются сведения (сообщения) о количестве классов, количестве и наименовании предметов, количестве часов по тому или иному предмету, возможностях учителей и т.п. и на их основе составляется сетка занятий.

Сообщения могут быть непрерывными и дискретными (цифровыми).

Непрерывное (аналоговое) сообщение представляется некоторой физической величиной (электрическим напряжением, током и др.), изменения которой во времени отображают протекание рассматриваемого процесса, например изменение температуры в печи (рис. 1.1.). Физическая величина, передающая непрерывное сообщение, может в определённом интервале принимать любые значения и изменяться в произвольные моменты времени.

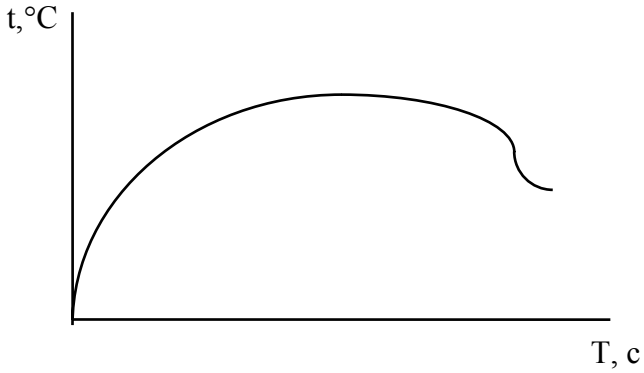


Рис. 1.1. Изменение физической величины (температуры) во времени

Для *дискретных сообщений* характерно наличие фиксированного набора элементов, из которых в некоторые моменты времени формируются различные последовательности. Важным является не физическая природа элементов, а то обстоятельство, что набор элементов конечен и поэтому любое дискретное сообщение конечной длины передаёт конечное число значений некоторой величины. Предположим, что температура в печи фиксируется только в определённые моменты времени, тогда график температуры будет соответствовать рис. 1.2. Мы получили дискретное по времени сообщение.

Элементы, из которых состоит дискретное сообщение, называют *буквами* или *символами.* Набор этих букв образует *алфавит.* Здесь под буквами в отличие от обычного представления понимаются любые знаки (обычные буквы, цифры, знаки препинания, математические и прочие знаки), используемые для представления дискретных сообщений.

При дискретной форме представления информации отдельным элементам её могут быть присвоены числовые (цифровые) значения. В таких случаях говорят о *цифровой (числовой) информации.*

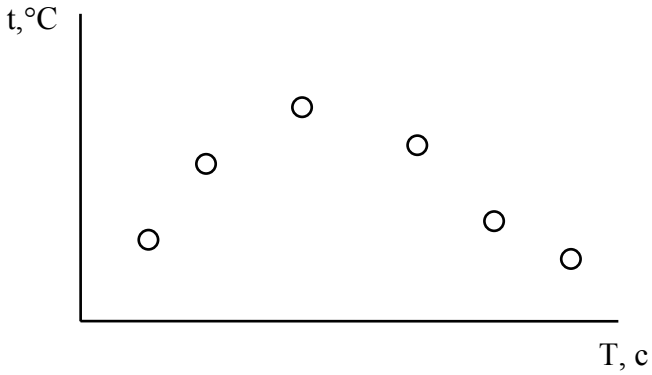


Рис. 1.2. Представление графика изменения физической величины (температуры) во времени при измерении температуры в определённые моменты времени

Передача и преобразования дискретной информации любой формы (например, обычного текста, содержащего обычные буквы и цифры) могут быть сведены к эквивалентным передаче и преобразованиям цифровой информации. Более того, возможно с любой необходимой степенью точности непрерывные сообщения заменять цифровыми путём *квантования* (т.е. разбиения диапазона непрерывного сигнала на отрезки и присвоения им определённого цифрового значения) непрерывного сообщения по уровню и времени (рис. 1. 3). Таким образом, любое сообщение может быть представлено в цифровой форме.

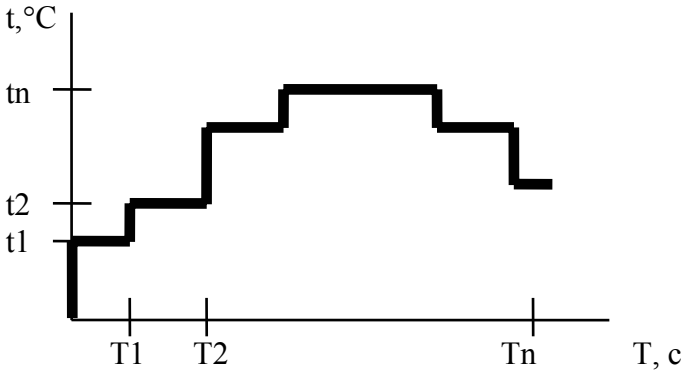
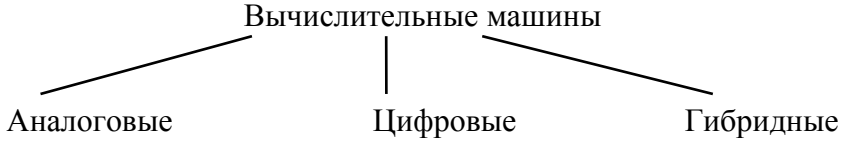


Рис. 1.3. Представление графика изменения физической величины (температуры) после квантования

Электронные вычислительные машины или *компьютеры* являются преобразователями информации. В них исходные данные задачи преобразуются в результат её решения. В соответствии с используемой формой представления информации машины делятся на два класса: *непрерывного действия - аналоговые* и *дискретного действия - цифровые.* Существует также отдельный класс машин, использующих обе формы представления информации и преобразующих одну форму в другую. Такие машины называются *гибридными.* Таким образом, существуют три класса вычислительных машин: аналоговые (АВМ), цифровые (ЦВМ) и гибридные (ГВМ).



В силу универсальности цифровой формы представления информации цифровые вычислительные машины представляют собой наиболее универсальный тип устройства обработки информации. Именно эти машины мы будем изучать в рамках данного курса.

2. Структура ЦВМ

Рассмотрим упрощённую структуру ЦВМ (рис. 2.1). Она содержит следующие основные устройства: *арифметико-логическое устройство (АЛУ), память, управляющее устройство (УУ), устройства ввода и вывода и пульт ручного управления.* Рассмотрим назначение этих устройств.

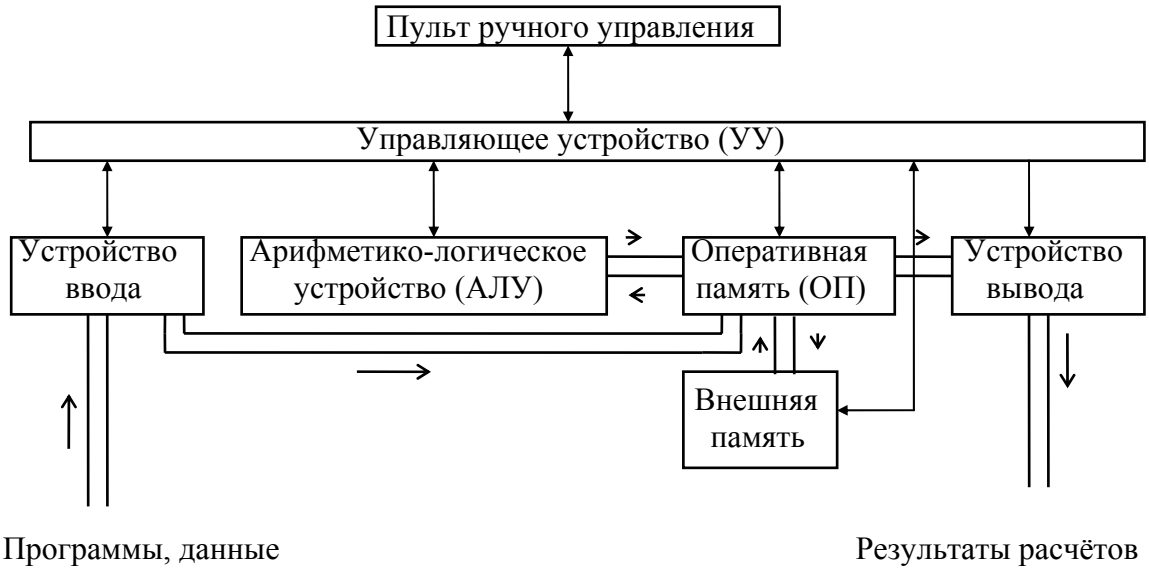


Рис. 2.1. Структура ЦВМ

АЛУ производит арифметические и логические преобразования над поступающими в него *машинными словами,* т.е. кодами определённой длины, представляющими собой числа или другой вид информации. Количество разрядов в машинном слове обычно совпадает с количеством разрядов в основных регистрах АЛУ.

Память хранит информацию, передаваемую из других устройств, в том числе поступающую в машину извне через устройства ввода, и выдаёт во все остальные устройства информацию, необходимую для протекания вычислительного процесса. Память обычно состоит из двух частей: быстродействующей *основной* или *оперативной* памяти и сравнительно медленно действующей, но значительно большей по объёму *внешней* памяти. Содержимое ОП при выключении машины обычно не сохраняется, внешняя память - долговременная.

Управляющее устройство (УУ) автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам сигналы, предписывающие им те или иные действия.

Устройство ввода служит для ввода информации и преобразования её к машинному виду.

Устройство вывода служит для преобразования информации от машинного представления к виду, воспринимаемому человеком или другими потребителями информации, и передачи её этим потребителям.

При помощи пульта ручного управления оператор пускает и останавливает машину, а при необходимости может вмешаться в процесс решения задачи.

Рассмотренную нами модель ЦВМ называют *фон-неймановской*, по имени американского математика Дж. фон Неймана, который впервые в 1945 году предложил принцип построения ЦВМ с программой, хранимой в общей оперативной памяти вместе с данными (такую модель также называют принстонской). Возможно построение машины с отдельной памятью команд и отдельной памятью данных, и, соответственно, двумя шинами (магистралями): шиной команд и шиной данных. Такая модель была реализована в 1944 году в Гарвардском университете (США) и получила название гарвардской.

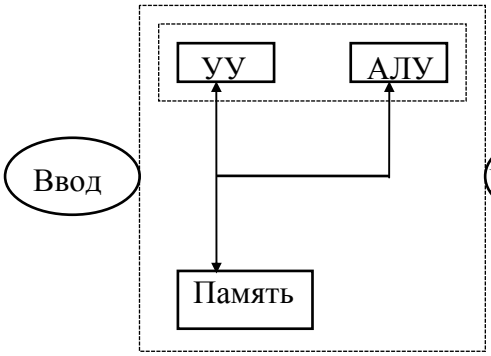


Рис. 2.2. Принстонская модель ЦВМ

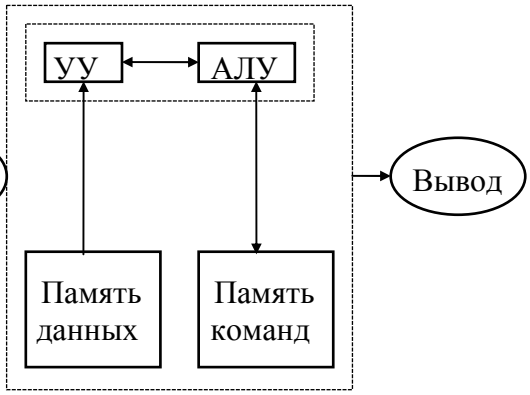


Рис. 2.3. Гарвардская модель ЦВМ

3. Принципы действия ЦВМ

Алгоритм - это конечное число формальных правил, приводящих к выполнению данной задачи за конечное число шагов.

Автоматическое управление процессом решения задачи достигается на основе *принципов программного управления* (принципов фон Неймана). Рассмотрим их:

1. Информация кодируется в двоичной форме и разделяется на единицы (элементы) информации, называемые *словами*.

2. Разнотипные слова различаются по способу использования, но не способом кодирования.

3. Слова информации размещаются в ячейках памяти машины и идентифицируются номерами ячеек, которые называются *адресами.*

4. Алгоритм решения задачи представляется в форме последовательности управляющих слов, которые определяют наименования операций и слова информации, участвующие в этой операции. Эти управляющие слова называются *командами*. Алгоритм, представленный в виде последовательности машинных команд называется *программой.*

5. Выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, определяемом программой.

Рассмотрим базовые принципы построения ЦВМ. На основе принципов программного управления возникает принцип *процессора. Процессор* - это некоторая абстракция, которая определяет совокупность средств, реализующих некоторый информационный процесс. Процессор действует на основе принципов микропрограммного управления, аналогичных принципам программного управления, но на более низком уровне. Принципы микропрограммного управления реализуются с помощью *автоматов* (их устройство будет рассмотрено на одной из следующих лекций).

Таким образом, можно проследить последовательную декомпозицию ЦВМ сверху вниз:

1. Принципы программного управления.
2. Принцип процессора.
3. Принципы микропрограммного управления.
4. Принцип автомата.

4. Аппаратное и программное обеспечение. Архитектура ЦВМ. Структура программного обеспечения

Для придания ЦВМ тех или иных свойств используют средства двух видов: *аппаратные (hardware) и программные (software).* Последние также называются *средствами программного (математического) обеспечения* или просто *программным обеспечением.*

Часть свойств ЦВМ приобретает благодаря наличию в её составе электронного или электромеханического оборудования, специально предназначенного для реализации этих свойств (Например: АЛУ).

Ряд других свойств реализуется без специальных аппаратных средств программным путём. При этом используются имеющиеся аппаратные средства машины, работающие в соответствии с программой, которая обеспечивает выполнение необходимой функции. Например, машина может не иметь аппаратно реализованной операции извлечения квадратного корня, но, если в машине существует программа извлечения квадратного корня на основе тех аппаратных средств, которые имеются, то с точки зрения пользователя машина обладает свойством извлекать квадратный корень.

Обычно функции, реализованные аппаратно, выполняются значительно быстрее, чем те же функции, реализованные программным путём. Поэтому можно в различных ЦВМ встретить различные варианты реализации одних и тех же функций. Зависит это от необходимого быстродействия машины.

Часто для придания ЦВМ того или иного свойства используют комбинацию аппаратных и программных средств. Это позволяет при сравнительно небольших аппаратурных затратах достигать достаточно высокого быстродействия при выполнении заданной функции.

Средства программного и аппаратного обеспечения являются двумя взаимосвязанными компонентами вычислительной техники и в совокупности образуют *вычислительную систему.*

При создании новой ЦВМ разработка аппаратных и программных средств производится одновременно и взаимосвязано.

Сложность современных вычислительных машин закономерно привела к понятию *архитектуры вычислительной машины.* Под архитектурой ЦВМ понимают комплекс общих вопросов её построения, существенных в первую очередь для пользователя, интересующегося главным образом возможностями машины, а не деталями её технического исполнения.

Круг вопросов, связанных с архитектурой ЦВМ, можно условно разделить на:

- вопросы общей структуры;
- вопросы организации вычислительного процесса;
- вопросы общения пользователя с машиной;
- вопросы логической организации представления, хранения и преобразования информации;
- вопросы логической организации совместной работы различных устройств, а также аппаратных и программных средств.

К этим вопросам мы будем постоянно возвращаться в течение нашего курса, а пока перейдём к структуре программного обеспечения вычислительных систем.

Программное обеспечение (ПО) делиться на:

- операционную систему;
- системы программ;
- системы управления файлами (СУФ) или базами данных (СУБД);
- утилиты.

Совокупность системных управляющих и обрабатывающих программ, предназначенная для наиболее эффективного выполнения вычислительных процессов и организации интерфейса пользователя ЦВМ называется *операционной системой (ОС).* Операционная система - это те программы, которые обеспечивают взаимодействие других программ и непосредственно пользователя с аппаратурой ЦВМ. На рис. 4.1 изображена структура программного обеспечения ЦВМ. Обрати-

те внимание, что ни одна из программ не имеет непосредственного контакта с аппаратурой, кроме ОС.

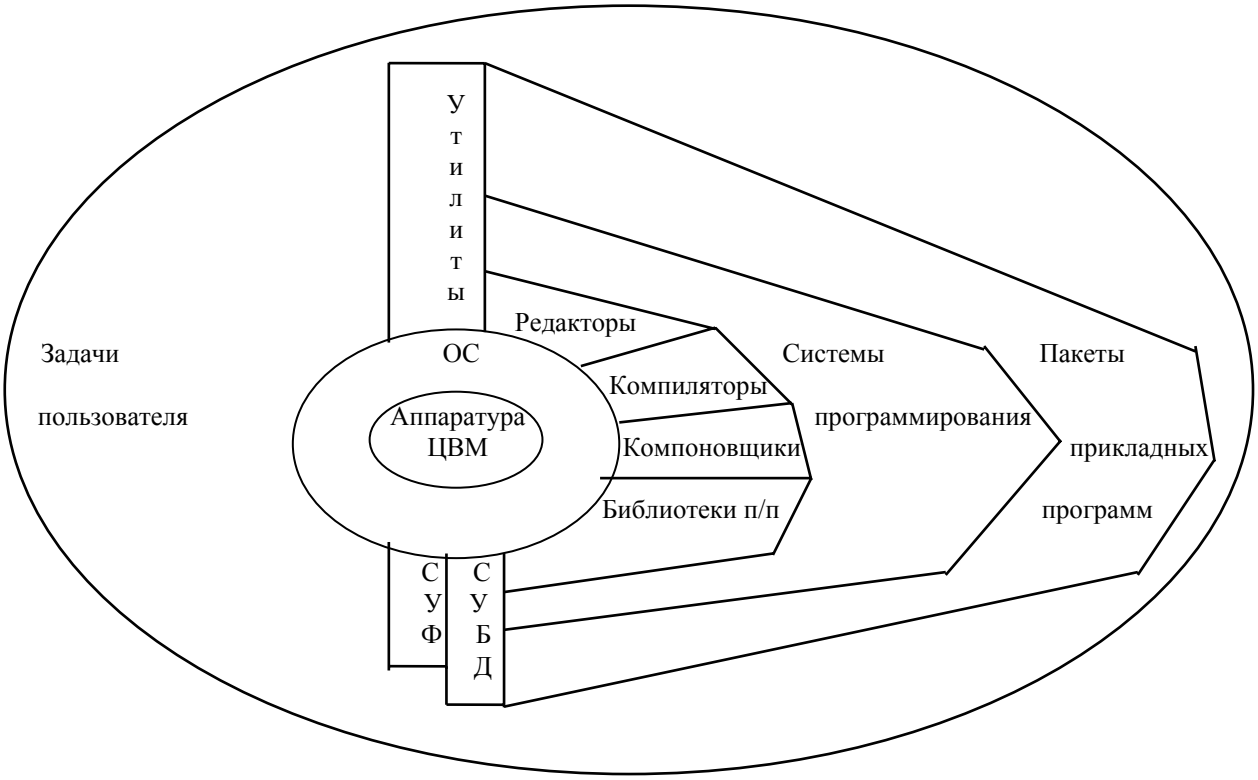


Рис 4.1. Структура ПО ЦВМ

Рассмотрим основные функции ОС:

- загрузка задач пользователя в ОЗУ;
- передача процессору этой задачи и организация мультипрограммирования;
- распределение ресурсов по требованию задачи (планирование, диспетчеризация);
- распределение памяти;
- организация виртуальной памяти (если это необходимо и если позволяет аппаратура);
- защита программ от несанкционированного доступа;
- предоставление различных услуг по восстановлению процесса в результате частичного сбоя.

Операционная система допускает непосредственный контакт с собой задач пользователя и самого пользователя. Этот контакт может осуществляться с помощью *систем управления файлами (СУФ)* или *систем управления базами данных (СУБД),* а также посредством *пакетов прикладных программ.* Операционная система, а именно *супервизор* используя СУФ и СУБД организует вычислительный процесс. Пакеты прикладных программ представляют собой структурированные комплексы, предназначенные для решения определённых достаточно широких классов задач, а также для расширения функций операционных систем (управление базами данных, реализация режимов телеобработки данных, реального времени и т.п.). Прикладные программы реализуются с помощью *систем программирования,* обычно включающих в себя текстовый редактор, компилятор (транслятор) ,компоновщик и пакеты стандартных подпрограмм. Текстовый редактор используется в данном случае для набора текста программ. Компилятор проверяет синтаксическую и лексическую правильность программы и преобразует её в объектный код. Компоновщик преобразует объектный код в машинный, подключая стандартные подпрограммы из библиотек. Любая система программирования ориентирована на свою операционную систему.

Говоря о системах программирования нельзя не сказать несколько слов о языках программирования, которые реализуются с помощью этих систем.

Языки программирования бывают машинно-зависимые и машинно-независимые. К машинно-зависимым языкам относят макроязыки, ассемблеры и машинные языки, т.е. те языки, ко-

манды которых непосредственно зависят от аппаратной реализации ЦВМ. Машинно-независимые языки программирования (языки высокого уровня) делят на проблемно-ориентированные и процедурно-ориентированные. К первому типу относятся PROLOG, LISP , ко второму PASCAL, Си, FORTRAN и другие.

Ещё один вид программ непосредственно общающихся с операционной системой - *утилиты.* Утилитами называют набор специальных системных программ, предназначенных для реализации системных задач (т.е. поддержанию самой ОС), таких как форматирование дисков, проверка поверхностей дисков, оптимальное распределение места на диске, перенос данных с одного носителя на другой и т.п.

Ни одна программа не может выполняться, не используя при этом модуля операционной системы.

5. Представление информации в ЦВМ

5.1. Позиционные системы счисления

Под *системой счисления* понимается способ представления любого числа с помощью некоторого алфавита символов, называемых цифрами. Существуют различные системы счисления. От их особенностей зависят наглядность представления числа при помощи цифр и сложность выполнения арифметических операций.

Наглядность представления чисел и сравнительная простота выполнения арифметических операций характерны для *позиционных систем счисления.*

Система счисления называется позиционной, если одна и та же цифра имеет различное значение, в зависимости от того, какую позицию она занимает в последовательности цифр, изображающих число. Причём, это значение меняется в зависимости от позиции однозначно, по некоторому закону. В качестве примера позиционной системы счисления можно привести используемую в повседневной жизни *десятичную систему*, а в качестве примера непозиционной системы счисления - *римскую систему*.

В вычислительной технике применяются только позиционные системы счисления, поэтому в дальнейшем будем рассматривать именно их.

Количество S различных цифр, употребляемых в позиционной системе счисления, называется её *основанием.* Эти цифры обозначают S целых чисел, обычно 0, 1, ... (S-1). В десятичной системе используются десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; эта система имеет основанием число десять.

В общем случае в системе с основанием S любое число может быть представлено в виде полинома от основания S:

$$X = A_r S^r + A_{r-1} S^{r-1} + ... + A_0 S^0 + A_{-1} S^{-1} + ..., \quad (5.1)$$

где в качестве коэффициентов A_i могут стоять любые из S цифр, используемых в системе счисления. Например, число 10 (десятичная система) можно представить следующим образом:

$$10 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 ,$$

где в качестве коэффициентов a_i стоят цифры 0 и 1, используемые в двоичной системе счисления.

Принято представлять числа в виде соответствующей формуле (5.1) последовательности цифр:

$$X = A_r A_{r-1} A_{r-2} ... A_1 A_0 . A_{-1} A_{-2...}$$

В этой последовательности точка отделяет целую часть числа от дробной (в литературе по вычислительной технике и программированию принято отделять целую часть числа от дроб-

ной точкой, а не запятой), т.е. коэффициенты при положительных степенях, включая ноль, от коэффициентов при отрицательных степенях. Точка опускается, если нет отрицательных степеней. Таким образом, число десять можно представить как 1 0 1 0 в двоичной системе счисления.

В позиционных системах счисления значение каждого разряда больше значения соседнего справа разряда в число раз, равное основанию системы (S).

В ЦВМ применяют позиционные системы счисления с недесятичным основанием: *двоичную, восьмеричную, шестнадцатеричную* и др. В дальнейшем для обозначения используемой системы счисления будем заключать число в скобки и в индексе указывать основание системы счисления.

Наибольшее распространение в ЦВМ имеет *двоичная система счисления*. В этой системе используются только две цифры: 0 и 1.

Для двоичной системы счисления формулу (5.1) можно записать в следующем виде:

$$X = A_m * 2^m + A_{m-1} * 2^{m-1} + ... + A_1 * 2^1 + A_0 * 2^0 + A_{-1} * 2^{-1} + ...,$$

где Ai либо 0, либо 1.

Вернёмся к примеру с числом десять, но на этот раз будем рассматривать число 10.625 (т.е. добавим дробную часть):

$$(1010.101)_2 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 + 1*2^{-1} + 0*2^{-2} + 1*2^{-3} = (10.625)_{10}$$

Двоичное изображение числа требует большего (для многоразрядного числа примерно в 3,3 раза) числа разрядов, чем его десятичное представление. Тем не менее применение двоичной системы счисления удобно для проектирования ЦВМ, так как для представления числа позволяет использовать элементы, имеющие только два устойчивых состояния. Примерами таких элементов являются триггеры, ферромагнитные сердечники и т.д. Другим важным достоинством двоичной системы является простота двоичной арифметики.

В *восьмеричной системе* употребляются восемь цифр: 0, 1, 2, 3, 4, 5, 6, 7. Для восьмеричной системы формула (5.1) выглядит следующим образом:

$$X = A_m * 8^m + A_{m-1} * 8^{m-1} + ... + A_1 * 8^1 + A_0 * 8^0 + A_{-1} * 8^{-1} + ...,$$

где коэффициенты Ai принимают значения от 0 до 7.

Например, представим восьмеричное число 307.05 в десятичном виде:

$$(307.05)_8 = 3*8^2 + 0*8^1 + 7*8^0 + 0*8^{-1} + 5*8^{-2} = (199.078125)_{10}$$

В *шестнадцатеричной системе* для изображения чисел применяются 16 цифр от 0 до 15, при этом вводятся специальные обозначения для цифр, соответствующих числам от 10 до 15. Цифры обозначаются первыми шестью буквами латинского алфавита: 10 - A, 11 - B, 12 - C, 13 - D, 14 - E, 15 - F.

Для того, чтобы вам было удобно работать с различными системами счисления запомните соответствие цифр в различных системах счисления, приведённое в табл. 5.1.

Запишем формулу (5.1) для шестнадцатеричной системы счисления:

$$X = A_m * 16^m + A_{m-1} * 16^{m-1} + ... + A_1 * 16^1 + A_0 * 16^0 + A_{-1} * 16^{-1} + ...,$$

где в качестве коэффициентов A_i выступают числа от 0 до 15.

Переведём шестнадцатеричное число C4A.8 в десятичную систему счисления:

$$(C4A.8)_{16} = 14*16^2 + 4*16^1 + 10*16^0 + 8*16^{-1} = (3658.5)_{10}.$$

Как вы, наверное, заметили мы уже научились переводить числа из различных систем счисления в десятичную. Рассмотрим обратный перевод.

Таблица 5.1

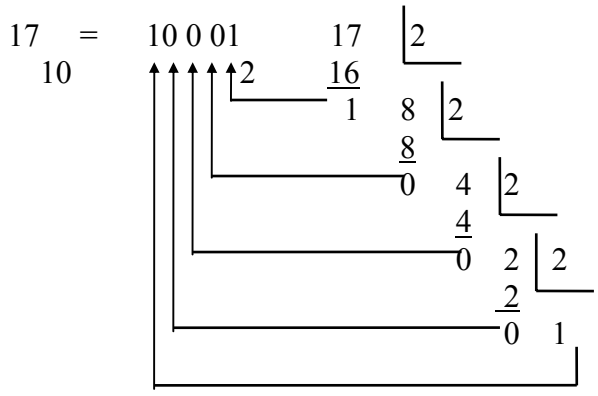
Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

В общем виде, при некратных основаниях, перевод числа из системы с основанием S в систему с основанием S1 сводится к выполнению следующего алгоритма:

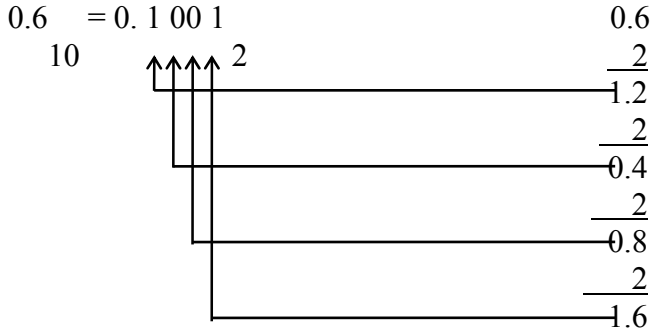
1. Разделение целой и дробной частей числа в системе с основанием S;
2. Последовательное целочисленное деление (деление с остатком) целой части и образующихся целых частных на основание новой системы счисления S1 до тех пор, пока частное не станет меньше этого основания;
3. Запись целой части числа в новой системе счисления, причём в старший разряд записывается последнее частное, а затем последовательно, в порядке обратном делению записываются полученные остатки;
4. Последовательное умножение дробной части числа и получающихся дробных частей произведений на основание новой системы счисления S1 до тех пор, пока не получиться целое произведение, либо не будет получено необходимое количество цифр для записи дроби в системе с основанием S1;
5. Запись дробной части числа в новой системе счисления, причём записываются все полученные целые части произведений так, чтобы последняя целая часть попала в младший разряд, а первая в старший;
6. Объединение целой и дробной части в новой системе счисления.

При выполнении умножения и деления основание новой системы счисления S1 записывается цифрами исходной системы счисления S.

Рассмотрим перевод десятичного числа 17.6 в двоичную систему счисления.



(17.6) = (10001.1001)
10 2



Аналогично производится перевод из десятичной в восьмеричную и шестнадцатеричную системы счисления. Попробуйте самостоятельно перевести число 191.6875 в восьмеричную систему счисления (должно получиться 277.54 в восьмеричной системе счисления) и в шестнадцатеричную систему счисления (должно получиться BF.B).

Существует более простой способ перевода десятичных чисел в двоичную форму, который часто используется на практике. Для перевода десятичного числа X в двоичную форму необходимо найти ближайшую к этому числу меньшую степень двойки (2 в степени n), которая вычитается из числа X, а в старший разряд записывается единица. Далее рассматривается число 2 в степени (n-1), если оно больше полученного остатка, то в следующий (меньший) разряд записывается 0, а если меньше, то оно вычитается из остатка, образуя новый остаток, и в следующий разряд записывается 1. Затем рассматривается степень n-2 и т.д. Процесс продолжается до тех пор, пока в остатке не окажется ноль. Для дробных чисел точка ставится после двойки в нулевой степени, перед отрицательными степенями. Например, переведём число 10.5:

$2^3 = 8$, $8 < 10.5$ записываем 1 в старший разряд и $10.5 - 8 = 2.5$
$2^2 = 4$, $4 > 2.5$ записываем в следующий разряд 0
$2^1 = 2$, $2 < 2.5$ записываем в следующий разряд 1 и $2.5 - 2 = 0.5$
$2^0 = 1$, $1 > 0.5$, записываем в следующий разряд 0,
$2^{-1} = 0.5$, записываем в следующий разряд 1 и $0.5 - 0.5 = 0$

Рассмотрим правила преобразования систем счисления с кратными основаниями на примере двоичной, восьмеричной и шестнадцатеричной систем. Эти правила очень просты, поскольку основания восьмеричной и шестнадцатеричной систем есть целые степени числа два:

$8 = 2^3$, $16 = 2^4$.

Для перевода восьмеричного или шестнадцатеричного числа в двоичную форму достаточно заменить каждую цифру этого числа соответствующим трёхразрядным или четырёхразрядным двоичным числом, при этом отбрасывая ненужные нули в старших разрядах, например:

$(5 \quad 1 \quad 3. \quad 7)_8 = (101001011.111)_2$
101 001 011 111

$(3 \quad E \quad 7. \quad B)_{16} = (1111100111.1011)_2$
0011 1110 0111 1011

Для перехода от двоичной к восьмеричной (или шестнадцатеричной) системе поступают следующим образом: двигаясь от точки влево и вправо, разбивают двоичное число на группы по три (четыре) разряда, дополняя при необходимости нулями крайние левую и правую группы. Затем группу из трёх (четырёх) разрядов заменяют соответствующей восьмеричной (шестнадцатеричной) цифрой. Например, переведём двоичное число 10011101.110 в восьмеричное:

010 011 101 . 110 = $(235.6)_8$,
2 3 5 6

а затем в шестнадцатеричное:

1001 1101 . 1100 = $(9D.C)_{16}$.
9 D C

В настоящее время в большинстве ЦВМ используется двоичная система и двоичный алфавит для представления различной информации, а также при выполнении арифметических и логических операций.

Шестнадцатеричная и восьмеричная системы применяются для более короткой и удобной записи двоичных кодов команд, адресов, операндов и т.д.

Задания для тренировки:

1. AC2F.F
   (16) ⟶ (2) ⟶ (8) ⟶ (10)
   (AC2F.F - 10101100001011111.1111 - 126057.74 - 44079.9375)

2. 156.85
   (10) ⟶ (2) ⟶ (8) ⟶ (16)
   (156.85 - 10011100.11011 - 234.66 - 9C.D8)

3. 113.7
   (8) ⟶ (2) ⟶ (16) ⟶ (10)
   (113.7 - 1001011.111 - 4B.E - 75.875)

4. 10011011.1101
   (2) ⟶ (10) ⟶ (8) ⟶ (16)
   (10011011.1101 - 155.8125 - 233.64 - 9B.D)

5.2 Двоичная арифметика

Правила выполнения арифметических действий над двоичными числами задаются таблицами двоичного сложения, двоичного вычитания и двоичного умножения.

Таблица двоичного сложения	Таблица двоичного вычитания	Таблица двоичного умножения
0 + 0 = 0	0 - 0 = 0	0 * 0 = 0
0 + 1 = 1	1 - 0 = 1	0 * 1 = 0
1 + 0 = 1	1 - 1 = 0	1 * 0 = 0
1 + 1 = 0 плюс единица переноса в старший разряд	10 -1 = 1	1 * 1 = 1

Сложение двоичных чисел. При сложении двоичных чисел в каждом разряде в соответствии с таблицей двоичного сложения производится сложение двух цифр слагаемых или двух этих цифр и 1, если имеется перенос из соседнего младшего разряда. Необходимо помнить, что сумма двух единиц даёт ноль и единицу переноса в старший разряд. В результате получается цифра соответствующего разряда суммы и, возможно, также 1 переноса в старший разряд. Например, необходимо сложить два двоичных числа:

```
  1010.01        (10.25)
+
  1011.10        (11.05)
 10101.11        (21.75)
```

Справа показано сложение тех же чисел, представленных в десятичной системе счисления.

Вычитание двоичных чисел. При вычитании двоичных чисел в данном разряде при необходимости занимается 1 из следующего, старшего разряда. Эта занимаемая единица равна двум единицам данного разряда. Заём производится каждый раз, когда цифра в разряде вычитаемого больше цифры в том же разряде уменьшаемого. Например, если нужно из одного двоичного числа вычесть другое:

```
  10101.11        (21.75)
-
   1010.01        (10.25)
   1011.10        (11.5)
```

Умножение двоичных многоразрядных чисел производится путём образования частичных произведений и последующего их суммирования. В соответствии с таблицей двоичного умножения каждое частичное произведение равно нулю, если в соответствующем разряде множителя стоит 0, или равно множимому, сдвинутому на соответствующее число разрядов влево, если в соответствующем разряде множителя стоит 1. Таким образом, операция умножения сводится к операциям сдвига и сложения. Положение точки определяется так же, как и при умножении десятичных чисел. Например, умножим 1010.1 на 101.01:

```
1010.1 * 101.01 = 110111.001
(10.5)  * (5.25) = (55.125)
```

```
      10101
     *
      10101
      10101
   +
     00000
  +
    10101
 +
    00000
+
   10101
   110111001
```

Деление двоичных чисел. При делении двоичных чисел используются таблицы двоичного умножения и вычитания. Рассмотрим деление двоичного числа 1010.101 на двоичное число 10.1:

1010.101 / 10.1 = 100.01

```
1010101  | 10100
10100      100.01
000010100
   010100
   000000
```

Деление выполняется по тем же правилам, что и деление в десятичной системе счисления.

5.3 Прямой, обратный и дополнительный коды

В ЦВМ в целях упрощения выполнения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака результата операции. Операция вычитания (или алгебраического сложения) сводится к арифметическому сложению кодов, облегчается выработка признаков переполнения разрядной сетки. В результате упрощаются устройства ЦВМ, выполняющие арифметические операции.

Для представления чисел со знаком в ЦВМ применяют прямой, обратный и дополнительный коды.

Общая идея построения кодов такова. Код трактуется как число без знака, а диапазон представляемых кодами чисел без знака разбивается на два поддиапазона. Один из них представляет положительные числа, а другой - отрицательные. Разбиение выполняется таким образом, чтобы принадлежность к поддиапазону определялась максимально просто. Очень удобно формировать коды так, чтобы значение старшего разряда указывало на знак представляемых чисел. Использование такого кодирования позволяет говорить о старшем разряде как о знаковом и об остальных как о цифровых разрядах кода.

Прямой код двоичного числа G, представляемого в n-разрядной сетке, определяется как:

$$G_{пр} = \begin{cases} G, & \text{при } G \geq 0; \\ A + |G|, & \text{при } G \leq 0, \end{cases}$$

где A - величина, равная весу старшего разряда сетки (для дробей A=1, а для целых A=2 в степени n-1). Диапазон представляемых кодом чисел $0 \leq |G| < A$.

Из определения следует, что положительные числа представляются кодами (числами без знака) $0 \le G_{пр} < A$, а отрицательные $A \le G_{пр} < 2A$. Признаком представления положительных чисел является наличие нуля в старшем разряде, называемом знаковым. Наличие в этом разряде 1 говорит о том, что число отрицательное. Цифровые разряды прямого кода содержат модуль представляемого числа, что обеспечивает наглядность представления чисел в прямом коде. Таким образом, прямой код двоичного числа совпадает по изображению с записью самого числа, но в разряде знака стоит 0 или 1 соответственно для положительных или отрицательных чисел. Например, для четырёхразрядной сетки:

0101	(5)
1101	(-5)

Сложение в прямом коде чисел, имеющих одинаковые знаки, производится достаточно просто. Числа складываются, и сумме присваивается знак слагаемых. Однако для операции алгебраического сложения чисел с разными знаками прямой код неудобен. В этом случае приходится определять большее по модулю число, производить вычитание, а затем разности присваивать знак большего по модулю числа.

Операция вычитания (алгебраического сложения) сводится к операции простого арифметического сложения при помощи обратного и дополнительного кодов.

Обратный код двоичного числа G, представляемого в n-разрядной сетке, определяется как

$$G = \begin{cases} G & \text{при } G \ge 0; \\ B - |G| & \text{при } G \le 0, \end{cases}$$

где B - величина наибольшего числа без знака, размещающегося в n-разрядной сетке (для дробей $B = 2 - 2^{-(n-1)}$, а для целых $B = 2^{n-1}$). Диапазон представляемых обратным кодом чисел такой же, как и у прямого кода: $0 \le |G| < A$. По определению обратный код отрицательного числа представляет собой дополнение модуля исходного числа до наибольшего числа без знака, помещающегося в разрядную сетку. В связи с этим получение обратного кода двоичного отрицательного числа сводится к получению инверсии n-разрядного кода модуля этого числа. Так как модуль чисел, представимых в n-разрядной сетке, $|G| < A$, в старшем (знаковом) разряде обратного кода у положительных чисел будет 0, а у отрицательных 1. В цифровых разрядах обратного двоичного кода представляется либо модуль числа (для положительных чисел), либо его инверсия (для отрицательных чисел). Например, для четырёхразрядной сетки:

0101	(5)	0101	(5)
1010	(-5)	+	+
		1101	(-2)
		10010	(3)
		↓ ← 1	
		0011	

Перенос, возникающий из знакового разряда , при использовании обратного кода должен прибавляться в младший разряд суммы. Такой перенос называется круговым или циклическим.

При выполнении расчётов на машине может возникнуть как "положительный", так и "отрицательный" ноль. Положительный ноль в прямом и обратном коде имеет вид:

$(+0)_{пр}$ = 00000.....00;

$(+0)_{обр}$ = 00000.....00.

Отрицательный ноль изображается соответственно для прямого и обратного кода:

$(-0)_{пр}$ = 100000...00;

(-0)обр = 111111...11.

Для того чтобы избежать ситуации возникновения двух нулей применяют дополнительный код.

Дополнительный код двоичного числа G, представляемого в n-разрядной сетке, определяется как

$$G = \begin{cases} G & \text{при } G \geq 0; \\ C - |G| & \text{при } G < 0, \end{cases}$$

где C - величина, равная весу разряда, следующего за старшим разрядом используемой разрядной сетки (для дробей $C=2^0$, для целых чисел $C=2^n$). Диапазон представляемых дополнительным кодом чисел отличается от диапазона прямого или обратного кода. Для положительных и отрицательных чисел поддиапазоны различны. Для положительных чисел, как и у прямого кода $0 \leq G < A$, а для отрицательных $0 < |G| \leq A$. Из определения дополнительного кода следует, что старший (знаковый) разряд у кода положительного числа равен 0, а у кода отрицательного числа 1. В цифровых разрядах дополнительного кода положительного числа представляется модуль этого числа. Дополнительный код отрицательного числа удобно получать через обратный код.

Если рассматривать обратный и дополнительный коды числа как двоичные числа без знаков, то для отрицательных двоичных дробей $Gдоп = Gобр + 2^{-(n-1)}$, а для отрицательных целых чисел $Gдоп = Gобр + 1$.

Таким образом, дополнительный код отрицательного числа может быть получен из обратного путём прибавления 1 к младшему разряду обратного кода. Учитывая, что обратный код отрицательного числа получается путём инверсии модуля этого числа, можно получить дополнительный код отрицательного числа как инверсию модуля этого числа плюс единица в младший разряд. Например, для четырёхразрядной сетки:

0101 (5)
1011 (-5)

Рассмотрим алгебраическое сложение в дополнительном коде. В общем виде:

$A - B = A + (-B)доп = A + \overline{B} + 1$. Например:

```
  0101      (5)
+
  1101      (-3)
 10010      (2).
```

При выполнении алгебраического сложения в дополнительном коде перенос из старшего разряда отбрасывается.

Необходимо помнить, что результат алгебраического сложения слагаемых, представленных в обратном или дополнительном коде, получается в том же коде, что и слагаемые. Для того, чтобы изменить знак числа необходимо:

- в прямом коде изменить знаковый разряд числа на противоположный;
- в обратном коде инвертировать код;
- в дополнительном коде инвертировать код и прибавить 1 в младший разряд.

Признак переполнения разрядной сетки. При алгебраическом сложении двух чисел, помещающихся в разрядную сетку, может возникнуть переполнение, т.е. может образоваться сумма, требующая для своего представления на один цифровой разряд больше по сравнению с разрядной сеткой слагаемых. Можно сформулировать следующее правило (признак) для обнаружения переполнения разрядной сетки.

При алгебраическом сложении двух двоичных чисел с использованием дополнительного (обратного) кода для их представления признаком переполнения разрядной сетки является нали-

чие переноса в знаковый разряд суммы при отсутствии переноса из её знакового разряда (положительное переполнение) или наличие переноса из знакового разряда суммы при отсутствии переноса в её знаковый разряд (отрицательное переполнение). Если и в знаковый, и из знакового разряда суммы есть переносы или нет этих переносов, то переполнение отсутствует. При положительном переполнении результат операции положительный, при отрицательном - отрицательный. Рассмотрим положительное и отрицательное переполнение:

```
  0   101          1   011
+              +
  0   101          1   011
  1<010        1<0   110.
```

Пример отсутствия переполнения, при наличии переносов в знаковый и из знакового разряда представлен при рассмотрении алгебраического сложения в дополнительном коде.

5.4. Формы представления чисел в ЦВМ

В ЦВМ любая информация представляется в виде двоичных кодов (слов) фиксированной или переменной длины. Отдельные элементы двоичного кода , имеющие значение 0 или 1, называют разрядами или битами. Слова часто разбивают на слоги, называемые байтами (8 бит).

Двоичный разряд представляется в ЦВМ некоторым устройством, имеющим два устойчивых состояния, например триггером. Набор соответствующего количества таких устройств служит для представления многоразрядного двоичного числа (слова).

В ЦВМ применяют две формы представления чисел: *с фиксированной точкой* и *с плавающей точкой.* Эти формы называют также соответственно естественной и полулогарифмической.

При представлении чисел с фиксированной точкой положение точки фиксируется в определённом месте относительно разрядов числа. Обычно подразумевается, что точка находится или перед старшим цифровым разрядом, или после младшего. В первом случае могут быть представлены только числа, которые по модулю меньше 1, во втором - только целые числа.

На рис. 5.1 представлены примеры форматов данных для двоичных чисел с фиксированной точкой и соответствующие разрядные сетки для ЦВМ на базе процессора Intel 8080 и выше, и аналогичных процессоров, а также ЕС ЭВМ.

Под разрядной сеткой указана нумерация разрядов, а над - веса этих разрядов.

При представлении числа со знаком для кода знака выделяется "знаковый" разряд (обычно крайний слева). В этом разряде 0 соответствует плюсу, а 1 - минусу.

На рис. 5.1, б показан формат для чисел с точкой, фиксированной перед старшим цифровым разрядом. В этом формате числа (правильные дроби) могут быть представлены с точностью до $2^{-(n-1)}$. Если для представления используется прямой код, то они могут принимать значения в диапазоне:

$$2^{-(n-1)} \le |x| \le 1 - 2^{-(n-1)} \quad .$$

В настоящее время, как правило, форму с фиксированной точкой применяют для представления целых чисел. Используют два варианта представления целых чисел: со знаком и без знака. В последнем случае все разряды разрядной сетки служат для представления модуля числа.

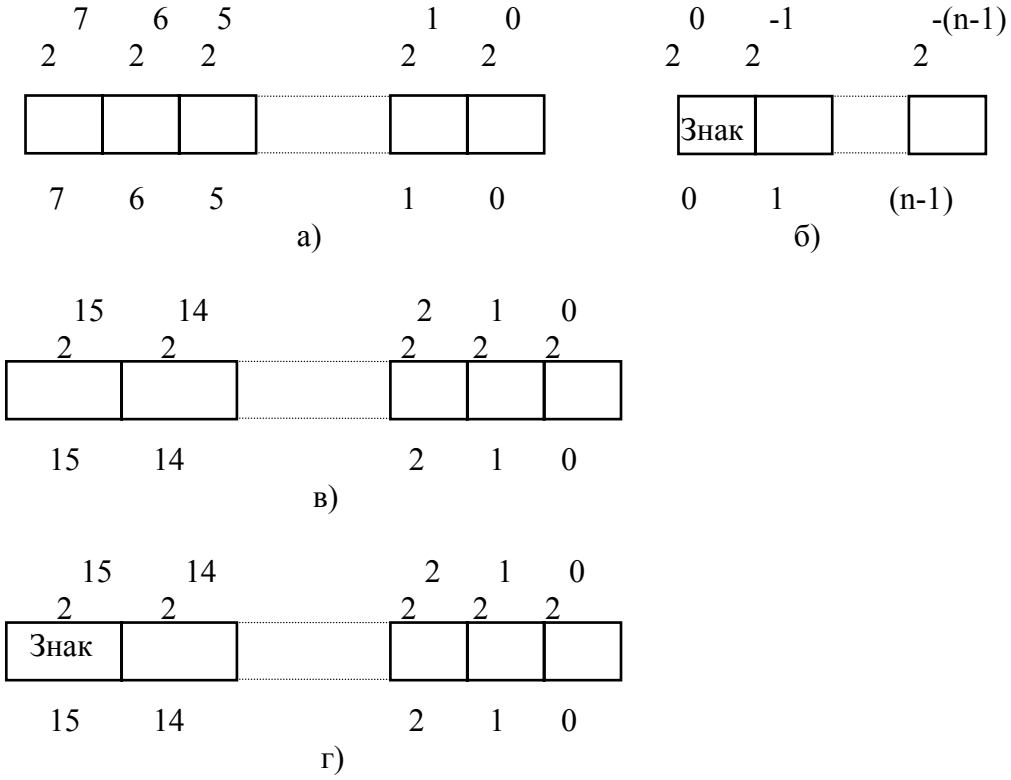


Рис. 5.1. Форматы данных для представления чисел с фиксированной точкой

Большинство ПК оперируют с целыми числами без знака в формате 8- (рис. 5.1, а) и 16-разрядного (рис. 5.1, в) слова, а также числами со знаком в формате 16-разрядного слова (рис. 5.1, г).

Представление чисел с фиксированной точкой используется как основное и единственное лишь в сравнительно небольших по своим вычислительным возможностям машинах, применяемых в системах передачи данных, для управления технологическими процессами и обработки измерительной информации в реальном масштабе времени.

В машинах, предназначенных для решения широкого круга вычислительных задач, основным является представление чисел в формате с плавающей точкой, не требующее масштабирования данных. Однако в таких машинах часто наряду с этой формой представления чисел используется и представление чисел в формате с фиксированной точкой для целых чисел. Это связано с тем, что операции над этими числами занимают меньше времени. К таким операциям относятся операции адресной арифметики.

Представление числа с плавающей точкой в общем случае имеет вид:

$$x = S^p \cdot q ,$$

где q - мантисса числа x; S^p - характеристика числа x; p - порядок; S - основание характеристики (обычно целая степень числа 2).

Мантисса (дробь со знаком) и порядок (целое число со знаком) представляются в системе счисления с основанием, равным S (в соответствующей двоично-кодированной форме). Знак числа совпадает со знаком мантиссы.

Порядок p, который может быть положительным или отрицательным целым числом, определяет положение точки в числе x.

На рис. 5.2 представлены примеры форматов данных для чисел с плавающей точкой в математическом сопроцессоре Intel 8087.

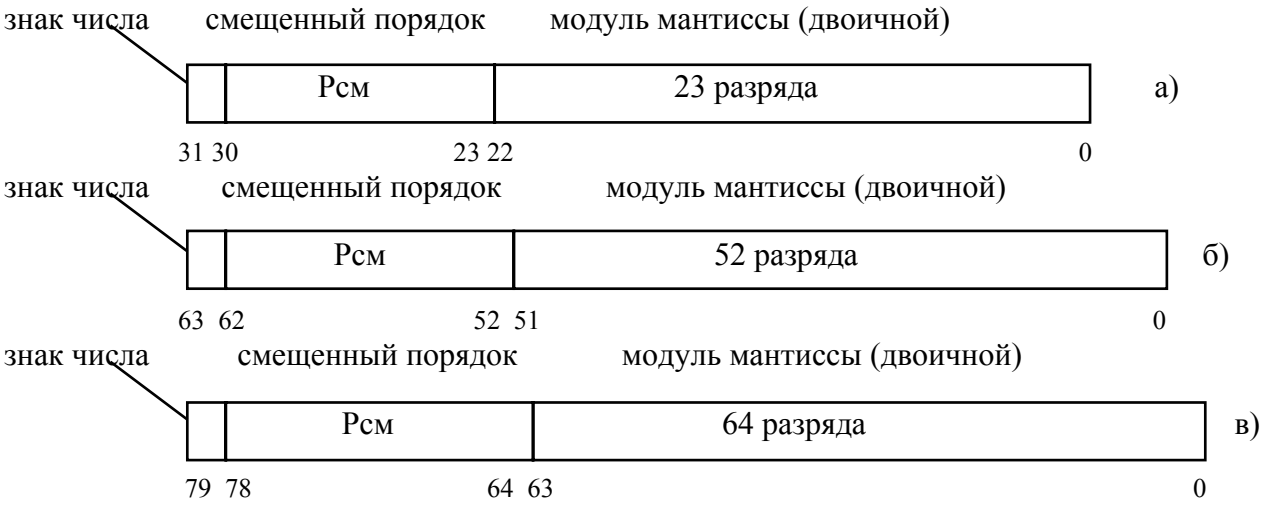


Рис. 5.2. Представление в ЦВМ чисел с плавающей точкой

Одна часть бит формата используется для представления порядка, а другая - мантиссы.

Арифметические действия над числами с плавающей точкой требуют выполнения помимо определённых операций над мантиссами определённых операций над порядками. Для упрощения операций над порядками их сводят к действиям над целыми положительными числами (целыми числами без знаков), применяя для представления порядков смещённый код (представление чисел с плавающей точкой со смещённым порядком).

В случае представления числа с плавающей точкой со смещённым порядком к его порядку p прибавляется целое число - смещение $A=2^k$, где k - число двоичных разрядов, используемых для модуля порядка.

Смещённый порядок Рсм = P + A всегда положителен.

При фиксированном числе разрядов мантиссы любая величина представляется в машине с наибольшей возможной точностью нормализованным числом.

Число $X = S^p * q$ называется *нормализованным,* если мантисса q удовлетворяет условию

$$1 > |q| \geq 1 / S.$$

В процессе вычислений может получаться ненормализованное число. В этом случае машина, если это предписано командой, автоматически нормализует его ("нормализация результата" операции).

Пусть r старших разрядов мантиссы равны 0. Тогда нормализация заключается в сдвиге мантиссы на r разрядов влево и уменьшения порядка на r единиц. При этом в младшие разряды мантиссы записываются нули. При нулевой мантиссе нормализация невозможна.

Диапазон представимых в машине чисел с плавающей точкой зависит от основания системы счисления и числа разрядов, выделенных для изображения порядка. Точность вычислений при плавающей точке определяется числом разрядов мантиссы.

Наличие в ЦВМ нескольких длин форматов для представления чисел позволяет с учётом требований задачи выбирать или короткие форматы для чисел и тем самым экономить память машины и снижать продолжительность операций ЦВМ, или использовать более длинные форматы для получения большей точности.

Для представления чисел с плавающей точкой разработан стандарт, применяемый при разработке микропроцессоров для ПК, в частности фирмой Intel.

По этому стандарту для представления порядка используется смещённый код с отрицательным нулём, а условие нормализованности числа имеет вид $2 > |q| \geq 1$. Мантисса представляется прямым кодом (модуль и знак) со "скрытым" старшим разрядом, имеющим вес 1.

В 4- и 8-байтном словах формат чисел имеет вид, представленный на рис. 5.2, а и б. Используется также и 10-байтное слово для представления результатов промежуточных вычислений (рис. 5.2, в).

Согласно стандарту смещение порядка равно $(2^{(k-1)} - 1)$, где k - число разрядов кода порядка. Код Рсм , содержащий во всех разрядах 1 не используется. Он зарезервирован для указания на переполнение порядка или потерю значимости мантиссы (q = 0), при этом положительное и отрицательное переполнения идентифицируются соответственно положительной и отрицательной мантиссами, содержащими в обоих случаях 1 во всех цифровых разрядах. Указанием на потерю значимости мантиссы служит отрицательность мантиссы с 0 во всех цифровых разрядах. При нулевых кодах порядка и мантиссы представляемое число полагается равным нулю.

5.5. Кодирование десятичных чисел и алфавитно-цифровой информации

Современные ЦВМ обрабатывают не только и не столько числовую информацию, но и символьную. В символьном (текстовом) виде представляется экономическая, организационная, учётная информация, тексты, как на естественных языках, так и на языках программирования и многое другое. Характер этой информации такой, что для её представления требуются слова переменной длины.

Совокупность всех символов, используемых в вычислительной системе, представляет собой её *алфавит*. Символу соответствует машинная единица информации *слог*. Так называют группу двоичных разрядов, служащую для представления символа в машине (двоичный код символа). Применяются различные варианты кодирования символов, использующие коды разной длины.

При выборе способа кодирования необходимо учитывать объём алфавита символов и требования, связанные с облегчением автоматической обработки данных.

Наибольшее распространение получило представление алфавитно-цифровой информации с помощью 8-разрядных слогов, называемых *байтами* (КОИ-8, ASCII, ДКОИ). Существует также представление с помощью 7-разрядных слогов (КОИ-7), а также 5-разрядное кодирование (телеграфное).

Алфавитно-цифровая информация представляется словами переменной длины, содержащими нужное количество байт-символов.

Для упрощения автоматизации обработки данных применяют весовой принцип кодирования символов. Двоичное число, соответствующее коду символа, называется его весом. При весовом кодировании веса кодов цифр последовательно возрастают, а веса кодов букв увеличиваются в алфавитном порядке. Вес кода буквы Б на 1 больше веса кода буквы А и т. д., а код пробела меньше веса кода буквы А.

Такой способ кодирования упрощает сортировку символьной информации, которая производится путем сравнения и упорядочения весов кодов символов.

Наряду с обычной системой кодирования алфавитно-цифровой информации в машинах сохраняют также отдельную систему кодирования для данных, состоящих только из десятичных цифр.

Десятичные цифры 0, 1, ... , 9 представляются в двоично-десятичной форме - коде 8421, в котором десятичная цифра изображается соответствующим 4-разрядным двоичным числом. Не используемые при этом комбинации 4-разрядных кодов (1010-1111) служат для кодирования знаков и служебных символов.

Код 8421 удобен для выполнения машиной (а не вручную) преобразований из десятичной системы в двоичную и обратно. Этот код аддитивен, т. е. сумма представлений двух цифр есть код их суммы. Однако использование этого кода связано с трудностями обнаружения переноса в следующий десятичный разряд и сложностью перехода к обратным и дополнительным кодам для десятичных чисел, облегчающим выполнение алгебраического сложения. Это объясняется тем, что код 8421 не является самодополняющимся, т. е. инверсия его двоичных цифр не даёт кода дополнения десятичной цифры до 9. Поэтому этот код обычно применяется при выполнении операций ввода-вывода.

6. Основы алгебры логики

6.1 Логические функции

В этом разделе мы будем рассматривать двоичное множество B и двоичные переменные, принимающие значения из B. Его элементы часто обозначают 0 и 1, однако они не являются числами в обычном смысле. Наиболее распространённая интерпретация двоичных переменных - логическая: "ДА" - "НЕТ", "Истинно" (И) - "Ложно" (Л). Практически во всех языках программирования присутствуют логические переменные, принимающие значения "True" (Истина) и "False" (Ложь). В общем виде рассматривают множество B={0,1}, где 0 и 1 - формальные символы, не имеющие арифметического смысла.

Алгебра, образованная множеством B вместе со всеми возможными операциями на нём, называется *алгеброй логики. Функцией алгебры логики* (или *логической функцией*) от n переменных называется n-арная операция на множестве B, т.е. логическая функция F (X1, ... , Xn) - это функция , принимающая значения 0 и 1.

Всякая логическая функция n переменных может быть задана таблицей, в левой части которой перечислены все 2^n наборов значений переменных (т.е. двоичных векторов длины n), а в правой части - значения функции на этих наборах. Такая таблица называется таблицей истинности функции F. Например, табл. 6.1 задаёт функцию трёх переменных.

Таблица 6.1

X1	X2	X3	F(X1,X2,X3)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Наборы, на которых функция F=1, называют единичными наборами функции F, а наборы, на которых F=0 - нулевыми наборами функции F.

В таблицах истинности наборы обычно располагают в порядке, который совпадает с порядком возрастания наборов, рассматриваемых как двоичные числа. Функция n переменных полностью определяется вектор-столбцом длины 2^n. Поэтому число различных функций n переменных равно числу различных двоичных векторов длины 2^n, т.е. 2^{2^n} .

Переменная Xi в функции F(X1, ... , Xi-1, Xi, Xi+1, ... , Xn) называется *несущественной* (или *фиктивной*), если

$$F(X1, \dots ,Xi-1, 0, Xi+1, \dots , Xn)= F(X1, \dots ,Xi-1, 1, Xi+1, \dots , Xn)$$

при любых значениях остальных переменных, т.е. если изменение значения Xi в любом наборе значений X1, ... , Xn не меняет значения функции. В этом случае функция F(X1, ... , Xn) по существу зависит от n-1 переменной, т.е. представляет собой функцию G(X1, ... , Xi-1,Xi+1, ... , Xn) от n-1 переменной. Говорят, что функция G получена из функции F удалением фиктивной переменной, а функция F получена из функции G введением фиктивной переменной, причём эти функции по определению считают равными. Например, F(X1,X2,X3)=G(X1,X2) означает, что при любых значениях X1 и X2 F=G независимо от значения X3. Обычно фиктивные переменные удаляют, поскольку они не влияют на значение функции и являются с этой точки зрения лишними. Однако,

иногда возникают ситуации, когда в функцию необходимо ввести фиктивную переменную, например, когда необходимо выровнять число переменных двух функций.

Рассмотрим примеры логических функций. Логических функций одной переменной - четыре; они приведены в табл. 6.2.

Таблица 6.2

x	f0	f1	f2	f3
0	0	0	1	1
1	0	1	0	1

Функции f0 и f3 - константы 0 и 1 соответственно, их значения не зависят от значения переменной, и, следовательно, переменная X для них несущественна. Функция f1 "повторяет" x: f1(x)=x. Функция f2(x) называется *отрицанием x* (или функцией НЕ, инверсией) и обозначается ¬x. Её значение противоположно значению x.

Логических функций двух переменных - 16; они приведены в табл. 6.3.

Таблица 6.3

x1	x2	f0	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Функции f0 и f15 - константы 0 и 1. Функция f1(x1,x2) называется *конъюнкцией* x1 и x2 (или логическим умножением, функцией И); её обозначения: x1&x2, x1∧x2, x1•x2 (во всех случаях знак конъюнкции аналогично умножению часто опускают и пишут x1x2). Конъюнкция равна 1 только если x1 и x2 равны 1, поэтому и появилось название "функция И". Логическим умножением эту функцию называют потому, что её таблица истинности совпадает с таблицей двоичного умножения.

Функция f7(x1,x2) называется *дизъюнкцией* x1 и x2 (или логическим сложением, функцией ИЛИ); её обозначения: x1∨x2, x1+x2. Она равна 1, если хотя бы одна переменная (x1 или x2) равна 1.

Функция f6(x1,x2) - это *сложение по модулю 2*. Её обозначение: x1⊕x2. Она равна 1, когда значения её аргументов различны, и равна 0, когда они равны. Поэтому функцию f6 иногда называют *неравнозначностью*.

Большая часть оставшихся функций не имеет названий и обычно не используется, а если и используется, то легко выражается через функции, которые мы рассмотрели. Перечислим названия тех функций, которые их имеют:

f9 - *эквивалентность* или *равнозначность* (x1~x2, x1≡x2);

f13 - *импликация* (x1→x2, x1⊃x2);

f8 - *стрелка Пирса* (x1↓x2);

f14 - *штрих Шеффера* (x1 | x2).

Суперпозицией функций f1, ... , fm называется функция f , полученная с помощью подстановок этих функций друг в друга и переименования переменных, а *формулой* называется выражение, описывающее эту суперпозицию. Всякая формула, выражающая функцию f как суперпозицию других функций, задаёт способ её вычисления (при условии, что известно, как вычислить исходные функции). Этот способ определяется следующим правилом: формулу можно вычислить, только если уже вычислены значения её подформул. Например, на наборе x1=1, x2=1, x3=0, формула (x3 ∨ x1)⊕(x1&(x1⊕x2)) вычисляется следующим образом: x3∨x1=1; x1(x1⊕x2)=x1&0=0; (x3∨x1)⊕(x1(x1⊕x2))=1⊕0=1. Для вычисления использовалась табл. 6.3.

Таким образом, формула каждому набору значений аргументов ставит в соответствие значение функции и, следовательно, может служить наряду с таблицей истинности способом задания и вычисления функции. В частности, по формуле, вычисляя её на всех 2^n наборах, можно

восстановить таблицу истинности функции. О формуле, задающей функцию, говорят также, что она *реализует* или *представляет* эту функцию.

В отличие от табличного задания представление данной функции формулой не единственно. Например, если в качестве исходного множества функций зафиксировать множество (f1,f7,f2), т.е. функции И, ИЛИ, НЕ, то функцию штрих Шеффера можно представит формулами

$$\neg x1 \vee \neg x2 \text{ и } \neg(x1 x2),$$

а функцию стрелка Пирса - формулами

$$\neg x1 \neg x2 \text{ и } \neg(x1 \vee x2).$$

Формулы, представляющие одну и ту же функцию, называются *эквивалентными* или *равносильными*. Эквивалентность формул обозначается знаком равенства; поэтому можно записать $f14(x1,x2)= \neg x1 \vee \neg x2 = \neg(x1x2)$. Существует стандартный метод проверки эквивалентности формул: по каждой формуле восстанавливается таблица истинности функции, а затем полученные две таблицы сравниваются. Иначе говоря, для каждого набора значений переменных проверяется, равны ли на нём значения формул. Этот метод требует 2•2^n вычислений (если считать, что обе формулы зависят от n переменных) и на практике оказывается слишком громоздким. Существуют и другие методы, которые мы рассмотрим в дальнейшем.

6.2 Булева алгебра

В дальнейшем мы будем рассматривать представление логических функций в виде суперпозиций функций И, ИЛИ, НЕ.

Рассмотрим разложение функций по переменным. Всякая логическая функция может быть представлена в виде:

$f(x1, \dots , xn)=\neg x1 f(0,x2, \dots , xn) \vee x1 f(1, x2, \dots , xn)$.

Это разложение функции по одной переменной. Ясно, что это разложение справедливо для любой из n переменных. Разложить функцию можно не только по одной, но и по нескольким переменным. Например, разложим функцию четырёх переменных по двум из них:

$f(x1,x2,x3,x4)=\neg x1 \neg x2 f(0,0,x3,x4) \vee \neg x1 x2 f(0,1,x3,x4) \vee x1 \neg x2 f(1,0,x3,x4) \vee x1 x2 f(1,1,x3,x4)$.

Справедливость такого разложения доказывается путём подстановки в обе части равенства некоторого произвольного набора всех переменных. В правой части в 1 обратится только та конъюнкция вынесенных переменных, которой соответствует функция, где на месте вынесенных переменных стоят значения, соответствующие данному набору. Все остальные конъюнкции равны 0. Поэтому получим $f(p1, \dots ,pn)=1 \& f(p1, \dots , pn)$, где p1, ... , pn - произвольный набор.

Частным, но наиболее важным случаем является разложение по всем n переменным. При этом все переменные в правой части получают фиксированные значения и функции в конъюнкциях правой части становятся равными 0 или 1, что даёт:

$$f(x1, \dots , xn) = \bigvee_{f(p1, \dots ,pn)=1} y1 \dots yn,$$

где y1, ... ,yn - вынесенные переменные, а дизъюнкция берётся по всем наборам (p1, ... , pn), на которых f=1. Такое разложение называется совершенной дизъюнктивной нормальной формой (СДНФ) функции f. СДНФ функции f содержит столько конъюнкций, сколько 1 в таблице истинности функции f; каждому единичному набору (p1, ... , pn) соответствует конъюнкция всех переменных, в которой xi взято с отрицанием, если pi=0, и без отрицания, если pi=1. Таким образом, существует взаимно однозначное соответствие между таблицей истинности функции f(x1, ... ,xn) и её СДНФ, и, следовательно, СДНФ для всякой логической функции единственна (учитывая, что формулы, получаемые перестановкой конъюнкций и букв в конъюнкции, не различаются и считаются одной и той же СДНФ).

Единственная функция, не имеющая СДНФ - константа 0.

В дальнейшем формулы, содержащие, кроме переменных и скобок, только знаки дизъюнкции, конъюнкции и отрицания, будем называть *булевыми формулами*.

Из вышесказанного следует, что *всякая логическая функция может быть представлена булевой формулой,* т.е. как суперпозиция конъюнкции, дизъюнкции и отрицания.

Действительно, для всякой функции, кроме константы 0, таким представлением может служить СДНФ, а константа ноль может быть представлена как x¬x.

Рассмотрим составление СДНФ на примере функции трёх переменных f0, которая задана с помощью таблицы истинности (табл. 6.4).

Таблица 6.4

x1	x2	x3	f0	f1	f2	f3	f4
0	0	0	0	0	1	0	1
0	0	1	1	0	0	1	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	0	1
1	0	0	1	0	0	1	1
1	0	1	1	0	0	1	0
1	1	0	0	1	1	1	0
1	1	1	0	1	1	0	1

Составим конъюнкции, соответствующие всем единичным наборам и объединим их с помощью дизъюнкции:

$f0(x1,x2,x3) = \neg x1\neg x2x3 \vee x1\neg x2\neg x3 \vee x1\neg x2x3.$

Попробуйте самостоятельно для функций f1,f2,f3,f4 (табл. 6.4) определить содержат ли они фиктивные переменные, и если содержат, то какие, и составить СДНФ для каждой из этих функций.

Алгебра, основным множеством которой является всё множество логических функций, а операциями - дизъюнкция, конъюнкция и отрицание, называется *булевой алгеброй логических функций.* Операции булевой алгебры также часто называют булевыми операциями.

Рассмотрим теперь основные свойства булевых операций.

Ассоциативность:

а) $x1(x2x3) = (x1x2)x3$; б) $(x1\vee x2)\vee x3 = x1\vee(x2\vee x3)$.

Коммутативность:

а) $x1x2 = x2x1$; б) $x1\vee x2 = x2\vee x1$.

Дистрибутивность конъюнкции относительно дизъюнкции:

$x1(x2\vee x3) = x1x2 \vee x1x3.$

Дистрибутивность дизъюнкции относительно конъюнкции:

$x1 \vee (x2x3) = (x1\vee x2)(x1\vee x3).$

Идемпотентность:

а) $xx = x$; б) $x \vee x = x$.

Двойное отрицание:

$$\neg\neg x = x.$$

Свойства констант:

а) $x\&1 = x$; б) $x\&0 = 0$; в) $x \vee 1 = 1$; г) $x \vee 0 = x$; д) $\neg 0 = 1$; е) $\neg 1 = 0$.

Правила де Моргана:

а) $\neg(x1x2) = \neg x1 \vee \neg x2$; б) $\neg(x1 \vee x2) = \neg x1 \neg x2$.

Закон противоречия:

$$x\neg x = 0.$$

Закон "исключённого третьего":

$$x \vee \neg x = 1.$$

Все указанные соотношения можно проверить вычислением обеих частей равенств на всех наборах значений переменных. Ясно, что результат вычисления не зависит от того, как получены значения переменных, входящих в эти равенства, т.е. от того, являются ли эти переменные независимыми или, в свою очередь, получены в результате каких-то вычислений. Поэтому вышеприведённые равенства остаются справедливыми при подстановке вместо переменных любых логических функций и, следовательно, любых формул, представляющих эти функции. Важно лишь соблюдать следующее *правило подстановки формулы вместо переменной:* при подстановке формулы F вместо переменной x все вхождения переменной x в исходное соотношение должны быть *одновременно* заменены формулой F.

Для того, чтобы упростить формулы (т.е. получить эквивалентные формулы, содержащие меньше символов) часто используют следующие правила.

а) Поглощение:

$$x \vee xy = x;$$

$$x\,(x \vee y\,) = x.$$

Доказательство: $x \vee xy = x \,\&\, 1 \vee xy = x(\,1 \vee y\,) = x \,\&\, 1 = x;$

$x(x \vee y\,) = xx \vee xy = x \vee xy = x.$

б) Склеивание:

$$xy \vee x\neg y = x$$

Доказательство: $xy \vee x\neg y = x\,(y \vee \neg y) = x \,\&\, 1 = x.$

Например:

$$\neg xyz \vee x\neg yz \vee \neg x\neg yz \vee xyz = \neg yz \vee\ yz = z.$$

Часто требуется привести формулу к СДНФ, чтобы построить таблицу истинности. Для этого формулу приводят сначала к дизъюнктивной нормальной форме, которая может быть не единственной. *Дизъюнктивной нормальной формой* (ДНФ) называется формула, имеющая вид дизъюнкций элементарных конъюнкций, т.е. конъюнкций переменных или их отрицаний, в которых каждая переменная встречается не более одного раза.

Приведение к ДНФ производится следующим образом. Сначала с помощью свойства двойного отрицания и правил де Моргана все отрицания "спускаются" до переменных. Затем раскрываются скобки, с помощью свойства идемпотентности и законов противоречия и "исключён-

ного третьего" удаляются лишние конъюнкции и повторения переменных в конъюнкциях, а с помощью свойств констант удаляются константы. Например:

$\neg x(y \vee \neg z)(\neg y \vee \neg(xz)) = (\neg xy \vee \neg x\neg z)(\neg y \vee \neg x \vee \neg z) =$

$= \neg xy\neg y \vee \neg x\neg xy \vee \neg xy\neg z \vee \neg x\neg z\neg y \vee \neg x\neg z\neg x \vee \neg x\neg z\neg z =$

$= 0 \vee \neg xy \vee \neg xy\neg z \vee \neg x\neg y\neg z \vee \neg x\neg z \vee \neg x\neg z = \neg xy \vee \neg x\neg z.$

Всякую ДНФ можно привести к СДНФ расщеплением конъюнкций, которые содержат не все переменные, например:

$\neg xy \vee \neg x\neg z = \neg xyz \vee \neg xy\neg z \vee \neg xy\neg z \vee \neg x\neg y\neg z.$

С помощью СДНФ легко строится таблица истинности функции.

Примеры для тренировки.

Составить СДНФ и таблицу истинности для следующих функций:

1) $f1 = \neg xy \vee \neg(x(\neg yz \vee \neg(zx)) \vee z\neg y);$
2) $f2 = y\,\neg(xz \vee zx) \vee yx;$
3) $f3 = zx \vee \neg(x(zx \vee \neg(x\,(y \vee zx)))).$

6.3. Построение логических схем

Булева алгебра широко применяется при построении различного рода цифровых устройств, в том числе и вычислительных машин. Эти цифровые устройства строятся из так называемых логических элементов.

Логический элемент - это электронная схема (устройство), реализующая ту или иную функцию алгебры логики. Наиболее часто используются элементы, построенные на основе булевой алгебры и реализующие булевы функции, такие как: И (рис. 6.1,а), ИЛИ (рис. 6.1,б), НЕ (рис. 6.1,в), И-НЕ (рис. 6.1,г), ИЛИ-НЕ (рис. 6.1,д).

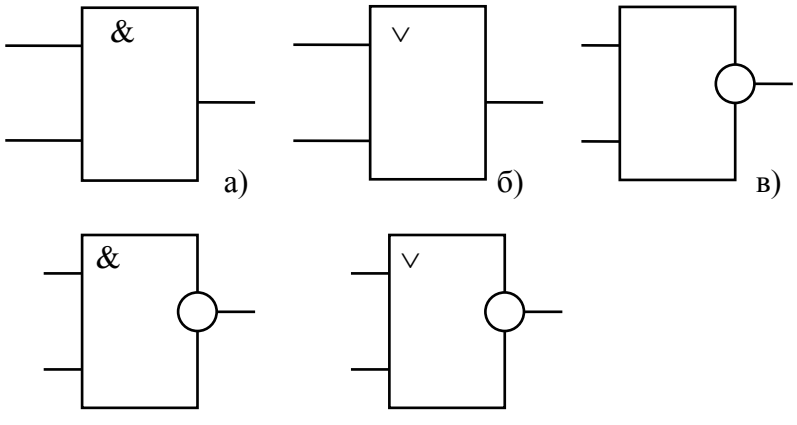


Рис. 6.1. Типы логических элементов

Такие элементы обычно имеют два, три, четыре или восемь входов. На рисунке представлены двухвходовые элементы, реализующие функции двух переменных. Предположим, что нам необходимо реализовать следующую функцию:

$$f(x,y,z) = \overline{(x \vee y) \,\&\, (z \vee x)} \vee \; (x \,\&\, y \,\&\, z).$$

Для реализации этой функции нам необходим один трёхвходовый элемент И {A = (x & y & z)}, два двухвходовых элемента ИЛИ {B = (x ∨ y) и C = (z ∨ x)}, один двухвходовый элемент И-НЕ {D = $\overline{B \,\&\, C}$} и ещё один двухвходовый элемент ИЛИ {f(x,y,z) = D ∨ A}. Схема, реализующая эту функцию, представлена на рис. 6.2. Вариантов логических схем может быть множество, но обычно пытаются минимизировать аппаратные затраты, т.е. количество логических элементов, или строят схемы на определённых типах логических элементов. Для этого приводят исходную функцию к виду, соответствующему заданным параметрам, и по новой формуле строят логическую схему. Логические схемы также часто называют комбинационными схемами, так как совокупность выходных сигналов КС в любой момент времени однозначно определяется комбинацией входных сигналов.

6.4. Конечные автоматы

Следующий класс устройств, используемых при построении вычислительных систем, называется цифровыми автоматами. Цифровой автомат в отличие от комбинационной схемы имеет некоторое конечное число различных внутренних состояний. Переход из состояния в состояние осуществляется под воздействием входного слова. Выходное слово автоматы могут выдавать при переходе из состояния в состояние (автомат Мили) или находясь в определённом состоянии (автомат Мура). В общем случае выходное слово зависит от входного слова и от того состояния, в котором находится или находился автомат. Таким же образом определяется состояние, в которое переходит автомат под воздействием входного слова. Из сказанного можно сделать вывод, что цифровой автомат обладает памятью. В общем виде цифровой автомат описывается с помощью конечного автомата.

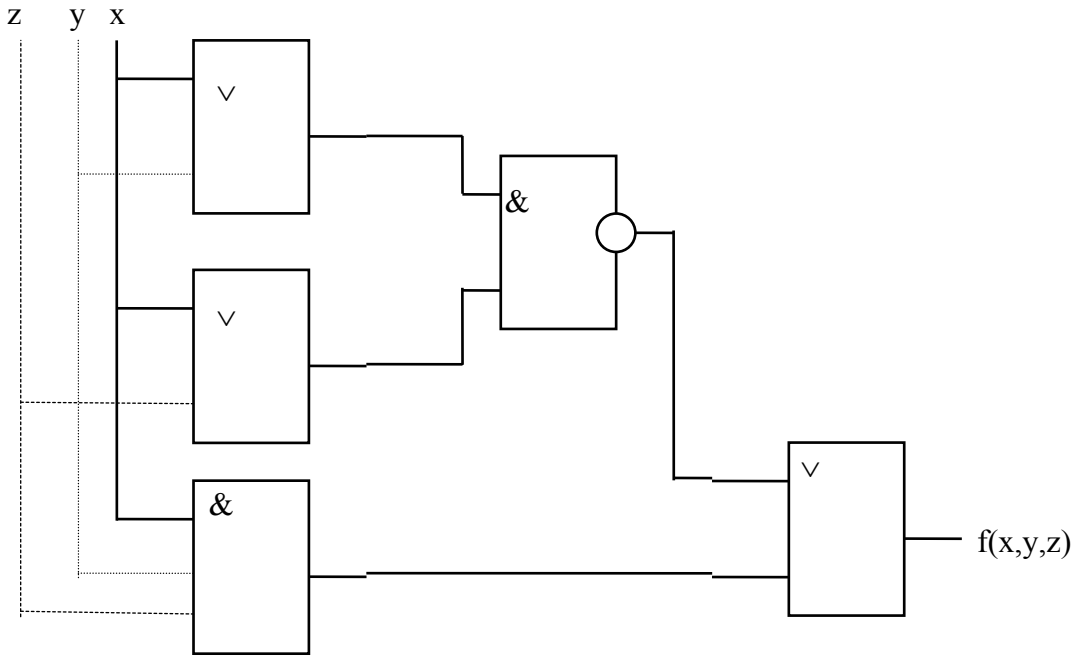


Рис. 6.2. Пример логической схемы

Конечным автоматом называется система S={A,Q,V,δ,λ}, в которой A={a1,...,am}, Q={q1,...,qn}, V={v1,...,vk} - конечные множества (алфавиты), а δ: Q×A→Q и λ: Q×A→V - функ-

ции, определённые на этих множествах. A называется входным алфавитом, V - выходным алфавитом, Q - алфавитом состояний, δ - функцией переходов, λ - функцией выходов. Если, кроме того, в автомате S выделено одно состояние, называемое начальным (обычно считается, что это q1), то полученный автомат называют *инициальным* и обозначают (S,q); таким образом, по неинициальному автомату с n состояниями можно n различными способами определить инициальный автомат.

Поскольку функции δ и λ определены на конечных множествах, их можно задавать таблицами. Обычно эти таблицы называют соответственно таблицами переходов и выходов, а иногда объединяют в одну таблицу переходов-выходов. Например, для автомата с алфавитами A={a1,a2,a3}, Q={q1,q2,q3}, V={v1,v2} таблица может выглядеть следующим образом:

Таблица 6.3

	a1	a2	a3
q1	q3/v1	q2/v2	q3/v2
q2	q1/v1	q2/v1	q1/v1
q3	q2/v1	q3/v2	q1/v2

Ещё один распространённый и наглядный способ представления (задания) автомата - ориентированный мультиграф, называемый графом переходов или диаграммой переходов. Вершины графа соответствуют состояниям, а рёбра переходам между ними, причём, если из вершины qi в вершину qj ведёт ребро, то оно соответствует функции перехода и функции выхода, и над ним записываются соответстветствующие входной и выходной сигнал. Такой граф соответствует автомату Мили. Пример для автомата с представленной таблицей переходов-выходов изображён на рис. 6.4.

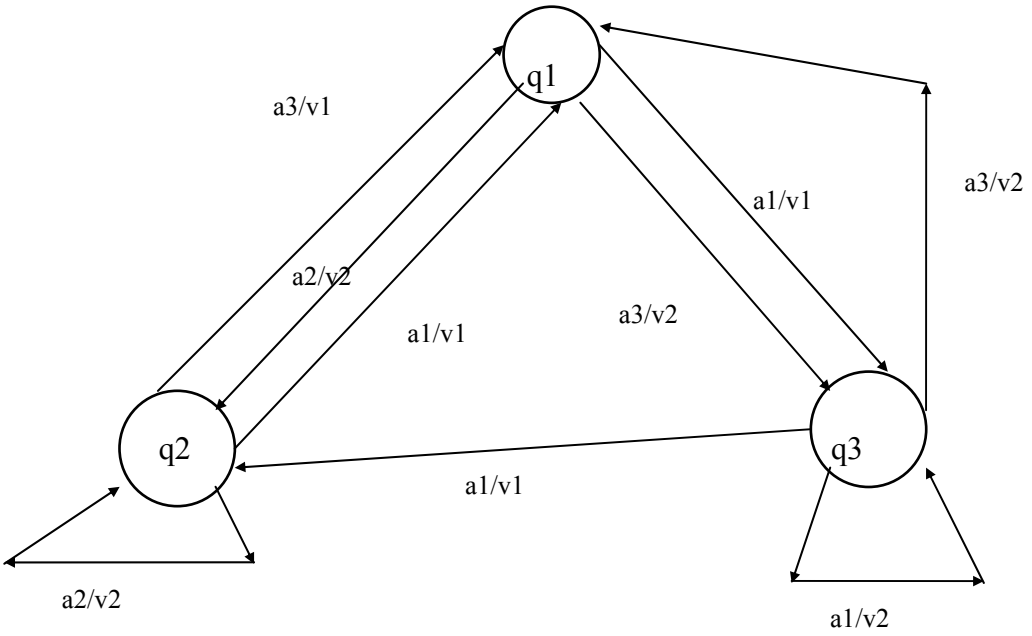


Рис. 6.4. Пример автомата Мили, представленного в виде графа

При изображении в виде графа автомата Мура выходной сигнал записывается под обозначением состояния, а на рёбрах записывается только входной сигнал, т.е. автомат Мура вырабатывает выходной сигнал при нахождении в определённом состоянии независимо от входного сигнала, а не при переходе из состояния в состояние, как автомат Мили. Пример автомата Мура изображён на рис. 6.5. В приведённом примере считается, что состояниям q1 и q2 соответствует выходной сигнал v1, а состоянию q3 - выходной сигнал v2.

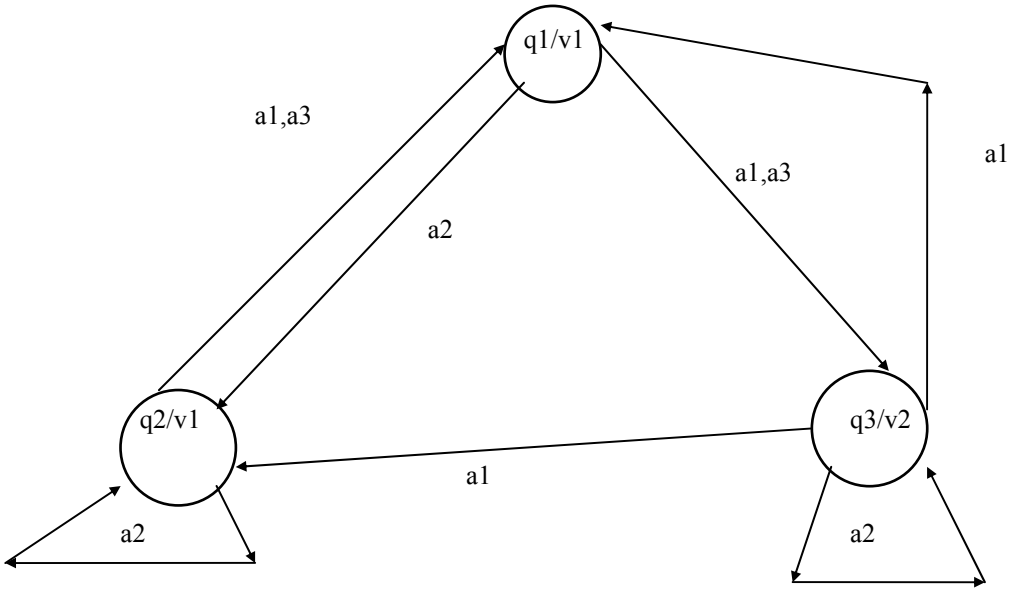


Рис. 6.4. Пример автомата Мура

Список вопросов к промежуточному зачёту по курсу "Введение в информационные технологии"

1. ЭВМ: цифровые, аналоговые, гибридные.
2. Структура ЦВМ.
3. Принципы действия ЦВМ.
4. Аппаратное и программное обеспечение. Архитектура ЦВМ.
5. Структура программного обеспечения. Основные функции операционной системы.
6. Системы счисления. Позиционные системы счисления. Общий случай.
7. Системы счисления с кратными основаниями. Двоичная, восьмеричная, шестнадцатеричная.
8. Перевод из одной системы счисления в другую. Общий случай. Пример.
9. Перевод из двоичной системы счисления в десятичную и обратно. Пример.
10. Перевод из двоичной системы счисления в восьмеричную и шестнадцатеричную и обратно. Пример.
11. Двоичная арифметика. Сложение, вычитание, умножение, деление.
12. Прямой и обратный код. Операция вычитания в кодах.
13. Дополнительный код. Вычитание в дополнительном коде. Признак переполнения разрядной сетки.
14. Формы представления чисел в ЦВМ. Формат с фиксированной точкой.
15. Формы представления чисел в ЦВМ. Формат с плавающей точкой. Нормализация.
16. Кодирование десятичных чисел и алфавитно-цифровой информации.
17. Алгебра логики. Основные понятия. Фиктивная переменная.
18. Логические функции одной и двух переменных.
19. Булева алгебра. Основные понятия. СДНФ.
20. Основные свойства булевых операций.
21. Правила упрощения логических формул. Приведение логической формулы к СДНФ.

22.Базисы Пирса и Шеффера. Построение логических схем.
23.Конечные автоматы.

В билет включаются три вопроса. Два вопроса - теоретические из приведённого списка. Третий вопрос практический, по следующим темам:

- перевод из системы с основанием S в систему с основанием S1;
- представление числа в виде заданного кода;
- арифметические операции в кодах;
- построение СДНФ по таблице истинности;
- приведение логической формулы к СДНФ;
- построение логической схемы.