МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ »

ОСНОВЫ ПОСТРОЕНИЕ ЭВМ

Учебное пособие по изучению курса «Архитектура ЭВМ и основы программирование на ассемблере», для студентов направления подготовки «прикладная математика и информатика».

Махачкала-2021

УДК.681.3.06

Печатается согласно постановлению Совета Дагестанского государственного технического университета протокол №2. от 25.12.2021г. рег.6864.

Учебное пособие предназначено для изучения дисциплины «Архитектура ЭВМ и основы программирование на ассемблере» для студентов специальности 010400.62- «Прикладная математика и информатика», для всех профилей. Махачкала: ДГТУ, 2021, с.

Канаев М.М.

Учебное пособие предназначено для самостоятельной работы студентов, при изучении дисциплины «Архитектура ЭВМ и основы программирование на ассемблере», по разделу основы построение ЗВМ.

В учебном пособии рассмотрены математические и логические основы построение ЭВМ. Подробно рассмотрены внутреннее представление числовой информации в компьютере, принципы работы основных узлов и блоков ЭВМ, а также интегральная схемотехника.

Составитель: Канаев М.М.,

Рецензенты: д.т.н. проф. Мелехин В.Б.
к.т.н., проф. Курбанмагомедов К.Д. зав.каф. ИТ, МОГУ.

Оглавления

Введение

При изучении любой дисциплины нельзя забывать об историческом аспекте. Знание истории предмета расширяет наши представления о нем и позволяет лучше понять, с чего все начиналось, и в каком направлении идет развитие.

Основные этапы развития вычислительной техники можно поставить в следующие хронологические рамки:

1. Домеханический – с 30-40-го тысячелетия до н.э.
2. Механический – с середины XVII века.
3. Электромеханический – с 90-х годов XIX века.
4. Электронный – со второй половины 40-х годов XX века.

На первом этапе использовались простейшие приспособления: счеты, палочки, расчетные доски (абака). Механический этап продолжался от изобретения машины Шиккарда (1624 г.) и суммирующей машины Паскаля (1642 г.) до создания электромеханического табулятора Голлерита (1887 г.). В этот период над созданием механических вычислительных устройств работали много специалистов во всем мире.

В 1843 году Чарльз Бэбидж высказал идею программного управления для построения устройства, автоматически выполняющего арифметические вычисления. Вместе с Ч. Бэбиджем работала Августа Ада Лавлейс (дочь английского поэта Д.Г. Байрона), которую считают первой программисткой мира.

Первая ЭВМ, с начала эксплуатация которой начинается отсчет времени электронной цифровой вычислительной техники, вступила в строй в 1945 г.

Для реализации, разработанной Дж. Маучли ЭВМ ЭНИАК было использовано 18000 электронных ламп и 1500 электромеханических реле. Электронные лампы стали элементной базой машин первого поколения. Идея программного управления вычислительным процессом была существенно развита математиком Дж. фон Нейманом, который в 1945 году сформулировал принцип хранимой в памяти программы. Первой машиной построенной на этих принципах считается ЭВМ ЭДВАК, построенная в 1949 году в университете шт. Пенсильвании.

Первые проекты отечественных ЭВМ были предложены С.А. Лебедевым и Б.И. Рамеевым в 1948 году. В 1951 году была построена первая малая электронная счетная машина(МЭСМ). В 1952 году была закончена разработка БЭСМ-1 (большая электронная счетная машина). В то время это была одна из лучших серийно выпускаемых машин в мире. Она содержала 5000 ламп и могла выполнять до 10 тыс. операций в секунду.

Наиболее совершенная ЭВМ первого поколения М-20 была запущена в производство в 1958 году. Она имела быстродействие до 20 тыс. операций в секунду.

С появлением транзисторов в середине 50-х годов на смену ЭВМ первого поколения пришли ЭВМ второго поколения, построенные на дискретных полупроводниковых приборах. Полупроводниковая схемотехника позволила создать новые схемы вычислительных машин, с

меньшей мощностью потребления, меньшими габаритами и превосходящие ламповые схемы по быстродействию и надежности.

В нашей стране были созданы ЭВМ второго поколения для различного применения: малые ЭВМ серий «Наири» и «Мир», средние ЭВМ – Минск, Раздан, БЭСМ-4, М220 и лучшая из машин второго поколения БЭСМ-6, быстродействие которой составляло 1 млн. операций в секунду. Почти для всех этих машин было характерно применение индивидуальной схемотехники.

С появлением интегральных микросхем произошла революция в вычислительной технике: уменьшились не только габариты, масса и потребляемая мощность ЭВМ, но и улучшились такие параметры как: надежность, быстродействие, стоимость и т.д.

ЭВМ третьего поколения появились во второй половине 60-х годов, когда фирма IBM (США) разработала систему машин IBM-360. Эта система оказала существенное влияние на логическую организацию машин третьего поколения. СССР и другие соцстраны совместно разработали и организовали серийное производство Единой системы ЭВМ (ЕС ЭВМ) – семейства программно совместимых машин третьего поколения на интегральных микросхемах. Всего было выпущено три ряда этих машин. Модели третьего ряда отличались от 1-го и 2-го обладали большей производительностью и более широкими функциональными возможностями.

Для решения не очень сложных задач в рамках третьего поколения выпускались мини-ЭВМ: М6000, М7000, серия СМ, Электроника-100 и др.

Наряду с серийными ЭВМ разрабатывались вычислительные системы, в которых использовались параллельные вычисления, позволявшие значительно увеличить их производительность. Такие ЭВМ производились штучно и, благодаря очень высокой производительности, их стали называть супер-ЭВМ. Наиболее известные из них: зарубежные семейства IILIAC, CRAY,CYBER, а также отечественное семейство вычислительных систем Эльбрус.

На смену машинам 3-го поколения пришли машины 4-го поколения, характерной особенностью которых было использование сверхбольших интегральных микросхем, что позволило перейти к интеграции подсистем ЭВМ. Такая комплексная микроминиатюризация позволила резко уменьшить габариты и вес, а также улучшить все основные характеристики ЭВМ. Это позволило создать персональные ЭВМ (ПЭВМ).

Архитектура первых ПЭВМ определялась в основном 8-ми разрядными микропроцессорами (МП) типа Mostek 6205, I 8080, Zilog 80/86; 8-ми разрядными системными интерфейсами; внешней памятью на гибких магнитных дисках; видеосистемами малого разрешения; операционными системами (ОС) типа CP/M.

Следующее поколение ПЭВМ базировалось на 8 и 16-ти разрядных МП; внешней памятью на магнитных дисках типа «винчестер» небольшой емкости и быстродействия; видеоподсистемах среднего разрешения; операционных системах типа DOS/PC DOS, CP/M-86. Затем проявились новые типы 16-ти разрядных МП с расширенными функциональными

возможностями (I 80286, Motorola 68020, NS 32032), соответственно 16-ти разрядные магистрали и развитые версии ОС MS DOS/PC DOS.

Развитие ПЭВМ определялось динамикой развития и совершенствования микроэлектронных технологий, обеспечивающих непрерывное повышение уровня технических характеристик микропроцессоров. Совершенствовались также системные интерфейсы и программное обеспечение.

Летом 1995 г. два токийских университета продемонстрировали специализированный (предназначенный для моделирования задач астрофизики) суперкомпьютер GRAPE-4, собранный из 1692 микропроцессоров и обошедшийся всего в 2 млн. долл. Он первым в мире преодолел порог в 1 трлн. оп./с с результатом 1,08 Тфлопс. Через 15 месяцев компания Cray Research сообщила, что модель Cray T3E-900, насчитывавшая 2048 процессоров, побила рекорд японцев и достигла 1,8 Тфлопс. К тому времени результат NEC SX-4 составлял 1 Тфлопс, Hitachi SR2201 — 0,6 Тфлопс, а Fujitsu Siemens VPP700 — 0,5 Тфлопс.

В 1997 г. появились сообщения о проекте моделирования ядерного взрыва (ASCI) в Лос-Аламосской лаборатории. Созданный в соответствии с этим проектом комплекс ASCI Red на 9632 процессорах Pentium Pro компании Intel показал производительность сначала 1,8 Тфлопс, а затем 3,2 Тфлопс.

Мировыми лидером по производительности среди суперкомпьютеров стали в 2002 г. суперкомпьютер ASCI White компании IBM с 8192 процессорами и с производительностью 7,3 Tfops, в ноябре 2003 г. - компьютер Earth Simulator японской компании NEC с производительностью в 35,9 Tflops, установленный в Японии в 2002 г. и включающий 5120 процессоров, а в ноябре 2004 г. - компьютер IBM BlueGene/L с 70,7 Tflops .

В 2004 г. порог в 10 Tflops преодолел китайский суперкомпьютер "Шугуан4000А", установленный в Шанхае.

В 2002 г. в список 500 наиболее производительных компьютеров мира (Top500) впервые вошел российский суперкомпьютер, заняв 74-е место. Это суперкомпьютер МВС1000М, установленный в Межведомственном суперкомпьютерном центре (создан в 1996 году совместным решением Российской академии наук, Министерством науки и технологии, Министерством образования и Российским фондом фундаментальных исследований) и имеющий производительность 735 Gflops

Во всех компьютерах, начиная с первого поколения использовался единый подход к принципам построения элементной базы, математическим основам (преобразования десятичных чисел в двоичное и наоборот, выполнения арифметических операции над двоичными числами и т.д.).

Использовалось и используется логические элементы, которые работают одинакова, а реализованно на разной элементной базе.

В данном учебно-методическом указании рассмотренно математические основы построение ЭВМ и рассмотренны основные логические элементы, используемые для изготовления ЭВМ.

1. Математические основы построения ЭВМ

1.1. Выбор системы счисления для представления числовой информации

Под системой счисления понимается способ представления любого числа посредством некоторого алфавита символов, называемых цифрами. Все системы счисления делятся на два типа: позиционные и непозиционные. Система счисления называется позиционной, если одна и та же цифра (символ алфавита) имеет различное значение (вес) в зависимости от ее положения в последовательности цифр (символов алфавита), изображающей число. Это значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. Позиционной является десятичная система, используемая в повседневной жизни. Помимо десятичной системы существуют и другие позиционные системы счисления.

Все символы, используемые в той или иной системе счисления, образуют ее алфавит. Количество символов в алфавите называют основанием системы счисления. В десятичной системе счисления используется десять символов: 0, 1,2,3,4,5,6,7,8,9 и, следовательно, число десять является ее основанием.

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

$$x = \varepsilon_r s^r + \varepsilon_{r-1} s^{r-1} + \ldots + \varepsilon_1 s^1 + \varepsilon_0 s^0 + \varepsilon_{-1} s^{-1} + \varepsilon_{-2} s^{-2} + \ldots \quad ,$$

где в качестве коэффициентов ε_i могут стоять любые символы из алфавита системы счисления.

Число в позиционной системе счисления записывается в виде последовательности символов:

$$x = \varepsilon_r\, \varepsilon_{r-1} \ldots \varepsilon_1\, \varepsilon_0, \varepsilon_{-1} \varepsilon_{-2} \ldots$$

В этой последовательности запятая отделяет целую часть числа от дробной. Позиции символов, отсчитываемые от запятой, называют разрядами. В позиционной системе счисления значение каждого разряда, при одинаковых символах, больше значения соседнего справа разряда в s раз.

С учетом сказанного, в десятичной системе счисления запись 3027,105 означает число:

$$3 \cdot 10^3 + 0 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 1 \cdot 10^{-1} + 0 \cdot 10^{-2} + 5 \cdot 10^{-3} .$$

В ЭВМ применяют двоичную позиционную систему счисления и системы счисления, построенные на базе двоичной: двоично-восьмеричную, двоично-шестнадцатеричную, двоично-десятичную. Это объясняется техническими причинами. Очень сложно реализовать электронную схему, имеющую несколько устойчивых состояний, и также сложно реализовать схему соответствующей арифметики. Электронные схемы с двумя устойчивыми состояниями реализуются достаточно просто, и достаточно просто реализуется двоичная арифметика. Кроме того, электронные схемы с двумя устойчивыми состояниями имеют наивысшую помехозащищенность.

Система счисления — совокупность приемов и правил для записи чисел цифровыми знаками или символами.

Существуют системы позиционные и непозиционные. Наиболее известна десятичная позиционная система счисления, в которой для записи чисел используются цифры 0, 1, ..., 9. Способов записи чисел цифровыми знаками существует бесчисленное множество. Любая предназначенная для практического применения система счисления должна обеспечивать:

- возможность представления любого числа в рассматриваемом диапазоне величин;
- единственность представления (каждой комбинации символов должна соответствовать одна и только одна величина);
- простоту оперирования числами.

Самый простой способ записи чисел может быть описан выражением

$$A_D = D_1 + D_2 + ... + D_k = \sum_{i=1}^{i=k} D_i \ ,$$

где A_D — запись числа A в системе счисления D; D_i — символы системы, образующие базу $D = \{D_1, D_2, ..., D_k\}$.

По этому принципу построены непозиционные системы счисления.

Непозиционная система счисления — система, для которой значение символа не зависит от его положения в числе.

Принципы построения таких, систем не сложны. Для их образования используют в основном операции сложения и вычитания. Например, система с одним символом (палочкой) встречалась у многих народов. Для изображения какого-то числа в этой системе нужно записать количество палочек, равное данному числу. Эта система неэффективна, так как запись числа получается длинной. Другим примером непозиционной системы счисления является римская система, использующая набор следующих символов: I, X, V, L, C, D, M и т. д., которые соответствуют следующим величинам: I(1), V(5), X(10), L(50), C(100), D(500), M(1000). В этой системе существует отклонение от правила независимости значения цифры от положения в числе. В числах LX и XL символ X принимает два различных значения: +10 — в первом случае и -10 — во втором.

Таким образом, цифры в непозиционных системах счисления соответствуют некоторым фиксированным числам, например: I I I (три); LI X (пятьдесят девять); D L V (пятьсот пятьдесят пять).

Недостатком непозиционных систем, является отсутствие формальных правил записи чисел и, следовательно, арифметических действие над ними.

Огромными преимуществами в наглядности представления чисел и в простоте выполнения арифметических операций обладают позиционные системы счисление.

Система счисления называется позиционной, если одна и та же цифра имеет различные значение, определяющееся позицией цифры в последовательности цифр, изображающей число. Это значение меняется в однозначной зависимости от позиции, занимаемой цифрой, по некоторому закону. К позиционной системе относятся, кроме десятичной, двоичная, восьмеричная, шестнадцатеричная и ряд других.

В позиционных системах счисления число, изображенное в виде

$$x_m x_{m-1} \ldots x_2 x_1 x_0,$$

которое можно представить следующем образом

$$x_m P_m + x_{m-1} P_{m-1} + \ldots + x_2 P_2 + x_1 P_1 + x_0 P_0 = \sum_{i=0}^{m} x_i P_i , \qquad (1)$$

где $x_m, x_{m-1}, \ldots, x_2, x_1, x_0$ - символы, обозначающие целые числа;

$P_m, P_{m-1}, \ldots P_2, P_1, P_0$ - веса, т.е. количественные значения каждой единицы, определяемые местом (позицией), занимаемым соответствующим символом в изображения числа. В дальнейшем для обозначения используемой системы счисления будем заключать число в скобки и в нижнем индексе указывать основание системы счисления, если это не десятичная система.

Позиционная система счисления — система, удовлетворяющая равенству (1). Любая позиционная система характеризуется своим основанием.

Основание позиционной системы счисления - это количество различных знаков или символов, используемых для изображения цифр в данной системе.

За основание можно принять любое натуральное число - два, три, четыре, шестнадцать и т.д. Следовательно, возможно бесконечное множество позиционных систем.

Естественная позиционная система счисления имеет место, если q — целое положительное число.

В позиционной системе счисления значение цифры определяется ее положением в числе: один и тот же знак принимает различное значение. Например, в десятичном числе 222 первая цифра справа означает две единицы, соседняя с ней — два десятка, а левая — две сотни.

Любая позиционная система счисления характеризуется основанием. ***Основание (базис) q естественной позиционной системы счисления*** — количество знаков или символов, используемых для изображения числа в данной системе. Возможно бесчисленное множество позиционных систем, так как, приняв за основание любое число, можно образовать новую систему. Например, запись числа в шестнадцатеричной системе может проводиться с помощью следующих знаков (цифр): 0, 1, ..., 9, *A, B, C, D, E, F* (вместо *A, ..., F* можно записать любые другие символы, например, $\bar{1}, \bar{2}, \ldots, \bar{5}$).

Рассмотрим примеры.

Десятичная система счисления. Пришла в Европу из Индии, где она появилась не позднее VI века н.э. В этой системе 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, но информацию несет не только цифра, но и место, на котором цифра стоит (то есть ее позиция). В десятичной системе счисления особую роль играют число 10 и его степени: 10, 100, 1000 и т.д. Самая правая цифра числа показывает число единиц, вторая справа - число десятков, следующая - число сотен и т.д.

Пример 1. С учетом сказанного десятичное число 5087,109 можно представить в виде:

$$5 \cdot 10^3 + 0 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0 + 1 \cdot 10^{-1} + 0 \cdot 10^{-2} + 9 \cdot 10^{-3}.$$

Двоичная система счисления. В этой системе всего две цифры - 0 и 1. Особую роль здесь играет число 2 и его степени: 2, 4, 8 и т.д. Самая правая цифра числа показывает число единиц, следующая цифра - число двоек, следующая - число четверок и т.д. Двоичная система счисления позволяет закодировать любое натуральное число - представить его в виде последовательности нулей и единиц. В двоичном виде можно представлять не только числа, но и любую другую информацию: тексты, картинки, фильмы и аудиозаписи. Инженеров двоичное кодирование привлекает тем, что легко реализуется технически.

Пример 2. Двоичное число $(10101101{,}101)_2 = 1\cdot 2^7+0\cdot 2^6+1\cdot 2^5+0\cdot 2^4+1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0+1\cdot 2^{-1}+0\cdot 2^{-2}+1\cdot 2^{-3}$
если посчитать полученное число, то оно равно в десятичной системе счисление $(173{,}625)_{10}$.

Восьмеричная система счисления. В этой системе счисления 8 цифр: 0, 1, 2, 3, 4, 5, 6, 7. Цифра 1, указанная в самом младшем разряде, означает - как и в десятичном числе - просто единицу. Та же цифра 1 в следующем разряде означает 8, в следующем 64 и т.д. Число 100 (восьмеричное) есть не что иное, как 64 (десятичное). Чтобы перевести в двоичную систему, например, число 611 (восьмеричное), надо заменить каждую цифру эквивалентной ей двоичной триадой (тройкой цифр). Легко догадаться, что для перевода многозначного двоичного числа в восьмеричную систему нужно разбить его на триады справа налево от запятой и заменить каждую триаду соответствующей восьмеричной цифрой.

Шестнадцатеричная система счисления. Запись числа в восьмеричной системе счисления достаточно компактна, но еще компактнее она получается в шестнадцатеричной системе. В качестве первых 10 из 16 шестнадцатеричных цифр взяты привычные цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а вот в качестве остальных 6 цифр используют первые буквы латинского алфавита: A, B, C, D, E, F. Цифра 1, записанная в самом младшем разряде, означат просто единицу. Та же цифра 1 в следующем - 16 (десятичное), в следующем - 256 (десятичное) и т.д. Цифра F, указанная в самом младшем разряде, означает 15 (десятичное). Перевод из шестнадцатеричной системы в двоичную и обратно производится аналогично тому, как это делается для восьмеричной системы только вместе триады(три двоичных разряда), берется тетрада (четыре двоичных разряда).

Система счисления - это совокупность правил для обозначения и наименования чисел.

Для позиционной системы счисления справедливо равенство

$$A_q = \sum_{i=-m}^{i=n} a_i q^i, \qquad (2)$$

или $A_q = a_n q^n + \ldots + a_1 q^1 + a_0 q^0 + a_{-1} q^{-1} + \ldots + a_{-m} q^{-m}$, где A_q—произвольное число, записанное в системе счисления с основанием q; $n + 1$, m — количество целых и дробных разрядов.

На практике используют сокращенную запись чисел:

$$A_q = a_n a_{n-1} \ldots a_1 a_0 a_{-1} \ldots a_{-m}.$$

В восьмеричной системе счисления числа изображают с помощью цифр 0,1,...,7. Например, $124{,}537_8 = 1\,8^2 + 2\cdot 8^1 + 4\cdot 8^0 + 5\cdot 8^{-1} + 3\cdot 8^{-2} + 7\cdot 8^{-3}$.

В двоичной системе счисления используют цифры 0, 1. Например, $1001{,}1101_2 = 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4}$.

Для записи чисел в троичной системе берут цифры 0,1,2. Например, $2122_3 = 2\cdot 3^3 + 1\cdot 3^2 + 2\cdot 3^1 + 2\cdot 3^0$.

В таблице 1 приведены эквиваленты десятичных цифр в различных системах счисления.

Таблица .1

Десятичная цифра	Эквиваленты в других системах счисления				
	$q = 2$	$q = 3$	$q = 5$	$q = 8$	$q = 16$
0	0000	000	00	00	0
1	0001	001	01	01	1
2	0010	002	02	02	2
3	0011	010	03	03	3
4	0100	011	04	04	4
5	0101	012	10	05	5
6	0110	020	11	06	6
7	0111	021	12	07	7
8	1000	022	13	10	8
9	1001	100	14	11	9
10	1010	101	20	12	A
11	1011	102	21	13	B
12	1100	110	22	14	C
13	1101	111	23	15	D
14	1110	112	24	16	E
15	1111	120	30	17	F

Для любой позиционной системы счисления справедливо, что основание изображается символом 10 в своей системе, т. е. любое число можно записать в виде

$$A_q = a_n \cdot 10^n + ... + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + ... + a_{-m} \cdot 10^{-m}. \quad (3)$$

В ЭВМ используют позиционные системы счисления. В дальнейшем для простоты изложения будем употреблять термин «система счисления», имея в виду позиционные системы.

Вес разряда p_i, числа в позиционной системе счисления выражается соотношением

$$p_i = q^i / q^0 = q^i, \quad (4)$$

где i — номер разряда при отсчете справа налево.

Если разряд имеет вес $p_i = 10^k$ (Десятичная система счисления), то следующий старший разряд будет иметь вес $p_{i+1} = 10^{k+1}$, а соседний младший разряд — вес $p_{i-1} = 10^{k-1}$. Такая взаимосвязь разрядов приводит к необходимости передачи информации между ними.

Если в данном разряде накопилось значение единиц, равное или большее q, то должна происходить передача единицы в соседний старший разряд. При сложении такие передачи информации называют ***переносами,*** а при вычитании — ***заемами***. Передача переносов или заемов происходит последовательно от разряда к разряду.

Длина числа (ДЧ) — количество позиций (или разрядов) используемые при записи числа. Обычно его принято называть как длина разрядной сетки (ДРС) занимаемое числом.

Для разных систем счисления характерна разная длина разрядной сетки, необходимая для записи одного и того же числа. Например, $96 = 120_8 = 10120_3 = 1100000_2$. Здесь одно и то же число, записанное в разных базисах, имеет разную длину разрядной сетки. Чем меньше основание системы, тем больше длина числа.

Если длина разрядной сетки задана, то это ограничивает максимальное (или минимальное) по абсолютному значению число, которое может быть записано.

Пусть длина разрядной сетки равна любому положительному числу, например n.

Тогда $A_{q\max} = q^n - 1$; $A_{q\min} = -(q^n - 1)$.

Диапазон представления* (ДП) *чисел в заданной системе счисления — интервал числовой оси, заключенный между максимальным и минимальным числами, представленными длиной разрядной сетки:

$$A_{q\max} \geq \text{ДП} \geq A_{q\min}. \qquad (5)$$

Правильный выбор системы счисления — важный практический вопрос, поскольку от его решения зависят такие технические характеристики проектируемой ЭВМ, как скорость вычислений, объем памяти, сложность алгоритмов выполнения арифметических операций. При выборе системы счисления для ЭВМ необходимо учитывать следующее:

*основание системы счисления определяет количество устойчивых состояний, которые должен иметь функциональный элемент, выбранный для изображения разрядов числа;

*длина числа существенно зависит от основания системы счисления;

*система счисления должна обеспечить простые алгоритмы выполнения арифметических и логических операций.

Десятичная система, столь привычная в повседневной жизни, не является наилучшей с точки зрения ее технической реализации в ЭВМ. Известные в настоящее время элементы, обладающие десятью устойчивыми состояниями (элементы на основе сегнетокерамики, декатроны и др.), имеют невысокую скорость переключения, и следовательно, не могут обеспечить соответствующее быстродействие машины.

Подавляющее большинство компонентов электронных схем, применяемых для построения цифровых систем обработки информации — двухпозиционные. С этой точки зрения для ЭВМ наиболее подходит двоичная система счисления. Но рационально ли использование этой системы с точки зрения затрат оборудования? Для ответа на этот вопрос введем показатель экономичности системы C — произведение основания системы на длину разрядной сетки, выбранную для записи чисел в этой системе:

$$C = qN, \qquad (6)$$

где q — основание системы счисления; N— количество разрядов.

Если принять, что каждый разряд числа представлен не одним элементом с q устойчивыми состояниями, а q элементами, каждый из которых имеет одно устойчивое состояние, то показатель экономичности укажет словное количество оборудования, которое необходимо затратить на представление чисел в этой системе.

Максимальное число, которое можно изобразить в системе с основанием q,

$$A_{q\max} = q^N - 1. \qquad (7)$$

Из (7) можно найти требуемую длину разрядной сетки:

$$N = \log_q (A_{q\max} + 1). \qquad (8)$$

Тогда для любой системы счисления $C = q \log_q (A_{q\max} + 1)$.

Представим, что величина q принимает любые значения (целочисленные и дробные), т. е. является непрерывной величиной. Это необходимо для того, чтобы рассматривать величину C как функцию от величины q. Данное допущение не является строгим, однако позволяет получить интересный вывод: если за единицу измерения оборудования принят условный элемент с одним устойчивым состоянием, то для сравнения двух систем счисления можно ввести относительный показатель экономичности

$$F = q \log_q (A_{q\max} + 1) / [2 \log_2 (A_{2\max} + 1)], \qquad (9)$$

позволяющий сравнить любую систему счисления с двоичной.

Если функция F непрерывна, то, как видно из приведенного ниже соотношения, она имеет минимум.

q	2	3	4	6	8	10
F	1,000	0,946	1,000	1,148	1,333	1,505

На рис. 1 представлена зависимость величины F от основания системы счисления q. Нижняя точка графика соответствует минимуму функции F, определяемому из условия $dF/dt = 0$, что соответствует значению $q = e$. Следовательно, с точки зрения минимальных затрат условного оборудования, наиболее экономичной является система счисления с основанием, равным $e \approx 2{,}72$.

Используя (9), можно доказать, что троичная система счисления экономичнее двоичной системы счисление. В подавляющем большинстве ЭВМ используют двоичную систему счисления, однако для ЭВМ это связано с преодолением дополнительных трудностей, возникающих при переводе входной информации в двоичную систему счисления и двоичной информации в выходную информацию.

Первым, который, использовал два символа для кодирование информации был известный философ XVII в. Ф. Бэкон.

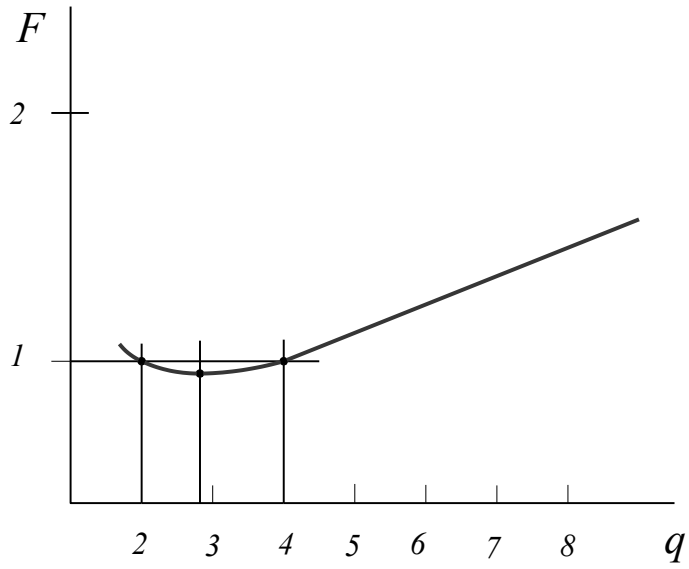


Рис. 1. Зависимость относительного показателя экономичности от основания системы счисления

1.2. Преобразование чисел с одной системы счисления в другую.

В процессе преобразования информации в компьютере возникает необходимость перевода чисел из одной позиционной системы счисления в другую. Это обусловлено тем, что в качестве внутреннего алфавита наиболее целесообразно, как было сказано выше, использовать двоичный алфавит с символами **0** и **1**, т.е.двоичная система счисления.

В зависимости от системы счисления необходимо использовать такие физические базовые элементы, которые имели бы столько устойчивых состояния, сколько сама система счисления. Например, двоичной системе - два устойчивых состояние, в троичной системе - три, десятичной системе – десять и т.д. В системах электронной обработки информации используется двоичное представление информации. Это связанно с тем, что для физической реализации двоичной информации наиболее удобно использовать логические элементы с двумя устойчивыми состояниями. Эти состояния обозначают двоичными символами: «0» и «1»,т.е. сигнал есть или нет сигнала.

В системах обработки цифровой информации очень часто, наряду с двоичной системой счисления, используется шестнадцатеричная система счисления, которая достаточно просто представляется в двоичной форме.

Изображения некоторых чисел в десятичной, двоичной, троичной, пятеричной, восьмеричной и шестнадцатеричной системе показано в таблице 2.

Как видно из таблицы 2. двоичное изображения числа требует большего количества разрядов, чем его десятичное представление (примерно в 3.3 раза для многоразрядных чисел). Тем не менее, применение двоичной системы позволяет существенно уменьшить аппаратурные затраты, при создания устройств обработки цифровой информации, в том числе и ЭВМ.

Кроме двоичной системы счисления в компьютерах используется и шестнадцатеричная система счисления, имеющая основанием число 16. В шестнадцатеричной системе для изображения чисел используют десять арабских цифр (0-9) и шесть латинских букв (A,B,C,D,E,F), как показано в таблице 2.

Применение шестнадцатеричной системы счисления для изображения десятичных чисел, позволяет существенно увеличить диапазон представления чисел, при тех же используемых двоичных разрядов.

Следует, отметит, что переход между двоичными и шестнадцатеричными числами и наоборот, осуществляется достаточно просто. Для перехода от шестнадцатеричной системы к двоичной системе каждая цифра шестнадцатеричного числа заменяется соответствующим четырехразрядным двоичным числом (тетрадой). При этом ненужные нули справа и слева отбрасываются. Для обратного перехода, от двоичной системы счисления в шестнадцатеричной, необходимо, начиная от запятой выделять в право и влево по четыре разряда и заменять их шестнадцатеричными числами. Там, где цифр меньше, чем четыре, необходимо добавлять нули.

Таблица 2

Десятичное Число	Двоичное Число	Шестнадцате-ричное число	Десятичное Число	Двоичное Число	Шестнадцате-ричное число
0	0	0	15	1111	F
1	01	1	16	10000	10
2	10	2	17	10001	11
3	11	3	18	10010	12
4	100	4	0,5	0,1	0.8
5	101	5	0,25	0,01	0.4
6	110	6	0,125	0,001	0.2
7	111	7	0,875	0,111	0.E
8	1000	8	0,375	0,011	0.6
9	1001	9	0,0625	0,0001	0.1
10	1010	A	0,4375	0,0111	0.7
11	1011	B	0,03125	0,00001	0.08
12	1100	C	0,015625	0,000001	0.04
13	1101	D	3,125	11,001	3.2
14	1110	E	4,5	100,1	4.8

Пример 1. Преобразовать шестнадцатеричное число (B2E,4**)** в десятичное число:

$(B2E,4)_{16} = B\cdot 16+2\cdot 16^{1}+E\cdot 16^{0}+4\cdot 16^{-1}=(2862,25)_{10}$.

Пример 2. Преобразовать шестнадцатеричное число (7B2,E) в двоичное число:

$(7B2,E)_{16}$ =(0111 1011 0010 , 1110) =(11110110010,111).

Для перехода от двоичной к шестнадцатеричной системе поступают следующим образом: двигаясь от запятой влево и вправо, разбивают двоичное число на группы по четыре разряда, дополняя при необходимости,

нулями крайние левую и правую группы. Затем каждую группу из четырех разрядов заменяют соответствующей шестнадцатеричной цифрой.

Пример 3. Двоичное число 111111111011,100111 перевести в шестнадцатеричное.

$$(0111\ 1111\ 1011,\ 1001\ 1100)_2 = (7FB,9C)_{16}.$$

В процессе работы компьютера происходит многократные преобразование двоичных чисел в десятичные и наоборот, особенно при вводе-выводе чисел.

Рассмотрим общие правила преобразование чисел с одной системы счисление в другую.

Пусть необходимо перевести число У, представленное в системе счисления с основанием S:

$$У = r_m\ r_{m-1} r_{m-2}\ \ldots r_1\ r_0\ r_{-1} \ldots r_{-k}$$

преобразовать в h- систему, выполняя нужные для этого арифметические действия в новой h-системе. Для этого достаточно число представить в виде соответствующей суммы степени s:

$$У = r_m \cdot s^m + r_{m-1} \cdot s^{m-1} + \ldots + r_1 \cdot s^1 + r_0 \cdot s^0 + r_{-1} \cdot s^{-1} + \ldots + r_{-k} \cdot s^{-k},$$

в которой основание s и все коэффициенты r_i выражены в новой h-системе, и выполнить в h-системе все необходимые для вычисления этой суммы действия.

Пример 4. Перевести в десятичную систему шестнадцатеричную число

$$У = (2E5,A)_{16}.$$

Имеем $У = 2\cdot16^2+14\cdot16^1+5\cdot16^0+10\cdot16^{-1}=(741,625)_{10}$, где шестнадцатеричные цифры Е и А заменены соответствующими их десятичными значениями.

Пример 5. Перевести в десятичную систему двоичную число $z=(11011,101)_2$.

$$Z = 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 1\cdot2^{-3} = (27,625)_{10}.$$

Рассмотрим перевод чисел из s-системы в h-систему посредством арифметических операции исходной s-системы.

В этом случае правила для перевода целых чисел и дробей различны, поэтому перевод надо делать отдельно для целой части и дробной части.

Перевод целых чисел. Пусть целое число y, представленное в s-системе, требуется перевести в h- систему. Искомая запись числа y в h-системе имеет вид:

$$У = \delta_r\ \delta_{r-1} \ldots \delta_0 = \delta_r \cdot h^r + \delta_{r-1} \cdot h^{r-1} + \ldots + \delta_0 \cdot h^0 .$$

Разделив y на h, получим:

$$\frac{y}{h} = \delta_r h^{r-1} + \delta_{r-1} h^{r-2} + \ldots + \delta_1 + \frac{\delta_0}{h} ,$$

отсюда

$$y = y_1 \cdot h + \delta_0,$$

где y_1 - частное от деления числа y на основание системы h, а младшая цифра искомого представления числа y в h-системе есть остаток от этого деления.

Если теперь разделить y_1 на h, то получим:

$$\frac{y_1}{h} = \delta_r h^{r-2} + \delta_{r-1} h^{r-3} + ... + \delta_2 + \frac{\delta_1}{h}.$$

Отсюда

$$y_1 = y_2 \cdot h + \delta_2,$$

остаток от второго деления есть цифра δ_2 следующего разряда в представления числа y в h- системе и т.д.

Таким образом, получаем правило для перевода целого числа из s-системы счисления в h- систему нужно:

- шаг1: разделить число находящего в системе счисление s на основание системы счисление h;
- шаг 2: Полученное частное делим на основание системы счисления h;
- шаг 3: Шаг 2 повторяем до тех пор, пока, частное от деления не станет меньше h;
- шаг 4: Старшей цифрой в записи числа в h – системе служит последнее частное, а следующие за ней цифры дают остатки, выписываемые в последовательности, обратной их получению.

Пример 6. Перевести десятичное число 75 в двоичное число.

В результате получаем Рис.2.

$(75)_{10} = (1001011)_2$. Как видно, полученное число записываем, начиная с конца.

Перевод дробных чисел. Перевод в h- систему правильной дроби z, представленной в системе счисления с основанием s, означает запись этой

75	2					
74	37	2				
1	36	18	2			
	1	18	9	2		
		0	8	4	2	
			1	4	2	2
				0	2	1
					0	

Рис.2.

дроби в виде

$$z = 0, \alpha_{-1}\ \alpha_{-2} \ldots \alpha_q \ \ldots = \alpha_{-1} \cdot h^{-1} + \alpha_{-2} \cdot h^{-2} + \ldots + \alpha_q \cdot h^{-q} + \ldots$$

Умножая z на h, получаем:

$$Z \cdot h = \alpha_{-1} + \alpha_{-2} h^{-1} + \ldots + \alpha_{-q} h^{-(q-1)} + \ldots + = \alpha_{-1} + z_1,$$

где α_{-1} и z_1 соответственно целая и дробные части этого произведения. При этом целая часть α_{-1} есть старшая цифра в представлении числа z в h-системе.

Если теперь умножить на h правильную дробь z_1, то целая часть произведения дает следующую цифру α_{-2} в представлении числа в h системе.

Таким образом, можно сформулировать правило: для перевода правильной дроби из s- системы в систему счисления с основанием h нужно умножить исходную дробь и дробные части получающихся произведений на основание h, представленное в старой s- системе. Целые части получающихся произведений дают последовательность цифр в представлении дроби в h –системе.

Пример 7. Записать десятичные дроби 0,1875 и 0,2879 в двоичной системе счисления.

Первое число имеет вид:

0,1875	0,3750	0,7500	0,5000
x	x	x	x
2	2	2	2
0,3750	0,7500	1,5000	1,0000
↓	↓	↓	↓
0	0	1	1

В результате значение первого числа равно

$(0,1875)_{10} = (0,0011)_2$

Второе число получим следующим образом:

0.2879	0.5758	0.1516	0.3032	0.6064	0.2128
x 2	x 2	x 2	x 2	x 2	x 2
0,5758	1,1516	0,3032	0,6064	1,2128	0,4356
↓	↓	↓	↓	↓	↓
0	1	0	0	1	0

0.4356	0.8712	0.7424	0.4848
x 2	x 2	x 2	x 2
0,8712	1,7424	1,4848	0,9796
↓	↓	↓	↓
0	1	1	0

$(0,2879)_{10} =$

$(0,0100100110)_2$

Как видно по этому примеру перевод осуществлен приближенно. Обычно перевод дробей из одной системы в другую производиться приближенно. Необходимо отметит, что количество двоичных разрядов, определяется примерно с коэффициентом 3.3, т.е. количество десятичных разрядов умножают на 3,3 и округляют в большую сторону, до целого.

При переводе неправильной дроби переводят отдельно целую и дробную части, руководствуясь соответствующими правилами.

При переводе правильных дробей из одной системы счисления в другую можно получить дробь в виде бесконечного или расходящегося ряда. Процесс перевода можно закончить, если появится дробная часть, имеющая во всех разрядах нули, или будет достигнута заданная точность перевода (получено требуемое количество разрядов результата). Последнее означает, что при переводе дроби необходимо указать количество разрядов числа в новой системе счисления. Естественно, что при этом возникает погрешность перевода чисел, которую надо оценивать.

Табличный метод перевода. В простейшем виде табличный метод заключается в следующем: имеется таблица всех чисел одной системы с соответствующими эквивалентами из другой системы; задача перевода сводится к нахождению соответствующей строки таблицы и выбору из нее эквивалента. Такая таблица очень громоздка и требует большой емкости памяти для хранения.

Другой вид табличного метода заключается в том, что имеются таблицы эквивалентов в каждой системе только для цифр этих систем и степеней основания (положительных и отрицательных); задача перевода сводится к тому, что в выражение формулы (2) для исходной системы счисления надо подставить эквиваленты из новой системы для всех цифр и степеней основания и произвести соответствующие действия (умножения и сложения) по правилам q_2 - арифметики. Полученный результат этих действий будет изображать число в новой системе счисления.

Пример 8. *Перевести десятичное число A = 113 в двоичную систему счисления, используя следующее соотношение эквивалентов и степени основания:*

Десятичное число..... 10^0 10^1 10^2
Двоичный эквивалент..... 0001 1010 1100110

Решение. Подставив значения двоичных эквивалентов десятичных цифр и степеней - основания в (11), получим

$A=113=1\cdot10^2+1\cdot10^1+3\cdot10^0=0001\cdot1100100+0001\cdot1010+0011\cdot0001=1110001_2$.

Ответ: 1110001_2.

Пример 9. *Перевести двоичное число $A_2 = 11001{,}1$ в десятичную систему счисления:*

Двоичное число..... 0,1 00001 00010
Десятичный эквивалент..... $2^{-1}=0{,}5$ $2^0 = 1$ $2^1 = 2$
Двоичное число..... 00100 01000 10000
Десятичный эквивалент..... $2^2 = 4$ $2^3 = 8$ $2^4 = 16$

Решение. $A = 1\cdot16+1\cdot8+0\cdot4+0\cdot2+1\cdot1+1\cdot0{,}5 = 25{,}5$. Ответ: A = 25,5.

Использование промежуточной системы счисления. Этот метод применяют при переводе из десятичной системы в двоичную и наоборот. В качестве промежуточной системы счисления можно использовать, например, восьмеричную систему.

Рассмотрим примеры, в которых перевод одного и того же числа в разные системы счисления осуществляется методом деления на основание новой системы. Запись будем вести в столбик, где справа от вертикальной черты записываются остатки деления на каждом шаге, а слева — целая часть частного.

Пример 10. Перевести десятичное число А =121 в двоичную систему счисления, используя в качестве промежуточной восьмеричную систему счисления.

Решение.

$q_2 = 8$	
121	1
15	7
1	1

3 шага

$q_2 = 2$	
121	1
60	0
30	0
15	1
7	1
3	1
1	1

7 шагов

Ответ: $A = 121 = 171_8 = 1111001_2$.

Сравнивая эти примеры, видим, что при переводе числа из десятичной системы в восьмеричную требуется в два с лишним раза меньше шагов, чем при переводе в двоичную систему. Если при этом учесть, что восьмеричная система связана с двоичной соотношением $8^k = (2^3)^k$, то перевод из восьмеричной системы в двоичную и наоборот можно осуществить простой заменой восьмеричных цифр их двоичными эквивалентами. Триада — двоичный эквивалент восьмеричных цифр.

Пример 11. Перевести двоичное число A_2 = 1011,0111 в восьмеричную систему счисления.

Решение. Исходное число условно разбиваем на триады справа налево для целых чисел и слева направо для правильной дроби. Затем заменяем каждую триаду в соответствии с нижеприведенным соответствием.

Восьмеричная цифра.... 0 1 2 3 4 5 6 7
Двоичный эквивалент.. 000 001 010 011 100 101 110 111

$A_2 = 001\ \ 011{,}011\ \ 100.$

$A_8 = \ 1\ \ \ 3{,}3\ \ \ 4.$

Ответ: $A_8 = 13{,}34$.

В качестве промежуточных систем счисления целесообразно использовать системы с основанием $q = 2^k$. При этом существенно упрощается преобразование информации из системы счисления с основанием $q = 2^k$ в двоичную систему и наоборот. Преобразование фактически сводится к тому, что символы первоначальной информации, заданной в системе с основанием $q = 2^k$, заменяются соответствующими двоичными эквивалентами (см. табл. 1). Представление десятичных чисел в таком виде называется ***десятично-двоичным***. Обратное преобразование из двоичной системы в систему с основанием $q = 2^k$ сводится:

- что двоичный код разбивается на группы по k двоичных разрядов в каждой (начиная от младших разрядов для целых чисел или с первого разряда после запятой для правильных дробей);

- эти группы (триады, тетрады (табл.3) и т. д.) заменяются соответствующими символами исходной системы счисления.

Таблица 3

Десятичное число	Двоичный эквивалент для $q = 2^4$	Десятичное число	Двоичный эквивалент для $q = 2^4$	Десятичное число	Двоичный эквивалент для $q = 2^4$
0	0000	6	0110	11	1011
1	0001	7	0111	12	1100
2	0010	8	1000	13	1101
3	0011	9	1001	14	1110
4	0100	10	1010	15	1111
5	0101				

1.3. Другие системы счисления

При наладке аппаратных средств (программа BIOS и т.д.) и написании новых программ (особенно на языках низкого уровня типа ассемблера или С) часто возникает необходимость заглянуть в память машины, чтобы оценить ее текущее состояние. Но там все заполнено длинными последовательностями нулей и единиц, очень неудобных для восприятия.

Кроме того, естественные возможности человеческого мышления не позволяют оценить быстро и точно величину числа, представленного, например , комбинацией из 16 нулей и единиц. Для облегчения восприятия двоичного числа решили разбить его на группы разрядов, например, по триили четыре разряда. Эта идея оказалась удачной, так как последовательность из 3 бит имеет 8 комбинаций, а последовательность из 4 бит –16 комбинаций.

Числа 8 и 16 – степени двойки, поэтому легко находить соответствие между двоичными числами. Развивая эту идею, пришли к выводу, что группы разрядов можно закодировать, сократив при этом последовательность знаков.

Для кодировки трех битов (триад) требуется 8 цифр, и поэтому взяли цифры от 0 до 7 десятичной системы. Для кодировки четырех битов (тетрад) необходимо 16 знаков, и взяли 10 цифр десятичной системы и 6 букв латинского алфавита: A,B,C,D,E,F. полученные системы, имеющие в основании 8 и 16 , назвали соответственно восьмеричной и шестнадцатеричной.

Таблица 4. *Восьмеричная и шестнадцатеричная системы*

Десятичное число	Восьмеричное число	Триада	Шестнадцатеричное число	Тетрада
0	0	000	0	0000
1	1	001	1	0001
2	2	010	2	0010
3	3	011	3	0011

4	4	100	4	0100
5	5	101	5	0101
6	6	110	6	0110
7	7	111	7	0111
8	10	001 000	8	1000
9	11	001 001	9	1001
10	12	001 010	A	1010
11	13	001 011	B	1011
12	14	001 100	C	1100
13	15	001 101	D	1101
14	16	001 110	E	1110
15	17	001 111	F	1111
16	20	010 000	10	10000

В таблице 4 приведены числа в десятичной, восьмеричной и шестнадцатеричной системах и соответствующие группы бит в двоичной системе.

16-разрядное двоичное число со знаковым разрядом можно представить 6-разрядным восьмеричным, причем старший байт в нем будет принимать значения лишь 0 или 1. В шестнадцатеричной системе такое число займет 4 разряда.

Пример 1 Получение восьмеричных и шестнадцатеричных чисел

$$\underbrace{1011}_{B}\ \underbrace{1000}_{8}\ \underbrace{0000}_{0}\ \underbrace{1111}_{F}{}_{2} = B80F_{16}$$

$$\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{001}_{1}\ \underbrace{110}_{6}\ \underbrace{010}_{2}{}_{2} = 57162_{8}$$

Арифметические операции над числами в восьмеричной или шестнадцатеричной системах проводятся по тем же правилам, что и в десятичной системе. Только надо помнить, что если имеет место перенос, то переносится не после 10, а 8 или 16.

Таблица 5 дает представление о переводе чисел в различные системы.

Таблица 5 *Перевод чисел из одной системы счисления в другую*

Двоичные числа	Восьмеричные числа	Десятичные числа	Шестнадцатеричные числа
0,0001	0,04	0,0625	0,1
0,001	0,1	0,125	0,2
0,01	0,2	0,25	0,4
0,1	0,4	0,5	0,8
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	10	8	8

1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F
10000	20	16	10

1.4. Двоично-десятичное кодирование

Для кодирования десятичных символов используются три варианта двоичных кодов: код 8421, код с избытком 3, код 2 из пяти и код Айкена. Коды приведены в таблице 6.

Таблица 6 .

Десятичные символы.	**Код 8421**	**Код с избытком 3.**	**Код 2 из 5.**	**Код Айкена (2421)**
0	0000	0011	11000	0000
1	0001	0100	00011	0001
2	0010	0101	00101	0010
3	0011	0110	00110	0011
4	0100	0111	01001	0100
5	0101	1000	01010	1011
6	0110	1001	01100	1100
7	0111	1010	10001	1101
8	1000	1011	10010	1110
9	1001	1100	10100	1111

Из двоично-десятичных кодов наибольшее распространение в вычислительной технике нашел код 8421. Этот код удобен для выполнения преобразований из десятичной системы в двоичную и обратно. Однако использование этого кода связано с трудностями обнаружения переноса в следующий десятичный разряд и сложностью перехода к обратным и дополнительным кодам для десятичных чисел.

Код с избытком 3 удобен для выполнения арифметических операций над двоично-десятичными числами, но неудобен для преобразования чисел из одной системы счисления в другую. Код “с избытком 3”получается из обычного двоично-десятичного арифметическим прибавлением числа 3 (двоичное число 0011).

Код 2 из 5 обладает избыточностью, что удобно при контроле правильности передачи по линиям связи. Одиночные и некратные двойке ошибки легко обнаруживаются, так как количество единиц в кодовой комбинации становится неравным 2.

Для выполнения сложения и вычитания двоично-десятичных чисел наиболее удобно использовать самодополняющиеся коды, к числу которых относятся код Айкена.

Код Айкена отличается от обычного двоично-десятичного кода, имеющего весовые коэффициенты разрядов в тетрадах 8421, другими значениями весовых коэффициентов разрядов: 2421.

Например, число $(A)_{10} = 361$ в двоично-десятичном коде записывается в виде $(A)_{2/10} = 0011\ 0110\ 0000$. Как видно из таблицы 5.2 обратный код $\overline{A}'_{2/10}$ числа, представленного в каком-либо само дополняющемся двоично-десятичном коде $A'_{2/10}$, является его двоичным дополнением до 9.

Например, число 5 в коде «с избытком 3» $A'_{2/10}=1000$ имеет обратный код $\overline{A}'_{2/10}=0111$, соответствующий числу 4 в коде «с избытком 3», которое «дополняет» число 5 до 9, так как 5+4=9.

При записи чисел в кодах ASCII цифрам от 0 до 9 поставлены в соответствие восьмиразрядные двоичные коды от 00110000 до 00111001.

ЭВМ, предназначенные для обработки экономической информации, позволяют производить арифметические операции в десятичной системе счисления над числами, представленными в двоично-десятичных кодах и кодах ASCII.

Для сложения и вычитания чисел в десятичной системе счисления в микро-ЭВМ используется не одна команда, как это имеет место в больших ЭВМ, а две: команда двоичного сложения и команда десятичной коррекции.

Подробнее процедура сложения десятичных чисел будет рассмотрена ниже.

Отрицательные десятичные числа представляются также с использованием десятичного дополнительного кода. Для кодирования знакового разряда S используются, например, такие комбинации:

$$S = \begin{cases} 0000 - \text{для положительных чисел,} \\ 1001 - \text{для отрицательных чисел.} \end{cases}$$

+	0000, 0101 0010	(+52)
	1001, 0101 0111	(—43 в дополнительном коде)
	1001, 1010 1001	Двоичное сложение
+	0110 0110	Десятичная коррекция
	0000, 0000 1001	(+9)

1.5. Арифметические действия над двоичными числами

Арифметика двоичной системы счисления основана на использовании таблиц сложения, вычитания и умножения. Эти таблицы чрезвычайно просты:

Таблица сложения

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

Таблица вычитания

0	–	0	=	0
1	–	0	=	1
1	–	1	=	1
10	–	1	=	1

Таблица умножения

0	*	0	=	0
0	*	1	=	0
1	*	0	=	0
1	*	1	=	1

Двоичное сложение. Двоичное сложение выполняется по тем же правилам, что и десятичное, с той лишь разницей, что перенос в следующий разряд производиться после того, как сумма достигнет не десяти, а двух.

Пример 1. *Сложение двоичных чисел* $(101101)_2$ *и* $(111110)_2$

$$
\begin{array}{ll}
+ & 101101 \\
 & 111110 \\
\hline
 & 010011 \text{ – поразрядная сумма без учета переносов} \\
 & \\
+ & 1011000 \text{ – переносы} \\
 & 0010011 \\
\hline
 & 1001011 \text{ – поразрядная сумма без учета повторных переносов} \\
 & \\
+ & 0100000 \text{ – повторные переносы} \\
 & 1001011 \\
\hline
 & 1101011 \text{ – окончательный результат}
\end{array}
$$

Легко произвести проверку:

$(101101)_2 = 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = 32 + 8 + 4 + 1 = (45)_{10}$,

$(111110)_2 = 1\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 = 32 + 16 + 8 + 4 + 2 = (62)_{10}$,

$(1101011)_2 = 1\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 64 + 32 + 8 + 2 = (107)_{10}$,

$(45)_{10} + (62)_{10} = (107)_{10}$.

Пример 2. *Сложение двоичных чисел* $(110{,}1011)_2$ *и* $(10111{,}10101)_2$

$$
\begin{array}{ll}
+ & 110{,}1011 \\
 & 10111{,}10101 \\
\hline
 & 10001{,}00011 \text{ – поразрядная сумма без учета переносов} \\
 & \\
+ & 11\ 1,\ 1 \text{ – переносы} \\
 & 10001{,}00011 \\
\hline
 & 11100{,}01011 \text{ – поразрядная сумма без учета повторных переносов} \\
 & \\
+ & 1\ , \text{ – повторные переносы} \\
 & 11100{,}01011 \\
\hline
 & 11110{,}01011 \text{ – окончательный результат}
\end{array}
$$

Сложение нескольких чисел вызывает некоторые трудности, так как в результате поразрядного сложения могут получится переносы, превышающие единицу.

Двоичное вычитание. Вычитание в двоичной системе выполняется аналогично вычитанию в десятичной системе счисления. При необходимости, когда в некотором разряде приходится вычитать единицу

из нуля, занимается единица из следующего старшего разряда. Если в следующем разряде нуль, то заем делается в ближайшем старшем разряде, в котором стоит единица. При этом следует понимать, что занимаемая единица равна двум единицам данного разряда, т. е. вычитание выполняется по следующему правилу:

Пример 3. *Вычитание двоичных чисел* $(11010{,}1011)_2$ *и* $(1101{,}01111)_2$

$$\begin{array}{r} 11010{,}1011 \\ -\ 1101{,}01111 \\ \hline 1101{,}00111 \end{array}$$

Конечно, математически вычитание выполнить несложно. Однако, если поступать таким образом, то к примеру в ЭВМ придется для выполнения сложения и вычитания иметь два блока: сумматор и вычитатель. Поэтому поступают следующим образом: вычитание можно представить как сложение положительного и отрицательного чисел, необходимо только подходящее представление для отрицательного числа.

Рассмотрим четырехразрядный десятичный счетчик, какие в автомобиле отсчитывают пройденный путь. Пусть он показывает число 2, если вращать его в обратном направлении, то сначала появится 1, затем 0, после 0 появится число 9999. Сложим, к примеру, 6 с этим числом:

$$\begin{array}{r} 6 \\ +\ 9999 \\ \hline 10005 \end{array}$$

Если пренебречь единицей переноса и считать 9999 аналогом –1, то получим верный результат: $6 + (-1) = 5$.

Число 9999 называется *десятичным дополнением* числа 1. Таким образом, в десятичной системе счисления отрицательные числа могут быть представлены в форме десятичного дополнения, а знак минус можно опустить.

Двоичное дополнение числа определяется как то число, которое будучи прибавлено к первоначальному числу, даст только единицу переноса в старшем разряде.

Пример 4. *Двоичное дополнение числа* $(10101111)_2$

$$\begin{array}{r} 010101111 \text{ – число} \\ +\ 101010001 \text{ – двоичное дополнение} \\ \hline 1000000000 \text{ – сумма} \end{array}$$

↑ – единица переноса

Для получения двоичного дополнения необходимо:

- получить обратный код, который образуется инвертированием каждого бита:

010101111 – число

101010000 – обратный код

- прибавить к обратному коду единицу, образовав таким образом дополнительный код:

$$\begin{array}{r} 101010000 \text{ – обратный код} \\ +\ 1 \\ \hline 101010001 \text{ – дополнительный код} \end{array}$$

Пример 5. *Вычитание в дополнительном коде*

$12 - 7$
$12_{10} = 01100_2$
$7_{10} = 00111_2$

11000 – обратный код,
11001 – дополнительный код.

$$\begin{array}{r} 01100 \\ +\ 11001 \\ \hline 100101 \end{array}$$

$100101_2=5_{10}$ (верно).

Двоичное умножение. Умножение двух двоичных чисел выполняется так же, как и умножение десятичных. Сначала получаются частичные произведения и затем их суммируют с учетом веса соответствующего разряда множителя.

Отличительной особенностью умножения в двоичной системе счисления является его простота, обусловленная простотой таблицы умножения. В соответствии с ней, каждое частичное произведение или равно нулю, если в соответствующем разряде множителя стоит нуль, или равно множимому, сдвинутому на соответствующее число разрядов, если в соответствующем разряде множителя стоит единица. Таким образом, операция умножения в двоичной системе сводится к операциям сдвига и сложения.

Умножение производится, начиная с младшего или старшего разряда множителя, что и определяет направление сдвига. Если сомножители имеют дробные части, то положение запятой в произведении определяется по тем же правилам, что и для десятичных чисел.

Пример 6. *Умножение двоичных чисел* $(101)_2$ *и* $(011)_2$

$$\begin{array}{l} 5 \cdot 3 \\ 101 \\ \underline{11} \\ 101 \\ \underline{101} \\ 1111 = 15_{10} \end{array}$$

Двоичное деление. Деление чисел в двоичной системе производится аналогично делению десятичных чисел. Рассмотрим деление двух целых чисел, так как делимое и делитель всегда могут быть приведены к такому виду путем перениесения запятой в делимом и делителе на одиноаковое число разрядов и дописывания необходимых нулей. Деление начинается с того, что от делимого слева отделяется минимальная группа разрядов, которая, рассматриваемая как число, превышает или равна делителю. Дальнейшие действия выполняются по обычным правилам,

причем последняя целая цифра частного получается тогда, когда все цифры делимого исчерпаны.

Таким образом, выполнение арифметических операций в двоичной системе счисления достаточно просто. Особенно просто выполнять операции сложения, вычитания и умножения. Благодоря этому, применение двоичной системы в вычислительных машинах позволяет упростить схемы устройств, в которых осуществляются операции над числами.

Пример 7. *Деление двоичных чисел*

1) 18:2

```
10010 | 10
10    | 1001=(9)
  00
  00
  001
  000
    10
    10
    00
```

2) 14:4

```
1110 | 100
100  | 11,1=(3,5)
  110
  100
   100
   100
     0
```

1.5. Прямой, обратный и дополнительный коды. Модифицированный код

При рассмотрении элементарных арифметических операций над двоичными числами мы уже коснулись темы отрицательных двоичных чисел. Теперь рассмотрим ее подробнее.

Для кодирования знака двоичного числа используется старший ("*знаковый*") разряд (ноль соответствует плюсу, единица – минусу).

Такая форма представления числа называется *прямым кодом*.

В ЭВМ прямой код применяется только для представления положительных двоичных чисел. Для представления отрицательных чисел применяется либо дополнительный, либо обратный код, так как над отрицательными числами в прямом коде неудобно выполнять арифметические операции.

Правила для образования дополнительного и обратного кода состоят в следующем:

- для образования дополнительного кода отрицательного числа необходимо в знаковом разряде поставить единицу, а все цифровые разряды инвертировать (заменить 1 на 0, а 0 – на 1), после чего прибавить 1 к младшему разряду;
- для образования обратного кода отрицательного числа необходимо в знаковом разряде поставить единицу, а все цифровые разряды инвертировать;
- при данных преобразованиях нужно учитывать размер разрядной сетки.

Прямой код можно получить из дополнительного и обратного по тем же правилам, которые служат для нахождения дополнительного и обратного кодов.

В таблице 7 приведены десятичные числа и их двоичные представления в трех различных формах. Интересно в ней вот что. Если начать счет с числа 1000 (–8) и двигаться вниз по столбцам, то в дополнительном коде каждое последующее число получается прибавлением единицы к предыдущему без учета переноса за пределы четвертого разряда. Так просто эту операцию в прямом и обратном кодах не осуществить. Эта особенность дополнительного кода и явилось причиной предпочтительного применения его в современных ЭВМ.

Итак, числа, представленные в дополнительном коде, складываются по правилам двоичного сложения, но без учета каких либо переносов за пределы старшего разряда. Рассмотрим это на примерах.

Таблица 7. *Прямой, обратный и дополнительный коды*

Десятичное число	Прямой код	Обратный код	Дополнительный код
-8	–	–	1000
-7	1111	1000	1001
-6	1110	1001	1010
-5	1101	1010	1011
-4	1100	1011	1110
-3	1011	1100	1101
-2	1010	1101	1110
-1	1001	1110	1111
0	1000 0000	1111 0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

Пример 1. *Двоичное сложение в дополнительном коде*

1)	+2	0010	2)	-2	1110	3)	+5	0101
+		+		+		+		+		+	
	+5	0101		-6	1010		-4	1100
	+7	0111		-8	1000		+1	0001

Еще одним достоинством дополнительного кода является то, что нуль, в отличие от прямого и обратного кодов, представляется одним кодом. Наличие 0 в знаковом бите при представлении нуля определяет его как величину положительную, что согласуется с математической теорией чисел и соглашениями, принятыми во всех языках программирования.

Из приведенных примеров следует, что положительные числа в прямом, обратном и дополнительном кодах совпадают. В прямом и обратном коде нуль имеет два представления – «положительный» и «отрицательный» нуль.

Отметим, что при представлении с плавающей запятой отдельно кодируется мантисса и порядок числа. При этом возможно представление мантисс и порядков чисел в одном и том же или разных кодах. Например, порядок числа может быть представлен в прямом, а мантисса – в дополнительном кодах и т. п.

Таким образом, используя обратный и дополнительный коды, операцию алгебраического сложения можно свести к арифметическому сложению кодов чисел, которое распространяется и на разряды знаков, которые рассматриваются как разряды целой части числа.

При сложении чисел, меньших единицы, в машине быть получены числа, по абсолютной величине большие единицы. Для обнаружения переполнения разрядной сетки в ЭВМ применяются *модифицированные прямой, обратный и дополнительный коды*. В этих кодах знак кодируется двумя разрядами, причем знаку "плюс" соответствует комбинация 00, а знаку "минус" - комбинация 11.

Правила сложения для модифицированных кодов те же, что и для обычных. Единица переноса из старшего знакового разряда в модифицированном дополнительном коде отбрасывается, а в модифицированном обратном коде передается в младший цифровой разряд.

Признаком переполнения служит появление в знаковом разряде суммы комбинации 01 при сложении положительных чисел (*положительное переполнение*) или 10 при сложении отрицательных чисел (*отрицательное переполнение*). Старший знаковый разряд в этих случаях содержит истинное значение знака суммы, а младший является старшей значащей цифрой числа. Для коррекции переполнения число нужно сдвинуть в разрядной сетке на один разряд вправо, а в освободившийся старший знаковый разряд поместить цифру, равную новому значению младшего знакового разряда. После корректировки переполнения мантиссы результата необходимо увеличить на единицу порядок результата.

1.7. Особенности сложения чисел в двоично-десятичных кодах

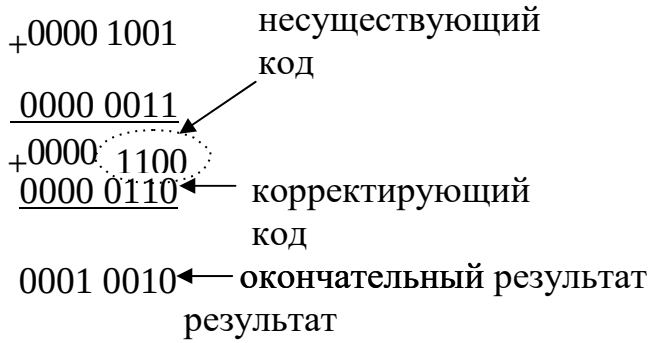
Поскольку в быту мы привыкли пользоваться десятичной системой счисления, то ввод информации в ЭВМ и вывод ее из ЭВМ осуществляется, в основном, в десятичной системе счисления. Мы вводим десятичные числа, которые преобразуются в двоичные, над которыми производятся арифметические операции, а результаты перед выводом преобразуются вновь в десятичные цифры. Десятичные цифры в ЭВМ представляются в виде двоично-десятичных кодов в формате 8421 или с избытком 3. В некоторых ЭВМ используют арифметико-логические устройства, которые выполняют арифметические операции непосредственно в двоично-десятичных кодах. Поскольку все арифметические операции в ЭВМ, в

конечном счете, сводятся к сложению, рассмотрим особенности выполнения сложения в двоично-десятичных кодах.

Рассмотрим сложение целых чисел сначала в формате 8421. Здесь следует рассмотреть две характерные ситуации: первая ситуация, когда в десятичном разряде суммы формируется несуществующий код и вторая ситуация, когда из десятичного разряда суммы происходит естественный перенос:

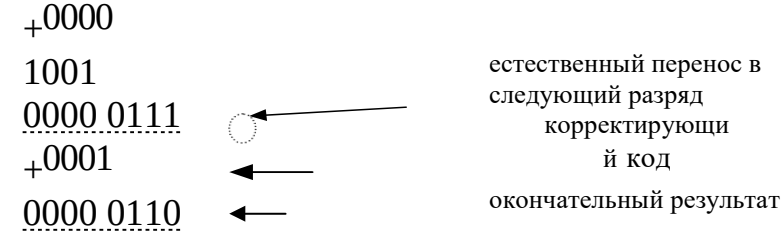
Десятичная система счисления Двоично-десятичная система счисления

09 + 03 = 12



Десятичная система счисления Двоично-десятичная

09 +07=16



Как видно из приведенных примеров первоначально получается неверный результат, который требуется скорректировать. В первом случае необходимо осуществить искусственный перенос из разряда, где сформировался несуществующий код и, чтобы сохранить результат, из этого разряда нужно вычесть десять (единица следующего по старшинству разряда весит десять единиц) или прибавить шестерку, так как она представляет дополнительный код десятки. Следует отметить, что при добавлении шестерки искусственный перенос формируется автоматически. Поэтому достаточно просто не блокировать межразрядный перенос при добавлении шестерки.

Во втором случае происходит естественный перенос в следующий по старшинству разряд, вес которого оценивается как десятка, хотя он реально имеет вес шестнадцать, то есть не хватает шесть единиц в разряде, из которого он произошел. Поэтому эти шесть единиц необходимо добавить в этот разряд.

Сформулируем теперь общее правило сложения двоично-десятичных чисел в формате 8421. Сложение производится в два этапа:

- сначала числа складываются как обычные двоичные числа, и получается промежуточный результат;
- затем производится коррекция промежуточного результата путем добавления корректирующего кода. К тем разрядам, в которых сформировался несуществующий код или естественный перенос, добавляются шестерки (0110), и межразрядный перенос не блокируется. Если таких разрядов нет, то корректирующий код будет равен нулю.

Аналогично можно рассмотреть и для других кодов.

1.8. Кодирование алфавитно-цифровой информации

Помимо чисел в современных ЭВМ обрабатывается и алфавитно-цифровая информация, содержащая цифры, буквы, знаки препинания, математические и другие символы, с помощью которых удобно представлять экономическую, планово-производственную, учетную, бухгалтерскую, статистическую и другую информацию. Совокупность всех символов, используемых в вычислительной системе, представляет ее алфавит. Символу соответствует машинная единица информации – слова. Он представляет группу двоичных разрядов, служащую для представления символа в машине (двоичный код символа). Существует много способов кодирования символов. Выбор способа кодирования определяется объемом алфавита символов, а также требованиями, связанными с облегчением автоматической обработки данных.

Для представления символов ранее широко применялся шестиразрядный слов, но он позволял представить только 64 различных символа, что оказалось недостаточным. В ЭВМ третьего поколения (ЕС ЭВМ) использовалось байтовое представление алфавитно-цифровых символов, то есть можно было закодировать 256 различных символов.

В качестве внутреннего кода для представления алфавитно-цифровых символов в памяти применялся двоичный код для обработки информации (ДКОИ) или код обмена информацией (КОИ-8).

В ЭВМ четвертого поколения используются различные системы кодирования. В первых моделях современных ПК использовался 8-ми разрядный код ASCII или код обмена информацией (КОИ-8). Эти коды между собой существенно не отличаются. В последних версиях ПК в качестве внутреннего формата для хранения и обработки текстовых строк используется Unicode – стандартная кодировка, которая поддерживает многие известные в мире наборы символов и в которой каждый символ представляется 16-ти битным (двухбайтовым) кодом. Если запускается приложение, работающее с 8-ми битной кодировкой символов, то входные строковые параметры перед обработкой преобразуются в Unicode, а после обработки преобразуются к исходной системе кодирования.

1.9. Формы представления числовой информации

В двоичной системе счисления используются только два символа 0 (нуль) и 1 (единица), что хорошо согласуется с техническими характеристиками цифровых схем. Действительно очень удобно представлять отдельные составляющие информации с помощью двух состояний:

- Отверстие есть или отсутствует (перфолента или перфокарта);
- Материал намагничен или размагничен (магнитные ленты, диски);
- Уровень сигнала большой или маленький.

Это связанно с тем, что для физической реализации двоичной информации наиболее удобно использовать логические элементы с двумя устойчивыми состояниями. Эти состояния обозначают двоичными символами: 0 и 1, т.е. сигнал есть или нет сигнала. Каждый из этих символов называют битом, и они являются минимальными двоичными единицами информации. Если объединит два или более символов образуется слово. С помощью одного бита можно выразить всего два понятия. Если количество бит в одном слове увеличить до двух, то можно закодировать 4 понятия. В современных компьютерах принято работать минимальными словам, содержащих 8 бит. Такое двоичное слова называют **байтом.**

Крайний слева бит числа называют *старшим разрядом* (он имеет наибольший вес), крайний справа – *младшим разрядом* (он имеет наименьший вес).

Многие типы ЭВМ и дискретных систем управления перерабатывают информацию порциями (словами) по 8, 16, 32, 64 или 80 бита (1, 2, 4,8 или 10 байта). Двоичное слово, состоящее из двух байт, показано на рис. 3.

На практике существуют различные способы записи чисел.

Например число 0,028 можно записать так: $28 \cdot 10^{-3}$, или 0,03 (с округлением), или $2{,}8 \cdot 10^{-2}$ и т. д. Разнообразие форм в записи одного числа, независимо от системы счисления, может послужить причиной затруднений для работы цифрового устройств (или автомата). Во избежание этого

нужно либо создать специальные алгоритмы распознавания числа, либо указывать каждый раз форму его записи. Второй путь проще.

Рис. 3. *Слово из двух байтов*

В ЭВМ используются основные две формы представления чисел: с фиксированной запятой и с плавающей запятой. При представлении чисел с фиксированной запятой положение запятой фиксируется в определенном месте относительно разрядов числа. Запятая может быть зафиксирована перед старшим разрядом (слева), после младшего разряда (справа) и между конкретными разрядами числа. При представлении чисел с плавающей запятой используются два варианта: со смещенным порядком и несмещенным порядком. Граф-схема на рис.4. отображает варианты представления чисел в ЭВМ.

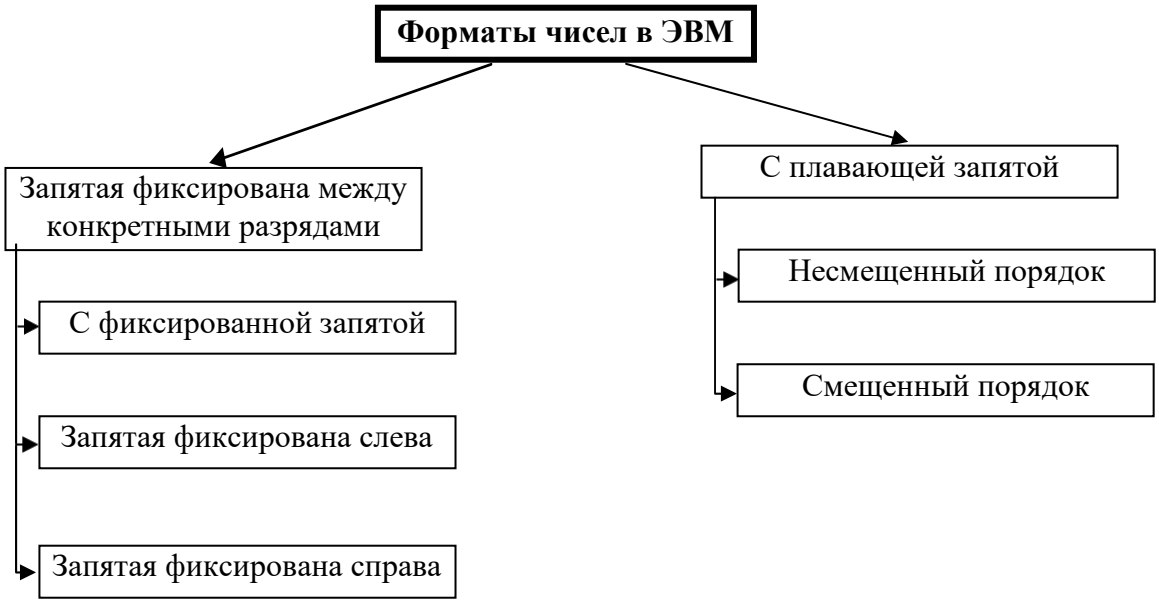


Рис.4.

Если запятая фиксирована слева, то могут быть представлены только числа по модулю меньшие 1, а при фиксации запятой справа – только целые числа. На рис. 5 представлены форматы чисел с фиксированной запятой для n-разрядного машинного слова. Здесь a, b_i, g_i Î{0,1}, i=1,…, n-1. Для положительного числа a=0, для отрицательного числа a=1. Числа на рис. 5 а), б), в) соответственно интерпретируются следующим образом:

При естественной форме число записывается в естественном натуральном виде, например 12560 — целое число; 0,003572 — правильная дробь; 4,89760 — неправильная дробь. При экспоненциальной форме запись одного числа может принимать разный вид в зависимости от

ограничений, накладываемых на ее форму. Например, число 12560 может быть записано так:
$12560 = 1{,}256 \cdot 10^4 = 0{,}1256 \cdot 10^5 = 125600 \cdot 10^{-1}$ и т. д.

Автоматное (машинное) изображение числа— представление числа A в разрядной сетке цифрового автомата. Условно обозначим автоматное изображение числа символом $[A]$. Тогда справедливо соотношение: $A = [A]K_A$, где K_A — коэффициент, величина которого зависит от формы представления числа в автомате.

Представление чисел с фиксированной запятой (точкой). Естественная форма представления числа в ЭВМ (цифровом автомате) характеризуется тем, что положение его разрядов в автоматном изображении остается всегда постоянным независимо от величины самого числа. Существует также другое название этой формы записи чисел — представление чисел с фиксированной запятой (точкой).

Чтобы упростить функционирование цифрового автомата, необходимо ограничить входную информацию какой-то одной областью чисел (на вход автомата желательно подавать либо целые числа, либо правильные дроби, либо любые числа), что позволит определить значения масштабного коэффициента K_A. Например, если на вход цифрового автомата поступают только правильные дроби, то

$$-1 < [A]_\phi < 1, \qquad (10)$$

где $[A]_\phi$ — машинное изображение числа для формы представления с фиксированной запятой.

Тогда число A будет представлено в виде $A = [A]_\phi K_A$.

Величина масштабного коэффициента K_A, удовлетворяющего условию (10), определяет тот факт, что в машинном изображении запятая всегда стоит после целой части дроби, т. е. перед ее старшим разрядом. Следовательно, можно хранить только дробную часть числа (цифровую часть), а в разряде целой части писать дополнительную информацию.

Так как числа бывают положительные и отрицательные, то формат (разрядная сетка) автоматного изображения разбивается на знаковую часть и поле числа (рис.5, *а).* В знаковую часть записывается информация о знаке. Примем, что знак положительного числа «+» изображается символом 0, а знак отрицательного числа «-» изображается символом 1.

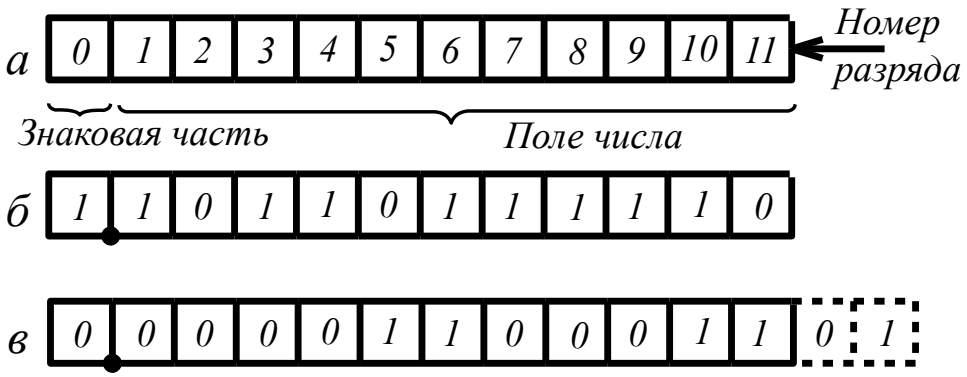


Рис. 5. Представление чисел в форме с фиксированной запятой

Если на вход цифрового автомата поступают целые числа, то в разрядной сетке (в формате машинного изображения) один разряд отводится под знак числа, а последующие разряды образуют поле числа. Диапазон представимых чисел в этом случае от - (2^n -1) до + (2^n -1), где n — количество разрядов без знаковой части.

Задачу выбора масштабного коэффициента K_A усложняет необходимость сохранять соответствие разрядов всех чисел, которыми оперирует цифровой автомат. Пусть имеется цифровой автомат с разрядной сеткой длиной 12 двоичных разрядов (рис. 5, а). Надо определить масштабный коэффициент для чисел A_1 = - $1011{,}0111110_2$ и A_2 = $0{,}110001101_2$.

Для того чтобы выполнить условие (19), необходимо число, большее по абсолютному значению, записать в виде $A_1 = -0{,}10110111110 \cdot 2^4$. Отсюда $[A_1]_ф$ = 1,10110111110, что соответствует величине масштабного коэффициента $K_{A_1} = 2^4$. Число A_2 должно войти в разрядную сетку автомата с сохранением соответствия разрядов, т. е. $K_{A_2} = K_{A_1}$. Следовательно, A_2 = $+0{,}0000110001101 \cdot 2^4$ или $[A_2]_ф$ = 0,00001100011 (рис. 5, *б, в*).

Из примера видно, что представление чисел в форме с фиксированной запятой может привести к погрешности представления. Так, для числа A_2 абсолютная погрешность представления оценивается величиной части числа, не уместившейся в разрядную сетку, т. е. величиной $0{,}0000000000001 \cdot 2^4$. В некоторых случаях очень малые числа представляются в машине изображением, называемым ***машинным нулем***. Следовательно, ошибка представления зависит от правильности выбора масштабных коэффициентов. Вычисление последних должно проводиться таким образом, чтобы исключить возможность появления в процессе функционирования автомата чисел, машинные изображения которых не удовлетворяют условию (10). Если в результате операции появится число, по абсолютному значению большее единицы, то возникает переполнение разрядной сетки автомата, что нарушает нормальное функционирование цифрового автомата.

Использование представления чисел с фиксированной запятой позволяет упростить схемы машины, повысить ее быстродействие, но представляет определенные трудности при программировании. В настоящее время представление чисел с фиксированной запятой используется как основное только в микроконтроллерах.

Достоинство представления чисел в форме с фиксированной запятой состоит в простоте выполнения арифметических операций.

Недостатки – в необходимости выбора масштабных коэффициентов и в низкой точности представления с малыми значениями модуля (нули в старших разрядах модуля приводит к уменьшению количества разрядов, занимаемых значащей частью модуля числа).

Представление чисел в форме с плавающей запятой. Число в формате с плавающей запятой называется нормализованным, если в старшем разряде мантиссы не стоит ноль. Для двоичной системы счисления число будет нормализованным, если $1 > |q_x| \geq 1/2$, то есть в старшем разряде

мантиссы стоит 1. При фиксированном количестве разрядов, отводимых под мантиссу, максимальная точность достигается для нормализованного числа.

В случае представления чисел со смещенным порядком при записи числа в память, к его порядку прибавляется целое число – смещение $N=2^k$, где k - число двоичных разрядов, отведенных для представления модуля порядка. Смещенный порядок всегда положителен. При выводе чисел порядок восстанавливается.

Формат чисел с плавающей запятой позволяет расширить диапазон представления чисел по сравнению с форматом с фиксированной запятой при одинаковой разрядности машинного слова. Рассмотрим пример для случая, когда машинное слово содержит 16 разрядов. Если использовать формат с фиксированной справа запятой, то для числа со знаком диапазон представления чисел составит от - $(2^{15}-1)$ до + $(2^{15}-1)$. Для случая с плавающей запятой, когда два разряда отведены под знаки мантиссы и порядка, а для мантиссы выделено 8 разрядов и 6 – для порядка, диапазон представления чисел составит от $-(1-2^{-8})\times 2^{63}$ до $+(1-2^{-8})\times 2^{63}$. Как видно из рассматриваемого примера диапазоны отличаются на 48 двоичных порядков в пользу формата с плавающей запятой.

Кроме того, форматы с фиксированной справа и слева запятой имеют еще один недостаток: при обработке действительных чисел приходится проводить масштабирование при вводе и выводе данных. Например, если необходимо выполнить арифметические действия с использованием формата с фиксированной справа запятой над числами 1,5; 0,45; 15, то их придется сначала промасштабировать, то есть привести к целым: 150; 45; 1500 (сдвинуть влево на два разряда). Затем можно проводить над ними операции сложения и вычитания: 150 + 45 = 195; 1500 - 150 = 1350. При выводе числа 195 и 1350 необходимо предварительно сдвинуть вправо на два разряда, что даст соответственно: 1,95 и 13,5.

В экспоненциальной форме

$$A_H = m_A q^{P_A} \qquad (11)$$

где m_A —мантисса числа (или количество значащих разрядов числа) A; p_A — порядок числа A.

Как видно из ранее изложенного, такое представление чисел не однозначно; для определенности обычно вводят некоторые ограничения.

Наиболее распространено и удобно для представления в ЭВМ ограничение вида

$$q^{-1} \le \left| m_A \right| < 1, \qquad (12)$$

где q — основание системы счисления.

Нормализованная форма представления чисел — форма представления чисел, для которой справедливо условие (12).

Поскольку в этом случае абсолютное значение мантиссы лежит в пределах от q^{-1} до $1 - q^{-n}$, где n — количество разрядов для изображения мантиссы без знака, положение разрядов числа в его автоматном изображении не постоянно. Поэтому такую форму представления чисел называют также ***формой представления с плавающей запятой***. Формат машинного изображения числа с плавающей запятой должен содержать знаковые части и поля для мантиссы и порядка (рис. 6, *а).* Выделяются

специальные разряды для изображения знака числа (мантиссы) и знака порядка или характеристики (рис. 6, *а, б).* Кодирование знаков остается таким же, как было с фиксированной запятой.

Рассмотрим пример записи чисел в форме с плавающей запятой. Пусть в разрядную сетку цифрового автомата (рис. 6) необходимо записать двоичные числа A_1 = -10110,1111$_2$ и A_2 = +0,000110010111$_2$.

Прежде всего эти числа необходимо записать в нормальной форме (рис. 6, *в, г).* Порядок чисел выбирают таким образом, чтобы для них выполнялось условие (21), т. е. $A_1 = -0{,}101101111 \cdot 2^5$ и $A_2 = +0{,}110010111 \cdot 2^{-3}$, он должен быть записан в двоичной системе счисления. Так как система счисления для заданного автомата остается постоянной, то нет необходимости указывать ее основание, достаточно лишь представить показатель степени.

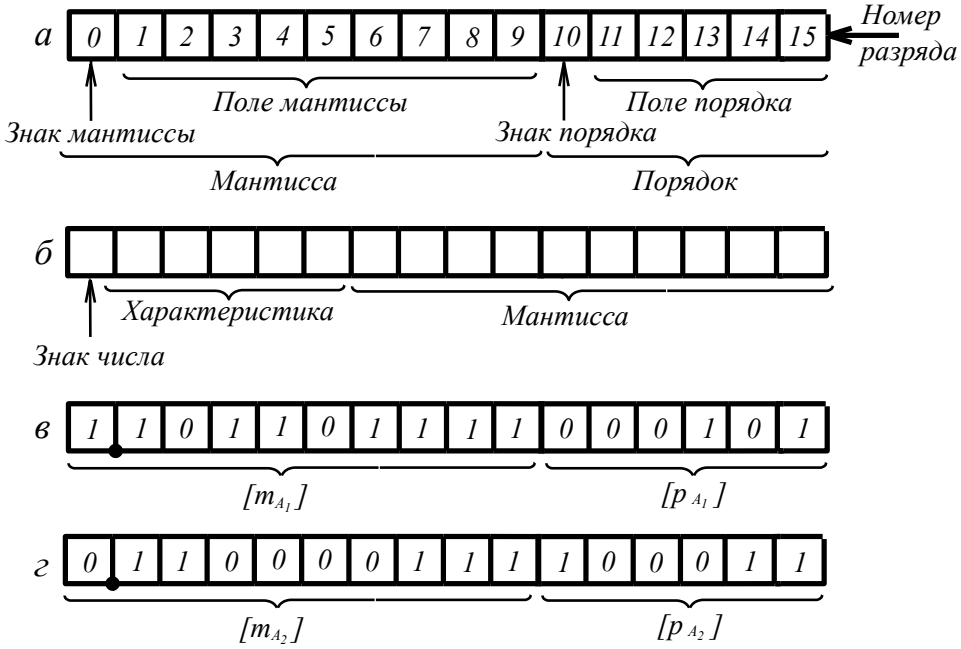


Рис. 6. Представление чисел в форме с плавающей запятой

Поскольку для изображения порядка выделено пять цифровых разрядов и один разряд для знака, их машинные изображения и машинные изображения их мантисс соответственно

$$[\,p_{A_1}\,] = 000101;\ [\,p_{A_2}\,].= 00011;$$

$$[\,m_{A_1}\,] = 1{,}101101111;\ [\,m_{A_2}\,] = 0{,}110010111.$$

В универсальных ЭВМ основным является представление чисел с плавающей запятой. Широкий диапазон представления чисел с плавающей запятой удобен для научных и инженерных расчетов. Для повышения точности вычислений во многих ЭВМ предусмотрена возможность использования формата двойной длины, однако при этом происходит увеличение затрат памяти на хранение данных и замедляются вычисления.

1.10. Погрешности представления числовой информации

Представление числовой информации в цифровом автомате, как правило, влечет за собой появление погрешностей (ошибок), величина которых зависит от формы представления чисел и от длины разрядной сетки автомата.

Абсолютная погрешность представления — разность между истинным значением входной величины A и ее значением, полученным из машинного изображения $A_м$, т. е. $\Delta[A] = A - A_м$.

Относительная погрешность представления — величина

$$\delta[A] = \Delta[A]/A_м . \qquad (13)$$

Входные величины независимо от количества значащих цифр могут содержать грубые ошибки, возникающие из-за опечаток, ошибочных отсчетов показаний каких-либо приборов, некорректной постановки задачи или отсутствия более полной и точной информации. Например, часто принимают $\pi = 3{,}14$. Однако эта величина может быть получена с более высокой точностью. Если принять, что точное значение $\pi = 3{,}14139265$, то абсолютная погрешность равна $\Delta[\pi] = 0{,}00159265$.

Часто некоторая величина в одной системе счисления имеет конечное значение, а в другой системе счисления становится бесконечной величиной, например, дробь 1/10 имеет конечное десятичное представление, но, будучи переведена в двоичную систему счисления, становится бесконечной дробью 0,00011000110011... .

Следовательно, при переводе чисел из одной системы счисления в другую неизбежно возникают погрешности, оценить которые нетрудно, если известны истинные значения входных чисел.

В соответствии с (10) числа изображаются в машине в виде $A_q = [A]K_A$, где масштабный коэффициент K_A выбирают так, чтобы абсолютное значение машинного изображения числа A в системе счисления с основанием $q = 2$ было всегда меньше 1: $A_q = K_A [a_{-1}q^{-1} + a_{-2}q^{-2} + ... + a_{-n}q^{-n} + \ldots]$.

Так как длина разрядной сетки автомата равна n двоичных разрядов после запятой, абсолютная погрешность перевода десятичной информации в систему с основанием q будет

$$\Delta[A] = a_{-(n+1)}q^{-(n+1)} + ... + a_{-(n+s)}q^{-(n+s)} + ... = \sum_{i=-(n+1)}^{-\infty} a_i q^i . \qquad (14)$$

Если $q = 2$, то при $a_i = 1$ максимальное значение этой погрешности

$$\Delta[A]_{\max} = \sum_{i=-(n+1)}^{-\infty} 1 \cdot 2^i = 2^{-n} \sum_{i=-1}^{-\infty} 2^i = 2^{-n} \qquad (15)$$

Из (15) следует, что максимальная погрешность при переводе десятичной информации в двоичную не будет превышать единицы младшего разряда двоичного числа. Минимальная погрешность перевода равна нулю.

Усредненная абсолютная погрешность перевода чисел в двоичную систему счисления $\Delta[A] = (0 + 2^{-n})/2 = 0{,}5 \cdot 2^{-n}$.

Для представления чисел в форме с фиксированной запятой абсолютное значение машинного изображения числа

$$2^{-n} \le \left|[A]_\phi\right| \le 1 - 2^{-n}. \qquad (16)$$

Следовательно, относительные погрешности представления для минимального значения числа

$$\delta[A]_{\phi\min} = \Delta[A]/[A]_{\phi\max} = 0{,}5\cdot 2^{-n}/(1-2^{-n})..$$

Для ЭВМ, как правило, n = 16...64, поэтому $1>>2^{-n}$, откуда $\delta[A]_{\phi\min} = 0{,}5\cdot 2^{-n}$.

Аналогично, для максимального значения:

$$\delta[A]_{\phi\max} = \Delta[A]_{\phi\max}/[A]_{\phi\min} = 0{,}5\cdot 2^{-n}/2^{-n}. \qquad (17)$$

Из (17) видно, что погрешности представления малых чисел в форме с фиксированной запятой могут быть очень значительными.

Для представления чисел в форме с плавающей запятой абсолютное значение мантиссы

$$2^{-1} \le \left|[mA]_n\right| \le 1-2^{-n}. \qquad (18)$$

Погрешность (15) — погрешность мантиссы. Для нахождения погрешности представления числа в форме с плавающей запятой величину этой погрешности надо умножить на величину порядка числа p_A:

$$\begin{aligned}\delta[A]_{n\max} &= 0{,}5\cdot 2^{-n}\, p_A/2^{-1}\cdot p_A = 2^{-n};\\ \delta[A]_{n\min} &= 0{,}5\cdot 2^{-n}\, p_A/(1-2^{-n})\, p_A,\end{aligned} \qquad (19)$$

где n — количество разрядов для представления мантиссы числа.

Из (19) следует, что относительная точность представления чисел в форме с плавающей запятой почти не зависит от величины числа.

1.11. Контрольные вопросы

1. Что понимают под термином «система счисления»?
2. Какие системы счисления называют позиционными?
3. Сформулируйте правила перевода чисел из одной позиционной системы в другую.
4. Приведите примеры непозиционных систем счисления.
5. Составьте таблицы двоичного сложения, вычитания и умножения.
6. Как в ЭВМ представляются числа с фиксированной запятой?
7. Как в ЭВМ представляются числа с плавающей запятой?
8. В каком формате диапазон представления чисел шире: с фиксированной запятой или с плавающей?
9. Как получить обратный код двоичного числа?
10. Как получить дополнительный код двоичного числа?
11. Сформулируйте правила сложения чисел в обратном и дополнительном кодах.
12. В чем преимущество использования обратного и дополнительного кода при выполнении операций сложения и вычитания?
13. Как кодируются символы в ЭВМ?

2. Булево алгебра и цифровая вычислительная техника

2.1 Физические формы представления информации в ЭВМ.

Ранее было отмечено, что в ЭВМ используется двоичная система счисления и системы счисления, построенные на основе двоичной системы счисления. Физическими аналогами знаков 0 и 1 в цифровых электронных схемах служат сигналы, которые могут принимать два хорошо различимых значения. Если для представления 0 и 1 используется уровень напряжения в электрической цепи, то говорят о потенциальном способе представления информации. Если единичное и нулевое значение двоичной переменной изображается наличием и отсутствием электрического импульса в цепи, то способ называется импульсным. В цифровых устройствах переменные и соответствующие им сигналы изменяются в дискретные моменты времени, то есть время разбивается на интервалы и в каждом таком временном интервале переменная имеет постоянное значение. По окончании текущего интервала двоичная переменная может менять свое значение или повторять предыдущее.

Временной интервал между двумя соседними моментами дискретного времени называются тактом или периодом представления информации.

Такты могут быть произвольной величины. Чем их меньше, тем больше быстродействие компьютера.

Информация может быть представлена последовательным или параллельным способом (последовательным или параллельным кодом).

При последовательном способе каждый временной интервал предназначен для отображения одного разряда информационного кода. В этом случае для передачи кода используется одна электрическая цепь.

При параллельном способе представления информации в каждом временном интервале представляются все разряды информационного кода. В этом случае используется количество электрических цепей, равное разрядности кода.

На рис. 7 изображено изменение во времени напряжения в электрической цепи при передаче последовательного двоичного кода при потенциальном а) и импульсном б) представлении информации. При потенциальном представлении обычно единице соответствует высокий уровень сигнала, который может быть как положительным (рис. 7. а)), так и отрицательным, а нулю соответствует низкий уровень сигнала. В случае импульсного представления единице соответствует фронт импульса (переход потенциала из низкого уровня в высокий или наоборот), а нулю – отсутствие фронта.

Изменение во времени напряжения в электрических цепях, по которым происходит передача параллельного двоичного кода, показано на рис. 8. для потенциального и импульсного представлении информации. Здесь двоичный код передается за время одного дискрета времени по нескольким электрическим цепям, количество которых определяется разрядностью передаваемого кода.

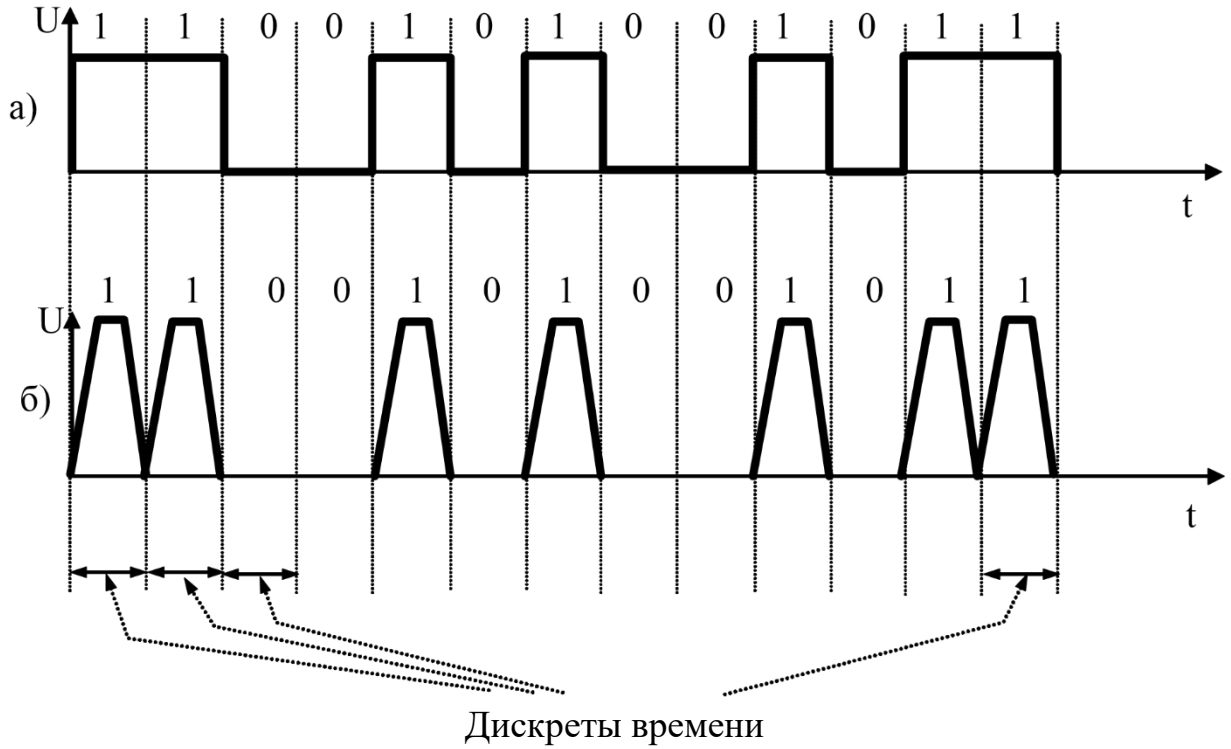


Рис. 7. Потенциальное а) и импульсное б) представление последовательного двоичного кода.

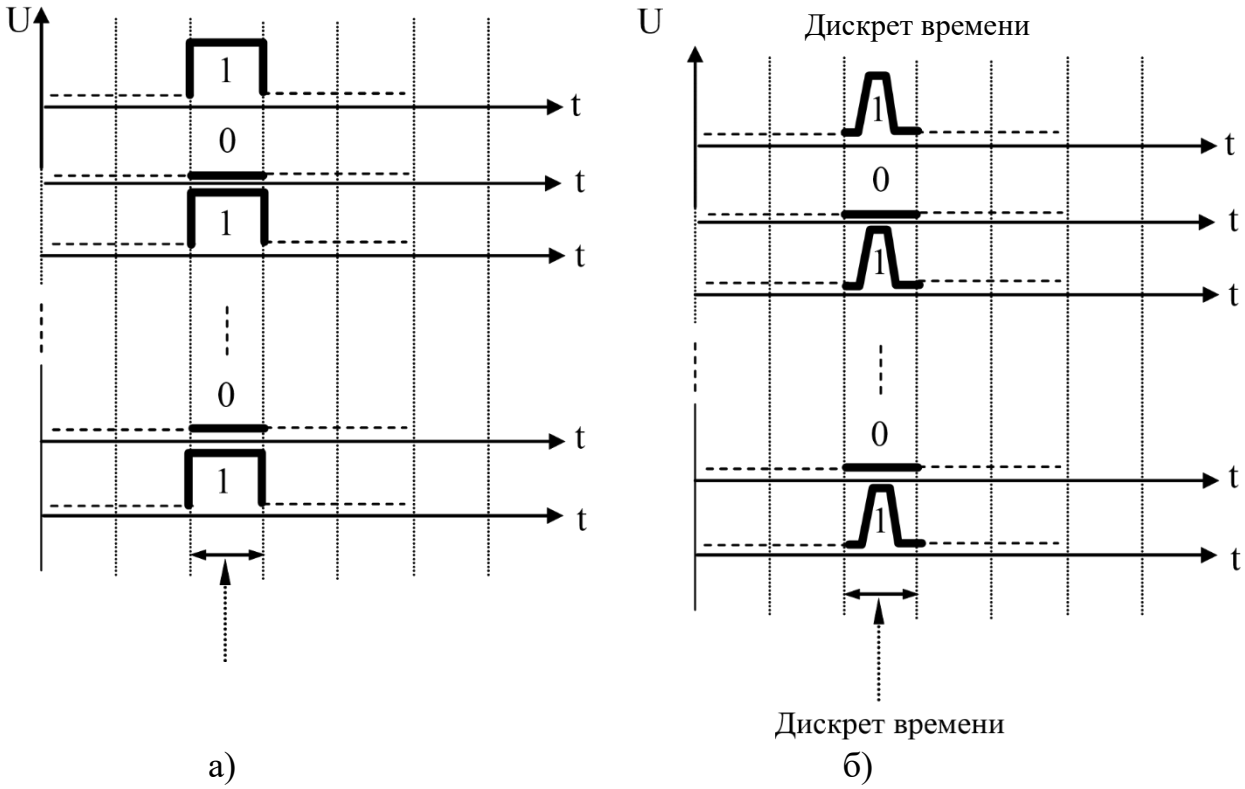


Рис. 8. Потенциальное а) и импульсное б) представление параллельного двоичного кода.

В ряде цифровых специализированных устройств используется также манипуляция фазы сигнала для представления нуля и единицы. На рис.9. изображена временная диаграмма сигнала, используемого для

представления последовательного двоичного кода, где применяется фазовая манипуляция. Фаза сигнала нуля и единицы отличаются на 180^{o}.

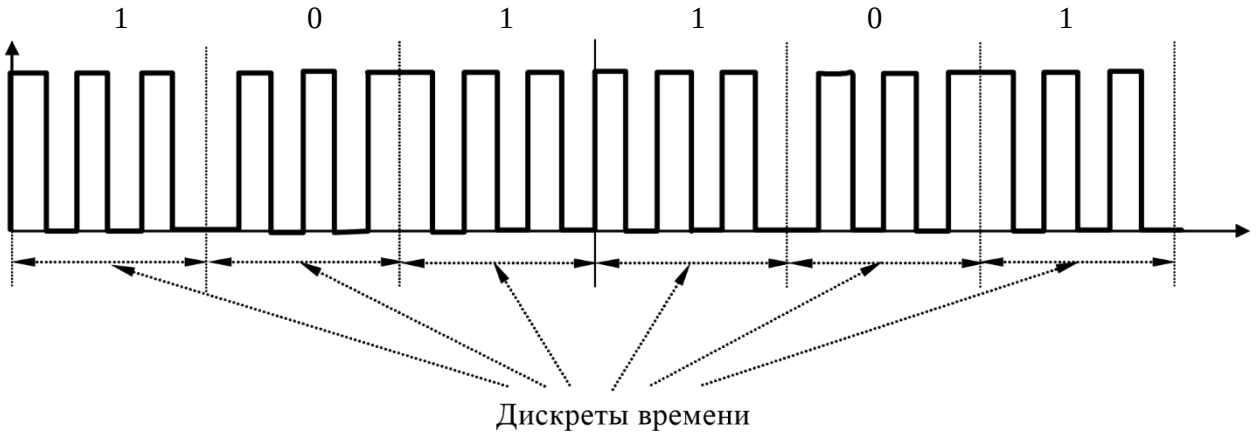


Рис. 9. Использование фазовой манипуляции для представления последовательного двоичного кода.

Используется также относительная фазовая манипуляция, когда изменение фазы происходит, если передается единица и – не происходит, если передается ноль. Импульсное представление информации и фазовая манипуляция не характерны для внутреннего представления информации в ЭВМ, где в основном используется потенциальное представление информации, а используются обычно при передаче данных по линиям связи.

2.2. Понятие цифровом автомате и булевых функциях

Преобразование информации в электронных вычислительных машинах производится электронными устройствами (логическими схемами) двух классов: комбинационными схемами и цифровыми автоматами с памятью.

На вход этих устройств поступает входные сигналы из некоторого конечного набора символов или букв. Выходные сигналы также принимают значения из конечного набора символов или букв. Обычно, как мы уже отметили в качестве этих символов или букв используется двоичные числа.

В комбинационных схемах совокупность выходных сигналов (выходное слово) в дискретный момент времени однозначно определяется только входными сигналами (входным словом), поступившими на входы в тот же дискретный момент времени. Закон функционирования комбинационной схемы определен, если задано соответствие между словами ее входного и выходного алфавитов.

Цифровой автомат памятью в отличие от комбинационной схемы имеет некоторое конечное число различных внутренних состояний. Под воздействием входного слова цифровой автомат переходит из одного

состояния в другое и выдает выходное слово. Выходное слово на выходе цифрового автомата в дискретный момент времени определяется входным словом, поступившим в этот момент времени на вход автомата, и внутренним состоянием автомата, которое явилось результатом воздействия на автомат входных слов в предыдущие дискретные моменты времени.

Цифровой автомат обязательно содержит память, состоящую из запоминающих элементов (триггеров, элементов задержки и др.) фиксирующих состояние, в котором он находится. Цифровой автомат всегда можно разбить на комбинационную схему и на элементы памяти.

Для описания законов функционирования комбинационных схем используется математический аппарат булевой функции, который однозначно устанавливает связи между входами и выходами.

Большой вклад в ее развитие внесли работы английского математика Дж. Буля (1847 г.), русского математика П.С. Порецкого (1884 г.), немецких математиков Э. Шредера (1890 – 1905 г.г.) и Г. Фреге (1879 – 1884 г.г.), итальянского математика Дж. Пеано (1894 г.) и др. В последствии Пеано и Фреге стали применять математическую логику к вопросам оснований арифметики и теории множеств. В разработке математической логики приняли участие советские математики И.И. Жегалкин, В.И. Гливенко, А.Н. Колмогоров, П.С. Новиков и др. Таким образом, теоретические основы цифровой вычислительной техники, использующей двоичную арифметику, бы- ли уже разработаны к моменту появления первых ЭВМ. Поставив значениям логических переменных «ложь» и «истина» в соответствие 0 и 1 получили готовый математический аппарат для двоичной цифровой схемотехники.

Преобразование информации в ЭВМ производится электронными устройствами двух классов: комбинационными схемами и цифровыми автоматами. Для описания законов функционирования комбинационных схем используется аппарат булевых функций.

Переменные, которые могут принимать только два значения (0 или 1), называются двоичными. Функция от двоичных переменных, принимающая также только два значения (0 или 1) называется булевой.

Булева функция описывается с помощью таблицы истинности, то есть таблицей ее значений в зависимости от значений аргументов. В таблице 8. приведены булевы функции одной и двух переменных.

Основное понятия алгебры логики – высказывание.

Высказывание - некоторое предложение, о котором можно утверждать, что оно истинно или ложно.

Любое высказывание можно обозначать символом, например x и считать, что x=1,если высказывание истинно, а x=0- если высказывание ложно.

Логическая (булева) переменная – такая величина x, которая может принимать только два значения: x={0,1}.

Высказывание абсолютно истинно, если соответствующая ей логическая величина принимает значение x=1 при любых условиях.

Высказывание абсолютно ложно, если соответствующая ей логическая величина принимает значение x=0 при любых условиях.

Логическая функция (функция алгебры логики) – функция $f(x_1,x_2 \ldots x_n)$**,** принимающая значение равное нулю или единице на наборе логических переменных x_1 **,** $\boldsymbol{x_2 \ldots x_n}$ **.**

Булевы функции одной переменной

Таблица8

Функция	Аргумент x		Условное обозначение функции	Название функции
	0	1		
f_0	0	0	0	Константа 0
f_1	0	1	x	Переменная x
f_3	1	0	$\bar{x}$	Отрицание или инверсия x
F_4	1	1	1	Константа 1

Булевы функции двух переменных

Таблица 8

Функция	$x_1\, x_2$				Название функции
	00	01	10	11	
$f_0(x_1, x_2)$	0	0	0	0	f_0 - абсолютная ложь
$f_1(x_1, x_2)$	0	0	0	1	$x_1 \wedge x_2$ $(x_1\, x_2)$- конъюнкция, логическое И
$f_2(x_1, x_2)$	0	0	1	0	$x_1 \wedge \overline{x_2}$ - запрет по x_2
$f_3(x_1, x_2)$	0	0	1	1	$x_1 \wedge \overline{x_2} \vee x_1\, x_2 = x_1$ переменная x_1
$f_4(x_1, x_2)$	0	1	0	0	$x_2 \wedge \overline{x_1}$ - запрет
$f_5(x_1, x_2)$	0	1	0	1	$x_2 \wedge \overline{x_1} \vee x_1 x_2 = x_2$ - переменная x_2
$f_6(x_1, x_2)$	0	1	1	0	$x_1 + x_2$ - Сложение по модулю 2
$f_7(x_1, x_2)$	0	1	1	1	$x_1 \vee x_2$ дизъюнкция, логическое ИЛИ
$f_8(x_1, x_2)$	1	0	0	0	$x_1 \downarrow x_2$ - стрелка Пирса
$f_9(x_1, x_2)$	1	0	0	1	$x_1 \equiv x_2$ - равнозначность
$f_{10}(x_1, x_2)$	1	0	1	0	$\overline{x_1}\ \overline{x_2} \vee x_1 \overline{x_2} = \overline{x_2}$ - отрицание $\overline{x_2}$
$f_{11}(x_1, x_2)$	1	0	1	1	$x_2 \rightarrow x_1$ - импликация
$f_{12}(x_1, x_2)$	1	1	0	0	$\overline{x_1}\ \overline{x_2} \vee x_2 \overline{x_1} = \overline{x_1}$ - отрицание $\overline{x_1}$
$f_{13}(x_1, x_2)$	1	1	0	1	$x_1 \rightarrow x_2$ - импликация
$f_{14}(x_1, x_2)$	1	1	1	0	$x_1 \mid x_2$ - штрих Шеффера
$f_{15}(x_1, x_2)$	1	1	1	1	f_1 - абсолютная истина

Аргументами и значениями булевых функции являются двоичные слова.

Некоторые из приведенных в таблице 8. функций могут быть распространены на случай более двух независимых переменных (конъюнкция, дизъюнкция, сложение по модулю 2, стрелка Пирса, штрих Шеффера).

Новые булевы функции могут конструироваться из других путем суперпозиции, которая подразумевает подстановку вместо аргументов булевой функции других булевых функций или перенумерации аргументов.

Это возможно, поскольку как аргументы, так и сами булевы функции могут принимать только два значения: 0 и 1. Так, например, функцию f_{14} можно получить из функции f_3, подставив вместо x функцию f_5 ($f_{13}= x_1/x_2= x_1x_2$). Здесь возникает вопрос, существует ли такой набор булевых функций, из которого методом суперпозиции можно получить любую булеву функцию. Этот вопрос связан с понятием функциональной полноты системы логических функций. Доказано, что функциональной полнотой обладают следующие системы:

1) ИЛИ, И, НЕ (f_6, f_5, f_3);
2) ИЛИ, НЕ (f_6, f_3);
3) И, НЕ (f_5, f_3);
4) стрелка Пирса (f_{14});
5) штрих Шеффера (f_{13}).

Функционально-полная система называется также базисом. Условные графические изображения логических элементов, реализующих функции базисов, приведены на рис. 10.

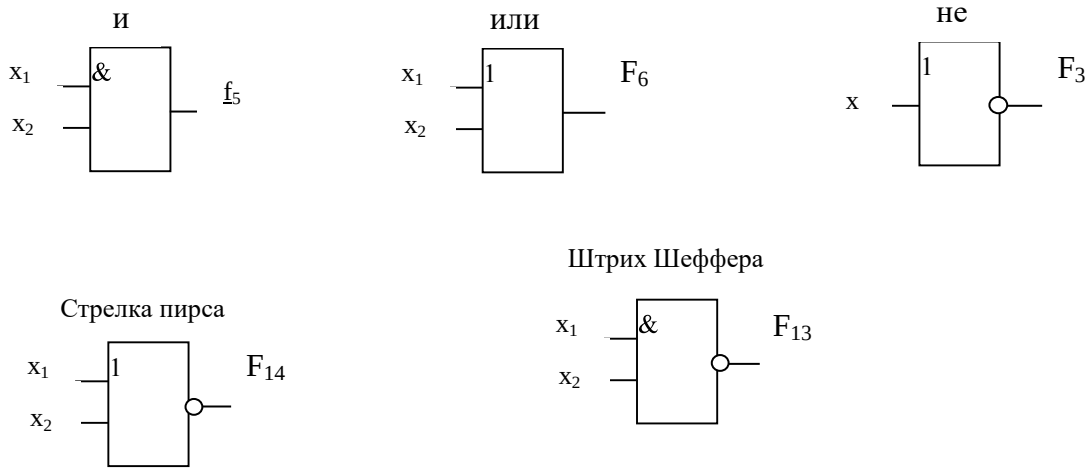


Рис 10.

Функция, выполняемая комбинационной схемой, обычно, описывается с помощью логической (булевой) функции. Прежде чем перейти к схемотехнической реализации, логическая функция должна быть минимизирована, чтобы обеспечить использование минимального количества логических элементов. Для этой цели разработан ряд методов оптимизации, которые используют теоремы булевой алгебры. Эти теоремы представляют основу для проектирования цифровой системы. Основные теоремы булевой алгебры приведены в таблице 9. Здесь A, B и C – двоичные переменные. Доказательство приведенных теорем может быть проведено с использованием таблиц истинности основных логических элементов. Возможно также доказательство одних теорем через другие.

Наиболее важной является теорема де Моргана под номером 12. Она позволяет, при описании логических функций, **пе**реходить от одного базиса к другому.

Таблица 9.

Номер п/п	Дизъюнктивная форма	Конъюнктивная форма
1	$A+0=A$	$A\cdot 1=A$
2	$A+B=B+A$	$A\cdot B=B\cdot A$
3	$A+B\cdot C=(A+B)\cdot(A+C)$	$A\cdot(B+C)=A\cdot B+AC$
4	$A+A=1$	$A\cdot A=0$
5	$A+A=A$	$A\cdot A=A$
6	$A+1=1$	$A\cdot 0=0$
7	$1=0$	
8	$A+A\cdot B=A$	$A\cdot(A+B)=A$
9	$A=A$	
10	$A+(B+C)=(A+B)+C$	$A\cdot(B\cdot C)=(A\cdot B)\cdot C=A\cdot B\cdot C$
11	$A+A\cdot B=A+B$	$A\cdot(A+B)=A\cdot B$
12	$\overline{A+B}=A\cdot B$	$\overline{A\cdot B}=\overline{A}+B$
13	$A\cdot B+\overline{A}\cdot C+B\cdot C=A\cdot B+A\cdot\overline{C}$	$(A+B)\cdot(\overline{A}+C)\cdot(B+C)=(A+B)\cdot(A+\overline{C})$

Например, для реализации функции ИЛИ в базисе Шеффера необходимо использовать три элемента «штрих Шеффера», самостоятельно проверти сами.

2.3. Логические элементы и их характеристики

Основой для построения цифровых схем служат логические элементы, являющиеся техническими аналогами булевых функций. В свою очередь, логические элементы реализуются на основе транзисторов, резисторов, диодов и конденсаторов. Все логические элементы производятся в настоящее время в интегральном исполнении, то есть в виде интегральных схем (ИС). Микроэлектронные технологии и схемотехника логических элементов развивались в тесной взаимосвязи и взаимовлиянии друг на друга. Особенности современных микроэлектронных технологий привели к тому, что в настоящее время основным элементом в ИС является

транзистор, который может использоваться и как активный элемент и как элемент, выполняющий функции резистора или конденсатора.

Существует ряд типов логических элементов, отличающихся схемотехникой и технологией изготовления. Развитие схемотехники преследовало в первую очередь улучшение технических характеристик и, естественно, движение шло от простых схем к более сложным. Каждый тип логики получил свое название и сокращенную аббревиатуру. Вначале появились типы логики :

РТЛ – резисторно-транзисторная логика;
ТЛНС – транзисторная логика с непосредственными связями;
ТРЛ – транзисторно-резисторная логика;
РЕТЛ – резисторно-емкостная транзисторная логика;
ДТЛ – диодно-транзисторная логика;
ВПЛ – высокопороговая логика.

Перечисленные выше типы логики в настоящее время не используются в ЭВМ, так как не обладают требуемыми на сегодняшний день техническими характеристиками.

Технология производства интегральных микросхем сегодня однозначно определяет тип логики. Приведем перечень основных типов современных логических элементов:

ТТЛ – транзисторно-транзисторная логика;
ТТЛШ - транзисторно-транзисторная логика с диодами Шотки;
ЭСЛ – эмиттерно-связанная логика;
МОПn – логика на полевых транзисторах (структура «металл – окисел – полупроводник» с каналом n-проводимости);
МОПр - логика на полевых транзисторах (структура «металл – окисел – полупроводник» с каналом р-проводимости);
КМОП – комплементарная логика (структуры «металл – окисел – полупроводник» с каналами n и р-проводимости на одном кристалле);
$И^2Л$ – инжекционная интегральная логика;
ИПЛ – инжекционно -полевая логика.

Транзисторно-транзисторная логика (ТТЛ) появилась с внедрением технологии изготовления многоэмиттерных транзисторов. Здесь входные сопротивления и выходное сопротивление одного порядка, что увеличивает коэффициент разветвления по выходу (количество входов логических элементов, которые можно подключить к одному выходу). Эта логика обладает достаточно высоким быстродействием, но имеет и недостаток: существенное потребление мощности.

Эмиттерно-связанная логика (ЭСЛ) является наиболее быстродействующей. Это объясняется тем, что транзисторы здесь не входят в насыщение и уровни напряжения изменяются в небольших пределах.

Логика КМОП является самой экономичной в смысле потребления электрической энергии. Ток от источника питания потребляется только в моменты перехода уровня выходного напряжения $U_{вых.}$ из логической единицы в логический ноль или наоборот. Поскольку закрытый полевой транзистор имеет очень большое сопротивление, то величина

потребляемого тока будет близка к нулю при установившемся значении напряжения на выходе.

Инжекционная интегральная логика ($И^2Л$) имеет свою особенность. Она работает при напряжениях питания от 1 до 1, 5 вольт. Это объясняется необычностью её схемотехники, то есть использованием инверсного режима работы многоэмиттерных n-p-n транзисторов, что возможно только при малых питающих напряжениях. Применять такую логику для вычислительных устройств, питающихся от промышленной сети проблематично, так как сложно реализовать стабильные источники питания с таким напряжением. Эта логика подходит для носимых вычислительных устройств, питающихся от стандартных батареек.

Интегрально-полевая логика отличается от $И^2Л$ тем, что в качестве источника базового тока n-p-n транзисторов здесь используется не биполярный p- n-p транзистор, а полевой транзистор с p-n переходом.

Логические элементы служат основой для реализации всех элементов цифровых устройств.

Для обеспечения работоспособности цифровых устройств необходимо, чтобы ЛЭ обладали следующими свойствами:

1. совместимостью входных и выходных сигналов;
2. достаточной нагрузочной способностью;
3. способностью обеспечивать стандартные параметры выходных сигналов;
4. помехоустойчивостью.

Совместимость входных и выходных сигналов. В сложных схемах цифровых устройств ЛЭ соединяются так, чтобы выход каждого элемента работал на один или несколько входов других ЛЭ, в том числе и на свои собственные входы. Для нормального функционирования таких схем должна быть обеспечена совместимостью уровней сигналов «0» и «1» по входам и выходам, т.е. соответствующие уровни напряжений логических сигналов должны лежать зоне отображения «0» и «1» (для ТТЛ-элементов соответственно 0.4В и 2.4В. см. рис. 1). только в этом случае возможна непосредственная работа одного ЛЭ без применения специальных схем для согласования уровней сигналов.

Нагрузочная способность. Для построения разветвленных логических схем необходимо, чтобы каждый ЛЭ обладал определенной нагрузочной способностью по входу и выходу, т.е. мог работать по нескольким логическим входам и одновременно управлять несколькими входами других ЛЭ.

Нагрузочную способность ЛЭ принято выражать коэффициентом разветвления по выходу (Краз) и коэффициентом объединения по входу (Коб). Под коэффициентом разветвления по выходу понимают наибольшее число входов ЛЭ, которые можно подключить к выходу данного ЛЭ, не вызывая искажения формы и амплитуды сигнала, выходящих за границы зон отображения «0» и «1» (рис. 11). Коэффициент объединения по входу равен допустимому числу входов ЛЭ. (Для ТТЛ-логических элементов «И-НЕ» К155ЛА1, -ЛА2, -ЛА3. -ЛА4 Краз=10, К155ЛА6 Краз=30, «НЕ» К155ЛН1, -ЛН2, Краз=10.)

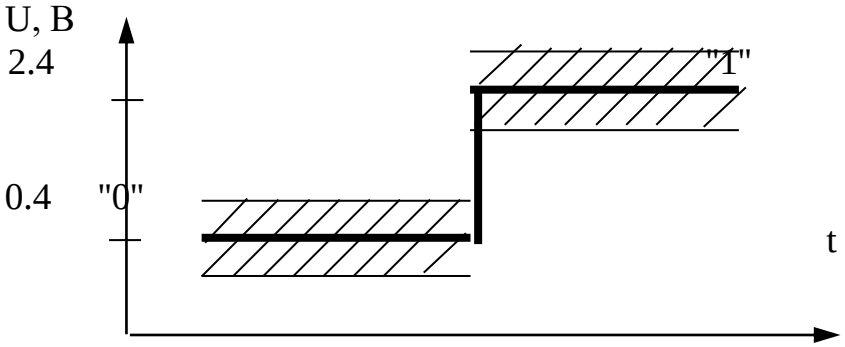


Рис.11. Зоны отображения уровней сигналов “0” и “1”.

2.4. Основные параметры сигналов.

Для нормального функционирования схем необходимо, чтобы сигнал, проходящий через активный ЛЭ, имел некоторые стандартные амплитудные и временные параметры (амплитуду, длительность фронтов, см. рис.12) и существенно не изменял их. Для этого требуется, чтобы ЛЭ обладали определенными формирующими свойствами.

Помехоустойчивость ЛЭ. При работе цифровых устройств допустимы даже кратковременные искажения информации. Поэтому ЛЭ должны обладать высокой помехоустойчивостью, то есть нечувствительностью к действию помех при нулевом и единичном уровнях входных сигналов.

Различают статическую помехоустойчивость по высокому уровню, определяемую по формуле U^1п.ст$=\left|U^1\text{вых.min}-U^1\text{пор}\right|$, и статическую помехоустойчивость по низкому уровню, определяемую по формуле U^0п.ст$=\left|U^0\text{пор}-U^0\text{вых.max}\right|$ ЛЭ характеризуется меньшим из этих двух значений. ТТЛ - элементы обладают статической помехоустойчивостью $U\text{п.ст} \le 0.4\text{В}$

Основные параметры ЛЭ. ЛЭ характеризуется следующими основными параметрами:

1) динамические,
2) статические,
3) интегральные.

Динамические параметры. Определяют быстродействие ЛЭ при переключениях. Оно определяется электрической схемой, технологией изготовления и характером нагрузки.

Для определения динамических параметров стандартами устанавливаются требования к амплитуде и длительности фронта входного сигнала, а также к задержке выходного сигнала относительно входного. Уровни отсчета напряжений для определения динамических параметров устанавливаются относительно выходных пороговых напряжений «0» и «1» (рис. 12.)

Основными динамическими параметрами ЛЭ являются:

1.0 - tздр - время задержки распространения сигнала при переходе выходного напряжения от «1» к «0»;

0.1 - tздр - время задержки распространения сигнала при переходе выходного напряжения от «0» к «1»;

1.0 tф - длительность фронта выходного сигнала при переходе напряжения от «1» к «0»;

0.1 - tф - длительность фронта выходного сигнала при переходе напряжения от «0» к «1»;

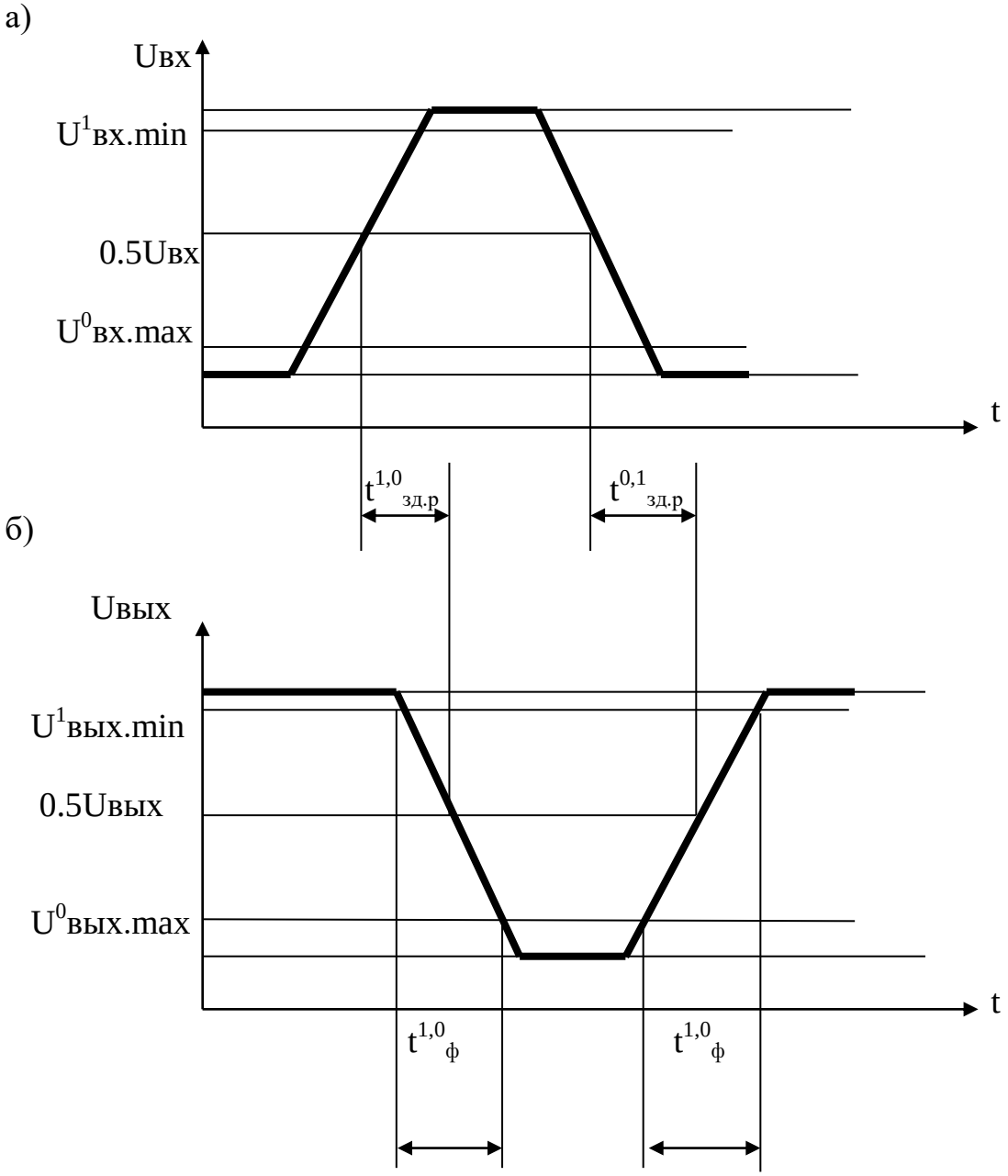


Рис.12. Входной (а) и выходной (б) сигналы инвертирующего ЛЭ

При расчете временной задержки сигнала последовательно соединенных ЛЭ используется средняя задержка

$$t_{зд\ р}^{ср} = (t_{здр}^{0.1} + t_{здр}^{1.0})/2$$

Длительности положительных и отрицательных фронтов измеряют по уровням 0.1 и 0.9 (рис.12).

Значения задержек некоторых ИМС ТТЛ- элементов приведены в табл.10.

Таблица 10.

	(И-НЕ) К155ЛА1 ЛА2,ЛА3	(ИЛИ-НЕ) К155ЛЕ1 ЛЕ2	(И) К155ЛИ1	(НЕ) К155ЛН1	(НЕ) К155ЛН2	(ИЛИ) К155ЛЛ1
$t_{здр}^{0.1}$ нс	≤ 22	≤ 22	≤ 19	≤ 22	≤ 55	≤ 15
$t_{здр}^{1.0}$ нс	≤ 15	≤ 15	≤ 27	≤ 15	≤ 15	≤ 22

Статические параметры. Статические параметры определяют условия формирования и значения высокого (В) и низкого (Н) уровней ЛЭ, его нагрузочную способность, потребляемую мощность при заданных напряжениях питания, нагрузке и температуре окружающей среды.

К статическим параметрам ЛЭ относятся:

выходные и входные напряжения логических «0» и »1»

($U^{0}_{вых}$, $U^{1}_{вых}$, $U^{0}_{вх}$, $U^{1}_{вх}$);

выходные и входные пороговые напряжения логических «0» и »1»

($U^{0}_{вых.пор}$, $U^{1}_{вых.пор}$, $U^{0}_{вх.пор}$, $U^{1}_{вх.пор}$);

выходные и входные токи логических «0» и »1» ($I^{0}_{вых}$, $I^{1}_{вых}$);

токи потребления в состоянии логических «0» и «1» ($I^{0}_{пот}$, $I^{1}_{пот}$);

потребляемая мощность (Рпот)

Для ТТЛ элементов К155ЛА1 Рпот=52мВт, К155ЛА2 Рпот=21мВт, К155ЛА3 Рпот=78мВт, К155ЛН1 Рпот=173мВт.

Интегральные параметры. Интегральные параметры отражают уровень развития технологии схемотехники, а также качество ИМС. Основными интегральными параметрами ИМС являются энергия переключения Эпт и уровень интеграции N.

Для мощных (быстродействующих) ТТЛ-элементов (МТТЛ)
Эпт = Рпот*tздр.ср =In*Uип*tздр.ср = 120÷150 пдж.
Для ТТЛ элементов средней мощности (СТТЛ) Эпт = 100 пдж.
Для ТТЛ элементов малой мощности (МТТЛ) Эпт = 33 пдж.

Степень интеграции ИМС определяется числом простейших (базовых) ЛЭ обычно двухвходовых вентилей на кристалле.

2.5. **Классификация микросхем и условное обозначение ИМС.**

По принятой системе условных обозначений все выпускаемые отечественные ИС делятся:

по конструктивно-технологическому исполнению на три группы: 1,5,6,7 – полупроводниковые; 2,4,8 – гибридные; 3 – прочие (пленочные, вакуумные, керамические и т.п.).

по характеру выполняемых функций в радиоэлектронной аппаратуре ИС подразделяются на подгруппы (например: генераторы, усилители, триггеры, преобразователи и т.д.) и виды (например: преобразователи частоты, фазы, напряжения и т.п.). Разделение ИС на подгруппы и виды приведено в табл.11.

По принятой системе обозначение ИС должно состоять из пяти элементов:

первый – буква(ы) К, КМ, КР – обозначающие условия приемки на заводе и тип корпуса;

второй – цифра обозначающая группу ИС;

третий – три цифры (от 000 до 999) или две цифры (от 00 до 99),обозначающие порядковый номер серии микросхем.

Таким образом, цифры второго и третьего элемента определяют полный номер серии ИС.

четвертый элемент – две буквы, соответствующие подгруппе и виду (см.табл.11)

пятый – условный номер ИС по функциональному признаку в данной серии.

Таблица 11.

Группа и функциональное назначение	**обозначение**	**Группа и функциональное назначение**	**обозначение**
1. Логические элементы:	ЛА	4. Схемы запоминающих устройств:	
И-НЕ	ЛЕ	оперативные	РУ
ИЛИ-НЕ	ЛИ	постоянные с	
И	ЛЛ	ультрафиолетовым	РФ
ИЛИ	ЛН	стиранием	
НЕ	ЛБ	постоянные с	
И-НЕ / ИЛИ-НЕ	ЛП	возможностью	РР
Исключающее ИЛИ		многократного	
2. Схемы арифметич.		перепрограммирования	ГГ
и дискр. устройств:	ИА	5. Генераторы:	ГФ
АЛУ	ИВ	прямоугольных сигналов	
Шифраторы	ИД	сигналов спец. формы	
Дешифраторы	ИР	6. Формирователи:	АГ
Регистры	ИЕ	импульсов прямоугольной	А
Счетчики	ИМ	формы	Ф
Сумматоры	ИЛ	импульсов спец. формы	
Полусумматоры		7. Схемы вычислительных	
3. Триггеры:	ТВ	средств:	
JK-типа	ТР	схемы управления вводом-	ВВ
RS-типа	ТМ	выводом	ВГ
D-типа	ТТ	контроллеры	ВЕ
Т-типа		микроЭВМ	В
Комбинированные:		микропроцессоры	М
DT,RST и др.		микропроцес. секции	ВС
		схемы управл. памятью	ВТ

Условные графические обозначения основных ЛЭ приведены выше (рис.10) , порядок функционирование их в таб.8.

Самостоятельная работа

1. Определить какие и сколько логических элементов размещено в корпусе исследуемых ИМС : К155ЛА3, К155ЛИ1,К155ЛЛ1, К155ЛН1, К155ЛЕ1, К155ЛП5.
2. Нарисовать назначение выводов каждой ИМС, указанной в п.1.

2.6. Триггеры

Триггером называется устройство, имеющее два устойчивых состояния, которые отличаются уровнем напряжения на его выходе. Высокому напряжению на выходе триггера приписывают состояние логической единицы, а низкому – логического нуля. Как правило, триггер имеет два выхода: прямой и инверсный.

Классификация триггеров приведена на рис.12. В основу классификации триггеров положены два основных признака: функциональный признак и способ записи информации в триггер.

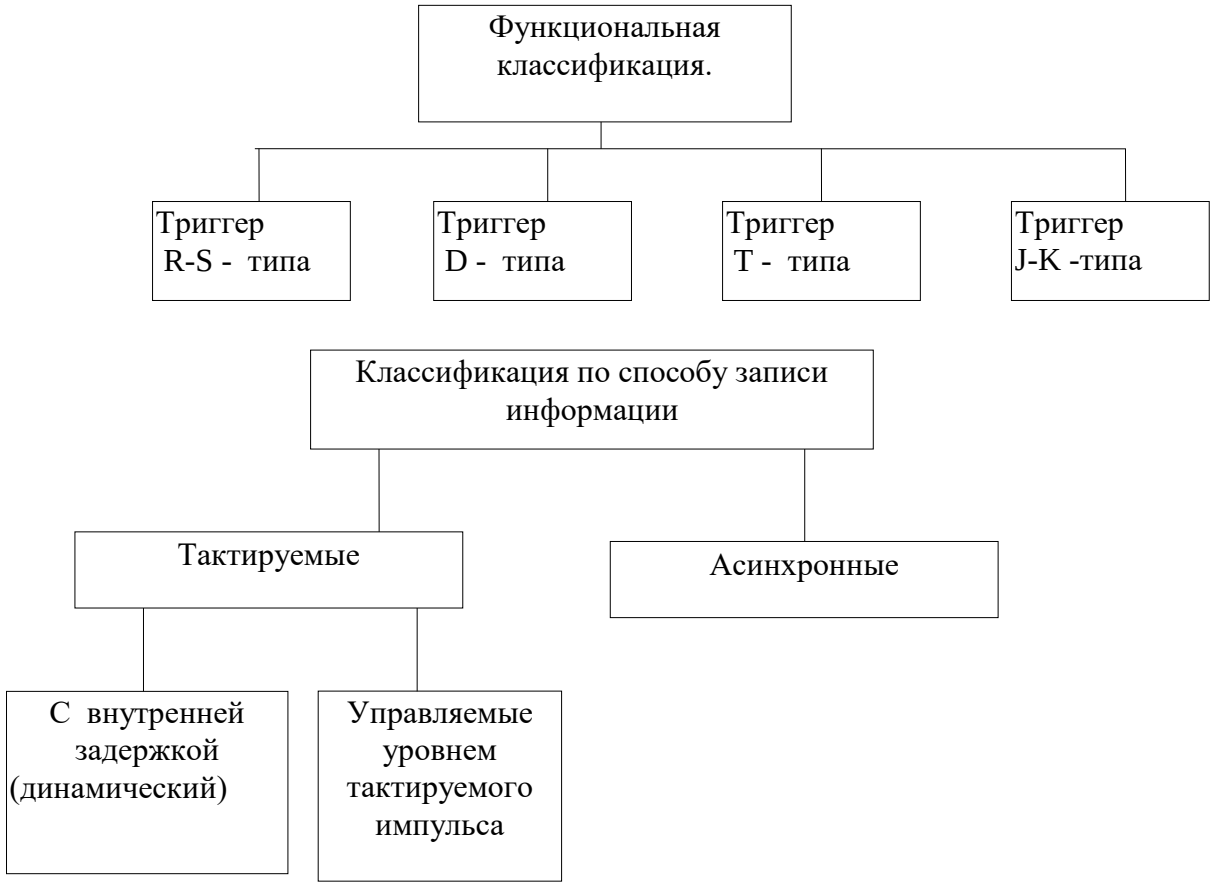


Рис. 12. Классификация триггерных устройств.

Функциональная классификация является наиболее общей и представляет собой классификацию триггеров по виду логического уравнения, характеризующего состояние входов и выходов триггера в

момент времени до его срабатывания (t) и после его срабатывания (t_{n+1}). В соответствии с функциональной классификацией различают триггеры R-S, D, T, J-K- типа.

Классификация по способу записи информации характеризует временную диаграмму работы триггера, т.е. определяет ход процесса записи информации в триггер.

По этой классификации триггеры подразделяют на две группы

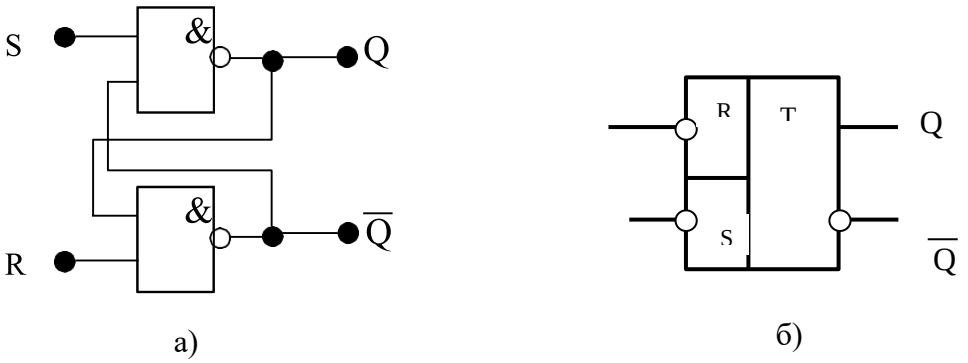
- асинхронные;
- синхронные (или тактируемые).

Отличительной особенностью асинхронных триггеров является то, что запись информации в них осуществляется непосредственно с поступлением информационного сигнала на вход. Запись информации в тактируемые триггеры, имеющие информационные и тактовые входы, осуществляется только при подаче разрешающего, тактирующего импульса (ТИ).

В свою очередь, тактируемые триггеры подразделяются на триггеры, работающие по уровню ТИ (срабатывание триггера происходит одновременно с поступлением тактирующего сигнала), и на триггеры с внутренней задержкой (срабатывание триггера происходит после окончания действия тактового сигнала)

Существует много типов триггеров. Рассмотрим основные из них.

Наиболее простым триггером является RS – триггер. Для его реализации достаточно двух двухвходовых логических элемента И-НЕ. На рис.13. а) приведена схема такого триггера, а на рис. 13. б) – его условное графическое изображение. Функционирование триггера описывается с помощью таблицы истинности (см. таблицу 13). Буквой Q обозначен прямой выход, а буквой $\overline{Q}$ – инверсный. Обозначение Q_{t-1} означает состояние выхода, которое имело место до подачи на входы R и S последней кодовой комбинации.



а) б)

Рис. 13. Схема RS-триггера (а) и его условное графическое изображение (б).

Таблица 13

R	S	Q	$\bar{Q}$
0	1	0	1
1	0	1	0
1	1	Q_{t-1}	Q_{t-1}
0	0	запрещенная комбинация	

Кодовая комбинация R=0, S=1 устанавливает триггер в ноль ($\bar{Q} = 0$), а кодовая комбинация R = 1, S = 0 устанавливает триггер в единицу (Q = 1). Если на оба входа R и S подать единицы, то триггер сохраняет предыдущее состояние. R = 0, S = 0 является запрещенной кодовой комбинацией, так как в этом случае оба выхода триггера окажутся в состоянии единицы и потеряется понятие прямого и инверсного выходов.

Из RS – триггера и двух двухвходовых логических элементов И-НЕ можно реализовать D – триггер с потенциальным управлением записью.

На рис. 14 приведена схема такого триггера. Функционирование триггера описывается таблицей 14. Если C = 1, то выход Q повторяет значение на входе D. При нулевом сигнале на входе C выходы сохраняют состояние, которое было в момент перехода уровня сигнала на входе C из единичного состояния в нулевое (Q_{t-1}).

Таблица 14.

C	D	Q	$\bar{Q}$
1	1	1	0
1	0	0	1
0	1	Q_{t-1}	$\bar{Q}_{t-1}$
0	0	Q_{t-1}	$\bar{Q}_{t-1}$

Триггеры D-типа. Триггером D-типа, известным в литературе под названием “триггера задержки”, называют логическое устройство с двумя устойчивыми состояниями, имеющее один синхронизирующий (C) и один информационный (D) входы. Закон функционирования триггера D-типа приведен в табл. 14. и описывается логическим уравнением Q(t+1)=D(t).

Логическое уравнение показывает, что состояние триггера в момент времени t+1 совпадает с кодом входного сигнала, действовавшего в момент времени t, т.е. осуществляется “задержка” выходного сигнала.

(x-выходное состояние триггера не зависит от состояния на входе.)

Триггер D-типа, управляемый уровнем тактового импульса, выполненного на элементах И-НЕ, показан на рис.14. Здесь вход D является информационным, а вход C - тактовым. Схема рис.4 тактируется сигналом логической 1.

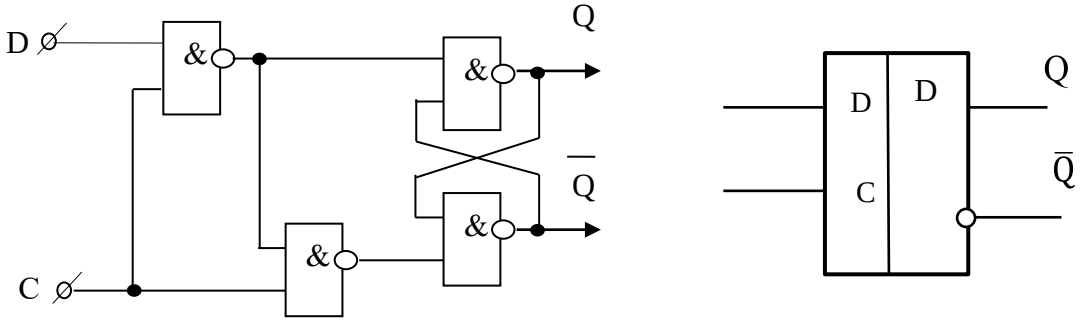


Рис. 14 Статический D-триггер.

Динамический D-триггер отличается от описанного выше тем, что информация на его выходе изменяется только в момент появления на его входе “активного” фронта (положительного или отрицательного – того, на который срабатывает триггер).

Функционирование триггера удобно представлять в виде временных диаграмм. На рис. 15 приведены временные диаграммы работы D – триггера со статическим управлением записью. Горизонтальные оси соответствуют времени, а вертикальные – уровню напряжения на входах D, C и выходе Q соответственно. Высокий потенциал соответствует логической единице, а низкий - логическому нулю.

Из двух D – триггеров со статическим управлением записью и одного инвертора можно составить схему D – триггера с динамическим управлением записью. На рис. 15 приведена схема и условное графическое изображение D – триггера с динамическим управлением записью. Значок > указывает, что вход C является динамическим и реагирует на фронт импульса, образуемый при переходе напряжения от уровня логического нуля к уровню логической единицы. Если вход динамический, но реагирует на фронт, образуемый переходом напряжения от уровня логической единицы к уровню логического нуля, то используется значок < .

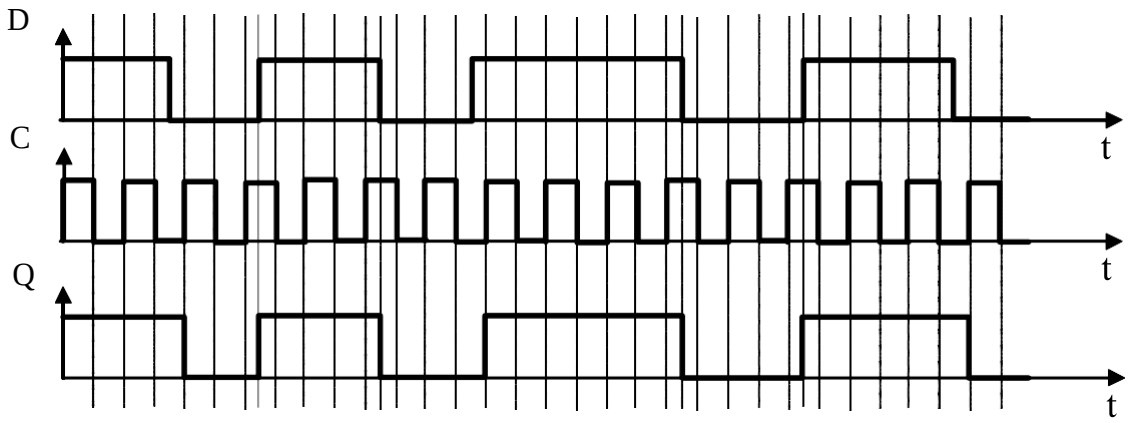
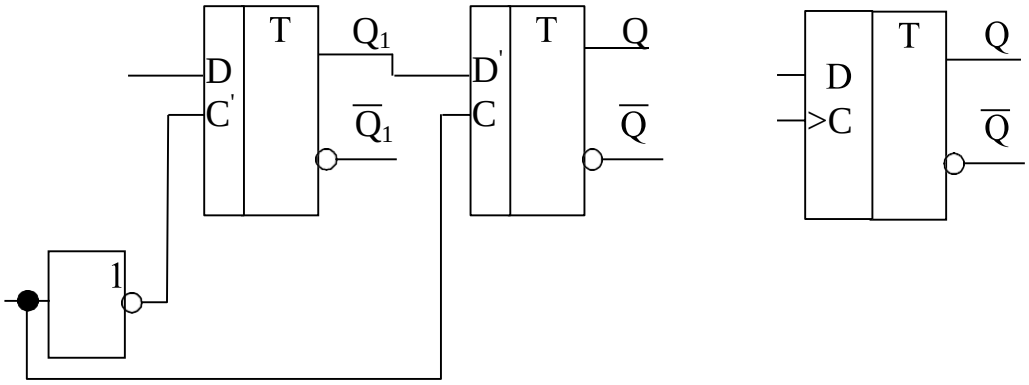


Рис. 15. Временные диаграммы работы D – триггера со статическим управлением записью.

В D – триггере с динамическим управлением записью напряжение на выходе принимает логический уровень, соответствующий входу D в момент, когда на входе C происходит переход напряжения от уровня логического нуля к уровню логической единицы, то есть происходит как бы запись информации по фронту, и информация сохраняется, пока не придет следующий такой же фронт.



а) б)

Рис. 15. Схема (а) и условное графическое обозначение (б) D – триггера с динамическим управлением записью.

На рис. .16 приведены временные диаграммы работы D – триггера с динамическим управлением записью. Горизонтальные оси соответствуют времени, а вертикальные – уровню напряжения на входах D, D', C, C', и выходе Q со- ответственно. Высокий потенциал соответствует логической единице, а низкий - логическому нулю.

D – триггеры с динамическим управлением записью служат основой для построения последовательных и параллельных регистров, а также счетчиков.

Если инверсный выход такого триггера соединить со входом D, то получится счетный триггер. Он будет менять состояние выхода на противоположное по каждому фронту на входе « >C» (перепаду напряжения с уровня логического нуля в уровень логической единицы).

На рис.18 приведены условные графические изображения D – триггера а) и JK – триггера б), выпускаемых в интегральном исполнении по технологии ТТЛ. Функционирование триггеров описываются таблицами истинности см. таблицу 18 и таблицу 1 9 .

Таблица 18

R	S	D	C	Q
0	1	x	x	0
1	0	x	x	1
1	1	0	↓	0
1	1	1	↑	1
1	1	0	↑	Q_{t-1}
1	1	1		Q_{t-1}

Таблица 19

R	S	J	K	C	Q
0	1	x	x	x	0
1	0	x	x	x	1
1	1	1	1	↓	Q
1	1	1	0	↓	0
1	1	0	1	↓	1
1	1	0	0	↑	Q_{t-1}
1	1	1	1	↓	Q_{t-1}

Приведенные в таблицах 18 и 19 обозначения следует понимать как:

- ↑ фронт импульса, образованный перепадом напряжения из уровня логического нуля в уровень логической единицы;
- ↑̸ отсутствие фронта импульса, образованного перепадом напряжения из уровня логического нуля в уровень логической единицы;

Q_{t-1} – сохранение состояния, которое было в предыдущий момент времени;

- фронт импульса, образованный перепадом напряжения из уровня логической единицы в уровень логического нуля;

- отсутствие фронта импульса, образованного перепадом напряжения из уровня логической единицы в уровень логического нуля;

x - любое состояние входного сигнала.

Выпускаемые промышленностью D – триггеры (в интегральном исполнении) имеют также входы R и S, то есть совмещают в себе и D - и RS – триггер. Входы R и S имеют высший приоритет по сравнению с входами D и C.

Если на входы R и S (рис. 16.а)) поданы логические единицы, то D – Триггер работает как триггер, представленный на рис. .15 (временные диаграммы приведены на рис. 16).

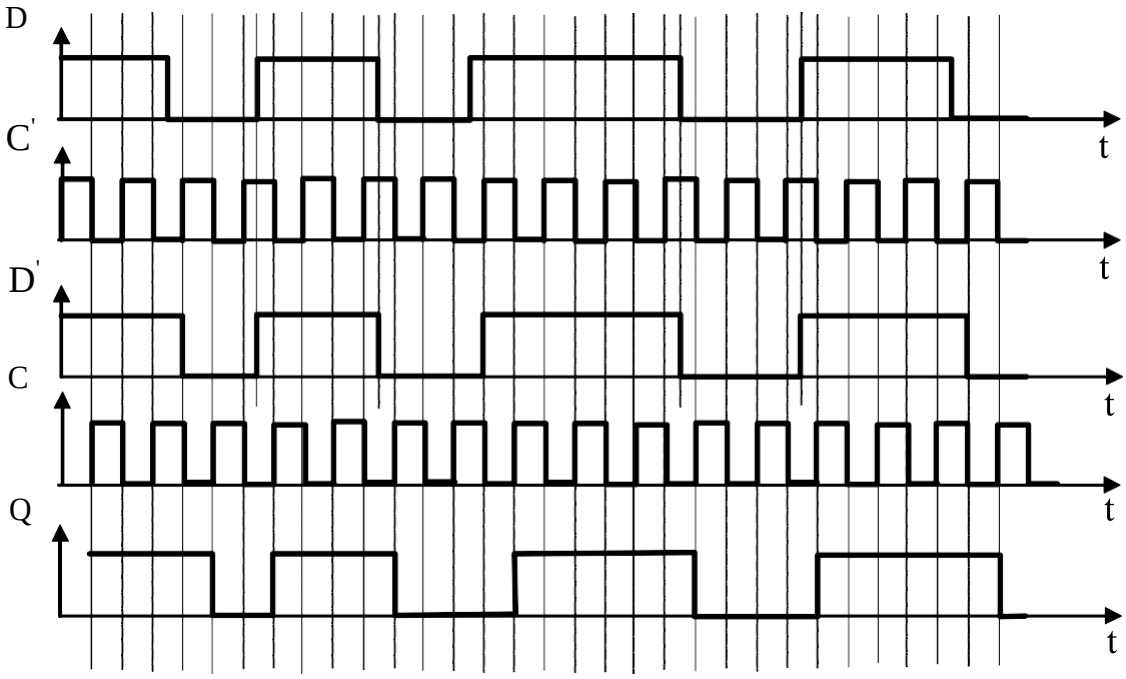


Рис. 16. Временные диаграммы работы D – триггера с динамическим управлением записью.

Триггеры Т-типа. Триггером Т- типа (**счетный триггер**) называют логическое устройство с двумя устойчивыми состояниями и одним входом Т, изменяющее свое состояние на противоположное всякий раз, когда на вход Т поступает “активный” фронт.

Так как триггер срабатывает на соответствующий фронт, то он является динамическим. Схемы несинхронизируемых Т- триггеров построенные на основе тактируемого R-S триггера и динамического D-триггера, приведены на рис. 17

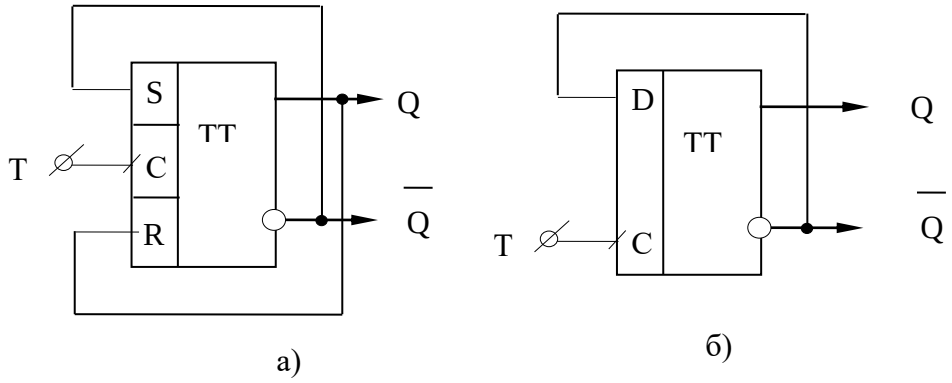


Рис. 17 Схемы Т- триггеров построенных на основе:
а) динамического тактируемого R-S триггера;
б) динамического D-триггера.

Разобраться в работе приведенных схем не сложно, если при этом пользоваться таблицами истинности используемых триггеров (см. табл.20).

Таблица 20 Таблица истинности Т-триггера.

T	Q(t+1)
Отсутствие “активного” фронта по входу «С»	Q(t)
”Активный” фронт по входу «С»	$\overline{Q(t)}$

Триггеры J-K-типа. Триггером J-K- типа называется логическое устройство с двумя устойчивыми состояниями, двумя информационными входами J и K и одним синхронизирующим входом С, которое при условии J*K=1 и наличии на синхровходе “активного” фронта осуществляет инверсию предыдущего состояния, т.е. при

J (t) * K (t) = 1, то Q (t+1) = Q (t) ,

а в остальных случаях функционирует аналогично динамическому R-S-триггеру, при этом вход J эквивалентен входу S, а вход K- входу R.

Работа J-K-триггера отображено в табл. 21

Таблица 21 Таблица истинности J-K триггера.

C	J	K	Q(t+1)
Отсутствие “активного” фронта по входу С	x	x	Q(t)
”Активный” фронт по входу С	0	0	Q(t)
”Активный” фронт по входу С	0	1	0
”Активный” фронт по входу С	1	0	1
”Активный” фронт по входу С	1	1	$\overline{Q(t)}$

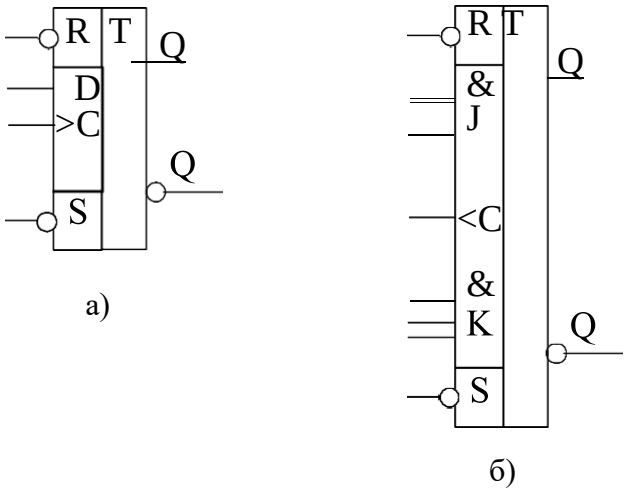


Рис. 18. Условное графическое изображение D – триггера а) и JK – триггера б).

При уровнях напряжения R = 0 и S = 1 на выходе Q будет уровень логического нуля независимо от состояния входов D и C. Если на вход S подать уровень логического нуля, а на вход R – уровень логической единицы, то на выходе Q будет уровень логической единицы независимо от состояния входов D и C.

Входы J и K (см. рис.18 б)) включают по три входа, объединенных по «И», то есть состояние J или K считается равным логической единице, если на все три объединенных входах поданы логические единицы. Входы J и K разрешают переход триггера в нулевое и единичное состояние соответственно при наличии перепада напряжения на входе C с уровня логической единицы в уровень логического нуля и значениях логической единицы на входах R и S.

Широко используется также JK – триггер, представляющий собой счетный триггер с входами J, K, C, R и S. Здесь также входы R и S имеют высший приоритет. Вход C является счетным. Имеется выход прямой и инверсный.

Домашнее задание

1 Изучить принципы построения и функционирования триггерных схем.

2 Проверить свои знания по контрольным вопросам.

3 На основе логических элементов разработать электрические принципиальные схемы (с указанием типа используемых микросхем и цоколевки их выводов) следующих триггеров: R-S-, R-S-E-, статический D-, J-K-типов.

2.7. Регистры

По функциональному назначению регистры можно разбить на три типа: параллельные, последовательные сдвиговые и сдвиговые кольцевые. Регистры строятся из триггеров. Параллельные регистры служат для

временного хранения параллельного двоичного или двоично-десятичного кода. Количество триггеров определяется разрядностью хранимого кода.

На рис. 19 приведена функциональная схема параллельного регистра для хранения n-разрядного двоичного кода, реализованного на D-триггерах с динамическим управлением записью.

По фронту импульса записи параллельный входной код записывается в D-триггеры и хранится в них до прихода очередного импульса записи. При этом можно использовать код, как с прямых, так и с инверсных выходов триггеров.

Последовательные сдвиговые регистры служат для преобразования последовательного двоичного кода в параллельный код и наоборот. На рис 20. представлена функциональная схема последовательного сдвигового регистра на D-триггерах с динамическим управлением записью.

Выход каждого триггера соединен с D-входом следующего триггера. Таким образом, последовательный входной код, поступающий на D-вход первого триггера, как бы задвигается в регистр по мере поступления сдвиговых импульсов и после n-ного импульса в регистре будет сформирован n-разрядный параллельный код. Если в регистр записать параллельный код, используя установочные входы триггеров (R и S), то на выходе последнего триггера можно сгенерировать последовательный код с тактом, равным периоду сдвиговых импульсов.

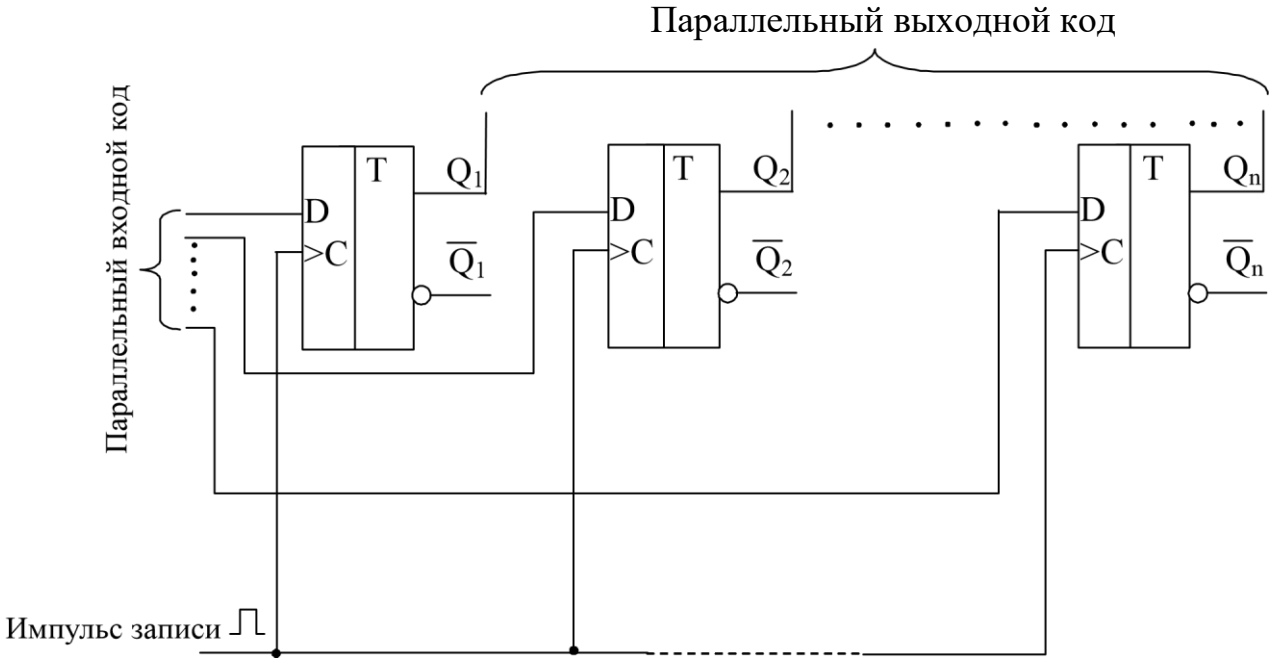


Рис. 19. Функциональная схема параллельного регистра на D триггерах с динамическим управлением записью.

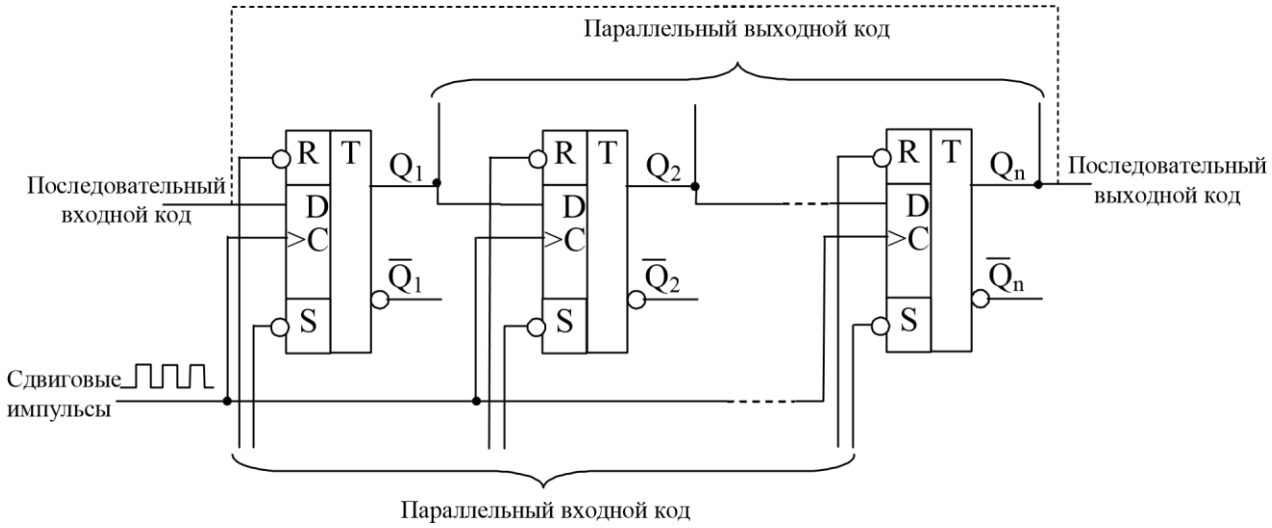


Рис. 20. Функциональная схема последовательного сдвигового регистра на D-триггерах с динамическим управлением записью.

Кольцевой сдвиговый регистр реализуется путем соединения выхода последнего триггера с D-входом первого триггера (на рис. .20 показано пунктирной линией). Он используется для формирования повторяющихся кодовых последовательностей или в качестве счетчика.

2.8. Счетчики

Электронное цифровое устройство, предназначенное для подсчета количества импульсов и деления частоты импульсного сигнала, получило название счетчик.

Классификацию счетчиков можно провести по следующим критериям: способу формирования переноса, коэффициенту пересчета (или деления), по характеру изменения состояния счетчика в момент поступления очередного счетного импульса. На рис. 21 приведена такая классификация в виде граф - схемы.

Счетчики реализуются обычно на счетных триггерах. Если импульсы подаются на счетный вход младшего триггера, а с его выхода сигнал подается на счетный вход следующего по старшинству триггера и т. д., то такой счетчик называется счетчиком с последовательным переносом. Если счетные импульсы подаются одновременно на счетные входы всех триггеров, то такой счетчик называется счетчиком с параллельным переносом.

К двоичным счетчикам относят счетчики с коэффициентами пересчета кратными двойке. Счетчик с коэффициентом пересчета равным десяти называется двоично-десятичным.

Если после прихода очередного счетного импульса код состояния счетчика увеличивается, то счетчик называется суммирующим, а если – уменьшается, то счетчик называется вычитающим. Счетчик, который попеременно может работать в обоих режимах, называется реверсивным.

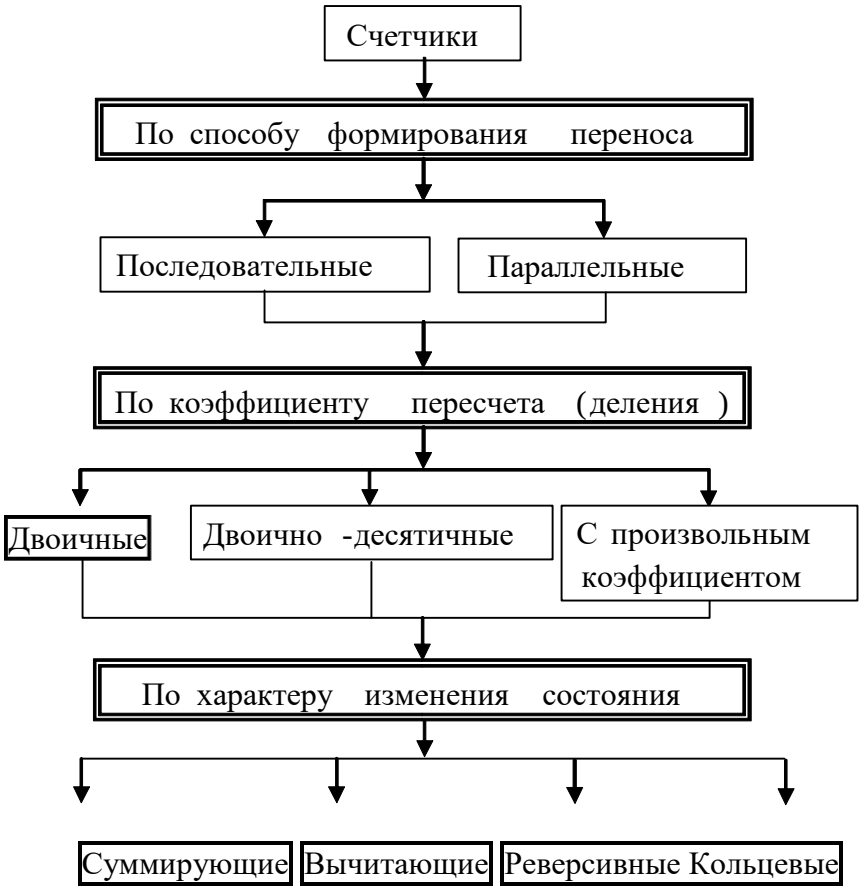
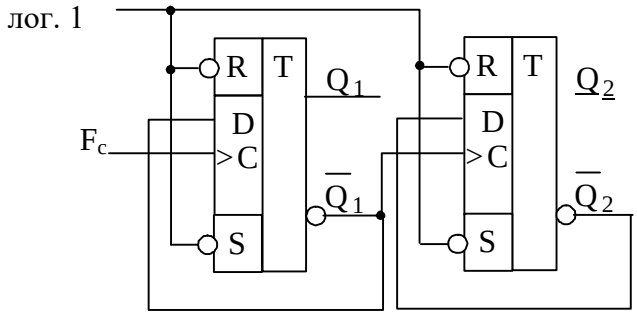


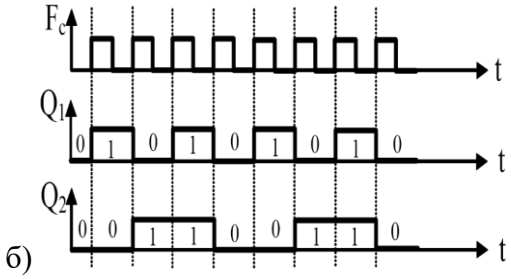
Рис. .21. Классификация счетчиков.

Кольцевой счетчик представляет собой кольцевой сдвиговый регистр, и принцип его работы отличается от всех остальных счетчиков. Обычно при использовании кольцевого сдвигового регистра в качестве счетчика в нем циркулирует код, содержащий одну единицу или один ноль. Коэффициент пересчета такого счетчика равен количеству триггеров, из которых он образован.

На рис. 22 а) изображена функциональная схема двухразрядного двоичного суммирующего счетчика с последовательным переносом, реализованная на D – триггерах с динамическим управлением записью. D – триггеры превращены в счетные триггеры путем соединения их инверсных выходов с D- входами. Работа счетчика иллюстрируется временной диаграммой (рис. 22 б)). Счетные импульсы F_c подаются на вход C первого триггера, который реагирует на каждый фронт, образованный перепадом напряжения из нулевого уровня в уровень логической единицы. После каждого такого фронта он меняет своё состояние на противоположное, то есть если он находился в единичном состоянии, то переходит в нулевое состояние и наоборот. Это приводит к тому, что на выходе Q_1 частота сигнала в два раза меньше, чем частота следования импульсов F_c. На вход C второго триггера подается сигнал с инверсного выхода Q_1 первого триггера, который делит частоту этого сигнала также пополам. Таким образом, на выходе Q_2 частота сигнала будет уже в 4 раза меньше, чем F_c.



а)

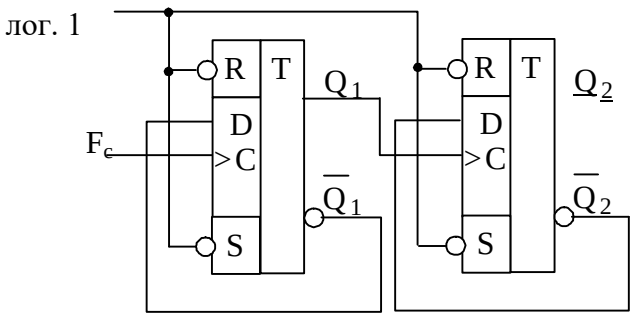


б)

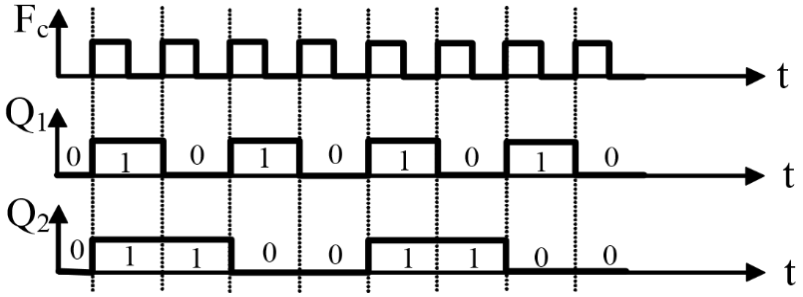
Рис. 22. Функциональная схема двухразрядного двоичного суммирующего счетчика с последовательным переносом а) и временная диаграмма его работы б).

Если Q_2 и Q_1 рассматривать как разряды двоичного кода, то из временной диаграммы (рис. 22 б)) можно заметить, что значение этого кода меняется в следующей закономерности: 00; 01; 10; 11; 00; 01; 10; 11;…и т. д. С приходом очередного счетного импульса значение кода увеличивается на единицу, пока не произойдет переполнение. Такие счетчики называются суммирующими. Чтобы суммирующий счетчик, представленный на рис. 22 стал вычитающим, необходимо на вход С второго триггера подать сигнал не с инверсного выхода, а с прямого Q_1.

Двоичный двухразрядный вычитающий счетчик с последовательным переносом представлен на рис. 23 а), а на рис. 23 б) приведена временная диаграмма, поясняющая его работу.



а)



б)

Рис. 23. Функциональная схема двухразрядного двоичного вычитающего счетчика с последовательным переносом а) и временная диаграмма его работы б).

Из временной диаграммы видно, что значение кода меняется в следующей закономерности: 00; 11; 10; 01; 00; 11; 10; 01;…и т. д. С приходом очередного счетного импульса значение кода уменьшается на единицу, пока не произойдет переполнение.

Чтобы из рассмотренных схем счетчиков реализовать реверсивный счетчик необходимо использовать мультиплексор с двумя информационными входами, на которые нужно подать сигналы с выходов Q_1 и Q_1, а сигнал с выхода мультиплексора подать на вход С второго триггера.

Функциональная схема двоичного двухразрядного реверсивного счетчика с последовательным переносом приведена на рис. 24. Если на вход $Вх._{упр.}$ будет подана логическая единица, то счетчик будет работать в режиме вычитания, а если – логический ноль, то – в режиме суммирования.

Двухразрядный двоичный счетчик способен считать от 0 до 3 (или от 3 до 0) и делит частоту счетных импульсов на 4, то есть имеет коэффициент пересчета 4. Если добавить к нему третий триггер, то получится счетчик с коэффициентом пересчета 8, а если составить счетчик из n-триггеров, то его коэффициент пересчета будет равен 2^n.

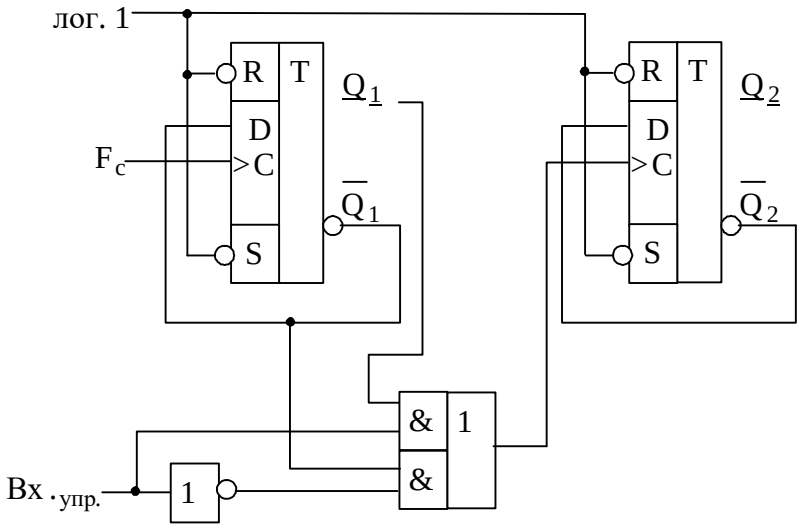
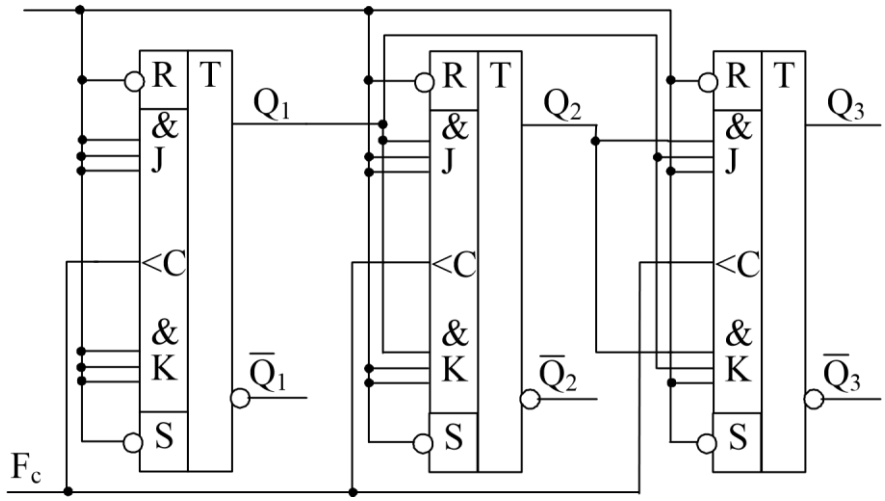


Рис. 24. Функциональная схема двухразрядного двоичного реверсивного счетчика с последовательным переносом.

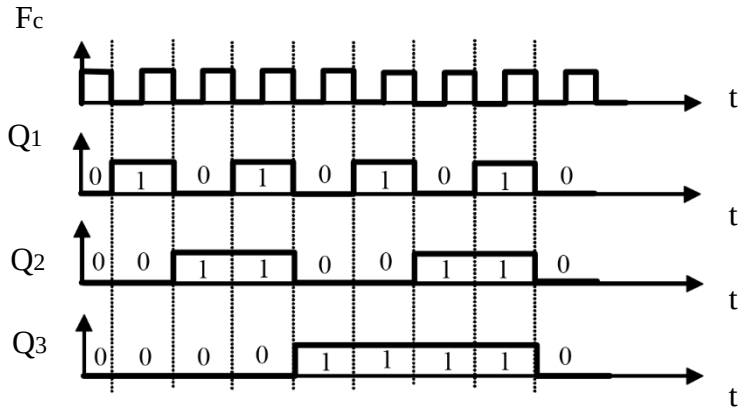
Счетчики с параллельным переносом удобнее реализовать на JK - триггерах. Здесь счетные импульсы подаются одновременно на счетные входы всех триггеров, образующих счетчик. Входы J и K используются для удержания триггеров от срабатывания в нужные моменты времени. Функциональная схема двоичного трехразрядного суммирующего счетчика с параллельным переносом, реализованная на JK – триггерах, приведена на рис. 25 а). Временные диаграммы (см. рис. 25 б)) поясняют его работу. Как видно из рисунка первый триггер срабатывает на каждый счетный импульс, второй триггер срабатывает через один импульс (на каждый второй), а третий – на каждый четвертый. Второй и третий триггеры удерживаются от срабатывания сигналами на J и K входах. Второй триггер может сработать на счетный импульс в случае, если первый триггер на выходе Q_1 имеет единичный потенциал в момент появления фронта счетного импульса. Третий триггер может сработать на счетный импульс в случае, если первый триггер на выходе Q_1 и второй триггер на выходе Q_2 имеют единичные потенциалы в момент появления фронта счетного импульса.

Если сравнить счетчики с последовательным и параллельным переносом, то следует отметить, что основным недостатком первых является накопление задержек сигнала. Это происходит потому, что триггер переходит из одного состояния в другое не мгновенно, а за определенное время, называемого временем задержки. Так как выходной сигнал каждого триггера в счетчике с последовательным переносом является входным для следующего триггера, то все задержки суммируются, и выходной сигнал последнего триггера сдвигается по фазе на величину суммарной задержки.

Лог. 1



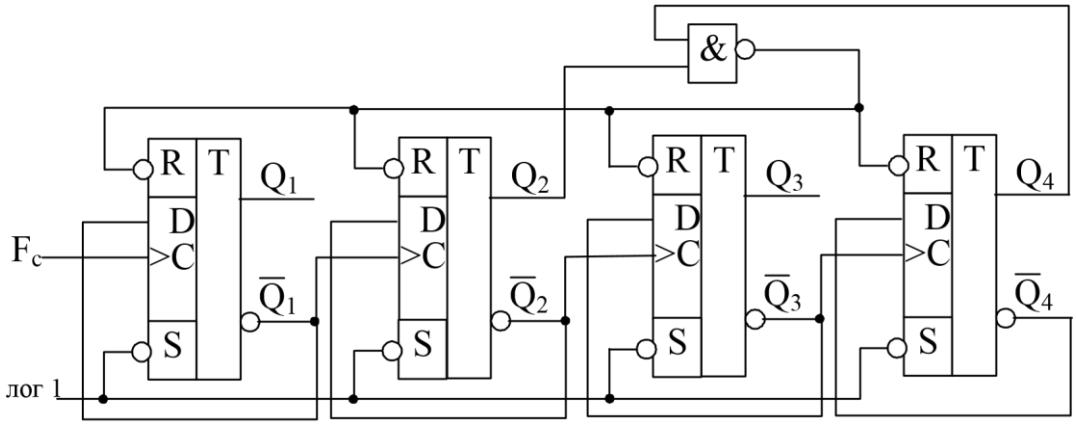
а)



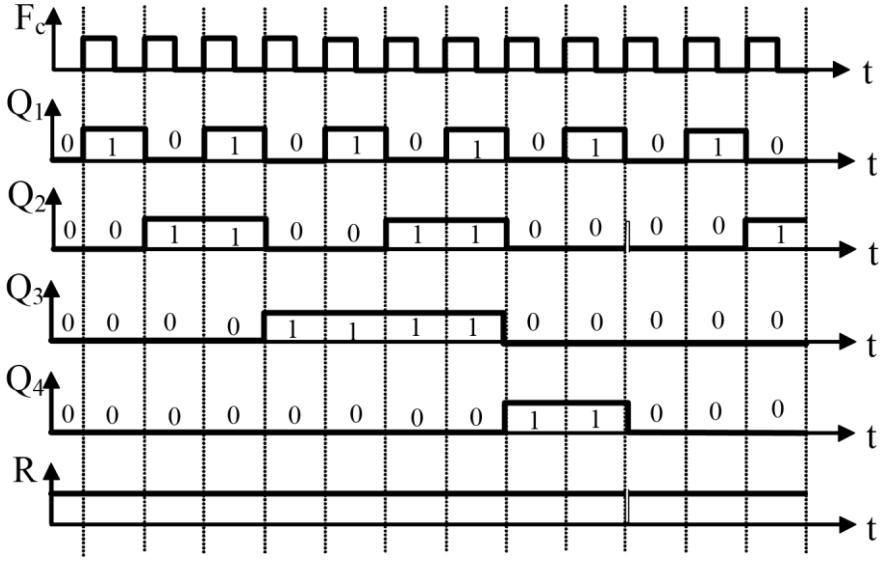
б)

Рис. 25. Функциональная схема трехразрядного двоичного суммирующего счетчика с параллельным переносом а) и временная диаграмма его работы б).

Счетчик с коэффициентом пересчета не кратным двойке реализуется путем исключения ряда его состояний. Например, для реализации двоично-десятичного счетчика нужно взять двоичный счетчик с коэффициентом пересчета 16 (четыре триггера) и исключить шесть состояний. Это можно сделать с помощью дешифратора соответствующего состояния счетчика. Выходной сигнал этого дешифратора подается на установочные входы триггеров и принудительно ставит их в нужные состояния. На рис. 26 а) приведена функциональная схема суммирующего двоично-десятичного счетчика (один двоично-десятичный разряд) с последовательным переносом, реализованная на D - триггерах с динамическим управлением записью. Исходное состояние счетчика – это нулевое состояние выходов всех четырех триггеров. Затем после прихода очередного счетного импульса триггеры принимают состояние 0001, затем 0010, … и т. д. до состояния 1001. В этот период времени счетчик работает как обычный двоичный счетчик. Но как только наступит состояние 1010, на выходе двухвходового логического элемента И-НЕ появится низкий потенциал (логический ноль), который поступит на входы R триггеров и все они перейдут в нулевое состояние, так как установочные входы имеют высший приоритет. Как только это произойдет, на выходе логического элемента И-НЕ вновь восстановится высокий потенциал, и счетчик опять начнет работать, как обычный двоичный пока не наступит состояние 1010. Затем все повторится и т.д. На рис. 26 б) приведены временные диаграммы, поясняющие работу двоично-десятичного счетчика. В данном случае были исключены состояния: 1010; 1011; 1100; 1101; 1110; 1111. Вообще, при реализации двоично-десятичного счетчика, возможно исключение любых других шести состояний, но приведенная схема является наиболее оптимальной с точки зрения аппаратных затрат.



а)



б)

Рис. 26. Функциональная схема двоично-десятичного суммирующего счетчика с последовательным переносом а) и временная диаграмма его работы б).

Как видно из временной диаграммы (рис. 26 б)), на входах R имеет место высокий потенциал (уровень логической единицы) до момента времени, когда счетчик переходит в состояние 1010. В этот момент напряжение переходит в уровень логического нуля (на обоих входах логического элемента И-НЕ оказываются единицы) и устанавливает все триггеры в нулевое состояние, то есть происходит сразу же замена состояния 1010 на 0000, и на выходе логического элемента И-НЕ и соответственно на входах R вновь устанавливается уровень логической единицы. Для дешифрации состояния 1010 в данном случае достаточно использовать двухвходовый логический элемент, так как ни в одном оставшемся состоянии счетчика не присутствуют одновременно единицы в разрядах 2^3 и 2^1 (1 0 1 0), 2^3 2^2 2^1 2^0

Аналогичным образом строятся счетчики с любым другим коэффициентом пересчета. Для этого определяется минимальное потребное

количество триггеров n, удовлетворяющее условию $2^n>k$, где k – коэффициент пересчета, и исключается соответствующее количество состояний, равное (2^n-k).

Кольцевые счетчики относятся к счетчикам с параллельным переносом и коэффициент пересчета у них равен количеству триггеров или количеству триггеров + 1. На рис.27 приведена функциональная схема кольцевого счетчика с коэффициентом пересчета 4, реализованная на D – триггерах с динамическим управлением записью. Счетчик последовательно принимает состояния: 111; 011; 101; 110; 111; 011; 101; 110 и т. д. Исходное состояние в этой схеме устанавливается автоматически.

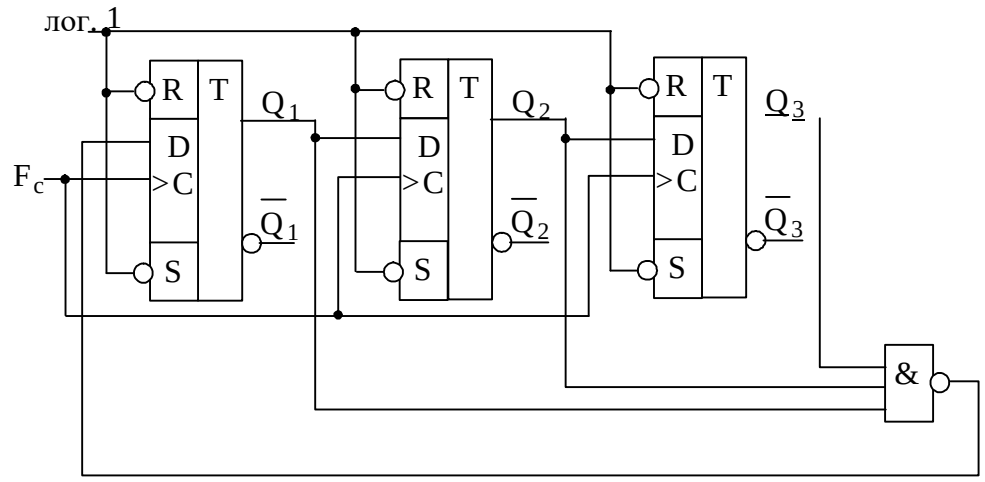


Рис. 27. Функциональная схема кольцевого счетчика с коэффициентом пересчета 4.

2.9. Сумматоры

Сумматор - это узел цифрового вычислительного устройства, выполняющий операцию суммирования (сложения) кодов двух чисел.

Сумматоры технически реализуются в виде комбинационных схем (без элементов памяти).

Складываемые числа могут быть представлены в двоичном, десятичном, двоично-десятичном или другом коде. В соответствии с этим сумматоры подразделяются на двоичные, десятичные, двоично-десятичные и т.п. В цифровой технике чаще всего применяются двоичные сумматоры.

Для сложения многоразрядных чисел используются многоразрядные сумматоры, которые**,** как правило, состоят из одинаковых ячеек одноразрядных сумматоров, имеющих два или три входа.

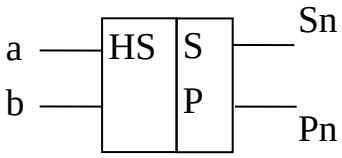
Для нахождения суммы многоразрядными сумматорами используются следующие способы:

- последовательно-поразрядно, начиная с младшего разряда;
- параллельно, т.е. одновременно всех разрядов;
- последовательно-параллельно, когда отдельно определяется сумма группы разрядов, а конечная сумма находится одновременным суммированием частных результатов.

В двоичной арифметике при сложении двух одноименных разрядов,

значение которых равно 1, образуется единица переноса в следующий, более старший разряд. Эта единица переноса либо не учитывается, тогда одноразрядный сумматор имеет два входа и называется полусумматором, либо складывается с суммой**,** полученной в старшем разряда, тогда одноразрядный сумматор имеет три входа и называется полным сумматором.

а. **Полусумматор одноразрядный.** Условное графическое обозначение полусумматора имеет вид, изображенный на рис.28.



a,b - значения складываемых разрядов
Sn, Pn- выходные сигналы,
Sn - сумма по модулю 2,
Pn - единица переноса.

Рис.28.

Логические уравнения, описывающие работу полусумматора, имеют вид:

$$\left.\begin{aligned} Sn &= a\bar{b} \cup \bar{a}b \\ Pn &= ab \end{aligned}\right\} \qquad (20)$$

Структурная схема полусумматора в соответствии с уравнениями имеет следующий вид (рис.29).

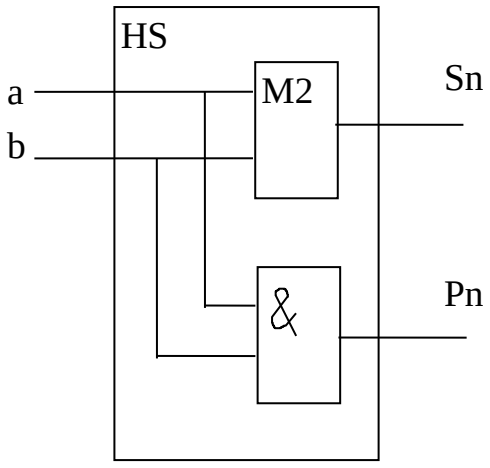


Рис.29

Таблица 20.

Входы		Выходы	
b	a	Sn	Pn
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
двоичные числа слагаемые		сумма	перенос
		исключ. “или”	“и”

В табл.20 приведена таблица истинности полусумматора.

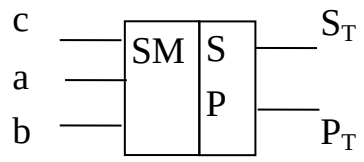
Для построения одноразрядного полусумматора на элементах И-НЕ представим уравнения (20) в базисе Шеффера:

$$\left.\begin{aligned} Sn &= a\bar{b} \cup \bar{a}b = \overline{\overline{a\bar{b}} \cup ab} = \overline{\overline{a \cup \bar{b} \cup ab}} = (a \cup b) \cup (\overline{ab}) = \\ &= a\overline{ab} \cup b\overline{ab} = \overline{\overline{a\overline{ab}} \cup \overline{b\overline{ab}}} = \overline{\overline{a}\overline{ab}\overline{b}\overline{ab}} \\ Pn &= ab = \overline{\overline{ab}} = \overline{\overline{a*b}*\overline{a*b}} \end{aligned}\right\} \qquad (21)$$

Таким образом , $\left.\begin{array}{l} Sn = \overline{\overline{\overline{a*\overline{a*b}}*\overline{b*\overline{a*b}}}} \\ Pn = \overline{\overline{a*b}*\overline{a*b}} \end{array}\right\}$ (22)

Для реализации логических уравнений (21) требуется 6 элементов двух входовых И-НЕ . Попробуйте самостоятельно составить логическую схему.

Б. Полный сумматор одноразрядный. Графическое обозначения полного сумматора имеет вид, изображенный на рис.30.



a,b-значения складываемых разрядов
c-единица переноса
S_T-полная сумма
P_T-единица переноса.

Рис.30.

Логические уравнения, описывающие работу полного сумматора, имеют вид:

$$\left\{\begin{array}{l} S_T = c(\overline{a+b}) \cup \bar{c}(a+b) = c(\overline{a\bar{b} \cup \bar{a}b}) \cup \bar{c}(a\bar{b} \cup \bar{a}b) = \\ = c(ab \cup \bar{a}\bar{b}) \cup \bar{c}(a\bar{b} \cup \bar{a}b) = abc \cup \bar{a}\bar{b}c \cup a\bar{b}\bar{c} \cup \bar{a}b\bar{c} \\ P_T = ab \cup cS_N = ab \cup c(a\bar{b} \cup \bar{a}b) = ab \cup a\bar{b}c \cup \bar{a}bc = \\ = a(b \cup \bar{b}c) \cup b(a \cup \bar{a}c) = a(b \cup c) \cup b(a \cup c) = \\ = ab \cup ac \cup ab \cup bc = ab \cup ac \cup bc \end{array}\right. \quad (23)$$

Структурная схема одноразрядного полного сумматора в соответствии с уравнениями (23) состоит из двух одноразрядных полусумматоров и одного двухвходового элемента “ИЛИ” (рис.31).

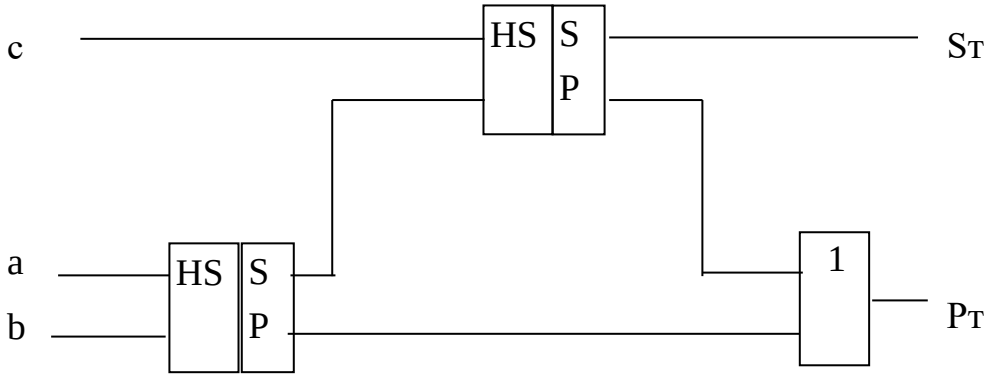


Рис.31

В таблице 22 приведена таблица истинности одноразрядного полного сумматора.

Табл.22.

Входы			Выходы	
c	b	a	S_T	P_T
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1
			Сумма	Перенос

Для построения одноразрядного полного сумматора на элементах И-НЕ представим уравнения (23) в базе Шеффера. Из преобразований для получения системы уравнений (21) можно сформулировать следующее правило перехода от логического уравнения, представленного в ДНФ, к уравнению в базисе Шеффера:

- заменить знаки логической связки дизьюнкции (V) на знаки логической связки коньюнции (Λ), при этом на каждую конъюнкцию поставить знак инверсии и поставить общую инверсию.

Используя это правило, получим:

$$\left.\begin{aligned} S_T &= a\bar{b}\bar{c} \cup \bar{a}\bar{b}c \cup \bar{a}b\bar{c} \cup abc = \overline{(\overline{a\bar{b}\bar{c}}) * (\overline{\bar{a}\bar{b}c}) * (\overline{\bar{a}b\bar{c}}) * (\overline{abc})} \\ P_T &= ab \cup ac \cup bc = \overline{\overline{ab} * \overline{ac} * \overline{bc}} \end{aligned}\right\} \qquad (24)$$

Для реализации уравнений (23) требуется 6 двухвходовых, 5 трехвходовых элементов и 1 четырехвходовой элемент “И-НЕ” (рис.32).

Рассмотрим принцип действия двухразрядного полного сумматора К155ИМ2. Это - простейшая схема без дополнительных инверсных и управляющих входов.

Графическое обозначения и цоколевка ИМС К155ИМ2 приведены на рис.33.

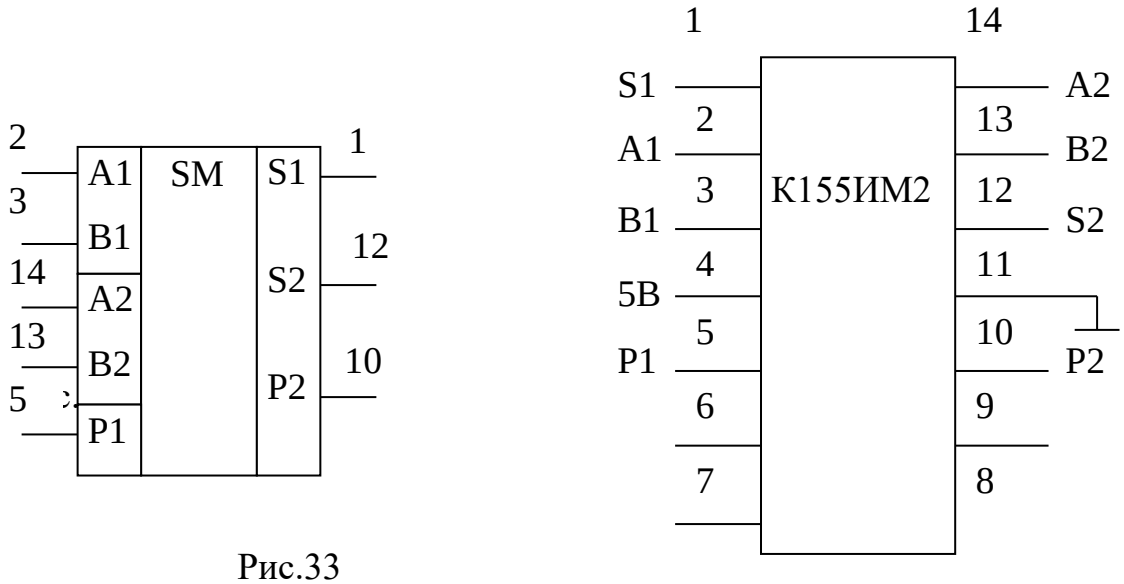


Рис.33

Структурная схема двухразрядного сумматора, построенного на ИМС К155ИМ2 имеет вид, показанный на рис.34.

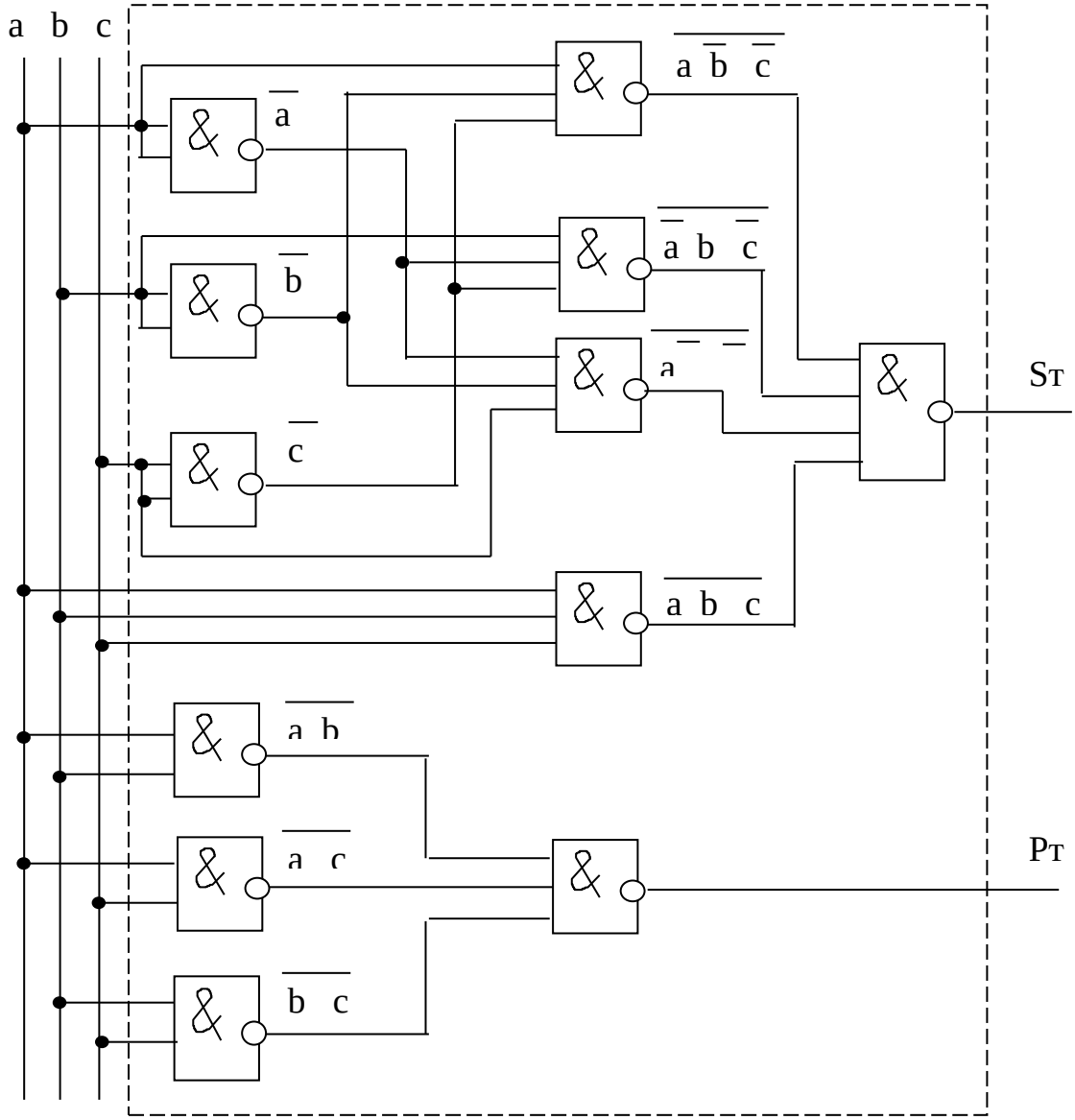
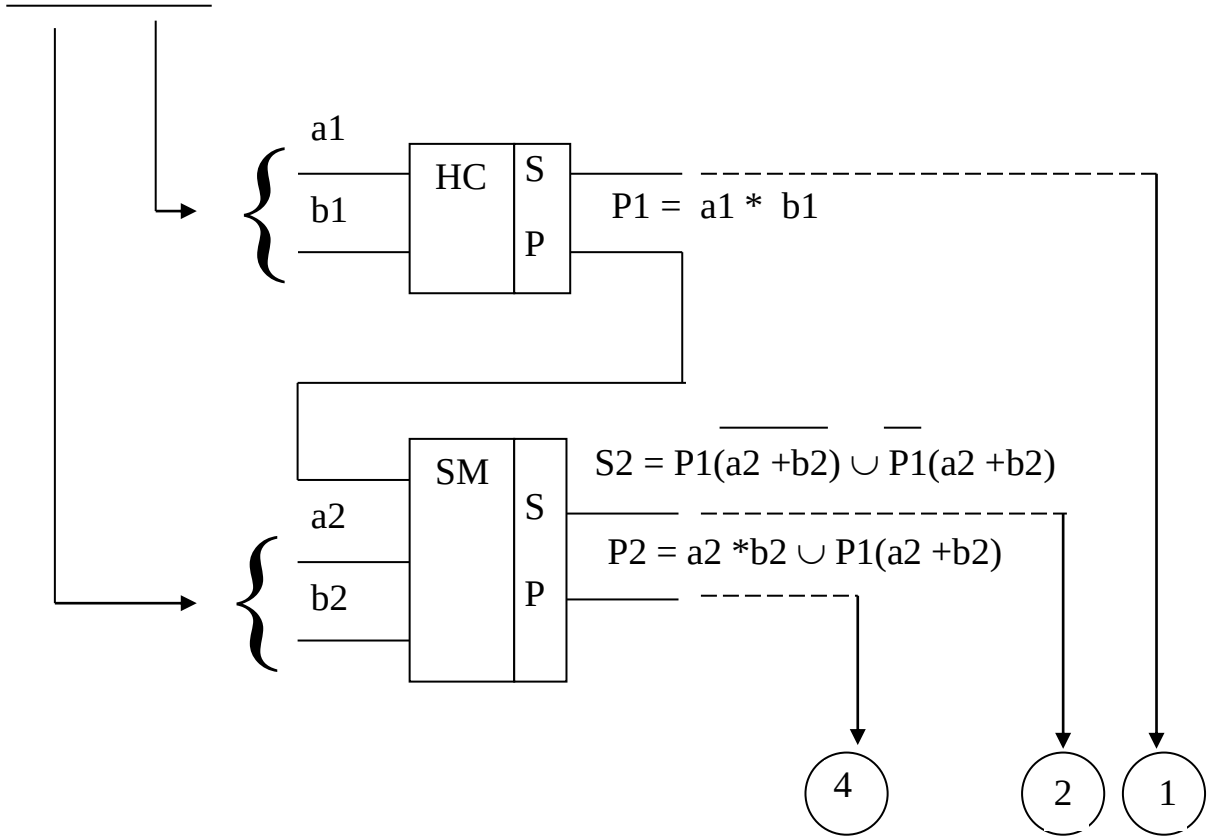


Рис.32



Используя законы Булевой алгебры, можно представить логические уравнения выходов S_1, S_2, P_2 в дизъюктивной совершенной нормальной форме (ДСФН).

$$S_1 = a_1 + b_1 = a_1\bar{b}_1 \cup \bar{a}_1 b_1$$

$$S_2 = a_1 b_2(\overline{a_2 + b_2}) \cup \overline{a_1 b_1}(a_2 + b_2) = a_1 b_1(\overline{a_2\bar{b}_2 \cup \bar{a}_2 b_2}) \cup \overline{a_1 b_1}(a_2\bar{b}_2 \cup \bar{a}_2 b_2) =$$

$$= a_1 b_1 a_2 b_2 \cup a_1 b_1 \overline{a_2 b_2} \cup \bar{a}_1 b_1 a_2\bar{b}_2 \cup \overline{a_1 b_1} a_2\bar{b}_2 \cup a_1\bar{b}_1 a_2\bar{b}_2 \cup \bar{a}_1 b_1\bar{a}_2 b_2 \cup \bar{a}_1\bar{b}_1\bar{a}_2 b_2 \cup a_1\bar{b}_1\bar{a}_2 b_2;$$

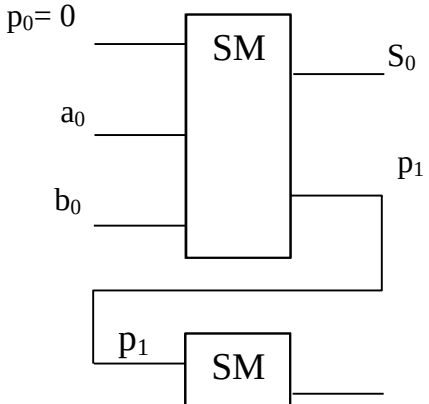
$$P_2 = a_2 b_2 \cup a_1 b_1(a_2 + b_2) = a_2 b_2 \cup a_1 b_1(a_2\bar{b}_2 \cup \bar{a}_2 b_2) =$$

$$= a_1 b_1 a_2 b_2 \cup \bar{a}_1 b_1 a_2 b_2 \cup a_1\bar{b}_1 a_2 b_2 \cup \bar{a}_1\bar{b}_1 a_2 b_2 \cup a_1 b_1 a_2\bar{b}_2 \cup a_1 b_1\bar{a}_2 b_2$$

Имея в наличии n одноразрядных сумматоров можно построить n-разрядный параллельный сумматор. На рис. 35 приведена функциональная схема n-разрядного параллельного сумматора с последовательным переносом. Здесь все разряды суммируемых двоичных кодов обрабатываются одновременно, но перенос от одного разряда к другому формируется последовательно, что увеличивает время выполнения операции.

Можно реализовать схему с параллельным формированием переноса для всех разрядов.

Затраты оборудования на построение сумматора такого типа при большом числе разрядов очень велики.



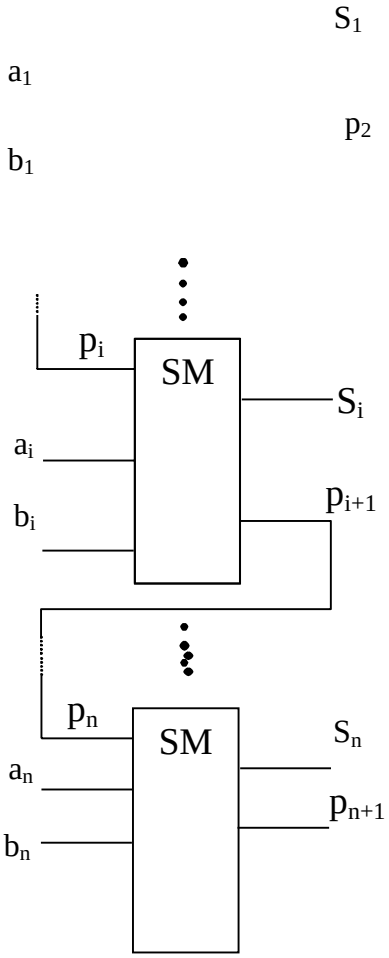


Рис. 35. Функциональная схема n-разрядного параллельного сумматора с последовательным переносом.

Для уменьшения аппаратных затрат реализуют сумматоры с групповым Переносом рис.36. В этом случае сумматор разбивается на несколько групп, равной
длины. Внутри групп перенос формируется параллельно, а между группами последовательно, либо в группе формируется ускоренный перенос при наличии условий распространения переноса во всех разрядах данной группы, который передается на младший разряд соседней, более старшей группы.

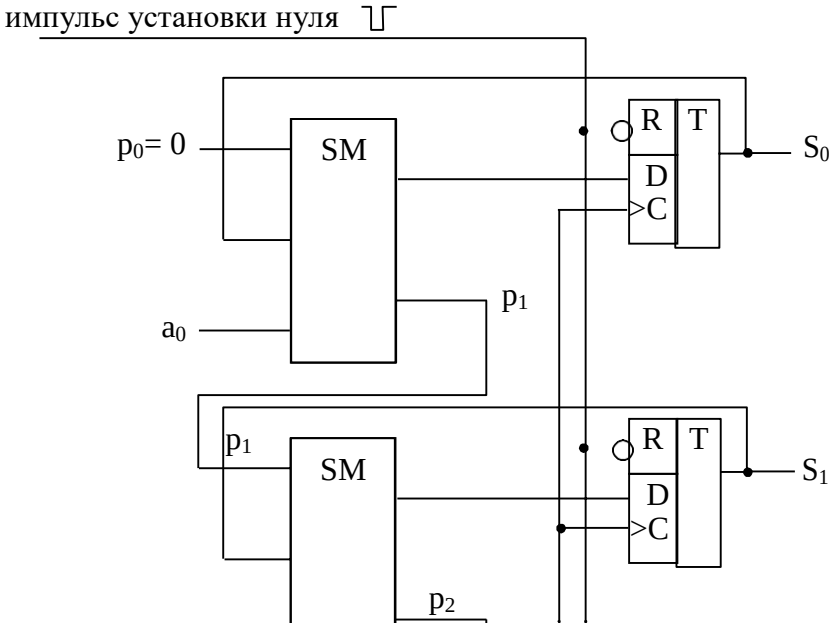


Рис. 36. Функциональная схема n-разрядного параллельного накапливающего сумматора с последовательным переносом.

Рассмотренные сумматоры являются синхронными. Время для выполнения операции здесь выделяется максимальное, то есть такое, чтобы успели завершиться все переносы в самом неблагоприятном случае.

В асинхронных сумматорах дополнительно формируется сигнал окончания всех переносов при выполнении сложения, что увеличивает аппаратные затраты, но увеличивает быстродействие. Появление этого сигнала свидетельствует об окончании текущей операции и разрешает начать следующую операцию. В этом случае время сложения различных операндов различно, и каждая операция выполняется за реально требуемое время.

2.10. Дешифраторы

Дешифраторами называются цифровые электронные устройства, имеющие много входов и много выходов и преобразующие параллельный двоичный код в параллельный унитарный двоичный код. Различают полные дешифраторы, способные преобразовывать все возможные коды на входе и неполные, которые преобразуют только часть или даже один из возможных кодов.

Если дешифратор предназначен для преобразования лишь одного из возможных кодов, то он называется дешифратором данного кода и имеет

один выход. На рис. 37 приведено условное графическое изображение полного дешифратора n – разрядного двоичного кода. Как видно из рисунка количество разрядов выходного кода равно 2^n. Схема полного дешифратора трехразрядного двоичного кода приведена на рис. 38 а таблица истинности 23 описывает его функционирование.

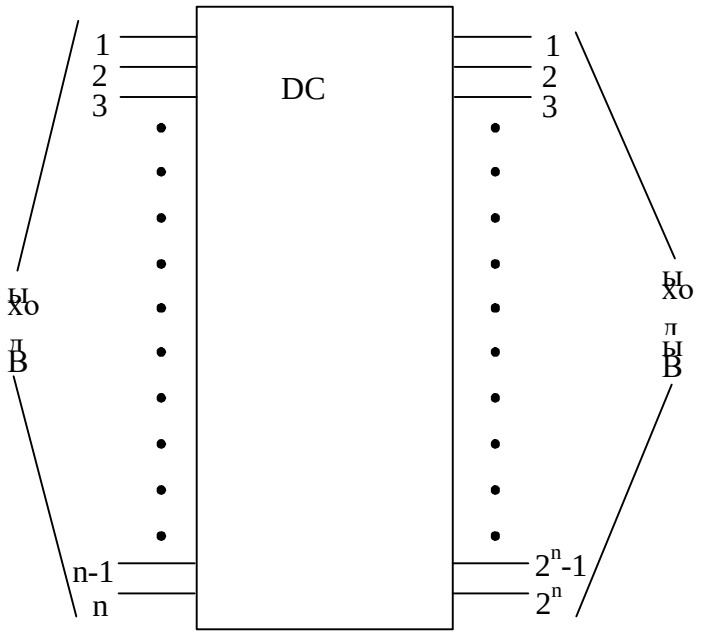


Рис. 37. Полный дешифратор n – разрядного двоичного кода.

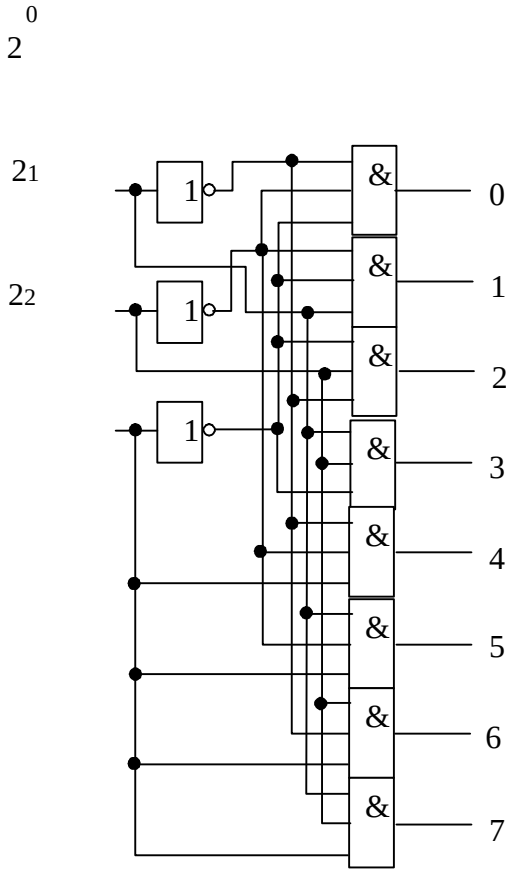


Рис. 38. Схема полного дешифратора трехразрядного двоичного кода.

Таблица 23

Входы			Выходы							
2_2	2_1	2_0	0	1	2	3	4	5	6	7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Каждый из трехвходовых логических элементов «И» на рис. 38 в совокупности с тремя инверторами образует дешифратор одного из восьми двоичных кодов. По единичному состоянию одного из выходов можно определить, какой двоичный код поступил на вход дешифратора. Например, если на выходе «3» единичный потенциал, то это означает, что на входе дешифратора присутствует код 011.

Дешифраторы используются при построении схем других цифровых устройств: счетчиков с различны им коэффициентами пересчета, шифраторов, мультиплексоров, демультиплексоров, схем индикации и т.д.

2.11. Шифраторы

Шифраторами называют цифровые электронные устройства, преобразующие двоичный параллельный код из одной системы кодирования в другую. Составной частью шифратора является дешифратор, который определяет, какой код присутствует на входе. На рис. 39 приведена структурная схема шифратора. Формирователь выходного кода представляет собой комбинационную схему, реализующую заданную логическую функцию.

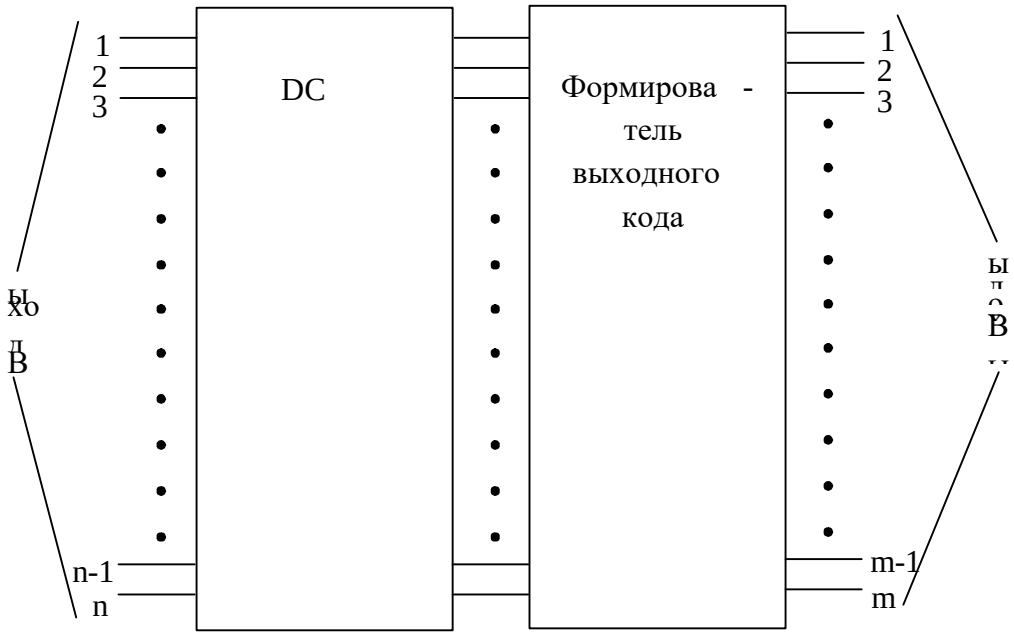


Рис.39. Структурная схема шифратора n – разрядного двоичного кода в m – разрядный двоичный код.

Необходимость преобразования кодов можно наглядно показать на примере формирования кодов управления приборами индикации. Так, например, для управления семисегментным индикатором требуется семиразрядный код. Если нам необходимо высвечивать в виде десятичных цифр состояние одного из разрядов двоично-десятичного счетчика, то необходимо преобразовать четырехразрядный двоично-десятичный код в семиразрядный двоичный код. Каждый разряд этого кода будет управлять зажиганием соответствующего сегмента.

Условное изображение семисегментного индикатора приведено на рис. 40. Будем считать, что сегмент светится, если разряд двоичного кода управляющий его зажиганием равен единице. Тогда таблица 24 описывает закон преобразования четырехразрядного двоично-десятичного кода в семиразрядный двоичный код управления зажиганием сегментов.

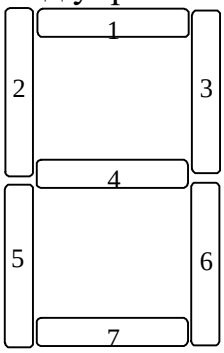


Рис. 40. Семисегментный индикатор

Таблица 24

Двоично-десятичный код (формат 8421)				Семиразрядный двоичный код управления индикатором						
				1	2	3	4	5	6	7
0	0	0	0	1	1	1	0	1	1	1

0	0	0	1	0	0	1	0	0	1	0
0	0	1	0	1	0	1	1	1	0	1
0	0	1	1	1	0	1	1	0	1	1
0	1	0	0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	1	0	1	1
0	1	1	0	1	1	0	1	1	1	1
0	1	1	1	1	0	1	0	0	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

Функциональная схема шифратора, реализующего закон преобразования, соответствующий таблице 24, представлена на рис. 41.

Каждый разряд кода управляет зажиганием сегмента с соответствующим номером. Так, например, сегмент с номером 1 должен гореть в случае поступления двоично-десятичных кодов, соответствующих 0, 2, 3, 5, 6, 7, 8, 9. Поэтому выходы дешифратора (DC) 0, 2, 3, 5, 6, 7, 8, 9 объединяются по ИЛИ и формируется разряд двоичного кода для управления зажиганием первого сегмента. Аналогично формируются остальные разряды двоичного кода.

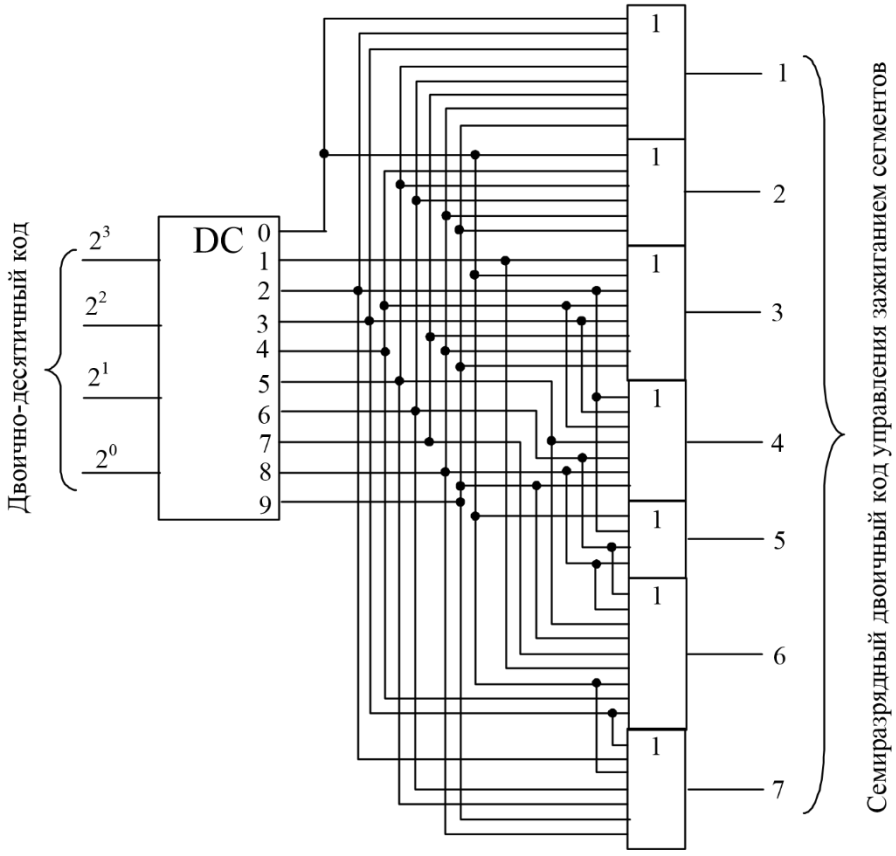


Рис. 41. Функциональная схема шифратора, формирующего управляющий код для зажигания семисегментного индикатора.

2.12. Мультиплексоры

Мультиплексорами называют цифровые электронные устройства, имеющие много входов и один выход. Все входы разделяются на две группы. Первая группа называется информационными входами, а вторая – управляющими. Мультиплексор реализует функцию коммутации любого из информационных входов на выход. Номер коммутируемого информационного входа определяет двоичный код на управляющих входах.

На рис .42 приведено условное графическое изображение мультиплексора содержащего n (Вх.$и_1$, Вх.$и_2$,…, Вх.$и_n$) информационных и m (Вх.$у_1$, Вх.$у_2$,…, Вх.$и_m$) управляющих входов. Соотношение между количеством информационных и управляющих входов определяется формулой $m=\log_2 n$.

Функциональная схема мультиплексора на четыре информационных входа приведена на рис. 43, а таблица 25 поясняет его работу. В состав мультиплексора входит полный дешифратор двухразрядного двоичного кода, формирующий сигналы, поступающие на четыре двухвходовых логических элемента «И». На другие входы этих же элементов поступают информационные сигналы, а их выходы объединены по «ИЛИ». Уровни напряжения на информационных входах обозначены буквами A, B, C, D, и они могут принимать значения логической единицы или логического нуля.

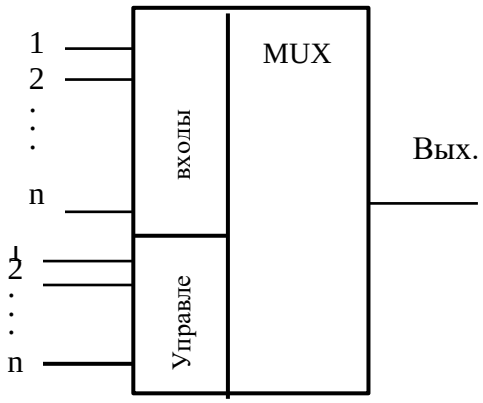


Рис.42. Условное графическое изображение мультиплексора

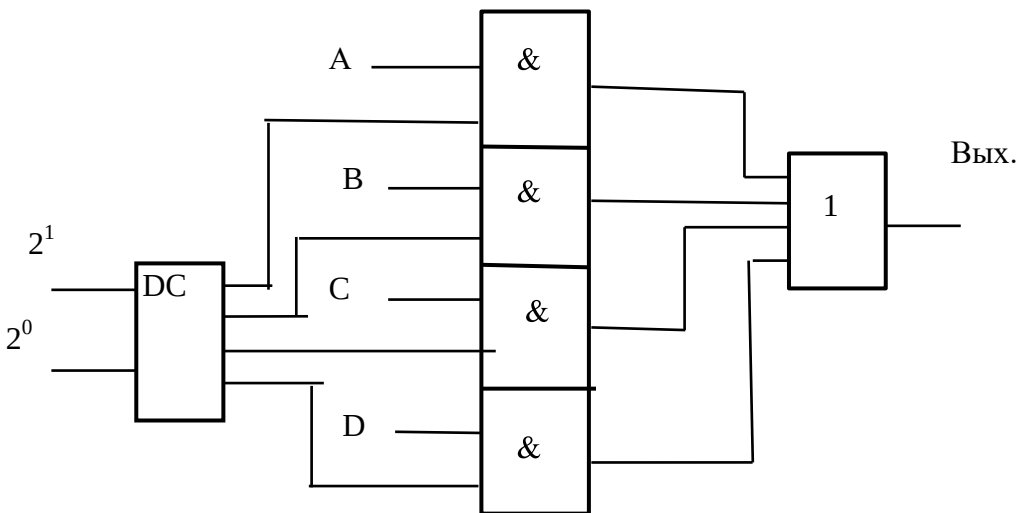


Рис. 43. Функциональная схема мультиплексора на четыре информационных входа.

Таблица 25

Управляющие входы		Сигнал на выходе
2^1		
0	0	A
0	1	B
1	0	C
1	1	D

Код на управляющих входах (2^1, 2^0) определяет, какой из входных информационных сигналов проходит на выход в данный момент. В качестве примера на рис. 43 указаны логические уровни сигналов в электрических цепях для кода: $2^1=0$ и $2^0=0$. Как видно из рисунка, выходной сигнал в этом случае определяется входом А.

В качестве примера рассмотрим принцип действия мультиплексора К155КП5, который позволяет коммутировать данные от восьми входов на общую выходную линию. Он имеет три адресных (управляющих) входа S_1, S_2, S_3 (рис.44), активный уровень которых высокий (“В” = “1”). Таблица истинности мультиплексора К155КП5 приведена в таблице 26.

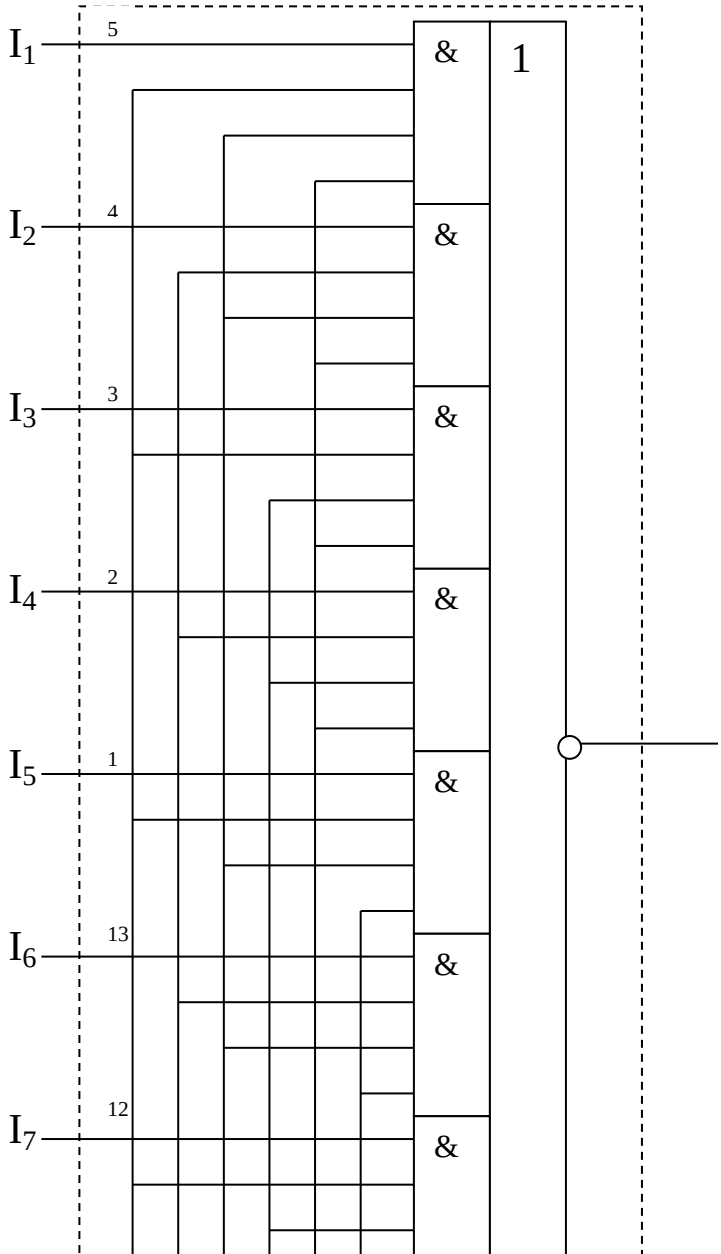
Таблица 26. Таблица истинности ИМС К155КП5.

Выходы упр.			***Выходы информационные***								***Выход***
S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	**Y**
0	0	0	0	X	X	X	X	X	X	X	1
0	0	0	1	X	X	X	X	X	X	X	0
0	0	1	X	0	X	X	X	X	X	X	1
0	0	1	X	1	X	X	X	X	X	X	0
0	1	0	X	X	0	X	X	X	X	X	1
0	1	0	X	X	1	X	X	X	X	X	0
0	1	1	X	X	X	0	X	X	X	X	1
0	1	1	X	X	X	1	X	X	X	X	0
1	0	0	X	X	X	X	0	X	X	X	1
1	0	0	X	X	X	X	1	X	X	X	0
1	0	1	X	X	X	X	X	0	X	X	1
1	0	1	X	X	X	X	X	1	X	X	0
1	1	0	X	X	X	X	X	X	0	X	1
1	1	0	X	X	X	X	X	X	1	X	0
1	1	1	X	X	X	X	X	X	X	0	1

1	1	1	X	X	X	X	X	X	X	1	0

Логическая функция мультиплексора К155КП5 как управляемого восьмипозиционного ключа описывается уравнением, полученным по таблице истинности:

$$Y= I_1S_0S_1S_2 \vee I_2S_0S_1S_2 \vee I_3S_0S_1S_2 \vee I_4S_0S_1S_2 \vee I_5S_0S_1S_2 \vee I_6S_0S_1S_2 \vee I_7S_0S_1S_2 \vee I_8S_0S_1S_2$$



а)

а)

1 14 **5В**

2 3 12

S_2

4 11 5 10 6

9 7 8

б)

5
4
3
2
1 13
12
10
9
8

I_5
I_4
I_3
I_2
I_1

К155КП5

I_6
I_7
I_8
S_0

11
12
13
14
15
16
17
18

MS

S0
S1
S2

а) – схема мультиплексора К155КП5;

б) – цоколевка ИМС мультиплексора К155КПКП5;

в) – условное графическое обозначение мультиплексора;

2.13. Демультиплексоры

Демультиплексор реализует функцию коммутации сигнала единственного информационного входа на один из многих выходов. Номер выхода определяется двоичным кодом на управляющих входах. Таким образом, демультиплексор представляет собой электронное цифровое устройство с одним информационным входом и имеет управляющие входы и информационные выходы. На рис.45 приведено условное графическое изображение демультиплексора содержащего n (Вых.$и_1$, Вых.$и_2$,…, Вых.$и_n$) информационных выходов и m (Вх.$у_1$, Вх.$у_2$,…, Вх.$и_m$) управляющих входов. Соотношение между количеством ин- формационных выходов и управляющих входов определяется формулой $m=\log_2 n$.

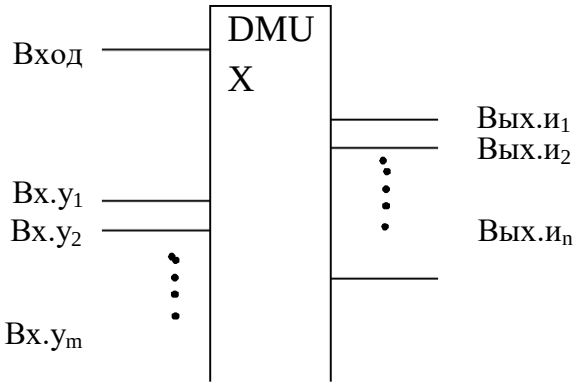


Рис. 45 Условное графическое изображение демультиплексора.

Таблица 27

Управляющие входы		Значение сигнала на информационных выходах			
2^1	2^0	Вых. $и_1$	Вых. $и_2$	Вых. $и_3$	Вых. $и_4$
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

Функциональная схема демультиплексора на четыре информационных выхода приведена на рис. 28 а таблица 45 поясняет его работу. В качестве примера на рис. 46 указаны логические уровни сигналов в электрических цепях для кода: $2^1=0$ и $2^0=0$. Как видно из рисунка, выходной сигнал Вых.1 в

этом случае определяется входом А, а на всех остальных выходах имеет место уровень логического нуля.

Рис. 46 Функциональная схема демультиплексора на четыре информационных выхода.

2.14. Контрольные вопросы

1. Чем отличается потенциальное представление информации от импульсного?
2. Чем отличается параллельный код от последовательного?
3. Где используют импульсное и фазоманипулированное представление информации?
4. Какая функция называется булевой?
5. Что понимают под функционально полной системой логических функций?
6. Какие системы логических функций обладают функциональной полнотой?
7. Что понимают под термином «логический элемент»?
8. Перечислите известные типы логики.
9. Какой тип логики обладает наибольшим быстродействием и почему?
10. В чем особенности логики $И^2Л$?
11. Перечислить основные свойства логических элементов.
12. Перечислить основные параметры ЛЭ.
13. Привести расшифровку условного обозначения ИМС.
14. Как конструктивно оформлена ИМС.
15. Знаете ли вы таблицы истинности логических элементов И, ИЛИ, И-НЕ, ИЛИ-НЕ, Исключающее ИЛИ ?
16. Триггеры и их классификация.
17. Какое устройство называют триггером?
18. Схемотехника статических триггеров и таблицы истинности их функционирования.
19. Схемотехника динамических триггеров и таблицы истинности их функционирования.
20. Приведите таблицы истинности для RS-триггера, D-триггера и JK-триггера.
21. Дайте определение дешифратора.

22. Какой дешифратор называется полным?
23. Каково соотношение между количеством входов и выходов в полном дешифраторе двоичного кода?
24. Приведите таблицу истинности для полного дешифратора трехразрядного двоичного кода.
25. Какое устройство называют мультиплексором?
26. Дайте определение демультиплексора.
27. Как реализуются регистры?
28. Перечислите типы регистров.
29. Какую функцию выполняет параллельный регистр?
30. Какую функцию выполняет последовательный сдвиговый регистр?
31. Какую функцию выполняет кольцевой сдвиговый регистр?
32. Что называют счетчиком?
33. Чем отличается счетчик с параллельным переносом от счетчика с последовательным переносом?
34. Какие счетчики называют двоичными?
35. Дайте определение сумматора.
36. Приведите таблицу истинности для одноразрядного комбинационного сумматора.
37. Приведите схему параллельного комбинационного сумматора с последовательным переносом.
38. Какой сумматор называют накапливающим?
39. Каково функциональное назначение сумматора?
40. Какие знаете сумматоры по принципу действия ?
41. Какие знаете способы суммирования многоразрдных чисел?
42. Напишите уравнения выходов одноразрядных полусумматора и полного сумматора.
43. По какому признаку счетчики разделяются суммирующие, вычитающие и реверсивные?
44. Какой недостаток у счетчиков с последовательным переносом?
45. Дайте определение шифратора.
46. Составьте таблицу преобразования двоично-десятичного кода формата 8421 в семиразрядный двоичный код для управления семисегментным индикатором.

Литература.

1. В. Л. Бройдо, О. П. Ильина Архитектура ЭВМ и систем. СПб.: Питер, 20 09 г. – 720 с.: ил.
2. А. П. Жмакин Архитектура ЭВМ. СПб.: БХВ-Петербург, 2010. — 352 с: ил.
3. Кучумов А.И. Электроника и схемотехника: Учебное пособие. – М.: Гелиос АРВ, 2002.
4. Н. В. Максимов, И. И. Попов, Т. Л. Партыка Архитектура ЭВМ и вычислительных систем: Учебник. – М.: ФОРУМ: ИНФРА-М, 2010. – 512 с.: ил. – (Профессиональное образование).
5. Д. Паттерсон, Дж. Хеннесси Архитектура компьютера и проектирование компьютерных систем. СПб.: Питер, 2012 г. – 784 с.: ил.
6. С. Л. Сергеев Архитектуры вычислительных систем. СПб.: БХВ-Петербург, 2010. — 240 с: ил.
7. Столлингс В. Структурная организация и архитектура компьютерных систем. 5-е издание. - М.: Издательский дом "Вильямс", 2002. - 896 с.
8. Таненбаум Э. Архитектура компьютера. СПб.: Питер, 2012 г. – 704 с.: ил.
9. Титце У., Шенк К. Полупроводниковая схемотехника: Справочное руководство. Пер. с нем.– М.: Мир, 1982.
10. Соловьев Г.Н. Схемотехника ЭВМ. - М.: Высшая школа 1985г.
11. Каган Б.М. Электронные вычислительные машины и системы. – М.: Энергоатомиздат 1985г.
12. Шило В.Л. Популярные цифровые микросхемы. Справочник. – М.: Радио и связь 1987г.198 с.
13. Милозоров В.П. Элементы информационных систем. - М. Высшая школа, 1989, с. 25-39.
14. Орлов С.А. Организация ЭВМ и систем: учебник для вузов / С.А. Орлов, Б.Я. Цилькер. – 3-е изд. – СПб.: Питер, 2014. – 688 с.
15. Новожилов О.П. Архитектура ЭВМ и системы. Учебное пособие для бакалавров. – М.: Изд-во Юрайт, 2015. – 527 с.
16. Бройдо В.Л. Вычислительные системы, сети и телекоммуникации: учебное пособие для вузов / В.Л. Брайдо, О.П. Ильина. – 4-е изд. – СПб.: Питер, 2011. – 555 с.

Интернет-ресурсы

1. Официальный сайт компании Intel, США. – http:// www.intel.com
2. Официальный сайт компании AMD, США. – http:// www.amd.com
3. Официальный сайт компании HP, США. – http:// www.hp.com
4. Официальный сайт компании IBM, США. – http:// www.ibm.com
5. Сайт информационных технологий. – http:// www.ixbt.com
6. Сайт высоких технологий IT-индустрии. – http://citforum.ru

Канаев Магомедимин Муталимович

Основы построение эвм

Учебное пособие по изучению курса «Архитектура ЭВМ и основы программирование на ассемблере», для студентов направления подготовки «прикладная математика и информатика».