ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ (МГТУ ГА) »

Кафедра вычислительных машин, комплексов, систем и сетей

А.И. Терентьев

АППАРАТНЫЕ СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Учебное пособие

Утверждено Редакционно-издательским советом МГТУ ГА в качестве учебного пособия

Москва
2019

УДК 681.3
ББК 6Ф7.3
Т35

Печатается по решению редакционно-издательского совета
Московского государственного технического университета ГА

Рецензенты:
Романчева Н.И. (МГТУ ГА) – канд. техн. наук., доцент,
Сычев А.М. (Департамент информационной безопасности Банка России) –
канд. техн. наук, доцент.

Терентьев А.И.
Т35 Аппаратные средства вычислительной техники: учебное пособие. / А.И. Терентьев — Воронеж.: ООО «МИР», 2019. — 72 с.

ISBN 978-5-6042813-0-7

Учебное пособие содержит краткий лекционный материал по дисциплине «Аппаратные средства вычислительной техники». В учебном пособии системно и доступно рассмотрены способы представления чисел и выполнения арифметических операций в ЭВМ, основополагающие принципы алгебры логики и базовые электрические (электронные) логические элементы и схемы современных аппаратных средств вычислительной техники. В конце каждого раздела учебного пособия приводятся контрольные вопросы и задачи (задания).

Учебное пособие издается в соответствии с учебным планом для обучающихся по специальности 10.05.02 Информационная безопасность телекоммуникационных систем и сетей (специалитет) очного обучения.

Рассмотрено и одобрено на заседании кафедры 24.04.2018 г. и методического совета 26.04.2018 г.

ББК 6Ф7.3
Св. тем. план 2018 г.
поз. 44

ТЕРЕНТЬЕВ Андрей Иванович

АППАРАТНЫЕ СРЕДСТВА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
Учебное пособие

В авторской редакции

Подписано в печать 19.11.2018 г.
Формат 60x80/16 Печ. л. Усл. печ.
л. Заказ 412/090454 Тираж 30 экз.

Московский государственный технический университет ГА
125993 Москва, Кронштадтский бульвар, д.20
Отпечатано ООО «МИР»
394033, г. Воронеж, Ленинский пр-т 119А, лит. Я, оф. 215

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящее учебное пособие строится на материале, который читается для обучающихся по специальности 10.05.02 Информационная безопасность телекоммуникационных систем и сетей (специалитет) очного обучения по дисциплине «Аппаратные средства вычислительной техники».

В настоящее время современные средства вычислительной техники в различных видах и конструктивном исполнении используются человеком во всех сферах его жизнедеятельности и являются необходимым инструментом дальнейшего развития научно-технического прогресса. При этом, несмотря на высокие темпы разработки, производства и вывода на потребительский рынок новых, все более совершенных, производительных, компактных и универсальных вычислительных средств и устройств, фундаментальные принципы их действия, а также лежащие в их основе базовые цифровые элементы остаются неизменными и подлежат обязательному изучению студентами соответствующих технических специальностей и направлений.

В настоящее время в общедоступном коммуникационном пространстве, в том числе глобальной сети Интернет, в большом количестве имеются разнообразные узкопрофессиональные и популярные электронные издания, источники и публикации, посвященные различным вопросам вычислительной техники. При этом отсутствуют компактные специализированные учебные пособия, в которых системно и доступно на необходимом научно-техническом и педагогическом уровне рассматриваются основополагающие принципы и элементы современных аппаратных средств вычислительной техники.

Настоящее учебное пособие по мнению автора в определенной степени исправляет сложившуюся ситуацию, поскольку в нем в достаточно сжатом виде содержится необходимый теоретический материал и последовательно рассматриваются базовые логические, цифровые и конструктивные элементы аппаратных средств вычислительной техники, а также архитектура, состав аппаратных средств и конструкция персональной ЭВМ, назначение, принцип действия и технические характеристики микропроцессора.

Материалы курса «Аппаратные средства вычислительной техники» являются обработанными и оптимизированными сведениями, содержащимися в открытых источниках и программах ведущих учебных заведений.

Пособие рассчитано на студентов, обучающихся по специальности 10.05.02 Информационная безопасность телекоммуникационных систем и сетей (специалитет) очного обучения, а также слушателей высших учебных заведений, обучающихся по техническим дисциплинам. Может быть использовано при проведении практических занятий по дисциплине «Аппаратные средства вычислительной техники», а также студентами других специальностей и слушателями курсов повышения квалификации при изучении вопросов, связанных с основными понятиями и базовой элементной базой современных средств вычислительной техники.

ГЛАВА 1. ОБЩИЕ ИСТОРИЧЕСКИЕ СВЕДЕНИЯ И МАТЕМАТИЧЕСКИЕ ОСНОВЫ

1.1. Этапы развития средств автоматизации вычислений и ЭВМ. Поколения ЭВМ

Отправной точкой процесса создания вычислительных устройств и машин считается изобретение более 1500 лет назад простейших механических «счетов» (арабское название – «абак»), которые оказались достаточно эффективным и дешевым инструментом для осуществления арифметических операций сложения и вычитания и практически в неизменном виде широко использовались как вычислительное устройство до середины 70-х годов XX века.

С развитием математики, как науки, ученые стремились создать механизмы или машины, облегчающие трудоемкий процесс вычислений. Принято считать, что первая механическая счетная машина появилась в 1642 году, благодаря молодому французскому изобретателю и математику Блезу Паскалю (1623-1662). Она представляла собой совокупность взаимодействующих зубчатых колес (диск единиц был связан с диском десятков, диск десятков был связан с диском сотен и т. д.) и могла суммировать шестизначные десятичные числа. Другие операции выполнялись с помощью неудобной процедуры повторных сложений, что являлось ее основным недостатком. Однако предложенный Паскалем принцип вычислений посредством связанных зубчатых колес явился основой для построения большинства современных механических вычислительных устройств.

Все четыре арифметических операции выполняла механическая машина, созданная в 1673 году немецким математиком Готфридом Вильгельмом Лейбницем (1546—1716), которая стала прототипом современных механических «арифмометров», широко используемых человечеством с начала XIX века до 70-х годов XX века.

Например, в 1874 году в России в г. С-Петербурге Вильгодтом Теофилом Однером (1845-1905) был изготовлен первый образец его «арифмометра», который после ряда усовершенствований (в частности, Однер вместо ступенчатых валиков Лейбница применил более совершенные и компактные зубчатые колеса с меняющимся числом зубцов) с 1890 года начал выпускаться серийно и имел большой успех, в связи с чем получил в России название «Арифмометр Однера».

С начала XIX века ученые и изобретатели многих стран, практически одновременно, предлагали различные идеи и решения, приближающие эру современных электронных вычислительных машин (ЭВМ).

Принципы построения управляемой программой «аналитической» счетной машины, использующей десятичную систему счисления и имеющей арифметическое устройство, устройства управления, ввода и вывода (печати)

были предложены в 1822 году английским математиком Чарльзом Бэббиджем. Совместно с ним над созданием программ для его счетных машин работала графиня Огаста Ада Лавлейс (1815-1852), дочь поэта Байрона, которой были введены многие первичные понятия современного программирования. К сожалению, их идеи существенно опережали технические возможности того времени и не были в полной мере реализованы на практике, однако по сути явились фундаментом для создания современных компьютеров.

Джордж Буль (1815 – 1864) создал алгебру логики, благодаря которой стало возможным оперировать различными высказываниями и утверждениями подобно тому, как в математике оперируют обычными числами. Основные операции такой алгебры – И, ИЛИ и НЕ, получили название «булевый базис».

Профессор физики Атанасофф Джон Винсент (1903-1995) одним из первых спроектировал и практические построил в 1941 году цифровую вычислительную машину на основе двоичной системы счисления. Достоинствами двоичной системы счисления являлись простота физического представления двух символов 0 и 1 в электрических схемах, а также простота основных математических операций с этими символами в вычислительной машине. К сожалению, в связи с началом Второй мировой войны работы по дальнейшей реализации этого проекта были остановлены.

Формально первым в мире компьютером принято считать электромеханическую (релейную) программируемую цифровую машину Z1, построенную в 1938 году немецким изобретателем Конрадом Цузе (немецкое написание фамилии – Zuse). Машина по логике построения была подобна аналитической машине Бэббиджа, занимала площадь 4 кв. м., имела клавиатуру для ввода данный и команд, а также панель с множеством лампочек для отображения результатов вычислений. Позднее Цузе стал кодировать и вводить данные и команды для машины, пробивая отверстия в 35-миллиметровой фотопленке. Усовершенствованная таким образом машина получила название – Z2.

В 1941 году Цузе построил программно-управляемую машину, основанную на двоичной системе счисления (получила название – Z3), которая по многим характеристикам превосходила вычислительные машины того времени, созданные независимо от него в других странах.

Первой электронной вычислительной машиной (ЭВМ) считается машина ЭНИАК (ENIAK, Electronic Numerical Integrator and Computer – электронный цифровой интегратор и вычислитель), построенная в 1945 году в Пенсильванском университете по заказу артиллерийского управления армии США коллективом американских ученых под руководством Дж. Мочли и Преспера Экерта. ЭВМ предназначалась для расчета траекторий полетов снарядов, имела объем 85 куб. м и массу около 30 тонн, содержала 18 тысяч электронных ламп и 1500 реле, потребляла около 150 кВт электроэнергии. Однако данные кодировались в этой ЭВМ в десятичной системе счисления. Позже, в 1950 году, ими была создана двоичная ЭВМ с программным

обеспечением, хранимым в памяти машины, получившая название ЭДВАК — Электронный Автоматический Вычислитель с дискретными переменными.

Джон фон Нейман (1903 – 1957), работая в группе Дк. Мочли и П. Экерта, обобщил работы по цифровым электронным машинам, предложив пять ключевых элементов компьютерной архитектуры, получившей название «архитектура фон Неймана».

Следует отметить, что начиная с 40-х годов XX века все работы по созданию вычислительных машин носили в основном прикладной военный характер и как правило были строго засекречены. Поэтому приоритет в создании первой ЭВМ до сих пор объективно не установлен и остается своеобразной загадкой истории. Традиционно в определение величин авторских вкладов и приоритетов постоянно вмешивались различные суды США, признавая те или иные авторские права.

В СССР независимо от США и других стран над созданием как цифровых, так и аналоговых вычислительных машин (АВМ) работали различные коллективы ученых. Огромный вклад в создание лучшей цифровой ЭВМ своего времени внесен Сергеем Алексеевичем Лебедевым (1902-1974), под руководством которого в 1948 году была создана Малая электронная счетная машина (МЭСМ), а впоследствии и Большая электронная счетная машина (БЭСМ), ставшие родоначальниками соответствующих серий современных ЭВМ. Кроме этого С. А. Лебедевым, независимо от фон Неймана, были разработаны более детальные и полные принципы (основы) построения ЭВМ.

Ученые СССР были также лидерами в создании аналоговых вычислительных машин (АВМ), обрабатывающих данные, представленные в виде непрерывного ряда значений. Например, в 1949 году были созданы отечественные АВМ, называемые «интеграторами постоянного тока», которые предназначались для решения линейных дифференциальных уравнений с постоянными и переменными коэффициентами. При этом эквивалентное (по сравнению с цифровыми машинами) быстродействие АВМ достигало десятков мегафлоп (миллионов операций с плавающей запятой в секунду), которым еще долгое время не обладали цифровые вычислительные машины.

В новейшей истории развития современных средств вычислительной техники можно условно выделить ряд последовательно сменяющих друг друга этапов, на которых создаваемые семейства или «поколения» вычислительных устройств имели схожую элементную базу, принципы построения, конструктивно-технологическое исполнение и другие характеристики.

Первое поколение ЭВМ (1945 – 1955) создавалось на базе вакуумных ламп, конденсаторов, резисторов, трансформаторов, реле и других элементов, являющихся самостоятельными функциональными устройствами, используемыми в то время в промышленности. Оперативная память выполнялась на базе различных индуктивных элементов (катушек с ферритовыми сердечниками). В качестве средств ввода-вывода данных первоначально использовалась приспособленная телеграфная аппаратура, а

впоследствии электромеханические устройства на перфокартах и перфолентах. ЭВМ первого поколения имели большие размеры и энергопотребление, в сочетании с низким быстродействием и надежностью, малым объемом оперативной памяти.

Второе поколение ЭВМ (1955 – 1965) создавалось на базе более компактных и экономных полупроводниковых элементов (транзисторов, диодов и т.п.), собранных в функциональные блоки посредством печатных плат. Для ввода-вывода и хранения данных стали использоваться различные магнитные носители (ленты, диски, барабаны), а вывод визуальной информации осуществлялся в пиксельной форме на электронно-лучевые дисплеи (мониторы). С учетом расширения производства и применения ЭВМ стали разрабатываться алгоритмические языки программирования. Соответственно, размеры и энергопотребление ЭВМ, существенно сократились, при возросших быстродействии, надежности и объеме оперативной памяти.

Третье поколение ЭВМ (1965 – 1980) создавалось на базе интегральных схем, выполненных на кристалле полупроводника и объединенных в функциональные узлы посредством многослойных печатных плат. В виде специальных интегральных схем стали выполняться микропроцессоры и элементы оперативной памяти. Магнитные носители информации (диски, ленточные кассеты), а также устройства ввода-вывода (принтеры, графопостроители, сканеры) стали более компактными и унифицированными. С учетом тенденции к бурному развитию и распространению ЭВМ стали создаваться универсальные операционные системы и языки программирования. Малые размеры и энергопотребление ЭВМ обусловили создание класса мини- и микроЭВМ.

Четвертое поколение ЭВМ (1980 – 2000) создавалось на базе больших (БИС) и сверхбольших (СБИС) интегральных схем, содержащих в своих корпусах на порядки больше полупроводниковых элементов, чем ранее, что позволило значительно сократить размеры и удешевить производство, как элементной базы, так и самих ЭВМ. Вследствие этого появились доступные и ориентированные на бытовое использование персональные ЭВМ, работающие под управлением общедоступных легко тиражируемых операционных систем и ориентированного на интересы пользователей различного по функционалу программного обеспечения. Персональные ЭВМ стали использоваться как средства связи, коммуникации и развлечения.

Пятое поколение ЭВМ (2000 – н/время) характеризуется дальнейшим совершенствованием технологии производства микропроцессоров и других компонентов ЭВМ, в том числе постоянным увеличением количества примитивных элементов в единице объема кристалла полупроводника. Персональные ЭВМ становятся более компактными, универсальными и многозадачными, а интерфейс взаимодействия и пользовательское программное обеспечение – более красочными и интуитивно понятными. Появляется семейство планшетных ЭВМ, совмещающих в одном устройстве

вычислительную часть, средства связи, сетевого взаимодействия, записи изображений и монитор, одновременно используемый как устройство для ввода данных. Параллельно развивается семейство супер-ЭВМ, обладающих феноменальной производительностью.

1.2. Системы счисления. Способы представления чисел и выполнения арифметических операций в ЭВМ

Основанием позиционной системы счисления называется количество P возможных различных цифр (элементов), используемых для изображения числа. Значения цифр лежат в интервале от 0 до $P - 1$. В общем случае запись любого числа N в системе счисления с основанием P представляет собой многочлен вида:

$$N = a_{m-1} \times P^{m-1} + a_{m-2} \times P^{m-2} + \ldots + a_k \times P^k + \ldots a_1 \times P^1 + a_0 \times P^0 + \ldots + a_{-1} \times P^{-1} + \\ + a_{-2} \times P^{-2} + \ldots + a_s \times P^s \qquad (1.1)$$

Нижние индексы определяют местоположение цифры в числе (разряд): положительные значения индексов — для целой части числа (m разрядов); отрицательные значения — для дробной (s разрядов). Максимальное целое число, которое может быть представлено m разрядами: $N_{max} = P^m - 1$

Наименьшее значащее число, не равное 0, которое может быть представлено s разрядами дробной части: $N_{min} = P^{-s}$

Имея в целой части числа m разрядов, а в дробной s разрядов, можно записать P^{m+s} разных чисел.

С развитием средств автоматизации вычислений доминирующее место в технических системах заняла двоичная система счисления, имеющая основание $P = 2$ и использующая для представления чисел две цифры: 0 и 1.

Для перевода чисел из позиционной системы счисления с произвольным основанием в более понятный для человека десятичный вид используются процедуры перевода, основанные, в том числе, на выражении (1.1). Например, двоичное число 101110,101 равно десятичному числу 46,625:

$$101110{,}101_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + \\ + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 46{,}625_{10}$$

Кроме двоичной системы счисления в технических системах также используется шестнадцатеричная и двоично-десятичная системы счисления.

Шестнадцатеричная система счисления часто используется при программировании. Перевод чисел из шестнадцатеричной системы счисления в двоичную выполняется поразрядно. Для изображения цифр, больших 9, в шестнадцатеричной системе счисления применяются буквы: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Например, шестнадцатеричное число F17B в двоичной системе счисления имеет вид: 1111 0001 0111 1011, а в десятичной: 61819.

Двоично-десятичная система счисления в качестве основания системы счисления использует число 10, но каждая десятичная цифра (0, 1, …, 9) представляется (кодируется) четырьмя двоичными цифрами (разрядами, битами), называемыми тетрадами.

Очевидно, что помощью четырех битов можно закодировать шестнадцать цифр, поэтому «лишние» комбинации в двоично-десятичном коде являются запрещенными. Соответственно, недостатком такой системы счисления является ее избыточность (около 20 %), т.е. большее по сравнению с двоичной системой счисления количество двоичных разрядов, необходимых для представления одного и того же десятичного числа. Достоинством является легкость перевода в десятичную систему счисления и обратно.

В ЭВМ применяются две формы представления чисел [1-16]:

- естественная форма, или форма с фиксированной запятой (точкой) – ФЗ (ФТ);
- нормальная форма, или форма с плавающей запятой (точкой) – ПЗ (ПТ).

В форме представления с фиксированной запятой числа изображаются в виде последовательности цифр с постоянным для всех чисел положением запятой, отделяющей целую часть от дробной. Например, в числе отведено по пять разрядов для целой (до запятой) и дробной (после запятой) частей числа. Тогда десятичные числа, записанные в такую разрядную сетку, будут иметь вид:

+00721,35500;

+00000,00328;

–10301,20260.

Такая форма записи числа наиболее проста, но имеет небольшой диапазон представления значащих чисел N:

$$P^{-s} \le N \le P^{m} - P^{-s},$$

где: P – основание системы счисления;

m – количество разрядов в целой части числа;

s – количество разрядов в дробной части числа.

Например, при $P = 2$, $m = 10$ и $s = 6$ числа изменяются в диапазоне $0{,}015 < N < 1024$. Если в результате операции получится число, выходящее за допустимые пределы, произойдет переполнение разрядной сетки, и дальнейшие вычисления теряют смысл.

В современных ЭВМ форма представления чисел с фиксированной запятой используется как вспомогательная и только для целых чисел. В памяти ЭВМ числа с фиксированной запятой хранятся в трех форматах:

полуслово – это обычно 16 бит или 2 байта;

слово – 32 бита или 4 байта;

двойное слово – 64 бита или 8 байтов.

В ЭВМ отрицательные числа с фиксированной запятой записываются в разрядную сетку в виде так называемых дополнительных кодов.

В форме представления с плавающей запятой число изображается в виде двух групп цифр, называемых мантисса и порядок. При этом абсолютная величина мантиссы должна быть меньше 1, а порядок должен быть целым числом. В общем виде число в форме с плавающей запятой может быть представлено следующим образом:

$$N = \pm M \times P^{\pm r},$$

где: M – мантисса числа ($|M| < 1$);
r – порядок числа (целое число);
P – основание системы счисления.

Например, приведенные ранее числа запишутся следующим образом в форме с плавающей запятой:

$+0{,}721355 \times 10^{3}$;
$+0{,}328 \times 10^{-3}$;
$-0{,}103012026 \times 10^{5}$.

Форма представления чисел с плавающей запятой (нормальная форма) является основной в современных компьютерах. Диапазон значащих чисел N в системе счисления с основанием P при наличии m разрядов у мантиссы и s разрядов у порядка (без учета знаковых разрядов порядка и мантиссы) будет следующим:

$$P^{-m} \times P^{-(P^{1}-1)} \le N \le (1 - P^{-m}) \times P^{(P^{1}-1)}.$$

Например, при $P=2$, $m = 22$ и $s = 10$ диапазон представляемых десятичных чисел составляет от 10^{-300} до 10^{300}.

Следует заметить, что число в форме с плавающей запятой может быть записано бесконечным множеством различных чисел (вариантов его представления). При этом с изменением порядка запятая в числе смещается в право или влево (как бы «плавает»). Поэтому в целях единого представления числа с плавающей запятой хранятся в ЭВМ в нормализованном виде. Нормализованным называют такое число, в мантиссе которого до запятой стоит нуль (нули), а после запятой стоит отличная от нуля цифра. Это означает, что мантисса должна являться правильной дробью, т.е. должно выполняться условие $1/P \le |M| < 1$ (для нормализованных двоичных чисел $0{,}5 \le |M| < 1$).

Например, нормализованные, т. е. приведенные к правильной дроби числа:
$10{,}35_{10} = 0{,}1035_{10} \times 10^{2}$;
$0{,}00007245_{8} = 0{,}7245_{8} \times 8^{-4}$;
$F5C{,}9B_{16} = 0{,}F5C9B_{16} \times 16^{-3}$.

Алгебраическое представление двоичных чисел (т. е. их представления с учетом знака) осуществляется в ЭВМ посредством специальных кодов:

- прямой код двоичного числа;
- обратный код двоичного числа;
- дополнительный код двоичного числа.

Знак двоичного числа кодируется двоичной цифрой: код 0 означает знак «+» (плюс), а код 1 означает знак «–» (минус).

Прямой код двоичного числа $N = a_1, a_2, a_3, ..., a_m$ обозначим $[N]пр$.

При $N > 0$ $[N]пр = 0, a_1, a_2, a_3, ..., a_m$;

при $N < 0$ $[N]пр = 1, a_1, a_2, a_3, ..., a_m$;

при $N = 0$ имеет место неоднозначность: $[0]пр = 0, 0, ..., 0 = 1, 0, ..., 0$.

Обратный код двоичного числа $N = a_1, a_2, a_3, ..., a_m$ обозначим $[N]обр$.

При $N > 0$ $[N]обр = 0, a_1, a_2, a_3, ..., a_m$;

при $N < 0$ $[N]обр = 1, \bar{a}_1, \bar{a}_2, \bar{a}_3, ..., \bar{a}_m$,

где $\bar{a}$ обозначает инверсию а, т. е. если $a = 1$, тогда $\bar{a} = 0$, и наоборот.

При $N = 0$ имеет место неоднозначность: $[0]обр = 0, 0, ..., 0 = 1, 1, 1, ..., 1$.

Таким образом, чтобы получить обратный код отрицательного двоичного числа, необходимо все цифры этого числа инвертировать, т. е. в знаковом разряде поставить 1, а во всех значащих разрядах нули заменить единицами, а единицы – нулями. Например, для $N = +10101$ $[N]обр = 0,10101$; для $N = –10101$ $[N]обр = 1,01010$.

Дополнительный код двоичного числа $N = a_1, a_2, a_3, ..., a_m$ обозначим $[N]доп$.

При $N \geq 0$ $[N]доп = 0, a_1, a_2, a_3, ..., a_m$;

при $N \leq 0$ $[N]доп = 1, \bar{a}_1, \bar{a}_2, \bar{a}_3, ..., \bar{a}_m + 0,000...1$.

Таким образом, чтобы получить дополнительный код отрицательного двоичного числа, необходимо все его цифры инвертировать и затем к младшему разряду прибавить единицу. В случае возникновения переноса единицы из первого после запятой разряда в знаковый разряд, она складывается с единицей знакового разряда и «вытесняется» из разрядной сетки. Например, для $N = +10101$ $[N]доп = 0,10101$; для $N = –10101$ $[N]доп = 1,01011$; для $N = 0$ $[N]доп = \underline{1}0,00000 = 0,00000$ (т.к. первая 1 «вытесняется» из разрядной сетки). Соответственно, неоднозначности при представлении нулевого двоичного числа в дополнительном коде нет.

Эмпирическое правило для получения дополнительного кода отрицательного двоичного числа – все символы этого числа инвертировать, кроме последней (младшей) единицы и нулей, которые за ней следуют.

Если при сложении двоичных чисел в ЭВМ оба слагаемых имеют одинаковый знак, то операция сложения выполняется обычным способом. Если при сложении слагаемые имеют разные знаки, то сначала определяется большее по абсолютной величине число, из которого вычитается меньшее по абсолютной величине число, затем полученной разности присваивается знак большего числа.

Выполнение операций умножения и деления осуществляется в прямом коде обычным образом, но знак результата определяется по совпадению или несовпадению знаков участвовавших в операции чисел.

Обратный и дополнительные коды позволяют заменить в ЭВМ операцию вычитания на операцию сложения с отрицательным числом. При этом дополнительный код обеспечивает более быстрое выполнение операций, поэтому в ЭВМ он применяется чаще.

В «десятичной» интерпретации правила выполнения вычитания с использованием дополнительного кода следующие: чтобы число *B* вычесть из числа *A* необходимо сложить число *A* с «дополнением» к числу *B* и отбросить перенос в соседний старший разряд. «Дополнением» числа В является число, сложение с которым дает переполнение установленной разрядной сетки. Например, «дополнением» к трехзначному числу 515 является трехзначное число 485, поскольку 515 + 485 = 1000. В этом случае, чтобы вычесть число 515 из числа 743, необходимо сложить 743 с числом 485, и отбросить цифру, вышедшую за разрядную сетку: 743 + 485 = 1228, получим трехзначное число 228.

Арифметические операции над двоичными числами выполняются ЭВМ в арифметико-логических устройствах (АЛУ). Базовой операцией является сложение, которое реализуется электронной схемой сумматора. Вычитание заменяется операцией сложения, при котором вычитаемое (отрицательное число) представляется в дополнительном или обратном коде [12, 14, 15, 16].

Умножение выполняется как серия сложений со сдвигами (перемещение группы цифр на разряд вперед или назад). Деление выполняется как серия вычитаний со сдвигами. В каждый момент времени в операции участвуют только два числа. Такой принцип выполнения операций называется обычной двоичной машинной арифметикой, которая позволяет все арифметические операции свести к одному сложению, что существенно упрощает устройство процессора ЭВМ.

В случае, если числа в ЭВМ представляются в двоично-десятичном коде (каждый десятичный разряд кодируется четырьмя битами, называемыми тетрадой), арифметические операции выполняются по правилам двоично-десятичной арифметики.

Суммирование двоично-десятичных чисел можно выполнять по правилам обычной двоичной арифметики с обязательной последующей двоично-десятичной коррекцией в случае, если в какой-либо тетраде сформировалась запрещенная комбинация или произошло переполнение (перенос в старшую тетраду).

Двоично-десятичная коррекция заключается в дополнительном суммировании такой тетрады с числом 0110 (шесть – общее количество запрещенных комбинаций), например:

1) 18+13 = 31		2) 19+19 = 38	
Сложение тетрад	Коррекция	Сложение тетрад	Коррекция
0001 1000	0010 1011	0001 1001	0011 0010
+ 0001 0011	+ 0000 0110	+ 0001 1001	+ 0000 0110
0010 1011	0011 0001	0011 0010	0011 1000

Существуют также другие виды машинной арифметики. Например, «циклическая арифметика» и «арифметика насыщения» (в настоящем пособии не рассматриваются).

1.3. Элементы алгебры логики. Логические функции. Булевый базис. Аксиомы алгебры логики

Алгебра логики – раздел математики, изучающий высказывания, рассматриваемые со стороны их логических значений (истинности или ложности) и логические операции над ними.

Современные математические подходы к описанию логических высказываний и операций над ними впервые были определены английским математиком конца XIX века Джорджем Булем, в честь которого алгебру логики (алгебру высказываний) также называют булевой алгеброй.

Алгебра логики оперирует логическими высказываниями (логическими переменными), которые могут принимать только два значения, называемые ИСТИНА и ЛОЖЬ, TRUE и FALSE, ДА и НЕТ, которые для удобства и краткости обозначают 1 и 0.

Под логическим высказыванием понимают повествовательное предложение, относительно которого можно утверждать, истинно оно или ложно. Высказывания (логические переменные) принято обозначать буквами латинского алфавита (иногда с индексами): *A, B, C, X, Y, Z, a, b, c, x, y, (x_0, x_1, ..., x_{n-1})* и т. д.

Если высказывание *Z* истинно, это обозначается как *Z = 1* (*Z* = ИСТИНА). Если оно ложно, это обозначается как *Z = 0* (*Z* = ЛОЖЬ). Соответственно, *Z* может принимать единовременно только одно значение из двух возможных значений. Например, высказывание «Россия больше Франции» ИСТИННО, а высказывание «10 больше 15» ЛОЖНО.

В алгебре логики над высказываниями можно производить определенные логические операции, в результате которых получаются новые высказывания. Истинность результирующих высказываний зависит от истинности исходных высказываний и использованных для их преобразования логических операций.

Пусть имеется некоторый набор *n* высказываний *(x_0, x_1, ..., x_{n-1})*. Поскольку каждый элемент x_i может принимать только одно значение из двух значений 1 и

0 (т.е. $x_i = 1$ или $x_i = 0$), то общее количество всех возможных комбинаций значений элементов набора $(x_0, x_1, ..., x_{n-1})$ равно 2^n. Соответственно, при $n = 1$ возможны всего два набора (0) и (1), которые в алгебре логики выполняют функции двух единственных констант. При $n = 2$ возможны четыре набора (0, 0), (0, 1), (1, 0) и (1, 1).

Функцию F, определенную на наборе вида $(x_0, x_1, ..., x_{n-1})$ и принимающую значение 1 или 0 в зависимости от значений своих аргументов, называют функцией алгебры логики, логической функцией или булевой функцией.

Одной из форм задания (преставления, описания) логической функции является таблица (табличная форма), в которой указываются все возможные комбинации значений аргументов $(x_0, x_1, ..., x_{n-1})$ и соответствующие этим комбинациям значения функции F. Значения аргументов в наборах записываются, как правило, в порядке естественного возрастания двоичных последовательностей. Такая таблица состоит из 2^n строк и $n + 1$ столбцов.

В математике свойство некоторого набора функций выражать через себя любую сложную функцию называется свойством полноты этого набора. Такой полный набор называют базисом.

Набор трех логических функций: И (логическое умножение или конъюнкция), ИЛИ (логическое сложение или дизъюнкция) и НЕ (логическое отрицание или инверсия), называют булевым базисом (в честь Джорджа Буля), поскольку используя только эти три функции (путем суперпозиции) можно выразить любую сложную логическую функцию.

Конъюнкция или операция логического умножения обозначается знаками И, AND, ^, & или знаком умножения «×». Высказывание $A \wedge B$ истинно только в том случае, если истинны оба входящих в него высказывания.

Дизъюнкция или операцией логического сложения обозначается знаками ИЛИ, OR, 1, ∨ или знаком сложения «+». Высказывание $A \vee B$ истинно, если истинно хотя бы одно из входящих в него высказываний.

Таблицы задания (таблицы истинности) функций конъюнкции и дизъюнкции для двух аргументов приведены ниже.

Таблица 1.1 – Таблица истинности конъюнкции и логической суммы высказываний

Конъюнкция			Дизъюнкция		
A	*B*	*A*^B	*A*	*B*	*A*∨*B*
0	*0*	*0*	*0*	*0*	*0*
0	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*1*	*0*	*1*
1	*1*	*1*	*1*	*1*	*1*

Инверсия или операцией отрицания обозначается знаками НЕ, NOT, $\bar{A}$. Если высказывание A истинно, то $\bar{A}$ ложно, и наоборот (табл. 1.2).

Таблица 1.2 – Таблица истинности операции инверсии (отрицания)

A	$\bar{A}$
0	*1*
1	*0*

Кроме булевого базиса в алгебре логики широко используются следующие функции, называемые элементарными логическими функциями.

Функция «Исключающее ИЛИ», называемая также «функцией сложения по модулю два» или «функцией неравнозначности» (обозначается знаками «XOR», «M2»).

Функция эквивалентности (равнозначности), обозначаемая как $A \sim B$, $A = B$, *A eqv B* (табл. 1.3).

Функция импликации или логического следования, обозначаемая как $A \rightarrow B$, *A* IMP *B*, «Если *A*, то *B*» (табл. 1.3).

Таблица 1.3 – Таблица истинности операций эквивалентности и импликации

Эквивалентность			Импликация		
A	B	$A \sim B$	A	B	$A \rightarrow B$
0	*0*	*1*	*0*	*0*	*1*
0	*1*	*0*	*0*	*1*	*1*
1	*0*	*0*	*1*	*0*	*0*
1	*1*	*1*	*1*	*1*	*1*

Функция «И–НЕ» или функция Шеффера. Функция «ИЛИ–НЕ» или функция Вебба. Таблица истинности связок «И–НЕ», «ИЛИ–НЕ» приведена ниже (табл. 1.4).

Таблица 1.4 – Таблица истинности операций «И–НЕ», «ИЛИ–НЕ»

Инверсия x И y			Инверсия x ИЛИ y		
x	y	$\overline{x \wedge y}$	x	y	$\overline{x \vee y}$
0	*0*	*1*	*0*	*0*	*1*
0	*1*	*1*	*0*	*1*	*0*
1	*0*	*1*	*1*	*0*	*0*
1	*1*	*0*	*1*	*1*	*0*

Исходя из определений дизъюнкции, конъюнкции и отрицания, устанавливаются свойства этих операций и взаимные распределительные законы (аксиомы). Например:

- коммутативность (перестановочность):
$A \wedge B = B \wedge A$,
$A \vee B = B \vee A$;

- закон идемпотентности:
$A \wedge A = A$, $A \vee A = A$;

- двойное отрицание:
$\overline{\overline{A}} = A$;

- сочетательные (ассоциативные) законы:
$A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C$,
$A \wedge (B \wedge C) = (A \wedge B) \wedge C = A \wedge B \wedge C$;

- распределительные (дистрибутивные) законы:
$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$,
$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$;

- поглощение:
$A \vee (A \wedge B) = A$,
$A \wedge (A \vee B) = A$;

- склеивание:
$(A \wedge B) \vee (\overline{A} \wedge B) = B$,
$(A \vee B) \wedge (\overline{A} \vee B) = B$;

- операция переменной с ее инверсией:
$A \wedge \overline{A} = 0$;
$A \vee \overline{A} = 1$;

- операция с константами (0 - false, 1 - true):
$A \wedge 1, A \vee 1 = 1$,
$A \wedge 0 = 0; A \vee 0 = A$;

- законы де Моргана:
$\overline{A \wedge B} = \overline{A} \vee \overline{B}$ (условно его можно назвать 1-й);

$\overline{A \vee B} = \overline{A} \vee \overline{B}$ (2-й) – описывает результаты отрицания переменных, связанных с операциями И, ИЛИ.

Таблица 1.5

Аргументы		Функции					
A	*B*	И	ИЛИ	И-НЕ	ИЛИ-НЕ	М2 (хог)	eqv
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Операции конъюнкции, дизъюнкции и отрицания распространяются на любое количество аргументов. Логические функции (высказывания), образованные с помощью нескольких операций конъюнкции, дизъюнкции и отрицания называются сложными логическими функциями (высказываниями). Например, сложная функция ($\overline{A} \wedge \overline{B}$) определена табл. 1.6.

Таблица 1.6 – Таблица истинности высказывания $\overline{A} \wedge \overline{B}$

A	*B*	$\overline{A}$	$\overline{B}$	$\overline{A} \wedge \overline{B}$
0	*0*	*1*	*1*	*1*
0	*1*	*1*	*0*	*0*
1	*0*	*0*	*1*	*0*
1	*1*	*0*	*0*	*0*

Если две логические функции $F_1\ (x_0, x_1, \ldots, x_{n-1})$ и $F_2\ (x_0, x_1, \ldots, x_{n-1})$ принимают одинаковые значения на всех возможных наборах аргументов (т.е. их таблицы истинности совпадают), то такие функции называют равносильными. Для обозначения равносильных логических функций (высказываний) используют знак « = ». Например, $F_1\ (x_0, x_1, \ldots, x_{n-1}) = F_2\ (x_0, x_1, \ldots, x_{n-1})$.

Рассмотрим логическую функцию, заданную табл. 1.7.

Таблица 1.7 – Таблица истинности выражения $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$

A	$\overline{A}$	*B*	$\overline{B}$	$A \wedge B$	$\overline{A} \wedge \overline{B}$	$(A \wedge B) \vee (\overline{A} \wedge \overline{B})$
0	*1*	*0*	*1*	*0*	*1*	*1*
0	*1*	*1*	*0*	*0*	*0*	*0*
1	*0*	*0*	*1*	*0*	*0*	*0*
1	*0*	*1*	*0*	*1*	*0*	*1*

Если сравнить эту таблицу с таблицей истинности операции эквивалентности высказываний А и В (см. табл.1.3 и 1.5), то можно увидеть, что высказывания $(A \wedge B) \vee (\overline{A} \wedge \overline{B})$ и $A \sim B$ равносильны (тождественны), т.е. $(A \sim B) = (A \wedge B) \vee (\overline{A} \wedge \overline{B})$.

В алгебре логики можно проводить тождественные преобразования, заменяя одни высказывания равносильными им другими высказываниями. Например, в табл. 1.8 приведено доказательство дистрибутивного закона.

Таблица 1.8 – Доказательство истинности дистрибутивного закона

A	B	C	$B \vee C$	$A \wedge (B \vee C)$	$A \wedge B$	$A \wedge C$	$(A \wedge B) \vee (A \wedge C)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

1.4. Формы представления и способы минимизации логических функций

Одновременно с табличной формой задания (преставления, описания) логических функций используют более компактные формы представления логических функций посредством специальных аналитических выражений, называемых нормальными формами логических функций. Рассмотрим на примере способ перехода от табличной формы представления логической функции к нормальной форме. Пусть произвольная логическая функция *F* трех аргументов *a*, *b*, *c* задана следующей таблицей истинности (табл. 1.9):

Таблица 1.9 – Таблица истинности логической функции *F*

№	*Аргументы* *a b c*	*Функция* *F*	№	*Аргументы* *a b c*	*Функция* *F*
0	0 0 0	0	4	1 0 0	0
1	0 0 1	1	5	1 0 1	0
2	0 1 0	0	6	1 1 0	1
3	0 1 1	1	7	1 1 1	1

Из таблицы истинности следует, что функция $F = 1$ тогда, когда:
конъюнкция строки № 1 равна 1, т. е. $a\,\bar{b}\,c = 1$;
конъюнкция строки № 3 равна 1, т.е. $\bar{a}\,b\,c = 1$;
конъюнкция строка № 6 равна 1, т.е. $a\,b\,\bar{c} = 1$;
конъюнкция строка № 7 равна 1, т.е. $a\,b\,c = 1$.

Все это можно записать в виде одного общего аналитического выражения логической функции $F(a, b, c)$:

$$F = \bar{a}\bar{b}c \vee \bar{a}bc \vee ab\bar{c} \vee abc\,, \qquad (1.3)$$

которое действительно равно 1 только при любой из четырех перечисленных комбинаций значений аргументов. При всех других комбинациях значений все четыре его конъюнкции равны 0. Следовательно, функция, записанная в виде (1.3), эквивалентна функции, заданной табл. 1.9, причем выражена она только через функции И, ИЛИ, НЕ.

Полученное аналитическое выражение для функции F называют совершенной дизъюнктивной нормальной формой (СДНФ). СДНФ состоит из элементарных конъюнкций, для которых значение $F = 1$, соединенных знаками дизъюнкции. Конъюнкцию называют элементарной, если в нее не входят одинаковые аргументы, т.е. нет повторений одинаковых аргументов (букв). При этом в каждую элементарную конъюнкцию СДНФ должны обязательно входить все аргументы функции F, каждый из которых может быть представлен либо в прямой, либо в инверсной форме (если значение аргумента равно 0). Число таких элементарных конъюнкций в СДНФ, называемых также минтермами, обязательно равно числу единичных значений функции F в таблице истинности.

Описанная процедура получения СДНФ есть процедура перехода от табличной формы задания логической функции к ее аналитической форме. Указанные формы представления (задания) логической функции F являются взаимообратимыми, т.е. допускающими переход от одной формы задания логической функции к другой и обратно.

Для обратного перехода (от аналитической формы к табличной) достаточно подставить в аналитическую форму по очереди все комбинации значений аргументов и вычисленные значения функции записать в виде таблицы. В случае СДНФ эта процедура очень проста, поскольку каждая из входящих в СДНФ конъюнкций представляет всегда одну строку таблицы, для которой значение функции равно 1. Строка таблицы однозначно определяется по сочетанию инверсных и прямых аргументов: $\bar{a}\bar{b}c$ значит 001, $\bar{a}bc$ значит 011 и т.д.

Поскольку процедура построения СДНФ применима к таблице истинности с любым количеством аргументов и любым расположении единичных значений функции, то очевидно, что с помощью набора функций И, ИЛИ, НЕ (булевого

базиса) можно выразить любую логическую функцию, какой бы сложной она ни была.

Необходимо отметить, что по аналогии с совершенной дизъюнктивной нормальной формой (СДНФ) возможно представление логической функции в виде совершенной конъюнктивной нормальной формы (СКНФ), при которой сначала формируются элементарные дизъюнкции аргументов, при которых значение логической функции равно 0, а затем выполняются конъюнкции таких наборов аргументов. При этом, если значение аргумента в строке таблицы задания логической функции равно 1, то в элементарной дизъюнкции он записывается в инверсной форме. Таким образом СКНФ состоит из элементарных дизъюнкций, называемых также макстермами, соединенных знаками конъюнкции.

Минимизация булевых функций

Запись произвольной логической функции в форме СДНФ не единственно возможная и, как правило, не самая короткая (экономичная) и, как следствие, поддается минимизации путем сокращения (с учетом аксиом алгебры логики) количества входящих в ее состав аргументов (членов).

Например, осуществляется группировка аргументов, которая позволяет после вынесения в соответствии с (1.5) общих конъюнктивных членов получить в скобках выражения вида $(a \vee 1)$ или $(a \vee \bar{a})$, которые в соответствии с базовыми законами равны 1. Это сокращает число конъюнкций исходной формы и число входящих в них аргументов. Аналогично пытаются выделить из формулы и другие соотношения, полезные для ее упрощения:

$$abc = a(bc) = (ab)c;\ a \vee b \vee c = a \vee (b \vee c) = (a \vee b) \vee c; \quad (1.4)$$

$$ab \vee ac = a(b \vee c); \quad (1.5)$$

$$a \vee (bc) = (a \vee b)(a \vee c) \quad (1.6)$$

$$a \cdot 0 = 0;\quad a \cdot 1 = a;\quad a \cdot a = a;\quad a \cdot \bar{a} = 0; \quad (1.7)$$

$$a \vee 0 = a;\ a \vee 1 = 1;\ a \vee a = a;\ a \vee \bar{a} = 1; \quad (1.8)$$

где a – аргумент, который может принимать любое значение: 0 или 1.

Полезно знать и помнить некоторые значения элементарных функций с повторяющимися аргументами и с аргументами-константами:

$$x \vee x = x \qquad x \vee \bar{x} = 1 \qquad x \vee 1 = 1 \qquad x \vee 0 = x$$

$$x \wedge x = x \qquad x \wedge \bar{x} = 0 \qquad \overline{x \wedge 1} = x \qquad \overline{x \wedge 0} = 0$$

$$\overline{x \vee x} = \bar{x} \qquad \overline{x \vee \bar{x}} = 0 \qquad \overline{x \vee 1} = 0 \qquad \overline{x \vee 0} = \bar{x}$$

$$\overline{x \wedge x} = \bar{x} \qquad \overline{x \wedge \bar{x}} = 1 \qquad \overline{x \wedge 1} = \bar{x} \qquad \overline{x \wedge 0} = 1$$

Например, минимизировать выражение (1.3) можно вынеся за скобки общие конъюнктивные члены:

$$F = \bar{a}\bar{b}c \vee \bar{a}bc \vee ab\bar{c} \vee abc = \bar{a}c(\bar{b} \vee b) \vee ab(\bar{c} \vee c) = \bar{a}c \vee ab \qquad (1.9)$$

Для минимизации были использованы соотношения:

$$\bar{b} \vee b = 1; \quad \bar{a}c \cdot 1 = \bar{a}c; \; \bar{c} \vee c = 1; \; ab \cdot 1 = ab.$$

Полученное после минимизации выражение вида (1.9) существенно проще, чем СДНФ (1.3). Оно называется дизъюнктивной нормальной формой (ДНФ) и является дизъюнкцией меньшего количества элементарных конъюнкций, причем, как правило, количество входящих в конъюнкции аргументов (ранг каждой конъюнкции) также меньше.

При построении таблиц истинности по ДНФ нужно учитывать особенности обращения с теми элементарными конъюнкциями, в которые входят не все аргументы. Например, конъюнкция *ab* не содержит *c*, следовательно, *F* не зависит от *c*, т.е. если $a = b = 1$ ($ab = 1$), то $F = 1$ как при $c = 0$, так и при $c = 1$. Поэтому ситуация, когда элементарная конъюнкция $ab = 1$, должна быть зафиксирована сразу в двух строках таблицы: abc и abc.

Последовательно выполняя несколько этапов минимизации можно привести ДНФ к виду, когда дальнейшая минимизация станет невозможна. Такая ДНФ называется тупиковой ДНФ. Причем, выполняя этапы минимизации путем различной группировки и сокращения аргументов можно получить несколько несовпадающих тупиковых ДНФ одной и той же исходной логической функции, записанной в форме СДНФ. Тупиковая ДНФ, имеющая наименьшее по сравнению с остальными тупиковыми ДНФ общее количество аргументов (членов), входящих во все ее элементарные конъюнкции, называется минимальной ДНФ. Для исходной логической функции может быть несколько различных минимальных ДНФ. Таким образом, целью минимизации логической функции является нахождение минимальной ДНФ или одной из минимальных ДНФ, если их несколько.

Универсального правила группировки и сокращения аргументов, гарантированно приводящего любую СДНФ или ДНФ к ее минимальной форме, не существует. На практике используются различные методы минимизации, например, метод карт Карнау-Вейча (или карт Карно), основанный на зрительном анализе геометрического представления логической функции и другие [12, 16].

Минимизация логических функций используется при синтезе и оптимизации логических схем, что на практике позволяет создать более компактные (т.е. содержащие меньшее количество базовых элементов)

логические электрические схемы (схемные элементы и блоки ЭВМ). Это, безусловно, повышает надежность и быстродействие таких электрических схем и блоков, при одновременном снижении их габаритов, энергопотребления и стоимости.

Контрольные вопросы

1. Опишите основные этапы развития методов и средств вычислений.
2. Как классифицируются поколения (этапы) развития ЭВМ?
3. Какие виды систем счислений нашли применение в настоящее время?
4. Какие системы счислений используются в вычислительной технике?
5. Каким образом представляются числа в ЭВМ?
6. Какие коды используются при алгебраическом представлении чисел в ЭВМ?
7. Какие основные способы и правила выполнения арифметических операций с числами, представленными в двоичном коде?
8. Какие основные способы и правила выполнения арифметических операций с числами, представленными в двоично-десятичном коде?
9. Выполнить операцию сложения чисел А и В, представленных в двоичном коде:

A = 101100111010
B = 011010101100

Привести десятичные представления чисел при выполнении арифметических операций.

10. Выполнить операцию вычитания из числа А числа В, представленных в двоичном коде, двумя способами:

1) посредством заимствования единиц из старших разрядов;
2) посредством представления чисел в дополнительном двоичном коде.

A = 101100111010
B = 011010101100

Привести десятичные представления чисел при выполнении арифметических операций.

11. Выполнить операцию умножения числа А на число В, представленных в двоичном коде.

A = 101100111010
B = 011010101100

Привести десятичные представления чисел при выполнении арифметических операций.

12. Выполнить операцию деления числа А на число В, представленных в двоичном коде.

A = 101100111010
B = 011010101100

Привести десятичные представления чисел при выполнении арифметических операций.

13. Выполнить операцию сложения чисел A и B, представленных в двоично-десятичном коде.

A = 0011 0010 0001 0101 $_{2\text{-}10}$

B = 0101 0110 0011 0010 $_{2\text{-}10}$

Привести десятичные представления чисел при выполнении арифметических операций.

14. Выполнить операцию вычитания из числа A числа B, представленных в двоично-десятичном коде.

A = 0011 0010 0001 0101 $_{2\text{-}10}$

B = 0101 0110 0011 0010 $_{2\text{-}10}$

Привести десятичные представления чисел при выполнении арифметических операций.

15. Выполнить операцию умножения числа A на число B, представленных в двоично-десятичном коде.

A = 0010 0001 0101 $_{2\text{-}10}$

B = 0110 0011 0010 $_{2\text{-}10}$

Привести десятичные представления чисел при выполнении арифметических операций.

16. Что изучает алгебра логики (булева алгебра)?

17. Что называется логической функцией? Какие формы задания (представления) логических функций известны?

18. Что в алгебре логики называют базисом? Какие элементарные логические функции составляют базис (булевый базис)?

19. Какие основные аксиомы алгебры логики (булевой алгебры)?

20. Чем обусловлена необходимость минимизации сложных логических функций?

21. Что называется совершенной дизъюнктивной нормальной формой (СДНФ) и как она образуется (записывается)?

22. Что называется с совершенной конъюнктивной нормальной формой (СКНФ) и как она образуется (записывается)?

23. В чем заключаются основные способы минимизации логических функций?

24. Как количество различных элементов, схем и блоков в технических устройствах влияет на их надежность и быстродействие?

25. Как количество различных элементов, схем и блоков в технических устройствах влияет на их габариты, энергопотребление и стоимость?

26. В чем заключается практическая (производственная, экономическая) целесообразность минимизации логических функций?

ГЛАВА 2. ОСНОВНЫЕ ЭЛЕКТРИЧЕСКИЕ ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ И СХЕМЫ ЭВМ

2.1. Комбинационные и последовательностные элементы и схемы ЭВМ. Базовые электрические логический элементы и их простейшие связки

Обработка данных (электрических сигналов) в ЭВМ осуществляется различными по назначению электрическими (релейными, электронными и т.п.) логическими элементами и схемами, а также их соединениями и объединениями в более сложные узлы, блоки и устройства, выполняющими в конечном итоге определенную операцию, функцию, процедуру или процесс [1-16].

Далее для ясности под электрической схемой будем понимать некую совокупность гальванически связанных между собой более простых логических электрических элементов, т.е. «схема» является иерархически более сложной структурой по отношению к «элементу» и состоит из «элементов».

Электрические схемы также могут гальванически соединяться и образовывать таким образом иерархически более сложные структуры, называемые узлами, которые в свою очередь могут объединяться в различные функциональные блоки, из которых в конечном итоге строятся хорошо знакомые конечному потребителю (пользователю) различные аппаратные устройства ЭВМ и средства вычислительной техники, например, арифметико-логические устройства, микропроцессоры и системные блоки, устройства отображения и передачи информации, печатающие и запоминающие устройства и т.д.

Электрические логические элементы и схемы ЭВМ бывают двух видов: комбинационные и последовательностные.

Состояние выходов комбинационных элементов (схем) однозначно определяется состояниями их входов в данный момент времени.

Состояние выходов последовательностных элементов (схем) определяется не только состоянием их входов, но и внутренними состояниями самого элемента (схемы), имевшими место в предыдущие моменты времени. Поэтому такие элементы (схемы) часто называют элементами (схемами) с памятью.

Если при соединении (объединении) комбинационных элементов (схем) к ним добавляется хотя бы один последовательностный элемент (схема с памятью), то вся полученная таки образом схема (узел, блок) будет являться последовательностной. В качестве примера последовательностных схем можно привести триггеры, счетчики и регистры, принцип работы которых будет рассмотрен в следующих параграфах настоящего учебного пособия.

Комбинационные схемы являются электрическим аналогом булевых функций. Подобно тому, как сложная булева функция может быть получена суперпозицией более простых функций, так и сложная комбинационная схема может строиться из более простых комбинационных электрических элементов и схем, в основе которые лежат базовые электрические логические элементы «И», «ИЛИ», «НЕ», составляющие булевый базис.

Базовые электрические логические элементы «И», «ИЛИ», «НЕ», а также их простейшие комбинированные связки «И-НЕ», «ИЛИ-НЕ» и другие, имеют свое условное графическое обозначение, используемое в электрических схемах и выражающее их логическую функцию. При этом, как правило, абсолютно не имеет значения, какой именно принцип построения, производства и технологический процесс, а также какая именно электронная (совокупность полупроводников и их внутренних соединений) или иная схема использованы в самих логических элементах.

Посредством электрических комбинационных и последовательностных логических схем можно реализовать практически любой узел, блок, устройство или ЭВМ в целом.

Работу электрических логических элементов и схем (также как булевых функций) удобно описывать с помощью таблиц истинности или таблиц состояний.

Электрический логический элемент, реализующий логическую функцию «И» (конъюнкцию) двух или более логических переменных, называют кратко элемент «И» или конъюнктор. Условное графическое обозначение элемента «И» на электрических схемах приведено на рис. 2.1.

Согласно общепринятым правилам электрические логические элементы обозначаются прямоугольниками, на левой стороне которых указываются входы, а на правой стороне – выходы. Выполняемая электрическим логическим элементом логическая или иная операция изображается специальным символом, размещаемым, как правило, в верхней части графического условного обозначения элемента. В частности, операция конъюнкции обозначается знаком «&».

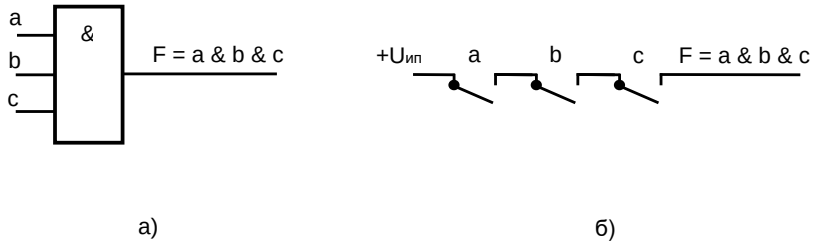


Рисунок 2.1 – 3-входовой конъюнктор:
а) условное обозначение на схемах,
б) реализация на переключателях (контактных группах)

Электрический логический элемент, реализующий логическую функцию «ИЛИ» (дизъюнкцию) двух или более логических переменных, называют соответственно элемент «ИЛИ» или дизъюнктор. Условное графическое обозначение элемента «ИЛИ» на электрических схемах приведено на рис. 2.2.

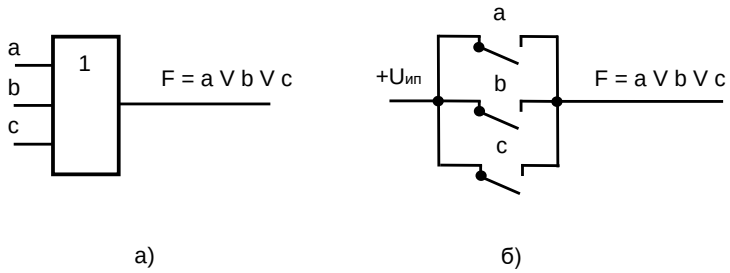


Рисунок 2.2 – 3-входовой дизъюнктор:
а) условное обозначение на схемах,
б) реализация на переключателях (контактных группах)

На рис. 2.1 показана реализация логической операции «И» двух переменных на контактах реле (переключателях). Замкнутый контакт реле соответствует значению логической переменной равному 1, разомкнутый контакт – значению логической переменной равному 0. Поскольку контакты реле соединены последовательно, то электрический сигнал (напряжение от источника питания), соответствующий значению 1, появится на выходе электрической цепи только в случае, если оба контакта замкнуты, т.е. соответствующие им логические переменные имеют значение 1.

Аналогичным образом на контактах реле (переключателя) можно реализовать логическую функцию «ИЛИ», если соединить их параллельно.

Электрический логический элемент, реализующий логическую функцию «НЕ» (операцию отрицания), называют соответственно элемент «НЕ» или инвертор. Условное графическое обозначение элемента «НЕ» на электрических схемах приведено на рис. 2.3.

Кроме электрических логических элементов «И», «ИЛИ», «НЕ», образующих булевый базис, на практике также широко используются их различные комбинированные связки (соединения), имеющие свои условные графические обозначения.

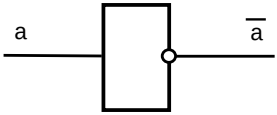


Рисунок 2.3 – Инвертор (элемент «НЕ»), обозначение на схемах

Например, последовательное соединение элемента «И» и элемента «НЕ», реализующее логическую операцию отрицания (инверсию) результата работы элемента «И», принято для компактности представлять (изображать) одним самостоятельным логическим электрическим элементом «И-НЕ», условное графическое обозначение которого приведено на рис. 2.4.

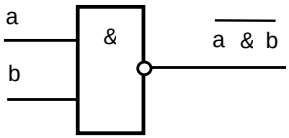


Рисунок 2.4 – Элемент «И-НЕ», обозначение на схемах

Аналогичным образом представляется на электрических схемах последовательное соединение «ИЛИ-НЕ», условное графическое обозначение которого приведено на рис. 2.5.

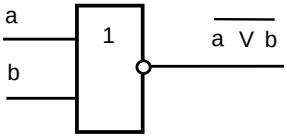


Рисунок 2.5 – Элемент «ИЛИ-НЕ», обозначение на схемах

2.2. Сумматор по модулю 2. Компараторы

Электрическую комбинационную логическую схему, представляющую собой соединение нескольких базовых электрических логических элементов «И», «ИЛИ», «НЕ» или их простейших комбинированных связок «И-НЕ», «ИЛИ-НЕ» и реализующую логическую функцию «Исключающее ИЛИ», называемую также «функцией сложения по модулю два» или «функцией неравнозначности» (обозначается знаками «XOR», «M2») двух или более логических переменных, принято называть на практике сумматором по модулю 2.

Построить сумматор по модулю 2 на элементах булева базиса можно различными способами. Однако, исходя из соображений компактности, на принципиальных электрических схемах его принято изображать одним самостоятельным электрическим логическим элементом, как это показано на рис. 2.6.

Электрическую комбинационную логическую схему реализующую логическую функцию «Эквивалентности», по сути сравнения или равнозначности, двух или более логических переменных, принято называть на практике компаратором. Простейшие компараторы формируют на выходе сигнал 1 при равенстве сравниваемых чисел или сигнал 0 при их неравенстве. Более сложные компараторы могут выполнять дополнительные операции, например, определять, какое из сравниваемых чисел больше (меньше) и т.п.

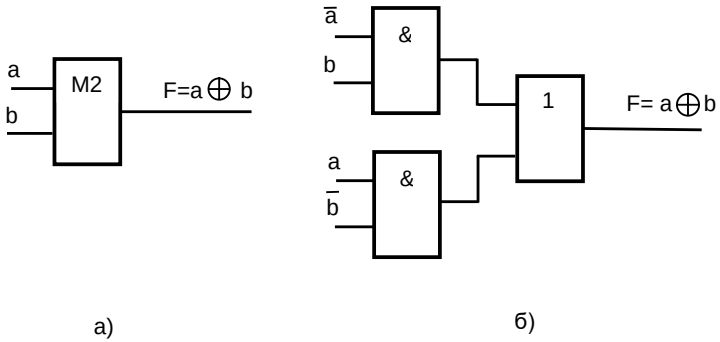


Рисунок 2.6 – Сумматор по модулю 2:
а) условное обозначение на схемах,
б) реализация на элементах булева базиса

Условное графическое обозначение (б) и пример реализации компаратора, осуществляющего сравнение двух 4-разрядных двоичных чисел, показаны на рис. 2.7.

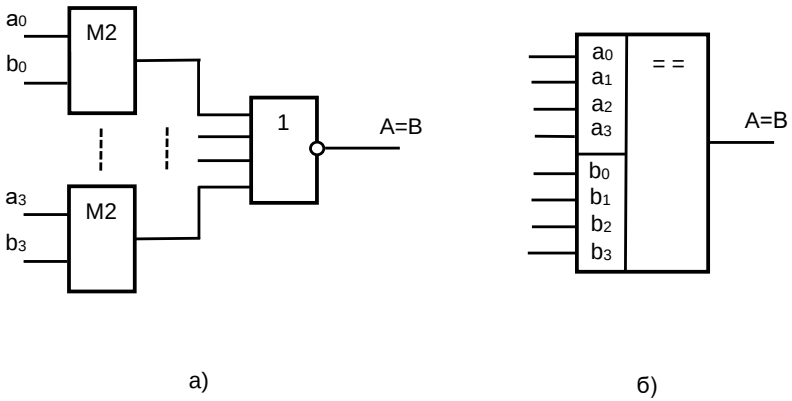


Рисунок 2.7 – Компаратор на равенство двух 4-разрядных двоичных чисел:
а) реализация на простейших логических элементах,
б) условное обозначение на электрических схемах

2.3. Кодеры. Декодеры. Шифраторы. Дешифраторы

2.3.1. Преобразователи произвольных кодов

Изменение по какому-либо известному закону вида цифровых данных, адекватное целям их дальнейшего использования, будем называть преобразованием цифровых данных, которое осуществляется в цифровой электронике различными специализированными преобразователями. Если такое преобразование представляет собой установление взаимно-однозначного соответствия между одним и другим кодом, имеющими один и тот же алфавит, то оно называется кодированием, а преобразователь его осуществляющий – кодирующим (декодирующим, если выполняется обратное преобразование) устройством или просто – кодером (декодером). Следует оговориться, что в современной теории информации выделяют восемь характерных видов

преобразования информации, в том числе различают такие виды ее преобразования как кодирование и шифрация [1]. Однако в целях настоящего учебного пособия такое углубленное рассмотрение теории не целесообразно.

Шифратором в цифровой электронике (промышленно выпускаемой элементной базе) принято называть (так сложилось исторически) электрическую комбинационную логическую схему, осуществляющую преобразование одного простейшего кода, например, одинарного (унитарного), поступающего на ее входы, в другой простейший код, например, двоичный, снимаемый с ее выходов. Такой шифратор будет иметь m входов и n выходов, связанных соотношением $m = 2^n$. При подаче активного сигнала, например, логической 1, на один из m его входов (обязательно только на один вход), на его n выходах отобразится двоичный код, соответствующий номеру активного входа. Условное обозначение на принципиальных электрических схемах шифратора вида «8-разрядный унарный код – 3-разрядный двоичный код» и пример его построения на элементах «4-входовой ИЛИ» показано на рис. 2.8.

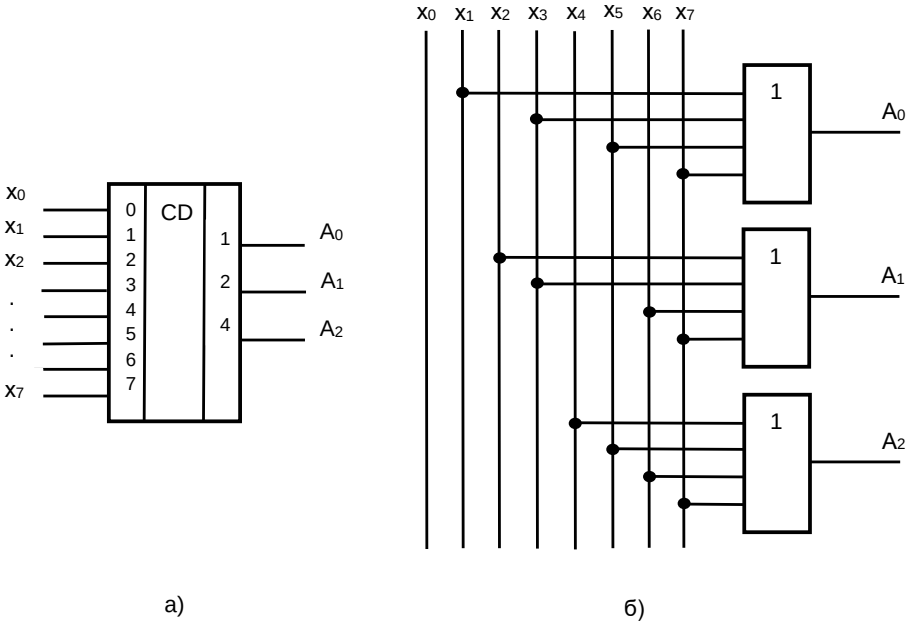


Рисунок 2.8 – Шифратора вида «8-разрядный одинарный код – 3-разрядный двоичный код»: а) условное обозначение на электрических схемах, б) пример построения на элементах «4-входовой ИЛИ»

Дешифратором принято называть электрическую комбинационную логическую схему, осуществляющую преобразование, обратное преобразованию шифратора. Дешифратор (обратный рассмотренному выше шифратору) будет иметь *n* входов и *m* выходов, связанных соотношением $n = log_2 m$. При подаче *n*-разрядного двоичного кода на его входы, на m его выходах отобразится соответствующий m-разрядный одинарный код. Условное обозначение на принципиальных электрических схемах дешифратора вида «3-разрядный двоичный код – 8-разрядный одинарный код» и пример его построения на элементах «НЕ» и «4-входовой И» показано на рис. 2.9.

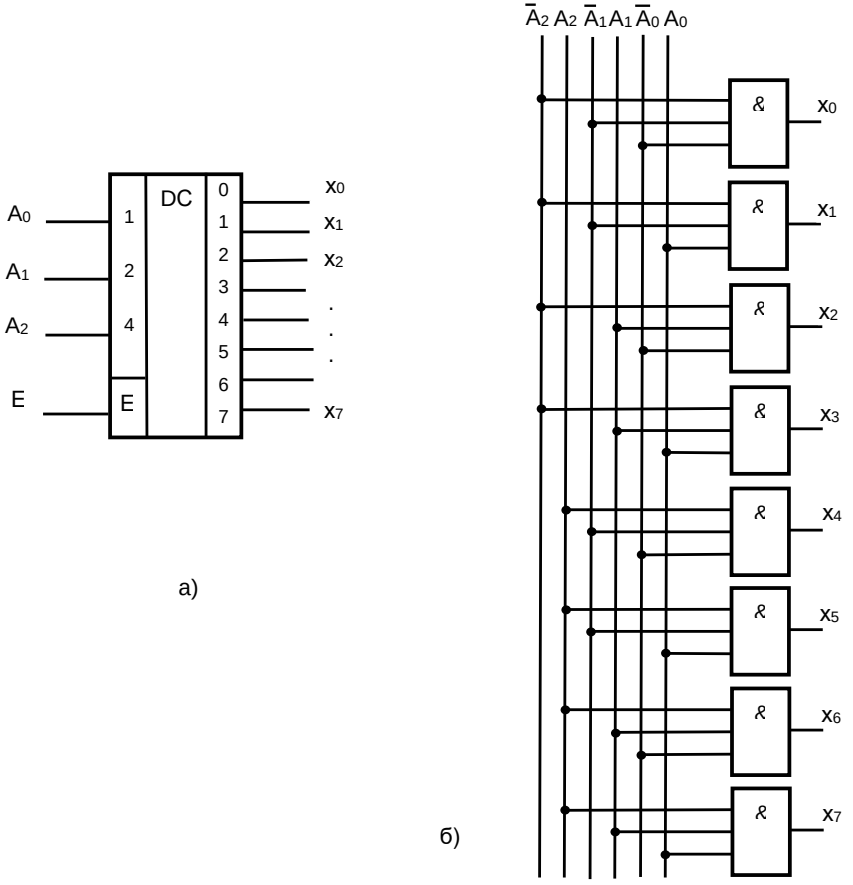


Рисунок 2.9 – Дешифратора вида «3-разрядный двоичный код – 8-разрядный одинарный код»: а) условное обозначение на электрических схемах, б) пример построения на элементах «НЕ» и «4-входовой И»

Разрешающий вход *E* служит для перевода дешифратора в активное (рабочее) или неактивное (ждущее, запертое) состояние. В нашем примере при подаче на вход *E* сигнала 1 он будет в активном состоянии, т.е. работать в режиме преобразования кода. В случае подачи на вход Е сигнала 0, независимо от того, присутствуют ли сигналы на входах дешифратора, все его выходы будут не активны, т.е. на каждом из них будет сигнал 0.

Аналогичным образом разрешающим входом *E* можно оснастить шифратор или иную другую логическую электрическую схему (узел, блок или устройство).

На практике часто возникает необходимость построения специализированных преобразователей (кодеров, декодеров), выполняющих преобразование входного кода А в выходной код Б по своему уникальному закону. Такие преобразователи принято называть преобразователями произвольных кодов.

В этом случае закон преобразования не описывается каким-либо регулярным правилом, как например, работа, рассмотренных выше шифратора и дешифратора, и единственным способом его описания (задания) является табличная форма (таблица преобразования). На практике часто возникает необходимость построения специализированных преобразователей (кодеров, декодеров), выполняющих преобразование входного кода А в выходной код Б по своему уникальному закону. Такие преобразователи принято называть преобразователями произвольных кодов. В этом случае закон преобразования не описывается каким-либо регулярным правилом, как например, работа, рассмотренных выше шифратора и дешифратора, и единственным способом его описания (задания) является табличная форма (таблица преобразования). В качестве примера подобного преобразователя рассмотрим устройство, управляющее работой трехцветного светофора по закону, заданному следующей таблицей преобразования:

Код А		Код Б		
А1	А2	З (Зел.)	Ж (Жел.)	К (Крас.)
0	0	1	0	0
0	1	0	0	1
1	0	0	1	1
1	1	0	0	0

Построить преобразователь кода, заданного указанной таблицей, можно несколькими способами, например, посредством синтеза необходимых логических элементов (системы булевых функций) или посредством использования связки «декодер – кодер» («дешифратор – шифратор»), как это показано на рис. 2.10 и рис. 2.11. Пример возможного условного обозначения преобразователя произвольного кода на принципиальных электрических схемах показан на рис. 2.10.

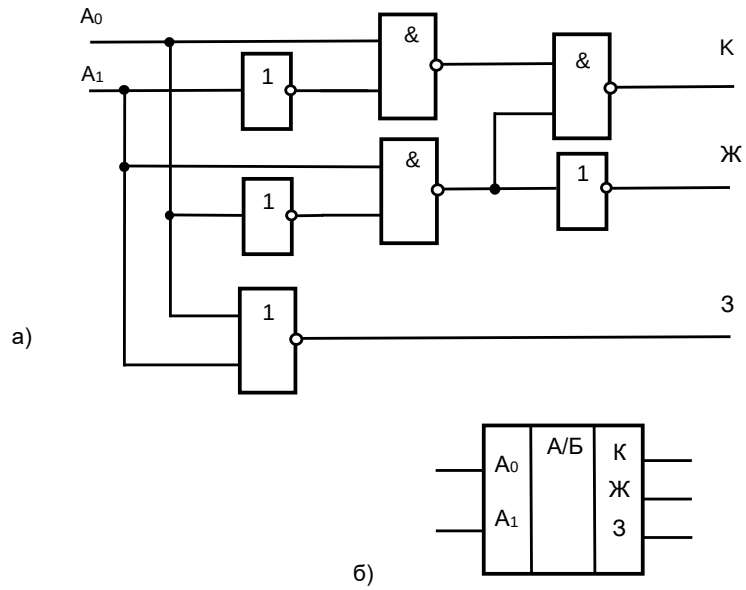


Рисунок 2.10 – Преобразователь кода для управления трехцветным светофором, построенный на отдельных логических элементах (а), условное обозначения преобразователя кода А в код Б по произвольному закону (б)

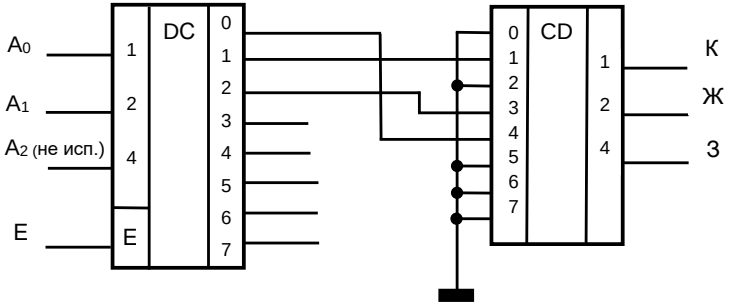


Рисунок 2.11 – Преобразователь кода для управления трехцветным светофором, реализованный посредством использования связки «дешифратор – шифратор»

2.4. Мультиплексоры. Демультиплексоры

Мультиплексором называется электрическая комбинационная логическая схема, осуществляющая подключение (коммутацию) одного из нескольких входов (шины) данных, номер (адрес) которого соответствует коду, поданному на адресные входы к единственному выходу (шине выхода). Если адрес кодируется двоичным кодом, то для коммутации m выходов необходимо $n = log_2 m$ адресных входов. По сути, это «многопозиционный переключатель», соединяющий единственный выход с одним из множества входов, направляя, таким образом, данные из нескольких «источников» к одному «приемнику». Мультиплексоры (в зависимости от специфики использования) часто называют также селекторами или коммутаторами. Условное обозначение на принципиальных электрических схемах 4-входового мультиплексора, который также называют мультиплексор 4–1, имеющего разрешающий вход E (при $E=1$ мультиплексор активен, при $E = 0$ – неактивен, заперт), и пример его реализации на логических электрических элементах, показаны на рис. 2.12.

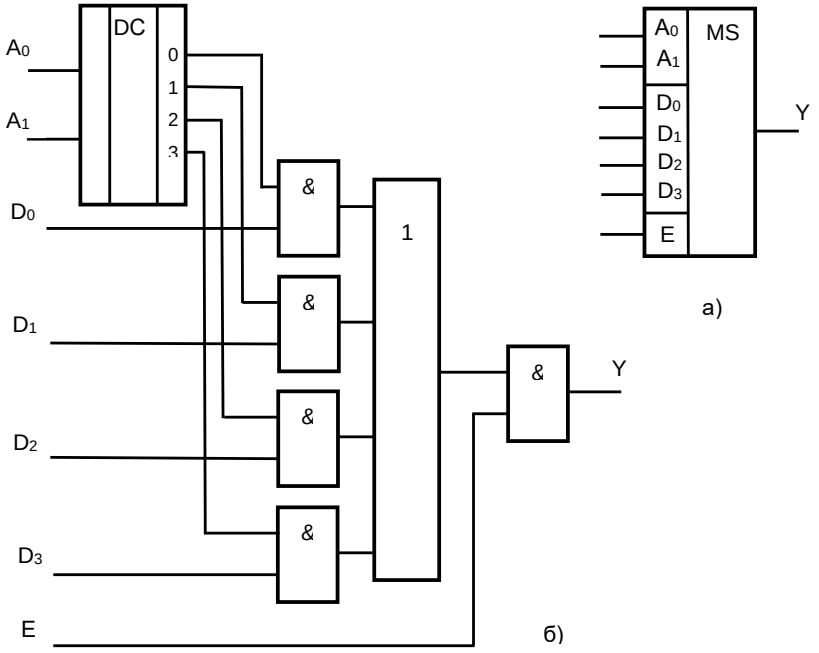


Рисунок 2.12 – Условное обозначение на принципиальных электрических схемах 4-входового мультиплексора 4–1, имеющего разрешающий вход E, (а) и пример его реализации на электрических логических элементах (б)

Демультиплексор осуществляет обратную по отношению к мультиплексору операцию, т.е. соединяет единственный вход с одним из множества выходов, распределяя, таким образом, данные из одного «источника» по нескольким «приемникам».

2.5. Триггеры

2.5.1. RS-триггер

Триггером называют электрическую последовательностную логическую схему с положительной обратной связью, имеющую два устойчивых состояния, которые называют единичным состоянием (обозначается как 1) и нулевым состоянием (обозначается как 0). Триггер может находиться в устойчивом состоянии сколь угодно долго, представляя собой по сути элементарную ячейку памяти.

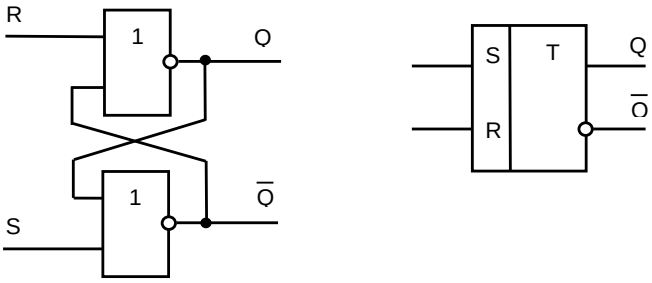
Триггер имеет необходимые входы управления (их количество и назначение зависят от типа триггера) и два выхода, один из которых называется прямым выходом (обозначается Q), а другой – инверсным выходом (обозначается $\bar{Q}$). Значение сигнала на инверсном выходе всегда противоположно значению сигнала на прямом выходе. Таким образом, если триггер находится в состоянии 1, то на выходе Q сигнал равен 1, а на выходе $\bar{Q}$ сигнал равен 0 и наоборот, т.е. выходы триггера взаимно инверсны.

Перевод триггера в единичное состояние посредством воздействия на соответствующий вход (обычно такой вход обозначают буквой S, от английского слова «Set») называют установкой триггера. Перевод триггера в нулевое состояние посредством воздействия на соответствующий вход (обычно такой вход обозначают буквой R, от английского слова «Reset») называют сбросом или гашением триггера.

Простейший RS-триггер можно построить из двух элементов ИЛИ-НЕ, как это показано на рис. 2.13 (а). Его условное обозначение на принципиальных электрических схемах показано на рис. 2.13 (б).

Если на обоих управляющих входах R и S уровень сигнала равен 0, то триггер находится каком-либо одном из устойчивых состояний, в которое он был установлен ранее. Например, путь триггер находится в состоянии 1. Тогда сигнал 1 с выхода Q по цепи обратной связи поступает на нижний элемент ИЛИ-НЕ триггера, обеспечивая удержание на инверсном выходе триггера $\bar{Q}$ сигнала 0, которых в свою очередь по цепи обратной связи, поступая на вход верхнего элемента ИЛИ-НЕ, поддерживает на выходе Q сигнал 1. В этом состоянии триггер может находиться сколь угодно долго. Аналогичным образом при уровне сигнала на управляющих входах R и S равным 0 и нахождении триггера в этом момент времени в состоянии 0 он будет и дальше за счет обратных связей

удерживаться сколь угодно долго в состоянии 0. Такой режим работы триггера (при $R = 0$ и $S = 0$) называют режимом хранения.



а) б)

Рисунок 2.13 – RS-триггер на двух элементах ИЛИ-НЕ (а) и его условное обозначение на принципиальных электрических схемах (б)

Указанный RS-триггер имеет следующую таблицу управления:

Уровни сигналов на управляющих входах	Состояние триггера
S = 0 R = 0	Режим хранения текущего состояния
S = 1 R = 0	Установка триггера в состояние 1
S = 0 R = 1	Установка триггера в состояние 0
S = 1 R = 1	Запрещенная комбинация

При подаче на вход *S* триггера сигнала 1 ($S = 1$), при условии одновременного наличия на входе *R* сигнала 0 ($R = 0$), он установится в состояние 1, если до этого момента времени находился в состоянии 0. Это произойдет по причине формирования на выходе нижнего элемента ИЛИ-НЕ триггера сигнала 0, который по цепи обратной связи обеспечит формирование на выходе верхнего элемента ИЛИ-НЕ триггера сигнала 1 (т.е. $Q = 1$). Если до подачи на вход *S* триггера сигнала 1 он уже находился в состоянии 1, то уровни

сигналов на выходах верхнего и нижнего элементов ИЛИ-НЕ не изменятся и триггер продолжит пребывать в состоянии 1.

Аналогичным образом триггер будет установлен в состояние 0 при подаче на его вход *R* сигнала 1 (*R = 1*) и одновременном наличии на входе *S* сигнала 0 (*S = 0*), если до этого момента времени он находился в состоянии 1. Если триггер уже находился в состоянии 0, то и далее продолжит пребывать в этом состоянии.

Следует отметить, что элементы ИЛИ-НЕ рассмотренного простейшего триггера изменяют свое состояние не одновременно, а последовательно, друг за другом. Это будет хорошо видно на временных диаграммах работы триггера. При этом в определенные, очень короткие, моменты времени на обоих выходах триггера может быть одинаковый уровень сигнала, что необходимо учитывать при построении различных схем и устройств с использованием триггеров, предусматривая необходимые временные задержки и компенсации для завершения переходных процессов в его элементах и установления на выходах триггера взаимно инверсных сигналов. Эта особенность характерна, впрочем, для любых электрических (электромагнитных, электронных и т.п.) логических и иных элементов, имеющих внутренние емкость, индуктивность и сопротивление, влияющих на скорость их переходных процессов.

По этой причине также запрещена одновременная подача на входы рассматриваемого простейшего триггера сигнала 1, т.е. комбинация *S = 1* и *R = 1* является запрещенной. Это недопустимо по двум причинам. Во-первых, в этом случае на обоих выходах триггера установятся нули, что противоречит определению и логике работы триггера, предусматривающего только два взаимно инверсных устойчивых состояния. Во-вторых, после одновременного изменения на входах *S* и *R* уровня сигнала на 0 оба внутренних элемента ИЛИ-НЕ начнут на перегонки переключаться в единичное состояние, стремясь удержать друг друга в нуле. Какой из этих элементов первым обеспечит на своем выходе сигнал 1 является непредсказуемым и будет зависеть от скоростей их внутренних переходных процессов и других физических факторов. Таким образом, состояние триггера является неуправляемым и неопределенным, что неприемлемо при создании цифровых схем и устройств. Кроме этого, спонтанное переключение триггера может возникать и вследствие воздействия на его входы или выходы различных внешних помех, вызывающих за счет обратных связей непредусмотренное и непредсказуемое изменение состояния его внутренних логических элементов и, следовательно, нарушение нормальной работы триггера. Указанные особенности обуславливают необходимость разработки более совершенных и устойчивых к помехам типов триггеров, например, синхронных триггеров, которые будут рассмотрены далее.

Синхронный триггер переключается в состояние, определенное сигналами на своих управляющих входах, только при подаче на специальный дополнительный *C*-вход сигнала синхронизации. Такой сигнал называют синхросигналом, синхроимпульсом, *C*-сигналом, *C*-импульсом, и т.п. Существуют синхронные триггеры, управляемые синхросигналом и синхронные

триггеры, управляемые синхроимпульсом. В отличие от синхросигнала, который является статической величиной, принимающей значение 0 или 1, синхроимпульс имеет ограниченную во времени длительность и триггер реагирует (в зависимости от принципов построения) на фронт или спад синхроимпульса.

При неактивном уровне синхросигнала (синхроимпульса) триггер не реагирует ни на какие управляющие воздействия на своих других входах, находясь в режиме ожидания (хранения текущего состояния). Таким образом, по сути, блокируются все входы триггера, кроме *C*-входа. При подаче на *C*-вход триггера активного уровня синхросигнала происходит разблокировка его управляющих входов, и триггер находится в режиме готовности к отработке управляющих воздействий, т.е. к переходу в состояние, определенное уровнями сигналов на его входах. Аналогичным образом (для соответствующего типа триггера), подача на *C*-вход триггера активного перепада синхроимпульса позволяет в этот короткий момент времени отработать комбинацию уровней сигнала на его управляющих входах. На рис. 2.14 приведена принципиальная схема простейшего синхронного RS-триггера с *C*-входом.

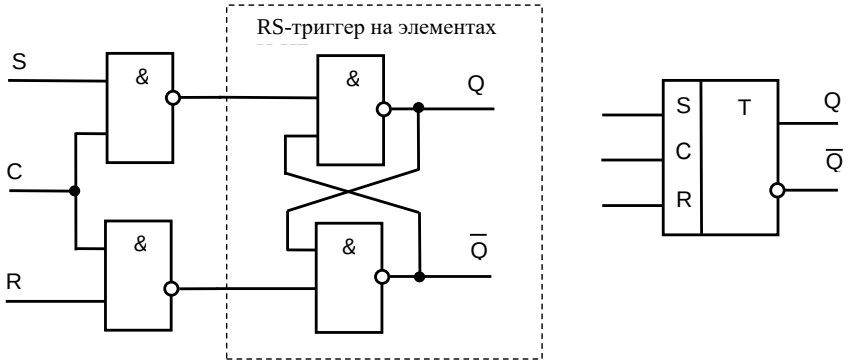
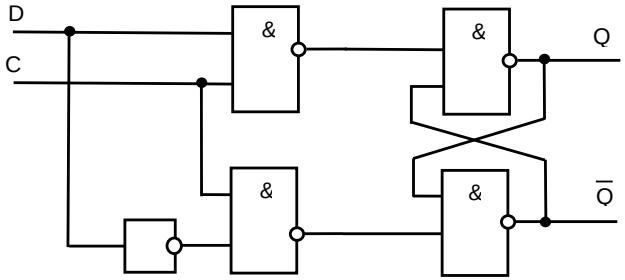


Рисунок 2.14 – Синхронный RS-триггер (а) и его условное обозначение на принципиальных электрических схемах (б)

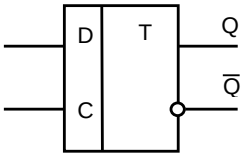
В отличие от RS-триггера, приведенного на рис. 2.13, триггер, выделенный пунктирной линией на рис. 2.14, построен на элементах И-НЕ и имеет следующую таблицу управления:

Уровни сигналов на управляющих входах	Состояние триггера
S = 1 R = 1	Режим хранения текущего состояния
S = 0 R = 1	Установка триггера в состояние 1
S = 1 R = 0	Установка триггера в состояние 0
S = 0 R = 0	Запрещенная комбинация

Как видно из указанной таблицы логика управления RS-триггера на элементах И-НЕ инверсна логике управления RS-триггера на элементах ИЛИ-НЕ. Однако синхронный триггер, приведенный на рис. 2.14, за счет верхнего и нижнего элементов И-НЕ в целом будет управляться как RS-триггер на элементах ИЛИ-НЕ. При уровне синхросигнала равном 0 (C = 0) на выходах входных элементов И-НЕ будет присутствовать 1, независимо от уровней сигнала на управляющих входах *S* и *R*. Таким образом внутренний RS-триггер, построенный на элементах И-НЕ, будет практически отключен от входов *S* и *R* и находиться в режиме хранения своего текущего состояния. При уровне синхросигнала равном 1 (*C = 1*) на выходах входных элементов И-НЕ будет уровень сигнала, инверсный по отношению к сигналам, присутствующим в данный момент времени на управляющих входах *S* и *R*, т.е. двухвходовые элементы И-НЕ становятся инверторами сигналов S и R. Например, при *S = 1* и *R = 0* на выходе верхнего элемента И-НЕ будет присутствовать логический 0, а на выходе нижнего элемента И-НЕ будет логическая 1, что обеспечить установку всего синхронного триггера в состояние 1 приведена на рис. 2.15.



а)



б)

Рисунок 2.15 – D-триггер (а) и его условное обозначение на принципиальных электрических схемах (б)

Как видно на приведенной принципиальной схеме для преобразования синхронного RS-триггера в D-триггер потребовалась минимальная доработка, заключающаяся в преобразовании посредством инвертора двух управляющих входов *S* и *R* в один вход *D*. Таким образом при подаче сигнала (0 или 1) на *D*-вход он преобразуется в два, взаимно инверсных сигнала, поступающий на верхний и нижний входные элементы И-НЕ, формируя (при условии *C = 1*), соответствующие управляющие сигналы для внутреннего RS-триггера, как это было показано при рассмотрении принципиальной схемы синхронного триггера, представленного на рис. 2.15. При этом необходимо помнить, что при уровне сигнала 0 на *C*-входе (т.е. при *C = 0*) D-триггер будет находиться в режиме хранения своего текущего состояния.

Рассмотренный простейший D-триггер в ряде источников также называют «прозрачной защелкой», поскольку любое изменения уровня сигнала на *D*-входе, пока на C-входе присутствует 1 (т.е. при *C = 1*), вызывает изменение текущего состояния D-триггера, что в силу возможного многократного изменения (искажения) сигнала на *D*-входе (в том числе в силу различных помех в линиях связи, схемотехнических элементах и соединяющих их проводниках) является существенным недостатком. Такой D-триггер («прозрачная защелка») зафиксирует свое состояние (конкретные уровни сигналов на выходах *Q* и $\overline{Q}$) только в момент изменения синхросигнала на *C*-входе от уровня 1 к уровню 0 (т.е. по спаду (срезу) синхросигнала) в момент времени, когда входные элементы И-НЕ «решат», что на их входы поступил 0 и переведут внутренний RS-триггер в режим хранения.

Недостатки, характерные для рассмотренных типов триггеров, устранены в так называемых двухступенчатых триггерах, которые кратко будут рассмотрены далее. Более подробные сведения о различных типах триггеров и способах их построения приведены в [1, 2, 4, 6, 11-16].

2.5.2. Двухступенчатые триггеры

Как следует из наименования, двухступенчатые триггеры состоят из двух последовательно включенных синхронных RS-триггеров, первый из которых предопределяет состояние второго. Поэтому первый RS-триггер принято называть М-триггером, ведущим или повелевающим (от слова «master» – господин, повелитель и т.п.), а следующий за ним второй RS-триггер соответственно называют S-триггером, ведомым, зависимым или рабским (от слова «slave» - рабский). Благодаря общему синхросигналу связка этих триггеров работает как единое целое и называется MS-триггером.

Принципиальная схема MS-триггера, построенного на базе рассмотренного ранее синхронного RS-триггера (два таких триггера включены друг за другом), приведена на рис. 2.16.

Показанный на рис. 2.16 MS-триггер «непрозрачен» по отношению к управляющим входам *S* и *R* ни при каком уровне сигнала на *C*-входе (т.е. ни при *C = 0*, ни при *C = 1*). Какая-либо одна из двух ступеней MS-триггера всегда оказывается отключенной от управляющих воздействий (т.е. запертой или синхросигналом *C,* или его инвертированным значением). Указанный MS-триггер может изменить свое текущее состояние, предписываемое управляющими входами *S* и *R*, только по спаду (срезу) *C*-сигнала.

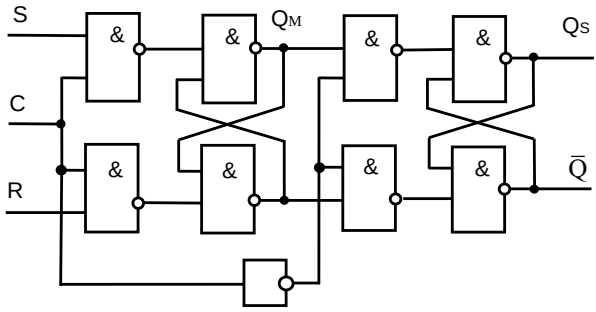


Рисунок 2.16 – MS-триггер, построенный из двух последовательно соединенных синхронных RS-триггеров

Если в рассмотренном MS-триггере добавить инвертирующие (заведенные накрест) обратные связи, идущие от выходов триггера к входным элементам И-

НЕ, то получится распространенный вид непрозрачного двухступенчатого триггера, получившего название JK-триггера, принципиальная схема которого приведена на рис. 2.17.

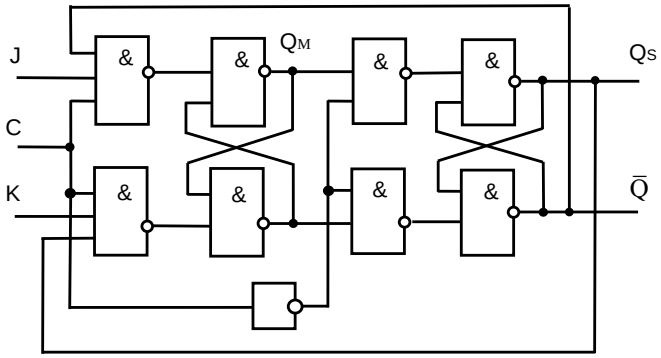


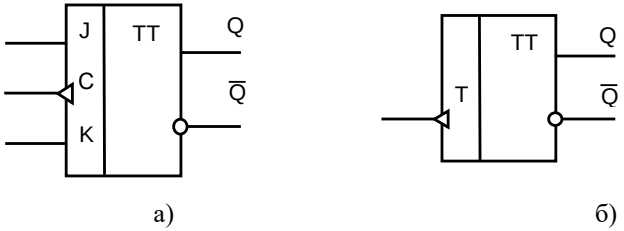
Рисунок 2.17 – JK-триггер, построенный на базе MS-триггера

Управляющие входы JK-триггера принято называть *J* и *K*, соответственно. При уровне сигнала 0 на входах *J* и *K* (*J = 0* и *K = 0*) входные элементы И-НЕ заперты и не реагируют на синхроимпульс *C*. При сочетании управляющих сигналов *J = 1* и *K = 0* триггер установится в состояние 1 по спаду (срезу) *C*-импульса (если до этого он находился в состоянии 0). В силу симметрии вход *K* обеспечивает установку JK-триггера в состояние 0. Таким образом, при инверсных уровнях сигнала на управляющих входах *J* и *K* триггер работает как синхронный непрозрачный RS-триггер.

В отличие от RS-триггера, у которого комбинация *S = 1* и *R = 1* является запрещенной, JK-триггер при уровне сигнала 1 на обоих управляющих входах (*J = 1* и *K = 1*) по спаду (срезу) каждого синхроимпульса *C* меняет свое состояние на противоположное. Такой режим работы JK-триггера называется счетным режимом или Т-режимом работы триггера.

Условное обозначение JK-триггера на принципиальных схемах показано на рис. 2.18 (а). Согласно общепринятому правилу, две буквы Т в обозначении триггера указывают на наличие в нем двух ступеней. Если на оба управляющих входа *J* и *K* постоянно подан сигнал 1 (т.е. всегда *J = 1* и *K = 1*), то JK-триггер превращается в так называемый Т-триггер или «счетный триггер» и его единственный синхровход в этом случае называется счетным входом или *T*-входом. Условное обозначение Т-триггера на принципиальных схемах показано на рис. 2.18 (б). Также часто говорят, что Т-триггер делит частоту

входных импульсов на два. Оба триггера, приведенные на рис. 2.18 переключаются по спаду (срезу) синхроимпульса *C*.



а) б)

Рисунок 2.18 – Условное обозначение JK-триггера (а) и Т-триггера (б) на принципиальных электрических схемах

2.6. Двоичные счетчики

Счетчиком называют электрическую последовательностную схему, осуществляющую счет (отображение числа) сигналов, поступающих на ее вход. Счет (отображение числа) поступающих сигналов осуществляется посредством последовательного изменения состояний выходов счетчика. После поступления очередного входного сигнала на каждом из выходов счетчика устанавливается логический 0 или 1, комбинация которых соответствует (согласно принятому для счетчика правилу кодирования) числу поступивших входных сигналов.

Общее количество всех возможных состояний выходов счетчика конечно и называется емкостью N счетчика или модулем пересчета (основанием пересчета). При этом одно из возможных состояний выходов счетчика, как правило, то при котором на всех выходах присутствует логический 0, принимается за начальное (нулевое) состояние счетчика. Очевидно, что счетчик всегда будет возвращаться в начальное (нулевое) состояние через каждые N входных сигналов.

На практике нашли применение различные типы счетчиков. По виду кодирования состояния выходов счетчика различают:

1. Двоичные счетчики – состояние выходов которых представляется двоичным кодом. Емкость двоичного счетчика $N = 2^n$, где n – разрядность двоичной кодовой комбинации (количество выходов счетчика).
2. Одинарные (унитарные) счетчики – состояние выходов которых представляется одинарным (унитарным) кодом (расположением единственной 1). Емкость одинарного счетчика $N = n$, где n – разрядность одинарной кодовой комбинации (количество выходов счетчика).

3. Унарные счетчики – состояние выходов которых представляется унарным кодом (энтропийное кодирование, определяемое количеством единиц к кодовой комбинации). Емкость унарного счетчика $N = n$, где n – максимальная разрядность унарной кодовой комбинации (количество выходов счетчика) или $N = n - 1$, если унарная кодовая комбинация обязательно должна замыкаться нулем (количество выходов счетчика минус 1).
4. Счетчики с более сложными видами кодирования.

   По порядку следования (перебора) состояний выходов счетчика (направлению счета) различают:

   1. Суммирующие счетчики – состояние выходов которых изменяется (перебирается) в возрастающем порядке.

   2. Вычитающие счетчики – состояние выходов которых изменяется (перебирается) в убывающем порядке.

   3. Реверсивные счетчики – состояние выходов которых может изменяться (перебираться) в возрастающем или убывающем порядке, в зависимости от выбора пользователя.

   По способу управления внутренними процессами (по сути – триггерами) счетчика различают:

   1. Асинхронные счетчики – для изменения состояния внутренних триггеров которых не требуется подавать на них специальный синхронизирующий сигнал (команду), а достаточно поступления на вход счетчика только входного (счетного) сигнала.

   2. Синхронные счетчики – для изменения состояния внутренних триггеров которых необходимо с поступлением входного (счетного) сигнала на вход счетчика подать также специальный синхронизирующий сигнал (команду).

   Условное обозначение 3-разрядного (см. выходы *Q0*, *Q1* и *Q2*) двоичного суммирующего (см. вход «+1», вычитающий счетчик имел бы обозначение «–1») асинхронного счетчика, имеющего также выход переноса *CR* (carry) и вход установки в нулевое состояние *R* (reset) показано на рис. 2.19.

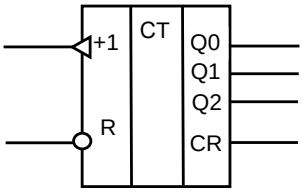


Рисунок 2.19 – Условное обозначение 3-разрядного суммирующего счетчика

Последовательность состояний 3-разрядного двоичного суммирующего счетчика приведена в табл. 2.1.

Таблица 2.1 – Последовательность состояний счетчика

Номер входного сигнала	Состояние выходов счетчика			
	Q2	Q1	Q0	CR
Установка в начальное «нулевое» состояние – сигнал «reset»	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0
8 (установка в «нулевое» состояние)	0	0	0	1
9	0	0	1	0
10	0	1	0	0
11	0	1	1	0
… и т.д.	…	…	…	…

Двоичные счетчики удобно строить на основе Т-триггеров, которые способны как хранить свое состояние, так и суммировать с ним по модулю 2 сигнал, поступающий на их вход. Для построения *n*-разрядного счетчика необходимо использовать *n* Т-триггеров. Пример построения простейшего 3-разрядного двоичного суммирующего асинхронного счетчика, в котором переключение составляющих его триггеров осуществляется по спаду (срезу) сигнала на их входах и происходит последовательно, друг за другом, т.е. после завершения перехода в соответствующее состояния предыдущего триггера, показан на рис. 2.20.

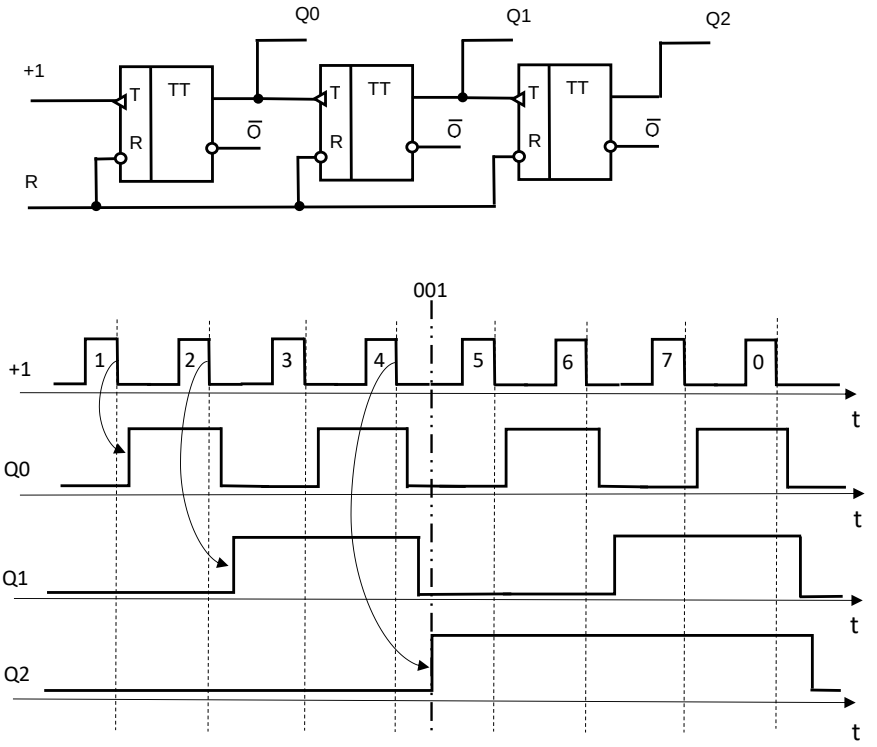


Рисунок 2.20 – Трехразрядный суммирующий счетчик с непосредственной связью и временная диаграмма его работы

Из представленной схемы видно, что входной сигнал для каждого последующего Т-триггера формируется на прямом выходе предыдущего Т-триггера. Очевидно, что при таком способе соединения триггеров они не могут изменить свое состояние одновременно. Поэтому считывание (определение) состояния такого счетчика, который принято называть счетчик с «непосредственной» связью триггеров, следует осуществлять после определенного промежутка времени, необходимого для завершения процессов переключения всех входящих в его состав Т-триггеров. Соответственно, чем больше разрядность счетчика, определяемая количеством входящих в последовательный тракт Т-триггеров, тем больше времени требуется счетчику для завершения всех переходных процессов и установки в текущее состояние,

соответствующее номеру поступившего на его вход сигнала. Эта особенность является очевидным недостатком таких счетчиков.

К достоинствам счетчиков с непосредственной связью, кроме простоты их построения, можно отнести легкость наращивания их разрядности, посредством последовательного подключения дополнительных триггеров.

2.7. Регистры. Регистровая память

Регистром называют последовательностную электронную схему, состоящую из n одноразрядных запоминающих элементов (триггеров), осуществляющую хранение n-разрядных двоичных последовательностей (чисел, кодов) и выполнение над ними простейших логических операций (арифметический, логический или циклический сдвиги вправо и/или влево, преобразование последовательного кода в параллельный и наоборот, инверсию хранимого числа (кода).

Логический и циклические сдвиги не имеют арифметического смысла. При логическом сдвиге высвобождающиеся разряды заполняются нулевыми значениями. При циклическом сдвиге (по аналогии с замкнутым кольцом) высвобождающиеся разряды заполняются значениями из вытесняемых разрядов.

Арифметические сдвиги эквивалентны определенным арифметическим операциям над числами, записанными в регистр. Например, арифметический сдвиг по дополнительному коду вправо эквивалентен умножению записанного числа на отрицательную степень двойки (2^{-m}, где m – целое число). Арифметический сдвиг по дополнительному коду влево эквивалентен умножению записанного числа на положительную степень двойки (2^{m}, где m – целое число). При их выполнении высвобождающиеся старшие разряды регистра заполняются значением знакового разряда, а младшие заполняются нулями.

По способу записи (ввода) двоичной последовательности в регистр, а также ее вывода (выдачи) на выходы (выход) регистра различают параллельные и последовательные регистры. Параллельные (по способу записи) n-разрядные регистры имеют n входов данных (разрядов) и осуществляют запись n-разрядной двоичной последовательности одновременно во все соответствующие разрядам n триггеров, т.е. за один такт (за одну команду на запись). Последовательные (по способу записи) n-разрядные регистры имею один вход данных, по которому потактно, т.е. один за другим, поступающие значения записываются в регистр, продвигаясь по цепочке из n триггеров. Таким образом, для записи n-разрядной двоичной последовательности необходимо n тактов (команд на запись).

Если вывод (передача во внешнюю среду) n-разрядной записанной двоичной последовательности осуществляется одновременно на n выходов регистра, т.е. параллельным кодом, то такой регистр принято считать параллельным по способу вывода данных. В противном случае, если выход данных у регистра один и записанная двоичная последовательность выводится

(выдается) посредством последовательного кода, то такой регистр принято считать последовательным по способу вывода данных. Также существуют универсальные (комбинированные) регистры, позволяющие осуществлять как параллельную, так и последовательную запись (вывод) двоичной последовательности.

На рис. 2.21 приведена принципиальная схема 4-разрядного параллельного регистра со следующими входами управления:

C-вход, разрешающий (при поступлении на него сигнала 1) запись во внутренние RS-триггеры регистра двоичной последовательности, поступающей на входы данных $X_1 – X_4$;

R-вход, устанавливающий (при подаче на него сигнала 1) все внутренние RS-триггеры регистра в нулевое состояние;

D-вход, обеспечивающий (при подаче на него сигнала 1) выдачу на выходы $Y_1 – Y_4$ регистра прямого кода записанной двоичной последовательности (числа);

$\overline{D}$-вход, обеспечивающий (при подаче на него сигнала 1) выдачу на выходы $Y_1 – Y_4$ регистра обратного кода записанной двоичной последовательности (числа).

При это необходимо учитывать, что для корректной выдачи записанной в регистр двоичной последовательности (числа) уровни сигналов на входах *D* и $\overline{D}$ должны быть взаимно инверсными, поскольку комбинация $D = \overline{D} = 1$ является запрещенной. В случае $D = \overline{D} = 0$ на всех выхода $Y_1 – Y_4$ регистра будут установлены нули, что является полезной опцией, например, когда необходимо отключить выходы $Y_1 – Y_4$ регистра от внешних шин.

Двоичная последовательность (число), поданная параллельным кодом на входы данных $X_1 – X_4$ будет записана во внутренние RS-триггеры регистра в момент подачи на управляющий *C*-вход сигнала 1, который переведет входные 2-входовые элементы И в режим обычных повторителей, по сути ретранслирующих сигналы, присутствующие на входах данных $X_1 – X_4$, на управляющие *S*-входы внутренних RS-триггеров. Согласно уровням сигналов на *S*-входа соответствующие RS-триггеры установятся в состояние 1 или останутся состоянии 0.

При изменении сигнала на управляющем *C*-входе с 1 на 0 произойдет отключение *S*-входов внутренних RS-триггеров от входов данных $X_1 – X_4$ и регистр перейдет в режим хранения и выдачи (в зависимости от уровней сигналов на других управляющих входах регистра) записанной двоичной последовательности (числа) на выходы $Y_1 – Y_4$. Следует отметить, что на практике управляющий *C*-сигнал, как правило, кратковременный (импульсный), поскольку это уменьшает вероятность занесения в регистр искаженной помехами (ошибочной) двоичной последовательности (числа).

Регистр представляет собой, по сути, простейшую *n*-разрядную ячейку многократно перезаписываемой, т.е. оперативной памяти. Если объединить в один узел (в одну сборку) сразу несколько однотипных регистров, то можно получить оперативное запоминающее устройство или, другими словами,

устройство регистровой памяти, общий объем которого будет определятся количеством входящих в него регистров, а разрядность шины данных – разрядностью базового регистра.

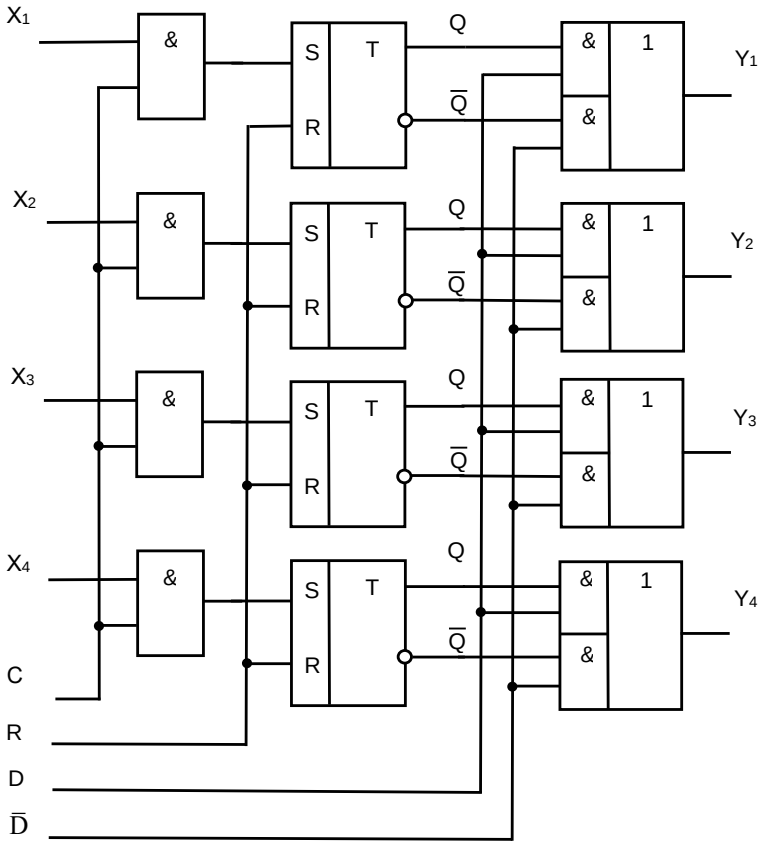


Рисунок 2.21 – Принципиальная схема 4-разрядного параллельного регистра на RS-триггерах, с возможностью вывода прямого и обратного кодов записанной двоичной последовательности (числа)

Способы построения таких узлов могут быть разными. Естественно, для обеспечения возможности выбора по адресу того или иного регистра необходимо использовать в схеме узла регистровой памяти соответствующие

дешифраторы, а для вывода в шину данных содержимого того или иного регистра логично использовать соответствующие мультиплексоры.

2.8. Программируемые логические матрицы. Постоянные запоминающие устройства (ПЗУ)

На практике широко используются преобразователи произвольных кодов, построенные по матричному принципу «декодер – кодер» на многовходовых элементах И и ИЛИ, имеющих так называемое выжигаемые (разрушаемые) перемычки на входах, выполненные из специальных материалов, позволяющие при их выборочном разрушении реализовывать однократно (необратимо) программируемый кодовый преобразователь, называемый программируемой логической матрицей (ПЛМ или programmable logic array – PLA). Способ построения простейшей программируемой логической матрицы приведен на рис. 2.22. Если входной декодер (на рис. 2.22 реализован на многовходовых элементах И) осуществляет преобразование n-разрядного двоичного кода в m-разрядный унитарный (одинарный) код, т.е. является полным дешифратором, имеющим n входов и m выходов, связанных соотношением $n = log_2 m$, то программируемая логическая матрица (ПЛМ) будет представлять собой постоянное запоминающее устройство (ПЗУ). На рис. 2.23 показан способ построения ПЗУ с выжигаемыми (разрушаемыми) перемычками на входах элементов ИЛИ (показаны звездочками).

Входной двоичный код $a_0 - a_{n-1}$, подаваемый на входы ПЗУ, называется адресом, 2^n вертикальных шин называются числовыми линейками, а все m выходов ПЗУ называются m-разрядной шиной данных $D_0 - D_{m-1}$, на которую выводятся соответствующие разряды хранимой двоичной последовательности (байта, слова и т.п.). При поступлении на вход ПЗУ двоичного кода будет активирована (выбрана) только одна числовая линейка. Если гальваническая связь этой числовой линейки и подключенных к ней входов элементов ИЛИ не разрушена, то на выходе таких элементов ИЛИ появится 1. Это означает, что в данном разряде хранимой двоичной последовательности записана 1. На выходах остальных элементов ИЛИ появятся нули, поскольку их связь с выбранной числовой линейкой была выжжена (разрушена) при программировании ПЗУ, т.е. при записи в ПЗУ по указанному адресу соответствующей двоичной последовательности. Таким образом, ПЗУ может хранить 2^n m-разрядных двоичных последовательностей (байт, слов и т.п.), которые однократной записываются в ПЗУ и при работе ПЗУ (со временем) не изменяются. На выходах ПЗУ (шине данных) появляется записанная двоичная последовательность, адрес которой был подан на входы ПЗУ. Как правило, ПЗУ имеют вход разрешения E, при подаче на который активного уровня сигнала ПЗУ выполняет свои функции, выводя в шину данных хранимую двоичную последовательность, а при

запрещающем сигнале – находится в режиме ожидания (на всех выходах ПЗУ появляются нули).

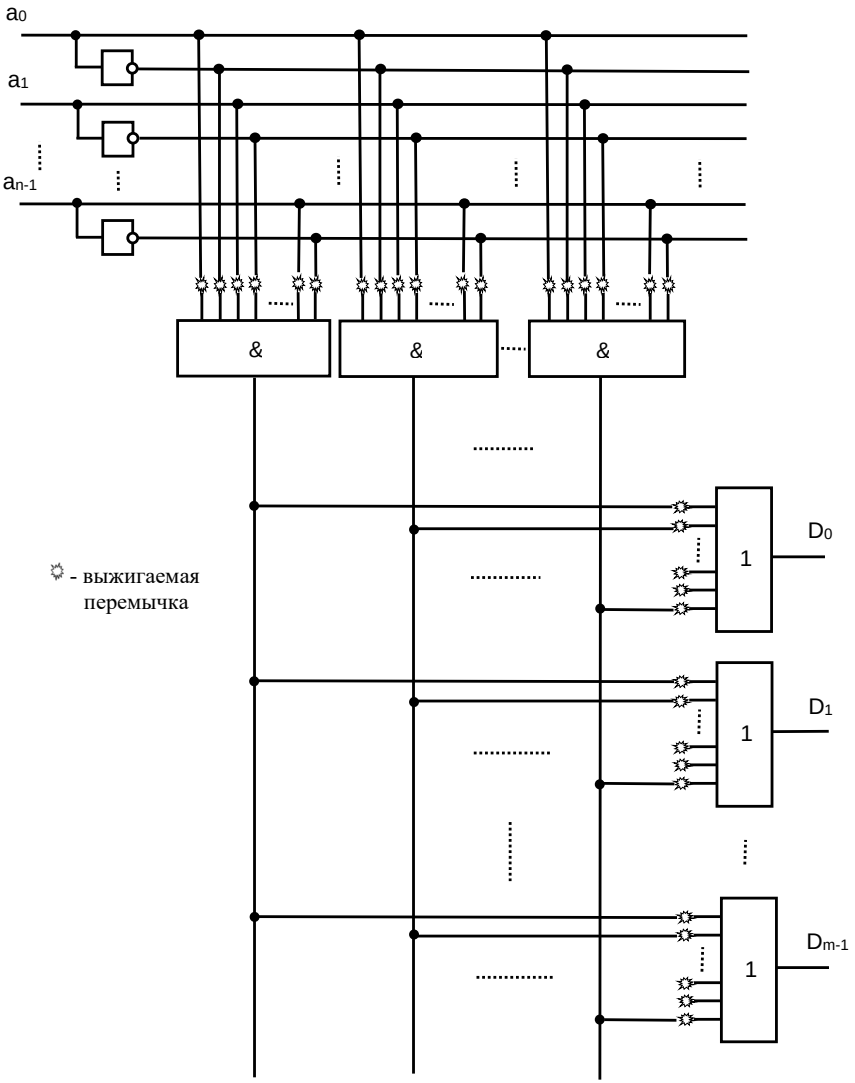


Рисунок 2.22 – Программируемая логическая матрица

На принципиальных электрических схемах ПЗУ принято обозначать аббревиатурой ROM (Read Only Memory – память, которая только считывается).

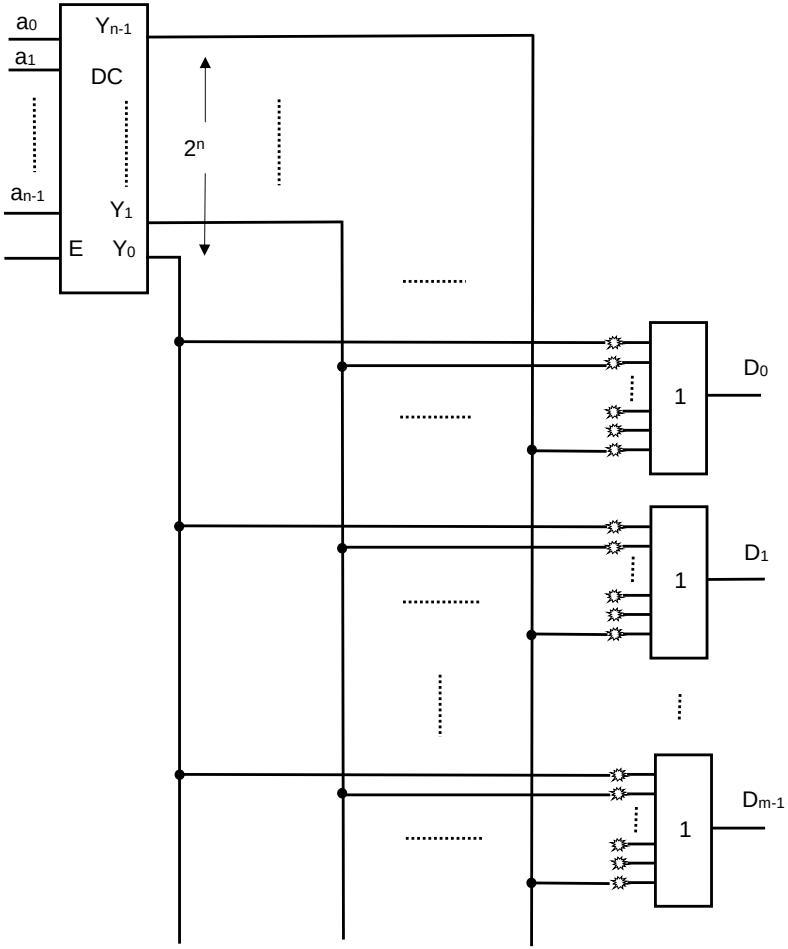


Рисунок 2.23 – Постоянное запоминающее устройство (ПЗУ)

2.9. Арифметико-логические устройства (АЛУ)

Арифметико-логическое устройство (АЛУ) – это совокупность электронных[1] схем (сумматоров, регистров, компараторов, мультиплексоров и т.п.), а также базовых электронных логических элементов, которые объединены единой логикой работы и управления для выполнения определенных арифметических и логических операций над многоразрядными входными данными, называемыми операндами. По суди операнд это элемент данных, над которым выполняется операция. В качестве операнда могут выступать число, символ, грамматическая конструкция и т.п. Например, в выражении $c = a + b$ операндами являются величины a и b.

В зависимости от типа и назначения АЛУ такими операциями, например, могут быть:

- операции двоичной арифметики для чисел с фиксированной запятой;
- операции двоичной (или шестнадцатеричной) арифметики для чисел с плавающей запятой;
- операции десятичной арифметики (над числами, представленными в двоично-десятичном коде);
- операции адресной арифметики (при модификации адресов команд);
- операции специальной арифметики;
- логические операции;
- операции над алфавитно-цифровыми полями.

Основными арифметическим операциям являются сложение и вычитание, а также умножение и деление, являющиеся производными операциями от сложения (вычитания) и реализуемые как последовательность простейших сложений (вычитаний) и сдвигов. Кроме этого, АЛУ могут выполняться специальные арифметические операции, например, арифметические сдвиги, эквивалентные определенным арифметическим операциям над числами. Основными логическими операциями являются конъюнкция (логическое умножение), дизъюнкция (логическое сложение), инверсия, эквивалентность (равнозначность), сложение по модулю два (исключающее ИЛИ) и другие.

Для выполнения перечисленных операций в состав АЛУ включаются следующие функциональные узлы:

- сумматор для выполнения соответствующей операции над многоразрядными операндами;
- регистры для записи, сдвига, инверсии и хранения операндов;
- базовые логические элементы и их комбинационные схемы для реализации различных логических операций и управления данными (операндами).

---

[1] Следует напомнить, что арифметико-логические устройства, как любые другие логические устройства и элементы, могут строятся на иной, например, релейной элементной базе.

АЛУ можно классифицировать по различным признакам. Например, по способу представления чисел различают АЛУ для чисел с фиксированной запятой, для чисел с плавающей запятой и для десятичных чисел.

По способу приема и обработки операндов различают параллельные и последовательные АЛУ. Соответственно, в параллельных АЛУ операнды представляются параллельным кодом и операции выполняются одновременно над всеми их разрядами. В последовательных АЛУ операнды представляются в последовательном коде, а операции выполняются последовательно (поразрядно) над парами соответствующих разрядов.

По способу построения различают блочные и универсальные (многофункциональные) АЛУ. В блочных АЛУ производные (сложные) арифметические операции над операндами выполняются в отдельных, специально предназначенных для этого, блоках, что позволяет увеличить скорость работы АЛУ, в том числе за счет организации параллельной работы таких блоков. К недостаткам такого подхода следует отнести существенно большее количество внутренних схемных элементов, необходимых для построения АЛУ, что является не экономным, увеличивает затраты на материалы и энергопотребление.

В универсальных (многофункциональных) АЛУ операции для различных форм представления чисел выполняются одними и теми же базовыми блоками, которые управляются (коммутируются) нужным образом в зависимости от выполняемой производной (сложной) операции.

Арифметико-логическое устройство (АЛУ) является главным функциональным узлом микропроцессора, осуществляющем непосредственное преобразование данных, и располагается на одном кристалле с его другими функциональными узлами.

АЛУ также выпускаются как самостоятельные функциональные интегральные схемы (микросхемы), широко используемые в различных технических системах и устройствах. Рассмотрим в качестве примера простейшее универсальное 4-разрядное АЛУ в интегральном исполнении, выпускаемое много лет российской промышленностью в комплектах ТТЛ-элементов (К155ИП), ТТЛШ-элементов (К531ИПЗП), ЭСЛ-элементов (К500ИП181) и КМДП-элементов (К564ИПЗ). Его условное обозначение на принципиальных схемах показано на рис. 2.24.

Указанные АЛУ построены на базе одноразрядных сумматоров, образующих при определенной коммутации 4-разрядный сумматор с параллельным переносом, что обеспечивает выполнение над 4-разрядными двоичными числами (операндами) A и B шестнадцати поразрядных логических и арифметико-логических операций, указанных в табл. 2.2.

Выбор той или иной операции осуществляется посредством подачи комбинации соответствующих сигналов на управляющие входы V_0, V_1, V_2 и V_3, а также на вход M. При $M = 0$ АЛУ выполняет арифметические операции, а при $M = 1$ выполняет логические операции.

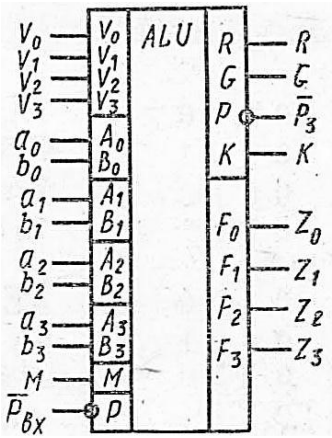


Рисунок 2.24 – Условное обозначение 4-разрядного АЛУ

Четырехразрядные двоичные числа (операнды) $a_3a_2a_1a_0$ и $b_3b_2b_1b_0$ подаются на входы данных $A_3 - A_0$ и $B_3 - B_0$ соответственно. Результат выполненной операции фиксируется на выходах $F_3 - F_0$, а результат сравнения отображается на выходе *K*.

Таблица 2.2 – Выполняемые логические и арифметико-логические операции

Выбор функции $V_3V_2V_1V_0$	Положительная логика		
	M = 1, логические операции	M = 0, арифметические операции	
		$\overline{P}_{вх} = 1$	$\overline{P}_{вх} = 0$
0000	$\overline{A}$	A	A + 1
0001	$\overline{AVC}$	A V B	A V B +1
0010	$\overline{A}B$	A V $\overline{B}$	A V $\overline{B}$ +1
0011	0	Минус 1 (дополнение до 2)	0
0100	$\overline{A\overline{B}}$	A + A$\overline{B}$	A + A$\overline{B}$ + 1
0101	$\overline{B}$	AB + A$\overline{B}$	AB + A$\overline{B}$ + 1
0110	$A \oplus B$	A – B – 1	A – B
0111	A$\overline{B}$	A$\overline{B}$ – 1	A$\overline{B}$
1000	$\overline{A}$ V B	A + AB	A + AB + 1
1001	$\overline{A \oplus B}$	A + B	A + B + 1

1010	B	A V $\overline{B}$ + AB	A V $\overline{B}$ + AB + 1
1011	AB	AB – 1	AB
1100	1	A + A*	A + A*+ 1
1101	A V $\overline{B}$	A V B + A	A V B + A + 1
1110	A V B	A V $\overline{B}$ + A	A V $\overline{B}$ + A + 1
1111	A	A – 1	A

Контрольные вопросы

1. Определение и назначение комбинационных и последовательностных элементов и схем ЭВМ.
2. Назначение и реализация на логических элементах двоичных сумматоров.
3. Назначение и реализация на логических элементах двоичных компараторов.
4. Назначение и реализация на логических элементах двоичных кодеров, декодеров, шифраторов, дешифраторов.
5. Назначение и реализация на логических элементах двоичных мультиплесоров и демультиплексоров.
6. Определение, назначение и реализация на логических элементах триггеров.
7. Виды (типы) триггеров и их реализация на логических элементах.
8. Определение и назначение счетчиков в вычислительных устройствах.
9. Классификация, основные параметры и построение двоичных счетчиков.
10. Принципиальные схемы двоичных счетчиков на основе различных типов триггеров.
11. Определение, назначение, классификация и реализация на логических элементах (схемах) двоичных регистров.
12. Перечислите основные операции, выполняемые регистрами. Назначение синхросигнала в работе регистров.
13. Назначение и особенности работы параллельных и последовательных регистров.
14. Организация регистровой памяти.
15. Как построить выходной каскад, обеспечивающий считывание информации из регистра в прямом и обратном кодах?
16. Назначение различных типов регистров в вычислительных устройствах.
17. Назначение и реализация на логических элементах программируемых логических матриц.
18. Назначение и реализация на логических элементах постоянных запоминающих устройств.
19. Назначение и принципы построения арифметико-логических устройств.
20. Приведите общую классификацию АЛУ и выполняемые ими операции.

ГЛАВА 3. ТИПОВАЯ СТРУКТУРА ЭВМ. ОСНОВНЫЕ КОМПОНЕНТЫ И ФУНКЦИОНАЛЬНЫЕ УСТРОЙСТВА

3.1. Принципы Лебедева-фон Неймана. Структурная схема и основные блоки ЭВМ

Общие принципы работы электронных вычислительных машин (ЭВМ) были сформулированы математиком Доном фон Нейманом (1903 – 1957) в сороковых годах прошлого века во время его работы совместно с другими американскими учеными и инженерами над первыми ЭВМ с программным управлением.

Принцип двоичного кодирования заключается в том, что команды и данные кодируются двоичным кодом, разделяемым на одинаковые элементы информации, называемые словами. Слова хранятся в идентифицируемых посредством адресов (номеров) ячейках памяти.

Принцип программного управления заключается в том, что процессор вычислительной машины выполняет программу, представляющую собой набор управляющих слов (команд), которые выполняются автоматически в заданной последовательности. Каждая команда задает определенную операцию для выполнения процессором. Команды хранятся в ячейках памяти вычислительной машины и выполняются либо в естественной последовательности их следования в программе, либо иным образом, что задается с помощью специальной команды условного или безусловного перехода.

Принцип однородности памяти заключается в том, что программы (команды) и данные хранятся в одной и той же памяти. При этом вычислительная машина не различает, что хранится в ячейке памяти – команда, символ или число, поэтому содержимое ячейки памяти может использоваться как данные, команда или адрес, в зависимости от способа обращения к нему. Над командами можно производить такие же действия, как и над данными, что позволяет, например, получать команды одной программы как результат исполнения другой программы, т.е. осуществлять трансляцию.

Принцип адресности заключается в том, что структурно основная память состоит из пронумерованных ячеек. Процессору в произвольный момент доступна любая ячейка, что позволяет давать имена областям памяти и обращаться в данным в памяти с использованием присвоенных имен.

Примерно в это же время в Союзе Советских Социалистических Республик (СССР) советским ученым академиком Лебедевым Сергеем Алексеевичем (1902 – 1974) независимо от фон Неймана (работы по созданию ЭВМ в то время во всех странах велись в закрытом режиме) были сформулированы свои более детальные и полные принципы построения и работы ЭВМ, которые кратко заключались в следующем:

- в состав ЭВМ должны входить взаимосвязанные устройства арифметико-логических вычислений (операций), памяти, ввода-вывода данных и устройства управления;

- кодирование данных и команд осуществляется дискретным образом в двоичной системе счисления;

- помимо арифметических операций вводятся и выполняются логические операции;

- программа вычислений представляет собой набор (последовательность) команд, каждая из которых задает (предписывает) определенную операцию;

- программа кодируется и хранится в памяти подобно данным (числам);

- вычисления должны выполняться автоматически на основе хранимой в памяти программы и операций над командами;

- память строится по иерархическому принципу;

- для вычислений используются численные методы решения задач.

Указанные принципы лежали в основе построения советских ЭВМ, которые были лучшими вычислительными машинами своего времени.

Таким образом ЭВМ являются системами с программным управлением, при котором непосредственным «инициатором» выполнения процессором очередной операции является команда [1-7,11,14-16]. Вначале из памяти извлекается команда, а затем по установленному (заданному) адресу извлекаются соответствующие данные или другая команда.

Согласно рассмотренным принципам Лебедева - фон Неймана можно упрощенно представить структуру ЭВМ, включающую следующие основные элементы:

1. Процессор (микропроцессор) – программно-управляемое устройство, представляющий собой, по сути, арифметико-логическое устройство (АЛУ), объединенное с необходимыми блоками управления, коммутации и внутренней оперативной памяти.

2. Память (запоминающее устройство) необходимого объема.

3. Устройства ввода/вывода информации (данных).

Упрощенная структурная схема ЭВМ, включающая указанные основные компоненты, приведена на рис. 3.1. На ней стрелками показано направление передачи данных. Процессор, кроме непосредственного выполнения арифметико-логических операций, управляет также работой других блоков (компонентов) ЭВМ.

Устройства ввода осуществляют преобразование внешних данных (информации), подлежащих обработке в ЭВМ, в адекватный для ЭВМ вид и непосредственную передачу (ввод) преобразованных данных по внутренним каналов передачи (шинам) в соответствующий блок ЭВМ. Наиболее известными современному потребителю устройствами ввода данных являются, например, клавиатура, различные манипуляторы (типа: «мышь», «джойстик», «трекбол», «точпад»), сенсорные панели (экраны), сканеры, графопостроители, видео-аудио преобразователи и т.п.

Устройства вывода осуществляют преобразование внутренних данных (информации), обрабатываемых в ЭВМ, в адекватный для восприятия человеком или другим техническим устройством вид. Наиболее известными современному потребителю устройствами вывода данных являются, например, монитор (экран, видеопанель), принтеры, графопостроители, звуковые источники (колонки, синтезаторы звука), различные промышленные и бытовые манипуляторы и т.п.

Рисунок 3.1 – Упрощенная структурная схема ЭВМ

Память (внутреннее запоминающее устройство) осуществляет хранение в течение необходимого времени исходных, промежуточных, результирующих и иных данных, обрабатываемых в ЭВМ. Ее размер, внутренняя организация и техническое исполнение определяются назначением и параметрами конкретной ЭВМ.

3.2. Структурная схема, принцип работы и классификация микропроцессоров

Микропроцессор это функционально законченное устройство, которое по назначению, функциям и составу по сути аналогично центральному процессору ЭВМ, но конструктивно выполнено в виде одной сверх большой интегральной микросхемы, имеющей малые геометрические размеры, высокую тактовую частоту работы и низкое энергопотребление за счет сверхкомпактной

реализации составляющих его полупроводниковых и иных элементов практически в одном объеме на одном или нескольких кристаллах (в настоящее время достигнут 7 нм технологический процесс его промышленного изготовления) и используемый в различных средствах вычислительной и иной техники для решения очень широкого круга разнотипных задач.

Микропроцессор выполняет действия над дискретными словами (двоичными данными), определенные соответствующей программой – упорядоченной последовательности команд, каждая из которых содержит информацию об одной элементарной операции. В простейшем случае команда это формализованное типовое предписание, представленное (записанное) на языке, понятном ЭВМ, и предписывающее (определяющее) действия машины при выполнении отдельной операции или какой-то части вычислительного процесса. Операцией принято называть нахождение некоторой величины в результате выполнения ЭВМ специального действия, предписанного командой программы, над одной или несколькими исходными величинами (данными). Операция могут быть арифметическими, логическими, программными, специальными и другими [8-10].

На практике большинство операций не могут быть выполнены за одно элементарное действие микропроцессора, что обуславливает необходимость выполнения внутри микропроцессора нескольких упорядоченных подготовительных действий, приводящих к требуемому результату, которые принято называть микрооперациями. Количество микроопераций зависит от вида конкретной команды. Время, в течение которого выполняется одна микрооперация, называется машинным циклом. При этом микрооперации также могут выполняться в несколько этапов, предусматривающих выполнение более простых – элементарных действий, которые производятся между внутренними устройствами (блоками) микропроцессора. Такие элементарные действия (одно или несколько одновременно) выполняются в течении очень короткого интервала времени, называемым машинным тактом, длительность которого задается специальным генератором тактовой частоты.

Учитывая изложенное, рассмотрим структурную схему микропроцессора, которая в упрощенном виде приведена на рис. 3.2. Микропроцессор состоит из трех основных блоков: блока обработки данных, устройства управления и блока обработки команд.

Блок обработки данных является внутренним вычислительным устройством микропроцессора, выполняющим все необходимые операции с поступающими и образовавшимися в ходе вычислений данными, в том числе прием данных, выполнение над ними заданной операции и передачу ее результата для хранения во внешнее оперативное запоминающее устройство (ОЗУ). Для непосредственного выполнения операций блок обработки данных содержит свое собственное сверхоперативное запоминающее устройство малого (по сравнению с внешним ОЗУ) объема, реализованное на базе нескольких программно доступных регистров, называемых регистрами общего назначения

(РОН), а также специальный регистр для хранения операндов и промежуточных результатов вычислений, называемый аккумулятором (АК). Регистры общего назначения (РОН) могут использоваться для хранения операндов, необходимых для выполнения операции (команды), промежуточных и иных данных, а также в ряде случаев выполнять строго определенные специальные функции. Соответственно, арифметико-логическое устройство (АЛУ) непосредственно выполняет все необходимые арифметические и логические действия над поступающими в него данными (операндами).

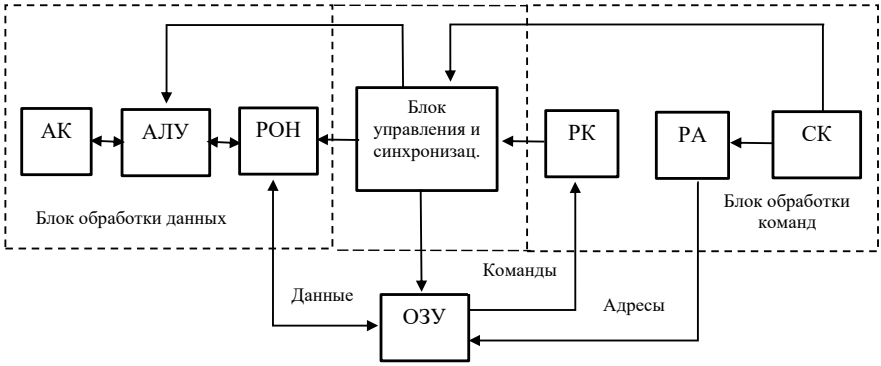


Рисунок 3.2 – Упрощенная структурная схема микропроцессора

Блок управления и синхронизации преобразует код операции в последовательность микроопераций для ее выполнения и формирует служебные коды и сигналы, управляющие работой всех блоков микропроцессора.

Блок обработки команд вырабатывает код адреса[2] очередной команды, который помещается в специальный регистр адреса (РА). По указанному адресу код команды извлекается из внешнего оперативного запоминающего устройства (ОЗУ) и помещается в регистр команд (РК), который передает его в блок управления и синхронизации (БУС). Для хранения кода адреса ячейки памяти, в

[2] Адрес – цифровое или буквенно-цифровое обозначение зоны запоминающего устройства или отдельной его ячейки, определяющее место хранения информации в памяти ЭВМ или другого технического устройства.

которой в свою очередь хранится код очередная выполняемой команды, служит специальных регистр, называемый счетчиком команд (СК).

В качестве исторического примера реального, технически устаревшего в настоящее время, но очень простого микропроцессора, приближенного к рассмотренной выше типовой структуре, приведем отечественный однокристальный микропроцессор КР580ВМ80А, который выпускался промышленно в СССР начиная с середины 70-х годов прошлого века и являлся аналогом американского микропроцессора фирмы Intel i8080 (1974 год изготовления). Он имел следующие основные характеристики:

1. Разрядность шины данных – 8 бит (1 Байт).

2. Максимальная тактовая частота – 2,5 МГц, что обеспечивало быстродействие до 625000 операций в секунду. Длительность такта при частоте 2,0 Мгц составляла 500 нс.

3. Разрядность шины адреса – 16 бит, что обеспечивало прямую адресацию внешней памяти объемом до 64 Кбайт (65536 байт) и подключение до 256 устройств ввода/вывода.

4. Технология изготовления – n-МДП[3] (6 мкм), что позволяло разместить на кристалле около 5000 транзисторов, а также обеспечить совместимость по уровням и мощности сигналов с другими микросхемами микропроцессорного комплекта серии КР580, часть из которых выполнялась по технологии ТТЛШ.

5. Система команд содержала 78 базовых команд или 244 кода, которые можно было разделить на пять групп: 1) команды передачи данных, 2) арифметические команды, 3) логические команды, 4) команды переходов, 5) команды управления, ввода/вывода и работы со стеком.

6. Потребляемая мощность около 750 мВт.

Входы и выходы микропроцессора были маломощными и соответствовали стандартным ТТЛ уровням. Шина данных и шина управления были совмещены. Конструктивно микропроцессор выпускался в прямоугольном корпусе с 40 выводами – по 20 выводов с каждой стороны (тип DIP). Внешний вид микропроцессора показан на рис. 3.3.

В состав блока регистров входили: 16-разрядный регистр адреса команды (IP), 16-разрядный регистр указателя стека (SP), 16-разрядный регистр временного хранения (WZ), 16-разрядная схема инкремента-декремента и шесть 8-разрядных регистров общего назначения (B, C, D, E, H, L), которые могли использоваться как три 16-разрядных регистра (BC, DE, HL).

Арифметико-логическое устройство микропроцессора было 8-разрядным и обеспечивало выполнение арифметических и логических операций над двоичными данными (числами), представленными в дополнительном коде, а также обработку двоично-десятичных упакованных чисел.

---

[3] МДП – технология «металл – диэлектрик – полупроводник», МОП – технология «металл – окисел – полупроводник», ТТЛШ – технология «транзисторно-транзисторная логика Шотки»

Наиболее простые команды выполнялись в КР580ВМ80А за один машинный цикл. На сложные команды требовалось 5 машинных циклов с восемнадцатью тактами. В начале каждого машинного цикла на шине данных микропроцессора формировалось 8 специальных сигналов, называемых байтом состояния. Байт состояния указывал, какой из машинных циклов выполняется в текущий момент, то есть к какому из внешних устройств происходит обращение.

Рисунок. 3.3 – Внешний вид микропроцессора КР580ВМ80А

Для индикации процесса передачи байта состояния служил выход синхронизации SYNC микропроцессора, на котором появлялось значение SYNC = 1 при выводе байта состояния. Сигнал SYNC = 1 позволял выделить байт состояния из информации, передаваемой по шине данных. Байт состояния появлялся на шине данных в течение SYNC = 1, записывался в специальный регистр слово состояния и далее использовался на протяжении всего машинного цикла. Запись производилась в момент комбинации сигналов SYNC = 1 и С2 = 1. Дешифратор преобразовывал байт состояния в требуемые для текущего машинного цикла системные управляющие сигналы. При формировании этих управляющих сигналов для согласования внутренних блоков микропроцессора по временным характеристикам использовались выходные сигналы DBIN и WR. Регистр слова состояния и дешифратор, обеспечивающие формирование системных управляющих сигналов, были названы системным контроллером.

Условное графическое обозначение микропроцессора КР580ВМ80А на принципиальных схемах и назначение его входов и выходов показано на рис. 3.4. Следует отметить, что также широко использовалось на практике обозначение входов и выходов микропроцессора посредством сокращений и аббревиатур русского алфавита.

Непосредственное назначение всех выводов микропроцессора КР580ВМ80А, включая выводы, необходимые для его питания, приведено в табл. 3.1.

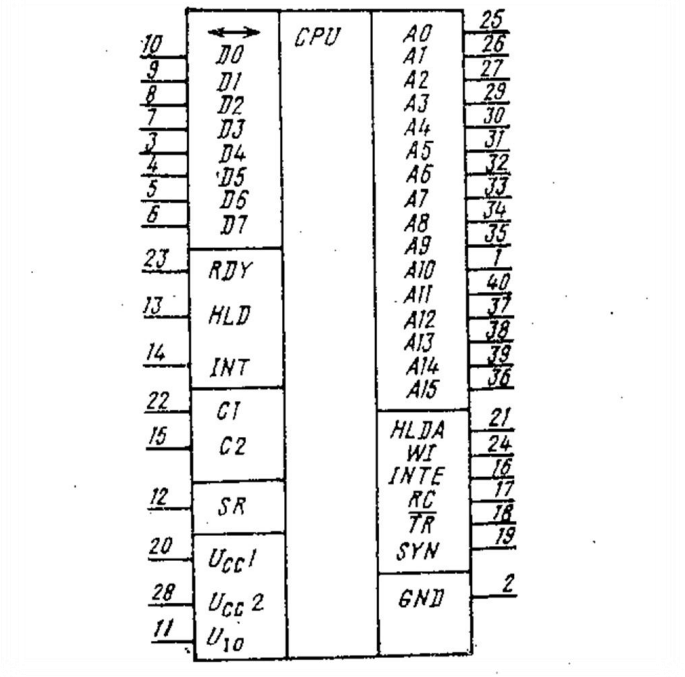


Рисунок 3.4 – Условное графическое обозначение микропроцессора КР580ВМ80А

Назначение выводов микропроцессора.

D0–D8 – двунаправленная 8-разрядная шина данных, которая выполняет передачу управляющего слова и обмен данными между регистрами микропроцессора и блоками.

A0–A15 – 16-разрядная шина адреса, которая выполняет передачу адреса соответствующей ячейки памяти при обращении к запоминающему устройству, а также передачу адреса внешнего устройства. В этом случае 8-разрядный адрес УВВ появляется на выводах *A0–A7* и дублируется на выводах *A8–A15*.

Таблица 3.1 – Назначение выводов микропроцессора КР580ВМ80А

Номер вывода	Назначение	Разряд	Обозначение	Тип
1	Шина адреса	10	A10	Выход
2	Общий	—	*GND*	—
3	Шина данных	4	Д4	Вход-Выход
4		5	Д5	
5		6	Д6	
6		7	Д7	
7		3	Д3	
8		2	Д2	
9		1	Д1	
10		0	Д0	
11	Питание 3 (—5 В)	—	$U_{CC}3$	—
12	Сброс	—	*R*	Вход
13	Захват	—	Зх	
14	Запрос прерывания	—	ЗПр	
15	Фаза 2	—	Ф2	
16	Разрешение прерывания	—	РПр	Выход
17	Прием	—	П	
18	Выдача	—	В	
19	Синхро	—	Сн	
20	Питание 2 (+5 В)	—	$U_{CC}2$	
21	Подтверждение захвата	—	ПЗХ	Выход
22	Фаза 1	—	Ф1	Вход
23	Готовность	—	Г	Вход
24	Ожидание	—	Жд	Выход
25	Шина адреса	0	A0	Выход
26		1	A1	
27		2	A2	
28	Питание 1 (+12 В)	—	$U_{CC}1$	—
29	Шина адреса	3	A3	Выход
30		4	A4	
31		5	A5	
32		6	A6	
33		7	A7	
34		8	A8	
35		9	A9	
36		15	A15	
37		12	A12	
38		13	A13	
39		14	A14	
40		11	A11	

Примечание. Шины данных и адреса трехстабильные; сигналы фаз Ф1 и Ф2 должны иметь амплитуду 12 В.

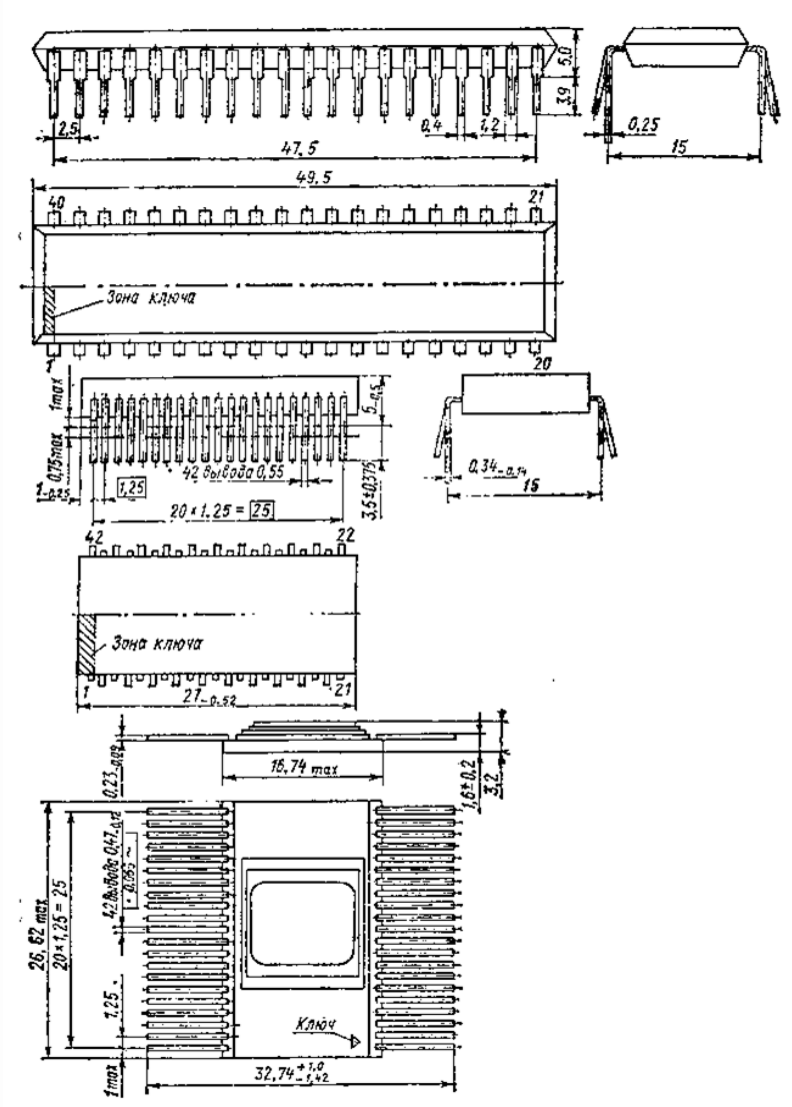


Рисунок 3.5 – Конструктивное исполнение корпусов микропроцессора

Сигналы управления шиной данных:

DBIN – выходной сигнал «Прием». При состоянии *DBIN = 1* шина данных настроена на прием данных в микропроцессор из памяти или УВВ. При *DBIN = 0* шина данных настроена на вывод информации из микропроцессора.

WR – выходной сигнал «Выдача данных». При состоянии *WR = 0* микропроцессор фиксировал на шине данных 8-разрядный код, который был предназначен для запоминающего устройства или УВВ.

Сигналы управления вводом-выводом:

RDY (READY) – входной сигнал «Готовность» от УВВ или запоминающего устройства. При состоянии *READY = 1* УВВ или память готовы к обмену данными с микропроцессором. При *READY = 0* УВВ или память не были готовы к обмену данными с микропроцессором. В этом случае микропроцессор переходил в режим «Ожидание».

WI (WAIT) – выходной сигнал «Ожидание». При состоянии *WAIT = 1* микропроцессор находился в режиме «Ожидание».

INT – входной сигнал «Запрос прерывания» от УВВ. При состоянии *INT = 1* одному из УВВ требуется обслуживание.

INTE – выходной сигнал «Разрешения прерывания», информирующий УВВ о возможности или невозможности обслуживания микропроцессором запросов на прерывание. При состоянии *INTE = 1* прерывания разрешены. При *INTE = 0* прерывания запрещены.

HLD (HOLD) – входной сигнал «Запрос захвата шин» от УВВ. При состоянии *HOLD = 1* одно из УВВ требует обмена по прямому доступу к памяти.

HLDA – выходной сигнал «Подтверждение захвата шин». При состоянии *HLDA = 1* микропроцессор отключался от системных шин, которые в этот момент времени могли использоваться УВВ или памятью.

Сигналы синхронизации:

C1 и *C2* – входные сигналы от тактового генератора.

SYNC – выходной сигнал «Синхронизация». При состоянии *SYNC = 1* на шину данных микропроцессора выдавались восемь управляющих сигналов.

SR (RESET) – входной сигнал «Сброс», осуществляющий начальную установку микропроцессора. При состоянии *RESET = 1*, длящемся 3 - 4 периода тактовой частоты, микропроцессор прекращал свою работу, обнулял счетчик команд и переходил в режим ожидания. При поступлении *RESET = 0* микропроцессор начинал выполнять команду, записанную по адресу 0000H.

Классифицировать микропроцессоры можно по разным признакам. На рис. 3.6 приведена одна из возможных условных классификаций.

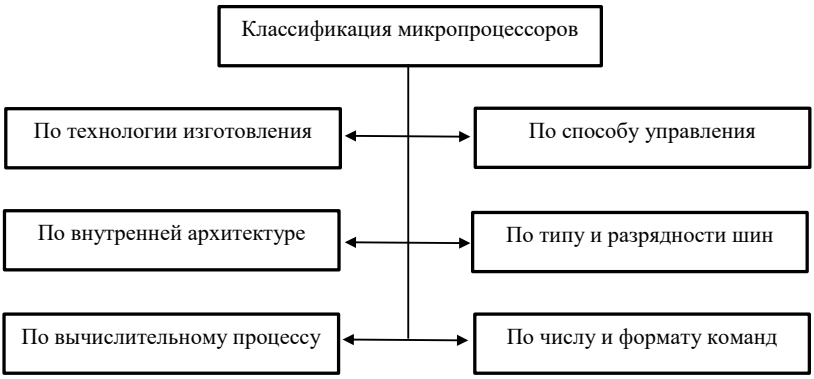


Рисунок 3.6 – Классификация микропроцессоров

3.3. Основные компоненты и устройства ЭВМ

В зависимости от функционального назначения и технических характеристик в состав современных ЭВМ и систем могут входит различные компоненты и устройства, необходимые для реализации ЭВМ своих функций. Безусловно базовыми компонентами являются: центральный процессор, выполняющий в том числе функции управления и синхронизации, различные внешние и внутренние запоминающие устройства (память), необходимые для обеспечения взаимодействия с внешней средой устройства ввода и вывода данных (информации), а также внешние и внутренние интерфейсы[4]. Дополнительно ЭВМ и их системы могут содержать различные устройства, необходимые для обеспечения связи с другими техническими устройствами и коммуникаций, например, модемы.

Типовыми и широко используемыми в повседневной жизни устройствами ввода являются: клавиатуры, различные манипуляторы (типа: «мышь», «джойстик», «трекбол», «точпад»), сенсорные панели (экраны), сканеры, видео-

[4] Интерфейсом принято называть средства и способы установления и поддержания информационного обмена между исполнительными устройствами ЭВМ или иной другой технической системы, а также между человеком и машиной (системой).

аудио-преобразователи (камеры, микрофоны и т.п.). Наиболее известными современному потребителю устройствами вывода являются: мониторы (жидкокристаллические, сенсорные и иные экраны, видеопанели), принтеры, плоттеры, графопостроители, звуковые источники (колонки, синтезаторы звука), различные промышленные и бытовые манипуляторы и т.п. При этом некоторые устройства ввода/вывода являются универсальными, т.е. могут совмещать выполняемые функции, осуществляя одновременно как вывод, так и ввод информации, как это реализовано, например, в сенсорных видеопанелях.

Всю совокупность компонентов и устройств ЭВМ можно условно разделить на внутренние и внешние устройства (компоненты), как это показано на рис. 3.7. Одинаковые по функциональному назначению устройства могут кардинально различаться по конструкции, принципу работы и основным технические параметрам.

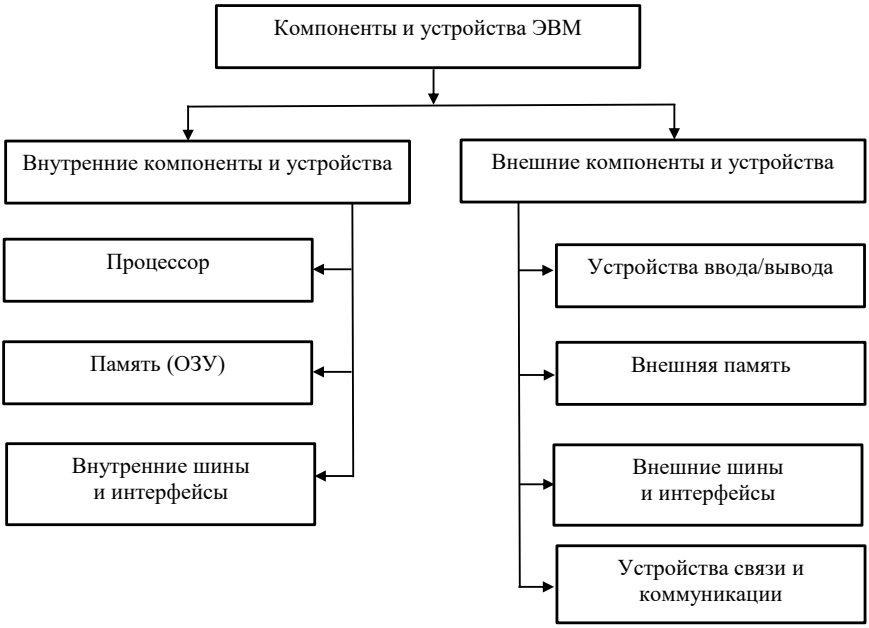


Рисунок 3.7 – Упрощенная классификация компонентов и устройств ЭВМ

Необходимость использования и конкретный выбор типа того или иного устройства зависит от функционального назначения, технических параметров, условий эксплуатации, конструктивного исполнения и других характеристик ЭВМ или вычислительной системы [1-16].

Контрольные вопросы

1. Типовые блоки и узлы микропроцессора: назначение и взаимодействие.
2. История развития, поколения и типы микропроцессоров.
3. Основные технические характеристики и классификация современных микропроцессоров.
4. Семейство микропроцессоров КР580 и Intel, технические параметры.
5. Принципы Лебедева – фон Неймана. Упрощенная структура ЭВМ.
6. Аппаратное и программное обеспечение ЭВМ.
7. Поколения и классификация ЭВМ. Особенности персональных ЭВМ.
8. Характеристики ЭВМ, определяемые микропроцессором.
9. Состав аппаратных средств ЭВМ на примере современной ПЭВМ.
10. Устройства ввода/вывода ЭВМ. Назначение, основная терминология, функции и типы.
11. *Конструкция и состав системного блока ПЭВМ.
12. *Конструкция и состав материнской платы ПЭВМ.
13. *Клавиатура и манипуляторы ПЭВМ.
14. *Структура памяти ЭВМ. Сравнение видов памяти.
15. *Статические оперативные запоминающие устройства (ОЗУ).
16. *Динамические оперативные запоминающие устройства (ОЗУ).
17. *Матричная система адресации ячеек памяти ОЗУ.
18. *Разновидности, конструкция и параметры модулей ОЗУ.
19. *Постоянные запоминающие устройства (ПЗУ).
20. *Магнитные и оптические дисковые накопители: классификация, конструкция, принцип работы, основные параметры
21. *Мониторы: классификация, конструкция, принцип работы, основные параметры. Структурная схема монитора.
22. *Сканеры: классификация, конструкция, принцип работы, основные параметры.
23. *Принтеры и плоттеры: классификация, конструкция, принцип работы, основные параметры.
24. *Сенсорные панели: классификация, конструкция, принцип работы, основные параметры.
25. *Модемы: классификация, конструкция, принцип работы, основные параметры.

(* - факультативные вопросы)

Список рекомендуемой литературы

1. Архитектура компьютера: учебное пособие / Н.Б. Доганин. – М.: Бином. Лаборатория знаний, 2008. – 271 с.: ил. – (Педагогическое образование).
2. Архитектура ЭВМ и систем: учеб. пособие для бакалавров / О.П. Новожилов. – М.: Издательство Юрайт, 2012. – 527 с. – Серия: Бакалавр.
3. Горнец Н.Н., Рощин А.Г., ЭВМ и периферийные устройства. Устройства ввода-вывода. Учебник для студентов учреждений высшего профессионального образования. М.: Академия, 2013- 244с.
4. Доганин Н.Б. Архитектура компьютера. – М.: Бином,2015. –260 с.
5. Горнец Н.Н., Рощин А.Г., Соломенцев В.В. Организация ЭВМ и систем, М.: Издательский центр Академия,2008- 320 с.
6. Калиш Г.Г. Основы вычислительной техники. Учеб. пособ. для средн. проф. учебных заведений. – М.: Высшая шк., 2000. – 271 с.: ил.
7. Мюллер Скотт. Модернизация и ремонт ПК: Пер. с англ. – М.: ООО «И.Д.Вильямс», 2010.
8. Микропроцессоры: Справочное пособие для разработчиков судовой РЭА/Г.Г. Гришин, А.А. Мошков, О.В. Ольшанский, Ю.А. Овечкин. – 2-е изд., стереотип. – Л.: Судостроение, 1988. – 520 с., ил.
9. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник. В 2 т. / В.-Б. Б. Абрайтис, Н.Н. Аверьянов, А.И. Белоус и др.; Под ред. В.А. Шахнова. – М.: Радио и связь, 1988. Т. 1. – 368 с.: ил.
10. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник/С.Т. Хвощ, Н.Н. Варлинский, Е.А. Попов; Под общ. Ред. С.Т. Хвоща. – Л.: Машиностроение. Ленингр. отд-ние, 1987. – 640 с.: ил.
11. Попов И.Т., Партыка Т.Л. Электронные вычислительные машины и системы. – М.: Форум: Инфра-М, 2000.
12. Потемкин И.С. Функциональные узлы цифровой автоматики. – М.: Энергоатомиздат, 1988. – 320 с.: ил.
13. Шило В.Л. Популярные цифровые микросхемы: Справочник. – М.: Радио и связь, 1987. – 352 с.: ил. – (Массовая радиобиблиотека. Вып. 1111).
14. ЭВМ и периферийные устройства. Устройства ввода-вывода: учебник для студ. учреждений высш. проф. образования / Н.Н. Горнец, А.Г. Рощин. – М.: Издательство центр «Академия», 2013. – 224 с. – (Сер. Бакалавриат).
15. Электронные вычислительные машины и системы: учебное пособие / Т.Л. Партыка, И.И. Попов. – М.: Форум: Инфра-М, 2010. – 368 с.: ил. – (Профессиональное образование).
16. Электронные вычислительные машины и системы. Учеб. для техникумов спец. ЭВТ. – 2-е изд., доп. и перераб. – М.: Высш. шк., 1989. – 366 с.: ил.