Учреждение образования
«Брестский государственный технический университет»

Факультет электронно-информационных систем
Кафедра «Интеллектуальные информационные технологии»

СОГЛАСОВАНО
Заведующий кафедрой
_____ В. А. Головко
«21» 03 2025 г.

СОГЛАСОВАНО
Декан факультета
_____ В. С. Разумейчик
«22» 04 2025 г.

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

«АРХИТЕКТУРА ЭВМ»

(название дисциплины)

для специальности (направления специальности):

6-05-0612-03 Системы управления информацией

Составители: Савицкий Юрий Викторович, к.т.н., доцент
Муравьев Геннадий Леонидович, к.т.н., доцент

Рассмотрено и утверждено на заседании Научно-методического совета университета 26.06.2025 г., протокол № 4.

рег. № УМК 241125-100

Пояснительная записка

Дисциплина «Архитектура ЭВМ» является одной из базовых в процессе подготовки студентов по специальности 6-05-0612-03 Системы управления информацией; относится к государственному компоненту; в значительной степени носит системный характер, формируя основу для освоения ряда последующих курсов; обеспечивает выработку систематизированных знаний по основам структурной, функциональной и арифметико-логической организации современных средств компьютерной техники; имеет выраженную практическую направленность.

Цель преподавания учебной дисциплины: изучение основ организации ЭВМ; формирование у студентов систематизированного представления об особенностях архитектуры современных ЭВМ и их базовых элементов; получение практических навыков по организации вычислительного процесса в компьютерных системах.

Задачи учебной дисциплины:

- приобретение знаний по организации современных ЭВМ и принципам их функционирования;
- изучение принципов построения типовых цифровых устройств ЭВМ;
- овладение методами организации вычислений в ЭВМ.

Электронный учебно-методический комплекс (ЭУМК) объединяет структурные элементы научно-методического обеспечения образования и представляет собой сборник материалов теоретического и практического характера для организации работы студентов специальности 6-05-0612-03 Системы управления информацией дневной формы получения высшего образования, заочной формы получения высшего образования, заочной формы получения высшего образования, интегрированного со средним специальным образованием по изучению дисциплины «Архитектура ЭВМ».

ЭУМК разработан на основании Положения об учебно-методическом комплексе на уровне высшего образования, утвержденного Постановлением Министерства образования Республики Беларусь от 8 ноября 2022 г. № 427.

ЭУМК разработан в полном соответствии с учебной программой Учреждения образования «Брестский государственный технический университет», утвержденной 28.06.2021 г., регистрационный № УД-21-1-233/уч. по учебной дисциплине государственного компонента «Архитектура ЭВМ».

Цели ЭУМК можно сформулировать следующим образом: обеспечение качественного методического сопровождения процесса обучения; организация эффективной самостоятельной работы студентов.

Актуальность создания ЭУМК по курсу «Архитектура ЭВМ» обусловлена несколькими ключевыми факторами:

- фундаментальная важность дисциплины. «Архитектура ЭВМ» — базовая дисциплина в подготовке IT-специалистов (инженеров, программистов, системных администраторов). Понимание устройства ЭВМ необходимо для эффективной работы с современными вычислительными системами, включая многопроцессорные и многомашинные системы;

- необходимость систематизации знаний в условиях быстрого развития технологий ЭВМ требует актуализации учебных материалов;

- ЭУМК по курсу «Архитектура ЭВМ» позволяет структурировать материал и обеспечить единый стандарт обучения.

Содержание и объем ЭУМК полностью соответствуют образовательным стандартам высшего образования специальности 6-05-0612-03 Системы управления информацией, а также учебно-программной документации образовательных программ высшего образования. Материал представлен на требуемом методическом уровне и адаптирован к современным образовательным технологиям.

Структура электронного учебно-методического комплекса по дисциплине «Архитектура ЭВМ»

Теоретический раздел ЭУМК содержит материалы для теоретического изучения учебной дисциплины и представлен конспектом лекций.

Практический раздел ЭУМК содержит материалы для проведения практических учебных занятий в виде методических указаний для выполнения практических работ.

Раздел контроля знаний ЭУМК содержит материалы для итоговой аттестации (экзаменационные вопросы, практические задачи), позволяющие определить соответствие результатов учебной деятельности обучающихся требованиям образовательных стандартов высшего образования и учебно-программной документации образовательных программ высшего образования.

Вспомогательный раздел включает учебную программу учреждения высшего образования по учебной дисциплине «Архитектура ЭВМ».

Рекомендации по организации работы с ЭУМК:

- лекции проводятся с использованием представленных в ЭУМК учебных теоретических материалов; при подготовке к экзамену, практическим занятиям студенты могут использовать конспект лекций и набор практических задач;

- практические занятия проводятся в условиях учебной аудитории с использованием представленных в ЭУМК методических указаний;

- экзаменационные материалы приведены в разделе контроля знаний.

ОГЛАВЛЕНИЕ

1 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

1.1 КОНСПЕКТ ЛЕКЦИЙ

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Начало развития машин для выполнения вычислений можно отнести к семнадцатому столетию, когда в 1642 г. выдающийся французский математик Блез Паскаль изобрел и сконструировал первую суммирующую машину. Однако из-за несовершенства развития механики того времени созданная им машина не нашла практического применения. В конце этого же столетия в 1694 г. немецкий математик Гофрид Лейбниц построил вычислительную машину, которая могла выполнять не только операции сложения, но и операции умножения. В 1874 г. российский инженер В.Т. Однер сконструировал арифмометр, которой был лучшим для того времени и долгое время оставался прототипом разрабатывавшихся впоследствии машин подобного назначения.

Идея создания машины с элементами программирования, т.е. с автоматической организацией вычислительного процесса, принадлежит Чарльзу Боббиджу, предложившему в 1833 г. проект «аналитической машины», которая помимо использования принципа программного управления имела память для хранения программы, исходных данных и результатов их обработки. В предложенной машине были предусмотрены средства ввода исходной информации с перфоленты и средства вывода информации в печатной форме. Эта машина была даже построена, однако реализация достаточно сложных идей, положенных в основу этой «аналитической машины», на базе сравнительно несовершенной механики того времени привела к тому, что созданная машина оказалась настолько ненадежной, что не нашла фактически никакого практического использования.

Все разрабатываемые машины для выполнения расчетов до 40-х гг. строились на основе механических узлов. Появление первых электронных вычислительных машин можно отнести к сороковым годам. Одной из первых ЭВМ является «ENIAC», созданная в США Джоном Мокли и Дж. Преспером Эккертом. При построении машины было использовано около 20 тысяч электронных ламп. Она имела колоссальные габариты (для ее размещения было построено специальное помещение площадью более 100 метров) и имела фантастическое по тем временам быстродействие - 5 тысяч сложений в секунду.

С 40-х гг. развитие вычислительной техники с точки зрения используемой технологической базы можно подразделить на этапы создания ЭВМ следующих поколений:

- ЭВМ на электронных вакуумных приборах;
- ЭВМ на полупроводниковых приборах;
- ЭВМ на интегральных схемах;
- ЭВМ на больших интегральных схемах;
- ЭВМ на сверхбольших интегральных схемах.

При переходе от поколения к следующему поколению почти на порядок улучшались основные параметры ЭВМ, к числу которых относятся:

- быстродействие;
- емкость памяти;
- потребляемая мощность;

- габариты.

За это время существенно изменилась структурная организация ЭВМ, её внешняя память, средства ввода - вывода информации.

Современные ЭВМ при габаритах, как правило, соответствующих настольному варианту, обладают быстродействием, измеряемым десятками миллионов операций в секунду, имеют только оперативную память емкостью в десятки (а в некоторых случаях сотни) миллионов байт, оснащены разнообразными средствами ввода -вывода, позволяющими вести обмен информации с пользователем в удобной для последнего форме.

Структура современной ЭВМ включает следующие основные компоненты:

- ОП - оперативная память, используемая для хранения исполняемых в данное время программ, исходных данных, промежуточных и окончательных результатов;

- процессор (ЦП, CPU) - устройство, осуществляющее основную обработку информации в соответствии с исполняемой программой;

- ПУ - периферийные устройства, включающие средства ввода - вывода информации, представленной в различной форме, осуществляющие двусторонний обмен данными с пользователем, а также устройства типа внешней памяти, имеющие огромную информационную ёмкость, позволяющую хранить все исходные данные, программы, промежуточные и конечные результаты обработки информации.

В данном материале рассматриваются основополагающие материалы, связанные с ЭВМ, а именно:

- *арифметические основы ЭВМ;*
- *логические основы ЭВМ;*
- *схемотехнические основы ЭВМ.*

I Общие вопросы организации ЭВМ

1 Понятие архитектуры ЭВМ

Однозначно определить понятие архитектуры ЭВМ довольно трудно, потому что при желании в него можно включить все, что связано с компьютерами вообще и какой-то конкретной моделью компьютера в частности. Попытаемся все же его формализовать.

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

- структурная схема ЭВМ;
- организация и разрядность интерфейсов ЭВМ;
- набор и доступность регистров;
- организация и способы адресации памяти;
- способы представления и форматы данных ЭВМ;
- набор и форматы машинных команд ЭВМ;
- правила обработки нештатных ситуаций (прерываний).

Таким образом, описание архитектуры включает в себя практически всю необходимую для программиста информацию о компьютере. Понятие архитектуры ЭВМ — иерархическое. Допустимо вести речь как об архитектуре компьютера в целом, так и об архитектуре отдельных его составляющих, например, архитектуре процессора или архитектуре подсистемы ввода-вывода.

Все современные компьютеры обладают некоторыми общими и индивидуальными архитектурными свойствами. Индивидуальные свойства присущи только конкретной модели компьютера. Общие архитектурные свойства, наоборот, присущи некоторой, часто довольно большой группе компьютеров. На сегодняшний день общие архитектурные свойства большинства современных компьютеров подпадают под понятие *фон-Неймановской архитектуры*. Так названа архитектура по имени ученого фон Неймана. Когда фон Нейман начал заниматься компьютерами, программирование последних осуществлялось способом коммутирования. В первых ЭВМ для генерации нужных сигналов необходимо было с помощью переключателей выполнить ручное программирование всех логических схем. В первых машинах использовали десятичную логику, при которой каждый разряд представлялся десятичной цифрой и моделировался 10 электронными лампами. В зависимости от нужной цифры одна лампа включалась, остальные девять оставались выключенными. Фон Нейман предложил схему ЭВМ с программой в памяти и двоичной логикой вместо десятичной. Логически машину фон Неймана составляли пять блоков (рисунок 1.1): оперативная память, арифметико-логическое устройство (АЛУ) с аккумулятором, блок управления, устройства ввода и вывода. Особо следует выделить роль аккумулятора. Физически он представляет собой регистр АЛУ. Для процессоров Intel, в которых большинство команд — двухоперандные, его роль не столь очевидна. Но существовали и существуют процессорные среды с

однооперандными машинными командами. В них наличие аккумулятора играет ключевую роль, так как большинство команд используют его содержимое в качестве либо второго, либо единственного операнда команды.

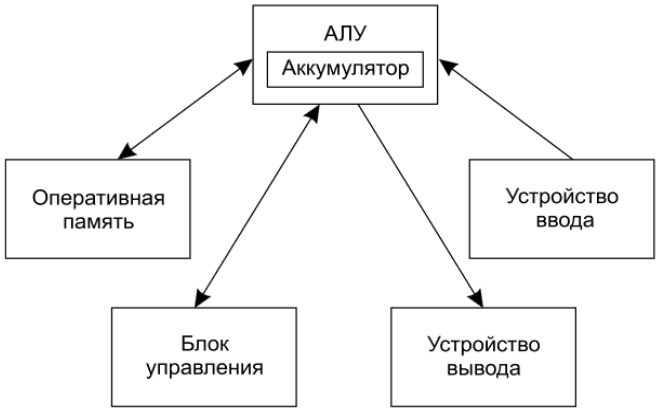


Рисунок 1.1 – Схема машины фон Неймана

Ниже описаны свойства и принципы работы машины фон Неймана.

Линейное пространство памяти. Для оперативного хранения информации компьютер имеет совокупность ячеек с последовательной нумерацией (адресами) 0, 1, 2, … Данная совокупность ячеек называется оперативной памятью.

Принцип хранимой программы. Согласно этому принципу, код программы и ее данные находятся в одном и том же адресном пространстве оперативной памяти (ОП).

Принцип микропрограммирования. Суть этого принципа заключается в том, что машинный язык еще не является той конечной субстанцией, которая физически приводит в действие процессы в машине. В состав процессора входит устройство микропрограммного управления, поддерживающее набор действий-сигналов, которые нужно сгенерировать для физического выполнения каждой машинной команды.

Последовательное выполнение программ. Процессор выбирает из памяти команды строго последовательно. Для изменения прямолинейного хода выполнения программы или осуществления ветвления необходимо использовать специальные команды. Они называются командами условного и безусловного переходов.

Отсутствие разницы между данными и командами в памяти. С точки зрения процессора, нет принципиальной разницы между данными и командами. Данные и машинные команды находятся в одном пространстве памяти в виде последовательности нулей и единиц. Это свойство связано с предыдущим. Процессор, поочередно обрабатывая некоторые ячейки памяти, всегда пытается трактовать содержимое ячеек как коды машинных команд, а если это не так, то происходит аварийное завершение программы. Поэтому важно всегда четко разделять в программе пространства данных и команд.

Безразличие к назначению данных. Машине все равно, какую логическую

нагрузку несут обрабатываемые ею данные.

2 Организация ЭВМ с магистральной архитектурой

На рисунке 1.2 приведена обобщенная схема типовой ЭВМ с магистральной архитектурой.

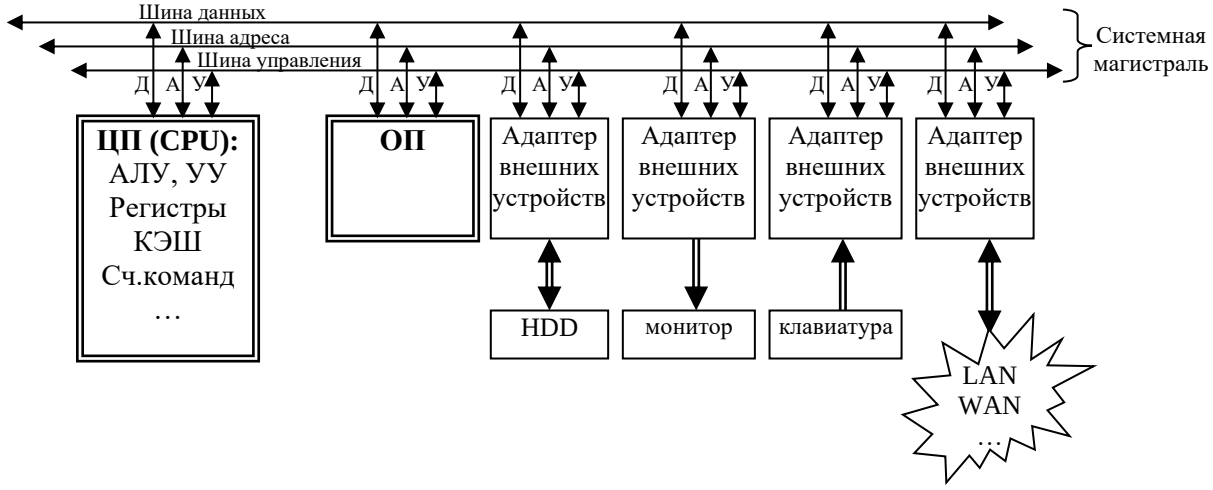


Рисунок 1.2 – Обобщенная схема типовой ЭВМ с магистральной архитектурой

Все устройства ЭВМ делятся на центральные и периферийные. В центральных устройствах основным узлом, связывающим микропроцессорный комплект в единое целое, является системная магистраль (СМ). Она состоит из трех узлов, называемых шинами: шина данных (ШД), шина адреса (ША), шина управления (ШУ). В состав системной магистрали входят регистры защелки, в которых запоминается передаваемая информация, шинные формирователи, шинные арбитры, определяющие очередность доступа к системной магистрали, и др.

Организация функционирования ЭВМ с магистральной архитектурой.

1. Управляющая работой ЭВМ программа перед началом выполнения загружается операционной системой в основную память. Адрес первой выполняемой команды передается микропроцессору и запоминается в счетчике команд.

2. Начало работы процессора (ЦП) заключается в том, что адрес из счетчика команд (в котором всегда хранится адрес очередной команды) выставляется на шину адреса системной магистрали.

3. Одновременно на шину управления выдается команда: выборка из ОП, которая воспринимается основной памятью.

4. Получив с шины управления системной магистрали команду, основная память считывает адрес с шины адреса, находит ячейку с этим номером и ее содержимое выставляет на шину данных.

5. Процессор, получив по шине управления сигнал об окончании работы ОП, вводит число с шины данных на внутреннюю магистраль процессора и через нее пересылает введенную информацию в регистр команд.

6. В регистре команд полученная команда разделяется на кодовую и адресную части. Код команды поступает в блок управления ЦП для выполнения заданной операции, и для определения адреса следующей команды (который сразу заносится в счетчик команд).

7. Адресная часть команды выставляется на шину адреса системной магистрали и сопровождается сигналом выборки из ОП на шине управления. Выбранная из ОП информация через шину данных поступает на внутреннюю магистраль МП, с которой вводится в арифметико-логическое устройство (АЛУ). На этом заканчивается подготовка ЦП к выполнению операции и начинается ее выполнение в АЛУ.

8. Результат выполнения операции выставляется микропроцессором на шину данных, на шину адреса ставится адрес ОП, по которому этот результат необходимо записать, а на шину управления – команда запись в ОП.

9. Получив с шины управления команду, ОП считывает адрес и данные с системной магистрали, организует запись данных по указанному адресу и после выполнения команды выставляет на шину управления сигнал, обозначающий, что число записано.

10. Процессор, получив этот сигнал, начинает выборку очередной команды: выставляет адрес из счетчика команд на шину адреса, формирует команду выборки из ОП на шине управления и т.д.

В каждом цикле, получив команду в регистр команд и выделив код операции, процессор определяет, к какому устройству она относится. Если команда должна выполняться процессором, организуется ее выполнение по описанному циклу. Если же команда предназначена для выполнения в другом устройстве ЭВМ, центральный процессор передает ее соответствующему устройству. Процесс передачи команды другому устройству предусматривает следующие действия: ЦП выставляет на шину адреса СМ адрес интересующего его устройства. По шинам управления передается сигнал поиска устройства. Все устройства, подключенные к системной магистрали, получив этот сигнал, читают номер устройства с шины адреса и сравнивают его со своим номером. Устройства, для которых эти номера не совпадают, на эту команду не реагируют. Устройство с совпавшим номером вырабатывает сигнал отклика по шине управления.

ЦП, получив сигнал отклика, в простейшем случае выставляет имеющуюся у него команду на шину данных и сопровождает ее по шине управления сигналом "передаю команду". Получив сигнал о приеме команды, ЦП переходит к выполнению очередной своей команды, выставляя на шину адреса содержимое счетчика команд.

В более сложных случаях, получив сигнал, что устройство откликнулось, прежде чем передавать команду, ЦП запрашивает устройство о его состоянии. Если устройство включено и готово к работе, то байт состояния - нулевой.

Взаимодействие МП с внешними устройствами предусматривает выполнение логической последовательности действий, связанных с поиском устройства, определением его технического состояния, обменом командами и информацией. Эта логическая последовательность действий вместе с устройствами, реализующими ее, получила название "интерфейс ввода-вывода".

Для различных устройств могут использоваться разные логические последовательности действий, поэтому интерфейсов ввода-вывода может в одной и той же ЭВМ использоваться несколько. Если их удается свести к одному, универсальному, то такой интерфейс называется стандартным.

Интерфейсы постоянно совершенствуются, поэтому с появлением новых ЭВМ, новых внешних устройств и даже нового программного обеспечения появляются и новые интерфейсы.

Если при обращении ЦП к внешнему устройству продолжение выполнения основной программы ЦП возможно только после завершения операции ввода-вывода, то ЦП, запустив внешнее устройство, переходит в состояние ожидания и находится в нем до тех пор, пока внешнее устройство не сообщит ему об окончании обмена данными. Это приводит к простою большинства устройств ЭВМ, так как в каждый момент времени может работать только одно из них. Такой режим работы получил название однопрограммного – в каждый момент времени все устройства находятся в состоянии ожидания, и только одно устройство выполняет основную (и единственную) программу.

Для ликвидации таких простоев и повышения эффективности работы оборудования внешние устройства сделаны автономными: получив от ЦП необходимую информацию, они самостоятельно организуют свою работу по обмену данными. Процессор же, запустив внешнее устройство, пытается продолжить выполнение программы. При необходимости (если встретятся соответствующие команды) он может запустить в работу несколько других устройств (так как внешние устройства работают значительно медленнее процессора). Если же ему приходится переходить в режим ожидания, то, пользуясь тем, что в ОП может одновременно находиться не одна, а несколько программ, ЦП переходит к выполнению очередной программы. При этом создается ситуация, когда в один и тот же момент времени различные устройства ЭВМ выполняют либо разные программы, либо разные части одной и той же программы. Такой режим работы ЭВМ называется многопрограммным.

Логика работы системной магистрали, количество разрядов (линий) в шинах данных, адреса и управления, порядок разрешения конфликтных ситуаций, возникающих при одновременном обращении различных устройств ЭВМ к системной магистрали, образуют интерфейс системной шины.

Таким образом, в состав центральных устройств ЭВМ входят: центральный процессор, основная память и ряд дополнительных узлов, выполняющих служебные функции.

Периферийные устройства делятся на два вида: внешние ЗУ (НМД, НГМД, НМЛ) и устройства ввода-вывода (УВВ): клавиатура, дисплей, принтер, мышь, адаптер каналов связи (КС) и др.

3 Общие сведения о системном программном обеспечении ЭВМ

Системное программное обеспечение (СПО) — это комплекс программ, обеспечивающих управление аппаратными ресурсами компьютера, взаимодействие пользователя с системой и работу прикладного программного обеспечения.

Основные функции СПО:

- управление аппаратными ресурсами (процессором, памятью, устройствами ввода-вывода);
- обеспечение интерфейса между пользователем и аппаратурой;
- организация выполнения прикладных программ;
- обеспечение безопасности и защиты данных.

3.1 Основные компоненты системного ПО

1. Операционная система (ОС) – главная часть СПО, управляющая всеми процессами в ЭВМ. Примеры: Windows, Linux, macOS, Unix.

Функции:

- управление процессами и памятью;
- файловая система;
- взаимодействие с пользователем (интерфейс);
- управление устройствами (драйверы).

2. Драйверы устройств – программы, обеспечивающие взаимодействие ОС с аппаратными компонентами (видеокарта, принтер, сетевая карта).

3. Утилиты – вспомогательные программы для обслуживания системы:

- архиваторы (WinRAR, 7-Zip);
- антивирусы (Kaspersky, Avast);
- программы диагностики (AIDA64, HWMonitor);
- утилиты для работы с дисками (Defrag, Partition Magic).

4. Системные библиотеки – наборы функций, используемых прикладными программами (DLL в Windows, .so в Linux).

5. Среды разработки и компиляторы – инструменты для создания ПО (GCC, Visual Studio).

6. BIOS/UEFI – микропрограмма, выполняющая начальную загрузку компьютера и настройку оборудования.

3.2 Базовая система ввода-вывода (BIOS/UEFI)

Базовая система ввода-вывода (BIOS – Basic Input/Output System, UEFI – Unified Extensible Firmware Interface) – это встроенное в материнскую плату программное обеспечение, которое выполняет первоначальную настройку и проверку оборудования при включении компьютера, а затем загружает операционную систему.

Основные функции BIOS/UEFI.

1. POST (Power-On Self-Test) – самотестирование оборудования при включении – проверка процессора, памяти, видеокарты, дисков и другого оборудования.

2. Инициализация и настройка оборудования (чипсет, USB, SATA, сетевые контроллеры).

3. Загрузка ОС (поиск загрузчика на HDD/SSD, USB, DVD, сети).

4. Предоставление API для ОС – базовые функции ввода-вывода до загрузки драйверов.

5. Настройка параметров аппаратного обеспечения (разгон, управление питанием, безопасность).

Основные настройки BIOS/UEFI.

1. Boot Priority – порядок загрузки (HDD, USB, DVD).
2. Secure Boot – защита от несанкционированного ПО.
3. AHCI/NVMe/Raid – режимы работы дисков.
4. XMP/D.O.C.P – профили разгона оперативной памяти.
5. CPU Settings – управление частотой процессора, множителем.
6. TPM (Trusted Platform Module) – для Windows 11 и шифрования.
7. Virtualization (VT-x/AMD-V) – для виртуальных машин.

Таким образом, BIOS/UEFI – критически важный компонент компьютера, отвечающий за старт системы и низкоуровневые настройки оборудования. Современные ЭВМ используют UEFI как более быструю и безопасную замену устаревшему BIOS.

II Арифметические основы ЭВМ

1 Системы счисления

Арифметическая обработка чисел во многом определяется системами счисления, представляющими собой совокупность используемых цифр и набором правил, позволяющих однозначно представлять числовую информацию.

В своей повседневной деятельности человек использует различные системы счисления, к числу которых относятся десятичная система счисления, римская система, система исчисления времени и т.д. Все системы счисления можно подразделить на позиционные и непозиционные.

В не позиционных системах счисления «доля» цифры или её вес в количественном измерении записанного числа не зависит от местоположения данной цифры в записи этого числа. Типичным примером такой системы счисления является римская система счисления. В этой системе используются цифры:

I	V	X	L	C	D	M	и т.д.	- римские цифры;
1	5	10	50	100	500	1000		- десятичные эквиваленты римским цифрам.

При количественной оценки числа его значение определяется как сумма значений цифр, составляющих запись числа, кроме пар, состоящих из цифры меньшего веса, предшествующей цифре большего веса, значение которой определяется как разность веса большей и меньшей цифр. Например, значение числа

MMMCMLIX

определяется как сумма

1000 + 1000 + 1000 + (1000-100) + 50 + (10 - 1), что соответствует десятичному эквиваленту 3959.

Позиционная система счисления характеризуется тем, что «доля» некоторой цифры в количественной оценки записанного числа определяется не только видом цифры, но и местоположением (позицией) данной цифры в записи числа, т.е. каждая позиция (разряд) в записи числа имеет определенный вес.

Количественная оценка записанного числа в такой системе счисления определяется как сумма произведений значения цифр, составляющих запись числа, умноженных на вес позиции, в которой располагается цифра.

Примером такой системы счисления является широко используемая десятичная система счисления. Например, количественная оценка десятичного числа 3959_{10} определяется как

$3 \cdot 1000 + 9 \cdot 100 + 5 \cdot 10 + 9 \cdot 1$, где 1000, 100, 10, 1 - соответственно веса четвертого, третьего, второго, первого разрядов записи оцениваемого числа.

Десятичная *система счисления* является также системой с равномерно распределенными весами, которые характеризуются тем, что соотношение весов двух любых соседних разрядов имеют для такой системы одинаковое значение.

Это соотношение называется основанием системы счисления, которое в дальнейшем будем обозначать как «q».

Общая запись числа в системе с равномерно распределенными весами имеет вид

$$N_q = A_n A_{n-1} .... A_2 A_1 A_0. \quad (2.1)$$

Значение такого числа определяется как

$$N_q = A_n \cdot q^n + A_{n-1} \cdot q^{n-1} + A_{n-2} \cdot q^{n-2} + ... A_2 \cdot q^2 + A_1 \cdot q^1 + A_0 \cdot q^0, \quad (2.2)$$

где A_i - цифра записи числа, удовлетворяющая условию

$0 <= A_i <= (q-1)$;

q - основание системы счисления.

При q =10 A изменяется в диапазоне от 0 до 9, т.е. до (10-1).

Запись числа N в виде (2.1) называется кодированной, а запись в форме (2.2) называется расширенной записью.

Помимо q = 10 (десятичная *система счисления*) возможны другие значения для основания системы счисления:

- двоичная *система счисления*;
- восьмеричная *система счисления*;
- шестнадцатеричная *система счисления* и т.д.

Для обозначения цифр в различных системах счисления в качестве цифр используются обозначение соответствующих цифр десятичной системы счисления - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а в случае, когда десятичных цифр «не хватает» (для систем счисления с основанием q, большим чем 10), для цифр, превышающих 9, вводятся дополнительные обозначения, например, для q =16 это будут обозначения A, B, C, D, E, F, которые соответствуют шестнадцатеричным цифрам, десятичные эквиваленты которых равны соответственно 10, 11, 12, 13, 14, 15.

В связи с тем, что в дальнейшем изложении будут использоваться различные системы счисления, примем обозначение:

N_q - число N, представленное в системе счисления с основанием q.

Примеры записи чисел в различных системах счисления:

$N_2 = 10011011 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$,

$N_8 = 471025 = 4 \cdot 8^5 + 7 \cdot 8^4 + 1 \cdot 8^3 + 0 \cdot 8^2 + 2 \cdot 8^1 + 5 \cdot 8^0$,

$N_{16} = 84FE4A = 8 \cdot 16^5 + 4 \cdot 16^4 + F \cdot 16^3 + E \cdot 16^2 + 4 \cdot 16^1 + A \cdot 16^0$,

$N_{10} = 35491 = 3 \cdot 10^4 + 5 \cdot 10^3 + 4 \cdot 10^2 + 9 \cdot 10^1 + 1 \cdot 10^0$.

На основании вышеизложенного можно заключить, что запись одного и того же числа в различных системах счисления будет тем длиннее, чем меньше основание системы счисления. Например, число N, десятичное значение которого равно 2063, в различных системах счисления представляется как

$N = 2063_{10} = 100000001111_2 = 4017_8 = 80F_{16}$.

При работе с различными системами счисления полезно помнить соотношения, приведенные в таблице 2.1.

Таблица 2.1 – Вспомогательные соотношения

n	0	1	2	3	4	5	6	7	8	9	10	11
2^n	1	2	4	8	16	32	64	128	256	512	1024	2048

Человек в своей практической деятельности наиболее часто использует *десятичную* систему счисления. *Двоичная система счисления* является удобной для обработки информации в ЭВМ. Промежуточное место между этими системами занимает *двоично-десятичная система* счисления. Эта система в принципе является десятичной, но отдельные десятичные цифры в ней записываются в виде набора двоичных разрядов. Существуют различные двоично-десятичные системы, которые отличаются способом представления набором двоичных разрядов десятичных цифр. Наиболее широкое распространение получила двоично-десятичная система 8,4,2,1. Данная система характеризуется тем, что отдельные десятичные цифры в ней представляются их четырехбитовым двоичным эквивалентом, как это показано в таблице 2.2.

Например, десятичное число 804714 в двоично-десятичной системе 8,4,2,1 представляется в виде 1000 0000 0100 0111 0001 0100.

Таблица 2.2 – Представление чисел в различных системах счисления

N п.п.	q = 2	q = 8	q = 16	q = 10	Десятичный эквивалент	Двоичный эквивалент
1	0	0	0	0	0	0000
2	1	1	1	1	1	0001
3		2	2	2	2	0010
4		3	3	3	3	0011
5		4	4	4	4	0100
6		5	5	5	5	0101
7		6	6	6	6	0110
8		7	7	7	7	0111
9			8	8	8	1000
10			9	9	9	1001
11			A		10	1010
12			B		11	1011
13			C		12	1100
14			D		13	1101
15			E		14	1110
16			F		15	1111

2 Перевод чисел из одной системы счисления в другую

Наличие различных систем счисления предполагает использование разных способов перевода записи числа из одной системы счисления в другую. Для этой цели применяются следующие способы преобразований:

– метод преобразования с использованием весов разрядов в исходной и в искомой записи числа;

– метод деления (умножения) на новое основание;

– метод с использованием особого соотношения заданной и искомой систем счисления.

2.1 Метод преобразования с использованием весов разрядов

Метод преобразования с использованием весов разрядов записи числа в исходной и в искомой системах предполагает использование расширенной записи числа (2) в некоторой системе счисления.

Метод имеет две разновидности в зависимости от того, какая система счисления (исходная или искомая) является более привычной. Если более привычной является искомая система счисления, то на основании расширенной записи исходного числа подсчитываются значения её отдельных разрядов в новой системе счисления. Далее полученные значения суммируются.

Например, при преобразовании целого двоичного числа

$N_2 = 110011010$

в десятичную систему счисления исходное число представляется в расширенной записи

$N = 2^8 + 2^7 + 2^4 + 2^3 + 2^1$

и рассчитывается вес отдельных (ненулевых) двоичных разрядов в десятичной системе счисления:

256 , 128, 16, 8, 2.

Затем искомая запись числа определяется как сумма весов всех ненулевых разрядов записи числа в заданной системе счисления:

256 + 128 + 16 + 8 + 2 = 410.

При преобразовании правильных дробей в принципе используется тот же подход, но при расчете весов отдельных разрядов берутся отрицательные степени основания счисления.

Кроме того, учитывая, что при преобразовании правильных дробей в общем случае результат получается неточный, перед началом преобразования необходимо подсчитать количество разрядов представления числа в новой системе счисления. Разрядность результата выбирается таким образом, чтобы ошибка представления результата была бы не более половины единицы младшего разряда в исходной записи числа.

Например, при использовании двоичной и десятичной систем счисления берется соотношение, согласно которому один десятичный разряд соответствует точности представления четырехразрядного двоичного числа.

При преобразовании правильных дробей сначала ищется предварительное значение представления заданного числа в новой системе счисления с количеством разрядов, на единицу большим, чем расчетная разрядность представления числа в новой системе счисления. Дополнительный разряд в предварительном результате преобразования используется для округления, позволяющего с рассчитанным числом разрядов найти окончательный результат.

Пример

Представить правильную двоичную дробь 0.101 в десятичной системе счисления.

Перед началом преобразования определяется, что разрядность записи заданного числа в новой системе счисления должна быть равна 1, поэтому сначала ищется предварительная запись заданного числа в новой системе счисления с двумя десятичными разрядами

$0.101_2 = 2^{-1} + 2^{-3} = 0.5 + 0.13 = 0.63$,

и после округления

$0.101_2 = 0.6_{10}$.

Если более привычной является исходная система счисления, то запись заданного числа в новой системе счисления определяется разряд за разрядом, начиная со старшего. Первым значащим разрядом будет являться разряд с максимально возможным весом, но не превышающим значения преобразуемого числа. При этом, определив старшую цифру (старший разряд) с ненулевым значением, из исходного числа вычитаем вес этого разряда, таким образом формируя остаток, который должен быть представлен еще не найденным младшим разрядом искомой записи числа в новой системе счисления. Далее, используя полученный остаток, аналогичным приемом ищем второй старший разряд записи числа в новой системе счисления, определяем новый остаток и переходим к определению следующего разряда и т.п. Процесс этот напоминает процедуру взвешивания некоторого тела посредством уравновешивания его веса с помощью эталонных гирь.

Пример

Найти двоичный эквивалент десятичного числа:

$436_{10} =$____?_____ 2.

Решение

Первый (старший) разряд, имеющий значение 1 в искомой двоичной записи числа, будет разряд весом $2^8 = 256$. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 180 (180 - остаток, полученный как 436-256).

Второй разряд с весом 2^7 - 128 будет иметь в искомой двоичной записи числа значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 52 (52 - остаток, полученный как 188 - 128).

Третий разряд с весом 2^5 - 64 будет иметь в искомой двоичной записи числа значение 0.

Четвертый разряд с весом 2^5 - 32 будет иметь в искомой двоичной записи числа

значение 1, а остаток - 20.

Пятый разряд с весом 2^4 - 16 будет иметь в искомой двоичной записи числа значение 1, а остаток - 4.

Шестой разряд с весом $2^3 = 8$ будет иметь в искомой двоичной записи числа значение 0.

Седьмой разряд с весом $2^2 = 4$ будет иметь в искомой двоичной записи числа значение 1, а остаток - 0.

Восьмой разряд с весом $2^1 = 2$ будет иметь в искомой двоичной записи числа значение 0.

Девятый разряд с весом $2^0 = 1$ будет иметь в искомой двоичной записи числа значение 0.

Таким образом,

$436_{10} = 110110100_2$.

Пример

Найти двоичный эквивалент числа $0.7_{10} =$_____?_____$_2$.

Предварительный результат ищется с точностью до пяти двоичных разрядов, причем пятый разряд используется только для округления при переходе к четырехразрядному окончательному результату.

Первый (старший) разряд с весом $2^{-1} = 0.5$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (младших) разрядов искомой записи числа необходимо представить значение 0.2 (0.2 - остаток, полученный как 0.7 - 0.5 = = 0.2).

Второй (старший) разряд с весом $2^{-2} = 0.25$ в искомой двоичной записи числа будет иметь значение 0.

Третий разряд с весом $2^{-3}= 0.13$ в искомой двоичной записи числа будет иметь значение 1. С помощью остальных (более младших) разрядов искомой записи числа необходимо представить значение 0.07 (0.07 - остаток, полученный как 0.2 - 0.13).

Четвертый разряд с весом $2^{-4} = 0.06$ в искомой двоичной записи числа будет иметь значение 1, а остаток - 0.01.

Пятый разряд с весом $2^{-5} =0.03$ искомой двоичной записи числа будет иметь значение 0.

Таким образом, десятичное число $0.7_{10} = 0.10110_2$.

После округления имеет место

$0.7_{10} = 0.1011_2$.

2.2 Метод деления (умножения) на новое основание

Метод деления (умножения) имеет две разновидности соответственно для преобразования целых и дробных чисел.

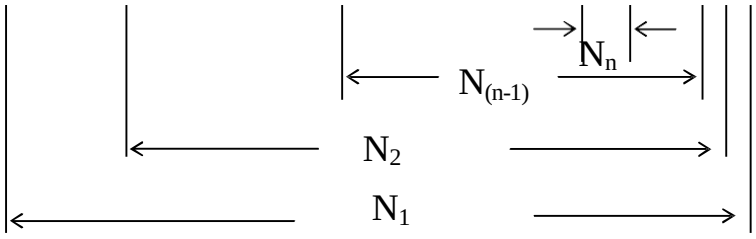
Преобразование целых чисел.

Задачу представления числа N, заданного в системе q_1, в системе счисления с основанием q_2 можно рассматривать как задачу поисков коэффициентов полинома, представляющего собой расширенную запись числа N в системе счисления q_2:

$$N_{q1} = a_0 + a_1 \cdot q_2^{1} + a_2 \cdot q_2^{2} + \ldots + a_{n-2} \cdot q_2^{n-2} + a_{n-1} \cdot q_2^{n-1} + a_n \cdot q_2^{n} = Nq_2. \qquad (2.3)$$

Введем скобочную форму для выражения (2.3):

$$N_{q1} = N_{q2} = a_0 + q_2 (a_1 + q_2 (a_2 + q_2 (a_3 + . + q_2 (a_{n-1} + q_2 (a_n) \ldots))));$$



Обозначим выражение в первой скобке как N_1, выражение во второй скобке как N_2, выражение в третьей скобке - как N_3 и т.д., выражение в (n-1)-й скобке - как $N_{(n-1)}$, выражение в n-й скобке - как N_n. Теперь, основываясь на выражении (2.3), можно утверждать, что при делении N_{q1}/ q_2 будет получена целая часть частного $int(N_{q1}/q_2)$ и остаток $rest(N_{q1}/q_2)$. Это можно записать как

$N_{q1}/q_2 \rightarrow int(N_{q1}/q_2)$,- целая часть частного N_1, и остаток $rest(N_{q1}/q_2)$, равный a_0;

Аналогично для остальных скобок будем иметь:

$N_1/q_2 \rightarrow int(N_1/q_2)$, равно N_2 и остаток $rest(N_1/q_2)$, равный a_1;

$N_2/q_2 \rightarrow int(N_2/q_2)$, равно N_3, и остаток $rest(N_2/q_2)$, равный a_2;

.....

$N_{(n-2)}/q\,2 \rightarrow int(N_{(n-2)}/q_1)$, равно $N_{(n-1)}$), остаток $rest(N(n=2)/q_1)$, равный $a_{(n-2)}$;

$N_{(n-1)}/q_2 \rightarrow int(N_{(n-1)}/q_2) = N_n = a_n$, и остаток $rest(N_{(n-1)}/q_2)$, равный $a_{(n-1)}$; при этом $N_n < q_2$.

Отсюда вытекает правило формирования коэффициентов полинома (3) или разрядов записи заданного числа N в системе счисления с основанием q_2:

- необходимо разделить исходное число N_{q1} на новое основание q_2, при этом получив целое частное и остаток;

- полученный остаток снова необходимо разделить на q_2, процесс деления продолжается до тех пор, пока частное будет не меньше нового основания q_2. Если очередное сформированное частное будет меньше, чем q_2, то процесс формирования записи заданного числа в новой системе с основанием q_2 считается законченным, а в качестве искомых разрядов новой записи числа используются результаты выполненных операций деления следующим образом:

- в качестве старшего разряда берется значение последнего частного, для остальных разрядов используются значения остатков в порядке, обратном порядку их получения.

Пример

Найти запись в двоичной форме десятичного числа $N_{10} = 436$.

Решение

Делим сначала исходное число N_{10}, а затем получаемые частные на значение нового основания 2 до получения частного со значением, меньше, чем два:

436/2→int(436/2) =218 и rest (436/2) = 0;
218/2→int(218/2) =109 и rest (218/2) = 0;
109/2→int(109/2) =54 и rest (109/2) = 1;
54/2→int(54/2) = 27и rest (54/2) = 0;
27/2→int(27/2) = 13 и rest (27/2) = 1;
13/2→int(13/2) = 6 и rest (13/2) = 1;
6/2→int(6/2) = 3 и rest (6/2) = 0;
3/2→int(3/2) = 1 и rest (3/2)= 1.

Таким образом, 436 = 11 0110100.

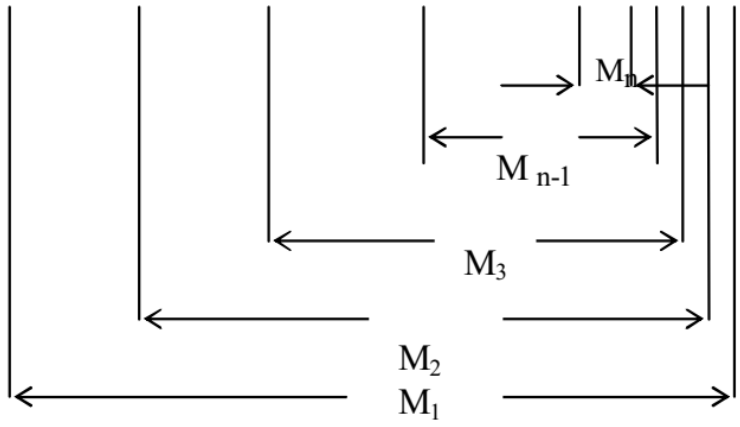
Преобразование дробных чисел.

Задача представления дробного числа Mq_1, заданного в системе q_1, в системе счисления с основанием q_2, можно рассматривать как задачу поиска коэффициентов полинома, представляющего собой расширенную запись числа M в системе счисления q_2:

$$b_1 \cdot q_2^{-1} + b_2^{-2} \cdot q_2 + b_3 \cdot q_2^{-3} + \ldots + b_{n-2} \cdot q_2^{-(n-2)} + b_{n-1} \cdot q_2^{-(n-1)} + b_n \cdot q_2^{-n} = M_{q2}. \qquad (2.4)$$

Введем скобочную форму для выражения (2.4):

$$M_{q2} = M_{q1} = q_2^{-1}(b_1 + q_2^{-1}(b_2 + q_2^{-1}(b_3 + \ldots + q_2^{-1}(b_{n-1} + q_2^{-1}(b_n))\ldots))));$$



Обозначим выражение в первой скобке как M_1, выражение во второй скобке - как M_2, выражение в третьей скобке - как M_3 и т.д., выражение в (n-1)-й скобке как M_{n-1}, выражение в n-й скобке как M_n.

Число M_{q1} - правильная дробь, поэтому при умножении $M\ q_1 \cdot q_2$ будет полу- чено произведение, в общем случае состоящее из целой части $int(M\ q_1 \cdot q_2)$ и дробной части $DF\ (M\ q_1 \cdot q_2)$. Использование введенных обозначений позволяет записать:

$M\ q_1 \cdot q_2 = (int(M\ q_1 \cdot q_2) = b_1) + (DF(M\ q_1 \cdot q_2) = M_1);$

аналогично для остальных скобок будем иметь:

$M_1 \cdot q_2 = (int(M_1 \cdot q_2) = b_2) + (DF(M_1 \cdot q_2) = M_2);$

$M_2 \cdot q_2 = (int(M_2 \cdot q_2) = b_3) + (DF(M_2 \cdot q_2) = M_3);$

$M_3 \cdot q_2 = (int(M_3 \cdot q_2) = b_4) + (DF(M_3 \cdot q_2) = M_4);$

.....

$M_{n-2} \cdot q_2 = (int(M_{n-2} \cdot q_2) = b_{n-1}) + (DF(M_{n-2} \cdot q_2) = M_{n-1});$

$M_{n-1} \cdot q_2 = (int(M_{n-1} \cdot q_2) = b_n) + (DF(M_{n-1} \cdot q_2) = M_n);$

$M_n \cdot q_2 = (int(M_n \cdot q_2) = b_{n+1}) + (DF(M_n \cdot q_2) = M_{n+1}).$

Отсюда вытекает следующее правило формирования коэффициентов полинома, которые одновременно являются разрядами записи заданного числа M в системе счисления с основанием q2:

- определяется количество разрядов «n» в записи числа M_{q2} в новой системе счисления (см. подразд.1.2.1);
- исходное число M_{q1} умножается на q2, при этом будет получено смешанное число;
- дробная часть полученного произведения снова умножается на q2 и т.д.;
- процесс умножения повторяется n раз. В качестве искомых разрядов новой записи числа используются результаты выполненных операции деления следующим образом:
- в качестве первого старшего разряда искомой записи числа в новом основании берется значение целой части первого произведения, в качестве второго старшего разряда искомой записи числа в новом основании берется значение целой части второго произведения и т.д.

Пример

Найти запись в двоичной форме десятичного числа M_{10} =0.7.

Решение

Определяем количество разрядов числа M_2. Так как исходная запись числа содержит один десятичный разряд, то запись данного числа в двоичном основании

должна содержать четыре разряда. Учитывая округление, ищем предварительный двоичный эквивалент с пятью разрядами.

Умножаем исходное число M_{10}, а затем дробные части последовательно получаемых произведений на новое основание 2. Выполняется пять таких операций умножения, в результате получаем:

$0.7 \cdot 2 = 1.4$ (int$(0.7 \cdot 2) = 1$ и DF $(0.7 \cdot 2) = 0.4$);
$0.4 \cdot 2 = 0.8$ (int$(0.4 \cdot 2) = 0$ и DF(rest $(0.4 \cdot 2) = 0.8$);
$0.8 \cdot 2 = 1.6$ (int$(0.8 \cdot 2) = 1$ и DF(rest $(0.8 \cdot 2) = 0.6$);
$0.6 \cdot 2 = 1.2$ (int$(0.6 \cdot 2) = 1$ и DF(rest $(0.6 \cdot 2) = 0.2$);
$0.2 \cdot 2 = 0.4$ (int$(0.2 \cdot 2) = 0$ и DF(rest $(0.2 \cdot 2) = 0.4$).

Таким образом, 0.7 = 0.10110, а окончательный результат перехода в двоичную систему будет

$0.7_{10} = 0.1011_2$.

2.3 Метод с использованием особого соотношения оснований заданной и искомой систем счисления

Данный метод применим тогда, когда исходное q_1 и новое q_2 основания могут быть связаны через целую степень, т.е. когда выполняется условие

$q_1^m = q_2$ (*условие 1*) или $q_2^m = q_1$ (*условие 2*).

Если имеет место *условие 2*, то для преобразования заданного числа

$N = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$

запись его в новом основании q_2 ищется следующим образом:

— каждому разряду a_i исходной записи числа ставится в соответствие его m-разрядный эквивалент в системе счисления с основанием q_2;

— исходная запись всего заданного числа формируется за счет объединения всех полученных m- разрядных групп.

Если имеет место *условие 1*, то запись заданного числа

$N = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$

в новом основании q_2 ищется следующим образом:

— исходная запись числа разбивается на группы по m разрядов, двигаясь от точки вправо и влево (недостающие разряды в крайних группах (слева и справа) дополняются нулями;

— каждой полученной группе ставится в соответствие цифра новой системы счисления;

— искомая запись заданного числа в новой системе счисления образуется из цифр, соответствующих группам, на которые была разбита исходная запись.

Пример 1

Найти двоичный эквивалент восьмеричного числа 67401.64_8.

Решение

Основания исходной и новой систем счисления можно выразить через целую степень:

$2^3 = 8$.

Поэтому применяем третий метод для случая перехода из системы с большим основанием в систему с меньшим основанием. Ставим в соответствие каждой цифре исходной записи числа трехразрядный двоичный эквивалент:

6 7 4 0 1 . 6 4
110 111 100 000 001 110 100

Формируем окончательный результат посредством объединения полученных трехразрядных двоичных чисел в единый двоичный эквивалент:

$67401.64_8 = 110111100000001.110100.$

Пример 2

Найти шестнадцатеричный эквивалент двоичного числа

$N = 11100101110110.111011001_2$.

Решение

Основания исходной и новой систем счисления можно выразить через целую степень:

$2^4 = 16$.

Поэтому применяем третий метод для случая перехода из системы с меньшим основанием в систему с большим основанием. Разбиваем исходную запись числа на группы по четыре разряда вправо и влево от точки, в крайних левой и правой группах недостающие разряды заполняем нулями и каждой полученной группе из четырех разрядов ставим в соответствие цифру шестнадцатеричной системы счисления

0011 1001 0111 0110 . 1110 1100 1000
3 9 7 6 E C 8

Формируем окончательный результат посредством объединения полученных цифр в единый шестнадцатеричный эквивалент

$11100101110110 .111011001_2 = 3\ 9\ 7\ 6.\ E\ C\ 8_{16}$.

Пример 3

Найти шестнадцатеричный эквивалент числа 67401.64_8, представленного в восьмеричной системе счисления.

Решение

Основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень, поэтому напрямую третий метод перехода неприменим. Однако существует система с двоичным основанием, для которой допустим третий метод перехода и восьмеричную (исходную для данного примера), и в шестнадцатеричную (новую систему для данного примера) системы счисления, т.к. $2^3 = 8$ и $2^4 = 16$.

Поэтому в данном случае для решения поставленной задачи целесообразно использовать два быстрых перехода из восьмеричной системы счисления в двоичную (промежуточную), а затем из двоичной системы счисления в шестнадцатеричную третьим методом. Это будет гораздо быстрее, чем использовать для заданного преобразования второй или третий метод.

Таким образом, поставленная задача решается в следующем порядке:

$67401.64_8 = 110\ 111\ 100\ 000\ 001\ .\ 110\ 100_2$;

$110\ 111\ 100\ 000\ 001\ .\ 110\ 100_2 = 0110\ 1111\ 0000\ 0001.\ 1101\ 0000_2 =$

$= 6\ F\ 0\ 1.\ D\ 0_{16}$, т.е.:

$67401.\ 64_8 = 6F01.\ D0_{16}$.

Пример 4

Найти двоичный эквивалент числа 6740_{10}.

Решение

Основания исходной q_1 и новой q_2 систем счисления не могут быть связаны через целую степень, поэтому третий метод перехода неприменим. В принципе здесь целесообразно использовать второй метод - метод деления на новое основание. Однако в этом случае потребуется большое количество операций деления на два. Для сокращения количества операций деления может оказаться целесообразным решить эту задачу за счет перехода с использованием второго метода в промежуточную шестнадцатеричную систему счисления, а затем, используя третий метод, быстро перейти в заданную двоичную систему счисления:

– выполняем переход в промежуточную систему счисления:

6740/16 = 421 (остаток 4);

421/16 = 26 (остаток 5);

26/16 = 1 (остаток 10);

– для промежуточной системы счисления имеем:

$6740_{10} = 1A54_{16}$,

– выполняем переход из промежуточной системы счисления в заданную:

$1A54_{16} = 0001\ 1010\ 0101\ 0100_2$.

Как видно из вышеприведенного, для заданного перехода потребовалось выполнить только 4 операции деления на 16 вместо 13-ти операций деления на 2.

3 Арифметические операции над положительными числами

3.1 Операции сложения в двоичной системе счисления

При выполнении любой операции результат ищется согласно соответствующим правилам, которые удобно представлять в табличной форме, где для всех возможных комбинаций значений одноразрядных операндов приводятся значения результата.

Правила выполнения операции сложения в двоичной системе счисления задаются в виде, приведенном в таблице 2.3.

Таблица 2.3 – Операция сложения

+	0	1
0	0	1
1	1	0*

Все возможные значения первого слагаемого задаются во второй и третьей строках первой колонки. Все возможные значения второго слагаемого задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат их сложения. В

таблице знаком * отмечен случай, когда в текущем разряде результата получен ноль и имеет место перенос в ближайший старший разряд.

Пример

```
  1011100110
+ 0101101101
 10001010011
```

В общем случае при формировании значения в текущем разряде результата приходится дважды применять приведенную таблицу сложения: первый раз при сложении соответствующих разрядов операндов, формируя так называемую поразрядную сумму, и второй раз - при сложении разряда сформированной поразрядной суммы и переноса, пришедшего из ближайшего младшего разряда.

При машинной реализации операции сложения сначала формируется поразрядная сумма операндов без учета переноса, далее формируется код переноса, и затем с помощью специальных логических цепей учитываются возникшие переносы. При этом перенос, возникший в некотором разряде, может изменить не только ближайший старший разряд, но и целую группу старших разрядов. В худшем случае перенос, возникший в самом младшем разряде, может изменить значение старших разрядов сформированной поразрядной суммы, вплоть до самого старшего.

При формировании поразрядной суммы и учете возникших переносов используется следующая классификация разрядов складываемых операндов:

- разряд, генерирующий перенос (оба операнда в этом разряде имеют 1);
- разряд, пропускающий перенос (операнды в этом разряде имеют разные значения);
- разряд, блокирующий распространение переноса (операнды в этом разряде имеют одинаковые значения).

3.2 Операция вычитания

Правила выполнения операции сложения в двоичной системе счисления задаются в виде, приведённом в таблице 2.4.

Таблица 2.4 – Операция вычитания

-	0	1
0	0	1
1	1*	0

Все возможные значения вычитаемого задаются во второй и третьей строках первой колонки. Все возможные значения уменьшаемого задаются во второй и третьей колонках
первой строки. На пересечении отмеченных значениями операндов строк и

колонок располагается результат вычитания второго операнда из первого операнда. В таблице знаком * отмечен случай, когда в текущем разряде результата получена единица при заёме из ближайшего старшего разряда.

Пример

1000111001
- 0101101101
0011001100

Как видно из таблицы и приведенного примера, реализация операция вычитания не сложнее операции сложения.

В ЭВМ никогда в перечне выполняемых операций арифметического устройства не присутствует одновременно операция сложения и операция вычитания. При этом, как правило, присутствует только операция сложения. Что же касается операции вычитания, то она реализуется за счет прибавления к уменьшаемому значения вычитаемого, взятого с противоположным знаком.

3.3 Операция умножения

Умножение в двоичной системе счисления задаются в виде, приведенном в таблице 2.5.

Таблица 2.5 – Операция умножения

*	0	1
0	0	0
1	0	1

Все возможные значения множимого задаются во второй и третьей строках первой колонки. Все возможные значения множителя задаются во второй и третьей колонках первой строки. На пересечении отмеченных значениями операндов строк и колонок располагается результат умножения первого операнда на второй операнд.

При умножении многоразрядных операндов, как правило, (особенно в десятичной системе счисления) используется метод, при котором формирование произведения выполняется за счет суммирования частичных произведений, которые формируются посредством умножения множимого на отдельные разряды множителя с учетом веса соответствующего разряда множителя.

Таблица умножения одноразрядных операндов в двоичной системе существенно упрощает задачу формирования частичного произведения:

- частичное произведение для разряда множителя равняется нулю, если этот разряд равен нулю;

- частичное произведение для разряда множителя равняется множимому, взятому с соответствующим весом, если разряд множителя равен единице.

При последовательном способе формирования частичных произведений, последние могут рассчитываться поочередно для отдельных разрядов

множителя, начиная с младшего или старшего разряда. При десятичном основании, как правило, формирование частичных произведений осуществляется, начиная с младшего разряда множителя.

Пример 1

Найти произведение двоичных чисел 1011 и 1101, начиная формирование частичных произведений со старшего разряда множителя.

Решение

При формировании частичных произведений, начиная со старшего разряда множителя, процесс формирования произведения заданных операндов можно представить следующим образом:

```
  * 1 0 1 1
    1 1 0 1
  + 1 0 1 1
    + 1 0 1 1
      + 0 0 0 0
        + 1 0 1 1
  ---------------
  1 0 0 0 1 1 1 1
```

При реализации умножения рассматриваемым способом требуется использование 2n-разрядного сумматора для подсчета промежуточных и конечного произведения и 2n-разрядного сдвигающего регистра для хранения множителя.

Пример 2

Найти произведение двоичных чисел 1011 и 1101, начиная формирование частичных произведений с младшего разряда множителя

Решение

```
    * 1 0 1 1
      1 1 0 1
    + 1 0 1 1
   + 0 0 0 0
  + 1 0 1 1
+ 1 0 1 1
-------------
 1 0 0 0 1 1 1 1
```

Операция умножения в общем случае дает точный результат – 2n-разрядное произведение, где n - разрядность операндов.

В ЭВМ при выполнении различных операций, в том числе и операции умножения, разрядность операндов и результатов одинаковая. Это означает, что при умножении правильных дробей последние, младшие, n разрядов отбрасываются, а старшие n разрядов округляются с учетом отбрасываемых младших разрядов.

Например, произведение

$0.1011 \cdot 0.1101 = 0.10001111$

представляется в n-разрядном варианте как 0.1001.

В этом случае операция умножения считается приближенной.

Иногда в ЭВМ формируется точное произведение; в таких случаях специально оговаривается, что произведение формируется с удвоенной разрядной сеткой.

При операндах, представленных в виде целых чисел, формирование произведения с такой же разрядностью, что и разрядность операндов, в качестве конечного значения произведения также используются n старших разрядов, которые округляются с учетом отбрасываемых младших разрядов, но при этом масштаб представления произведения изменяется - он умножается на 2^n . Это является одной из причин, по которой в ЭВМ, как правило, используется представление чисел в виде правильных дробей.

3.4 Деление двоичных чисел

Деление в принципе является неточной операцией, поэтому при её выполнении прежде всего устанавливается количество разрядов частного, которые подлежат определению.

Деление в двоичной системе счисления может выполняться точно так же, как и в десятичной, однако формирования частного двоичных операндов реализуется гораздо проще, чем в десятичной системе, т.к.:

– упрощается процедура подбора очередной цифры вследствие того, что в двоичной системе очередной цифрой может быть одна из двух - либо 0, либо 1;

– упрощается процедура умножения найденной цифры частного на делитель.

Более детально операция деления будет рассмотрена далее.

3.5 Арифметика с положительными двоично-десятичными числами

В ЭВМ часто предусматривается обработка чисел не только в двоичной системе счисления, но в двоично-десятичной. При этом, как правило, стремятся реализовать двоично-десятичную арифметику по правилам двоичной с введением ограниченного количества коррекций. В качестве примера рассмотрим операцию сложения двоично-десятичных чисел.

Пример

Найти сумму двух десятичных чисел с использованием двоично-десятичной системы счисления:

A = D + C, где

D = 3927;

C = 4856.

Решение

Составляем двоично-десятичную запись для чисел D и C:

D =3927 = 0011 1001 0010 0111;

C = 4856 = 0100 1000 0101 0110.

Найти значение A можно, реализовав следующую последовательность операций из двоичного сложения и операции коррекции:

```
                *           **
  0011 1001 0010 0111 - D
+ 0100 1000 0101 0110 - C
  1000 0001 0111 1101 - двоичная сумма
+      0110      0110 - коррекция
  1000 0111 1000 0011 - двоично-десятичная сумма
```

Для получения двоично-десятичной суммы А на основании результата сложения операндов по правилам двоичной арифметики необходимо добавить шестерку (0110) в те тетрады, из которых был перенос. В данном примере это вторая тетрада (отмечена *). Необходимость такой коррекции обусловливается тем, что перенос, сформированный по правилам двоичного суммирования, унес из тетрады шестнадцать, а для десятичного сложения перенос должен был унести десять, т.е. перенос, сформированный по правилам двоичной арифметики, унес лишнюю шестерку. Кроме этого, шестерка добавляется в те тетрады, в которых получено значение, большее девяти.

Такая коррекция обусловливается тем, что по правилам десятичной арифметики в таких тетрадах должен быть выработан перенос и, чтобы его выработать по правилам двоичной арифметики, в тетраду нужно добавить шестерку. Для рассмотренного примера такой тетрадой является и четвертая тетрада (отмечена **).

4 Арифметика с алгебраическими числами

4.1 Кодирование алгебраических чисел

Для представления чисел со знаком используются специальные коды:

- *прямой код*;
- *дополнительный код*;
- *обратный код*.

Во всех трёх случаях используется следующий формат представления числа, содержащий два поля - поле знака и поле модуля (рисунок 2.1).

Рисунок 2.1 – Формат представления алгебраического числа

Поле знака представлено одним разрядом, в котором устанавливается 0, если число положительное, и 1, если число отрицательное.

Поле модуля отражает количественную оценку числа и для каждого кода формируется по-разному. Количество разрядов поля модуля определятся диапазоном изменения отображаемых чисел или точностью их представления.

В *прямой код* запись целого числа А формируется по следующему правилу:

$$[A]_{пк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1.\ |A|, \text{ если } A < 0. \end{cases}$$

В дополнительном коде запись целого числа А формируется по следующему правилу:

$$[A]_{дк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1\ .\ q^n + A\ , \text{ если } A < 0, \end{cases}$$

где n - разрядность модульного поля;

q - основание системы счисления;

q^n - максимальная невключенная граница диапазона изменения представляемых чисел, т.к. диапазон изменение чисел А определяется как

$$q^n > |A| >= 0\ .$$

Для случая правильной дроби запись числа В в дополнительном коде имеет вид

$$[A]_{дк} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1.\ (1 + A\ ), \text{ если } A < 0, \end{cases}$$

где 1 - максимальная невключенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$1 > |A| >= 0.$$

В обратном коде запись целого числа А формируется по следующему правилу:

$$[A]_{ок} = \begin{cases} 0.\ A, \text{ если } A >= 0; \\ 1\ .\ ((q^n - 1) + A)\ , \text{ если } A < 0, \end{cases}$$

где n - разрядность модульного поля;

q - основание системы счисления;

$(2^n - 1)$ - максимальная включенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$(q^n - 1) >= |A| >= 0.$$

Для случая правильной дроби запись числа В в обратном коде имеет вид

$$[A]_{ок} = \begin{cases} 0.\ B, \text{ если } B >= 0; \\ 1.\ (1 - q^{-n} + B), \text{ если } B < 0, \end{cases}$$

где $(1 - q^{-n})$ - максимальная включенная граница диапазона изменения представляемых чисел, т.е. диапазон изменение чисел А определяется как

$$(1 - q^{-n}) >= |A| >= 0.$$

Легко показать, что перевод отрицательного числа из обратного или дополнительного кода в прямой выполняется по тому же правилу, что и перевод

числа из прямого кода в обратный или *дополнительный код*:

– для того чтобы перевести отрицательное число из обратного в *прямой код*, необходимо дополнить его модуль до включенной границы;

– для того чтобы перевести отрицательное число из дополнительного в *прямой код*, необходимо дополнить его модуль до невключенной границы.

4.2 Дополнительный и обратный коды двоичных чисел

При переводе двоичных чисел в качестве включенной и невключенной границы диапазона изменения абсолютных значений представляемых чисел используется соответственно 2^n и 2^n -1.

Представление двоичных чисел в прямом и обратном кодах поясняется следующими примерами.

Пример

Найти запись чисел A= 532 и B = -150 в прямом, дополнительном и обратном двоичных кодах.

Решение

Найдем запись заданных чисел в двоичной системе:

$A = 532_{10} = 1000010100_2$, $B = -150_{10} = -10010110_2$.

Если считать, что представляются в заданных кодах только A и B, то разрядность n - модульного поля должна соответствовать разрядности двоичной записи большего числа, т.е. n=10.

Найдем запись заданных чисел в прямом коде:

$[A]_{пр} = 0.1000010100$, $[B]_{пр} = 1.0010010110$.

Найдем запись заданных чисел в дополнительном коде:

$[A]_д = 0.1000010100$,

$[B]_д$: для определения модульной части прибавим к невключенной границе диапазона ($2^n = 10000000000$) число B:

10000000000 + (-10010110) = 1101101010

и тогда $[B]_д = 1.1101101010$.

Найдем запись заданных чисел в обратном коде:

$[A]_о = 0.1000010100$,

$[B]_о$: для определения модульной части прибавим к включенной границе диапазона ($2^n – 1 = 1111111111$) число B:

1111111111 +(-10010110) = 1101101001

и тогда $[B]_о = 1.1101101001$.

Анализируя запись модульной части отрицательного числа C в обратном коде, можно заключить, что она представляет собой инверсию модульной части записи этого числа в прямом коде, т.е. 0 заменяются 1, а 1 заменяются 0. Отсюда вытекает правило формирования модуля обратного кода отрицательного двоичного числа:

чтобы сформировать модульную часть записи отрицательного числа в обратном коде, достаточно в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов, т.е. необходимо проинвертировать модуль прямого кода.

Переход от обратного кода отрицательного числа к представлению в прямом коде осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде.

Если сравнить запись модульных частей дополнительного и обратного кодов отрицательного числа В, то можно заметить, что они отличаются на значение, соответствующее единицы младшего разряда. Отсюда вытекает правило формирования модуля дополнительного кода отрицательного числа:

чтобы сформировать модульную часть записи отрицательного числа в дополнительном коде, достаточно в модульной части записи этого числа в прямом коде взять обратные значения всех двоичных разрядов, т.е. необходимо проинвертировать модуль прямого кода, и к полученному коду прибавить 1 в млаший разряд.

Переход от дополнительного кода отрицательного числа к прямому осуществляется по тому же правилу, т.е. необходимо проинвертировать модуль записи числа в дополнительном коде, и к полученному коду прибавить 1 в младший разряд.

При выполнении операций над числами со знаком в ЭВМ используются прямой, обратный и дополнительный коды. Как правило, информация в памяти хранится в прямом коде, а при выполнении операций применяется или обратный, или дополнительный код.

4.3 Операции с двоичными числами в дополнительном коде

При использовании *дополнительного* или *обратного* кода операция вычитания заменяется операцией сложения с изменением знака второго операнда. При сложении чисел, представленных в дополнительном коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимание на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возникший из крайнего левого разряда, игнорируется. В результате такого сложения будет получен *дополнительный код* суммы заданных операндов.

Пример

Найти значения для С1, С2, С3, С4, определяемых выражениями:

С1 = А+В, С2 = А-В, С3 = В-А, С4 = -А-В,

если А = 57_{10}, В = - 210_{10}. При выполнении операций использовать двоичный *дополнительный* код. Результат представить в *прямом* коде.

Решение

Преобразуем заданные числа в двоичную систему счисления:

А = 57_{10} ,= 111001_2, В = - 210_{10} = -11010010_2.

Определим количество разрядов для модульной части записи чисел, учитывая не только значения используемых операндов, но и ожидаемые результаты выполнения заданных операций. Исходя из абсолютного значения операндов разрядность представления модульной части n должна быть равна 8. Учитывая то, что в подлежащих реализации выражениях над числами выполняется только одна операция (или сложения, или вычитания), длину модульной части необходимо

взять на один разряд больше, т.е. n = 9.

Избавляясь от операции вычитания, приводим заданные выражения к виду

C1= A+B, C2 = A+(-B), C3 = B+(-A), C4 = (-A) + (-B).

Таким образом, в подлежащих реализации выражениях в качестве операндов присутствуют следующие величины: A , -A, B, -B. Представим их в прямом и дополнительном кодах:

$[A]_{пр}$ = 0.000111001, $[A]_{дк}$ = 0.000111001,
$[-A]_{пр}$ = 1.000111001, $[-A]_{дк}$ = 1.111000111,
$[B]_{пр}$ = 1.011010010, $[B]_{дк}$ = 1.100101110,
$[-B]_{пр}$ = 0.011010010, $[-B]_{дк}$ = 0.011010010.

Используя сформированный дополнительный код, реализуем выражения для C1,C2, C3, C4.

C1: 0.0 0 0 1 1 1 0 0 1 - $[A]_{дк}$
+ 1.1 0 0 1 0 1 1 1 0 - $[B]_{дк}$
1.1 0 1 1 0 0 1 1 1 - $[C1]_{дк}$
1.0 1 0 0 1 1 0 0 1 - $[C1]_{пк}$

C2: 0.0 0 0 1 1 1 0 0 1 - $[A]_{дк}$
+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{дк}$
0. 1 0 0 0 0 1 0 1 1 - $[C2]_{дк}$ =$[C2]_{пк}$

C3: 1.1 1 1 0 0 0 1 1 1 - $[-A]_{дк}$
+ 1.1 0 0 1 0 1 1 1 0 - $[B]_{дк}$
11.0 1 1 1 1 0 1 0 1 - $[C3]_{дк}$
1.1 0 0 0 0 1 0 1 1 - $[C3]_{пк}$.

При выполнении сложения в данном случае возникла единичка переполнения знакового поля. При работе с дополнительным кодом она игнорируется (в примере она перечеркнута).

C4: 1.1 1 1 0 0 0 1 1 1 - $[-A]_{дк}$
+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{дк}$
~~1~~0.0 1 0 0 1 1 0 0 1 - $[C4]_{дк}$ =$[C4]_{пк}$

В данном случае также возникло переполнение знакового поля, которое игнорируется.

4.4 Операции с двоичными числами в обратном коде

При сложении чисел, представленных в *обратном* коде, выполняется сложение разрядов, представляющих запись операндов, по правилам двоичной арифметики по всей длине записи чисел, не обращая внимания на границу, разделяющую знаковое и модульные поля. Переполнение знакового поля, т.е. перенос, возникший из крайнего левого разряда, должен быть учтен как +1 в младший разряд полученной суммы. В результате такого сложения будет получен *обратный* код суммы заданных операндов.

Пример

Найти значения для C1, C2, C3, C4, определяемых выражениями

C1 = A+B, C2 = A-B, C3 = B-A, C4 =-A-B,

если $A = 57_{10}$, $B = -210_{10}$. При выполнении операций использовать двоичный

обратный код. Результат представить в прямом коде.

Решение

В данном примере используются те же выражения и те же операнды, что и в предыдущем примере, поэтому при его решении используются уже найденные ранее двоичные представления операндов и их прямые коды.

Обратные коды операндов имеют вид

$[A]_{ок}$ = 0.000111001, $[-A]_{ок}$ = 1.111000110,

$[B]_{ок}$ = 1.100101101, $[-B]_{ок}$ = 0.011010010.

Используя сформированный дополнительный код, реализуем выражения для C1, C2, C3, C4.

C1: 0.0 0 0 1 1 1 0 0 1 - $[A]_{ок}$

+ 1.1 0 0 1 0 1 1 0 1 - $[B]_{ок}$

1.1 0 1 1 0 0 1 1 0 - $[C1]_{ок}$

1.0 1 0 0 1 1 0 0 1 - $[C1]_{пк}$

C2: 0.0 0 0 1 1 1 0 0 1 - $[A]_{ок}$

+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{ок}$

0.1 0 0 0 0 1 0 1 1 - $[C2]_{ок} = [C2]_{пк}$

C3: 1.1 1 1 0 0 0 1 1 0 - $[-A]_{ок}$

+ 1.1 0 0 1 0 1 1 0 1 - $[B]_{ок}$

1 1.0 1 1 1 1 0 0 1 1

\+ 1

1. 0 1 1 1 1 0 1 0 0 - $[C3]_{ок.}$

1. 1 0 0 0 0 1 0 1 1 - $[C3]_{пк}$

C4: 1.1 1 1 0 0 0 1 1 0 - $[-A]_{ок}$

+ 0.0 1 1 0 1 0 0 1 0 - $[-B]_{ок}$

10.0 1 0 01 1 0 0 0

\+ 1

0. 0 1 0 01 1 0 0 1 - $[C4]_{ок} = [C4]_{пк}$

В данном случае также возникло переполнение знакового разряда, которое должно быть учтено как +1 в младший разряд сформированной суммы.

4.5 Модифицированные коды

При расчете разрядности n модульного поля весьма трудно бывает учесть диапазон значений результатов, особенно когда последовательность операции, представленных в подлежащих реализации выражениях, достаточно сложны.

При несоответствии выбранной разрядности n диапазону изменения представляемых чисел при выполнении операции сложении чисел с одинаковыми знаками возможно появление ситуации переполнения, когда подлежащий представлению результат выходит за диапазон представления, определенный некорректно выбранной разрядностью n поля модуля.

Например, в случае сложения двух чисел, представленных в обратном коде:

$[D1]_{ок}$ = 1.00110 и $[D2]_{ок}$ =1.00110

сумма этих чисел

F1 = D1+ D2

будет подсчитана следующим образом:

F1:

```
  1. 0 0 1 1 0
 +1. 0 0 1 1 0
 10 .0 1 1 0 0
+             1
  0. 0 1 1 0 1
```

Пример, выполненный по всем формальным правилам, дал абсурдный результат, так как получена положительная сумма двух отрицательных операндов. Аналогичная ситуация может возникать и при использовании дополнительного кода.

Ситуацию переполнения можно обнаруживать по факту появления «абсурдного» результата, но для этого необходимо помнить то, что в суммировании принимают участие операнды с одинаковыми знаками и знак полученного при этом результата отличен от знака операндов.

Более просто ситуация переполнения определяется при применении *модифицированного* кода (обратного или дополнительного). Модифицированные коды отличаются от базовых кодов только тем, что поле знака операндов имеет два разряда, и эти разряды имеют одинаковые значения:

0 0 - для положительных чисел;

1 1 - для отрицательных чисел.

Если в результате сложения чисел в модифицированном коде полученный результат имеет в поле знака одинаковые значения в обоих разрядах (00 или 11), то переполнения нет, если же разряды знакового поля имеют неодинаковые значения (10 или 01), то имеет место переполнение. При этом, если в поле знака имеет место значение 01 - результат положительный, а если 10, то полученный результат отрицательный (основным носителем знака числа является левый разряд знакового поля).

Пример

Найти значения выражений

C1= A+B, C2=A-B, C3= B-A, C4=-A-B,

используя *модифицированный* обратный код если

$[A]_{пк} = 0.\ 1010011,$

$[B]_{пк} = 1.\ 0111001.$

Решение

Модифицированный обратный код для всех операндов, используемых в приведенных выражениях, имеет вид

$[A]_{мпк} = 0\ 0.\ 1010011,\ [A]_{мок} = 0\ 0.\ 1010011,\ [-A]_{мок} = 1\ 1.\ 0101100,$

$[B]_{мпк} = 1\ 1.\ 0111001,\ [B]_{мок} = 1\ 1.\ 1000110,\ [-B]_{мок} = 0\ 0.\ 0111001.$

Выполним действия, указанные в приведенных выражениях:

С1:

00. 1010011 - $[A]_{мок}$
+11. 1000110 - $[B]_{пок}$
100 .0011001
\+ 1
00. 0011010 - $[C1]_{мок}=[C1]_{мпк}$

С2:

00. 1010011 - $[A]_{мок}$
+00. 0111001 - $[-B]_{пок}$
01. 0001100

Результат положительный и имеет место переполнение.

С3:

11. 0101100 - $[-A]_{мок}$
+ 11. 1000110 - $[B]_{пок}$
110. 1110010
\+ 1
10. 1110011

Результат отрицательный и имеет место переполнение.

С4:

11. 0101100 - $[-A]_{мок}$
+00. 0111001 - $[-B]_{пок}$
11. 1100101 - $[C4]_{мок}$
11. 0011010 - $[C1]_{мпк}$

При формировании С1 был получен положительный результат (без переполнения).

При формировании С4 был получен отрицательный результат (без переполнения).

Факт переполнения при формировании С3 и С2 устанавливается по наличию в разрядах знакового поля различных значений.

5 Логические операции с двоичными кодами

Над двоичными кодами могут выполняться различные логические операции, среди которых особое место занимают:

– *логическое суммирование* (обозначения - ИЛИ, OR, «∨», «+»);

– *логическое умножение* (обозначения - И, AND, «∧», «*»);

– *логическое отрицание* (обозначения – НЕТ, NOT, «$\overline{x}$», т. е. штрих над отрицаемым кодом x);

– *суммирование по модулю* 2 (обозначается mod 2, « ⊕ »);

– *операции сдвига*.

Операция *логического суммирования* выполняется над двумя кодами и

генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если хотя бы в одном операнде в i-м разряде имеет место единица.

Пример

10001101 $\vee$ 11110000 = 11111101.

Операция *логического умножения* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если оба операнда в этом i-м разряде имеются единицу, и ноль во всех других случаях.

Пример

10001101 $\wedge$ 11110000 = 10000000

Операция *суммирования по модулю 2* выполняется над двумя кодами и генерирует код той же разрядности, что и операнды, у которого в некотором i-м разряде находится единица, если два заданных операнда в i-м разряде имеется противоположные значения. Иногда эта операция называется «исключающее ИЛИ».

Пример

10001101 $\oplus$ 11110000 = 01111101.

Операция логического отрицания выполняется над одним кодом и генерирует результирующий код той же разрядности, что и операнд, у которого в некотором i-м разряде находится значение, противоположное значению в i-м разряде отрицаемого кода.

Операции *сдвига* в свою очередь, подразделяются на:

– логические сдвиги, которые имеют разновидности - сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево;

– арифметический сдвиг вправо и в лево, выполнение которого зависит от знака и кода сдвигаемого числа.

Логические сдвиги.

Сдвиг *влево* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (освобождающийся самый правый т.е. самый младший, разряд заполняется 0, а «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 10011100.

Сдвиг *вправо* выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый, т.е. самый старший, разряд заполняется 0, «выталкиваемый» разряд пропадает). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

Циклический сдвиг влево - выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (в освобождающийся самый правый, т.е. самый младший, разряд заносится значение старшего, т.е. самого левого разряда исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 10011101.

Циклический сдвиг вправо - выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый т.е. самый старший, разряд заполняется значение в самом младшем разряде исходного кода). Например, код 11001110 после сдвига влево будет иметь вид 01100111.

6 Представление чисел с фиксированной точкой

Числовая информация представляется в машине в форме с фиксированной или с плавающей точкой. При представлении с фиксированной точкой положение последней в записи числа фиксировано.

Как правило, при использовании фиксированной точки числа представляются в виде целого числа или правильной дроби, форматы которых приведены на рисунке 2.3.

зн.	1р.	2р.	3р.	4р.	...	(n-1) р.	np.	«.»

а)

зн.	«.»	1р.	2р.	3р.	4р.	...	(n-1) р.	np.

б)

Рисунок 2.3 – а) – формат целого числа; б) – формат дробного числа

К заданному виду (целым числам или правильной дроби) исходные числа приводятся за счет введения масштабных коэффициентов.

Точка в записи числа не отображается, а так как она находится всегда в одном месте, то указание на её положение в записи числа отсутствует. При n-разрядном представлении модульной части форма с фиксированной точкой обеспечивает диапазон изменения абсолютного значения числа А, для которого выполняется неравенство

$2^{n.} > |A| > = 0.$

Одним из важнейших параметров представления чисел является ошибка представления. Ошибка представления может быть абсолютной (Δ) или относительной (δ). Для фиксированной точки максимальные значения этих ошибок определяются следующим образом.

В случае целых чисел:

$\Delta_{max} = 0.5$; $\delta_{max} = \Delta_{max} / A_{min} = 0.5$, где A_{min} - минимальное, отличное от нуля, значение числа.

В случае дробных чисел:

$\Delta_{max} = 0.5 \cdot 2^{-n} = 2^{-(n+1)}$; $\delta_{max} = \Delta_{max} / A_{min} = 2^{-(n+1)} / 2^{-n} = 0.5$,

т.е. в худшем случае относительная ошибка при фиксированной точке может достигать сравнительно большого значения - 50%.

6.1 Арифметические операции над числами, представленными с фиксированной точкой

К числу основных арифметических операций, непосредственно реализуемых в ЭВМ, относятся операции сложения, умножения, деления. Остальные операции (например, такие, как возведение в степень, извлечение квадратного корня) реализуются программным способом.

Выполнение операций с числами, представленными с фиксированной точкой, рассмотрено в рамках материала по выполнению операций с алгебраическими числами.

Выполнение длинных операций, таких, как умножение и деление, реализуется в два этапа:

– на первом этапе формируется знак искомого результата,

– на втором этапе, используя абсолютные значения операндов, ищем результат (произведение или частное), которому затем присваивается предварительно определенный знак.

Операнды, как правило, представлены в прямом коде, и знак результата, не зависимо от того, частное это или произведение, ищется за счет сложения по модулю 2 знаковых разрядов операндов. В результате этого знак результата положителен, если операнды имеют одинаковые знаки, или отрицательный, если операнды имеют разные знаки.

Выполнение второго этапа, т.е. умножение положительных чисел достаточно подробно изложено ранее.

6.2 Деление с фиксированной точкой

Реализация второго этапа деления, т. е. формирования частного двоичных положительных чисел в ЭВМ, представленных правильной дробью, может быть выполнена двумя методами:

– деление с восстановлением остатка;

– деление без восстановления остатка.

Метод деления с восстановлением остатка.

Деление выполняется потактно.

На каждом такте определяется один разряд частного. В процессе деления определяются разряд целой части (этот разряд имеет единичное значение, если деление двух правильных дробей дает частное, большее или равное 1), и n разрядов дробной части частного, где n-разрядность поля модуля представления чисел. Для того чтобы точность ошибка результата не превышала половины младшего разряда модуля, ищется дополнительный (n +1)-й разряд дробной части частного, который используется только для округления результата. Таким образом, деление выполняется за (n +2) такта.

На каждом такте выполняются следующие действия:

– из остатка, полученного на предыдущем такте (на первом такте из делимого), вычитается делитель (выполняется пробное вычитание), тем самым формируется новый остаток;

– анализируется знак нового остатка и, если знак отрицательный, то

осуществляется восстановление остатка, т.е. к полученному новому остатку прибавляется делитель (если знак положительный, то восстановление остатка не происходит);

– если после пробного вычитания был получен положительный результат, то в очередном разряде формируемого частного устанавливается единица;

– выполняется умножение на 2 (арифметический сдвиг влево) нового или восстановленного остатка.

На первом такте определяется разряд целой части искомого частного. Для правильной дроби этот разряд должен иметь нулевое значение, поэтому, если на первом такте будет установлено, что первый разряд, т.е. разряд целой части искомого частного, равен единице, то вырабатывается специальный сигнал о том, что искомое частное не является правильной дробью. После выполнения последнего (n
+2)-го такта анализируется последний найденный разряд частного и, если он равен единице, то в n-й разряд частного прибавляется единица.

Пример 1

Определить частное C1 от деления числа A на B, где

$[A]_{пк} = 0.1011,$

$[B]_{пк} = 1.1101.$

При выполнении операций использовать дополнительный код.

Решение

Знак искомого частного C1{зн} = 1, так как сумма по модулю «2» знаковых разрядов операндов равна 1 (A{зн} = 0, а B{зн} = 1. Определим абсолютное значение для C. В процессе поиска значений разрядов частного будут использованы числа $|A|$, $|B|$, $(-|B|)$, представление которых в модифицированном дополнительном коде имеет вид:

$[|A|]_{мдк} = 00.1011$

$[|B|]_{мдк} = 00.1101$

$[-|B|]_{мдк} = 11.0011.$

Процесс выполнения отдельных тактов представляется следующим образом, представленным в таблице 1.6.

Сформированное из последовательности найденных на отдельных тактах разрядов абсолютное значение частного будет равно

$|C| = 0.11011,$

после округления будет иметь место

$|C| = 0.1110.$

С учетом ранее полученного знака окончательный результат равен

$[C]_{пк} = 1.1110.$

Таблица 2.6 – Процесс выполнения отдельных тактов деления

Определяемый разряда частного	Выполняемые действия	Пояснение действий	Значение очередного разряда частного
1 р. (разряд целой части) - 1-й такт	00.1011 +11.0011 11.1110 +00.1101 00.1011 01.0110	Вычитание из абсолютного значения делимого абсолютное значение делителя; остаток <0 Восстановленный остаток Сдвинутый остаток	0
2 р. (старший разряд моду льной части)- 2-й такт	+11.0011 100.1001 00.1001 01.0010	Остаток >0 (перенос игнорируется) Сдвинутый остаток	1
3 р. - 3-й такт	+11.0011 100.0101 00.1010	Остаток >0 (перенос игнорируется) сдвинутый остаток	1
4 р. - 4-й такт	+11.0011 11.1101 +00.1101	Остаток <0 Восстановление остатка	0
	00.1010 01.0100	Сдвинутый остаток	
5 р. - 5-й такт	+11.0011 100.0111 00.1110	Остаток >0 Сдвинутый остаток	1
6 р. - 6-й такт	+11.0011 100.0001 00.0010	Остаток>0 Сдвинутый остаток	1

Метод деления без восстановления остатка.

Идея метода деления без восстановления остатка основана на следующем.

Действия на i-м и на (i +1)-м тактах зависят от знака получаемого остатка и представляются следующим образом:

$(O_{i-1} - D)\ 2 - D$, если $(O_{i-1} - D) >= 0$;

$((O_{i-1} - D) + D)2 - D = (O_{i-1} - D) + 2D - D = (O_{i-1} - D) + D$, если $(O_{i-1} - D) < 0$,

где O_{i-1} - остаток на (i-1)-м такте;

D - абсолютное значение делителя.

Отсюда следует, что в случае, если остаток на предыдущем такте был отрицательным, то можно не осуществлять операцию его восстановления, но на следующем такте нужно не вычитать, а прибавлять делитель.

Исходя из этого деление без восстановления остатка выполняется следующим образом.

1. Число тактов при рассматриваемом методе определяется точно так же, как и

при делении с восстановлением остатка.

2. На каждом такте выполняются следующие действия:

– анализируется знак остатка (на первом такте анализируется знак делимого) и, если знак положительный, то из остатка вычитается делитель, в противном случае делитель прибавляется; таким образом формируется новый остаток;

– если новый остаток положительный, то в очередном разряде формируемого частного устанавливается единица;

– выполняется умножение на два нового остатка.

3. Разряд, определенный на первом такте, также как и в предыдущем методе, является разрядом целой части и, если он ненулевой, то вырабатывается сигнал о нарушении формы представления чисел в виде правильной дроби.

4. После выполнения последнего (n + 2)-го такта выполняется округление.

Пример 2

Определить частное от деления числа А на В, где

$[A]_{пк} = 0.1011$,

$[B]_{пк} = 1.1101$.

При выполнении операций использовать дополнительный код.

Решение

Знак искомого частного С1{зн} = 1, так как сумма по модулю 2 знаковых разрядов операндов равна 1 (А{зн} = 0, а В{зн} = 1).

Определим абсолютное значение для С.

В процессе поиска значений разрядов частного будут использованы числа $|A|$, $|B|$, $(-|B|)$, представление которых в модифицированном дополнительном коде имеет вид: $[|A|]_{мдк} = 00.1011$, $[|B|]_{мдк} = 00.1101$, $[-|B|]_{мдк} = 11.0011$.

Процесс выполнения отдельных тактов представляется следующим образом, представленным в таблице 2.7.

Таблица 2.7 – Процесс выполнения отдельных тактов деления

Определяемый разряда частного	Выполняемые действия	Пояснение действий	Значение очередного разряда частного
1	2	3	4
1 р. (разряд целой части) -1-й такт	00.1011 +11.0011 11.1110 11.1100	Вычитание из абсолютного значения делимого абсолютное значение делителя остаток > 0 (перенос игнорируется) сдвинутый остаток	0
2 р. (старший разряд модуля) - 2-й такт	+00.1101 100.1001 01.0010	Прибавление $\|B\|$, т.к. остаток <0 Остаток >0 (перенос игнорируется) Сдвинутый остаток	1

1	2	3	4
3 р. - 3-й такт	+11.0011 100.0101 00.1010	Вычитание |B|, т.к. остаток >0 Остаток >0 (перенос игнорируется) Сдвинутый остаток	1
4 р.- 4-й такт	+11.0011 11.1101 11.1010	Вычитание |B|, т.к. остаток >0 Новый остаток <0 Сдвинутый остаток	0
5 р. - 5-й такт	+00.1101 100.0111 00.1110	Прибавление |B|, т.к. остаток<0 Новый остаток >0 Сдвинутый остаток	1
6 р. - 6-й такт	+11.0011 100.0001 00.0010	Вычитание |B|, т.к. остаток >0 Новый остаток >0 Сдвинутый остаток	1

Сформированное из последовательности найденных на отдельных тактах разрядов абсолютное значение частного будет равно

$|C| = 0.11011$,

после округления будет иметь место

$|C| = 0.1110$.

С учетом ранее полученного знака окончательный результат равен

$[C]_{пк} = 1.1110$.

7 Представление чисел с плавающей точкой

При представлении числа с плавающей точкой число в общем случае представляет собой смешанную дробь. Местоположение точки в записи числа может быть различным, а так как сама точка в записи числа не присутствует, то для однозначного задания числа необходима не только его запись, но и информация о том, где в записи числа располагается точка, отделяющая целую и дробную части.

Поэтому в случае с плавающей точкой число X представляется в виде двух частей:

- *мантисса* ($x_м$), отображающая запись числа, представляется в виде правильной дроби с форматом фиксированной точки;

- *порядок* ($x_п$), отображающий местоположения в этой записи положение точки, представляется в виде целого числа с форматом фиксированной точки.

Количественная оценка числа X определяется как

$X = q^{x_п} \cdot x_{м.}$,

где q - основание счисления.

Для двоичной системы счисления имеет место

$X = 2^{x_п} \cdot x_{м.}$

При s- разрядном представлении модуля записи мантиссы и k-разрядном представлении модуля записи порядка форма с плавающей точкой обеспечивает диапазон изменения абсолютного значения числа A, для которого выполняется

неравенство

$2^{|x_{п\,max}|}\cdot |x|_{м\,мах} = 2^{P}\cdot(1-2^{-S}) > = |X| > = 0$,

где $P = 2^{k}-1$.

В ЭВМ числа с плавающей точкой представляются в так называемой нормализованной форме, при которой в прямом коде мантисса нормализованного числа в старшем разряде модуля имеет ненулевое значение, а для двоичной системы счисления - нормализованная мантисса должна иметь в старшем разряде модуля прямого кода значение 1, т.е. для двоичной системы мантисса должна удовлетворять неравенству

$1 > |x_{M}| >= 0.5$.

Для плавающей точки максимальные значения абсолютной и относительной ошибок определяются следующим образом:

максимальная абсолютная ошибка представления чисел:

$\delta_{max} = \Delta_{max} / A_{min} = 2^{-(s+1)}\cdot 2^{p}/(x_{м\,мин}\cdot 2^{p}) = 2^{-(s+1)}\cdot 2^{p}/(2^{-1}\cdot 2^{p}) =$

$= 2^{-(s+1)}/(2^{-1}) = 2^{-s}$

Отсюда видно, что относительная ошибка при представлении чисел в форме с плавающей точкой существенно меньше, чем в случае с фиксированной точкой. Это, а также больший диапазон изменения представляемых чисел, является основным преимуществом представления чисел с плавающей точкой.

7.1 IEEE 754 – стандарт двоичной арифметики с плавающей точкой

Данный стандарт разработан ассоциацией IEEE (Institute of Electrical and Electronics Engineers) и используется для представления действительных чисел (чисел с плавающей точкой) в двоичном коде. Наиболее используемый стандарт для вычислений с плавающей точкой, используется многими микропроцессорами и логическими устройствами, а также программными средствами.

Полное название стандарта в ассоциации IEEE:

- IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985);
- IEEE стандарт для двоичной арифметики с плавающей точкой (ANSI/IEEE Sid 754-1985)

Название стандарта в международной электротехнической комиссии IEC: IEC 60559:1989, Binary floating-point arithmetic for microprocessor systems – двоичная арифметика с плавающей точкой для микропроцессорных систем.

В 2008 году ассоциация IEEE выпустила стандарт IEEE 754-2008, который включил в себя стандарт IEEE 754-1985.

Стандарт IEEE 754-1985 определяет:

- как представлять нормализованные положительные и отрицательные числа с плавающей точкой;
- как представлять денормализованные положительные и отрицательные числа с плавающей точкой;
- как представлять нулевые числа;
- как представлять специальную величину бесконечность (Infinity);
- как представлять специальную величину "Не число" (NaN или NaNs);

- четыре режима округления.

IEEE 754-1985 определяет форматы представления чисел с плавающей запятой:

- число половинной точности (Binary16, half precision) — компьютерный формат представления чисел, занимающий в памяти половину машинного слова или 16 бит (рисунок 2.4). В силу невысокой точности этот формат представления чисел с плавающей запятой обычно используется в видеокартах, где небольшой размер и высокая скорость работы важнее точности вычислений;

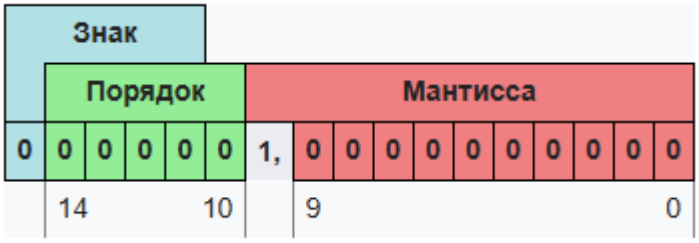


Рисунок 2.4 – Формат числа половинной точности

- число с одинарной точностью (single precision) 32 бита (рисунок 2.5)

Рисунок 2.5 – Формат числа одинарной точности

- число с двойной точностью (double precision) 64 бита (рисунок 2.6);

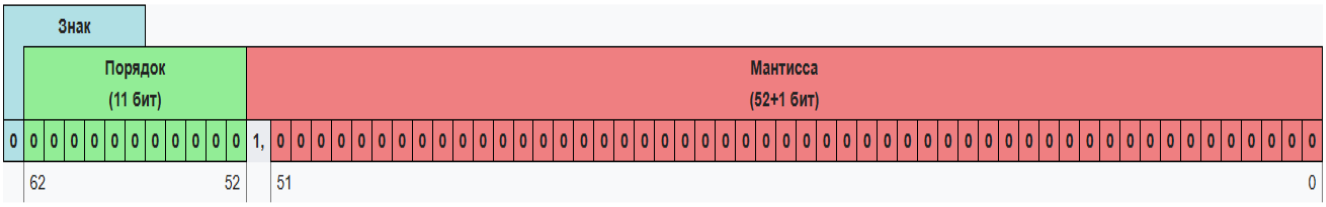


Рисунок 2.6 – Формат числа двойной точности

- число четверной точности (Binary128, quadruple precision) – компьютерный формат представления чисел, занимающий в памяти четыре машинных слова. Используется в случае необходимости крайне высокой точности (рисунок 2.7).

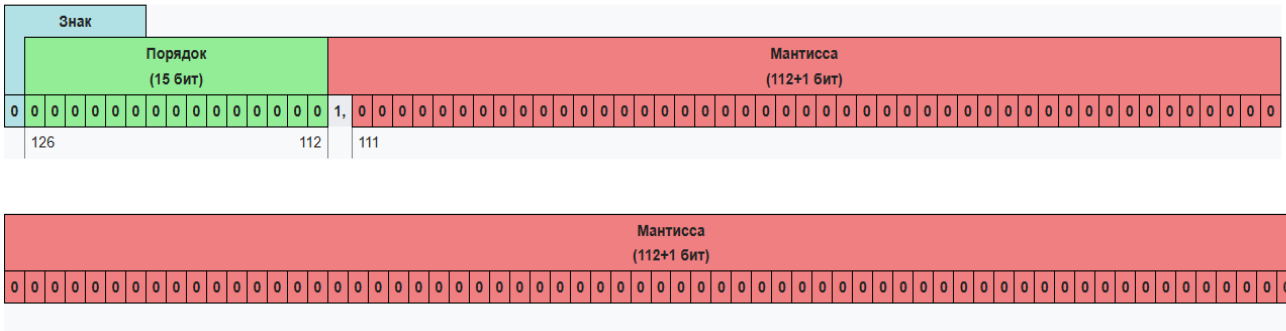


Рисунок 2.7 – Формат числа четверной точности

7.2 Основные понятия в представлении чисел с плавающей точкой

Представление числа в нормализованном экспоненциальном виде.

Возьмем, к примеру, десятичное число 155,625.

Представим это число в нормализованном экспоненциальном виде:

$1{,}55625 \cdot 10^{+2} = 1{,}55625 \cdot exp_{10}^{+2}$.

Число $1{,}55625 \cdot exp_{10}^{+2}$ состоит из двух частей: мантиссы M = 1.55625 и экспоненты $exp_{10} = +2$.

Если мантисса находится в диапазоне 1<=M<10, то число считается **нормализованным**.

Экспонента представлена основанием системы исчисления (в данном случае 10) и порядком (в данном случае +2).

Представление числа в денормализованном экспоненциальном виде.

Возьмем, к примеру, десятичное число 155,625

Представим это число в денормализованном экспоненциальном виде:

$0{,}155625 \cdot 10^{+3} = 0{,}155625 \cdot exp_{10}^{+3}$.

Число $0{,}155625 \cdot exp_{10}^{+3}$ состоит из двух частей: мантиссы M = 0,155625 и экспоненты $exp_{10} = +3$

Если мантисса находится в диапазоне 0,1<=M<1, то число считается **денормализованным.**

Экспонента представлена основанием системы исчисления (в данном случае 10) и порядком (в данном случае +3).

7.3 Преобразование десятичного числа в двоичное число с плавающей точкой

Пусть необходимо представить десятичное число с плавающей точкой в двоичное число с плавающей точкой в экспоненциальном нормализованном виде. Для этого разложим заданное число по двоичным разрядам:

$$155{,}625 = 1 \cdot 2^{7} + 0 \cdot 2^{6} + 0 \cdot 2^{5} + 1 \cdot 2^{4} + 1 \cdot 2^{3} + 0 \cdot 2^{2} + 1 \cdot 2^{1} + 1 \cdot 2^{0} + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

$$155{,}625 = 128 + 0 + 0 + 16 + 8 + 0 + 2 + 1 + 0{,}5 + 0 + 0{,}125$$

$155{,}625_{10} = 10011011{,}101_{2}$ – число в десятичной и в двоичной системе с плавающей точкой.

Приведем полученное число к нормализованному виду в десятичной и двоичной системе:

$1{,}55625 \cdot exp_{10}^{+2} = 1{,}0011011101 \cdot exp_{2}^{+111}$.

В результате получены основные составляющие экспоненциального нормализованного двоичного числа:

- мантиссу M = 1,0011011101;
- экспоненту $exp_2 = +111$.

7.4 Описание преобразования двоичного нормализованного числа в 32-битный формат IEEE 754

Основное применение в технике и программировании получили форматы 32 и 64 бита. Например, в Visual Basic используют типы данных single (32 бита) и double (64 бита). В Си аналогично используют float (32 бита) и double (64 бит)

Рассмотрим преобразование двоичного числа 10011011,101 в формат single precision (32 бита) стандарта IEEE 754.

Остальные форматы представления чисел в IEEE 754 являются увеличенной копией single precision.

Чтобы представить число в формате single precision IEEE 754, необходимо привести его к двоичному нормализованному виду. Ранее было проделано такое преобразование над числом 155,625 и было получено:

$1{,}0011011101 \cdot exp_2^{+111}$.

Далее будет рассмотрено, как двоичное нормализованное число преобразуется к 32 битному формату IEEE 754.

1. Число может быть положительным или отрицательным. Поэтому отводится 1 бит для обозначения знака числа:

- 0 – число положительное;
- 1– число отрицательное.

Этот самый старший бит в 32 битной последовательности.

2. Далее пойдут биты экспоненты, для этого выделяют 1 байт (8 бит).

Экспонента может быть, как и число, со знаком «+» или «–».

Для определения знака экспоненты, чтобы не вводить ещё один бит знака, добавляют смещение к экспоненте в половину байта $+127_{10}$ ($0111\ 1111_2$). То есть, если наша экспонента равна $+7_{10}$ ($+111_2$), то смещенная экспонента равна:

$127_{10} + 7_{10} = 134_{10}$.

К примеру, если бы наша экспонента была -7_{10}, то смещенная экспонента равнялась бы $127_{10} - 7_{10} = 120_{10}$. Смещенную экспоненту записывают в отведенные 8 бит. При этом, когда будет нужно получить экспоненту двоичного числа, необходимо будет просто вычесть 127_{10} от этого байта.

3. Оставшиеся 23 бита отводятся для мантиссы M. Но, у нормализованной двоичной мантиссы первый бит всегда равен 1, так как число лежит в диапазоне 1<=M<2. Следовательно, нет смысла, записывать единицу в отведенные 23 бита, поэтому в отведенные 23 бита записывают остаток от мантиссы.

Таким образом, в таблице 2.8 представлено десятичное число 155,625 в 32-битном формате IEEE 754.

Таблица 2.8 – Результаты преобразования (dec – десятичный формат, bin – двоичный формат, hex – шестнадцатеричный формат)

Знак числа	Смещенная экспонента	Остаток от мантиссы	Число 155,625 в формате IEEE 754
1 бит	8 бит	23 бит	IEEE 754
0 (bin)	1000 0110 (bin)	001 1011 1010 0000 0000 0000 (bin)	431BA000 (hex)
0 (dec)	134 (dec)	1810432 (dec)	

В результате десятичное число 155,625 представленное в IEEE 754 с одинарной точностью равно:

$0100\ 0011\ 0001\ 1011\ 1010\ 0000\ 0000\ 0000_2 = 431BA000_{16}$.

7.5 Арифметика с плавающей точкой

Операция сложения.

Операция сложения чисел предполагает наличие одинаковых масштабов складываемых величин. Для случая представления чисел с плавающей точкой это предполагает наличие одинаковых порядков у операндов, подлежащих суммированию. Поэтому при выполнении операции сложения чисел с плавающей точкой в общем случае должно быть реализовано три этапа:

- выравнивание порядков;
- сложение мантисс операндов, имеющих одинаковые порядки;
- определение нарушения нормализации и при необходимости её устранение.

Операция умножения

С точки зрения представления чисел с плавающей точкой поиск произведения

$C2 = A \cdot B$

сводится к поиску $C2_{п}$ и $C2_{м}$, соответственно порядку и мантиссе произведения на основании порядка $а_{п}$ и мантиссы $а_{м}$ множимого и порядка $в_{п}$ и мантиссы $в_{м}$ множителя. Учитывая общую запись чисел с плавающей точкой, произведение двух операндов представляется в виде

$$C2 = A \cdot B = 2^{а_{п}} \cdot а_{м} \cdot 2^{в_{п}} \cdot в_{м} = 2^{а_{п}+в_{п}} \cdot а_{м} \cdot в_{м} = 2^{c2_{п}} \cdot C2_{м},$$

откуда вытекает, что порядок произведения определяется как сумма порядков сомножителей, а мантисса произведения - как произведение мантисс сомножителей. Однако, учитывая то, что при умножении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого произведения и окончательное значение произведения будет найдено только после устранения нарушения нормализации.

Таким образом, имеем:

$C2_{п}` = а_{п} + в_{п}$;

$C2_{м}` = а_{м} \cdot в_{м}$.

Отсюда последовательность действий, обеспечивающих получение произведение двух чисел, заключается в следующем:

– определяется знак произведения как сумма по модулю два знаковых разрядов мантисс сомножителей;

– определяется предварительное значение порядка произведения посредством суммирования порядков сомножителей;

– определяется предварительное значение мантиссы произведения как произведения мантисс операндов;

– устраняется нарушение нормализации мантиссы произведения (если нарушение имеет место) соответствующей корректировкой предварительного значения порядка и мантиссы искомого произведения.

При формировании мантиссы произведения нормализованных чисел с плавающей точкой возможен только один вид нарушения нормализации – нарушение нормализации справа от точки с появлением нуля только в старшем разряде мантиссы.

Операция деления.

С точки зрения формирования частного представления чисел с плавающей точкой поиск частного СЗ =А/В сводится к поиску $СЗ_п$ и $СЗ_м$, соответственно порядку и мантиссы частного на основании порядка $а_п$ и мантиссы $а_м$ делимого и порядка $в_п$ и мантиссы $в_м$ делителя. Учитывая общую запись чисел с плавающей точкой, частное двух операндов представляется в виде

$СЗ = А/В = 2^{а п} \cdot а_м /(2^{в п} \cdot в_м) = 2^{ап-вп} \cdot (а_м / в_м) = 2^{сп} \cdot СЗ_м.$

Отсюда следует, что порядок частного определяется как разность порядка делимого и делителя, а мантисса - как частное от деления мантиссы делимого на мантиссу делителя. Однако, учитывая то, что при делении мантисс может произойти нарушение нормализации, в результате указанных действий будет найдено предварительное значения порядка и мантиссы искомого частного. Окончательные значения порядка и мантиссы частного будут определены после устранения нарушения нормализации в предварительном результате.

При формировании мантиссы частного нормализованных чисел с плавающей точкой возможно только один вид нарушения нормализации – нарушение нормализации слева от точки.

III Логические основы ЭВМ

1 Основные понятия алгебры логики

Алгебра логики находит широкое применение при синтезе и анализе схем ЭВМ. Это объясняется, с одной стороны, соответствием представления переменных и функций алгебры логики, с другой стороны, двоичным представлением информации и характером работы отдельных компонентов вычислительной техники, которые могут пропускать или не пропускать ток, иметь на выходе высокий или низкий уровень сигнала (напряжения или тока).

Основные понятия алгебры логики:

–*логическая переменная — это* такая переменная, которая может принимать одно из двух значений: истинно или ложно (да или нет, единица или ноль);

–*логическая константа — это* такая постоянная величина, значением которой может быть одно из двух значений: истинно или ложно (да или нет, единица или ноль);

–*логическая функция — это* такая функция, которая может принимать одно из двух значений (истинно или ложно, да или нет, единица или ноль), в зависимости от текущего значений её аргументов, в качестве которых используются логические переменные.

Логическая функция может быть одного (n=1) или нескольких (n >1) аргументов. Значение логической функции определяется комбинацией конкретных значений переменных, от которых она зависит. Комбинация конкретных значений переменных (аргументов функции) называется набором. Проведя аналогию с двоичным кодом, легко убедиться, что количество различных наборов N для n переменных определяется как $N= 2^n$.

Зависимость логической функции от переменных может задаваться по-разному. Это может быть задание функции в виде таблицы истинности, или словесное описание, или задание её в виде логического выражения.

Словесное описание, как правило, может использоваться в случае сравнительно несложной логической функции.

Таблица истинности является универсальным средством задания логической функции. Она включает все наборы для заданного количества переменных, определяющих значение логической функции, с указанием значений, которые принимает функция для каждого набора. В одной таблице истинности может задаваться несколько логических функций, зависящих от одних и тех же переменных. Таблица истинности для нескольких функций y_i трех переменных x_1, x_2, x_3 может быть задана виде таблицы 3.1.

Таблица 3.1 – Задание логической функции в виде таблицы истинности

№ п.п.	x_1		x_1	x_3	y_1	y_2	y_3	.....	y_n
0	0		0	0	0	1	1		0
1	0		0	1	1	1	0		1
2	0		1	0	1	1	1		0
3	0		1	1	0	1	0		-
4	1		0	0	1	0	1		0
5	1		0	1	0	0	0		1
6	1		1	0	0	0	1		-
7	1		1	1	1	1	0		1

В приведенной таблице истинности во второй, третей и четвертой колонках, помеченных соответственно x_1, x_2, x_3, даны все возможные наборы этих переменных. В следующих колонках приводятся значения функций y_1, y_2, y_n для каждого набора.

Логическая функция называется «*полностью определенная*», если для неё заданы значения по всем возможным наборам. Функция называется «*частично определенная*», если для некоторых наборов значения функции не заданы. В приведенной таблице истинности функции y_1, y_2, y_3 являются полностью определенными, а функция y_n – частично определенная (знак «-» означает неопределенность значения функции).

Логическим выражением называется комбинация логических переменных и констант, связанных элементарными базовыми логическими функциями (или логическими операциями), которые могут разделяться скобками.

Например, логическую функцию y_1, определенную в вышеприведенной таблице истинности, можно представить в виде логического выражения

$y1 = \overline{(x1 \bullet x2 + x1 \bullet x3 + x2 \bullet x3)} \bullet (x1 + x2 + x3) + x1 \bullet x2 \bullet x3,$ где, “+”, “$\bullet$”, а также верхнее подчеркивание – знаки базовых логических функций.

Набор элементарных логических операций, с помощью которых можно задать любую, сколь угодно сложную логическую функцию, называется «функционально полная система логических функций». Иногда такую систему называют базисом.

В качестве элементарных логических функций функционально полных систем логических функций используются функции одной или двух логических переменных.

Наиболее распространенной в алгебре логики является функционально полная система логических функций, которая в качестве базовых логических функций использует функцию одной переменной НЕ (функция отрицания), и две функции двух переменных – И (конъюнкция, или логическое умножение) и ИЛИ (дизъюнкция, или логическое сложение). Эта система получила название «система *булевых* функций», или *булевый базис*. В алгебре логики имеется целый раздел «*Алгебра Буля*»), посвященный этому базису.

В вышеприведенной таблице, описывающей функции от двух переменных, в последней колонке даны варианты записи этих функций в булевом базисе.

2 Элементы алгебры Буля

В алгебре Буля логические выражения включают логические операции И, ИЛИ, НЕ, которые могут быть использованы в самых различных сочетаниях. При оценке значения такого выражения необходимо решить это выражение для конкретного набора переменных. В алгебре Буля используется следующая приоритетность выполнения операций: сначала рассчитываются значения имеющих место отрицаний и скобок, затем выполняется операция И (логическое умножение); самый низший приоритет имеет операция ИЛИ (логическая сумма).

При работе с булевыми логическим выражениями используются следующие законы и правила.

Переместительный (коммутативный) закон. Закон справедлив как для конъюнкции, так и для дизъюнкции.

$x_1 + x_2 + x_3 + x_4 = x_4 + x_3 + x_2 + x_1$ – от перемены мест логических слагаемых сумма не меняется.

$x_1\ x_2\ x_3\ x_4 = x_4\ x_3\ x_2\ x_1$ – от перемены мест логических сомножителей их произведение не меняется.

Этот закон справедлив для любого количества логических операндов.

Сочетательный (ассоциативный) закон. Закон справедлив как для конъюнкции, так и для дизъюнкции.

$x_1 + x_2 + x_3 + x_4 = (x_2 + x_3) + x_1 + x_4 = (x_1 + x_4) + (x_2 + x_3)$ – при логическом сложении отдельные слагаемые можно заменить их суммой.

$x_1\ x_2\ x_3\ x_4 = (x_2\ x_3)\ x_1 x_4 = (x_1\ x_4)\ (x_2\ x_3)$ – при логическом умножении от- дельные логические сомножители можно заменить их произведением.

Распределительный (дистрибутивный) закон.

$(x_1 + x_2)\ x_3 = x_1\ x_3 + x_2\ x_3$.

$(x_1 + x_2)\ (x_1 + x_3) = x_1 + x_2\ x_3$.

Правило Де Моргана:

$\overline{x1 + x2} = \bar{x}1\bar{x}2$ –отрицание суммы равно произведению отрицаний,

$\overline{x1x2} = \bar{x}1 + \bar{x}2$ –отрицание произведения равно сумме отрицаний.

Операция склеивания:

$\bar{x}iA + xiA = A$ - операция *склеивания* для конъюнкций, где А - переменная или любое логическое выражение.

$(\bar{x}i + A)(xi + A) = A$ – операция склеивания для дизъюнкций.

Если в качестве А используется простая конъюнкция, т.е. конъюнкция, представляющая собой логическое произведение переменных и их отрицаний, то имеет место

$\bar{x}1x2\bar{x}3\bar{x}4x5\bar{x}6 + \bar{x}1\bar{x}2\bar{x}3\bar{x}4x5\bar{x}6 = \bar{x}1\bar{x}3\bar{x}4x5\bar{x}6$.

Как видно, в результирующем выражении количество переменных на единицу меньше, чем в склеенных конъюнкциях. Количество переменных в простой конъюнкции называется *рангом конъюнкции*, т.е. операция склеивания, примененная к простым конъюнкциям, дает результат с рангом, на единицу меньшим ранга исходных конъюнкций.

Операции с отрицаниями:

$\bar{\bar{x}} = x$ – двойное отрицание равносильно отсутствию отрицания.

$\bar{x} \bullet x = 0$

$\bar{x} + x = 1$

Операции с константами:

$x_1 + 1 = 1, \quad x_1 + 0 = x_1,$

$x_1 \ 1 = x_1, \quad x_1 \ 0 = 0.$

Операции с одинаковыми операндами:

$x_1 + x_1 + x_1 + x_1 + \ldots + x_1 = x_1$,

$x_1 \ x_1 \ x_1 \ldots . \ x_1 = x_1$ при любом числе повторений.

Законы и правила алгебры *Буля* могут быть доказаны путем логического рассуждения, однако такое доказательство применимо только для простейших случаев. Доказать справедливость того или иного правила можно, если с помощью различных преобразований привести правую часть правила к выражению в левой части (или наоборот). Универсальным приемом доказательства является использование таблицы истинности. Это основано на том утверждении, что два выражения (правая и левая части правила или закона) *эквивалентны*, если они принимают одинаковые значения на всех наборах логических переменных.

Например, правило двойного отрицания, которое справедливо не только относительно одной переменной, но и любого логического выражения, можно доказать следующим рассуждением: если неверно утверждение, что выражение ложно, то очевидно утверждение, что это выражение истинно.

Доказать справедливость распределительного закона в интерпретации выражением

$(x_1 + x_2)(x_1 + x_3) = x_1 + x_2 x_3$

можно за счет приведения левой части к выражению правой части, раскрыв скобки:

$(x_1 + x_2(x_1 + x_3) = x_1 x_2 + x_1 x_1 + x_1 x_3 . + x_2 x_3 = x_1 x_2 + x_1 + x_1 x_3 . + x_2 x_3 =$

$= x_1 (x_2 + 1 + x_3) + x_2 x_3 =$

помня, что логическая сумма с одним слагаемым, равным константе 1, равна 1, можно записать

$= x_1 + x_2 x_3$.

Используем таблицу истинности для доказательства правила Де Моргана в варианте

$\overline{x1 + x2} = \bar{x}1\bar{x}2$ - отрицание суммы равно произведению отрицаний.

Составим таблицу истинности для правой и левой частей и составляющих их функций, таблица 3.2.

Таблица 3.2 – Доказательство правила Де Моргана

x_1	x_2	$x_1 + x_2$	$\overline{x_1 + x_2}$	$\overline{x_1}$	$\overline{x_2}$	$\overline{x_1}\overline{x_2}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Из таблицы истинности видно, что правая и левая части доказываемого правила принимают одинаковые значения на всех наборах, следовательно, они эквивалентны.

3 Формы представления логических функций

Одну и ту же логическую функцию можно представить различными логическими выражениями. Среди множества выражений, которыми представляется логическая функция, особое место занимают две канонические формы:

совершенная конъюнктивная нормальная форма (СКНФ),

совершенная дизъюнктивная нормальная форма (СДНФ).

Совершенная дизъюнктивная нормальная форма представляет собой дизъюнкцию простых конъюнкций, где под термином простая *конъюнкция* имеется в виду конъюнкция переменных или их отрицаний. В СДНФ простые конъюнкции содержат все переменные в своей прямой или инверсной форме и отражают собой наборы, на которых представляемая функция имеет единичное значение. Такие конъюнкции называются конституентами единицы рассматриваемой функции. Поэтому СДНФ представляет собой дизъюнкцию (логическую сумму), слагаемыми которой являются *конституенты единицы*. Общая запись СДНФ функции y имеет вид

$$y = \vee\ x_1^{\delta 1}\, x_2^{\delta 2}\, x_3^{\delta 3} \ldots x_{(n-1)}^{\delta(n-1)}\, x_n^{\delta n},$$

где $$x_i^{\delta\iota} = \begin{cases} x_i, \text{ если } \delta\iota = 1, \\ \overline{x_i}, \text{ если } \delta\iota = 0. \end{cases}$$

СДНФ легко сформировать на основе таблицы истинности. Например, если функции задаются в виде таблицы 3.3, то СДНФ для них будет иметь следующий вид:

$$y1 = \overline{x1}\,\overline{x2}\,\overline{x3} + \overline{x1}\,\overline{x2}x3 + \overline{x1}x2x3 + x1x2\overline{x3} + x1x2x3;$$

$$y2 = \overline{x1}\,\overline{x2}\,\overline{x3} + \overline{x1}x2x3;$$

$$y3 = \overline{x1}\,\overline{x2}\,\overline{x3} + \overline{x1}\,\overline{x2}x3 + \overline{x1}x2x3 + x1\overline{x2}\,\overline{x3} + x1x2\overline{x3} + x1x2x3.$$

Таблица 3.3 – Пример формирования СДНФ

N	x_1	x_2	x_3	y_1	y_2	y_3
0	0	0	0	1	1	1
1	0	0	1	1	0	1
2	0	1	0	0	0	0
3	0	1	1	1	1	1
4	1	0	0	0	0	1
5	1	0	1	0	0	0
6	1	1	0	1	0	1
7	1	1	1	1	0	1

Совершенная конъюнктивная нормальная форма – это конъюнкция простых дизъюнкций, где под термином простая дизъюнкция имеется в виду дизъюнкция переменных или их отрицаний. В СКНФ простые дизъюнкции содержат все переменные в своей прямой или инверсной форме и представляют собой отрицание конституент нуля. Общая запись СКНФ функции y имеет вид

$$y = \wedge(x_1^{\delta 1} + x_2^{\delta 2} + x_3^{\delta 3} \ldots + x_{(n-1)}^{\delta(n-1)} + x_n^{\delta n}),$$

где $x_i^{\delta\iota} = \begin{cases} x_i, \text{ если } \delta\iota = 1, \\ \overline{x_i}, \text{ если } \delta\iota = 0. \end{cases}$

СКНФ легко сформировать на основе таблицы истинности. Например, для функций из предыдущей таблицы 2.3 имеем:

$y_1 = (x1 + \bar{x}2 + x3)(\bar{x}1 + x2 + x3)(\bar{x}1 + x2 + \bar{x}3);$

$y_2 = (x1 + \bar{x}2 + \bar{x}3)(x1 + \bar{x}2 + x3)(\bar{x}1 + x2 + x3)(\bar{x}1 + x2 + \bar{x}3)(\bar{x}1 + \bar{x}2 + x3)(\bar{x}1 + \bar{x}2 + \bar{x}3);$

$y_3 = (x1 + \bar{x}2 + x3)(\bar{x}1 + x2 + x3).$

СКНФ строится на основе конституент нуля. Конституента нуля представляет набор логических переменных, на котором логическая функция принимает значение 0. Каждая скобка в приведенных выражениях представляет собой отрицание конституенты нуля соответствующей функции, а запись функции в виде конъюнкция таких скобок представляет собой условие, при котором отсутствуют все конституенты нуля определяемой функции, при выполнении которого функция имеет единичное значение.

Например, для y_1 выражение первой скобки представляет собой отрицание набора значений переменных второй строки, на котором функция y_1 имеет нулевое значение, выражение второй скобки представляет собой отрицание набора значений переменных четвертой строки, на котором функция y_1 также имеет нулевое значение, выражение третьей скобки представляет собой

отрицание набора значений переменных пятой строки, на котором функция y_1 имеет нулевое значение.

Из вышеизложенного следует, что любую функцию можно представить или в СДНФ, или в СКНФ, а так как эти формы представлены в базисе Буля, то значит, что этот базис (базис И, ИЛИ, НЕ) является функционально полным.

Если функция задана в СДНФ и требуется найти ее СКНФ, то такой переход можно выполнить, составив по заданной СДНФ таблицу истинности для этой функции, а на основе полученной таблицы составить СКНФ заданной функции.

Однако в некоторых случаях может оказаться более удобным подход, который поясняется следующим примером.

Пример

По заданной СДНФ функции

$y_3 = x1x2\bar{x}3 + \bar{x}1\bar{x}2x3 + x1\bar{x}2\bar{x}3 + x1x2x3 + \bar{x}1\bar{x}2\bar{x}3 + \bar{x}1x2x3$

найти запись этой функции в СКНФ.

Решение

Запишем логические выражение отрицания заданной функции, т.е. найдем логическое условие, при котором эта функция имеет нулевое значение. В качестве такого выражения можно взять дизъюнкцию конъюнкций, где каждая конъюнкция представляет собой конституенту нуля заданной функции. Очевидно, что конституенты нуля это те наборы, которые не являются наборами, соответствующими конституентам единицы, которые использованы в СДНФ. Таким образом, можно записать

$$\bar{y}3 = \bar{x}1x2\bar{x}3 + x1\bar{x}2x3.$$

Эту запись можно интерпретировать как словесное описание функции:

– функция у равна нулю, если имеет место хотя бы одна из конституент нуля.

В этой записи представлена дизъюнкция тех наборов, которые не использовались в записи функции y_3. Возьмем отрицание правой и левой частей полученного уравнения и применим к правой части правило Де Моргана.

$$\bar{\bar{y}}3 = \overline{\bar{x}1x2\bar{x}3 + x1\bar{x}2x3} = \overline{\bar{x}1x2\bar{x}3} \bullet \overline{x1\bar{x}2x3}.$$

Применим *правило Де Моргана* к отрицаниям конъюнкций, полученным в правой части:

$$\bar{\bar{y}}3 = y3 = (x1 + \bar{x}2 + x3)(\bar{x}1 + x2 + \bar{x}3).$$

Полученная запись для у является искомой СКНФ.

4 Синтез логических схем по логическим выражениям

Логические схемы строятся на основе логических элементов, набор которых определяется заданным логическим базисом.

Для базиса Буля в качестве логических элементов используются элементы, реализующие базовые логические функции И, ИЛИ, НЕ, которые имеют обозначения, приведенные на рисунке 3.1.

При синтезе схемы по логическому выражению, составляющие логические

операции представляются в виде соответствующих логических элементов, связи между которыми определяются последовательностью выполнения логических операций в заданном выражении.

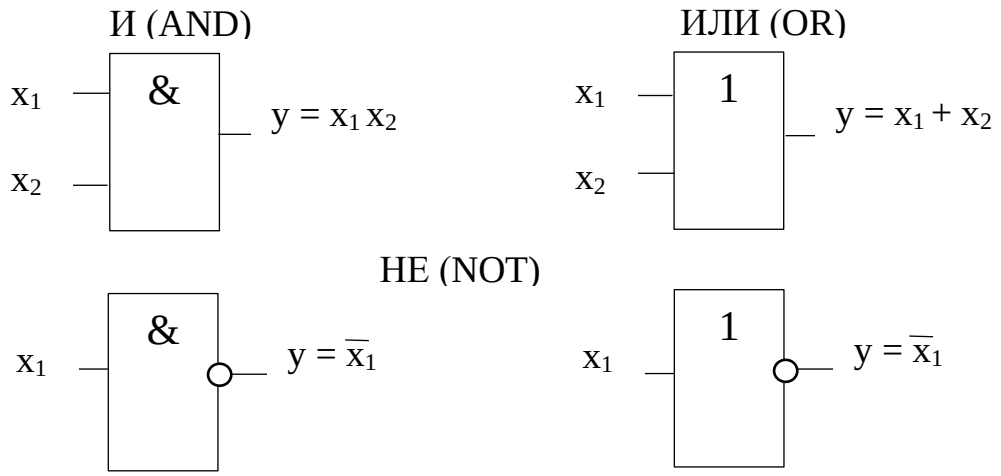


Рисунок 3.1 – Базовые логические элементы И, ИЛИ, НЕ

Пример

Синтезировать логическую схему в базисе И, ИЛИ, НЕ, реализующую логическое выражение

$$y_1 = \overline{(x_1x_2 + x_1x_3 + x_2x_3)}(x_1 + x_2 + x_3) + x_1 x_2 x_3.$$

Решение

Входными сигналами синтезируемой схемы являются x_1, x_2, x_3, а выходным – y_1.

Реализацию заданного выражения в виде логической схемы можно начать или с последней операции, или с первой.

Последней операцией в заданном выражении является операция логического сложения двух операндов:

$$\overline{(x_1x_2 + x_1x_3 + x_2x_3)}(x_1 + x_2 + x_3) \quad \text{и} \quad x_1 x_2 x_3,$$

поэтому для её реализации требуется элемент ИЛИ с двумя входами, на выходе которого будет сформирован сигнал, соответствующий y_1, если на его входы будут поданы эти два слагаемые (например, первое слагаемое на второй вход, а второе слагаемое на первый вход).

На первый вход ИЛИ подается логическое произведение $x_1\ x_2\ x_3$, для реализации которого необходимо использовать логический элемент И с тремя входами, на которые подаются входные переменные x_1, x_2, x_3. Аналогичным образом рассматривается последовательность формирования выражения

$$\overline{(x_1x_2 + x_1x_3 + x_2\ x_3)}(x_1 + x_2 + x_3),$$

которое соответствует сигналу, подаваемому на второй вход элемента ИЛИ. В результате синтезируется схема для заданного выражения, приведенная на рисунке 3.2.

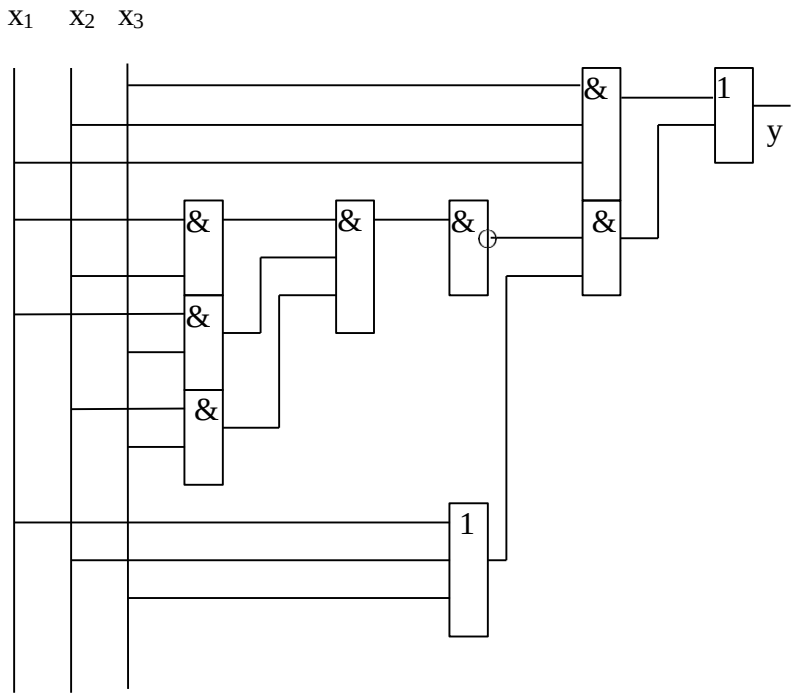


Рисунок 3.2 – Синтезированная логическая схема

5 Минимизация логических выражений

Учитывая то, что одну и ту же логическую функцию можно представить различными выражениями, перед реализацией функции в виде логической схемы весьма важным является выбор из всех возможных выражений, соответствующих данной функции, самого простого. Решить эту проблему можно за счет использования процедуры минимизации логического выражения. Из множества методов минимизации наиболее часто используются:

- *минимизация методом Квайна;*
- *минимизация с использованием диаграмм Вейча (или карт Карно).*

5.1 Минимизация методом Квайна

Метод минимизации *Квайна* предполагает следующее.

В качестве исходной формы представления логического выражения используется СДНФ.

Если подлежащее минимизации выражение имеет другую форму, то приведение к СДНФ осуществляется за счет открытия скобок, избавления от отрицаний логических выражений, более сложных, чем отрицание переменной (используется правило Де Моргана).

В результате таких действий будет получена дизъюнкция простых конъюнкций (т.е. конъюнкций, логическими сомножителями которой являются переменные или их отрицания). Далее для получения СДНФ, у которой используемые простые конъюнкции включают все переменные или их отрицания, в конъюнкции, где представлены не все переменные, вводят недостающие переменные x_i за счет умножения расширяемой конъюнкции на

$$(x_i + \bar{x}_i)$$

и раскрытия скобок.

Метод Квайна выполняется в два этапа.

Первый этап имеет своей целью получение тупиковой формы, представляющей собой дизъюнкцию, в качестве слагаемых которой используются конъюнкции, каждая из которых не склеивается ни с одной другой конъюнкцией, входящей в это выражение. Такие конъюнкции называются простыми импликантами.

Данный этап выполняется за счет реализации отдельных шагов. На каждом шаге на основании выражения, полученного на предыдущем шаге, выполняются все возможные операции склеивания для пар имеющихся конъюнкций. Каждый шаг понижает ранг исходных конъюнкций на единицу. Шаги повторяются до получения тупиковой формы.

Второй этап имеет своей целью устранения из тупиковой формы всех избыточных простых импликант, что дает в результате минимальное логическое выражение.

Пример

Найти методом Квайна минимальное выражение для функции y:

$$y = \bar{x}1\bar{x}2\bar{x}3x4 + x1\bar{x}2\bar{x}3\bar{x}4 + \bar{x}1x2\bar{x}3x4 + \bar{x}1x2x3\bar{x}4 + x1x2\bar{x}3\bar{x}4 + x1\bar{x}2x3x4 +$$
$$+ x1\bar{x}2x3\bar{x}4 + x1x2x3\bar{x}4 + x1\bar{x}2x3x4 + \bar{x}1x2x3\bar{x}4.$$

Решение

Первый этап

$$y = \underline{\bar{x}_1 \bar{x}_2 x_3 x_4}^{1} + \underline{x_1 x_2 x_3 x_4}^{2} + \underline{x_1 x_2 x_3 x_4}^{3} + \underline{x_1 x_2 x_3 x_4}^{4} + \underline{x_1 x_2 x_3 x_4}^{5} +$$
$$+ \underline{x_1 x_2 x_3 \bar{x}_4}^{6} + \underline{x_1 x_2 x_3 x_4}^{7} + \underline{x_1 x_2 x_3 x_4}^{8} + \underline{x_1 x_2 x_3 x_4}^{9} + \underline{\bar{x}_1 \bar{x}_2 x_3 x_4}^{10} =$$

(проставлены номера конъюнкций; подчеркнуты те конъюнкции, которые склеивались на очередном шаге)

$$= \underset{1\text{-}9}{\underline{x_2 x_3 x_4}} + \underset{1\text{-}10}{\underline{x_1 x_2 x_3}} + \underset{2\text{-}5}{\underline{x_2 x_3 x_4}} + \underset{2\text{-}6}{x_1 x_2 x_3} + \underset{2\text{-}8}{\underline{x_1 x_2 x_4}} + \underset{3\text{-}5}{x_1 x_2 x_3} + \underset{3\text{-}6}{\underline{x_2 x_3 x_4}} +$$

$$+ \underset{4\text{-}5}{\underline{x_1 x_2 x_4}} + \underset{4\text{-}8}{\underline{x_2 x_3 x_4}} + \underset{4\text{-}10}{\underline{x_1 x_3 x_4}} + \underset{7\text{-}8}{\underline{x_1 x_3 x_4}} + \underset{7\text{-}9}{\underline{x_1 x_2 x_3}} + \underset{7\text{-}10}{\underline{x_2 x_3 x_4}} =$$

(выражение представлено только результатами склеивания; если бы в исходном выражении не были подчеркнутые конъюнкции, то они должны были бы через знак «+» дописаны к сумме результатов склеивания; в скобке под конъюнкцией (i-j) указывают, что данная конъюнкция является результатом склеивания i-й и j-й конъюнкций исходного выражения):

$$= \underset{1\text{-}12}{x_2 x_3} + \underset{2\text{-}11}{x_2 x_3} + \underset{3\text{-}6}{x_2 x_3} + \underset{3\text{-}8}{x_2 x_4} + \underset{4\text{-}7}{x_2 x_4} + \underset{8\text{-}12}{x_3 x_4} + \underset{9\text{-}10}{x_3 x_4} + \underset{5}{x_1 x_2 x_3} =$$

$$= \underset{1\text{-}13}{x_2 x_3} + \underset{2\text{-}12}{x_2 x_3} + \underset{3\text{-}7}{x_2 x_3} + \underset{}{x_2 x_3} + \underset{3\text{-}9}{x_2 x_4} + \underset{5\text{-}8}{x_2 x_4} + \underset{9\text{-}13}{x_2 x_4} + \underset{10\text{-}11}{x_3 x_4} + \underset{6}{x_1 x_2 x_3} =$$

(к результатам склеивания логически добавлен ни с чем не склеенный пятый член исходного выражения)

$$= \underset{1\text{-}2}{x_2 x_3} + \underset{1\text{-}3}{x_2 x_3} + \underset{1\text{-}4}{x_2 x_4} + \underset{3\text{-}4}{x_3 x_4} + \underset{5}{x_1 x_2 x_3} = y_T$$ - тупиковая форма.

Последнее выражение получено из предыдущего посредством удаления повторяющихся членов

Второй этап

На основании исходного выражения и полученной тупиковой формы составляется и заполняется импликантная таблица 2.4. Колонки приведенной таблицы помечены конституентами единицы, имеющимися в исходном логическом выражении. Строки таблицы помечены простыми импликантами полученной тупиковой формы.

Таблица 2.4 – Импликантная таблица

	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3\bar{x}_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3\bar{x}_4$	$x_1x_2\bar{x}_3\bar{x}_4$	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$	$x_1x_2x_3\bar{x}_4$	$x_1x_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$
	1	2	3	4	5	6	7	8	9	10
$\bar{x}_2x_3$	*						*		*	*
$\bar{x}_2x_3$		*	*		*	*				
$\bar{x}_2x_4$		*		*	*			*		
$\bar{x}_3x_4$				*			*	*		*
$\bar{x}_1\bar{x}_2x_3$			*		*					

Звездочками в каждой строке отмечены те конституенты единицы, которые покрываются соответствующей простой импликантой (практически отмечаются те конституенты единицы, которые включают простую импликанту как свою составную часть).

Анализируя покрытия простыми импликантами конституент единицы заданной функции, составляем её минимальное выражение:

$$y_{min} = \underline{\ }1\ x_2\ x_3 + 2\ \underline{\ }\ x_2\ x_3 + 3\ \underline{\ }\ x_2 x_4.$$

Минимальное выражение y_{min} формируется за счет последовательного включения простых импликант. При этом используется следующая приоритетность включения импликант в формируемое минимальное выражение:

- простая импликанта является единственной, покрывающей одну из колонок;
- если импликант вышеуказанного типа нет, то выбирается импликанта, покрывающая большее количество еще не покрытых колонок.

Последовательность включения простых импликант в приведенное минимальное выражение:

$\bar{x}2x3$ - единственная импликанта, покрывающая колонку 1, при этом из дальнейшего рассмотрения исключаются все колонки, покрываемые этой импликантой, т.е. колонки 1,7,9 10;

$x2\bar{x}3$ - покрывает максимальное число колонок, оставшихся для рассмотрения (колонки 2,3,5,6) - эти колонки из дальнейшего рассмотрения исключаются;

$\bar{x}2x4$ - покрывает оставшиеся две колонки 4, 8; после выбрасывания этих двух колонок, для рассмотрения не останется ни одной колонки, не покрытой уже включенными в формируемое выражение простыми импликантами. Поэтому простые импликанты $x3\bar{x}4$ и $\bar{x}1x2\bar{x}3$ найденной тупиковой формы являются избыточными и в минимальном логическом выражении для заданной функции не присутствуют.

5.2 Минимизация методом Карт Карно, или диаграмм Вейча

Как было показано ранее, очевидным путем минимизации логических функций является последовательное использование законов булевой алгебры. Основными эквивалентными преобразованиями для минимизации являются склеивание и поглощение.

Поглощение: $X + XY = X\ (1 + Y) = X$.

Склеивание: $XY + X\bar{Y} = X$.

После проведения всех возможных преобразований получают функцию, не имеющую избыточных членов и не поддающуюся дальнейшей минимизации. Эту форму записи функции называют тупиковой. Функция может иметь несколько тупиковых форм.

Минимизация функций алгебраическим методом требует известного навыка. Далеко не всегда очевидно, что полученная форма является тупиковой; иногда трудно определить и склеивающиеся слагаемые.

Разработан метод минимизации как бы автоматизирующий процедуру поиска «склеивающихся» слагаемых — метод карт Карно.

Карта Карно, или *диаграмма Вейча*, — это таблица, имеющая ячейки для всех возможных минтермов функции. Можно построить карты Карно для функций, минтермы которых содержат два, три и более переменных (обычно не больше 5—6). На рисунке 3.3 показана карта Карно для функции двух переменных.

$X_2\backslash X_1$	"0"	"1"
"0"		
"1"		

Рисунок 3.3 – Карта Карно для функции двух переменных

Вдоль верхней грани проставлены возможные значения переменной X_1, вдоль левой боковой грани — возможные значения переменной X_2 («0» и «1»). Карта содержит четыре клетки. В каждой клетке изображают один из возможных минтермов $X_1X_2, \overline{X}_1X_2, X_1\overline{X}_2, \overline{X}_1\overline{X}_2$. Если какой-то из этих минтермов в совершенной дизъюнктивной нормальной форме (СДНФ) записи функции присутствует, то в соответствующей клетке карты Карно ставится «1». Если какого-то минтерма в полученной функции нет, то в соответствующей клетке карты Карно ставится «0».

Важно отметить, что рядом стоящие минтермы (наборы) *должны отличаться друг от друга только одной позицией* (т.е. только одной единицей, как 00 и 01, 01 и 11, 11 и 10).

Карта Карно для функций трех переменных показаны на рисунке 3.4.

$X_1\backslash X_2X_3$	"00"	"01"	"11"	"10"
"0"				
"1"				

Рисунок 3.4 – Карта Карно для функции трех переменных

Склеивание осуществляется между теми минтермами, которые записаны в виде «1» в соседних клетках карты (соседних по вертикали или горизонтали). При склеивания в карте Карно соседние «1» могут быть покрыты квадратами со стороной, кратной двум.

Соседними считаются также клетки верхнего и нижнего рядов карты и клетки крайнего левого и крайнего правого рядов (можно представить карту как развертку цилиндра, ось которого следует принять либо горизонтальной, либо вертикальной в зависимости от удобства объединения).

Два минтерма, находящиеся в соседних клетках, могут быть представлены в виде одного логического произведения переменных, число которых на одну

единицу меньше, чем в каждом из соседних минтермов. Если соседними оказались сразу четыре минтерма с «1», то такую группу минтермов можно заменить конъюнкцией переменных, число которых меньше, чем в каждом минтерме, уже на две «1». Учитывая, что A+A+A+...= A, одну единицу, изображающую минтерм, можно объединить в пары несколько раз, например, первый раз с соседней единицей по вертикали, второй раз — с соседней единицей по горизонтали.

Пример 1

Используя метод карт Карно, минимизируем функцию:

$$\overline{X}_1X_2X_3 + X_1\overline{X}_2X_3 + X_1X_2\overline{X}_3 + X_1X_2X_3.$$

Заполненная карта Карно, соответствующая этой функции, показана на рисунке 3.5.

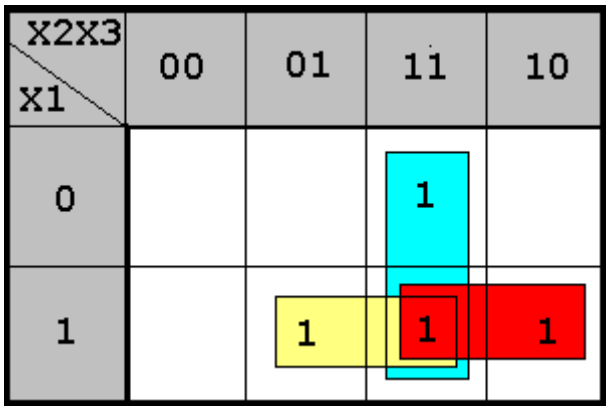


Рисунок 3.5 – Заполненная карта Карно

Единица, изображающая минтерм $X_1X_2X_3$, входит сразу в три объединения, обозначенные прямоугольниками красного, синего и желтого цвета. Объединение, соответствующее желтому прямоугольнику, отражает «склеивание» минтемов:

$$X_1X_2X_3 + X_1\overline{X}_2X_3,$$

$$X_1X_2X_3 + X_1\overline{X}_2X_3 = X_1X_3(X_2 + \overline{X}_2) = X_1X_3.$$

Объединение, отмеченное синим прямоугольником, отражает «склеивание» минтермов:

$$X_1X_2X_3 + \overline{X}_1X_2X_3,$$

$$X_1X_2X_3 + \overline{X}_1X_2X_3 = X_2X_3(X_1 + \overline{X}_1) = X_2X_3.$$

Объединение, отмеченное красным прямоугольником, отражает склеивание минтермов

$$X_1X_2X_3 + X_1X_2\overline{X}_3,$$

$$X_1X_2X_3 + X_1X_2\overline{X}_3 = X_1X_2(X_3 + \overline{X}_3) = X_1X_2.$$

Отсюда $Y = X_1X_2 + X_2X_3 + X_1X_3$.

Карта Карно позволила легко выявить «склеивающиеся» минтермы и облегчила задачу минимизации функции.

Пример 2

Минимизируем функцию четырех переменных, заданную совокупностью наборов, на которых функция принимает единичные значения (рисунок 3.6).

0000	1000	0101	1101	0011	0111	0010	1110	1010

Рисунок 3.6 – Функция четырех переменных, заданная совокупностью наборов, на которых функция принимает единичные значения

Соответствующая ей карта Карно и склеивающиеся области показаны на рисунке 3.7.

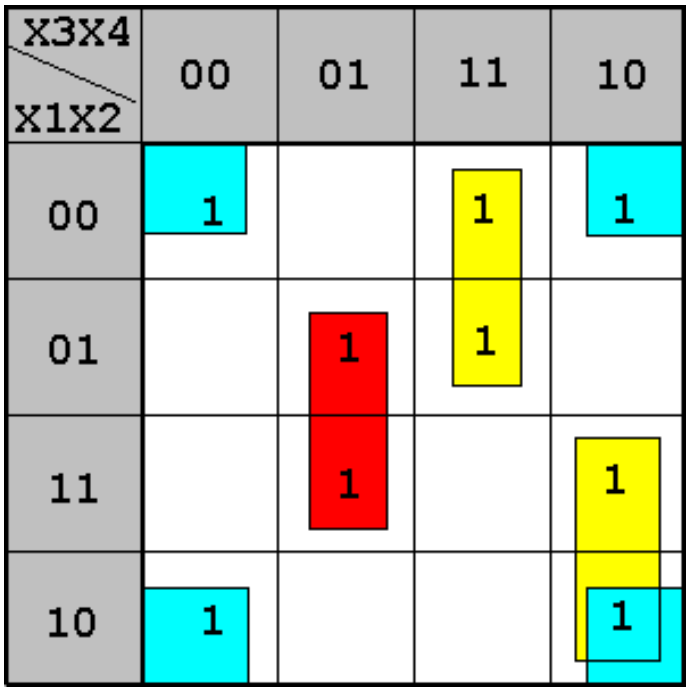


Рисунок 3.7 – Заполненная карта Карно

Склеивание дает 4 минтерма (слагаемого):

	Не зависит от X1
	Не зависит от X2
	Не зависти от X1 и X3

В итоге функция принимает вид

$X_2 \overline{X}_3 X_4 + \overline{X}_1 X_3 X_4 + X_1 X_3 \overline{X}_4 + \overline{X}_2 \overline{X}_4$.

Тупиковая форма записи функции, полученная при минимизации по карте Карно, используется при проектировании логических устройств так же, как и полученная методом алгебраической минимизации.

6 Логические базисы И-НЕ, ИЛИ-НЕ

Булевый базис не является единственной функционально полной системой логических функций. Среди других наибольшее распространение получили базис И-НЕ и базис ИЛИ-НЕ.

Чтобы доказать логическую полноту любого базиса, достаточно показать, что в этом базисе можно реализовать базовые функции И, ИЛИ, НЕ.

Для базиса И-НЕ в качестве базового элемента используется элемент, приведенный на рисунке 3.8а.

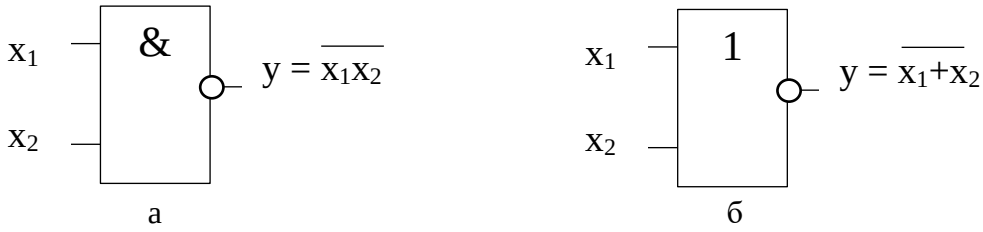


Рисунок 3.8 – Логические базисы И-НЕ, ИЛИ-НЕ

Реализация с помощью функции И-НЕ базовых функций алгебры Буля осуществляется следующим образом.

ИЛИ: $x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\overline{x_1}\,\overline{x_2}}$

И: $x_1\, x_2 = \overline{\overline{x_1 x_2}}$

Функция НЕ реализуется с помощью схемы И-НЕ с одним входом.

На рисунке 3.9 приведена схемная реализация функций И, ИЛИ, НЕ в базисе И - НЕ.

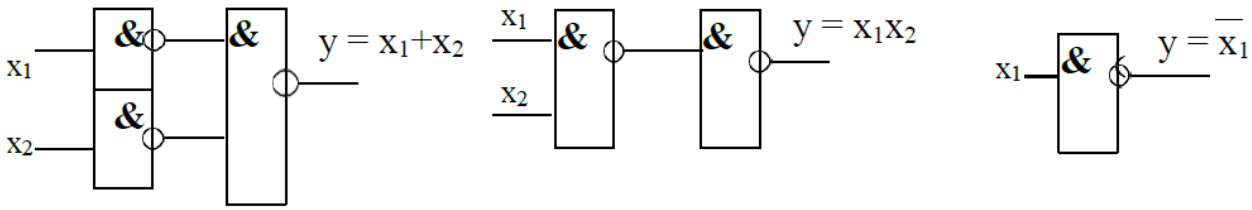


Рисунок 3.9 – Схемная реализация функций И, ИЛИ, НЕ в базисе И-НЕ

Реализация с помощью логической функции ИЛИ-НЕ базовых функций алгебры Буля осуществляется следующим образом.

ИЛИ: $x_1 + x_2 = \overline{\overline{x_1 + x_2}}.$

И: $x_1 x_2 = \overline{\overline{x_1 x_2}} = \overline{\overline{x_1} + \overline{x_2}}$

Функция НЕ реализуется с помощью схемы ИЛИ-НЕ с одним входом.

На рисунке 3.10 приведена схемная реализация операции И, ИЛИ, НЕ в базисе ИЛИ–НЕ.

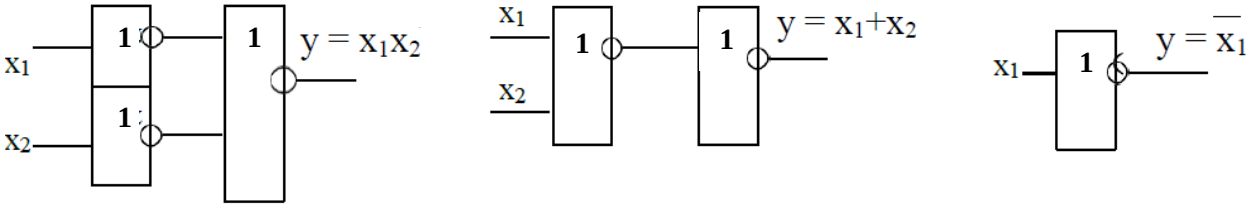


Рисунок 3.10 – Схемная реализация функций И, ИЛИ, НЕ в базисе ИЛИ-НЕ

При синтезе логических схем в заданном базисе логических элементов (например, в базисе И-НЕ или ИЛИ-НЕ) целесообразно предварительно исходное выражение привести к форме, в которой в выражении будут использованы только логические операции, соответствующие используемым логическим эле- ментам в заданном базисе.

Пример

Синтезировать логическую схему в базисе И-НЕ, соответствующую выражению

$$y = \overline{(x1x2 + x1x3 + x2x3)(x1 + x2 + x3)} + x1x2x3.$$

Решение

Используя правило Де Моргана, преобразуем исходное выражение таким образом, чтобы последней операцией было отрицание и в выражение были бы только операции И.

$$\overline{(x1x2 + x1x3 + x2x3)(x1 + x3 + x3)} + x1x2x3 = \overline{\overline{(x1x2 + x1x3 + x2x3)(x1 + x2 + x3)}} + x1x2x3 =$$

$$= \overline{\overline{(x1x2 + x1x3 + x2x3)}\,\overline{(x1x2x3)}} = \overline{(x1x2x1x3x2x3)(x1 + x2 + x3)}\,\overline{x1x2x3} =$$

$$= \overline{x1x2x1x3x2x3}\,\overline{x1x2x3}\,\overline{x1x2x3}.$$

Полученное выражение, представленное в виде вложенных операции И-НЕ, позволяет синтезировать соответствующую логическую схему в заданном базисе, которая приведена на рисунке 3.11.

Пример

Синтезировать логическую схему в базисе ИЛИ-НЕ, соответствующую выражению

$$y = \overline{(x1x2 + x1x3 + x2x3)(x1 + x2 + x3)} + x1x2x3.$$

Решение

Используя правило Де Моргана, преобразуем исходное выражение таким образом, чтобы последней операцией было отрицание и в выражение были бы только операции ИЛИ.

$$y = \overline{\overline{(x1x2+x1x3+x2x3)(x1+x2+x3)}} + x1x2x3 = \overline{\overline{(x1x2+x1x3+x2x3)(x1+x2+x3)}}\,\overline{x1x2x3} =$$

$$= \overline{\overline{(x1x2+x1x3+x2x3)} + \overline{(x1+x2+x3)}} + x1+x2+x3 = \overline{\overline{(x1x2+x1x3+x2x3)} + \overline{(x1+x2+x3)}}\,\overline{x1+x2+x3} =$$

$$= \overline{\overline{x1}\,\overline{x2}\,\overline{x3}\,\overline{x1}\,\overline{x2}\,\overline{x3}\,\overline{x1}\,\overline{x2}\,\overline{x3}}\,\overline{x1+x2+x3}.$$

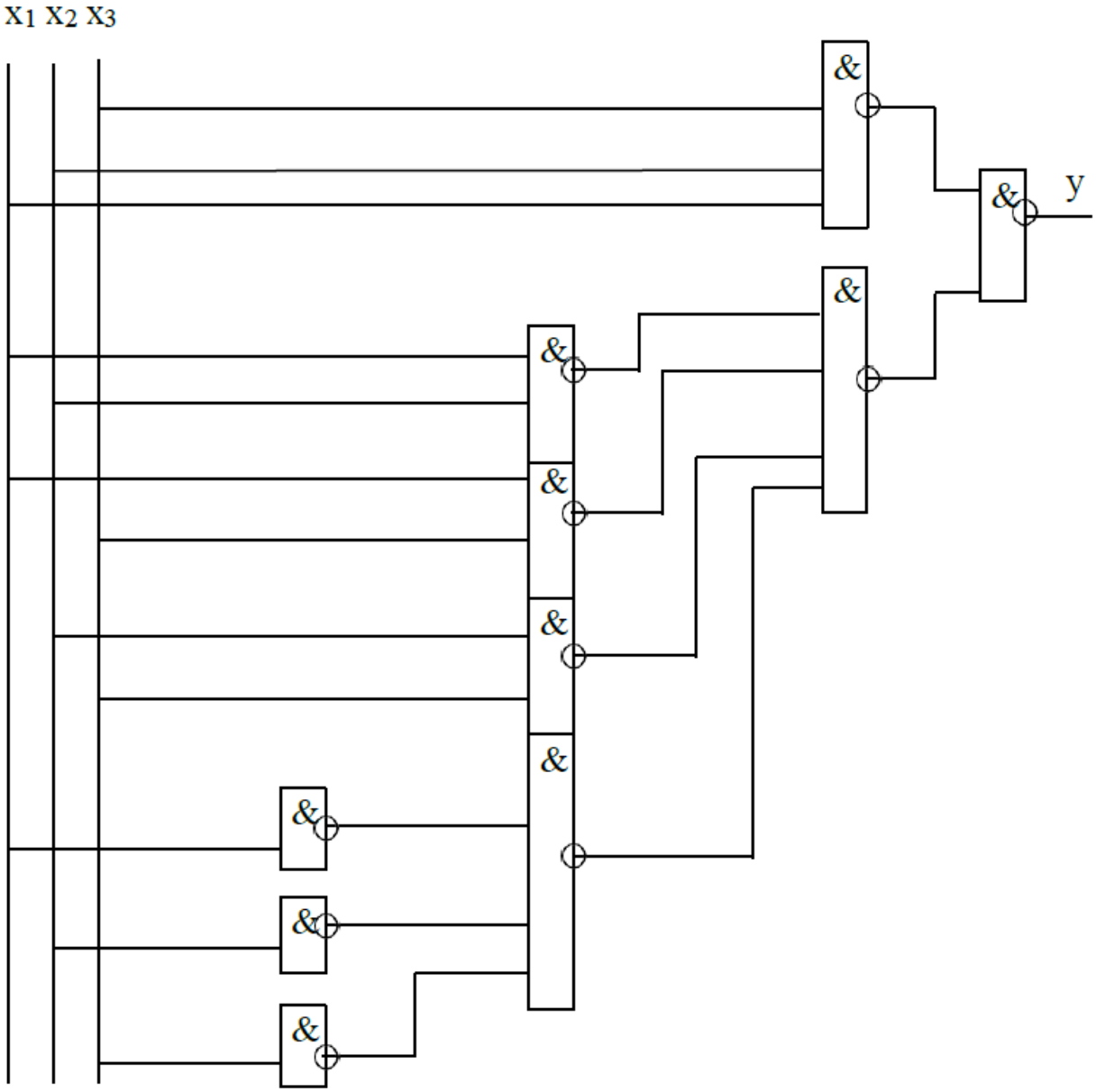


Рисунок 3.11 – Синтезированная схема в базисе И-НЕ

Полученное выражение, представленное в виде вложенных операций ИЛИ-НЕ, позволяет легко синтезировать соответствующую логическую схему в заданном базисе, которая приведена на рисунке 3.12.

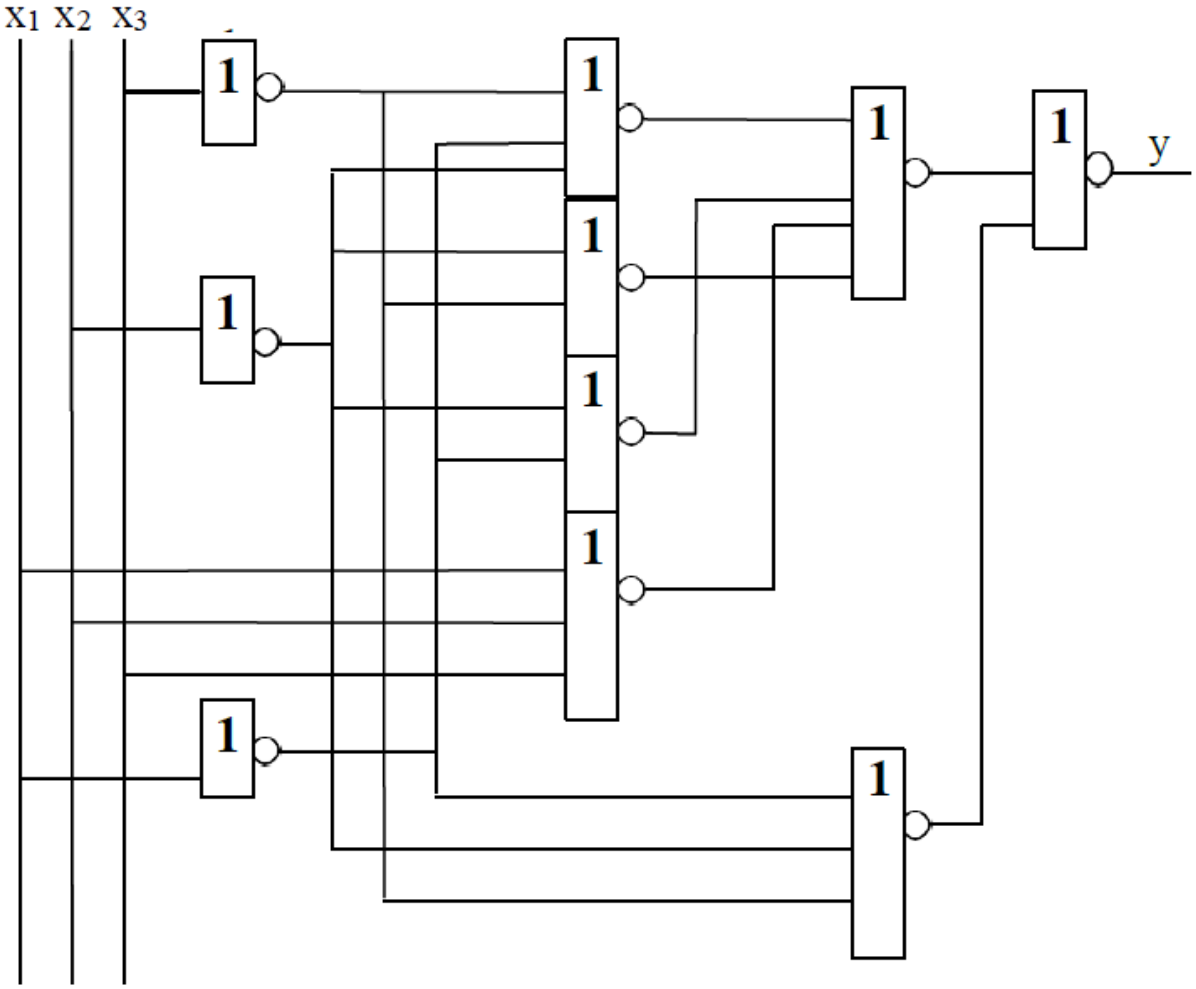


Рисунок 3.12 – Синтезированная схема в базисе ИЛИ-НЕ

IV Схемотехнические основы ЭВМ

1 Цифровые устройства комбинационного типа

Цифровыми устройствами комбинационного типа или цифровыми автоматами без памяти называются цифровые устройства, логические значения на выходе которых однозначно определяются совокупностью или комбинацией сигналов на входах в данный момент времени. К ним относятся суммирующие схемы, шифраторы и дешифраторы, мультиплексоры и демультиплексоры, цифровые компараторы и другие устройства. Цифровые устройства комбинационного типа выпускаются в виде интегральных микросхем или входят в состав больших интегральных микросхем, таких как процессоры, запоминающие и другие устройства.

1.1 Двоичные сумматоры

Одноразрядные сумматоры.

В цифровой вычислительной технике используются одноразрядные суммирующие схемы с двумя и тремя входами, причём первые называются полусумматорами, а вторые — полными одноразрядными сумматорами. Полусумматоры могут использоваться только для суммирования младших разрядов чисел. Полные одноразрядные сумматоры имеют дополнительный третий вход, на который подаётся перенос из предыдущего разряда при суммировании многоразрядных чисел.

На рисунке 4.1а приведена таблица истинности полусумматора, на основании которой составлена его структурная формула в виде СДНФ (рисунок 4.1б). Функциональная схема, составленная на элементах основного базиса в соответствии с этой структурной формулой, приведена на рисунке 4.1в.

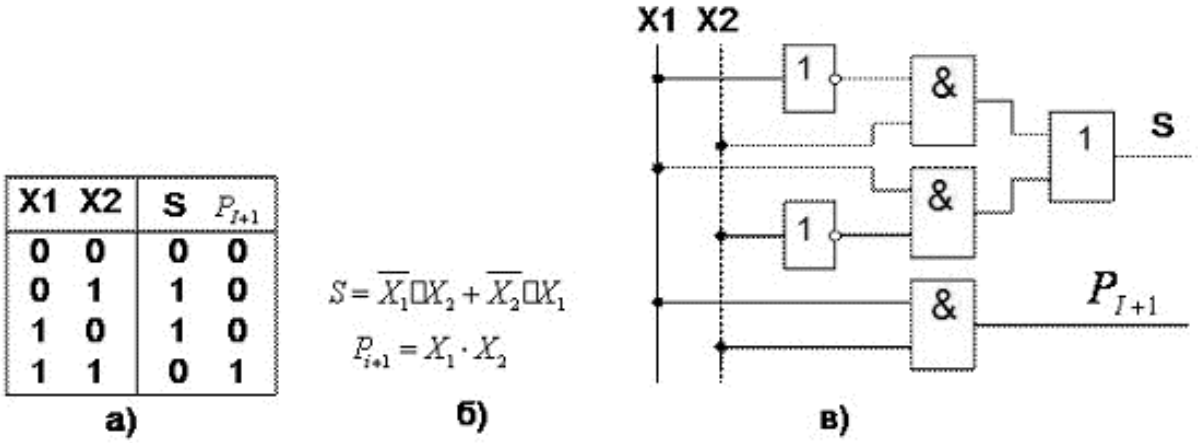


Рисунок 4.1 – Одноразрядный полусумматор (HS): а) таблица истинности, б) структурная формула, в) функциональная схема

Схему полного одноразрядного сумматора можно получить на основе двух схем полусумматоров и схемы «ИЛИ», как показано на рисунке 23,а).

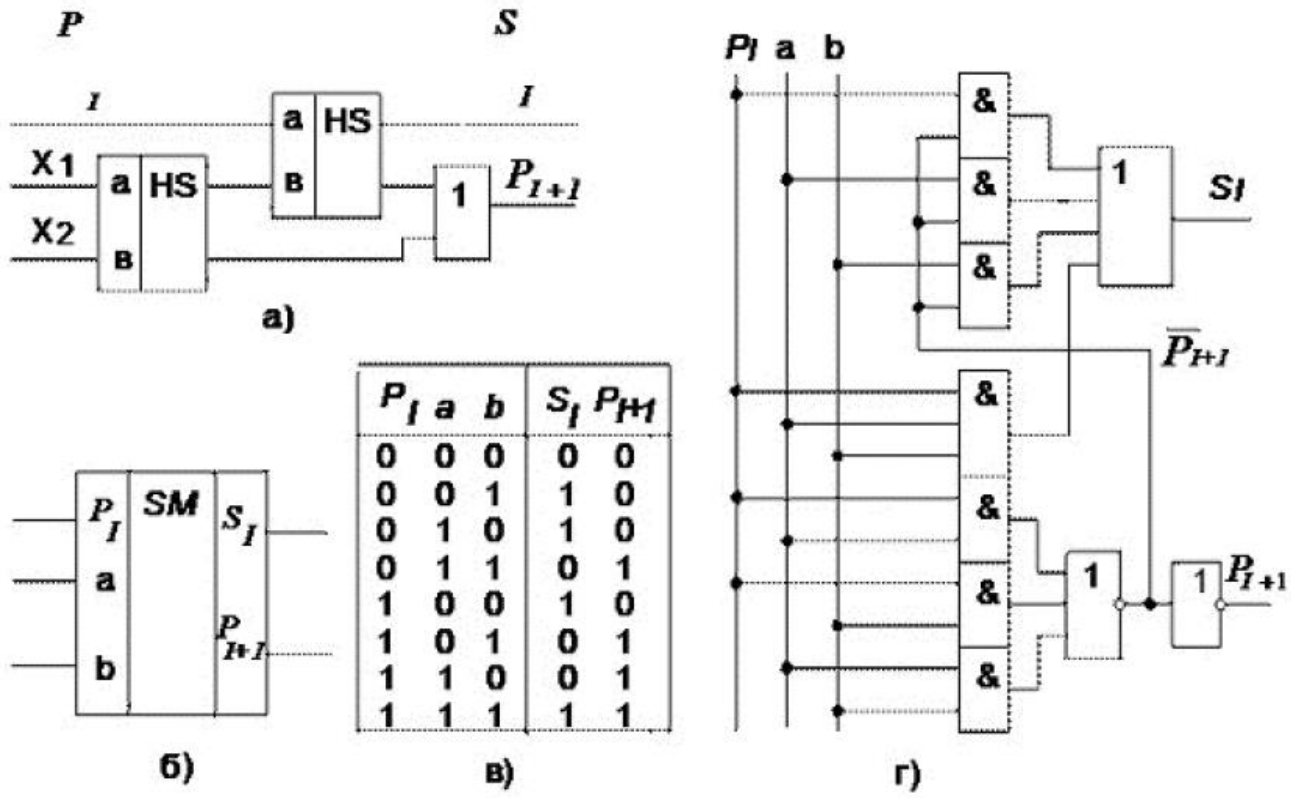


Рисунок 4.2 – Одноразрядный полный сумматор:
а) — функциональная схема на двух полусумматорах HS; б) — условное графическое изображение (УГО); в) — таблица истинности;
г) — минимизированная схема.

Безусловно, на практике в ЭВМ применяются более оптимальные по быстродействию схемы по сравнению с приведенной.

Многоразрядные сумматоры.

Методы построения многоразрядных сумматоров:

- последовательное суммирование;
- параллельное суммирование с последовательным переносом;
- параллельное суммирование с параллельным переносом.

При *последовательном суммировании* используется один сумматор, общий для всех разрядов (рисунок 4.3а).

Операнды должны вводиться в сумматор через входы a_I и b_I синхронно, начиная с младших разрядов. Цепь задержки обеспечивает хранение импульса переноса P_{I+1} на время одного такта, то есть до прихода пары слагаемых следующего разряда, с которыми он будет просуммирован. Задержку выполняет D-триггер. Результаты суммирования также считываются последовательно, начиная с младших разрядов. Для хранения и ввода операндов на входы сумматора, а также для записи результата суммирования обычно используются регистры сдвига.

Достоинство этого метода — малые аппаратные затраты.

Недостаток — невысокое быстродействие, так как одновременно суммируются только пара слагаемых.

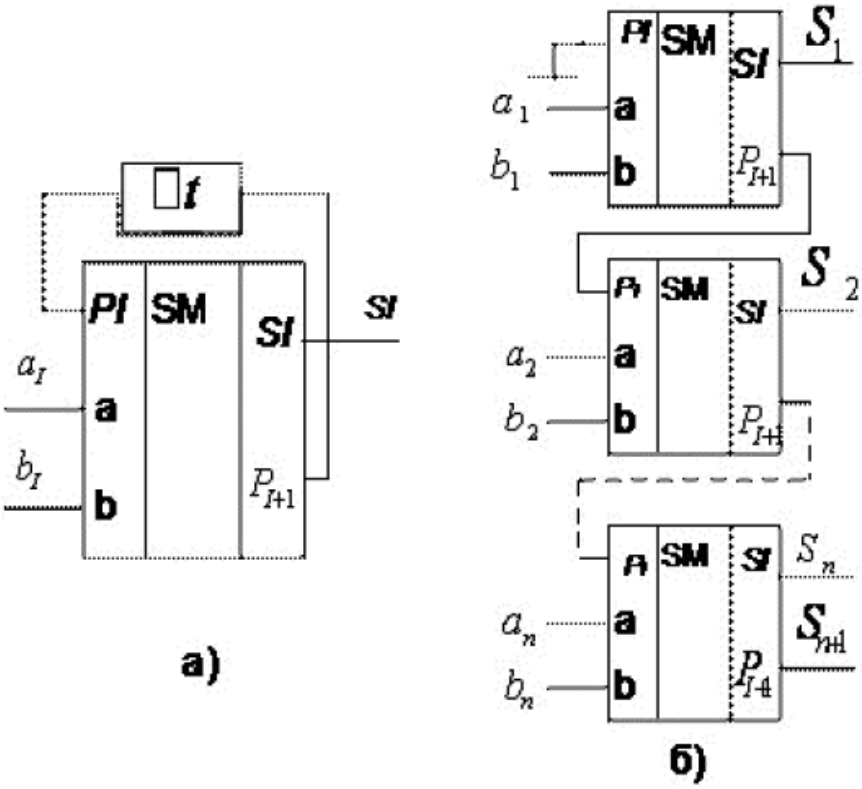


Рисунок 4.3 – Суммирование многоразрядных чисел:
а) — последовательное; б) — параллельное с последовательным переносом

Схема *параллельного сумматора с последовательным переносом* приведена на рисунке 4.3б. Количество сумматоров равно числу разрядов чисел. Выход переноса P_{I+1} каждого сумматора соединяется со входом переноса P_I следующего более старшего разряда. На входе переноса младшего разряда устанавливается потенциал «0», так как сигнал переноса сюда не поступает. Слагаемые a_I и b_I суммируются во всех разрядах одновременно, а перенос P_I поступает с окончанием операции сложения в предыдущем разряде.

Быстродействие таких сумматоров ограничено задержкой переноса, так как формирование переноса на выходе старшего разряда не может произойти до тех пор, пока сигнал переноса не распространится по всей цепочке сумматоров.

Параллельные сумматоры с параллельным переносом. Для организации параллельного переноса применяются специальные узлы — блоки ускоренного переноса.

1.2 Кодирующие и декодирующие устройства

Шифраторы.

Шифратор (кодер) — это функциональный узел, предназначенный для преобразования поступающих на его входы управляющих сигналов (команд) в n-разрядный двоичный код. В частности, такими сигналами или командами могут быть десятичные числа, например, номер команды, который с помощью шифратора преобразуется в двоичный код.

В качестве примера разработаем схему 3-разрядного шифратора. Вначале следует построить таблицу кодов (таблицу истинности), в которой код номера сигнала представим, например, двоичным кодом (рисунок 4.3а). Схема, реализованная на элементах ИЛИ, приведена на рисунке 4.3б.

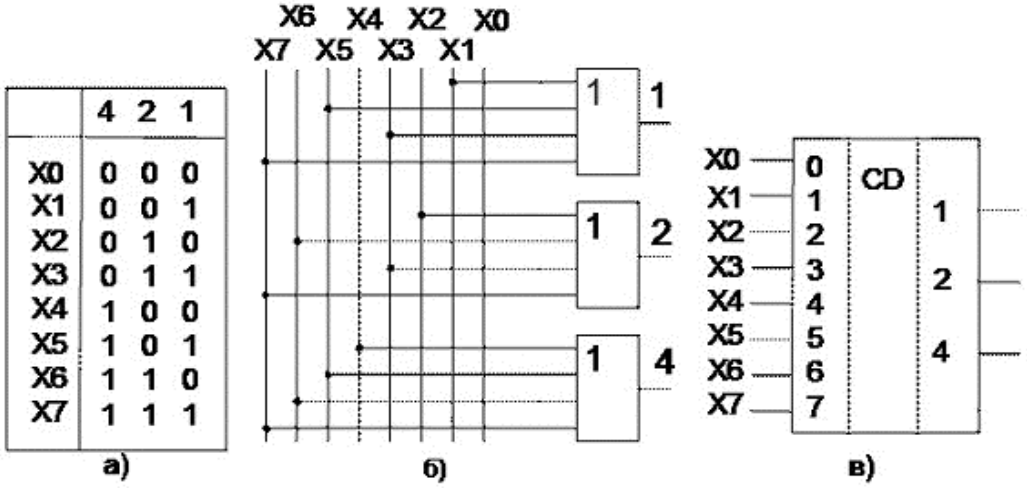


Рисунок 4.3 – Таблица кодов 3-разрядного шифратора а), его функциональная схема б) и УГО в).

В общем случае, при использовании двоичного кода, можно закодировать 2^n входных сигналов. В рассмотренной выше схеме выходной код «000» будет присутствовать на выходе при подаче сигнала на вход X0 и в случае, если входной сигнал вообще не подаётся ни на один из входов. Для однозначной идентификации сигнала X0 в интегральных схемах формируется ещё один выходной сигнал — признак подачи входного сигнала, который используется и для других целей.

Дешифраторы (декодеры).

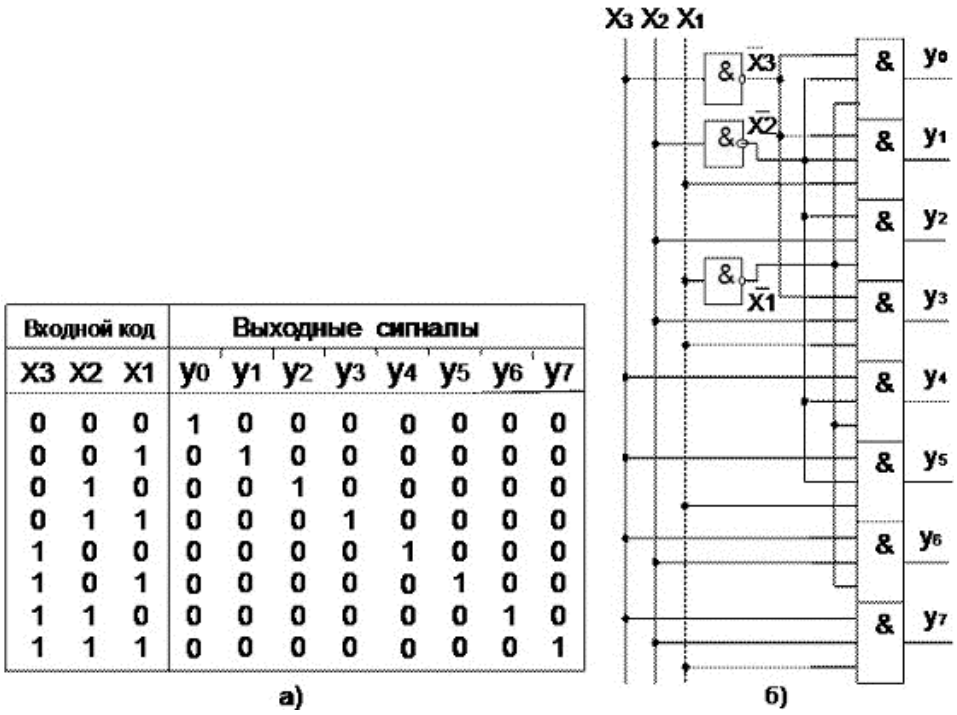
Дешифратор — функциональный узел, вырабатывающий сигнал «лог. 1» (дешифратор высокого уровня) или сигнал «лог. 0» (дешифратор низкого уровня) только на одном из своих 2^n выходах в зависимости от кода двоичного числа на n входах (рисунок 4.4).

Дешифраторы широко используются в устройствах управления, где они формируют управляющий сигнал в соответствии с входным кодом, который воздействует на какое-либо исполнительное устройство.

1.3 Коммутаторы цифровых сигналов

Мультиплексоры.

Мультиплексор — функциональный узел, который имеет n адресных входов, $N=2^n$ информационных входов, один выход и осуществляет управляемую коммутацию информации, поступающей по N входным линиям, на одну выходную линию. Коммутация определённой входной линии происходит в соответствии с двоичным адресным кодом $a_{n-1},\ldots a_2,a_1,a_0$.



Входной код			Выходные сигналы							
X3	X2	X1	y0	y1	y2	y3	y4	y5	y6	y7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Рисунок 4.4 – Дешифратор: а) – таблица истинности;
б) – функциональная схема

Если адресный код имеет n разрядов, то можно осуществить $N=2^n$ комбинаций адресных сигналов, каждая из которых обеспечит подключение одной из N входных линий к выходной линии. Такой мультиплексор называют «из N в одну». При наличии избыточных комбинаций адресных сигналов можно спроектировать мультиплексор с любым числом входных линий $N \le 2^n$.

В простейшем случае при двухразрядном адресном коде (n=2) максимальное число входных адресных линий равно $N=2^n=4$. Таблица истинности такого мультиплексора приведена на рисунке 4.5а.

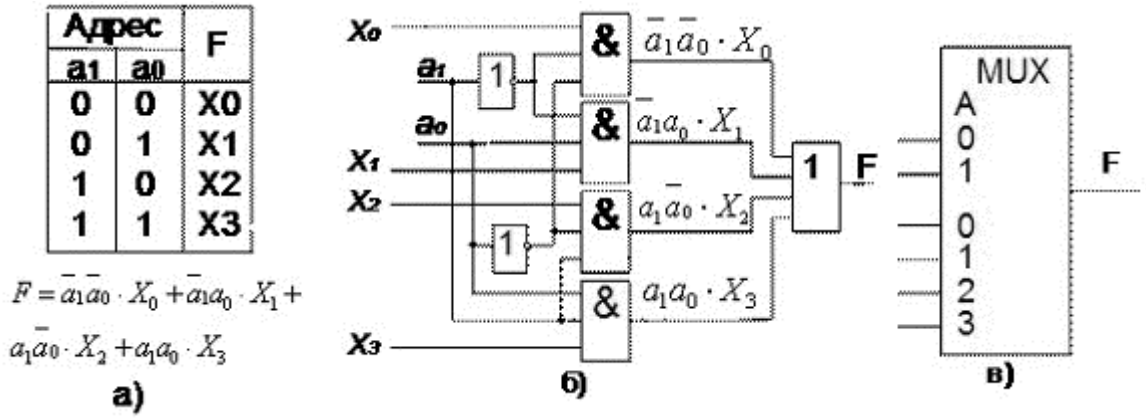


Рисунок 4.5 – Мультиплексор 4:1 а) — таблица истинности;
б) — функциональная схема; в) — условное графическое обозначение

Характеристическое уравнение такого мультиплексора, записанное в соответствии с таблицей истинности, имеет вид:

$$F = \bar{a}_1\bar{a}_0 \cdot X_0 + \bar{a}_1 a_0 \cdot X_1 + a_1\bar{a}_0 \cdot X_2 + a_1 a_0 \cdot X_3$$

Из полученного уравнения следует, что в состав функциональной схемы мультиплексора входят два инвертора, четыре схемы «И» и одна схема «ИЛИ» (рисунок 4.5б). Здесь адресными (управляющими) входами являются a_1, a_0, а информационными — *X0, X1, X2, X3*.

Условное графическое обозначение мультиплексора приведено на рисунке 4.5в.

1.4 Дешифраторы-демультиплексоры

Демультиплексор — это функциональный узел, осуществляющий управляемую коммутацию информацию, поступающую по одному входу, на N выходов. Таким образом, демультиплексор реализует операцию, противоположную той, которую выполняет мультиплексор.

Обобщённая схема демультиплексора приведена на рисунке 4.6. В общем случае число выходных линий N определяется количеством адресных входов n и равно $N=2^n$.

Для случая n=2 функционирование демультиплексора осуществляется в соответствии с таблицей истинности, приведённой на рисунке 4.7а.

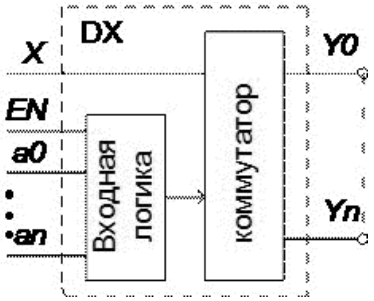
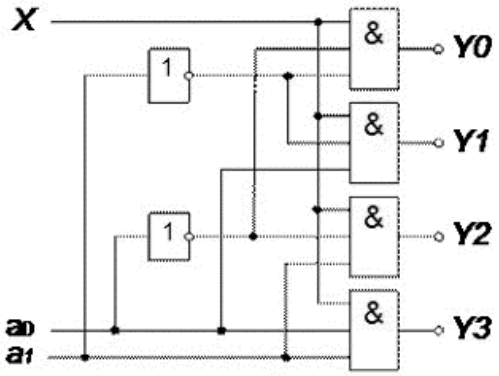


Рисунок 4.6 – Обобщённая схема демультиплексора

адрес		Y0	Y1	Y2	Y3
a1	a0				
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

а)



б)

Рисунок 4.7 – Таблица истинности — а) и функциональная схема 4-канального демультиплексора — б)

Из таблицы истинности записываем характеристические уравнения демультиплексора:

$$Y_0 = \bar{a}_1\bar{a}_0 \cdot X; \quad Y_1 = \bar{a}_1 a_0 \cdot X;$$

$$Y_2 = a_1\bar{a}_0 \cdot X; \quad Y_3 = a_1 a_0 \cdot X.$$

Соответствующая этим уравнениям функциональная схема демультиплексора приведена на рисунке 4.7б. Она имеет в своём составе два инвертора и четыре элемента «И».

Сравнивая таблицы истинности и функциональные схемы демультиплексора и дешифратора, легко увидеть схожесть их функций. Если функция X=1 постоянно, то демультиплексор выполняет функции дешифратора. Таким образом, данные схемы могут выполнять функции и дешифратора и демультиплексора.

2 Триггеры

В качестве запоминающего элемента в вычислительной технике используется триггер.

Триггер в общем случае обладает следующими свойствами:

триггер имеет два устойчивых состояния, которые называются состоянием «0» и состоянием «1»;

- триггер имеет парафазный выход, который представляется двумя выходами, всегда имеющими противоположные значения, при этом один выход имеет название «выход единицы», а другой – «выход нуля»;

- триггер имеет управляющий вход (или входы), подавая сигналы на который, можно менять состояние триггера.

Используются следующие типы триггера:

- RS- триггер;
- D -триггер;
- T -триггер;
- JK- триггер.

2.1 RS-триггер

Отличительной особенностью RS-триггера является наличие у него входа установки значения единицы (вход «S») и входа установки значения ноль (вход «R»). Своё название этот триггер получил по названию своих входов.

RS-триггер может быть реализован или в логическом базисе ИЛИ -НЕ, или в логическом базисе И -НЕ (рисунок 4.8).

Наличие в логической схеме обратных связей приводит к тому, что значения выходных переменные приведенной схемы зависят не только от входных переменных, но и от начального состояния схемы, т.е. от состояния триггера на момент поступления входных сигналов.

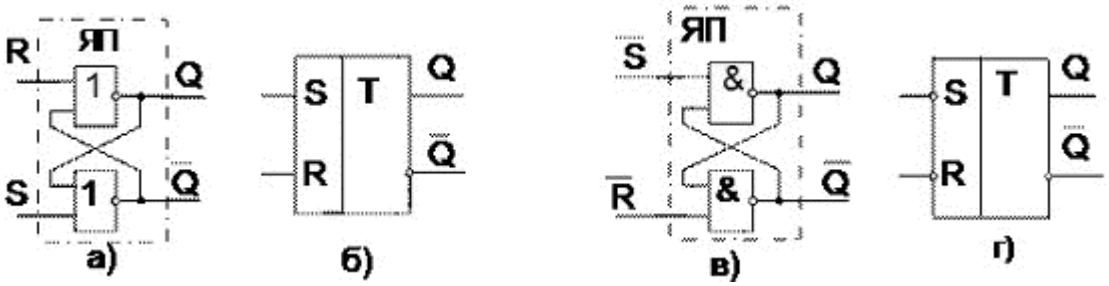


Рисунок 4.8 – RS-триггеры: а), б) — на логических элементах ИЛИ-НЕ, в), г) — на логических элементах И-НЕ

Доказательство того, что приведенная схема является триггером, имеющим вход для установки единицы и вход для установки нуля, приведено в виде таблицы 4.1. В таблице используются обозначения Q(t) и Q(t+1), которые отражают, соответственно, начальное и конечное значения на выходе Q триггера перед началом и после воздействия входных сигналов.

Таблица 4.1 – Состояние триггера при различных воздействиях входных сигналов S и R

S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	не определено
1	1	1	не определено

2.2 Синхронный RS-триггер

В устройствах современной цифровой техники, для исключения опасных состязаний входных сигналов, срабатывание всех узлов и элементов в каждом такте должно происходить строго одновременно. Для достижения этой цели применяется жёсткая синхронизация с помощью специальных синхроимпульсов. Для работы в схемах с синхронизацией режима разработаны синхронные RS-триггеры.

Кроме трёх основных входов, синхронные RS-триггеры снабжаются ещё входами асинхронной установки состояния триггера — Ś и Ŕ. Они предназначены для подачи приоритетных сигналов установки триггера в исходное состояние («0» или «1») в начале цикла работы независимо от воздействия сигналов на входах S и R, то есть в обход схемы управления.

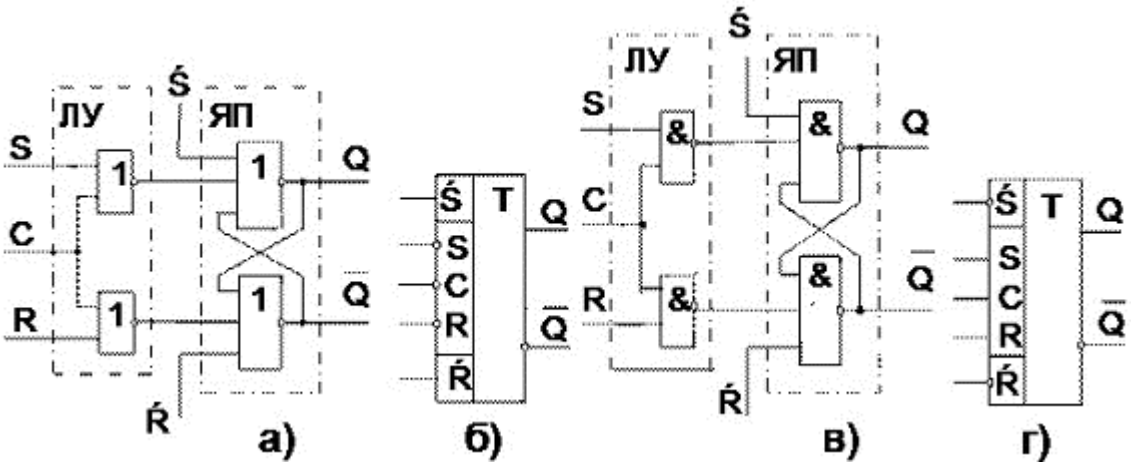


Рисунок 4.9 – Синхронные RS-триггеры: — а) на элементах ИЛИ-НЕ, — в) на элементах И-НЕ и их УГО б), и г)

2.3 Двухтактный RS-триггер

Двухтактный RS-триггер характеризуется тем, что у него разделены момент восприятия входных сигналов и момент изменения выходного сигнала в соответствии с комбинацией действующих входных сигналов. Построение такого триггера на базе обычного (однотактного) синхронного RS- триггер приведено на рисунке 4.10.

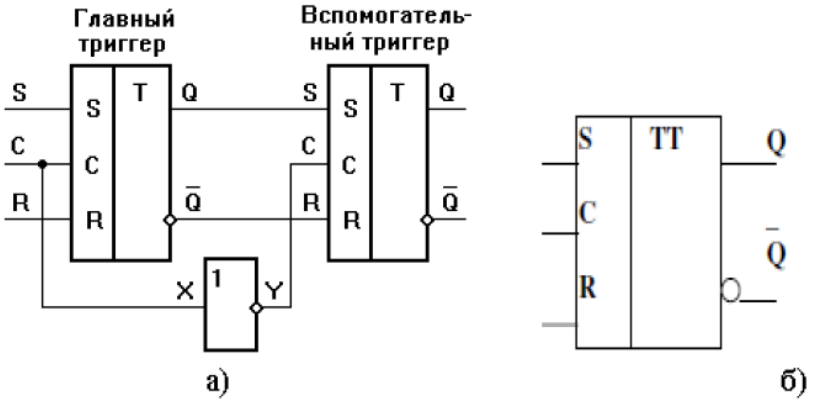


Рисунок 4.9 – Двухтактный RS-триггер – а) и его УГО – б)

При наличии сигнала на первом входе С имеется разрешение по входу синхронизации первого триггера реагировать на входные сигналы S и R. В то же самое время, пока есть «1» на входе С первого триггера, запрещена реакция второго триггера на сигналы, действующие на его входах R и S. Входными сигналами для второго триггера являются сигналы на выходах первого триггера. Как только на входе С первого триггера появляется низкий уровень (логический ноль), разрешается восприятие вторым триггером своих входных сигналов и запрещается реакция на входы для первого триггера. В результате второй триггер установится в состояние, соответствующее состоянию первого триггера.

2.4 Т-триггер

Т-триггер представляет собой триггер, имеющий один вход «Т», поступление единичного сигнала на который переводит Т-триггер в состояние, противоположное его исходному состоянию (фигурально говоря, по каждому входному сигналу триггер «кувыркается», меняя своё состояние на противоположное). На рисунке 4.10 приведена реализация Т-триггера на базе двухтактного RS-триггера.

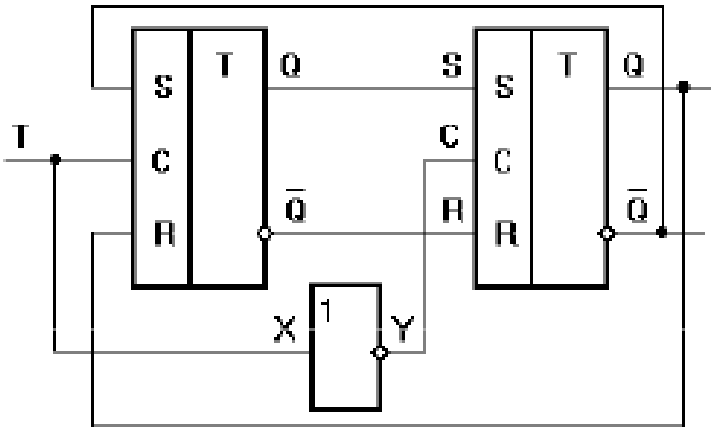


Рисунок 4.10 – Т-триггер

Имеющиеся на схеме обратные связи создают ситуацию, при которой сигналы на входах R и S стремятся перевести триггер в состояние, противоположное текущему. Поэтому при приходе очередного сигнала Т триггер воспринимает имеющиеся сигналы на его входах. Выходные сигналы триггера изменяются после снятия единичного сигнала на его входе Т, так как триггер двухтактный.

2.5 JK-триггер

Реализация JK-триггера приведена на рисунке 4.11. Вход «J» - это вход установки «1», вход «K» - вход установки «0».

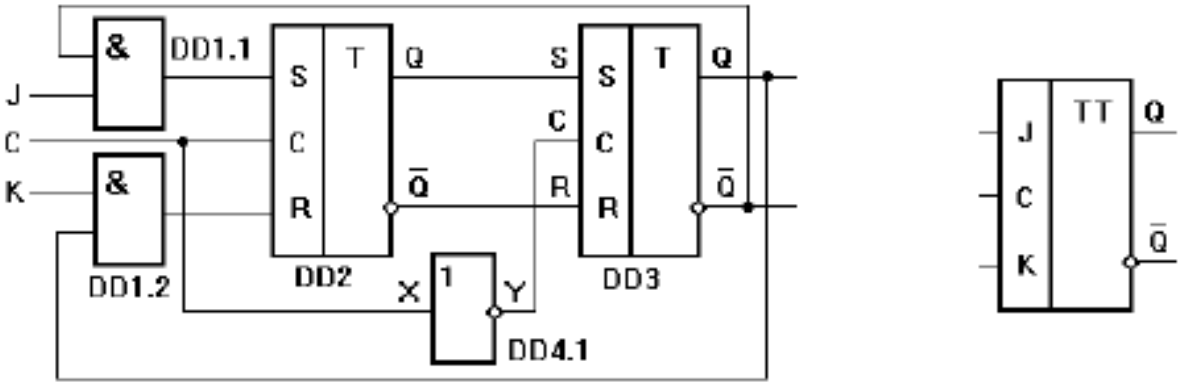


Рисунок 4.11 – JK-триггер

Из приведенной схемы видно, что сигналы J или K, стремящиеся установить триггер, соответственно, в «1» или «0», поступают на соответствующий вход «S» или «R» базового RS-триггера только тогда, когда его исходное состояние противоположно тому, в которое стремится перевести JK-триггер комбинация входных сигналов. В противном случае сигнал J или K на соответствующий вход S или R базового триггера не поступают. В связи с этим комбинация входных сигналов «1, 1» не является запретной, так как в этом случае на соответствующий вход базового RS-триггера поступит только тот сигнал, который стремится установить триггер в состояние, противоположное его исходному состоянию. Этот момент отражен в таблице истинности – при комбинации входных сигналов «1,1» триггер меняет исходное состояние, то есть работает как Т-триггер.

2.6 D-триггер

D–триггер по-другому называют элементом задержки. Схема его реализации на базе RS - триггера приведена на рисунке 4.12.

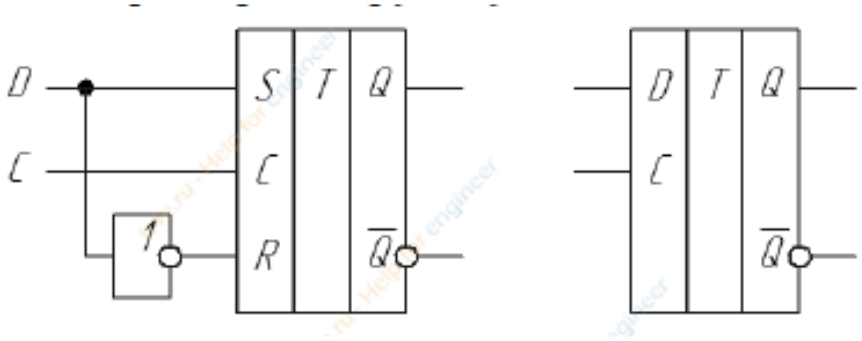


Рисунок 4.12 – D-триггер

Это означает, что, когда есть «1» на входе D, на входах базового триггера будут сигналы: «1» - на входе S и «0» - на входе «R». Поэтому по переднему фронту сигнала синхроимпульса в триггере устанавливается «1», если есть «1» на входе D, в противном случае в триггере будет установлен «0». Состояние, которое имеется у триггера в момент заднего фронда сигнала СИ, будет сохраняться («задерживаться») до поступления очередного сигнала синхроимпульса.

3 Регистры

Регистры — это функциональные узлы на основе триггеров, предназначенные для приёма, кратковременного хранения (на один или несколько циклов работы данного устройства), передачи и преобразования многоразрядной цифровой информации.

В зависимости от способа записи информации (кода числа) различают параллельные, последовательные и параллельно — последовательные регистры.

3.1 Параллельные регистры (регистры памяти)

Запись кода в параллельные регистры осуществляется параллельным кодом, то есть во все разряды регистра одновременно. Их функция сводится только к

приёму, хранению и передаче информации. В связи с этим параллельные регистры называют регистрами памяти.

Параллельный N-разрядный состоит из N триггеров, объединённых общими цепями управления.

В качестве примера на рисунке 4.13а приведена схема 4-разрядного параллельного регистра, построенного на RS-триггерах D5…D8. Элементы D1…D4 образуют цепь управления записью, а элементы D9…D12 — цепь управления чтением.

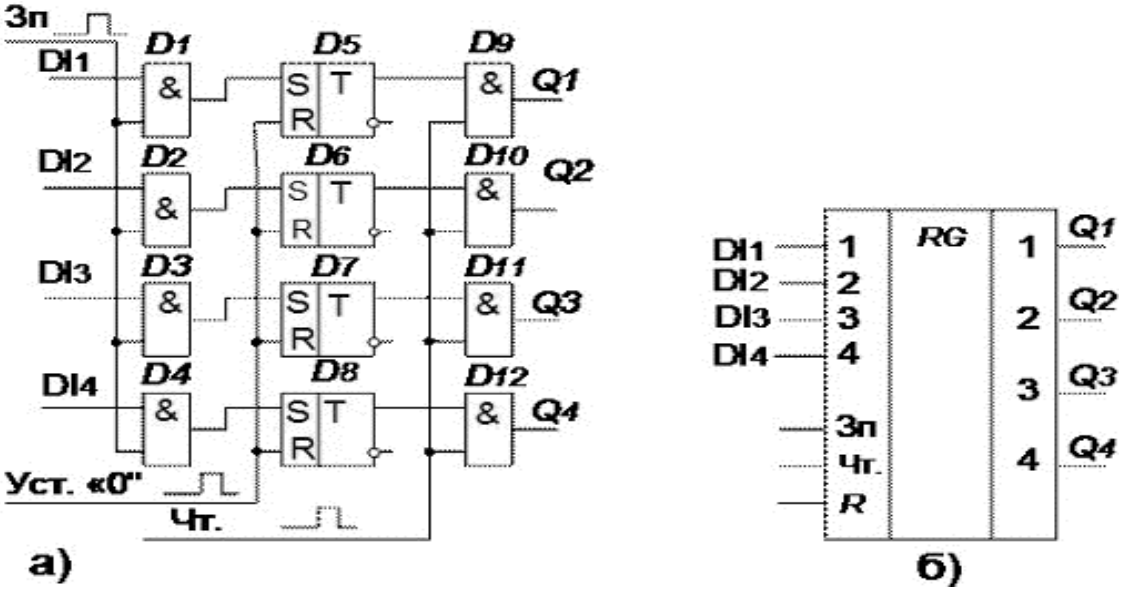


Рисунок 4.13 – Функциональная схема а) и УГО б) параллельного регистра

Перед записью информации все триггеры регистра устанавливают в состояние «0» путём подачи импульса «1» на их R-входы.

Записываемая информация подаётся на входы DI1…DI4. Для записи информации подаётся импульс «Зп», открывающий входные элементы «И». Код входного числа записывается в регистр. По окончании импульса «Зп» элементы D1…D4 закрываются, а информация, записанная в регистр, сохраняется несмотря на то, что входная информация может изменяться.

Для считывания информации подают сигнал «1» на вход «Чт». По этому сигналу на выходные шины регистра на время действия сигнала передаётся код числа, записанный в регистр. По окончанию операции чтения выходные ключи закрываются, а информация, записанная в регистр, сохраняется. То есть возможно многократное считывание информации. Условное графическое обозначение параллельного регистра приведено на рисунке 4.13б.

3.2 Регистры сдвига

Регистры сдвига представляют собой цепочку последовательно включённых D-триггеров или RS- и JK-триггеров, включённых в режим D-триггера. Появление импульса на тактовом входе регистра сдвига вызывает перемещение записанной в нём информации на один разряд вправо или влево. Как и другие регистры, регистры

сдвига используются для записи, хранения и выдачи информации, но основным их назначением является преобразование последовательного кода в параллельный или параллельного в последовательный.

Схема 4-разрядного регистра сдвига приведена на рисунке 4.14. Схема работает следующим образом. Благодаря тому, что выход предыдущего разряда соединён со входом «D» последующего, каждый тактовый импульс устанавливает последующий триггер в состояние, в котором до этого находился предыдущий. Так осуществляется сдвиг информации вправо.

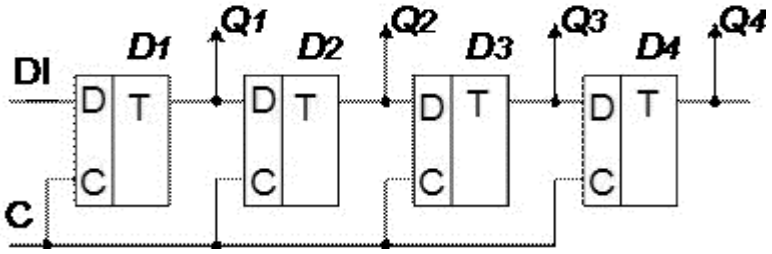


Рисунок 4.14 – 4-разрядный регистр сдвига

Вход «D» первого триггера служит для приёма в регистр входной информации DI в виде последовательного кода. С каждым тактовым импульсом на этот вход должен подаваться код нового разряда входной информации.

Запись параллельного кода информации может быть произведена через нетактируемые установочные входы $\overline{S}$, $\overline{R}$ триггеров (на рисунке 4.14 не показаны).

С выхода «Q_4» последнего триггера снимается последовательный выходной код. Код на этом выходе регистра появляется с задержкой относительно входного последовательного кода на число периодов тактовых импульсов, равное числу разрядов регистра.

Параллельный выходной код можно снять с выходов $Q_1 \ldots Q_4$ всех триггеров регистра сдвига, снабдив их выходными ключами, подобными выходным ключам параллельного регистра.

4 Счётчики импульсов

4.1 Требования, предъявляемые к счётчикам

В устройствах цифровой обработки информации измеряемый параметр (угол поворота, скорость, давление и т. п.) преобразуются в импульсы напряжения, число которых в соответствующем масштабе характеризует значение данного параметра. Эти импульсы подсчитываются *счётчиками импульсов* и выражаются в виде цифр.

Основными показателями счётчиков являются ёмкость и быстродействие.

Ёмкость, численно равная *КСЧ*, характеризует число импульсов, доступное счёту за один цикл. Как уже было показано выше, ёмкость определяется количеством разрядов счётчика.

Быстродействие или максимально возможная скорость работы оценивается двумя параметрами:

- *разрешающая способность* $t_{раз.сч}$ — минимальное время между двумя входными сигналами, в течение которого ещё не возникают сбои в работе счётчика. Величина, обратная разрешающей способности, называется *максимальной частотой счёта* f_{max}. f_{max} , определяет количество импульсов, которое может подсчитать счётчик за 1 сек:

$f_{max} = 1/t_{раз.сч}$;

- *время установки кода счётчика* $t_{уст}$ — это время между моментом прихода входного сигнала и переходом счётчика в новое устойчивое состояние.

Для удовлетворения потребностей разработчиков цифровых электронных устройств различного назначения разработаны интегральные микросхемы счётчиков с широким спектром параметров. Всё многообразие счётчиков можно классифицировать по следующим признакам.

1. По направлению счёта:
- суммирующие;
- вычитающие;
- реверсивные.
2. По коэффициенту счёта:
- двоичные;
- двоично-десятичные (декадные);
- с постоянным коэффициентом счёта;
- с переменным коэффициентом счёта.
3. По способу организации внутренних связей:
- с последовательным переносом;
- с параллельным переносом;
- с комбинированным переносом;
- кольцевые.

Классификационные признаки независимы и могут встречаться в разных сочетаниях. Например, суммирующие счётчики могут быть как с последовательным, так и с параллельным переносом и могут иметь двоичный или десятичный коэффициент счёта.

4.2 Суммирующие счётчики

Простейшим счётчиком является Т-триггер, считающий до 2-х, то есть осуществляющий счёт и хранение не более 2-х сигналов.

Счётчик, образованный цепочкой из n триггеров сможет подсчитать в двоичном коде 2^n импульсов. Число n определяет количество разрядов двоичного числа, которое может быть записано в счётчик. Число 2^n называется *модулем* или *коэффициентом счёта*:

$K_{СЧ} = 2^n$

Схема простейшего 4-х разрядного счётчика приведена на рисунке 4.15а. Принцип работы счётчика проиллюстрирован временными диаграммами, приведёнными на рисунке 4.15б.

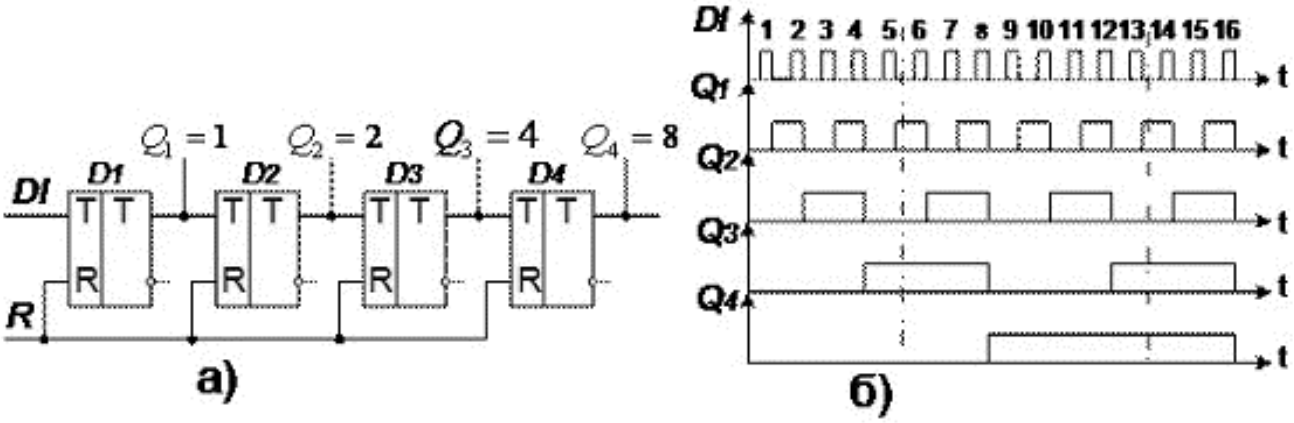


Рисунок 4.15 – Схема двоичного суммирующего счётчика а)
и временные диаграммы его работы б).

Первый разряд счётчика переключается с приходом каждого входного импульса, что соответствует алгоритму работы Т-триггера. На каждые два входных импульса Т-триггер формирует один выходной импульс.

Второй разряд переключается в состояние «1» после прихода каждого 2-го импульса. *Третий разряд* — после прихода каждого 4-го импульса. *Четвёртый разряд* — после прихода каждого 8-го импульса.

Таким образом, единичные значения сигналов на выходах триггеров регистра появляются с приходом 1, 2, 4, 8 импульсов, что соответствует весовым коэффициентам двоичного кода. Поэтому с выходов триггеров регистра можно прочитать параллельный двоичный код числа импульсов, поступивших на его вход. Например, после прихода 5 импульсов единичные значения установятся на выходах Q_1 и Q_3 (см. пунктирную линию на рисунке 60,б), что соответствует коду числа 5 или в двоичном виде 0101. Аналогично, после прихода 13-и импульсов на выходах триггеров установится двоичный код 1101.

Если число входных импульсов $N_{ВХ}>K_{СЧ}$, то при $N_{ВХ}=K_{СЧ}$ происходит переполнение счётчика, после чего счётчик возвращается в нулевое состояние и повторяет цикл работы.

После каждого цикла счёта на выходе последнего триггера возникают перепады напряжения, то есть формируется один импульс. Это свойство определяет второе назначение счётчиков — деление числа входных импульсов.

Если входные сигналы периодичны и следуют с частотой $f_{ВХ}$, то частота $f_{ВЫХ}$:

$f_{ВЫХ} = f_{ВХ} / K_{СЧ.}$

В этом случае коэффициент счёта определяется как коэффициент деления и обозначается $K_{ДЕЛ}$.

У счётчика в режиме деления частоты используется сигнал только последнего триггера, а промежуточные состояния остальных триггеров не учитываются.

Всякий счётчик может быть использован как делитель частоты.

4.3 Вычитающие и реверсивные счётчики

Реверсивный счётчик может работать в качестве суммирующего и вычитающего.

Суммирующий счётчик, как было показано выше, получается при подсоединении к входу последующего каскада прямого выхода предыдущего.

Каждый входной импульс увеличивает число, записанное в счётчик, на 1. Перенос информации из предыдущего разряда в последующий происходит при смене состояния предыдущего разряда (триггера) с 1 на 0.

Вычитающий счётчик получается при подсоединении к входу последующего каскада инверсного выхода предыдущего. Он действует обратным образом: двоичное число, хранящееся в счётчике, с каждым поступающим импульсом уменьшается на 1.

Перенос из младшего разряда в старший имеет место при смене состояния мл адшего разряда с 0 на 1.

Переполнение происходит после достижения счётчиком нулевого состояния, при этом в счётчик записывается максимально возможное значение, т.е. во все разряды — единицы.

Путём включения в схему двоичного суммирующего счётчика (рисунок 4.16), дополнительных логических элементов, переключающих на вход последующего триггера прямого и инверсного выходов предыдущего, получается схема реверсивного счётчика. Фрагмент схемы реверсивного счётчика приведён на рисунке 4.16.

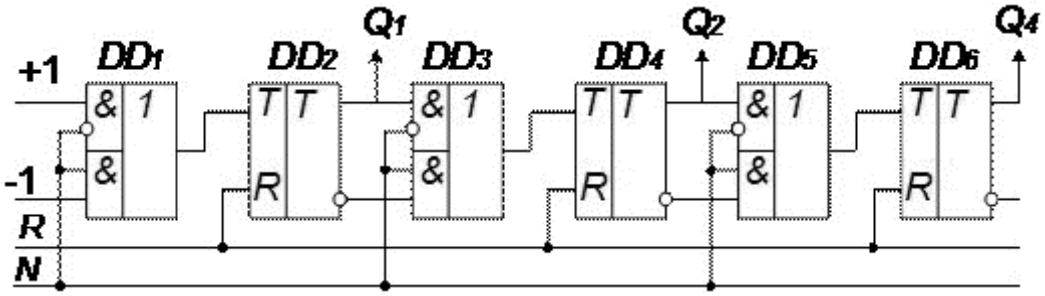


Рисунок 4.16 – Фрагмент схемы реверсивного счётчика

Схема имеет два входа для подачи входных сигналов: +1 — при работе в режиме суммирования, -1 — при работе в режиме вычитания. Дополнительный управляющий вход N задаёт направление счёта. При N=0 схема (рисунок 61) работает как суммирующий счётчик, а при N=1 — как вычитающий.

4.4 Счётчики с произвольным коэффициентом счёта

В двоичных счётчиках коэффициент счёта $K_{СЧ}=2^n$ и может быть равен 2, 4, 8, 16, 32 и т.д. На практике требуются счётчики с коэффициентом счёта не равным 2^n, например, 3, 6, 10, 12, 24 и др.

Они выполняются на основе двоичных счётчиков путём исключения у счётчиков с $K_{СЧ}=2^n$ соответствующего числа «избыточных» состояний S:

$S = 2^n - K_{СЧ.}$

Например, двоично-десятичный (декадный) счётчик получают из 4-х разрядного, имеющего $K_{СЧ}=16$, исключая 6 состояний.

Возможны 2 варианта построения схем:

а) Счёт циклически идёт от 0000 до 1001, а следующим импульсом обнуляется;

б) Исходным состоянием служит код 0110 числа 6 и счёт происходит до 1111_2=15, а следующим импульсом обнуляется.

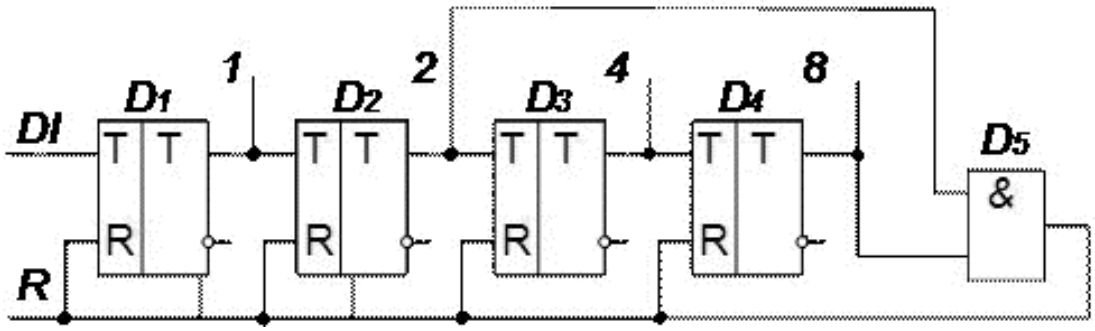


Рисунок 4.17 – Схема счётчика с *Ксч* =10

Схема счётчика с $K_{СЧ}$=10, реализованная по первому варианту, приведена на рисунке 4.17. По сравнению со схемой двоичного счётчика (Рисунок 60), имеющего $K_{СЧ}$=24=16, в схему дополнительно введён элемент D5, обнуляющий счётчик при совпадении двух «1» с весовыми коэффициентами 2 и 8. Использование приведённой выше схемы и логического элемента D5 с 4 входами, позволит получить счётчик с любым коэффициентом счёта от 2 до 15.

Для реализации схемы по второму варианту используются триггеры, имеющие входы асинхронной установки триггера $\overline{S}$.

5 Арифметико-логическое устройство ЭВМ

Арифметико-логическое устройство ЭВМ (АЛУ) служит для выполнения логических и арифметических операций. Структурная схема АЛУ представлена на рисунке 4.18. АЛУ можно разделить на два блока:

– управляющий блок (Упр. АЛУ);

– операционный блок.

Операционный блок состоит из следующих типовых узлов:

– регистры (*R*), служащие для хранения операндов и результатов;

– сумматор (*SM*), служащий для выполнения операции суммирования многоразрядных кодов;

– операционные узлы (ОУ), служащие для выполнения логических операций;

– мультиплексор (*MS*);

– счетчик (Сч), обеспечивающий подсчет тактов длинных операций;

– регистр флажков (*RF*), служащий для фиксации особой информации, характеризующей полученный результат.

Для передачи информации между отдельными узлами используются шины Ш1–Ш3. Шина Ш3 обеспечивает также связь с запоминающими устройствами (ЗУ) ЭВМ.

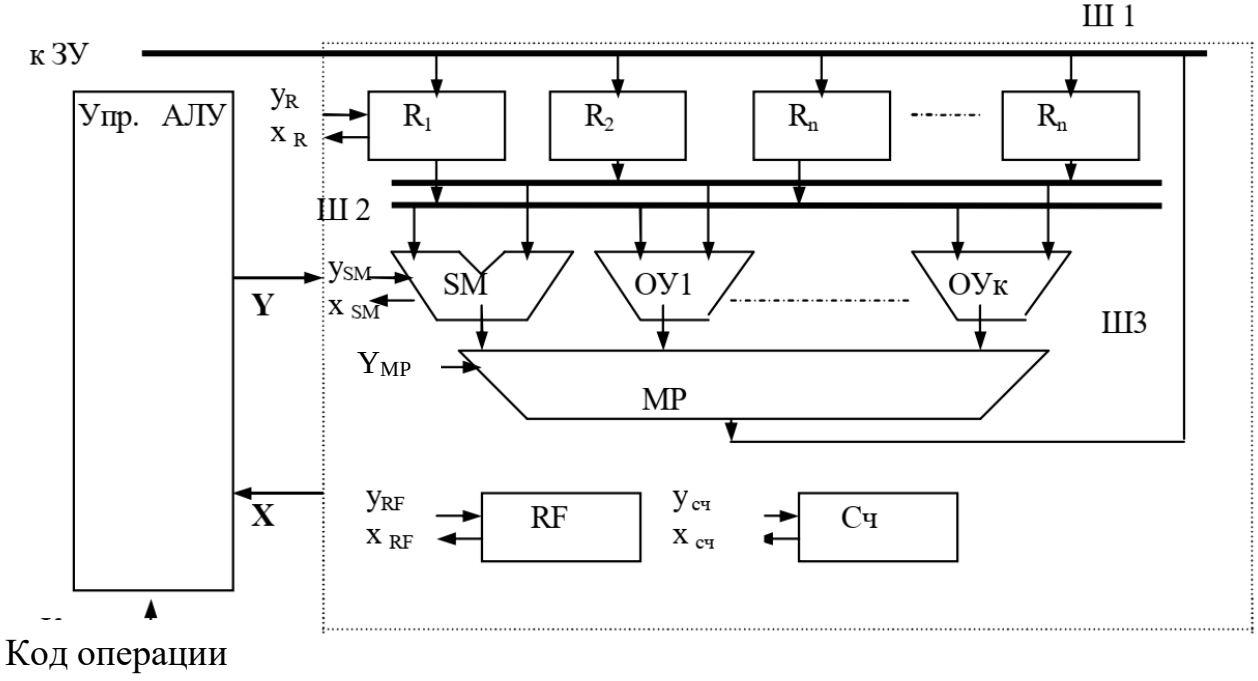


Рисунок 4.18 – Структурная схема АЛУ представлена на

Управляющий блок осуществляет выработку множества управляющих сигналов *Y*, обеспечивающих выполнение элементарных операций (микроопераций) типовыми узлами операционного блока.

При работе управляющая часть АЛУ использует код заданной операции (например, сложение, умножение, вычитание и т. п.), а также информацию о состоянии операционного блока, представленную в виде множества *X* признаков, формируемых типовыми узлами.

К признакам, вырабатываемым регистром и посылаемым в управляющую часть, относятся:

– «ноль регистра» ($R\{0...n\} = 0$) характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «ноль знака» ($R\{зн\} = 0$) – в знаковом разряде регистра находится значение 0;

– «единица старшего разряда» ($R\{1\} = 1$) – в старшем разряде регистра находится значение единица;

– «единица младшего разряда» ($R\{n\} = 1$) – в младшем разряде регистра находится значение единица.

К микрооперациям, которые может выполнять регистр при поступлении соответствующего управляющего сигнала y_i, относятся:

– прием кода;

– выдача прямого кода;

– выдача инверсного кода;

– установка единицы в некотором разряде регистра;

– обнуление знакового разряда;

– сдвиг кода влево;

– сдвиг кода вправо;

– обнуление регистра (во все разряды регистра устанавливается нулевое значение).

К признакам, вырабатываемым счетчиком и посылаемым в управляющую часть, относятся:

– «ноль счетчика» («0» Сч) – характеризует состояние, при котором во всех разрядах регистра имеет место нулевое значение;

– «переполнение счетчика» – при поступлении очередного счетного сигнала счетчик переходит от максимального значения к значению «0».

Счетчик может выполнять следующие операции, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– установка нуля в счетчике;

– установка в счетчике некоторого начального значения;

– установка режима счета (обратный или прямой счет);

– изменение находящегося в счетчике текущего значения на единицу.

К признакам, вырабатываемым сумматором и посылаемым в управляющую часть, относятся:

– признак нулевого результата;

– признак единичных значений во всех разрядах результата;

– признак единицы в первом знаковом разряде результата;

– признак единицы во втором знаковом разряде результата;

– признак переноса из старшего разряда сумматора;

– признак наличия в тетраде значения, большего 9;

– признак межтетрадного переноса.

Каждому из перечисленных состояний может соответствовать отдельный разряд (флажок) в регистре флажков.

Сумматор может выполнять следующие микрооперации, инициируемые по управляющим сигналам, поступающим из управляющего блока:

– прием кода двух операндов на свои входы;

– формирование поразрядной суммы операндов, поступающих на его входы;

– генерирование поразрядного переноса;

– распространение переносов через разряды поразрядной суммы, пропускающие перенос;

– прибавление единицы в младший разряд;

– прибавление корректирующих кодов в тетрады при сложении двоично-десятичных кодов.

Выполнение любой арифметической операции в АЛУ реализуется за счет выполнения определенной последовательности микроопераций в узлах операционной части АЛУ. Такие последовательности образуют алгоритм выполнения операций на уровне микроопераций. Удобной формой представления алгоритма выполнения операций является граф-схема алгоритма (ГСА).

Построение блока управления.

Устройство управления вообще и блок управления АЛУ в частности могут строиться по принципу микропрограммирования (программируемая логика) или по принципу с жесткой логикой (аппаратный принцип). В любом случае удобной формой задания поведения управляемого объекта является кодированная ГСА.

При использовании микропрограммного принципа выработка необходимой последовательности сигналов управления имеющимся объектом выполняется за счет реализации микропрограммы, разработанной в соответствии с заданной ГСА. При использовании аппаратного принципа блок управления строится в виде цифрового автомата, который вырабатывает последовательность выходных сигналов, используемых для управления объектом.

Аппаратный принцип построения блока управления.

Блок управления аппаратного принципа может строиться на базе цифрового автомата. Наличие большого количества входных сигналов и состояний цифрового автомата, а также значительного числа пар «входной сигнал – состояние» делает затруднительным использование классического подхода к синтезу цифрового автомата. Поэтому в данном случае синтез цифрового автомата, реализующего данный блок управления, осуществляется по несколько отличному принципу и включает следующие этапы:

– построение графа выбранного типа цифрового автомата на основе ГСА функционирования имеющегося объекта;

– составление объединенной кодированной таблицы переходов и выходов цифрового автомата;

– составления логических выражений для сигналов управления памятью и выходных сигналов цифрового автомата;

– синтез логических схем на основании полученных логических выражений в заданном логическом базисе.

Микропрограммный принцип построения блока управления.

При микропрограммном принципе построения блока управления алгоритм выполнения операции осуществляется за счет особой программы, состоящей из отдельных команд, реализующих требуемую последовательность микроопераций. Такие команды получили название «микрокоманда», а совокупность микрокоманд – микропрограмма. Микропрограмма хранится в памяти ЭВМ.

При представлении алгоритма операции в виде ГСА выполняемая микрокоманда должна обеспечить выработку сигналов соответствующих микроопераций и переход к следующей микрокоманде, в том числе и при ветвлении вычислительного процесса. Таким образом, в формате микрокоманды, в принципе, необходимо иметь несколько полей:

– поле микроопераций (Y), используемое для задания одной или нескольких микроопераций;

– поле условий (X), в котором задаются проверяемые условия, влияющие на ветвление вычислительного процесса;

– поле адреса (A), в котором необходимо задавать информацию, определяющую следующую микрокоманду при возможном ветвлении (по крайней

мере по двум направлениям) для продолжения вычислительного процесса.

Задание всей перечисленной информации в едином формате микрокоманды затруднительно. Как правило, используют два вида, а следовательно, и два формата микрокоманд: операционные и перехода.

Операционная микрокоманда используется для реализации операторных вершин ГСА. Ее основная задача – задание выполняемой микрооперации. Формат операционной микрокоманды приведен на рисунке 4.19а.

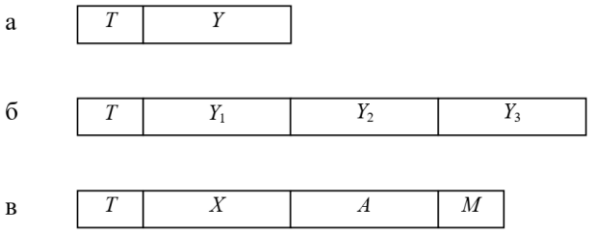


Рисунок 4.19 – Формат операционной микрокоманды

Микрокоманда включает два поля: поле типа микрокоманды (*Т*) и поле микрооперации (*У*).

Поле типа микрокоманды должно идентифицировать один из возможных типов микрокоманд. Так как используется всего два типа микрокоманды, то длина этого поля составляет один разряд.

Поле микрооперации задает, как правило, в кодированной форме подлежащую выполнению микрооперацию; его разрядность определяется множеством всех микроопераций, которые могут выполняться в управляемом устройстве. Если допустимая длина микрокоманды достаточно велика, то в одной операционной микрокоманде может задаваться более одной микрооперации (рисунок 4.19б).

После выполнения операционной микрокоманды выбирается микрокоманда, расположенная в следующем адресе ЗУ (после адреса расположения текущей микрокоманды).

Микрокоманда перехода в основном используется для организации ветвления на основании результата проверки некоторого условия (признака). Поэтому в ней необходимо задавать код проверяемого условия и информацию об адресах двух возможных ветвей продолжения выполнения алгоритма. Возможный формат микрокоманды перехода приведен на рисунке 4.19в. Приведенный формат включает следующие поля:

- поле типа микрокоманды (*Т*);
- поле условия (*Х*);
- поле адреса (*А*);
- поле модификатора дисциплины перехода (*М*).

Поле типа микрокоманды аналогично одноименному полю операционной микрокоманды.

Поле условия используется для задания кода условия, которое необходимо проверить при реализации данной микрокоманды. Его длина определяется общим количеством условий.

Поле адреса используется для задания местоположения в памяти адресов первых микрокоманд двух возможных ветвей продолжения процесса. При этом в качестве начальной микрокоманды одной из возможных ветвей продолжения используется микрокоманда, расположенная по адресу, следующему в памяти за адресом текущей выполняемой микрокоманды (*AT*), а адрес начальной микрокоманды другой ветви задается в самой микрокоманде. Возможны две дисциплины перехода к следующей (*Ac*) микрокоманде по результату проверки заданного условия:

1) $A_c = \begin{cases} A_T + 1, \ \textit{если} \ x_i = 1, \\ A, \ \textit{если} \ x_i = 0; \end{cases}$

2) $A_c = \begin{cases} A_T + 1, \ \textit{если} \ x_i = 0, \\ A, \ \textit{если} \ x_i = 1; \end{cases}$

При реализации алгоритма в одних случаях удобнее использовать первую, в других вторую дисциплину перехода. Наличие модификатора дисциплины перехода (поле *M*) позволяет в текущей команде перехода использовать дисциплину, наиболее удобную для организации данного ветвления. Однако это поле необязательно.

6 Процессор

Процессор представляет собой важнейшее устройство ЭВМ, которое используется для непосредственной реализации программы и управления взаимодействием всех других компонентов ЭВМ. Обобщенная структурная схема процессора приведена на рисунке 4.20.

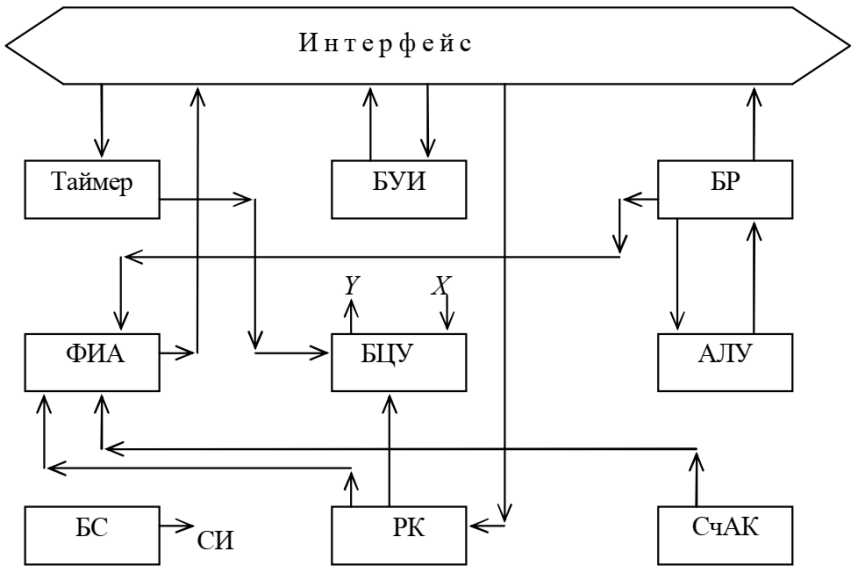


Рисунок 4.20 – Обобщенная структурная схема процессора

Схема включает:

- интерфейс – средства связи с другими компонентами ЭВМ;
- таймер – средство для измерения временных интервалов;
- БУИ – блок управления интерфейсом, обеспечивающий управление обменом информации через интерфейс;
- БР – блок регистров;
- ФИА – формирователь исполнительного адреса;
- БЦУ – блок центрального управления;
- АЛУ – арифметико-логическое устройство;
- БС – блок синхронизации;
- РК – регистр текущей команды выполняемой программы;
- СчАК – счетчик адреса команды.

Блок синхронизации формирует серии синхросигналов, различающихся по частоте и фазе.

СС – синхросигналы основной частоты. Частота этих сигналов может измеряться десятками, сотнями мегагерц. СС1 – синхросигналы той же основной частоты, но сдвинутые по фазе на полпериода. СС2, СС3 – синхросигналы с частотой в два раза меньшей, чем частота сигналов СС; они различаются длительностью.

СС4, СС5, СС6, СС7 – синхросигналы с частотой в четыре раза меньшей основной; между собой они различаются сдвигам по фазе на одну четвертую своего периода.

Синхросигналы различной частоты, как правило, формируются за счет деления основной частоты с помощью счетчиков, при этом сдвиг по фазе можно обеспечить установкой различных начальных значений в этих счетчиках.

7 Запоминающие устройства ЭВМ

Запоминающие устройства служат для хранения программ, данных и результатов обработки информации.

По месту в структуре ЭВМ запоминающие устройства бывают оперативными и внешними.

Оперативные запоминающие устройства, или оперативная память (ОП), представляют собой устройства, к которым обращается процессор за командами реализуемой программы, данными, подлежащими обработке; в эту же память помещаются промежуточные и конечные результаты непосредственно по завершении реализации программ. ОП характеризуется высоким быстродействием и сравнительно небольшим объемом памяти, измеряемым сотнями мегабайт.

Внешние запоминающие устройства (ВЗУ) представляют собой устройства, предназначенные для долговременного хранения программ, данных и конечных результатов выполненных программ. По мере необходимости информация, как правило, большими блоками передается в оперативную память и наоборот. ВЗУ характеризуются большой информационной емкостью, измеряемой десятками и сотнями гигабайт, и малой стоимостью хранения единицы информации. Однако по

сравнению с ОП внешние запоминающие устройства имеют меньшее быстродействие. Кроме того, время обращения к информации в ВЗУ зависит от местоположения информации в запоминающей среде поэтому на ее поиск может затрачиваться много времени, которое измеряется десятками миллисекунд и более. В отличие от ВЗУ оперативная память – это память с равновероятным доступом, т. е. время обращения к информации у этой памяти не зависит от местоположения этой информации в запоминающей среде.

Кроме этого, по месту в структуре ЭВМ выделяют сверхбыстродействующую память, или кэш-память). Данная память имеет меньшую информационную емкость (на порядок и более), чем оперативная память, но обладает быстродействием, в несколько раз превышающим быстродействие оперативной памяти. В структуре ЭВМ эта память располагается между процессором и оперативной памятью. Существующие виды памяти можно подразделить на энергозависимые и энергонезависимые. К энергозависимым относятся ЗУ, которые теряют информацию при отключении питания. В энергонезависимых запоминающих устройствах при каждом включении в памяти сохраняется информация, которая была до отключения питания.

По выполняемым функциям различают:

- память для чтения и записи;
- память только для записи, которая часто называется ПЗУ (постоянное запоминающее устройство).

Несмотря на ограниченность функций, ПЗУ широко используется наряду с памятью для чтения и записи. Это объясняется тем, что ПЗУ, как правило, обладает таким же быстродействием и не большей стоимостью, чем память для записи и чтения. Кроме того, ПЗУ – это энергонезависимая память, в то время как память для чтения и записи, как правило, является энергозависимой.

Все типы внешней памяти являются энергонезависимыми.

Оперативная память.

Различают два вида оперативной памяти:

- статическая память;
- динамическая память.

Статическая память характеризуется тем, что надежность считывания находящейся в ней информации не зависит от времени хранения. Строится на триггерах.

Динамическая память отличается тем, что спустя некоторый период времени после записи информации в память надежное считывание этой информации не гарантировано, т. к. информация в процессе хранения в динамической памяти как бы «затухает». Поэтому при использовании такой памяти через интервалы времени, не бóльшие, чем время гарантированного хранения, предусматривается процедура регенерации, при реализации которой происходит чтение информации по всему объему памяти и ее запись по старому адресу. Необходимость периодической регенерации снижает эффективность работы памяти, однако динамическая память широко используется благодаря тому, что при прочих равных условиях она обладает существенно меньшей стоимостью, чем память статическая.

Список использованных источников

1 Таненбаум Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – Санкт-Петербург : Питер, 2017. – 816 с.

2 Хамахер, К. Организация ЭВМ / К. Хамахер, З. Вранешич, С. Заки. – 5-е изд. – СПб. : Питер, 2003. – 848 с.

3 Пешков А.Т. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 1: Арифметические основы ЭВМ / А.Т. Пешков. – Мн.: БГУИР, 2004. – 61 с.

4 Пешков А.Т. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 2: Логические основы ЭВМ / А.Т. Пешков. – Мн.: БГУИР, 2005. – 36 с.

5 Пешков А.Т., Кобайло А.С. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 3: Схемотехнические основы ЭВМ / А.Т. Пешков, А. С. Кобайло. – Мн.: БГУИР, 2009. – 70 с.

6 Юров, В. И. Assembler : [учебник] / В. И. Юров. – 2-е изд. – Санкт-Петербург: Питер, 2011. – 637 с.

7 Афанасьев, В. А. Прикладная теория цифровых автоматов / В. А. Афанасьев. – М. : ВТ НГТУ, 2002. – 235 с.

8 Барилов, И. В. Арифметические и логические основы ЭВМ : пособие / И. В. Барилов, И. В. Москаленко, В. И. Науменко. – Шахты : ЮРГУЭС, 2005. – 32с.

9 Головчинер, М. Н. Введение в архитектуру ЭВМ : курс лекций / М. Н. Головчинер. – Томск : ТГУ, 2013. – 108 с.

2 ПРАКТИЧЕСКИЙ РАЗДЕЛ

2.1 Учебно-методический материал для проведения практических занятий

ОГЛАВЛЕНИЕ

1 Практическая работа №1. Системы счисления. Перевод чисел из одной системы счисления в другую

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Системы счисления. Перевод чисел из одной системы счисления в другую».
2. Теоретические сведения лекционного курса:

2.1 раздела II Арифметические основы ЭВМ, а именно:

2.2 подразделов:

- 1 Системы счисления**;**
- 2 Перевод чисел из одной системы счисления в другую;
- 2.1 Метод преобразования с использованием весов разрядов;
- 2.2 Метод деления (умножения) на новое основание;
- 2.3 Метод с использованием особого соотношения оснований заданной и искомой систем счисления.

3. Числа A, B, C, N, M, K, D, E для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Преобразования с использованием весов разрядов.

Представить правильную двоичную дробь A в десятичной системе счисления.

Найти двоичный эквивалент десятичного числа: B_{10} =____?____ 2.

Найти двоичный эквивалент числа:

C_{10} =_____?_____ 2 .

2.2 Метод деления (умножения) на новое основание.

Найти запись в двоичной форме десятичного числа N_{10}.

Найти запись в двоичной форме десятичного числа M_{10}.

2.3 Метод с использованием особого соотношения оснований систем счисления.

Найти двоичный эквивалент восьмеричного числа D_8.

Найти шестнадцатеричный эквивалент двоичного числа K_2.

Найти шестнадцатеричный эквивалент числа D_8, представленного в восьмеричной системе счисления.

Найти двоичный эквивалент числа E_{10}.

3. Подготовиться к защите.

2 Практическая работа №2. Арифметические операции в двоичной системе счисления

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Операции над числами в дополнительном и обратном кодах».
2. Теоретические сведения лекционного курса:

2.1 раздела II Арифметические основы ЭВМ, а именно:

2.2 подразделов:

- 3 Арифметические операции над положительными числами;
- 3.1 Операции сложения в двоичной системе счисления;
- 3.2 Операция вычитания;
- 3.3 Операция умножения.

3. Числа D, C, B, E, F, G для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Найти сумму S двоичных чисел D и C: S=D+C. Преобразовав число S в десятичную форму, проверить корректность выполненной операции.

2.2 Найти сумму R дробных двоичных чисел B и E: R=B+E. Преобразовав числа R, B, E в десятичную форму, проверить корректность выполненной операции.

2.3 Найти произведение P двоичных чисел F и G, начиная формирование частичных произведений со старшего разряда множителя: P=F*G. Преобразовав числа P, F, G в десятичную форму, проверить корректность выполненной операции.

2.4 Найти произведение P двоичных чисел B и E, начиная формирование частичных произведений с младшего разряда множителя: P=B*E. Преобразовав числа P, F, G в десятичную форму, проверить корректность выполненной операции.

3. Подготовиться к защите.

3 Практическая работа №3. Операции над числами в дополнительном и обратном кодах

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Операции над числами в дополнительном и обратном кодах».
2. Теоретические сведения лекционного курса:

2.1 раздела II Арифметические основы ЭВМ, а именно:

2.2 подразделов:

- 4 Арифметика с алгебраическими числами;
- 4.1 Кодирование алгебраических чисел;
- 4.2 Дополнительный и обратный коды двоичных чисел;
- 4.3 Операции с двоичными числами в дополнительном коде;
- 4.4 Операции с двоичными числами в обратном коде.

3. Числа A, B, C, D для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Найти значения для C1, C2, C3, C4, определяемых выражениями:

C1= A+B,

C2=A-B,

C3= B-A,

C4 = -A-B.

При выполнении операций использовать двоичный дополнительный код. Результат представить в прямом коде.

Проверить корректность операции, преобразовав результат в десятичный код.

2.2 Найти значения для C1, C2, C3, C4, определяемых выражениями

C1= C+D,

C2=C-D,

C3= D-C,

C4=-C-D.

При выполнении операций использовать двоичный обратный код. Результат представить в прямом коде.

Проверить корректность операции, преобразовав результат в десятичный код.

3. Подготовиться к защите.

4 Практическая работа №4. Операции деления с фиксированной точкой

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Операции деления с фиксированной точкой».
2. Теоретические сведения лекционного курса:

2.1 раздела II Арифметические основы ЭВМ, а именно:

2.2 подразделов:

- 6 Представление чисел с фиксированной точкой;
- 6.1 Арифметические операции над числами, представленными с фиксированной точкой;
- 6.2 Деление с фиксированной точкой.

3. Числа A, B, C, D для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Деление с восстановлением остатка.

Определить частное C1 от деления числа A на B.

При выполнении операций использовать дополнительный модифицированный код.

Проверить корректность операции, преобразовав результат в десятичный код.

2.2 *Деление без восстановления остатка*.

Определить частное C1 от деления числа C на D.

При выполнении операций использовать дополнительный модифицированный код.

Проверить корректность операции, преобразовав результат в десятичный код.

3. Подготовиться к защите.

5 Практическая работа №5. Представление чисел с плавающей точкой. Стандарт IEEE 754

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Представление чисел с плавающей точкой. Стандарт IEEE 754».
2. Теоретические сведения лекционного курса:

2.1 раздела II Арифметические основы ЭВМ, а именно:

2.2 подразделов:

- 7 Представление чисел с плавающей точкой;
- 7.1 IEEE 754 – стандарт двоичной арифметики с плавающей точкой;
- 7.2 Основные понятия в представлении чисел с плавающей точкой;
- 7.3 Преобразование десятичного числа в двоичное число с плавающей точкой;
- 7.4 Описание преобразования двоичного нормализованного числа в 32-битный формат IEEE 754.

3. Число D для выполнения индивидуального задания, приведенного ниже (выдается преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Представить десятичное число D в денормализованном экспоненциальном виде (в десятичной системе).

2.2 Представить десятичное число D в нормализованном экспоненциальном виде (*в десятичной системе*).

2.3 Представить число D в нормализованном экспоненциальном виде *в двоичной системе.*

2.4 Преобразовать двоичное нормализованное число D в 32-битный формат IEEE 754 и представить в шестнадцатеричной системе.

3. Подготовиться к защите.

6 Практическая работа №6. Формы представления логических функций. Синтез логических схем по логическим выражениям

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Формы представления логических функций. Синтез логических схем по логическим выражениям».
2. Теоретические сведения лекционного курса:

2.1 раздела III Логические основы ЭВМ, а именно:

2.2 подразделов:

- 1 Основные понятия алгебры логики;
- 2 Элементы алгебры Буля;
- 3 Формы представления логических функций;
- 4 Синтез логических схем по логическим выражениям.

3. Логическая функция y_i, заданная таблицей истинности; логическая функция Y, заданная в форме логического выражения – для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.

2.1 Представить логическую функцию y_i в совершенной конъюнктивной нормальной форме (СКНФ).

2) Представить логическую функцию y_i в совершенной дизъюнктивной нормальной форме (СДНФ).

3) Синтезировать логическую схему в базисе И-ИЛИ-НЕ, соответствующую логическому выражению Y.

3. Подготовиться к защите.

7 Практическая работа №7. Минимизация логических выражений

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Минимизация логических выражений».
2. Теоретические сведения лекционного курса:
2.1 раздела III Логические основы ЭВМ, а именно:
2.2 подразделов:
- 5 Минимизация логических выражений;
- 5.1 Минимизация методом Квайна;
- 5.2 Минимизация методом Карт Карно, или диаграмм Вейча.
3. Логическая функция Y, заданная в форме логического выражения – для выполнения индивидуального задания, приведенного ниже (выдается преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.
2.1 Найти минимальное выражение для функции Y с использованием правил и законов булевой алгебры.
2.2 Найти минимальное выражение для функции Y с использованием метода Квайна.
3. Подготовиться к защите.

8 Практическая работа №8. Логические базисы И-НЕ, ИЛИ-НЕ. Синтез логических схем

ИСХОДНЫЕ ДАННЫЕ

1. Объяснения преподавателя по теме «Логические базисы И-НЕ, ИЛИ-НЕ. Синтез логических схем».
2. Теоретические сведения лекционного курса:
2.1 раздела III Логические основы ЭВМ, а именно:
2.2 подразделов:
- 6 Логические базисы И-НЕ, ИЛИ-НЕ.
3. Логические функции Y1, Y2, заданные в форме логического выражения – для выполнения индивидуального задания, приведенного ниже (выдаются преподавателем).

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1. Изучить теоретические сведения.
2. Решить приведенные ниже задачи с оформлением письменного отчета.
2.1 Синтезировать логическую схему в базисе И-НЕ, соответствующую логическому выражению Y1.
2.2 Синтезировать логическую схему в базисе ИЛИ-НЕ, соответствующую логическому выражению Y2.
3. Подготовиться к защите.

3 РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

3.1 Экзаменационный материал

Экзамен.

При составлении экзаменационных билетов рекомендуется сочетать теоретические вопросы и практические задания таким образом, чтобы обеспечить их одинаковое соотношение в билете и исключить дублирование тематики. К экзамену допускаются студенты, успешно выполнившие курс практических работ.

Тематика теоретического экзаменационного материала

I Тематический раздел «Общие вопросы организации ЭВМ»

Понятие архитектуры ЭВМ. Организация ЭВМ с магистральной архитектурой

Общие сведения о системном программном обеспечении ЭВМ. Основные компоненты системного ПО. Базовая система ввода-вывода (BIOS/UEFI)

II Тематический раздел «Арифметические основы ЭВМ»

Системы счисления. Перевод чисел из одной системы счисления в другую. Метод преобразования с использованием весов разрядов. Метод деления (умножения) на новое основание. Метод с использованием особого соотношения оснований заданной и искомой систем счисления .

Арифметические операции над положительными числами. Операции сложения в двоичной системе счисления. Операция вычитания. Операция умножения. Деление двоичных чисел. Арифметика с положительными двоично-десятичными числами.

Арифметика с алгебраическими числами. Кодирование алгебраических чисел. Дополнительный и обратный коды двоичных чисел. Операции с двоичными числами в дополнительном коде. Операции с двоичными числами в обратном коде. Модифицированные коды.

Логические операции с двоичными кодами.

Представление чисел с фиксированной точкой. Арифметические операции над числами, представленными с фиксированной точкой. Деление с фиксированной точкой.

Представление чисел с плавающей точкой. IEEE 754 – стандарт двоичной арифметики с плавающей точкой. Основные понятия в представлении чисел с плавающей точкой. Преобразование десятичного числа в двоичное число с плавающей точкой. Арифметика с плавающей точкой.

III Тематический раздел «Логические основы ЭВМ».

Основные понятия алгебры логики. Элементы алгебры Буля. Формы представления логических функций. Синтез логических схем по логическим выражениям.

Минимизация логических выражений. Минимизация методом Квайна. Минимизация методом Карт Карно, или диаграмм Вейча.

Логические базисы И-НЕ, ИЛИ-НЕ.

IV Тематический раздел «Схемотехнические основы ЭВМ»

Цифровые устройства комбинационного типа. Двоичные сумматоры. Кодирующие и декодирующие устройства. Коммутаторы цифровых сигналов. Дешифраторы-демультиплексоры.

Триггеры. RS-триггер. Синхронный RS-триггер. Двухтактный RS-триггер. Т-триггер. JK-триггер D-триггер.

Регистры. Параллельные регистры (регистры памяти). Регистры сдвига.

Счётчики импульсов. Требования, предъявляемые к счётчикам. Суммирующие счётчики. Вычитающие и реверсивные счётчики. Счётчики с произвольным коэффициентом счёта.

Арифметико-логическое устройство ЭВМ.

Процессор.

Запоминающие устройства ЭВМ.

Тематика практического экзаменационного материала

Практический экзаменационный материал базируется на изученном студентами курсе практических занятий и может включать следующие направления практических заданий:

- по тематическому разделу «Арифметические основы ЭВМ» – системы счисления; перевод чисел из одной системы счисления в другую; арифметические операции в двоичной системе счисления; операции над числами в дополнительном и обратном кодах; операции деления с фиксированной точкой; представление чисел с плавающей точкой, стандарт IEEE 754;

- по тематическому разделу «Логические и схемотехнические основы ЭВМ» – формы представления логических функций; синтез логических схем по логическим выражениям; минимизация логических выражений; логические базисы И-НЕ, ИЛИ-НЕ и синтез логических схем в данных базисах.

4 ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

4.1 Учебная программа

Учреждение образования
«Брестский государственный технический университет»

УТВЕРЖДАЮ
Первый проректор
_____ М.В. Нерода
28.06. 2021 г.

Регистрационный № УД-21-1-233/уч.

Архитектура ЭВМ

Учебная программа учреждения высшего образования по учебной дисциплине для специальности:
1-53 01 02 Автоматизированные системы обработки информации

2021

Учебная программа составлена на основе образовательного стандарта ОСВО 1–53 01 02–2013, учебной программы по учебной дисциплине «Архитектура ЭВМ» № УД-2-1535/уч. (разработчик – учреждение образования «Белорусский государственный университет информатики и радиоэлектроники», кафедра информационных технологий автоматизированных систем) и учебного плана специальности 1-53 01 02 «Автоматизированные системы обработки информации».

СОСТАВИТЕЛЬ:
Савицкий Ю.В., доцент кафедры интеллектуальных информационных технологий, кандидат технических наук, доцент

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:
Кафедрой интеллектуальных информационных технологий
Заведующий кафедрой В.А. Головко
(протокол № 7 от 15 июня 2021);

Методической комиссией факультета электронно-информационных систем
Председатель методической комиссии С.С. Дереченник
(протокол № 9 от 21.06.2021);

Научно-методическим советом БрГТУ (протокол № 5 от 28.06. 2021)
Методист УМО В.И. Сергей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Место учебной дисциплины.

Дисциплина «Архитектура ЭВМ» является одной из базовых в процессе подготовки студентов по специальности «Автоматизированные системы обработки информации»; относится к государственному компоненту; в значительной степени носит системный характер, формируя основу для освоения ряда последующих курсов; обеспечивает выработку систематизированных знаний по основам структурной, функциональной и арифметико-логической организации современных средств компьютерной техники; имеет выраженную практическую направленность.

Цель преподавания учебной дисциплины:

изучение основ организации ЭВМ. Формирование у студентов систематизированного представления об особенностях архитектуры современных ЭВМ и их базовых элементов. Получение практических навыков по организации вычислительного процесса в компьютерных системах.

Задачи учебной дисциплины:

- приобретение знаний по организации современных ЭВМ и принципам их функционирования;
- изучение принципов построения типовых цифровых устройств ЭВМ;
- овладение методами организации вычислений в ЭВМ.

Изучение учебной дисциплины должно обеспечить формирование у студента следующих компетенций:

академических:

- умение применять базовые научно-теоретические знания для решения теоретических и практических задач;
- владение основными методами, способами и средствами получения, хранения, переработки информации с использованием компьютерной техники;
- владение междисциплинарным подходом при решении проблем;

социально-личностных:

- способность к социальному взаимодействию;
- обладание способностью к межличностным коммуникациям;

профессиональных:

- владение компьютерными методами сбора, хранения и обработки информации в сфере своей профессиональной деятельности;
- владение методами эффективной эксплуатации программных средств;
- приобретение новых знаний, используя современные информационные технологии.

В результате изучения учебной дисциплины студент должен:

знать:

- организацию архитектур современных ЭВМ;
- принципы организации вычислительного процесса в ЭВМ;
- формы преставления информации в ЭВМ;
- основы машинной арифметики;

- типовые устройства ЭВМ и принципы их построения;

уметь:

- работать с числами различных систем счисления;
- выполнять логическую и арифметическую обработку двоичных чисел;
- строить и использовать типовые компоненты ЭВМ с заданными параметрами;

владеть:

- основными приемами анализа архитектур ЭВМ различного назначения;
- навыками реализации арифметических и логических операций в двоичной системе счисления;
- основами построения логических схем.

Связи с другими учебными дисциплинами. Для успешного освоения дисциплины студент должен иметь базовую подготовку в рамках учебной программы средней школы по общематематическим дисциплинам, физике информатике и др.

План учебной дисциплины для дневной формы получения высшего образования

Код специальности	Наименование специальности	Курс	Семестр	Всего учебных часов	Количество зачетных единиц	Аудиторных часов (в соответствии с учебным планом УВО)					Академических часов на курсовой проект (работу)	Форма текущей аттестации
						Всего	Лекции	Лабораторные занятия	Практические занятия	Семинары		
1-53 01 02	«Автоматизированные системы обработки информации»	1	1	108	3	48	32	-	16	-	-	письмен-ный экза-мен

План учебной дисциплины для заочной формы получения высшего образования

Код специальности	Наименование специальности	Курс	Семестр	Всего учебных часов	Количество зачетных единиц	Аудиторных часов (в соответствии с учебным планом УВО)					Академических часов на курсовой проект (работу)	Форма текущей аттестации
						Всего	Лекции	Лабораторные занятия	Практические занятия	Семинары		
1-53 01 02	«Автоматизированные системы обработки информации»	1	1	108	3	12	8	-	4	-	-	письменный экзамен

План учебной дисциплины для заочной формы получения высшего образования, интегрированного со средним специальным образованием

Код специальности	Наименование специальности	Курс	Семестр	Всего учебных часов	Количество зачетных единиц	Аудиторных часов (в соответствии с учебным планом УВО)					Академических часов на курсовой проект (работу)	Форма текущей аттестации
						Всего	Лекции	Лабораторные занятия	Практические занятия	Семинары		
1-53 01 02	«Автоматизированные системы обработки информации»	1	1	108	3	12	8	-	4	-	-	письменный экзамен

1. СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

1.1. ЛЕКЦИОННЫЕ ЗАНЯТИЯ, ИХ СОДЕРЖАНИЕ

Раздел 1. ОРГАНИЗАЦИЯ ЭВМ

Тема 1. Введение. Основные понятия

Эволюция современных средств компьютерной техники. Понятие архитектуры ЭВМ. Фон-Неймановский принцип организации ЭВМ, **с**войства и особенности работы машины фон Неймана. Классификация ЭВМ.

Тема 2. Базовые компоненты ЭВМ

Процессор. Оперативная и внешняя память. Системный интерфейс. Подсистема ввода-вывода. Каналы передачи информации. Взаимодействие устройств и организация вычислительного процесса

Тема 3. Общие сведения о системном программном обеспечении ЭВМ

Базовая система ввода/вывода. Понятие, задачи, решаемые с помощью базовой системы ввода/вывода. Общие сведения об операционных системах. Принцип многозадачности

Раздел 2. АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭВМ

Тема 1. Системы счисления

Особенности систем счисления с равномерно распределенными весами. Методы перехода из одной системы записи чисел в другую. Особенности преобразования дробных чисел. Преобразования с использованием особого соотношения старого и нового оснований

Тема 2. Двоичная арифметика с положительными числами

Операции сложения и вычитания положительных чисел. Методы выполнения умножения и деления. Особенности операций с двоично-десятичными числами.

Тема 3. Арифметика с алгебраическими числами

Прямой, обратный и дополнительный коды. Модифицированные коды. Методы выполнения операций умножения алгебраических чисел.

Методы выполнения операций деления (с восстановлением и без восстановления остатка) алгебраических чисел. Представление чисел с фиксированной и плавающей точками. Стандарт IEEE 754-1985. Особенности арифметики с плавающей точкой.

РАЗДЕЛ 3. ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ. ОРГАНИЗАЦИЯ ТИПОВЫХ ЭЛЕМЕНТОВ ЭВМ

Тема 1. Логические функции и логические схемы устройств ЭВМ

Представление логических выражений в базовых логических функциях. Канонические нормальные формы представления логических выражений. Постановка задачи минимизации логических выражений. Методика построения логических схем устройств ЭВМ.

Тема 2. Типовые элементы ЭВМ

Элементы памяти на триггерах. Назначение, функции и построение регистров, кодеров, декодеров, мультиплексоров, демультиплексоров и счетчиков различного типа. Синхронизация цифровых схем.

Тема 3. Устройства ЭВМ

Принцип построения арифметико-логического устройства. Запоминающие устройства. Принципы построения оперативной памяти ЭВМ. Принципы построения блоков управления.

1.2. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ, ИХ СОДЕРЖАНИЕ

1. Структурная и функциональная организация ЭВМ
2. Системы счисления
3. Арифметические операции в двоичной системе счисления
4. Операции над числами в дополнительном и обратном кодах
5. Операции деления с фиксированной точкой
6. Построение логических схем устройств ЭВМ
7. Построение логических схем в заданных логических базисах
8. Задача минимизация логических выражений

2. УЧЕБНО-МЕТОДИЧЕСКАЯ ЧАСТЬ

2.1. УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ
для дневной формы получения высшего образования

Номер раздела, темы, лекции	Название раздела, темы	Количество аудиторных часов				Количество часов самост. работы	Форма контроля знаний
		Лекции	Лабораторные занятия	Практические занятия	Семинарские занятия		
1	2	3	4	5	6	7	8
1	Раздел 1. Организация ЭВМ						
1.1.1	Тема 1. Введение. Основные понятия	1				3	контрольные опросы
1.2.2	Тема 2. Базовые компоненты ЭВМ	2				2	контрольные опросы
	Практ. занятие № 1. Структурная и функциональная организация ЭВМ			2		2	письменные отчеты
1.3.3	Тема 3. Общие сведения о системном программном обеспечении ЭВМ	1				3	контрольные опросы
2	Раздел 2. Арифметические основы ЭВМ						
2.1.4	Тема 1. Системы счисления	2				2	контрольные опросы
	Практ. занятие № 2. Системы счисления			2		2	письменные отчеты
2.2	Тема 2. Двоичная арифметика с положительными числами						
2.2.5	Операции сложения и вычитания, умножения и деления чисел	2				3	контрольные опросы
2.2.6	Особенности сложения и вычитания двоично-десятичных чисел	2				2	контрольные опросы
1	2	3	4	5	6	7	8
	Практ. занятие № 3. Арифметические операции в двоичной системе счисления			2		2	письменные отчеты
2.3	Тема 3. Арифметика с алгебраическими числами						
2.3.7	Прямой, обратный и дополнительный коды	2				3	контрольные опросы

1	2	3	4	5	6	7	8
2.3.8	Методы выполнения операций деления	2				3	контрольные опросы
	Практ. занятие № 5. Операции деления с фиксированной точкой			2		2	письменные отчеты
3	Раздел 3. Логические основы ЭВМ. Организация типовых элементов ЭВМ						
3.1	Тема 1. Логические функции и логические схемы устройств ЭВМ						
3.1.9	Представление логических выражений в базовых логических функциях	2				2	контрольные опросы
3.1.10	Канонические нормальные формы представления логических выражений	2				2	контрольные опросы
3.1.11	Постановка задачи минимизации логических выражений	2				3	контрольные опросы
3.1.12	Методика построения логических схем устройств ЭВМ	2				2	контрольные опросы
	Практ. занятие № 6. Построение логических схем устройств ЭВМ			2		2	письменные отчеты
	Тема 2. Типовые элементы ЭВМ						
3.2.13	Элементы памяти на триггерах	2				3	контрольные опросы
	Практ. занятие № 7. Построение логических схем в заданных логических базисах			2		2	письменные отчеты
3.2.14	Назначение, функции и построение регистров, кодеров, декодеров	2				3	контрольные опросы
3.2.15	Назначение, функции и построение мультиплексоров, демультиплексоров, счетчиков	2				2	контрольные опросы
3.3	Тема 3. Устройства ЭВМ						
3.3.16	Принцип построения арифметико-логического устройства. Запоминающие устройства. Принципы построения оперативной памяти ЭВМ	2				3	контрольные опросы
	Практ. занятие № 8. Задача минимизация логических выражений			2		2	письменные отчеты
3.3.17	Принципы построения блоков управления	2				3	контрольные опросы

2.2. УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ для заочной формы получения высшего образования

Номер раздела, темы, лекции	Название раздела, темы	Количество аудиторных часов				Количество часов самост. работы	Форма контроля знаний
		Лекции	Лабораторные занятия	Практические занятия	Семинарские занятия		
1	Раздел 1. Организация ЭВМ						
1.1.1	Тема 1. Введение. Основные понятия	0,5				8	контрольные опросы
1.2.2	Тема 2. Базовые компоненты ЭВМ	0,5				8	контрольные опросы
	Практ. занятие № 1. Структурная и функциональная организация ЭВМ			1		8	письменные отчеты
1.3.3	Тема 3. Общие сведения о системном программном обеспечении ЭВМ	1				8	контрольные опросы
2	Раздел 2. Арифметические основы ЭВМ						
2.1.4	Тема 1. Системы счисления	1				8	контрольные опросы
	Практ. занятие № 2. Системы счисления			2		8	письменные отчеты
2.2.5	Тема 2. Двоичная арифметика с положительными числами	1				8	контрольные опросы
	Практ. занятие № 3. Арифметические операции в двоичной системе счисления			1		8	письменные отчеты
2.3.6	Тема 3. Арифметика с алгебраическими числами	1				8	контрольные опросы
3	Раздел 3. Логические основы ЭВМ. Организация типовых элементов ЭВМ						
3.1.7	Тема 1. Логические функции и логические схемы устройств ЭВМ	1				8	контрольные опросы
3.2.8	Тема 2. Типовые элементы ЭВМ	1				8	контрольные опросы
3.3.9	Тема 3. Устройства ЭВМ	1				8	контрольные опросы

2.3. УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ для заочной формы получения высшего образования, интегрированного со средним специальным образованием

Номер раздела, темы, лекции	Название раздела, темы	Количество аудиторных часов				Количество часов самост. работы	Форма контроля знаний
		Лекции	Лабораторные занятия	Практические занятия	Семинарские занятия		
1	2	3	4	5	6	7	8
1	Раздел 1. Организация ЭВМ						
1.1.1	Тема 1. Введение. Основные понятия	0,5				8	контрольные опросы
1.2.2	Тема 2. Базовые компоненты ЭВМ	0,5				8	контрольные опросы

1	2	3	4	5	6	7	8
	Практ. занятие № 1. Структурная и функциональная организация ЭВМ			1		8	письменные отчеты
1.3.3	Тема 3. Общие сведения о системном программном обеспечении ЭВМ	1				8	контрольные опросы
2	Раздел 2. Арифметические основы ЭВМ						
2.1.4	Тема 1. Системы счисления	1				8	контрольные опросы
	Практ. занятие № 2. Системы счисления			2		8	письменные отчеты
2.2.5	Тема 2. Двоичная арифметика с положительными числами	1				8	контрольные опросы
	Практ. занятие № 3. Арифметические операции в двоичной системе счисления			1		8	письменные отчеты
2.3.6	Тема 3. Арифметика с алгебраическими числами	1				8	контрольные опросы
3	Раздел 3. Логические основы ЭВМ. Организация типовых элементов ЭВМ						
3.1.7	Тема 1. Логические функции и логические схемы устройств ЭВМ	1				8	контрольные опросы
3.2.8	Тема 2. Типовые элементы ЭВМ	1				8	контрольные опросы
3.3.9	Тема 3. Устройства ЭВМ	1				8	контрольные опросы

3. ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

3.1. Перечень литературы

Основная

1. Таненбаум Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – Санкт-Петербург : Питер, 2017. – 816 с.

2. Хамахер, К. Организация ЭВМ / К. Хамахер, З. Вранешич, С. Заки. – 5-е изд. – СПб. : Питер, 2003. – 848 с.

3. Пешков А.Т. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 1: Арифметические основы ЭВМ / А.Т. Пешков. – Мн.: БГУИР, 2004. – 61 с.

4. Пешков А.Т. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 2: Логические основы ЭВМ / А.Т. Пешков. – Мн.: БГУИР, 2005. – 36 с.

5. Пешков А.Т., Кобайло А.С. Организация и функционирование ЭВМ: Методическое пособие: В 3 ч. Ч. 3: Схемотехнические основы ЭВМ / А.Т. Пешков, А. С. Кобайло. – Мн.: БГУИР, 2009. – 70 с.

Дополнительная

1. Юров, В. И. Assembler : [учебник] / В. И. Юров. – 2-е изд. – Санкт-Петербург : Питер, 2011. – 637 с.

2. Афанасьев, В. А. Прикладная теория цифровых автоматов / В. А. Афанасьев. – М. : ВТ НГТУ, 2002. – 235 с.

3. Барилов, И. В. Арифметические и логические основы ЭВМ : пособие / И. В. Барилов, И. В. Москаленко, В. И. Науменко. – Шахты : ЮРГУЭС, 2005. – 32с.

4. Головчинер, М. Н. Введение в архитектуру ЭВМ : курс лекций / М. Н. Головчинер. – Томск : ТГУ, 2013. – 108 с.

5. Асмаков, С. В. Железо 2011. КомпьютерПресс рекомендует / С. В. Асмаков, С. О. Пахомов. – Санкт-Петербург : Питер, 2011. – 416 с.

3.2. Перечень средств диагностики результатов учебной деятельности

- контрольные опросы;
- письменные отчеты по практическим занятиям;
- письменный экзамен.

3.4. Методические рекомендации по организации и выполнению самостоятельной работы

Самостоятельная работа предполагает:

- ведение и систематическую проработку конспекта лекций и учебной литературы;
- подготовку к практическим занятиям – включает проработку соответствующих разделов рекомендованной литературы и конспекта лекций по тематике практических занятий;
- подготовку письменных отчетов по результатам выполнения индивидуальных заданий практических занятий;
- подготовку к письменному экзамену по дисциплине.

Ссылки на рекомендуемые источники, учебную литературу (в разрезе тем изучаемой дисциплины) представлены в таблице ниже.

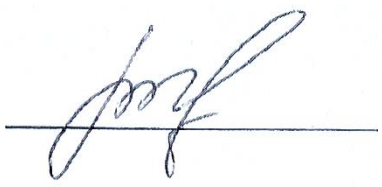
Тема учебной дисциплины	Литература
Раздел 1. Организация ЭВМ	
Тема 1. Введение. Основные понятия	[1, Введение, §§1.1,1.2], [2,§1]
Тема 2. Базовые компоненты ЭВМ	[1, §2]
Тема 3. Общие сведения о системном программном обеспечении ЭВМ	[1, §6]
Раздел 2. Арифметические основы ЭВМ	
Тема 1. Системы счисления	[3, §§1,2]
Тема 2. Двоичная арифметика с положительными числами	[3, §3]
Тема 3. Арифметика с алгебраическими числами	[3, §§4,5]
Раздел 3. Логические основы ЭВМ. Организация типовых элементов ЭВМ	
Тема 1. Логические функции и логические схемы устройств ЭВМ	[4, §§1.1, 1.2.1-1.2.5]
Тема 2. Типовые элементы ЭВМ	[5, §§1.1, 1.2]
Тема 3. Устройства ЭВМ	[5, §2]

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ К УЧЕБНОЙ ПРОГРАММЕ
по дисциплине
Архитектура ЭВМ
Регистрационный NoУД-21-1-233/уч. (утверждена 28.06.2021) на 2023/2024 учебный год

№ п/п	Дополнения и изменения	Основание
1	В учебную программу «Архитектура ЭВМ», утвержденную 28.06.2021 (регистрационный № УД- 21-1-233/уч. для специальности 1-53 01 02 АСОИ), внести изменение: заменить код и наименование специальности 6-05-0612-03 Системы управления информацией	Учебный план специальности на 6-05-0612-03 Системы управления | информацией (Осуществление набора на родственную специальность в рамках факультета ЭИС при совпадении объема и содержания курса)

Изменение к учебной программе «Архитектура ЭВМ», утвержденную 28.06.2021 (регистрационный № УД-21-1-233/уч. для специальности 1-53 01 02 АСОИ) рассмотрено и одобрено на заседании кафедры интеллектуальных информационных технологий (протокол № 1 от 31.08.2023)

Заведующий кафедрой
интеллектуальных информационных
технологий  В.А. Головко

УТВЕРЖДАЮ
Декан факультета электронно-
информационных систем  С.Ф. Лебедь

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ К УЧЕБНОЙ ПРОГРАММЕ
по дисциплине «Архитектура ЭВМ»
специальности 1-53 01 02 «Автоматизированные системы обработки информации»
(28.06.2021 УД-21-1-253/уч.)

№ п/п	Дополнения и изменения	Основание
	Текущая аттестация студентов по дисциплине проводится в форме следующих видов отчётности: 1) контрольного опроса в письменной форме по результатам практических занятий; 2) контрольного опроса по теоретической части курса. При расчете итоговой отметки текущей аттестации учитываются обе формы отчётности. Весовой коэффициент текущей аттестации $K_{тек}$ рассчитывается следующим образом: $K_{тек} = (1 — K_{промеж})/M$, где $K_{промеж} = 0,4$ — весовой коэффициент промежуточной аттестации; M — количество предусмотренных текущих аттестаций в семестре. Результаты текущей аттестации учитываются при проведении промежуточной аттестации по дисциплине.	Положение о текущей аттестации студентов БрГТУ от 08.02.2024 №5/1

Учебная программа пересмотрено и одобрена на заседании кафедры ИИТ (протокол № 5 от 15.05 2024г.)

Заведующий кафедрой _____ В. А. Головко

УТВЕРЖДАЮ
Декан ФЭИС _____ А. Н. Парфиевич