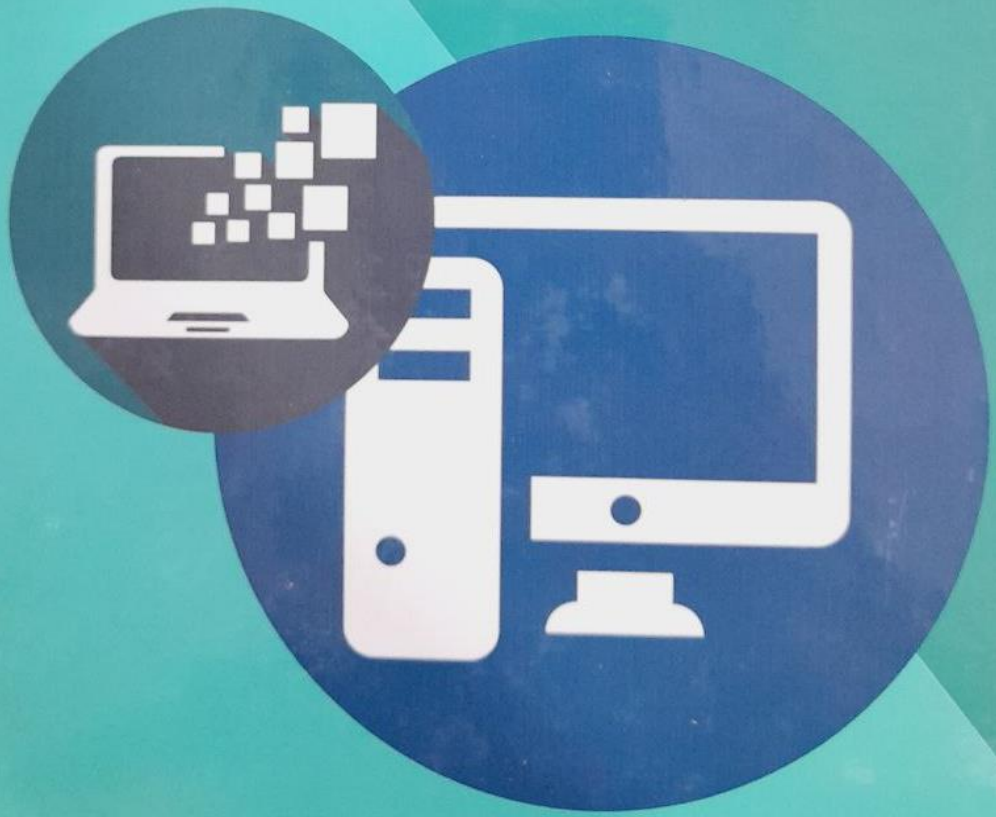
БОТИРОВ С.Р.

ОРГАНИЗАЦИЯ КОМПЬЮТЕРА



МИНИСТЕРСТВО ЦИФРОВЫХ ТЕХНОЛОГИЙ
РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ–ХОРАЗМИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕРНОГО ИНЖИНИРИНГА

БОТИРОВ С.Р.

ОРГАНИЗАЦИЯ КОМПЬЮТЕРА

Рекомендовано в качестве учебного пособия Ташкентским университетом информационных технологий имени Мухаммада Аль-Хорезми

Для студентов обучающихся по направлению:
60610500 – Компьютер инжиниринг (Компьютер инжиниринг)

Ташкент
"METODIST NASHRIYOTI"
2024

УДК: 004.3(075.8)
ББК: 32.973я7
Б 860

Ботиров С.Р.
Организация компьютера. Учебное пособие. - Ташкент: "METODIST NASHRIYOTI", 2024. – 206 стр.

В учебном пособии представлена последовательность выполнения практических работ по предмету «Организация компьютера», направленных на закрепление теоретических знаний полученных во время лекционных занятий и развитие практических навыков связанных с: архитектурной организацией компьютерных систем, которые определяют основные характеристики компьютеров, такие как производительность и функциональные возможности, изучением работы различных типов памяти, обмена информации с внешними и внутренними устройствами. Практические работы выполняются с использованием моделирования логических элементов компьютерной системы, а так же элементов программирования на языка ассемблер с использованием эмулятора КР580. Каждая тема включает в себя необходимый теоретический материал, дополняющий лекционные занятия, практическую часть, а так же вопросы для закрепления пройденного материала

Рецензенты:
Ш.С. Каримов
Нурафшанский филиал ТУИТ имени Мухаммада ал–Хоразмий
и.о. Заведующий кафедрой «Информационные технологии», PhD, доцент
Д.Т. Мухаммедиева
Кафедра «Программного обеспечения информационных технологий»
ТУИТ имени Мухаммада ал–Хоразмий д.т.н. (DsC), профессор

Публикация разрешена на основании приказа Ташкентского университета информационных технологий имени Мухаммада Аль-Хорезми № 439 от 25 апреля 2023 года.

ISBN 978-9910-63-230-1

ВВЕДЕНИЕ

Современные компьютеры представляют собой очень сложные системы. Еще более сложную организацию имеют компьютерные системы и сети.

Курс «Организация компьютера» включает широкий спектр вопросов, касающихся различных аспектов построения компьютерных систем. Для освоения предлагаемого материала необходимо знание основ схемотехники, программирования и операционных систем.

Термины «компьютер» и «компьютерная система», используемые в учебном пособии, эквивалентны терминам «электронно–вычислительная машина» (ЭВМ) и «вычислительная система» (ВС). Эти термины в настоящее время все чаще встречаются как в научно–технической литературе, так и в информационных источниках.

Материал излагается на примере развития процессоров фирм Intel и AMD, так как они служат основой для большинства (до 90%) настольных компьютеров и ноутбуков, продаваемых в Узбекистане. С другой стороны, в процессорах этих фирм применены многие наиболее эффективные архитектурные решения, которые изучаются на лекционных занятиях.

Настоящее учебное пособие дополняет лекционный материал и направлено на более глубокое изучение процессов ввода–вывода и обмена информации на самом низком уровне. В следствии этого освещенные в данном учебном пособии тематики описаны с элементами программирования на языке ассемблер, с использованием эмулятора КР580ВМ.

Учебное пособие включает в себя 5 глав состоящих из 10 тем.

Первая глава включает в себя две темы, в первой теме описана организация структуры компьютерной системы их основные показатели и характеристики. Во второй теме рассмотрены современные процессоры и принципы их работы.

Вторая глава включает в себя 2 темы, которые посвящены изучению арифметических и численно логических основ организации компьютера. Приводятся примеры выполнения арифметико–логических операций в разных системах счислений.

архитектуру системы команд процессора. В ней описаны операторы языка ассемблера. Тема посвящена выполнению сложных арифметических операций на языке ассемблер.

В четвертой главе рассмотрены виды памяти и их характеристики. Описываются программные способы маскирования данных и организации условных переходов на языке ассемблер.

В заключительной главе рассмотрены типы шин и портов, ввод и вывод данных на внешние устройства компьютера. Описаны процессы обмена данными.

ГЛАВА I
ОРГАНИЗАЦИЯ СТРУКТУРЫ КОМПЬЮТЕРНОЙ СИСТЕМЫ. ПРИНЦИПЫ РАБОТЫ СОВРЕМЕННЫХ ПРОЦЕССОРОВ

1.1 Организация структуры компьютерной системы, основные показатели и характеристики

Цель работы: Изучение внешних и внутренних устройств компьютера.

Теоретическая часть

Основные части компьютера

Для рядового пользователя персонального компьютера глубокие познания аппаратной части (так называемого "железа") совсем не обязательны. Иное дело — профессионалы, даже если пока они только студенты–технари. Будущим техническим специалистам следует знать и уметь опознавать компоненты типичной системы ПК.

По своей конструкции ПК является модульной системой. Системой он называется потому, что содержит все компоненты, требуемые для функционального компьютера.

Системный блок — основной компонент компьютера (который иногда называют корпусом), содержащий главные компоненты системы. Сюда входят основная плата логики (системная, или материнская, плата), процессор, память, один или несколько дисководов разных типов, импульсный источник питания и соединительные провода и кабели. Системный блок также содержит платы расширения для обеспечения аудио, видео, сетевой и других функциональности. Набор плат расширения обычно разный для различных систем.

Клавиатура. Наиболее знакомое компьютерное устройство ввода, клавиатура применяется для ввода символов и команд в систему.

Мышь — устройство ввода, применяемое с графическим пользовательским интерфейсом для указания, выбора или активизации изображений на видеомониторе. Передвигая

манипулятор типа "мышь" по гладкой поверхности, пользователь вызывает синхронное перемещение курсора на дисплее ПК.

Монитор — стандартное устройство отображения информации для ПК, служит для многоцветного отображения символов и графики.

Принтер — устройство вывода копии оцифрованного изображения, которое переносит данные на бумажный либо пленчатый материальный носитель. Обычно для вывода информации на бумагу применяются матричные, струйные и лазерные принтеры.

Звуковые колонки — устройства вывода звука для прослушивания речи, музыки и другой звуковой информации.

Внутренние части системного блока.

Системная плата. Основные и наиболее важные элементы персонального компьютера, а если быть точнее, то именно системного блока: видеокарта, центральный процессор, модули ОЗУ и большое количество микросхем располагаются именно на системной плате, а её более широко распространённое название – материнская плата (рис. 1.1.1).

Материнская плата – это основная системная плата компьютера, имеющая разъёмы для установки дополнительных плат расширения и служащая механической основой всей электронной схемы компьютера. Благодаря материнской плате обеспечивается полное взаимодействие компонентов компьютерной системы.

Основная задача материнской платы – объединение и обеспечение совместной работы всех комплектующих компьютера. Одной из важных характеристик «материнки» является её форм–фактор – стандарт, который определяет её размеры для компьютера, места крепления внутри системного блока, расположение на поверхности сокета CPU, портов ввода/вывода, слотов для оперативной памяти и др. При сборке нового компьютера (или выполняя ремонт компьютеров) обязательно нужно учитывать, что корпус должен поддерживать форм–фактор выбранной системной платы.

Элементы материнской платы

Чипсет (англ. chipset) – набор из нескольких микросхем, которые спроектированы для совместной работы и основная задача которых – выполнение набора определенных функций. В

компьютерах чипсет, находящийся на материнской плате, является связующим компонентом. Он обеспечивает совместную работу процессора, подсистем памяти, ввода–вывода и др.

От чипсета зависит, какой тип оперативной памяти и процессор поддерживаются «материнкой». Кроме этого от него зависит, с какой скоростью будут передаваться данные по шине ко всем устройствам компьютера.

Как мы уже определили – одна из основных функций материнской платы заключается объединении устройств между собой или, говоря образно, в "наведении мостов" между ними, поэтому главные составляющие чипсета называются "мостами".

Чипсет состоит из двух "мостов", каждый из которых является отдельной микросхемой и выполняет свою определенную задачу:

"северный" мост (англ. Northbridge) нужен для соединения между собой процессора и устройств, которые используют высокопроизводительные шины – оперативная память и видеокарта. От северного моста зависит частота системной шины, максимальный объем оперативной памяти и ее тип. Иногда северный мост содержит в себе интегрированный (встроенный) графический процессор.

"южный" мост (англ. Southbridge) необходим для подключения менее скоростных устройств, которые не требуют высокой пропускной способности – сетевые платы, жёсткий диск, шины USB, PCI и др., к которым подключаются дополнительные устройства.

В состав материнской платы входят следующие разъемы и слоты:

1. Разъемы для подключения питания.
2. Разъем (socket) для подключения процессора.
3. Слоты для установки оперативной памяти.
4. Разъемы для подключения жестких дисков и оптических приводов.
5. USB – разъемы.
6. Разъемы PCI (Взаимосвязь периферийных устройств).
7. Разъем для подключения видеокарты (PCI – EXPRESS).
8. Северный мост (Northbridge) – обеспечивает бесперебойную передачу данных в связке «ЦП – Оперативная память – Графический адаптер».

9. Южный мост обеспечивает передачу данных между портами USB, оптическими приводами и жесткими дисками, также отвечает за устройства ввода: клавиатуру, мышь.

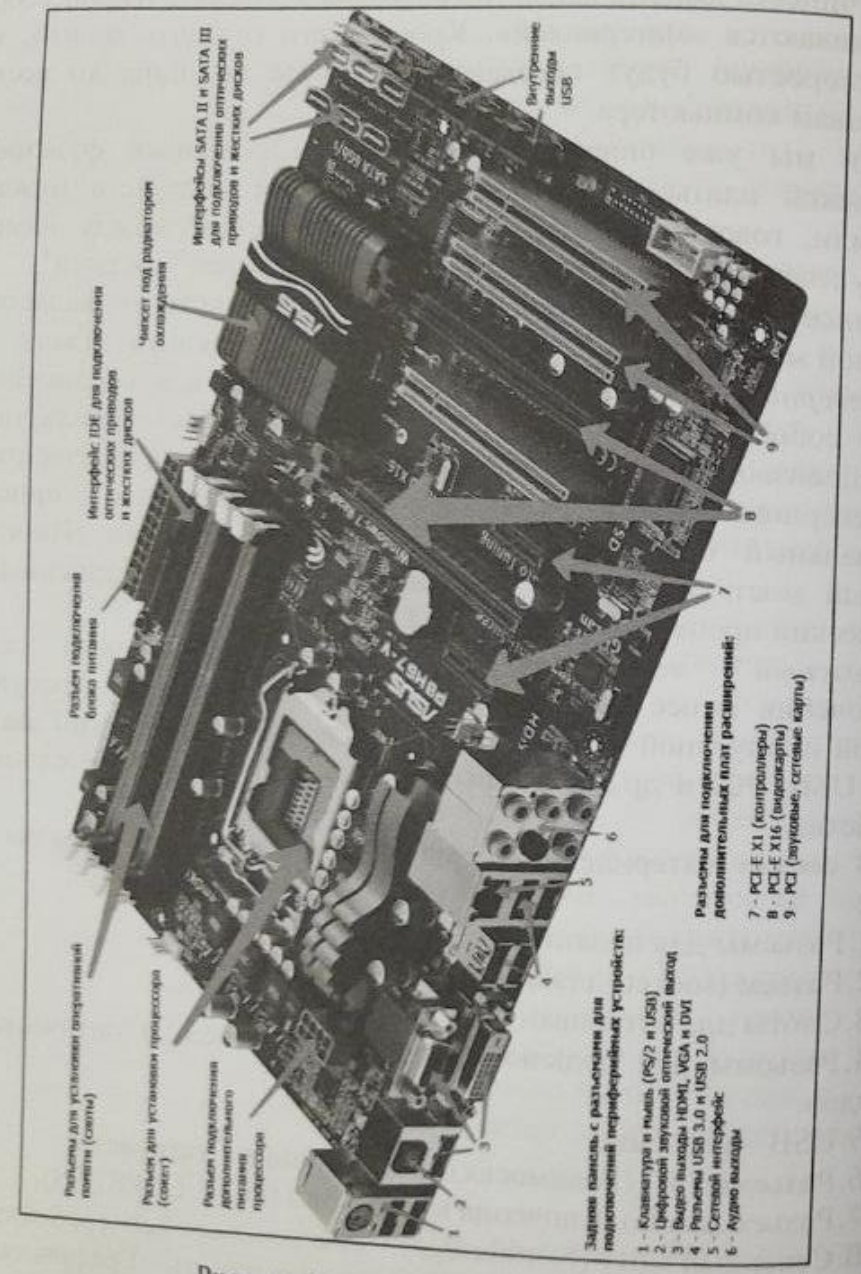


Рисунок 1.1.1 – Системная плата

Блок питания. Чтобы все компоненты могли выполнять свою задачу, их нужно запитать электрической энергией. Для снабжения этой энергией используется компьютерный блок питания (по–английски power supply unit или PSU), от которого тянутся провода по всему системному блоку (рис 1.1.2).

Большинство устройств имеют специальный разъем для подключения питания, но некоторые получают электрическую энергию через системную плату (которая в этом случае будет посредником между блоком питания и устройством).

Рисунок 1.1.2 – Компьютерные блоки питания

Центральный процессор – это мозг компьютера (рис 1.1.3). Его задача – выполнять программы, находящиеся в основной памяти. Он вызывает команды из памяти, определяет их тип, а затем выполняет одну за другой. Компоненты соединены шиной, представляющей собой набор параллельно связанных проводов, по которым передаются адреса, данные и сигналы управления. Шины могут быть внешними (связывающими процессор с памятью и устройствами ввода–вывода) и внутренними.

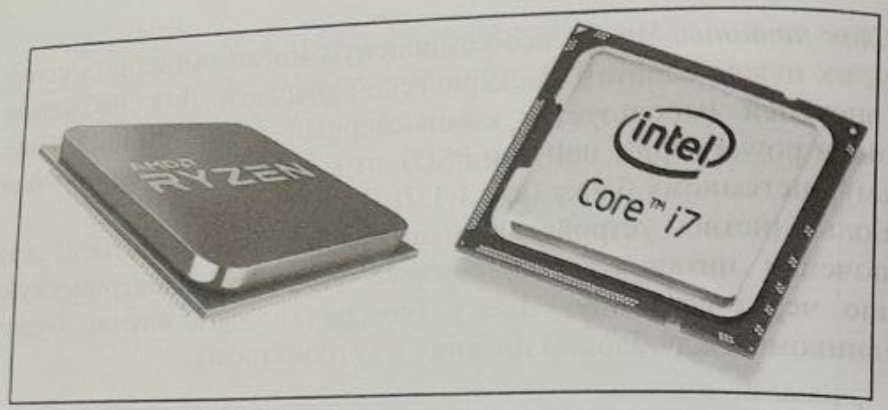


Рисунок 1.1.3 – Процессор

Оперативная память (ОЗУ, Random Access Memory, RAM), как и процессор, устанавливается в специальные разъемы на системной плате. Оперативная память выполнена в виде небольшой печатной платы с установленными на неё микросхемами памяти, всю эту конструкцию называют «модулем памяти». Из–за специфичной формы платы, её называют «планкой».

1. DDR SDRAM (Double Data Rate Synchronous Dynamic Random Access Memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, а также является самым старым видом оперативной памяти, которую можно еще сегодня купить (рис. 1.1.4).

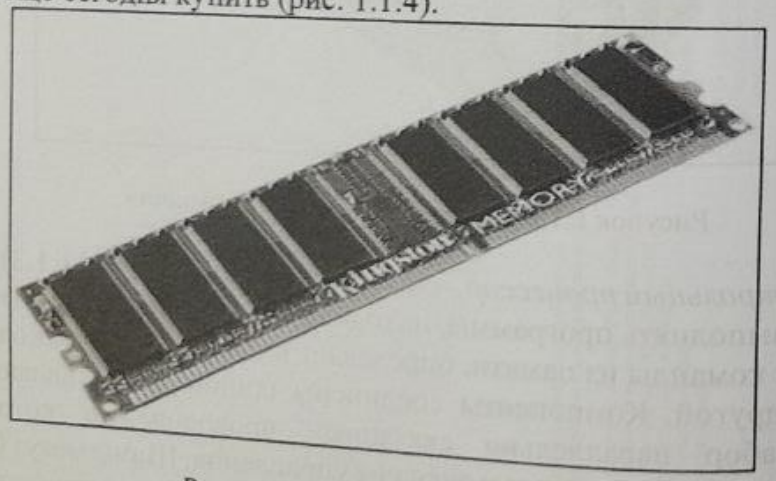


Рисунок 1.1.4 – DDR SDRAM

2. DDR2 SDRAM (double–data–rate two synchronous dynamic random access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, второе поколение (рис. 1.1.5). Тоже устаревший вид оперативной памяти.

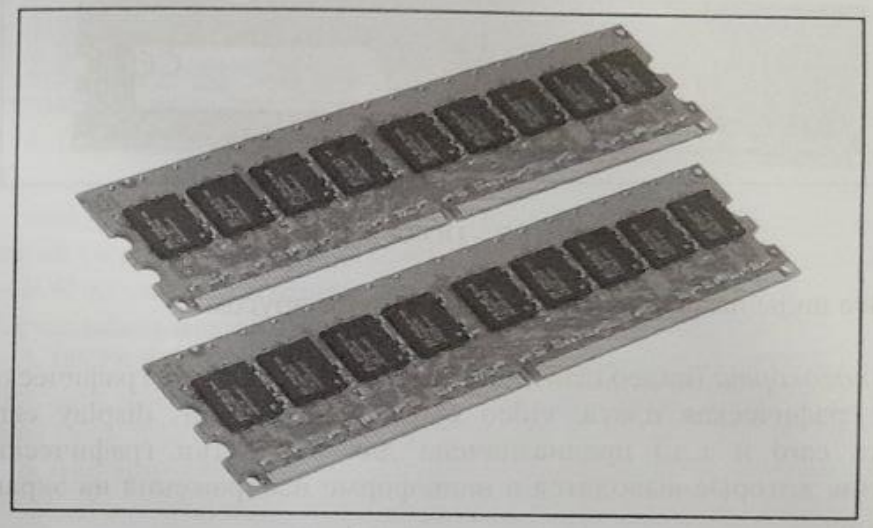


Рисунок 1.1.5 – DDR2 SDRAM

3. DDR3 SDRAM (double–data–rate three synchronous dynamic ran– dom access memory) – синхронная динамическая память с произвольным доступом и удвоенной скоростью передачи данных, третье поколение (рис. 1.1.6). Широко используется в современных компьютерах.

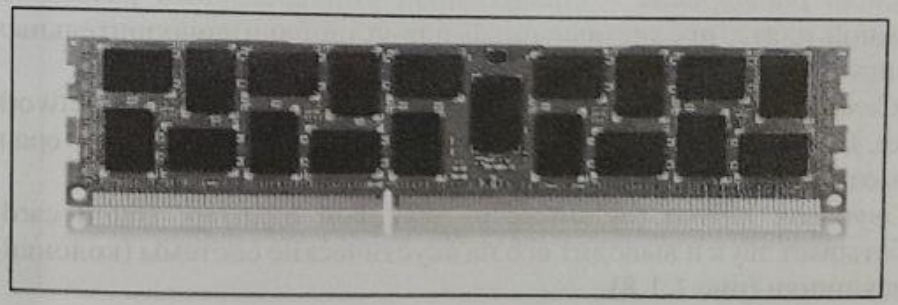


Рисунок 1.1.6– DDR3 SDRAM

4. DDR4 SDRAM (double–data–rate four synchronous dynamic ran– dom access memory) – новый тип оперативной памяти, являющийся эволюционным развитием предыдущих поколений DDR (рис. 1.1.7)

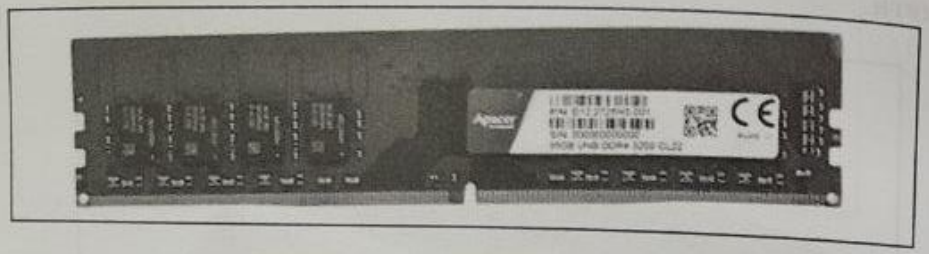


Рисунок 1.1.7 – DDR4 SDRAM

Все виды памяти не совместимы друг с другом.

Видеокарта (видеоадаптер, графический адаптер, графическая карта, графическая плата, video card, video adapter, display card, graphics card и т.д.) предназначена для обработки графических объектов, которые выводятся в виде/форме изображения на экране монитора.

Карта расширения – устройство в виде печатной платы с универсальным разъемом для установки на системную плату (например, видеокарта, сетевая карта, звуковая карта).

Карты расширения устанавливаются дополнительно к основным компонентам для того, чтобы расширить возможности компьютера, они могут иметь различное назначение (обработка графики, звука или соединение с компьютерной сетью и т.д.).

Слот расширения – специальный универсальный разъем на системной плате, предназначенный для установки дополнительных устройств компьютера выполненных виде карт расширения.

Сетевая карта (сетевой адаптер, Ethernet–адаптер, network adapter, LAN adapter) предназначена для подключения компьютера к компьютерной сети.

Звуковая карта (аудиокарта, звуковой адаптер, sound card) обрабатывает звук и выводит его на акустические системы (колонки) или наушники (рис 1.1.8).

Как и два предыдущих устройства, звуковая карта – это печатная плата, вставленная в разъем на системной плате. Правда, данный звуковой адаптер не обычный, он состоит из двух печатных плат, но это исключение из правил.

Можно выделить несколько основных типов данных устройств:

1. CD–ROM – данный привод способен читать только CD.
2. CD–RW – позволяет не только считывать информацию с обычных компакт–дисков, но и записывать её на CD–R и CD–RW.
3. DVD–ROM – устройство, способное читать компакт–диски DVD.
4. DVD–CD–RW Combo – так называемый Combo–драйв, который сочетает в себе функции таких устройств, как DVD–ROM и CD– RW и, соответственно, может записывать диски CD–R и CD–RW, считывать как обычные CD, так и DVD.
5. DVD–RW –позволяет не только читать диски CD/DVD, но и записывать как обычные CD–R/CD–RW–носители, так и куда более ёмкие DVD–R/DVD–RW/DVD+R/DVD+RW.
6. Blu–Ray, BD (blue ray — синий луч, и Disk – диск) — формат оптического носителя, используемый для записи и хранения цифровых данных, включая видео высокой чёткости с повышенной плотностью.

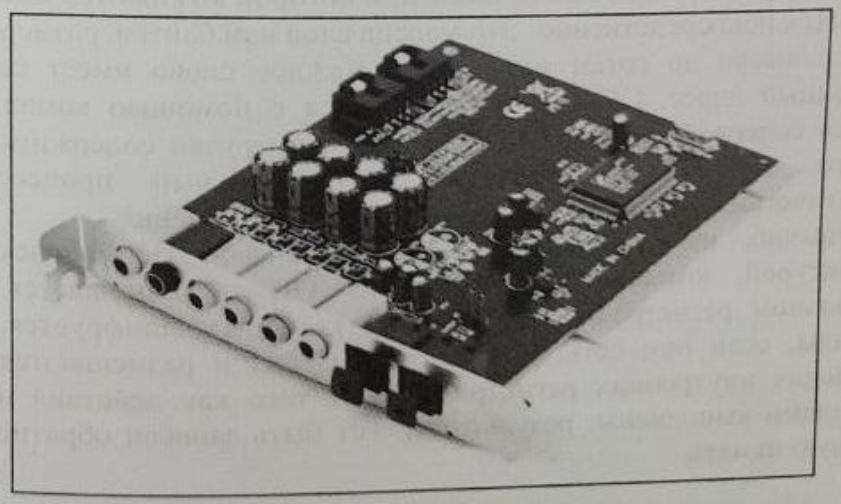


Рисунок 1.1.8 – Звуковая карта (плата расширение)

Жесткий диск. Жесткий диск в отличие от предыдущих компонентов, не устанавливается на системную плату, а крепится в специальном отсеке корпуса системного блока (посмотрите на фотографию).

Жесткий диск иногда называют аббревиатурой НМЖД (Накопитель на жёстких магнитных дисках), часто говорят «винчестер«, а на английском языке hard disk drive или HDD.

Оптический привод (DVD–привод, optical disc drive или ODD) нужен для чтения и записи DVD и CD дисков. Как и жесткий диск, оптический привод устанавливается в специальный отсек системного блока.

Самый важный регистр – счетчик команд, который указывает, какую команду нужно выполнять следующей. Название «счетчик команд» не соответствует действительности, поскольку он ничего не считает, но этот термин употребляется повсеместно. Еще есть регистр команд, в котором находится выполняемая в данный момент команда. У большинства компьютеров имеются и другие регистры, одни из них многофункциональны, другие выполняют лишь какие–либо специфические функции.

Структура памяти. Программа может выполняться компьютером только в том случае, если она размещена в оперативной (основной) памяти. Оперативная память – единственная большая область памяти, к которой компьютер может обращаться непосредственно. Это массив слов или байтов, размером от сотен тысяч до сотен миллионов. Каждое слово имеет свой собственный адрес. Обмен осуществляется с помощью команды загрузки содержимого памяти в регистр и выгрузки содержимого регистра в память. Кроме того, центральный процессор автоматически выбирает команды из ОП для выполнения.

Обычно, например, в компьютерах с фон Неймановской архитектурой, команда выбирается из ОП и размещается в специальном регистре команды. Затем команда дешифруется, а операнды, если они есть, выбираются из ОП и размещаются в нескольких внутренних регистрах. После того как действия над операндами выполнены, результат может быть записан обратно в основную память.

В идеале хотелось бы, чтобы программы и данные находились в основной памяти. Однако постоянно хранить их там невозможно по следующим двум причинам:

– основная память обычно слишком мала для размещения всех необходимых программ и данных;

– содержимое основной памяти теряется при отключении питания.

Поэтому в большинстве компьютеров имеется так называемая вторичная память как продолжение основной, то есть внешние запоминающие устройства. Главное назначение вторичной памяти – длительное хранение больших массивов данных.

В качестве накопителя вторичной памяти обычно используется магнитный диск, на котором размещаются как программы, так и данные.

Все разнообразие запоминающих устройств компьютера может быть организовано в иерархию по убыванию времени доступа, возрастанию цены и увеличению емкости, изображенную на рис. 1.1.9.

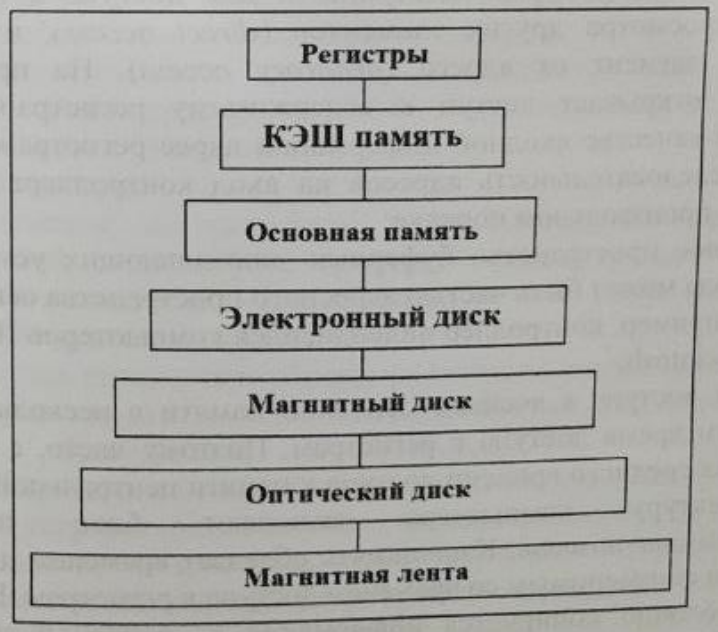


Рисунок 1.1.9 – Иерархия запоминающих устройств компьютера

Многоуровневую схему используют следующим образом. Информация, которая находится в памяти верхнего уровня, обычно хранится также на более низких уровнях. Если процессор не обнаруживает нужную информацию на некотором уровне, он начинает искать ее на нижележащих уровнях. Когда нужная информация найдена, она переносится в более быстрые уровни.

Обычно в центральной части сосредоточены регистры процессора (*CPU registers*), основная память(*main memory*) и кэш–память (*cashe memory*).

Система команд центрального процессора содержит команды, позволяющие работать с регистрами и основной памятью. Работа с кэш–памятью осуществляется на аппаратном уровне контроллером кэш–памяти. Для работы с вторичной (внешней) памятью нужна программа. Таким образом, программно–доступными являются основная и вторичная память, а также регистры центрального процессора. Кэш–память программно недоступна.

Метод доступа к адресуемым элементам памяти центральной части называется ***произвольным*** *(random access)*. Это означает, что все элементы доступа равнодоступны и для доступа к ним не требуется просмотра других элементов (*direct access*), и время доступа не зависит от адреса (*arbitrary access*). На практике контроллер открывает доступ к содержимому регистра/ячейки, используя в качестве входной информации адрес регистра/ячейки. Причем последовательность адресов на вход контроллера может следовать в произвольном порядке.

Адресное пространство буферных запоминающих устройств ввода–вывода может быть частью адресного пространства основной памяти. Например, контроллер видеодисплея компьютеров IBM PC и Apple Macintosh.

Время доступа к ячейкам основной памяти в несколько раз больше, чем время доступа к регистрам. Поэтому часто, с целью уменьшения среднего времени доступа к памяти центральной части в архитектуру компьютера включают блок памяти, называемый *кэш–память*. Кэш–память обладает временем доступа равным или соизмеримым со временем доступа к регистрам. В кэш–память поблочно копируется информация из основной памяти. Эффективность кэш–памяти с точки зрения увеличения производительности во многом зависит от стратегии копирования.

Вначале центральный процессор проверяет наличие требуемых данных в кэш–памяти, а затем в основной. Хорошая стратегия копирования обеспечивает нахождение информации в кэш–памяти в 80% – 99% случаев обращения процессора к памяти.

Необходимость вторичной памяти обусловлена двумя причинами:

– емкость основной памяти недостаточна для хранения всех программ и данных;

– основная память энергозависима и ее содержимое теряется после отключения напряжения питания.

В качестве накопителя вторичной памяти обычно используется магнитный диск. Накопитель на магнитном диске обычно состоит из нескольких дисков, покрытых слоем магнитного материала и расположенных на общей оси вращения. В процессе работы диски вращаются с постоянной скоростью. Информация наносится на поверхность магнитного материала побитно в виде концентрических окружностей, называемых ***дорожками*** *(tracks)*. Любая дорожка содержит одинаковое количество информации. Дорожки разделены на ***секторы*** *(sectors)*. Сектор является наименьшей адресуемой порцией информации для магнитного диска.

Запись/считывание данных осуществляется без остановки диска с помощью универсальных головок чтения/записи. Головки устанавливаются на соответствующую дорожку с помощью механизма доступа.

Накопители на магнитном диске отличаются количеством поверхностей, диаметром дисков, возможностью легкого извлечения накопителя из компьютера.

Структура ввода–вывода

В состав компьютера общего назначения входят центральный процессор и множество контроллеров, которые соединены общей шиной. Контроллеры периферийного оборудования специализированы по типу оборудования и обычно один контроллер управляет устройством одного типа. Однако иногда конструируются универсальные контроллеры, предназначенные для работы с несколькими разнотипными устройствами. Например, контроллер SCSI (Small Computer System Interface) позволяет подключать до семи различных устройств. Каждый контроллер снабжен памятью, включающей буферное запоминающее устройство, регистр

оманды, регистр состояния. Контроллер должен обладать пособностью перемещать данные между периферийным стройством, которым он управляет, и локальной буферной амятью.

Обычно в операционной системе имеется соответствующий ***драйвер*** (*driver*) для каждого контроллера. Драйвер умеет напрямую работать с устройством и предоставляет унифицированный интерфейс для остальной части операционной системы.

Данные, выводимые на носитель информации или вводимые с носителя информации, предварительно накапливаются в буфере. Размер буфера различен для разных устройств и определяется спецификой устройства. Контроллер выполняет приказы (команды) центрального процессора, поступающие в регистр команд. Например, получив команду прочитать данные с носителя, контроллер вырабатывает последовательность управляющих сигналов, в результате чего поверхность носителя перемещается, информация считывается, преобразуется и записывается в буфер.

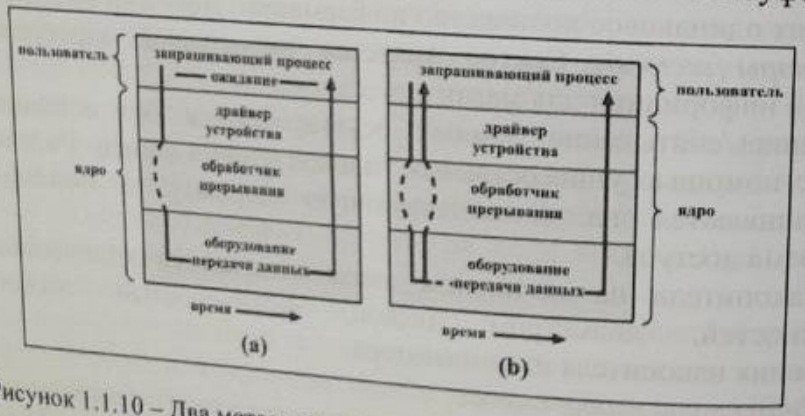


Рисунок 1.1.10 – Два метода взаимодействия центрального процессора и контроллера: (a) – синхронный, (b) –асинхронный

Существует два метода взаимодействия центрального процессора и контроллера: синхронный и асинхронный (рис. 1.1.10). При синхронном методе процесс, запрашивающий операцию ввода–вывода, после прерывания ожидает завершения операции контроллером. При асинхронном методе – после передачи в контроллер команды ввода–вывода, центральный процессор и контроллер работают раздельно.

При синхронном методе ожидание процессором завершения операции ввода–вывода означает, что он работает «вхолостую». Наиболее простой способ заставить процессор работать «вхолостую» заключается в организации «вечного» цикла. Этот цикл продолжается до тех пор пока не получено прерывание. Некоторые компьютеры включают в свою систему команд специальную команду wait, переводящую процессор в «холостой» режим работы. Главное преимущество синхронного метода состоит в легкости определения контроллера, требующего прерывания.

Асинхронный метод предполагает одновременную работу процессора и одного или нескольких контроллеров, что повышает эффективность использования оборудования, однако усложняет механизм прерываний. При асинхронном методе, например, возможна ситуация, когда несколько различных процессов требуют операции ввода–вывода с одним и тем же устройством.

Практическая часть

Для выполнения практической работы № 1 необходимо установить программу CPU-Z (рис. 1.1.11), с помощью которой можно узнать модель и характеристики вашего ПК или ноутбука. Все данные нужно записать в таблицу 1.1.1.

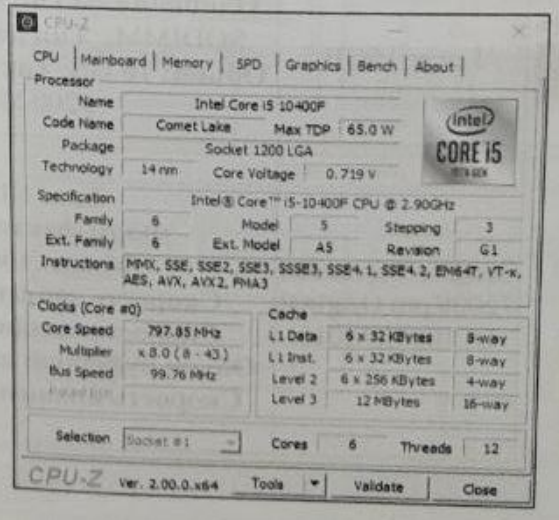
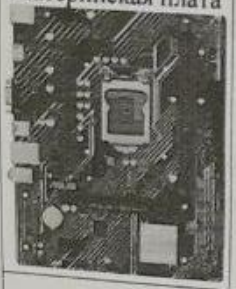


Рисунок 1.1.11 – Скриншот окна программы CPU–Z

Таблица 1.1.1

Характеристики элементов ПК(Ноутбука)

Название элемента	Фирма производитель и модель	Основные характеристики
Процессор 	Intel Core i5-10400F Comet Lake	**Количество ядер:** 6. **Количество потоков:** 12. Максимальная тактовая частотав режиме **Turbo:** 4,30 GHz. **Базовая** тактовая частота процессора: 2,90 GHz. **Максимальная потребляемая мощность:** 65 Вт
Материнская плата 	ASUS PRIME H510M-K	**Форм-фактор:** mATX; **Сокет:** LGA 1200; **Чипсет:** Intel H510; **Память:** DDR4 - 2слота, частотой до 2933 МГц; **Слоты:** PCI-E 4.0 x16 x 1, PCI-E x1 x 1; **Разъемы:** M.2 x 1, SATA3 x 4,HDMI x 1, VGA (D-Sub) x 1; **Сеть:** Gigabit Ethernet;
Оперативная память 	Lexar 16GB	**Одноканальный; Форм-фактор:** SODIMM; **Тип памяти:** DDR4; **Объем:** 16 GB; **Тактовая частота:** 3200 МГц
Жёсткий диск 	Seagate Barracuda 7200Rpm Original ST1000DM010	**Тип:** HDD; **Форм-фактор:** 3.5"; **Объем:** 1000 ГБ; **Скорость записи:** 156 МБ/с; **Скорость чтения:** 156 МБ/с; **Объем буферной памяти:** 64 МБ; **Скорость вращения:** 7200 rpm

Название элемента	Фирма производитель и модель	Основные характеристики
Твердый накопитель 	SAMSUNG SSD 980	**Объем:** 256Gb; **Форм фактор:** M2.2280; **Скорость записи:** 3000 МБ/с **Скорость чтения:** 3500 МБ/с **Интерфейс:** PCIe3.0
Видеокарта 	GIGABYTE GeForceGTX 1650	**Чип:** TU106, 12 нм, 896 CUDA- ядер, 1665 МГц; **Видеопамять:** 4 ГБ GDDR6, 12000 МГц, 128 бит; **Энергопотребление:** 90 Вт **Входы/Выходы:** DVI-D, HDMI2.0b, DisplayPort 1.4. **Интерфейс подключения:** PCIe3.0.
Блок питания 	Cooler Master Elite500 V3 230 500W	**Мощность:** 500 Вт; **Версия:** ATX12V2.3; **Форм-фактор:** ATX **Система охлаждения:** 1 вентилятор **Диаметр вентилятора:** 120 мм

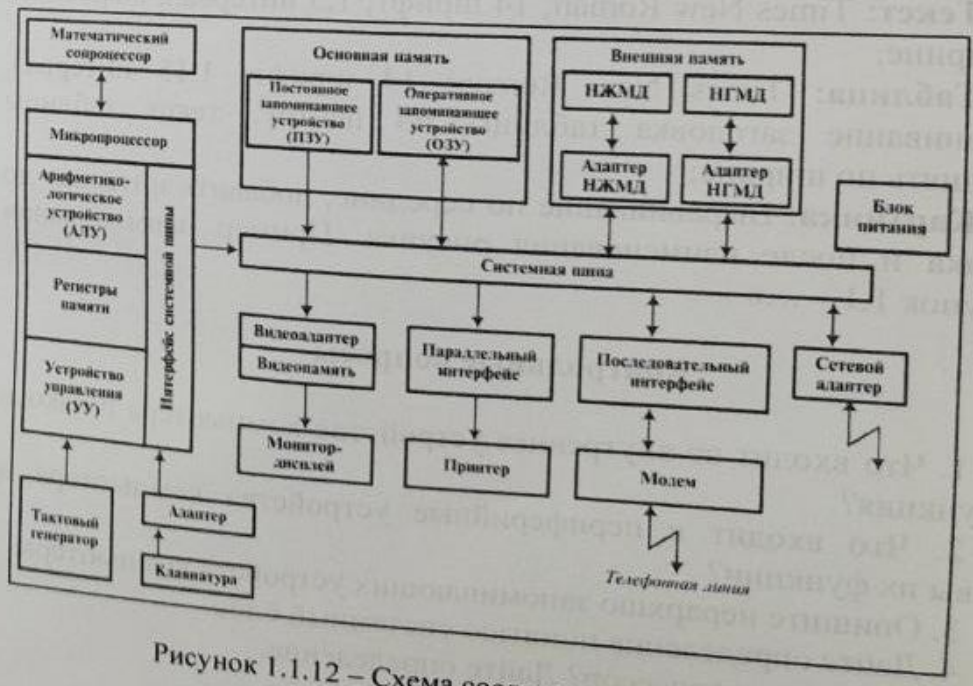


Рисунок 1.1.12 – Схема соединения элементов

Задания для выполнения практической работы № 1

1. Изучить каждый элемент ПК (ноутбука), записать в таблицу 1.1.2, название, фирму производителя и основные характеристики элементов.

2. Нарисовать схему соединение элементов ПК (ноутбука).

Таблица 1.1.2
Элементы ПК (Ноутбука)

Название элемента	Фирма производитель и модель	Основные характеристики

3. Установить программу CPU–Z, (установочный файл приложен к заданию, рис 1.1.11) и сделать скриншот окон приложения.

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 1.1 – …»

Контрольные вопросы

1. Что входит во внутреннее устройство компьютера и какова их функция?
2. Что входит в периферийные устройства компьютера и каковы их функции?
3. Опишите иерархию запоминающих устройств компьютера.
4. Дайте определение понятию системный блок.
5. Что такое процессор? Дайте определение.

6. Расскажите о функции центрального процессора и его видах.
7. Что входит в состав системной планы?
8. Что такое оперативная память?
10. Что такое оптический привод?

1.2. Современные процессоры и принципы их работы. Идентификация и установка процессора

Цель работы: Изучение принципов работы современных процессоров. Идентификация и установка процессора.

Теоретическая часть

Центральный процессор (ЦП или центральное процессорное устройство ЦПУ) – главная часть аппаратного обеспечения компьютера и его вычислительный центр. По сути, он является исполнителем машинных инструкций и предназначен для выполнения сложных компьютерных программ. У ЦПУ есть несколько главных характеристик, но для обычного обывателя важны лишь две – *тактовая частота и количество ядер*. Первые массовые многоядерные процессоры для настольных ПК были выпущены в начале 2006 года и на данный момент почти полностью вытеснили одноядерные.

Современный процессор – это сложное и высокотехнологическое устройство, включающее в себя все самые последние достижения в области вычислительной техники и сопутствующих областей науки.

Большинство современных процессоров состоит из:

– одного или нескольких ядер, осуществляющих выполнение всех инструкций;

– нескольких уровней КЭШ–памяти (обычно, 2 или три уровня), ускоряющих взаимодействие процессора с ОЗУ;

– контроллера ОЗУ;

– контроллера системной шины (DMI, QPI, HT и т.д.);

И характеризуется следующими параметрами:

– типом микроархитектуры;

– тактовой частотой;

– набором выполняемых команд;
– количеством уровней КЭШ–памяти и их объемом;
– типом и скоростью системной шины;
– размерами обрабатываемых слов;
– наличием или отсутствием встроенного контроллера памяти;
– типом поддерживаемой оперативной памяти;
– объемом адресуемой памяти;
– наличием или отсутствием встроенного графического ядра;
– энергопотреблением.

Упрощенная структурная схема современного многоядерного процессора представлена на рис. 1.2.1.

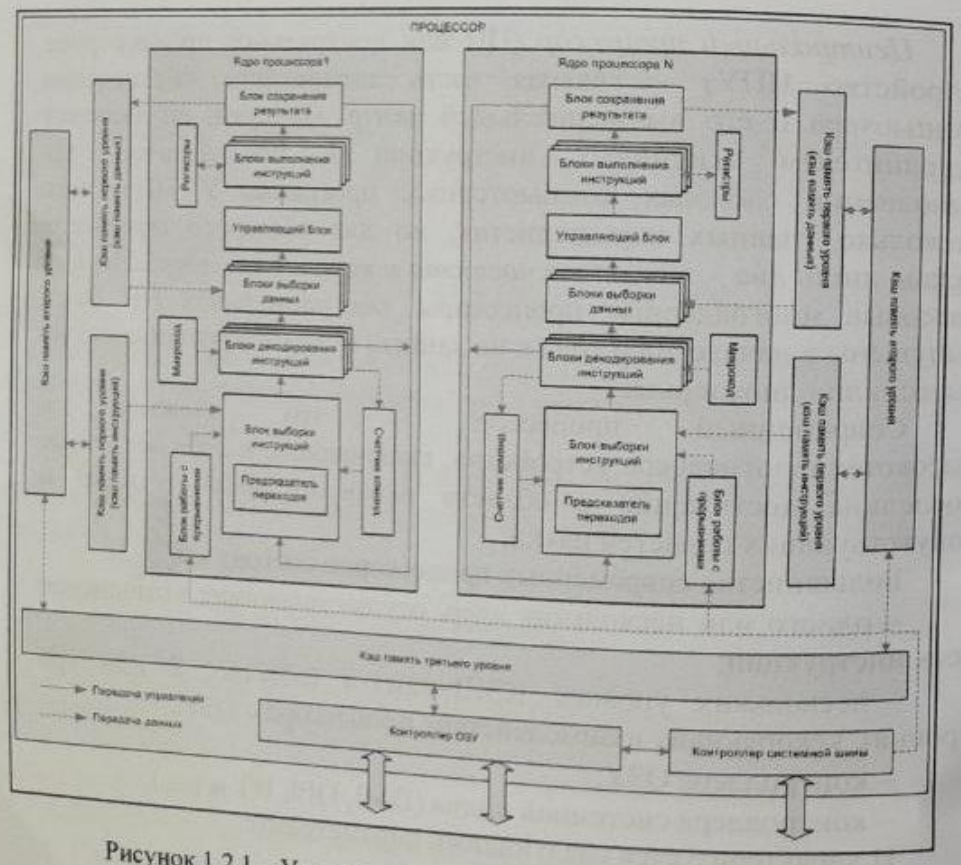


Рисунок 1.2.1 – Упрощенная структурная схема процессора

Ядро процессора

Ядро процессора – это его основная часть, содержащая все функциональные блоки и осуществляющая выполнение всех логических и арифметических операций.

На рисунке 1.2.1 приведена структурная схема устройства ядра процессора. Как видно на рисунке, каждое ядро процессора состоит из нескольких функциональных блоков:

– блока выборки инструкций;
– блоков декодирования инструкций;
– блоков выборки данных;
– управляющего блока;
– блоков выполнения инструкций;
– блоков сохранения результатов;
– блока работы с прерываниями;
– ПЗУ, содержащего микрокод;
– набора регистров;
– счетчика команд.

Блок выборки инструкций осуществляет считывание инструкций по адресу, указанному в счетчике команд. Обычно, за такт он считывает несколько инструкций. Количество считываемых инструкций обусловлено количеством блоков декодирования, так как необходимо на каждом такте работы максимально загрузить блоки декодирования. Для того чтобы блок выборки инструкций работал оптимально, в ядре процессора имеется предсказатель переходов.

Предсказатель переходов пытается определить, какая последовательность команд будет выполняться после совершения перехода. Это необходимо, чтобы после условного перехода максимально нагрузить конвейер ядра процессора.

Блоки декодирования, как понятно из названия, – это блоки, которые занимаются декодированием инструкций, т.е. определяют, что надо сделать процессору, и какие дополнительные данные нужны для выполнения инструкции. Задача эта для большинства современных коммерческих процессоров, построенных на базе концепции CISC, – очень сложная. Дело в том, что длина инструкций и количество операндов – нефиксированные, и это сильно усложняет

жизнь разработчикам процессоров и делает процесс декодирования нетривиальной задачей.

Часто отдельные сложные команды приходится заменять микрокодом – серией простых инструкций, в совокупности выполняющих то же действие, что и одна сложная инструкция. Набор микрокода прошит в ПЗУ, встроенном в процессоре. К тому же микрокод упрощает разработку процессора, так как отпадает надобность в создании сложноустроенных блоков ядра для выполнения отдельных команд, да и исправить микрокод гораздо проще, чем устранить ошибку в функционировании блока.

В современных процессорах, обычно, бывает 2–4 блока декодирования инструкций, например, в процессорах Intel Core 2 каждое ядро содержит по два таких блока.

Блоки выборки данных осуществляют выборку данных из КЭШ–памяти или ОЗУ, необходимых для выполнения текущих инструкций. Обычно, каждое процессорное ядро содержит несколько блоков выборки данных. Например, в процессорах Intel Core используется по два блока выборки данных для каждого ядра.

Управляющий блок на основании декодированных инструкций управляет работой блоков выполнения инструкций, распределяет нагрузку между ними, обеспечивает своевременное и верное выполнение инструкций. Это один из наиболее важных блоков ядра процессора.

Блоки выполнения инструкций включают в себя несколько разнотипных блоков:

ALU – арифметическое логическое устройство;

FPU – устройство по выполнению операций с плавающей точкой;

Блоки для обработки расширения наборов инструкций. Дополнительные инструкции используются для ускорения обработки потоков данных, шифрования и дешифрования, кодирования видео и так далее. Для этого в ядро процессора вводят дополнительные регистры и наборы логики.

На данный момент наиболее популярными расширениями наборов инструкция являются:

MMX (Multimedia Extensions) – набор инструкций, разработанный компанией Intel, для ускорения кодирования и декодирования потоковых аудио и видео–данных;

SSE (Streaming SIMD Extensions) – набор инструкций, разработанный компанией Intel, для выполнения одной и той же последовательности операций над множеством данных с распараллеливанием вычислительного процесса. Наборы команд постоянно совершенствуются, и на данный момент имеются ревизии: SSE, SSE2, SSE3, SSSE3, SSE4;

ATA (Application Targeted Accelerator) – набор инструкций, разработанный компанией Intel, для ускорения работы специализированного программного обеспечения и снижения энергопотребления при работе с такими программами. Эти инструкции могут использоваться, например, при расчете контрольных сумм или поиска данных;

3DNow – набор инструкций, разработанный компанией AMD, для расширения возможностей набора инструкций MMX;

AES (Advanced Encryption Standard) – набор инструкций, разработанный компанией Intel, для ускорения работы приложений, использующих шифрование данных по одноименному алгоритму.

Блок сохранения результатов обеспечивает запись результата выполнения инструкции в ОЗУ по адресу, указанному в обрабатываемой инструкции.

Блок работы с прерываниями. Работа с прерываниями – одна из важнейших задач процессора, позволяющая ему своевременно реагировать на события, прерывать ход работы программы и выполнять требуемые от него действия. Благодаря наличию прерываний, процессор способен к псевдопараллельной работе, т.е. к, так называемой, многозадачности.

Обработка прерываний происходит следующим образом. Процессор перед началом каждого цикла работы проверяет наличие запроса на прерывание. Если есть прерывание для обработки, процессор сохраняет в стек адрес инструкции, которую он должен был выполнить, и данные, полученные после выполнения последней инструкции, и переходит к выполнению функции обработки прерывания.

После окончания выполнения функции обработки прерывания, из стека считываются сохраненные в него данные, и процессор возобновляет выполнение восстановленной задачи.

Регистры – сверхбыстрая оперативная память (доступ к регистрам в несколько раз быстрее доступа к КЭШ–памяти)

небольшого объема (несколько сотен байт), входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций. Регистры процессора делятся на два типа: регистры общего назначения и специальные регистры.

Регистры общего назначения используются при выполнении арифметических и логических операций, или специфических операций дополнительных наборов инструкций (MMX, SSE и т.д.).

Регистры специального назначения содержат системные данные, необходимые для работы процессора. К таким регистрам относятся, например, регистры управления, регистры системных адресов, регистры отладки и т.д. Доступ к этим регистрам жестко регламентирован.

Счетчик команд – регистр, содержащий адрес команды, которую процессор начнет выполнять на следующем такте работы.

В зависимости от типов обрабатываемых инструкций и способа их исполнения, процессоры подразделяются на несколько групп:

- на классические процессоры CISC;
- на процессоры RISC с сокращенным набором команд;
- на процессоры MISC с минимальным набором команд;
- на процессоры VLIW с набором сверхдлинных команд.

CISC (Complex instruction set computer) – это процессоры со сложным набором команд. Архитектура CISC характеризуется:

- сложными и многоплановыми инструкциями;
- большим набором различных инструкций;
- нефиксированной длиной инструкций;
- многообразием режимов адресации.

Исторически, процессоры с архитектурой CISC появились первыми, и их появление было обусловлено общей тенденцией разработки первых ЭВМ. ЭВМ стремились сделать более функциональными и в то же время простыми для программирования. Естественно, для программистов вначале было удобнее иметь широкий набор команд, чем реализовывать каждую функцию целой отдельной подпрограммой. В результате, объем программ сильно сокращался, а вместе с ним и трудоемкость программирования.

Однако такая ситуация продолжалась недолго. Во–первых, с появлением языков высокого уровня отпала необходимость

непосредственного программирования в машинных кодах и на ассемблере, и, во–вторых, со временем количество различных команд сильно выросло, а сами инструкции усложнились. В результате, большинство программистов, в основном, использовали какой–то определенный набор инструкций, практически игнорируя наиболее сложные инструкции.

В результате, программисты уже не имели особой выгоды от широкого набора инструкций, так как компиляция программ стала автоматической, а сами процессоры обрабатывали сложные и разнообразные инструкции медленно, в основном, из–за проблем с их декодированием.

К тому же новые сложные инструкции разработчики процессоров отлаживали меньше, так как это был трудоемкий и сложный процесс. В результате, некоторые из них могли содержать ошибки.

Ну и, естественно, чем сложнее инструкции, чем больше действий они выполняют, тем сложнее их выполнение распараллеливать, и, соответственно, тем менее эффективно они загружают конвейер процессора.

Однако к этому моменту уже было разработано огромное количество программ для процессоров с CISC архитектурой, поэтому экономически было невыгодно переходить на принципиально новую архитектуру, даже дающую выигрыш в производительности процессора.

Поэтому был принят компромисс, и CISC процессоры, начиная с Intel486DX, стали производить с использованием RISC–ядра. Т.е., непосредственно перед исполнением, сложные CISC–инструкции преобразуют в более простой набор внутренних инструкций RISC. Для этого используют записанные в размещенном внутри ядра процессора ПЗУ наборы микрокоманд – серии простых инструкций, в совокупности выполняющих те же действия, что и одна сложная инструкция.

RISC (Reduced Instruction Set Computer) – процессоры с сокращенным набором инструкций.

В концепции RISC–процессоров предпочтение отдается коротким, простым и стандартизированным инструкциям. В результате, такие инструкции проще декодировать и выполнять, а, следовательно, устройство процессора становится так же проще, так

как не требуется сложных блоков для выполнения нестандартных и многофункциональных инструкций. В результате, процессор становится дешевле, и появляется возможность дополнительно поднять его тактовую частоту, за счет упрощения внутренней структуры и уменьшения количества транзисторов, или снизить энергопотребление.

Так же простые RISC–инструкции гораздо проще распараллеливать, чем CISC–инструкции, а, следовательно, появляется возможность больше загрузить конвейер, ввести дополнительные блоки обработки инструкций и т.д.

Процессоры, построенные по архитектуре RISC, обладают следующими основными особенностями:

– фиксированная длина инструкций;

– небольшой набор стандартизированных инструкций;

– большое количество регистров общего назначения;

– отсутствие микрокода;

– меньшее энергопотребление, по сравнению с CISC–процессорами аналогичной производительности;

– более простое внутреннее устройство;

– меньшее количество транзисторов, по сравнению с CISC–процессорами аналогичной производительности;

– отсутствие сложных специализированных блоков в ядре процессора.

В результате, хотя RISC–процессоры и требуют выполнения большего количества инструкций для решения одной и той же задачи, по сравнению с CISC–процессорами, они, в общем случае, показывают более высокую производительность. Во–первых, выполнение одной RISC–инструкции занимает гораздо меньше времени, чем выполнение CISC–инструкции. Во–вторых, RISC–процессоры более широко используют возможности параллельной работы. В–третьих, RISC–процессоры могут иметь более высокую тактовую частоту, по сравнению с CISC–процессорами.

Однако, несмотря на явное преимущество RISC, процессоры не получили столь серьезного распространения, как CISC. Правда, связано это в основном не с тем, что они по каким–то параметрам могли быть хуже CISC–процессоров. Они не хуже. Дело в том, что

CISC–процессоры появились первыми, а программное обеспечение для CISC –процессоров – несовместимо с RISC–процессорами.

В результате, экономически крайне невыгодно переписывать все программы, которые уже разработаны, отлажены и используются огромным количеством пользователей. Вот так и получилось, что теперь мы вынуждены использовать CISC–процессоры. Правда, как я уже говорил, разработчики нашли компромиссное решение данной проблемы, и уже очень давно в CISC–процессорах используют RISC–ядро и замену сложных команд на микропрограммы. Это позволило несколько сгладить ситуацию. Но все же RISC–процессоры по большинству параметров выигрывают даже у CISC–процессоров с RISC–ядром.

MISC (Minimal Instruction Set Computer) – дальнейшее развитие архитектуры RISC, основанное на еще большем упрощении инструкций и уменьшении их количества. Так, в среднем, в MISC–процессорах используется 20–30 простых инструкций. Такой подход позволил еще больше упростить устройство процессора, снизить энергопотребление и максимально использовать возможности параллельной обработки данных.

VLIW (Very long instruction word) – архитектура процессоров, использующая инструкции большой длины, содержащие сразу несколько операций, объединенных компилятором для параллельной обработки. В некоторых реализациях процессоров длина инструкций может достигать 128 или даже 256 бит.

Архитектура VLIW является дальнейшим усовершенствованием архитектуры RISC и MISC с углубленным параллелизмом.

Если в процессорах RISC организацией параллельной обработки данных занимался сам процессор, при этом, затрачивая часть ресурсов на анализ инструкций, выявление зависимостей и предсказание условных переходов (причем, зачастую, процессор мог ошибаться, например, в предсказании условных переходов, тем самым внося серьезные задержки в обработку инструкций, или просматривать код программы на недостаточную глубину для выявления независимых операций, которые могли бы выполняться параллельно), то в VLIW–процессорах задача оптимизации параллельной работы возлагалась на компилятор, который не был ограничен ни во времени, ни в ресурсах и мог проанализировать всю

программу для составления оптимального для работы процессора кода.

В результате, процессор VLIW выигрывал не только от упразднения накладных расходов на организацию параллельной обработки данных, но и получал прирост производительности, из–за более оптимальной организации параллельного выполнения инструкций.

Кроме этого упрощалась конструкция процессора, так как упрощались или вовсе упразднялись некоторые блоки, отвечающие за анализ зависимостей и организацию распараллеливания обработки инструкций, а это, в свою очередь, вело к снижению энергопотребления и себестоимости процессоров.

Однако даже компилятору тяжело справляться с анализом кода и организацией его распараллеливания. Часто код программы был сильно взаимозависимый, и, в результате, в инструкции компилятору приходилось вставлять пустые команды. Из–за этого программы для VLIW–процессоров могли быть гораздо длиннее, чем аналогичные программы для традиционных архитектур.

Первые VLIW–процессоры появились в конце 1980–х годов и были разработаны компанией Cydrome. Так же к процессорам с этой архитектурой относятся процессоры TriMedia фирмы Philips, семейство DSP C6000 фирмы Texas Instruments, Эльбру?с 2000 – процессор российского производства, разработанный компанией МЦСТ при участии студентов МФТИ и др. Поддержка длинных инструкций с явным параллелизмом есть и в процессорах семейства Itanium.

Для значительного ускорения вычислений, любой современный процессор оснащен встроенной памятью с очень быстрым доступом, которая предназначена для хранения данных, которые могут быть запрошены процессором с наибольшей вероятностью. Называется этот буфер кэшем и может быть первого (L1), второго (L2) или третьего (L3) уровня.

Самой быстрой памятью и по сути, неотъемлемой частью процессора, является кэш первого уровня, объем которого совсем невелик и составляет 128 Кб (64х2). Большинство современных ЦПУ без кэша L1 функционировать не могут.

Вторым по быстродействию следует L2–кэш и в объеме может достигать 1–12 Мб.

Ну и самым медленным, но зато и самым внушительным по размеру (может быть более 24 Мб) является кэш третьего уровня и имеется далеко не у всех процессоров.

Еще одним немаловажным моментом является понятие процессорного разъема или гнезда процессора, называемого сокетом (Socket), в который этот самый процессор устанавливается. Различные поколения или семейства ЦПУ, как правило, устанавливаются в свои уникальные разъемы и этот факт необходимо учитывать при подборе связки материнская плата – процессор.

Из–за сложности и высокотехнологичности производства, высочайшим требованиям к качеству продукции, конкурентоспособных компаний выпускающих центральные процессоры не так уж и много, а для рынка настольных ПК так и всего две – *Intel u AMD*. Их давнее соперничество началось еще в начале 90–ых, правда за эти 20 лет доля продаваемых процессоров компанией AMD, всегда была значительно ниже доли Intel. Тем не менее, продукция Advanced Micro Devices всегда отличалась привлекательным соотношением производительность/цена при достаточно демократичной розничной стоимости своей продукции, что дает ей возможность достаточно уверенно удерживать свою долю рынка, равной около 19% от общемировой доли.

Для удобства позиционирования на рынке, каждый производитель разделяет свою продукцию на различные семейства, в зависимости от возможностей и производительности процессоров.

AMD

– *Sempron* – самый низкобюджетный процессор для настольных ПК и мобильных устройств являющийся прямым конкурентом процессорам Celeronкомпании Intel. Основной нишей данного процессора являются простые приложения для повседневной работы.

– *PhenomII*– многоядерное семейство высокопроизводительных процессоров, предназначенных для решения любых задач. Является флагманской линейкой для настольных компьютеров и содержит в себе процессоры с количеством ядер от 2 до 6.

– *AthlonII* – многоядерное семейство процессоров, созданное

как очень бюджетная альтернатива более дорогим процессорам серии Phenom II. Предназначен для решения повседневных задач и ориентирован как вариант для "бюджетных" игровых систем и ПК с весьма приличной производительностью.

– *A–Series*– четырехъядерное семейство процессоров. Отличительной чертой данной серии служит встроенная в ядро процессора, графическая видеокарта Radeon.

– *Ryzen*—микропроцессоры AMD второй половины 2010–х годов. Данное семейство процессоров относится к архитектуре x86_64, применяется в настольных, мобильных и встроенных вычислительных системах и на данный момент использует процессорные микроархитектуры Zen, Zen+, Zen 2, Zen 3, Zen 4.

Компания AMD с выходом процессоров Ryzen стала использовать новые наименования продукции, который применяет логику Intel. До этого для процессоров FX маркировка была иная. Разберем на примерах, что означают цифры и буквы в названии процессоров AMD (рис. 1.2.2).

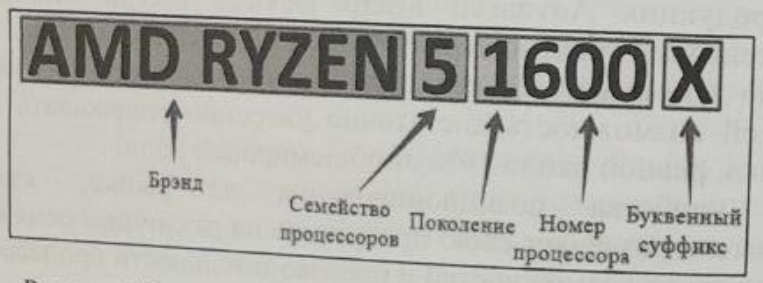


Рисунок 1.2.2 – Расшифровка процессоров AMD RYZEN

Брэнд

Эта часть состоит из названия компании и бренда процессоров. Кроме Ryzen, есть еще менее производительные процессоры Athlon, профессиональные Ryzen Threadripper и серверные Epyc. Что интересно, Intel и AMD при переходе на новые процессоры не отказались от своих прошлых именитых марок Celeron, Pentium и Athlon. Эти процессоры сейчас занимают самый доступный сегмент в линейках обеих компаний.

Семейство процессоров

Все Ryzen делятся на несколько семейств по уровню производительности. Чем выше цифра, тем мощнее процессор:

– Ryzen 3 — начальный уровень,
– Ryzen 5 — средний уровень,
– Ryzen 7 — предтоповый уровень,
– Ryzen 9 — топовый уровень.

Основные отличия заключаются в количестве ядер и потоков. Кроме того, могут различаться тактовые частоты, объем кэш-памяти и другие характеристики.

Существуют также процессоры с припиской PRO, например, Ryzen 7 PRO 3700. Это процессоры для корпоративных пользователей, поддерживают технологии шифрования и дополнительные функции безопасности. Но никто не запрещает их использовать и в домашних системах.

Поколение

Здесь важно не путать поколение процессоров и поколение архитектуры Zen, на которой эти процессоры основаны:

– 1-е поколение Ryzen — архитектура Zen;
– 2-е поколение Ryzen — архитектура Zen +;
– 3-е поколение Ryzen — архитектура Zen 2;
– 5-е поколение Ryzen — архитектура Zen 3.

Обратите внимание, что 4-е поколение состоит из APU и мобильных процессоров и оно все еще основано на Zen 2. Кроме того, процессоры в рамках одного поколения могут быть основаны на разной архитектуре. Так, мобильные процессоры Ryzen 3 5300U, Ryzen 5 5500U и Ryzen 7 5700U — это Zen 2.

Разница между поколениями выражается в производительности, в первую очередь за счет доработки архитектуры. Это и лучшая работа с памятью, и рост производительности на ядро, увеличение максимальной тактовой частоты. А вот число ядер в основном не меняется. Так, Ryzen 5 1600 и Ryzen 5 5600 имеют по 6 ядер и 12 потоков.

Номер процессора

В англоязычных странах этот пункт называется SKU (Stock Keeping Unit), что можно перевести на русский как артикул. Этот номер показывает положение конкретного процессора в рамках одного семейства. Чем больше число, тем лучше процессор. Встречается и еще более детальное наименование, причем разница

может быть существенной. Например, у Ryzen 9 3900X 12 ядер, а у 3950X уже 16.

Обратите внимание, что цифры не повторяются в разных семействах: 3600 — это всегда Ryzen 5, а 3700 — Ryzen 7. Не бывает Ryzen 5 3700 или Ryzen 7 3600.

Буквенный суффикс

Суффикса может и не быть, в таком случае перед вами обычный десктопный процессор. Возможность разгона никак не обозначается, так как все настольные процессоры Ryzen имеют разблокированный множитель.

- G — есть встроенная графика;
- E — энергоэффективный процессор со сниженным теплопакетом;
- GE — энергоэффективный процессор со сниженным теплопакетом и встроенной графикой;
- X — процессоры с более высокими тактовыми частотами, по сути, разогнанные производителем;
- XT — еще более производительные процессоры с большими максимальными частотами;
- H — производительная серия для ноутбуков;
- HX — еще более производительная серия процессоров для ноутбуков;
- HS — особая серия процессоров AMD, производительность которой равна серии H, но теплопакет снижен;
- U — энергоэффективная серия для ноутбуков со сниженным теплопакетом.

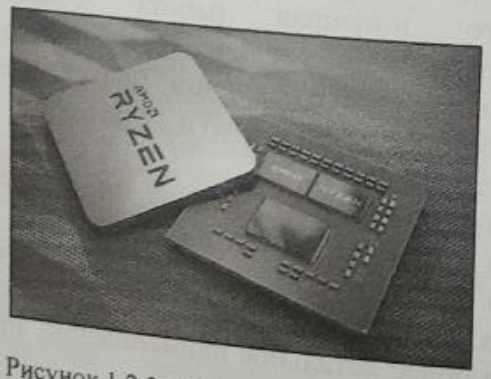


Рисунок 1.2.3 – Процессор AMD Ryzen

INTEL

– *Celeron* – большое семейство низкобюджетных процессоров, предназначенное для использования в домашних и офисных компьютерах начального уровня.

– *PentiumDual–Core* – устаревшее семейство бюджетных двухъядерных процессоров для недорогих домашних и офисных систем. Не смотря на то, что процессоры этой серии до сих пор повсеместно продаются, большинство пользователей в нынешнее время делает свой выбор в пользу более актуального и рентабельного Core i3.

– *Core i3* – новое поколение двухъядерных процессоров начального и среднего уровня цены и производительности. Призваны заменить морально устаревшие Pentium Dual–Core на архитектуре старого поколения Intel Core 2.

– *Core i5* – семейство процессоров среднего уровня цены и производительности. ЦПУ данной серии могут содержать 2 или 4 ядра и в большинстве своем встроенную графическую карту. Отличное решение для «игровых» и мультимедийных систем. Поддерживают технологию TurboBoost, которая заключается в автоматическом разгоне процессора под нагрузкой (рис. 1.2.4).

– *Core i7* –линейка процессоров от компании Intel. Устанавливаются в высокопроизводительные системы, предназначенные для решения задач любой сложности. Поддерживает Turbo Boost, с которой процессор автоматически увеличивает производительность тогда, когда это необходимо.

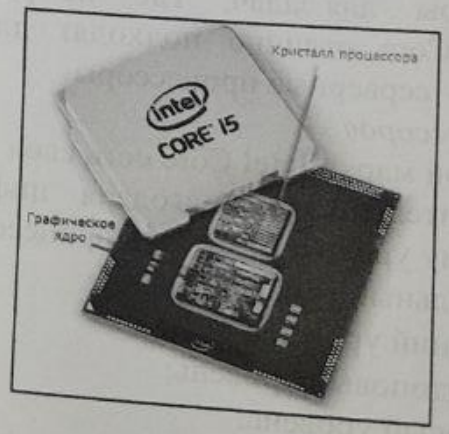


Рисунок 1.2.4 – Процессор Intel Core

– *Intel Core i9* — флагманская линейка семейство процессоров Intel с архитектурой X86–64. Модельный ряд был представлен в мае 2017 г. как решение для высокопроизводительных ПК. Линейка разработана на основе микроархитектуры Skylake.

У Intel довольно простая схема наименования процессоров. Ценовая категория, производительность, наличие встроенного видеоядра и другие параметры зашифрованы в названии. Например, Intel Core i5–9600K. Однако неподготовленного покупателя это может запутать, давайте подробно рассмотрим маркировку процессоров Intel на конкретных примерах (рис. 1.2.5).

Рисунок 1.2.5 – Расшифровка процессоров Intel Core

Под брэндом или торговой маркой подразумевается как название компании, так и процессора. У Intel есть множество разновидностей процессоров: Celeron, Pentium, Core и Xeon, каждый из которых решает свою задачу. Так, Celeron и Pentium — доступные процессоры для задач, где не требуется высокая производительность, Core отлично подходят для игр и рабочих приложений, а Xeon — серверные процессоры.

Семейство процессоров

В рамках торговой марки Intel Core есть своя дифференциация по уровню производительности. Благодаря цифрам в названии можно понять, к какому уровню относится процессор:

- Core i3 — начальный уровень;
- Core i5 — средний уровень;
- Core i7 — предтоповый уровень;
- Core i9 — топовый уровень.

Основные отличия заключаются в количестве ядер и потоков. Кроме того, могут различаться тактовые частоты, объем кэш-памяти и другие характеристики.

Поколение

Чем новее процессор, тем лучше. В 2021 году актуально 11-е поколение процессоров. Но это не значит, что все предыдущие сразу же устарели. Важно знать отличия между поколениями, так как характеристики постоянно меняются. Так, в седьмом поколении процессоры Core i5 имели всего 4 ядра, но в восьмом поколении их стало уже 6.

Номер процессора

В англоязычных странах этот пункт называется SKU (Stock Keeping Unit), что можно перевести на русский как артикул. По номеру можно понять положение конкретного процессора в своем семействе. Они отличаются в основном по базовой и максимальной тактовой частоте, а также объему кэш-памяти. Чем эта цифра больше, тем мощнее процессор. Проще говоря, i5–9600 лучше, чем i5–9400. Обратите внимание, что цифры не повторяются в разных семействах: 9600 — это всегда i5, а 9100 — i3. Не бывает i5–9100 или i3–9600.

В более старых поколениях часто встречалось еще более детальное обозначение, например Core i7–4770K. Также это распространено в современных мобильных версиях. Причем отличия более существенные, чем у настольных процессоров. Например, у i7–10850H только 6 ядер, а у i7–10870H уже 8.

Буквенный суффикс

Стоит сразу отметить, что его может не быть вообще — i3–9100, i7–8700 и т. д. Это значит, что процессор не имеет каких-либо специфических обозначений. Перед нами стандартный CPU для настольных ПК.

- K — разблокированный множитель, то есть процессор можно разогнать при наличии материнской платы на Z-чипсете;
- F — отсутствие встроенного видеоядра, то есть без отдельной видеокарты не обойтись;
- G1-G7 — процессоры с новой интегрированной графикой Intel Iris X;
- G — идет в комплекте с дискретной графикой, например,

на платформе Intel NUC;

– X или XE — процессоры из X-линейки. От обычных Core отличаются сокетом и большим количеством ядер;

– H — производительная серия для ноутбуков;

– HK — производительная серия для ноутбуков с разблокированным множителем;

– HQ — производительная серия для ноутбуков, 4 ядра;

– T — настольные процессоры со сниженным теплопакетом;

– U — энергоэффективная серия для ноутбуков со сниженным теплопакетом;

– Y — энергоэффективная серия для ноутбуков с максимально сниженным теплопакетом.

Возможны и различные комбинации вроде i7−10700KF, что означает отсутствие встроенной графики и поддержку разгона.

Практическая часть

Для выполнения практической работы необходимо сделать скриншот окна CPU в программе CPU-Z, выписать характеристики процессора, которые указаны в окне (рис 1.2.6)

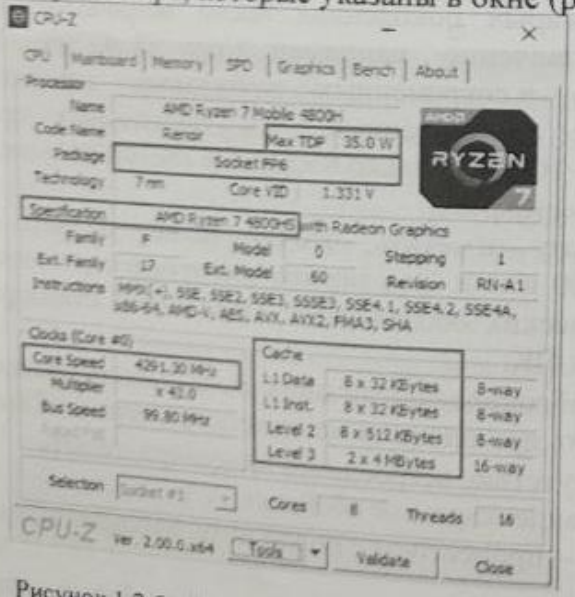


Рисунок 1.2.6 – Скриншот программы CPU–Z

Процессор AMD Ryzen 7 4800 HS with Radeon Graphics

– AMD Ryzen – процессор относится к семейству Ryzen и произведен компанией AMD;

– Socket FP6 – процессор устанавливается в разъем типа FP6;

– 7 4800 HS – модель процессора;

– 4.3GHz – тактовая частота процессора в данный момент времени (чем она выше, тем процессор быстрее);

– L3 2x4Mb –процессор имеет кэш третьего уровня, который равен 4 мегабайтам;

– Max TDP: 35 W – Теплопакет – это значение, которое используют в очень широком смысле Intel и AMD для обозначения информации о тепловыделении своих продуктов, у этого процессора значение TDP это 35 W.

Сравнение процессоров

Наименование	Ryzen 7 PRO 4750G	Core i7-9700K
Ядро	Renoir	Coffee Lake
Количество ядер	8	8
Количество потоков	16	
Техпроцесс, нм	7	14
Разъем	AM4	LGA 1151-2
Частота, МГц	3600 (4,4 ГГц ТС)	3600 (4900 TB)
Множитель	36	36
HTT/bclk	100 МГц	100
Тип памяти	DDR4-3200	DDR4-2666
кэш L1, КБ	8 x (32 + 32)	32+32 x8
кэш L2, КБ	8 x 512	256x8
кэш L3, КБ	8192	12228
Напряжение питания, В		?
TDP, Вт	65	95
Кол-во транзисторов, млн		?
Площадь кристалла, кв. мм	156	?
Предельная температура, °C		100
Набор инструкций	SSE4.1, SSE4.2, SSE4A, AVX, AVX2.0, FMA3, AES	SSE4.1, SSE4.2, AES, AVX, AVX2
Прочие особенности	Технологии защиты, разблокирован множитель	Множитель разблокирован на повышение
Дата выпуска	21.07.2020	
Стоимость, $	309	

Рисунок 1.2.7 – Сравнение процессоров

Задания для выполнения практической работы № 2

1. Расшифровать запись своего процессора, со скриншота сделанного в программе CPU–Z (рис 1.2.6);

2. Найти в интернете изображение и приготовить небольшой теоретический материал описывающий подробные характеристики процессора;

3. Сравнить характеристики процессора с любым другим процессором, как показано на рисунке 1.2.7, по ссылке https://www.overclockers.ua/cpu/info/intel/pentium–g620/, сделать скриншот. Если нет вашего процессора в списке, подберите ближайший по иерархии процессор из списка.

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 2.1 – …»

Контрольные вопросы

1. Укажите основные факторы повышения производительности 32–разрядного микропроцессора 80386.
2. Перечислите функции процессора
3. Прокомментируйте основные параметры процессора
4. Какие современные типы процессоров вы знаете?
5. Дайте описание процессору i3
6. Дайте описание процессору i5
7. Дайте описание процессору i7
8. Дайте описание процессору *Sempron*
9. Дайте описание процессору *PhenomII*
10. Дайте описание процессору *Ryzen*

ГЛАВА II
АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

2.1. Арифметические основы организации компьютера. Выполнение арифметических операций в разных системах счисления

Цель работы: Получить представление об арифметических операциях. Выполнение арифметических операций в разных системах счислений.

Теоретическая часть

Система счисления – символический метод записи чисел или способ представления чисел с помощью письменных знаков, именуемых цифрами.

Число – это некоторая абстрактная сущность для описания количества чего–либо.

Цифры – это знаки, используемые для записи чисел.

Поскольку чисел гораздо больше, чем цифр, то для записи числа обычно используется набор (комбинация) цифр.

Здесь значение каждого числа зависит от его положения в номере (позиции). Например, 23=2·10+3; 32=3·10+2.

Значение (число) различных чисел, используемых для представления числа в позиционной системе, является основой системы счисления R. Значения чисел от 0 до (R–1).

Как правило, требуемое число N может быть выражено в следующем порядке в системе счисления на основе «R»:

$$N=a_m\cdot R^m+a_{m-1}\cdot R^{m-1}+\ldots+a_0\cdot R^0+\ldots+a_{-1}\cdot R^{-1}+\ldots+a_{-s}\cdot R^{-s} \qquad (2.1.1)$$

Следующие индексы указывают на местоположение числа и на возводимую степень: положительные значения индексов указывают всю часть числа (m–цифры, от старшего разряда к самому младшему - 0), а отрицательные – дробную часть (s–цифры, от старшего разряда – 0, к младшему разряду s).

Основные системы счисления

Двоичная система счисления. В компьютерной технике в основном используется двоичная система счисления. Такую систему очень легко реализовать в цифровой микроэлектронике, так как для нее требуется всего два устойчивых состояния (0 и 1).

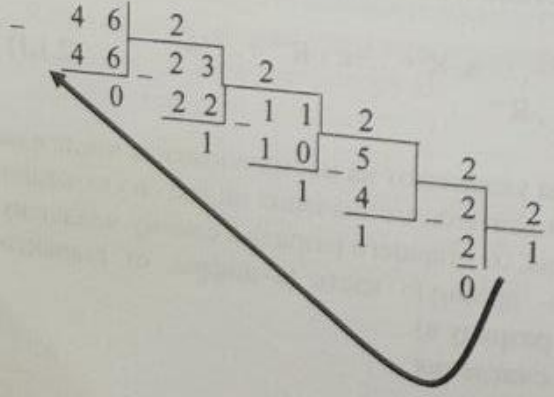
Двоичная система счисления может быть непозиционной и позиционной. Реализовано это может быть присутствием какого-либо физического явления или его отсутствием. Например: есть электрический заряд или его нет, есть напряжение или нет, есть ток или нет, есть сопротивление или нет, отражает свет или нет, намагничено или не намагничено, есть отверстие или нет и т. п.

Например, основой двоичной системы счисления является R = 2, и для представления данных используются только два числа: 0 и 1, следовательно число в двоичной системе счисления в приведенном примере будет переводиться следующим образом:
$101110.101_2 = 1\cdot2^5+0\cdot2^4+1\cdot2^3+1\cdot2^2+1\cdot2^1+0\cdot2^0+1\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3} = 46.625_{10}$

Таким образом, формула (2.1) может быть использована для преобразования числа из любой позиционной системы в десятичную систему. Сложно использовать формулу (2.1) для преобразования десятичной системы в любую основанную систему счисления. Для упрощения целесообразно преобразовать целую часть десятичного числа в отдельную двоичную систему, а дробную часть – в отдельную двоичную систему.

Целая часть, а затем разделенная часть также делятся на базу системы серийных номеров «R». Если результат последовательных делений становится равным «0», процесс останавливается. Например, 46.62510 → (2) конвертация из десятичного в двоичное.

Перевод целого числа 46 отдельно.



Остатки выписываются справа налево: $101110 = 46_{10}$

Для преобразования дробной части в двоичную систему счисления необходимо выписать дробную часть: 0.625

Далее необходимо провести черту между целой частью и дробной частью, после этого умножать дробную часть на 2, если число превышает дробный разряд, то в левой части выходит 1, если не превышает дробный разряд то остается 0.

0	625	·2
1	250	·2
0	500	·2
1	000	

$$0.625_{10} = 0.101_2$$

Восьмеричная система счисления – позиционная целочисленная система счисления с основанием 8. Для представления чисел в ней используются цифры от 0 до 7.

Восьмеричная система счисления часто используется в областях, связанных с цифровыми устройствами. Характеризуется легким переводом восьмеричных чисел в двоичные и обратно, путем замены восьмеричных чисел на триады двоичных.

Ранее эта система широко использовалась в программировании и компьютерной документации, однако в настоящее время почти полностью вытеснена шестнадцатеричной системой.

Для перевода двоичного числа в восьмеричное исходное число разбивают на триады влево и вправо от запятой; отсутствующие крайние цифры дополняют нулями. Затем каждую триаду записывают восьмеричной цифрой (см. табл. 2.1).

Пример: иллюстрация перевода двоичного числа в восьмеричное число:

$$N = (110011{,}100010)_2 = \left(\overbrace{110}^{6}\ \overbrace{011}^{3}, \overbrace{100}^{4}\ \overbrace{010}^{2}\right)_2 = (63{,}42)_8$$

Для осуществления обратного перевода из 16-ричной системы счисления в двоичную, каждое число разбивается на 3 разряда в двоичной системе счисления.

Шестнадцатеричная система счисления – позиционная система счисления по целочисленному основанию 16. Обычно в качестве шестнадцатеричных цифр используются десятичные цифры от 0 до 9 и латинские буквы от A до F для обозначения цифр от 1010_2 до 1111_{10}, то есть 16.

Для перевода двоичного числа в шестнадцатеричное исходное число разбивают на тетрады влево и вправо от запятой; отсутствующие крайние цифры дополняют нулями. Затем каждую тетраду записывают шестнадцатеричной цифрой (см. табл. 2.1.1).

Пример: иллюстрация перевода двоичного числа в шестнадцатеричное число:

$$N = \left(\overbrace{0111}^{7}\ \overbrace{1010}^{A}\ \overbrace{1011}^{B}, \overbrace{1110}^{E}\ \overbrace{1111}^{F}\right)_2 = (7AB, EF)_{16}$$

Для осуществления обратного перевода из 16-ричной системы счисления в двоичную, каждое число разбивается на 4 разряды в двоичной системе счисления.

Таблица 2.1.1
Системы счисления

10–ая	16–ая	Двоичная			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Двоичная арифметика

В микропроцессорах (МП) сложение, вычитание и умножение выполняются как простые арифметичесике действия. Во многих МП существуют команды сложения и вычитания, однако они не имеют команд умножения и деления (например, Intel 8086, 8088).

Далее приведены примеры на сложение, вычитание и умножение двоичных чисел.

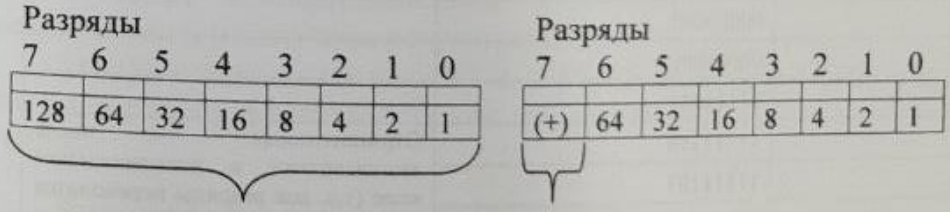
Сложение		Вычитание	
00111011	59	01010101	85
$^{+}$00101010	$^{+}$42	$^{-}$00111001	$^{-}$57
$1100101_{(2)}$	$101_{(10)}$	$00011100_{(2)}$	$28_{(10)}$

Умножение

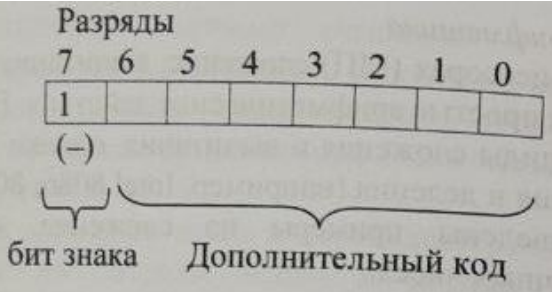
$1101_{(2)}$	$13_{(10)}$
$*\ 101_{(2)}$	$*\ 5_{(10)}$
$1101_{(2)}$	$65_{(10)}$
$+\ 0000_{(2)}$	
$1101_{(2)}$	
$1000001_{(2)}$	

Дополнительный код

Обычно компьютер обрабатывает информацию в двоичном коде. Если нужно обработать знаковое число, то используется дополнительный код. Для того, чтобы объяснить дополнительный код, изобразим регистры МП или ячейки памяти:



а) расположение двоичных позиций; б) расположение положительных чисел



в) расположение отрицательных чисел

Выше приведена структура образца 8–разрядного регистра. Обычно седьмой бит считается знаковым. Если число положительное пишется –"0", если отрицательное – "1" (таб. 2.1.2).

Таблица 2.1.2

Положительные и отрицательные числа в двоичной системе

Десятичные	Знаковые числа	Приложение
+127	0111 1111	Положительные числа записаны в прямом двоичном виде
...	...	
+8	0000 1000	
+7	00000111	
+6	00000110	
+5	00000101	
+4	00000100	
+3	00000011	
+2	00000010	
+1	00000001	
0	00000000	
–1	11111111	
–2	11111110	
–3	11111101	Отрицательные числа записываются в дополниельном коде (т.е. все разряды переводятся на обратный код (инверсия) и к маленькому (последнему) разряду прибавляется 1).
–4	11111100	
–5	11111011	
–6	11111010	
–7	11111001	
–8	11111000	
...	...	
–128	10000000	

Арифметика в дополнительном коде

Причина выполнения операций МП в дополнительном коде – это существование возможности выполнения операций инверсия (получение обратного кода) и инекрементация (добавление "1" к младшему разряду). Не умеет выполнять операции с прямым кодом. В его структуре имеются только сумматоры, поэтому МП для выполнения операций вычитания пользуется дополнительнм кодом.

Сложим числа "5" и "3" (в дополнительном коде).

Десятичный вид:	Двоичный вид:
(+5)	00000101
+ (+3)	+00000011
(+8)	00001000 $_{(2)}$=8$_{(10)}$

Дополнительный код положитльныхх чисел равняется их прямому коду.

Сложим числа "+7" и "–3". В дополнительном коде данные числа будут выглядеть так: +7$_{(10)}$=0000 0111$_{(2)}$ и –3$_{(10)}$=1111 1101$_{(2)}$. Выполним сложение:

1 число	(+7)	0000 0111
	+	+
2 число	(–3)	1111 1101
	(+4)	100000100

переполнение

В связи с переполнением 8–разрядного регистра, "1" выбрасывается и получаем результат: 0000 0100$_{(2)}$ т.е, "+4$_{(10)}$"

Теперь сложим числа "+3" и "–8". В дополнительном коде данные числа будут выглядеть так: +3$_{(10)}$=0000 0111$_{(2)}$ и –8$_{(10)}$=1111 1101$_{(2)}$.

Выполним сложение:

1 число	(+3)	0000 0011
	+	+
2 число	(–8)	1111 1000
	(–5)	1111
		1011

Практическая часть

В примерах приведено решение для варианта № 35.

Объяснение: Вместо буквы X нужно вставить свой номер из списка. В нужных местах перевести данное число на указанную систему счисления.

Двоичная система счисления:

011001X, Вариант № 35 – в двоичной системе счисления 100011, следовательно число будет выглядит так: 011001100011;

Шестнадцатеричная система счисления:

XF, Вариант № 35 – в шестнадцатеричной системе счисления 23, следовательно число будет выглядит так: 23F

Задание № 1. Перевод двоичный системы счисления в другую:

$$11011011X\ (2) \rightarrow (10) \text{ и } (16)$$
$$11011011100011_2 \rightarrow 14051_{10}$$
$$11011011100011_2 \rightarrow 36E3_{16}$$

Задание № 2. Перевод из десятичной системы счисления в двоичную:

$$23X\ (10) \rightarrow (2);$$
$$2335_{10} \rightarrow 100100011111_2$$

Задание № 3. Перевод из шестнадцатеричной системы счисления в двоичную:

$$XF\ (16) \rightarrow (2);$$
$$23F_{16} \rightarrow 1000111111_2$$

Задание № 4. Сложение двоичных чисел:

$$\begin{array}{rr} + & 010110X \\ & 0000X11 \end{array}$$

$$\begin{array}{rr} + & 010110100011 \\ & 000010001111 \\ & 011000110010 \end{array}$$

Задание № 5. Умножение двоичных чисел:

$$\begin{array}{rr} \times & 010110X \\ & 1X111 \end{array}$$

×	010110100011
	000010001111
·1	010110100011
·1	010110100011
·1	010110100011
·1	010110100011
·0	000000000000
·0	000000000000
·0	000000000000
·1	010110100011
·1	010110100011
	110010011000001101

Задания для выполнения практической работы № 3

1. Перевод двоичной системы счисления в другую:

a) 11011011X (2) → (10) и (16)

b) 0.101100X (2) → (10) и (16)

c) 110111,01X (2) → (10) и (16)

2. Преобразуйте следующие десятичные числа в двоичную систему счисления:

a) 23X (10) → (2);

b) 39X (10) → (2);

c) 55X (10) → (2);

d) 0,7X (10) → (2);

3. Преобразуйте следующие числа из шестнадцатеричной в двоичную систему счисления:

a) XF (16) → (2);

b) CXE (16) → (2);

c) 6DX (16) → (2).

4. Сложить следующие два двоичных числа:

а) 010110X + 0000X11

б) 001111X + 000X111

5. Умножить следующие два двоичных числа:

а)11X	б)11X	в)11X
	1X1	111
1X1		

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 3.1 – …»

Контрольные вопросы

1. Какие арифметические операции выполняет АЛУ?
2. Опишите метод преобразования с использованием весов разрядов.
3. В каком виде можно представить числа на компьютере?
4. В каком порядке выполняются сложение, вычитание и умножение микропроцессоров?

2.2. Численно–логические основы организации компьютера

Цель работы: Изучение элементов логических операций.
Выполнение логических операций.

Теоретическая часть

Арифметико–логическое устройство (АЛУ) — блок процессора, который под управлением устройства управления служит для выполнения арифметических и логических преобразований (начиная от элементарных) над данными, называемыми в этом случае операндами.

Обобщённая блок–схема арифметико–логического устройства (АЛУ) (рис 2.2.1). Стрелками указаны входные и выходные слова.

Флаги — признаки (например, результата сравнения операндов) выполнения предыдущей операции (вход) и результата выполнения текущей операции (выход). В одноместных операциях таких, например, как инверсия битов слова или битовый сдвиг второй операнд (В) не участвует в операции. Слово команды указывает необходимую операцию.

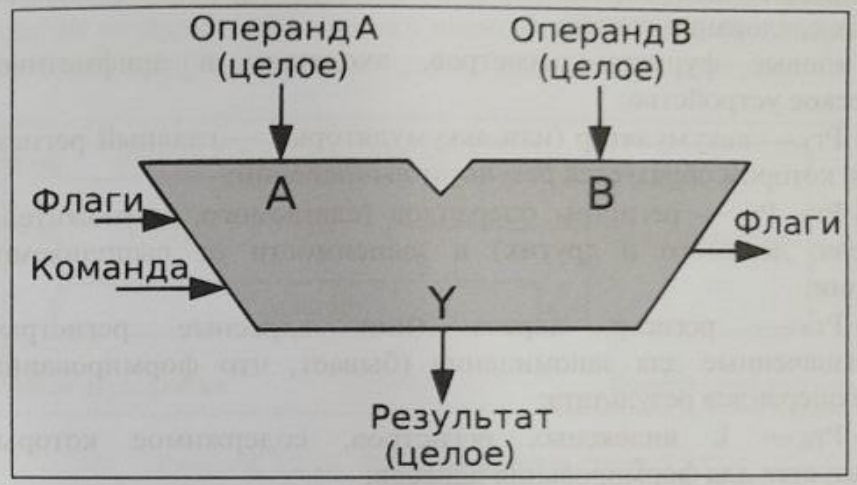


Рисунок 2.2.1 – Обобщённая блок–схема арифметико–логического устройства

Одноразрядное двоичное бинарное (двухоперандное) АЛУ с бинарным (двухразрядным) выходом может выполнять до 256 двоичных бинарных (двухоперандных) функций (операций) с бинарным (двухразрядным) выходом.

Арифметико–логическое устройство в зависимости от выполнения функций можно разделить на две части:

– микропрограммное устройство (устройство управления), задающее последовательность микрокоманд (команд);

– операционное устройство, в котором реализуется заданная последовательность микрокоманд (команд).

В состав арифметико–логического устройства условно включаются регистры $Рг_1$ — $Рг_7$, которые служат для обработки информации, поступающей из оперативной или пассивной памяти N_1, N_2, … N_S и логические схемы, которые используются для

обработки слов по микрокомандам, поступающим из устройства управления.

Различают два вида микрокоманд: внешние — такие микрокоманды, которые поступают в АЛУ от внешних источников и вызывают в нём преобразование информации и внутренние — те, которые генерируются в АЛУ и оказывают влияние на микропрограммное устройство, изменяя таким образом нормальный порядок следования команд.

Типовые функции регистров, входящих в арифметико–логическое устройство:

– $Рг_1$ — аккумулятор (или аккумуляторы) — главный регистр АЛУ, в котором образуется результат вычислений;

– $Рг_2$, $Рг_3$ — регистры операндов (слагаемого, сомножителя, делителя, делимого и других) в зависимости от выполняемой операции;

– $Рг_4$ — регистр адреса (или адресные регистры), предназначенные для запоминания (бывает, что формирования) адреса операндов результата;

– $Рг_6$ — k индексных регистров, содержимое которых используется для формирования адресов;

– $Рг_7$ — l вспомогательных регистров, которые по желанию программиста могут быть аккумуляторами, индексными регистрами или использоваться для запоминания промежуточных результатов.

Часть операционных регистров могут быть адресованы в команде для выполнения операций с их содержимым, и их называют программно–доступными. К таким регистрам относятся: сумматор, индексные регистры и некоторые вспомогательные регистры. Остальные регистры нельзя адресовать в программе, то есть они являются программно–недоступными.

Операционные устройства можно классифицировать по виду обрабатываемой информации, по способу её обработки и по логической структуре.

Такая сложная логическая структура АЛУ может характеризоваться количеством отличающихся друг от друга микроопераций, которые необходимы для выполнения всего комплекса задач, поставленных перед арифметико–логическим устройством. На входе каждого регистра собраны соответствующие

логические схемы, обеспечивающие такие связи между регистрами, что позволяет реализовать заданные микрооперации. Выполнение операций над словами сводится к выполнению определённых микроопераций, которые управляют передачей слов в АЛУ и действиями по преобразованию слов. Порядок выполнения микрокоманд определяется алгоритмом выполнения операций. То есть, связи между регистрами АЛУ и их функциями зависят в основном от принятой методики выполнения логических операций, в том числе арифметических или специальной арифметики.

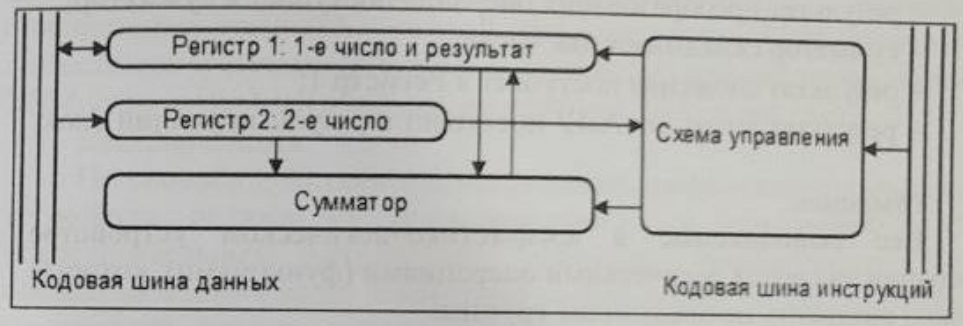


Рисунок 2.2.2 – Функциональная схема АЛУ

Пример работы АЛУ на операции сложения

Функционально АЛУ состоит из двух регистров (Регистр1, Регистр 2), схемы управления и сумматора (рис 2.2.2). Арифметическая операция выполняется по тактам:

– значения операнда 1, участвующего в арифметической операции по шине данных поступает в Регистр 1 или уже там находится;

– значения операнда 2, участвующего в арифметической операции по шине данных поступает в Регистр 2 или уже там находится;

– по шине инструкций поступает инструкция на выполнение операции в схему управления;

– данные из регистров поступают в сумматор, схема управления дает команду на выполнение сложения;

– результат сложения поступает в Регистр 1;

– признаки выполнения операции в АЛУ поступают в регистр флагов.

Пример работы АЛУ на операции вычитания:

– значение операнда 1, участвующего в арифметической операции по кодовой шине данных поступает в Регистр 1;

– значение операнда 2, участвующего в арифметической операции по кодовой шине данных поступает в Регистр 2;

– по кодовой шине инструкций, поступает инструкция на выполнение операции вычитания в схему управления;

– схема управления преобразовывает положительное число в отрицательное (в формате дополнительного кода до двух);

– результат преобразования операнда поступает в сумматор;

– сумматор складывает два числа;

– результат сложения поступает в Регистр 1;

– результат операции АЛУ поступает в результирующий блок

Операции

Все выполняемые в арифметико–логическом устройстве операции являются логическими операциями (функциями), которые можно разделить на следующие группы:

– операции двоичной арифметики для чисел с фиксированной точкой;

– операции двоичной (или шестнадцатеричной) арифметики для чисел с плавающей точкой;

– операции десятичной арифметики;

– операции индексной арифметики (при модификации адресов команд);

– операции специальной арифметики;

– операции над логическими кодами (логические операции);

– операции над алфавитно–цифровыми полями.

Современные компьютеры общего назначения обычно реализуют операции всех приведённых выше групп, а малые и микро ЭВМ, микропроцессоры и специализированные ЭВМ часто не имеют аппаратуры арифметики чисел с плавающей точкой, десятичной арифметики и операций над алфавитно–цифровыми полями. В этом случае эти операции выполняются специальными подпрограммами.

К арифметическим операциям относятся сложение, вычитание, вычитание модулей («короткие операции») и умножение и деление

(«длинные операции»). Группу логических операций составляют операции дизъюнкция (логическое ИЛИ) и конъюнкция (логическое И) над многоразрядными двоичными словами, сравнение кодов на равенство. Специальные арифметические операции включают в себя нормализацию, арифметический сдвиг (сдвигаются только цифровые разряды, знаковый разряд остаётся на месте), логический сдвиг (знаковый разряд сдвигается вместе с цифровыми разрядами). Обширна группа операций редактирования алфавитно–цифровой информации. Каждая операция в АЛУ является логической функцией или последовательностью логических функций описываемых двоичной логикой для двоичных ЭВМ, троичной логикой для троичных ЭВМ, четверичной логикой для четверичных ЭВМ, десятичной логикой для десятичных ЭВМ и так далее.

Классификация

По способу действия над операндами арифметико–логические устройства делятся на последовательные и параллельные. В последовательных устройствах операнды представляются в последовательном коде, а операции производятся последовательно во времени над их отдельными разрядами; в параллельных — параллельным кодом и операции совершаются параллельно во времени над всеми разрядами операндов.

По способу представления чисел различают арифметико–логические устройства:

– для чисел с фиксированной точкой;

– для чисел с плавающей точкой;

– для десятичных чисел.

По характеру использования элементов и узлов АЛУ делятся на блочные и многофункциональные. В блочном устройстве операции над числами с фиксированной и плавающей точкой, десятичными числами и алфавитно–цифровыми полями выполняются в отдельных блоках, при этом повышается скорость работы, так как блоки могут параллельно выполнять соответствующие операции, но значительно возрастают затраты оборудования. В многофункциональных АЛУ операции для всех форм представления чисел выполняются одними и теми же схемами, которые коммутируются нужным образом в зависимости от требуемого режима работы.

По своим функциям арифметико–логическое устройство является операционным блоком, выполняющим микрооперации, обеспечивающие приём из других устройств (например, памяти) операндов, их преобразование и выдачу результатов преобразования в другие устройства.

Логический элемент — элемент, осуществляющий определенные логические зависимость между входными и выходными сигналами. Логические элементы обычно используются для построения логических схем вычислительных машин, дискретных схем автоматического контроля и управления. Для всех видов логических элементов, независимо от их физической природы, характерны дискретные значения входных и выходных сигналов.

В зависимости от устройства схемы элемента, от ее электрических параметров, логические уровни (высокие и низкие уровни напряжения) входа и выхода имеют одинаковые значения для высокого и низкого (истинного и ложного) состояний.

Логическими элементами компьютеров являются электронные схемы И, ИЛИ, НЕ, И–НЕ, ИЛИ–НЕ и др. (называемые также вентилями), а также триггер.

Триггер – это устройство позволяющее запоминать, хранить и считывать информацию (каждый триггер может хранить 1 бит информации).

С помощью этих схем можно реализовать любую логическую функцию, описывающую работу устройств компьютера. Обычно у элементов бывает от 2 до 8 входов и один или два выхода. Чтобы представить два логических состояния 1 и 0, соответствующие им 3 входные и выходные сигналы имеют один из двух установленных уровней напряжения, например 5 и 0 В. Высокий уровень обычно со–ответствует значению «истинна» (1), низкий – значению «ложь» (0). Каждый логический элемент имеет свое условное обозначение, которое выражает его логическую функцию, но не указывает на то, какая электронная схема в нем реализована. Это упрощает запись и понимание сложных схем.

Работу логических элементов описывают с помощью таблиц истинности. Основные структурные схемы логических элементов компьютера и их таблицы истинности, представлены ниже.

Логический элемент «И» – конъюнкция, логическое умножение, AND

«И» – логический элемент, выполняющий над входными данными операцию конъюнкции или логического умножения. Данный элемент может иметь от 2 до 8 (наиболее распространены в производстве элементы «И» с 2, 3, 4 и 8 входами) входов и один выход.

Условные обозначения логических элементов «И» с разным количеством входов приведены на рисунке 2.2.3. В тексте логический элемент «И» с тем или иным числом входов обозначается как «2И», «4И» и т. д. – элемент «И» с двумя входами, с четырьмя входами.

На отечественных схемах — прямоугольник с символом «&» (рис 2.2.3).

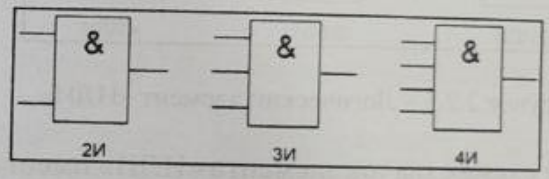
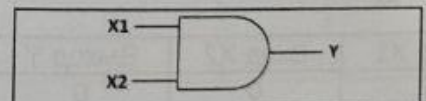


Рисунок 2.2.3 – Логический элемент «И»

На западных схемах значок элемента «И» имеет прямую черту на входе и закругление на выходе (рис 2.2.4).

Таблица истинности для элемента 2И показывает (рис. 2.2.4), что на выходе элемента будет логическая единица лишь в том случае, если логические единицы будутодновременно на первом входе И на втором входе. В остальных трех возможных случаях на выходе будет ноль.



Вход X1	Вход X2	Выход Y
0	0	0
1	0	0
0	1	0
1	1	1

Рисунок 2.2.4 – Таблица истинности «2И»

Логический элемент «ИЛИ» – дизъюнкция, логическое сложение, OR

«ИЛИ» – логический элемент, выполняющий над входными даннымиоперацию дизъюнкции или логического сложения. Он так же как и элемент «И» выпускается с двумя, тремя, четырьмя и т. д. входами и с одним выходом. Условные обозначения логических элементов «ИЛИ» с различным количеством входов показаны на рисунке 2.2.5.

На отечественных схемах — прямоугольник с символом «1». Обозначаются данные элементы так: 2ИЛИ, 3ИЛИ, 4ИЛИ и т. д.

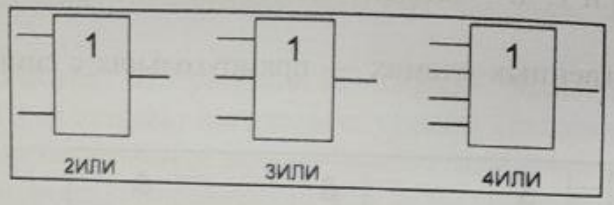
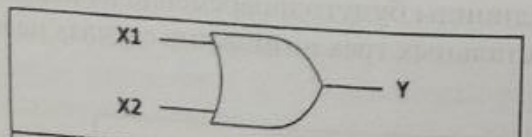


Рисунок 2.2.5 – Логический элемент «ИЛИ»

На западных схемах значок элемента «ИЛИ» имеет закругление на входе и закругление с заострением на выходе (рис. 2.2.6).

Таблица истинности (рис. 2.2.6) для элемента «2ИЛИ» показывает, что для появления на выходе логической единицы, достаточно чтобы логическая единица была на первом входе ИЛИ на втором входе. Если логические единицы будут сразу надвух входах, на выходе также будет единица.



Вход X1	Вход X2	Выход Y
0	0	0
1	0	1
0	1	1
1	1	1

Рисунок 2.2.6 – Таблица истинности «2ИЛИ»

Логический элемент «НЕ» – отрицание, инвертор, NOT

«НЕ» – логический элемент, выполняющий над входными данными операцию логического отрицания. Данный элемент, имеющий один выход и только одинвход, называют еще инвертором, поскольку он на самом деле инвертирует (обращает) входной сигнал. На рисунке 2.2.7 приведено условное обозначение логического элемента «НЕ». На отечественных схемах — прямоугольник с кружком на выходе (рис 2.2.7).

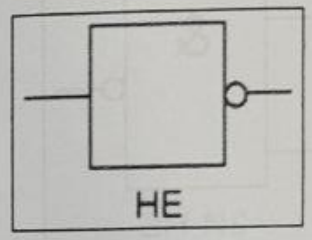
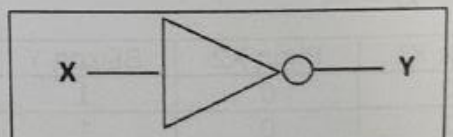


Рисунок 2.2.7 – Логический элемент «НЕ»

На западных схемах значок элемента «НЕ» имеет форму треугольника с кружочком на выходе (рис. 2.2.8).

Таблица истинности для инвертора показывает (рис. 2.2.8), что высокий потенциал на входе даёт низкий потенциал на выходе и наоборот.



Вход X	Выход Y
0	1
1	0

Рисунок 2.2.8 – Таблица истинности «НЕ»

Логический элемент «И–НЕ» – конъюнкция (логическое умножение) с отрицанием, NAND

«И–НЕ» – логический элемент, выполняющий над входными данными операцию логического сложения, и затем операцию

логического отрицания, результат подается на выход. Другими словами, это в принципе элемент «И», дополненный элементом «НЕ».

«И–НЕ» называют еще «элемент Шеффера» в честь математика Генри Мориса Шеффера, впервые отметившего значимость этой логической операции в 1913году.

На рисунке 2.2.9 приведено условное обозначение логического элемента «2И–НЕ».

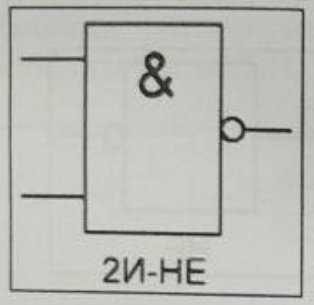
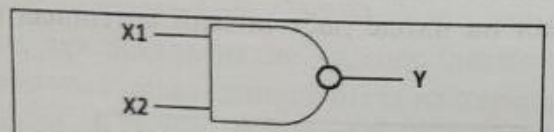


Рисунок 2.2.9 – Логический элемент «И–НЕ»

На западных схемах значок элемента «И-НЕ» имеет прямую черту на входе и закругление с кружочком на выходе (рис. 2.2.10).



Вход X1	Вход X2	Выход Y
0	0	1
1	0	1
0	1	1
1	1	0

Рисунок 2.2.10 – Таблица истинности «2И–НЕ»

Таблица истинности для элемента «И–НЕ» противоположна таблице истинности для элемента «И» (рис. 2.2.10). Вместо трех нулей и единицы — три единицы и ноль.

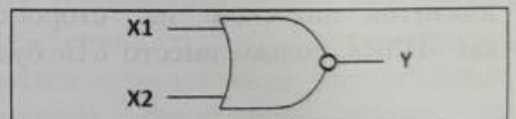
Логический элемент «ИЛИ–НЕ» – дизъюнкция (логическое сложение) с отрицанием, NOR

«ИЛИ–НЕ» – логический элемент, выполняющий над входными данными операцию логического сложения, и затем операцию логического отрицания, результат подается на выход. Иначе говоря, это элемент «ИЛИ», дополненный элементом «НЕ» – инвертором.

На рисунке 2.2.11 приведено условное обозначение логического элемента «2ИЛИ–НЕ».

Рисунок 2.2.11 – Логический элемент «ИЛИ–НЕ»

На западных схемах значок элемента «ИЛИ-НЕ» имеет закругление на входе и закругление с заострением и кружочком на выходе (рис. 2.2.12).



Вход X1	Вход X2	Выход Y
0	0	1
1	0	0
0	1	0
1	1	0

Рисунок 2.2.12 – Таблица истинности «2ИЛИ–НЕ»

Таблица истинности для элемента «ИЛИ–НЕ» противоположна таблице для элемента «ИЛИ» (рис. 2.2.12). Высокий потенциал на выходе получается лишь в одном случае – на оба входа подаются одновременно низкие потенциалы. Обозначается как «ИЛИ», только с кружочком на выходе, обозначающим инверсию.

Логический элемент «исключающее ИЛИ» – сложение по модулю 2, XOR

«Исключающее ИЛИ» – логический элемент, выполняющий над входными данными операцию логического сложения по модулю 2, имеет два входа и один выход. Часто данные элементы применяют в схемах контроля.

На рисунке 2.2.13 приведено условное обозначение данного элемента.

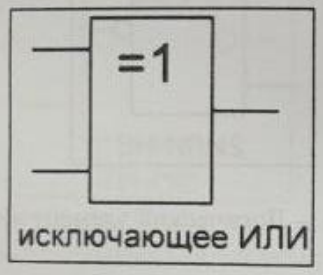
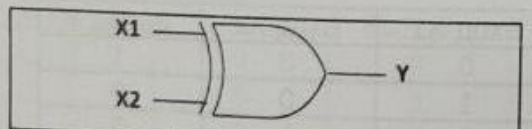


Рисунок 2.2.13 – Логический элемент «Исключающее ИЛИ»

Изображение в западных схемах — как у «ИЛИ» с дополнительной изогнутой полоской на стороне входа, в отечественной — как «ИЛИ», только вместо «1» будет написано «=1» (рис 2.2.14).



Вход X1	Вход X2	Выход Y
0	0	0
1	0	1
0	1	1
1	1	0

Рисунок 2.2.14 – Таблица истинности «Исключающее ИЛИ»

Этот логический элемент еще называют «неравнозначность». Таблица истинности для элемента «Исключающее ИЛИ» приведена

на рисунке 2.2.14. Высокий уровень напряжения будет на выходе лишь тогда, когда сигналы на входе не равны (на одном единица, на другом ноль или на одном ноль, а на другом единица) если даже на входе будут одновременно две единицы, на выходе будет ноль — в этом отличие от «ИЛИ». Данные элементы логики широко применяются в сумматорах.

На практике построение комбинационных схем усложняется, поскольку сигналы при прохождении через вентили ослабляются, искажают свою первоначальную форму, запаздывают. Поэтому необходимо наряду с логическими элементами включать в схему различного рода *согласующие элементы* (усилители, формирователи сигналов и др.). Задача этих элементов—сделать схему работоспособной и надежной.

Из сказанного ясно, что можно построить комбинационную схему (рис. 2.2.15) для решения любого конечного множества задач, решения которых однозначно определяются их условиями (подаваемыми на вход схемы). В частности, если ограничиться какой–либо фиксированной точностью представления вещественных чисел (разрядностью), то можно в принципе построить комбинационную схему, вычисляющую любую заданную вещественную функцию $y = f(x_i, \ldots, x_n)$ (в двоичных кодах).

На практике, однако, оказывается, что уже схема *умножителя* (вычисляющая функцию $y = X_1 \cdot X_2$) при разрядности (двоичной) 32 и более оказывается столь сложной, что умножение в современных ЭВМ предпочитают реализовать другим, так называемым алгоритмическим способом, о котором речь пойдет ниже.

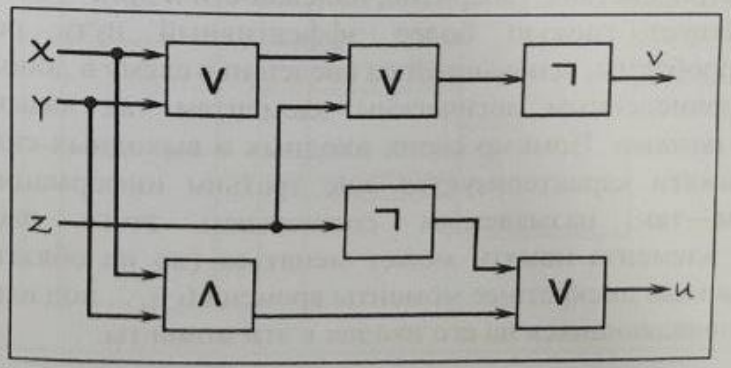


Рисунок 2.2.15 – Построение схемы из логических элементов

В то же время многие, более простые функции, например функции сложения двух чисел, реализуются комбинационными схемами приемлемой сложности. Соответствующая схема носит наименование *параллельного сумматора.*

Следует заметить, что успехи микроэлектроники делают возможным построение все более сложных схем. Если еще в 60–е годы каждый логический элемент собирался из нескольких физических элементов (транзисторов, диодов, сопротивлений и др.), то уже к началу 80–х годов промышленностью выпускаются так называемые *интегральные схемы,* содержащие многие сотни и даже тысячи логических вентилей. При этом важно подчеркнуть, что не только сами логические элементы, но и соединения между ними (т. е. вся схема в целом) изготовляются одновременно в едином технологическом процессе на тонких пластинках химически чистого кремния и других веществ размерами в доли квадратного сантиметра. Благодаря этому резко уменьшилась стоимость изготовления схем и повысилась их надежность.

Обладая возможностью реализовать любые фиксированные зависимости между входными и выходными сигналами» комбинационные схемы неспособны обучаться, адаптироваться к изменяющимся условиям. На первый взгляд кажется, что такая адаптация обязательно требует *структурных изменений* в схеме,. т. е. изменения связей между ее элементами, а возможно, и состава этих элементов. Подобные изменения нетрудно реализовать путем механических переключении. Однако такой путь практически неприемлем из–за резкого ухудшения практически всех параметров схемы (быстродействия, габаритов, надежности и др.).

Существует гораздо более эффективный путь решения указанной проблемы, основанный па введении в схему в дополнение к уже перечисленным логическим элементам так называемых *элементов памяти.* Помимо своих входных и выходных сигналов, элемент памяти характеризуется еще третьим информационным параметром—так называемым *состоянием* этого элемента. Состояние элемента памяти может меняться (но не обязательно) лишь в заданные дискретные моменты времени t_1, t_2, ... под влиянием сигналов, появляющихся на его входах в эти моменты.

Практическая часть

1. Создания компьютерной логической схемы:

Логические схемы нужны для того чтобы в наглядной графической форме отобразить последовательность выполнения операций при вычислении логических формул.

Таким образом компьютерные логические элементы строятся на основе логических уравнений.

В первую очередь, в схему необходимо включить инверторы (логический элемент «НЕ») для того что бы расположить по горизонтали все элементы, которые участвуют при построения компьютерной логической схемы (рис 2.2.16).

Пример к 1 пункту:

$$y = (x_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cup (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$$

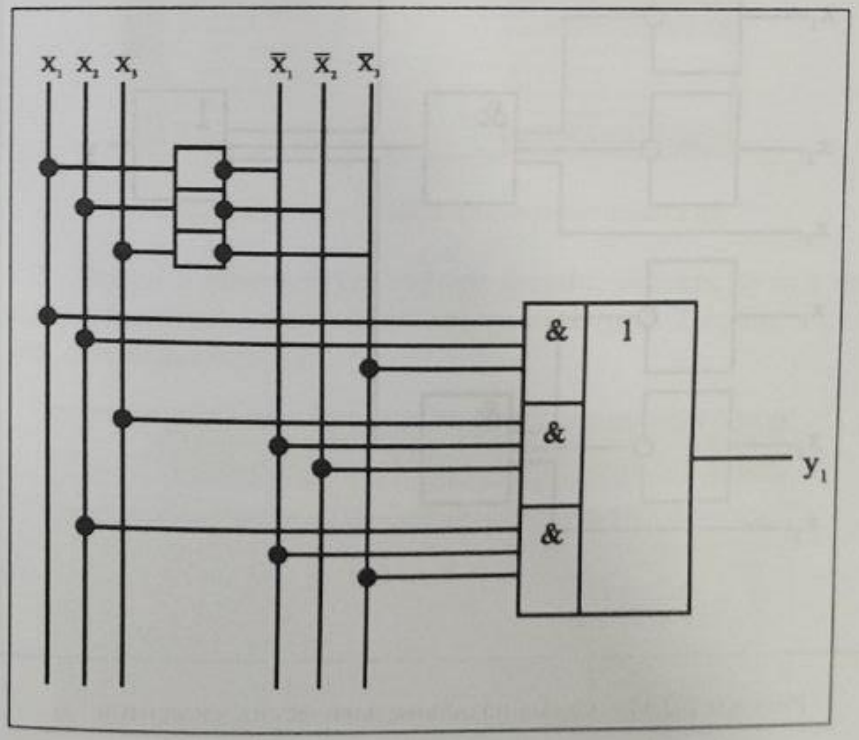


Рисунок 2.2.16 – Компьютерная логическая схема

По уравнению была построена схема «y»

2. Изображение схемы с помощью логических элементов

Пример к 2 пункту:

$y = (x_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cup (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$

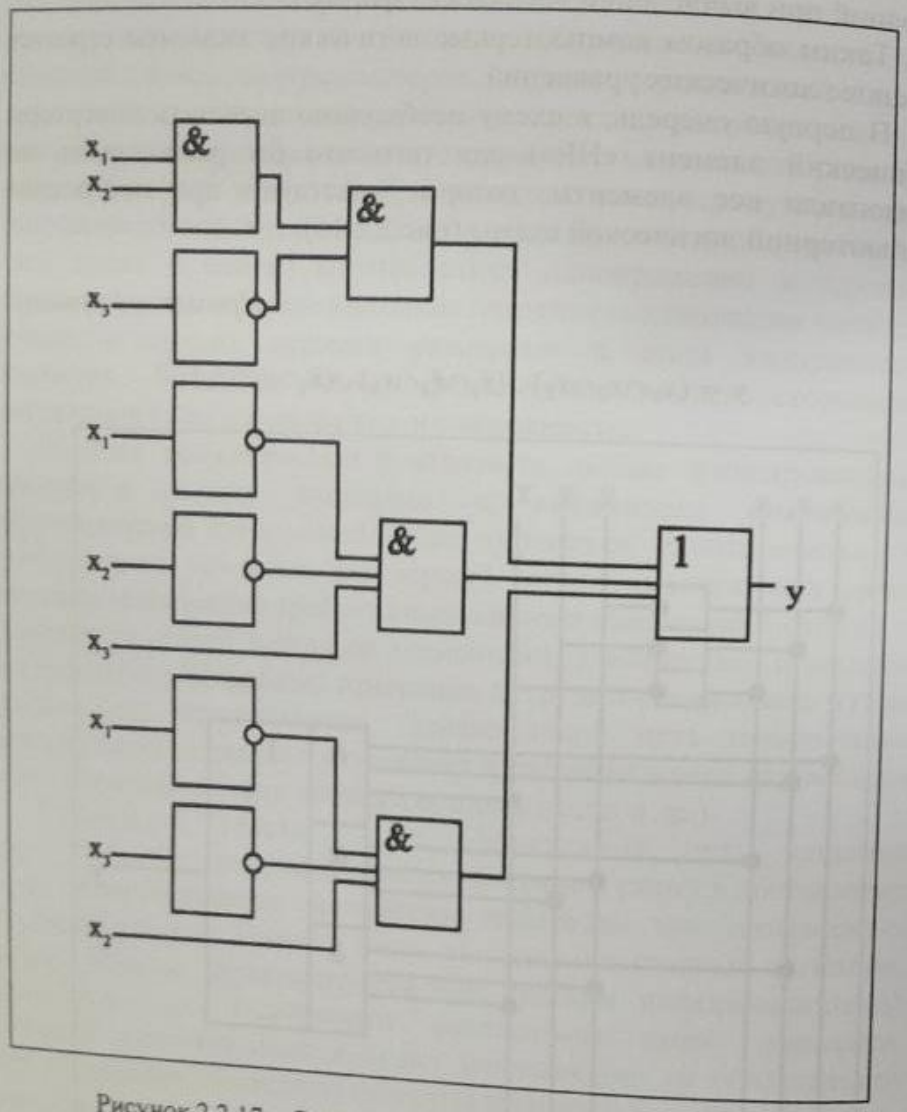


Рисунок 2.2.17 – Схема на основе логических элементов

3. Построение логической схемы на симуляторе

Запуск симулятора:

1. Скачать файл приложенный к заданию «SimulatorNew.jar»;
2. Для запуска данного симулятора необходима среда java, которую можно скачать с официального сайта или кликнув на слово «скачать» (версия должна быть 64/32);
3. Устанавливаем скаченный jre;
4. Для запуска симулятора, необходимо выполнить один из следующих пунктов:

4.1. Запустить симулятора с помощью установленного пакета jre кликнув правой кнопкой мыши и выбрав «Открыть с помощью» (рис 2.2.18)

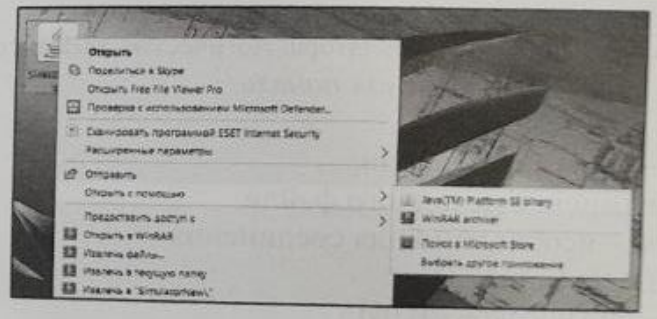


Рисунок 2.2.18 – Запуск с помощью пакета jre

4.2. Зайти в командную строку **«cmd»**, указать путь к файлу **«SimulatorNew.jar»** и прописать следующее (рис. 2.2.19):

java –jar ./SimulatorNew.jar

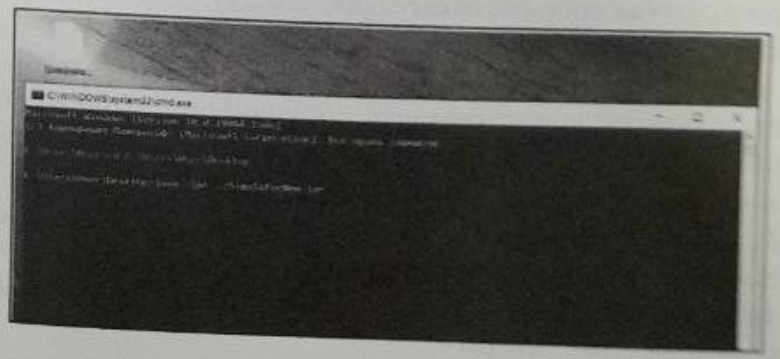


Рисунок 2.2.19 – Запуск симулятора через командную строку «cmd»

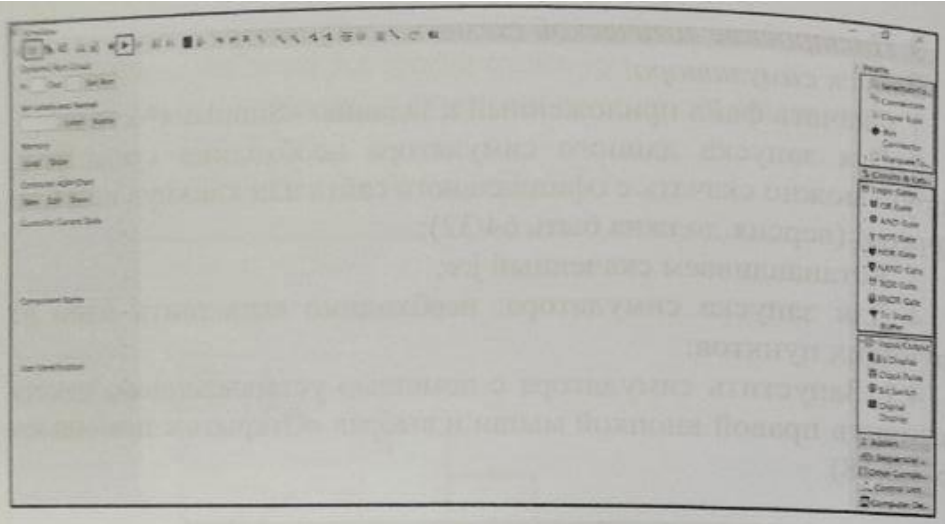


Рисунок 2.2.20 –Окно симулятора. Логические элементы

Основная панель:

Simulate – Запуск симуляции
Save – Сохранение исходного файла
Connection – необходимо для соединения
Панель «Logic Gates»:
OR Gate – Логическое «ИЛИ»
AND Gate – Логическое «И»
NOT Gate – Логическое «НЕ»
Панель «Input/output»:
Bit Display – Дисплей для вывода логического сигнала
Bit Switch – Сигнал который подается логическим устройствам

Для того, чтобы собрать схему (рис 2.2.21), необходимо, левой кнопкой мыши выбрать элемент, далее нажать на пустое место в рабочем окне. Соединение между элементами необходимо осуществить с помощью инструмента подключения (Connection).

Сигнал подается с элементов «Bit Switch», двойным нажатием левой кнопкой мыши можно переключить цифру, с «0» на «1» и обратно. В самый конец схемы необходимо подключить дисплей (Bit Display), для отображения полученных результатов.

Для запуска симуляции необходимо нажать кнопку «Simulate», на верхней панели симулятора.

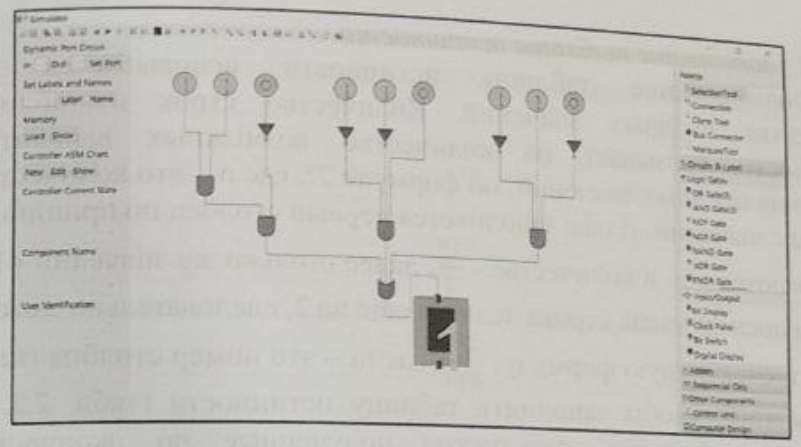


Рисунок 2.2.21 – Пример схемы (рис. 2.2.17), собранной в симуляторе, на основе логических элементов

Так как количество входных ножек на элементах симулятора равно 2, необходимо отдельно соединять вывод из 2 элементов.

Альтернатива программе, описанная выше представлена на рисунке 2.2.22, это онлайн ресурс по ссылке - https://logic.ly/demo/.

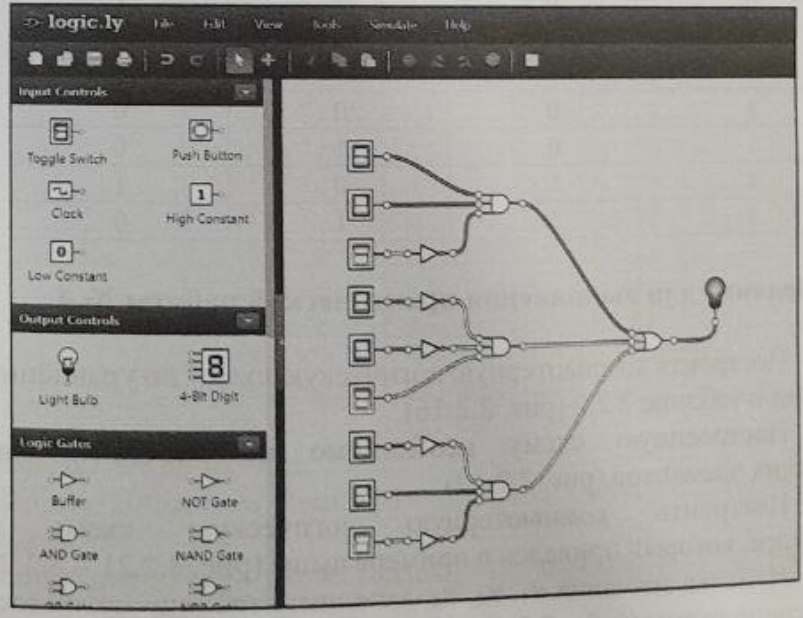


Рисунок 2.2.22 – Схема собранная в logic.ly

4. Заполнение таблицы истинности

Формирование таблицы истинности основывается на количестве входных значений. Количество строк в таблице истинности указывает на количество возможных вариантов состояния входных значений, по формуле 2^n, где n – это количество входных значений. Далее заполняется первый столбец по принципу: вписываются «0» в количестве - $\frac{2^n}{2}$, далее столько же значений «1», каждая последующая строка делится еще на 2, следовательно можно вывести следующую формулу: $\frac{2^n}{2 \cdot m}$, где m – это номер столбца ($m \geq 1$). Для того, чтобы заполнить таблицу истинности (табл. 2.2.1), необходимо записать результаты полученные по окончанию симуляции, изменяя значения x_1 x_2 x_3.

Таблица 2.2.1

Таблица истинности собранной схемы

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Задания для выполнения практической работы № 4

1. Построить компьютерную логическую схему по уравнениям заданным в таблице 2.2.2 (рис. 2.2.16).
2. Построенную схему необходимо изобразить в виде логических элементов (рис. 2.2.17).
3. Построить компьютерную логическую схему на симуляторе, который приведен в примере выше (рис. 2.2.21 – 2.2.22).
4. Изменяя значения x_1, x_2, x_3, заполнить таблицу истинности построенной схемы (табл. 2.2.1).

Таблица 2.2.2

Варианты для выполнения практической работы № 4

1	$y = (x_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$
2	$y = (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cup (x_1 \cap \bar{x}_2 \cap x_3) \cup (x_1 \cup x_2 \cup \bar{x}_3)$
3	$y = (x_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cup x_2 \cup x_3) \cup (x_1 \cap x_2 \cap \bar{x}_3)$
4	$y = (x_1 \cup x_2 \cup \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap x_3)$
5	$y = (\bar{x}_1 \cup x_2 \cup \bar{x}_3) \cap (\bar{x}_1 \cup x_2 \cup x_3) \cap (\bar{x}_1 \cup x_2 \cup \bar{x}_3)$
6	$y = (x_1 \cap x_2 \cap \bar{x}_3) \cup (x_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3)$
7	$y = (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cup (x_1 \cap x_2 \cap \bar{x}_3)$
8	$y = (x_1 \cup x_2 \cup \bar{x}_3) \cup (x_1 \cup x_2 \cup x_3) \cup (\bar{x}_1 \cup \bar{x}_2 \cup \bar{x}_3)$
9	$y = (\bar{x}_1 \cup \bar{x}_2 \cup \bar{x}_3) \cup (\bar{x}_1 \cup \bar{x}_2 \cup x_3) \cup (\bar{x}_1 \cup \bar{x}_2 \cup \bar{x}_3)$
10	$y = (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (x_1 \cap x_2 \cap x_3)$
11	$y = (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cap (x_1 \cap \bar{x}_2 \cap x_3) \cap (x_1 \cup \bar{x}_2 \cup \bar{x}_3)$
12	$y = (x_1 \cap x_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cup \bar{x}_2 \cup x_3) \cup (\bar{x}_1 \cup x_2 \cup \bar{x}_3)$
13	$y = (\bar{x}_1 \cup \bar{x}_2 \cup x_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap x_2 \cap x_3)$
14	$y = (x_1 \cup x_2 \cup x_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3)$
15	$y = (x_1 \cap x_2 \cap \bar{x}_3) \cap (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cap (\bar{x}_1 \cup x_2 \cup \bar{x}_3)$
16	$y = (x_1 \cup \bar{x}_2 \cup x_3) \cup (\bar{x}_1 \cup x_2 \cup x_3) \cup (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$
17	$y = (\bar{x}_1 \cap x_2 \cap x_3) \cup (x_1 \cap \bar{x}_2 \cap x_3) \cup (x_1 \cap x_2 \cap \bar{x}_3)$
18	$y = (x_1 \cup x_2 \cup \bar{x}_3) \cap (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cap (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$
19	$y = (\bar{x}_1 \cap \bar{x}_2 \cap x_3) \cup (\bar{x}_1 \cup x_2 \cup \bar{x}_3) \cup (\bar{x}_1 \cap x_2 \cap x_3)$
20	$y = (x_1 \cap x_2 \cap x_3) \cap (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3) \cap (\bar{x}_1 \cup \bar{x}_2 \cup \bar{x}_3)$
21	$y = (x_1 \cap x_2 \cap \bar{x}_3) \cap (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cap (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$
22	$y = (x_1 \cup \bar{x}_2 \cup x_3) \cap (\bar{x}_1 \cap x_2 \cap x_3) \cap (\bar{x}_1 \cup x_2 \cup \bar{x}_3)$
23	$y = (\bar{x}_1 \cup \bar{x}_2 \cup x_3) \cap (\bar{x}_1 \cap x_2 \cap x_3) \cap (x_1 \cap \bar{x}_2 \cap x_3)$
24	$y = (x_1 \cap \bar{x}_2 \cap \bar{x}_3) \cup (x_1 \cup x_2 \cup x_3) \cup (x_1 \cap x_2 \cap \bar{x}_3)$
25	$y = (\bar{x}_1 \cap x_2 \cap \bar{x}_3) \cap (x_1 \cap \bar{x}_2 \cap x_3) \cap (x_1 \cup \bar{x}_2 \cup \bar{x}_3)$
26	$y = (\bar{x}_1 \cap x_2 \cap x_3) \cap (x_1 \cap x_2 \cap \bar{x}_3) \cap (x_1 \cap x_2 \cap x_3)$
27	$y = (\bar{x}_1 \cup \bar{x}_2 \cup x_3) \cap (x_1 \cap x_2 \cap x_3) \cap (x_1 \cup x_2 \cup \bar{x}_3)$
28	$y = (x_1 \cup x_2 \cup \bar{x}_3) \cup (x_1 \cup \bar{x}_2 \cup x_3) \cup (\bar{x}_1 \cap x_2 \cap \bar{x}_3)$
29	$y = (\bar{x}_1 \cup x_2 \cup \bar{x}_3) \cap (\bar{x}_1 \cup x_2 \cup x_3) \cap (x_1 \cup \bar{x}_2 \cup \bar{x}_3)$
30	$y = (x_1 \cup \bar{x}_2 \cup \bar{x}_3) \cap (x_1 \cap \bar{x}_2 \cap x_3) \cap (\bar{x}_1 \cap \bar{x}_2 \cap \bar{x}_3)$

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 4.1 – …»

Контрольные вопросы

1. Каковы различия в уровнях представления вычислительных устройств?
2. Что такое регистры хранения и сдвига?
3. Какие существуют счетчики?
4. Что такое АЛУ?
5. Опишите работу элемента логическое «ИЛИ»
6. Опишите работу элемента логическое «И»
7. Опишите работу элемента логическое «ИЛИ – НЕ»
8. Опишите работу элемента логическое «И – НЕ»

ГЛАВА III
ВВЕДЕНИЕ В АССЕМБЛЕР, ИЗУЧЕНИЕ ОСНОВНЫХ ОПЕРАТОРОВ И ВЫПОЛНЕНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ

3.1. Архитектура системы команд. Выполнений простейших арифметических операций на ассемблере

Цель работы: Изучение операторов языка Ассемблера для микропроцессора КР580.

Теоретическая часть

Микропроцессор – центральное устройство (или комплекс устройств) ЭВМ (или вычислительной системы), которое выполняет арифметические и логические операции, заданные программой преобразования информации, управляет вычислительным процессом и координирует работу устройств системы (запоминающих, сортировальных, ввода — вывода, подготовки данных и др.).

Для облегчения изучения принципов программирования МП–систем на низком уровне на языке ассемблера, а также приобретения базовых понятий в области организации ЭВМ и МПС, была разработана программная модель–эмулятор МПС, построенная на базе особенностей рассматриваемого микропроцессора.

Данный эмулятор позволяет: написание программ на языке ассемблера, используя систему команд МП КР580ВМ80А, их отладку и выполнение в тактовом, командном и сквозном режимах; изучить принципы и порядок выполнения команд; приобрести навыки работы с внешними устройствами МП–системы; получить представления об организации внешней и внутренней (регистровой) памяти и стековой области.

Программа обладает приятным интерфейсом, удобна в использовании и имеет функции сохранения, экспорта и печати данных. Всё это позволяет легко и удобно освоиться в ней, а также, получить все необходимые вышерассмотренные навыки.

Также, эмулятор может быть полезен и для опытных людей, к

примеру, в качестве визуализированного помощника в программировании разрабатываемой ими МП–системы на базе КР580ВМ80 (i8080).

В возможности эмулятора входит: работа с 5–ю внешними устройствами, такими, как монитор, НГМД, НЖМД, сетевой адаптер и принтер; отладка и выполнение программ в тактовом, командном и сквозном режимах; работа со всем спектром системы команд данного МП; сохранение, загрузка и печать данных и результатов; ручной ввод данных в ОЗУ и РОН. Также, в состав дистрибутива включено подробное руководство пользователя, описание системы команд и файлы–образы ОЗУ эмулятора для примера.

Простейшую 8–разрядную микропроцессорную систему (МПС) можно построить на основе микропроцессорного комплекта (МПК) серии КР580.

МПК КР580 выполнен по n–МДП и ТТЛШ– технологиям. Он характеризуется архитектурным единством, обеспечиваемым автономностью и функциональной законченностью отдельных интегральных микросхем (ИМС), унификацией их интерфейса, программируемостью, их логической и электрической совместимостью. Структурная схема МП КР580ВМ80А приведена на рисунке 3.1.1.

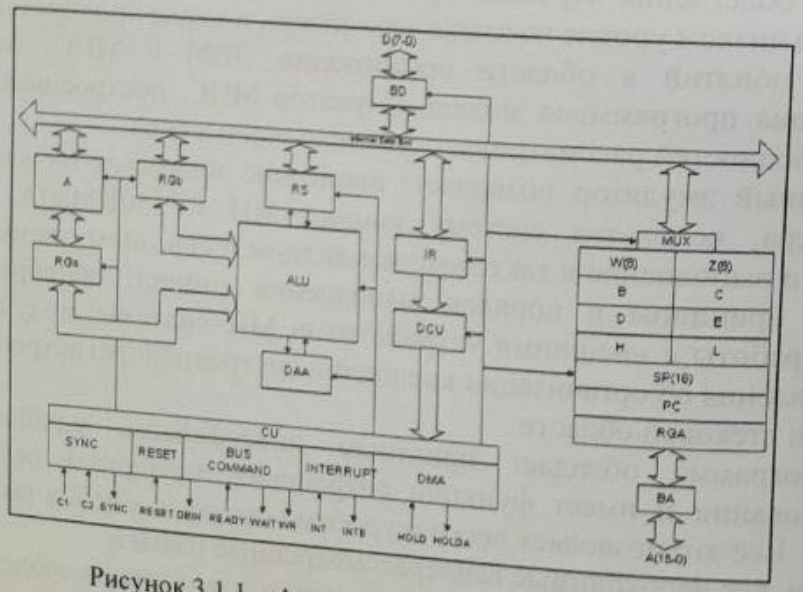


Рисунок 3.1.1 – Архитектура МП КР580ВМ80А

В состав микропроцессора входят:

– 8–разрядное арифметико–логическое устройство АЛУ (ALU);

– регистр признаков RS, фиксирующий признаки, вырабатываемые АЛУ в процессе выполнения команды;

– аккумулятор (A);

– регистр аккумулятора (RGa);

– регистр временного хранения операндов (RGb);

– десятичный корректор (DAA), выполняющий перевод информации из двоичной в двоично–десятичную форму;

– регистр команд (IR), предназначенный для хранения первого байта команды, содержащего код операции;

– дешифратор команд (DCU);

– блок регистров для приема, выдачи и временного хранения информации в процессе выполнения программ;

– схема управления и синхронизации (CU), формирующая последовательности управляющих сигналов для работы ALU и регистров;

– однонаправленный 16–разрядный буферный регистр адреса (BA);

– двунаправленный 8–разрядный буферный регистр данных (BD);

– двунаправленный мультиплексор (MUX) для обмена информацией между ALU и блоком регистров по внутренней шине данных (Internal Data Bus.

Блок регистров включает:

– программный счетчик (PC), предназначенный для хранения адреса очередной команды (при выполнении линейных программ этот адрес автоматически увеличивается на 1, 2, 3 в зависимости от длины выполняемой команды – 1, 2 или 3 байта соответственно);

– указатель стека (SP);

– регистр адреса (RGA);

– шесть 8–разрядных регистров общего назначения B, C, D, E, H, L, которые могут объединяться в парные 16–разрядные регистры BC, DE, HL;

– вспомогательные разрядные регистры W, Z.

Регистры RGa, RGb, IR, W, Z, RGA пользователю программно недоступны.

Кроме того, МП имеет 16–разрядный однонаправленный 3–стабильный канал адреса A(15–0), 8–разрядный двунаправленный 3–стабильный канал данных D(7–0), четыре входных (RESET, READY, INT, HOLD) и шесть выходных (SYNC, DBIN, READY, WAIT, INTE, HLDA) выводов сигналов управления.

Десятичный корректор DAA облегчает работу с числами, представленными в 10–чной системе счисления.

Буферные регистры данных BD и адреса BA используются для буферизации внутренних шин данных и адреса со стороны внешней магистрали.

Схема управления и синхронизации выполняет ряд функций управления и синхронизации:

- обеспечивает выборку команд и операндов;
- организует правильное функционирование АЛУ;
- обеспечивает доступ ко всем регистрам МП;
- синхронизирует УВВ и управляет их работой;
- приостанавливает работу МП в режиме ожидания и отключает МП от системной магистрали в режиме ПДП.

Мультиплексор MUX обеспечивает подключение к внутренней магистрали МП требуемого регистра из блока регистров.

МП КР580ВМ80А обеспечивает адресацию внешней памяти до 64 Кбайт и подключение до 256 устройств ввода–вывода.

Система команд МП КР580ВМ80А. В МП ВМ80А применяется довольно простой формат команд (рис. 3.1.2).

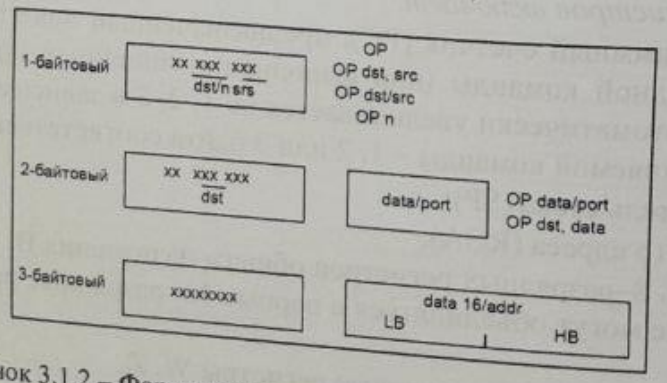


Рисунок 3.1.2 – Формат команд микропроцессора КР580ВМ80А

Здесь:

- OP (Operation) –код операции;
- dst (Destination) – адресат–приемник;
- src (Source) – адресат–источник;
- n = 0..7;
- data16 – 16–разрядные данные;
- addr (Address) – адрес;
- LB (Low Byte) – младший байт;
- HB (High Byte) – старший байт.

Всего в систему команд ВМ80А входят 78 базовых команд, содержащих 111 кодовых операций. В зависимости от своего назначения команда может иметь длину в один, два или три байта и соответственно занимает в памяти от одной до трех последовательных ячеек. Программный счетчик PC микропроцессора всегда содержит адрес первого байта команды, которая будет выполняться вслед за командой, которая выполняется в текущий момент времени.

Код операции всегда размещен в первом байте команды. В формате команд биты, задающие код операции, отмечены X. Эти биты определяют смысл данной команды, ее содержание/назначение. В первом же байте команды может находиться и информация о местонахождении операндов: dst, src или целое число 0..7.

Второй и, если необходимо, третий байты команды отводятся под непосредственные данные, адрес порта или ячейки памяти.

Примеры команд:

1. Однобайтовые команды:

MOV A, B LDAX B RST 7

2. Двухбайтовые команды:

MVI M, 85h SUI 8Eh IN 21h OUT 3Ah

3. Трехбайтовые команды:

LDA 1234h LXI B, 45AEeh CALL A34Ch JC B800h

В командах допускается явное задание только одного адреса памяти. По этой причине систему команд МП следует отнести к классу одноадресных.

Система команд МП состоит из 5 групп:

– команды пересылки (14 команд, 28 операций);
– логические команды (15 команд, 19 операций);
– арифметические команды (14 команд, 29 операций);
– команды передачи управления (28 команд, 28 операций);
– команды управления процессором (7 команд, 7 операций).

Система команд МП ВМ80А полностью входит в систему команд МП К1821ВМ85А (ВМ85А), обеспечивая полную совместимость этих микропроцессоров на уровне объектного кода. Они отличаются друг от друга лишь числом периодов тактовой частоты, необходимых для исполнения той или иной команды. Кроме того, МП ВМ85А содержит 2 дополнительные команды, не поддерживаемые МП ВМ80А.

Группа № 1. Команды пересылки.

Наряду с мнемоникой и кодом операции (первый байт команды) таблица содержит такие важнейшие для команды характеристики, как число обращений к системной магистрали и число периодов тактовой частоты, составляющих ее полный командный цикл. Здесь же представляется информация о влиянии команды на флаги регистра признаков F (при знаке "+" команда воздействует на соответствующий флаг).

В условных командах, рассматриваемых ниже, число обращений к шине и длительность МЦ в тактах зависят от выполнения условия. В соответствующих таблицах будут указаны два значения – невыполнения и выполнения условия.

Поля src и dst обозначают один из 8–разрядных регистров A, B, C, D, H, L, закодированных в соответствии со следующей таблицей:

Регистр	B	C	D	E	H	L	A
Код	000	001	010	011	100	101	111

Рассмотренная группа команд содержит команды обмена между памятью и регистрами. Это наиболее часто встречающиеся в

программах команды, занимающие около 45% процентов их общего числа. Группа команд пересылки не оказывает влияния на флаги, за исключением POP PSW.

Основу группы составляют следующие команды:

MOV, MVI	Перемещение
LDA, LDAX, LXI, LHLD	Загрузка
STA, STAX, SHLD	Сохранение

Эти команды оперируют как байтами (MOV, MVI, LDA(X), STA(X)), так и словами (LXI, LHLD, SHLD).

В командах пересылки поля src и dst используются для указания 8–разрядных регистров A…L, а M – обозначает косвенную адресацию через регистровую пару HL, которая должна содержать прямой адрес байта, участвующего в обмене (таб. 3.1.1).

В составе команд имеются также операции загрузки аккумулятора с прямой (LDA) и косвенной (LDAX) адресацией через регистровые пары BC и DE, а также их обратные эквиваленты STA, STAX. С учетом значимости 16–разрядного регистра HL предусмотрены операции загрузки LHLD и хранение SHLD его содержимого по прямому адресу.

Команды MVI и LXI используют непосредственную адресацию, обеспечивающую загрузку 8– и 16– разрядного регистра константой.

Для начальной установки SP предусмотрены две команды:

LXI SP, data 16; SP ← data16
SPHL; SP ← HL

Первая команда обеспечивает загрузку указателя стека SP константой. С помощью второй команды можно организовать установку SP в соответствии со значением некоторой переменной. Такая операция удобна при реализации нескольких стеков.

Благодаря командам PUSH и POP создается удобный механизм сохранения и восстановления текущего контекста регистровой области или ее части для передачи параметров через стек и др. Системный стек растет в сторону уменьшения адресов, а его указатель всегда адресует последний элемент стека или его вершину TOS (Top Of Stack).

Таблица 3.1.1
Команды пересылки

Мнемоника	Код	Число циклов BM80A	Число тактов BM80A	Флаги: CY, Z, M, P, C, AC	Содержание
MOV dst, srs	01DDDSSS	1	5	------	dst ← srs
MOV dst, M	01DDD110	2	7	------	dst ← (HL)
MOV M, srs	01110SSS	2	7	------	(HL) ← srs
MVI dst, data	00DDD110	2	7	------	dst ← data
MVI M, data	36	3	10	------	(HL) ← data
LDA addr	3A	4	13	------	A ← (addr)
STA addr	32	4	13	------	(addr) ← A
LDAX B	0A	2	7	------	A ← (BC)
LDAX D	1A	2	7	------	A ← (DE)
STAX B	02	2	7	------	(BC) ← A
STAX D	12	2	7	------	(DE) ← A
LXI B, data16	01	3	10	------	BC ← data16
LXI D, data16	11	3	10	------	DE ← data16
LXI H, data16	21	3	10	------	HL ← data16
LXI SP, data16	31	3	10	------	SP ← data16
LHLD addr	2A	5	16	------	HL ← (addr)
SHLD addr	22	5	16	------	(addr) ← HL
SPHL	F9	1	5	------	SP ← HL
PUSH B/D/H	C5/D5/E5	3	11	------	(SP) ← BC/DE/HL
PUSH PSW	F5	3	11	------	(SP) ← PSW
POP B/D/H	C1/D1/E1	3	10	------	BC/DE/HL ← (SP)+
POP PSW	F1	3	10	++++++	PSW ← (SP)+
XCHG	EB	1	4	------	DE ↔ HL
XTHL	E3	5	18	------	(SP) ↔ HL

Команда XCHG позволяет использовать содержимое регистровой пары DE практически с тем же результатом, что и содержимое HL:

XCHG; DE ⟷ HL.

XTHL; (SP) ⟷ HL

поддерживает прямой доступ к TOS. В тех ситуациях, когда TOS содержит адрес возврата, эта команда обеспечивает доступ к нему с целью модификации. Если же применяется механизм передачи в подпрограмму параметров непосредственно за командой вызова CALL, команда XTHL обеспечивает прямую настройку на область параметров, их выборку и модификацию адреса возврата в соответствии с длиной области. Важно то, что все эти операции производятся без потери содержимого HL.

Команда XTHL используется также для получения текущего состояния PC, например, при организации относительной адресации:

CALL M1; – (SP) ← M1, PC ← M1

.....

M1:XTHL; HL ⟷ SP

Здесь HL получает значение метки M1, которое может быть использовано произвольно.

Описание функционала команд

Команды пересылки данных:

MOV R1,R2 – пересылка из регистра в регистр;

LDAX B(D) – пересылка из ячейки памяти, адрес которой записан в регистровой паре BC (DE), в аккумулятор;

STAX B(D) – пересылка из аккумулятора в ячейку памяти, адрес которой записан в регистровой паре BC (DE);

LDA_B2_B3 – пересылка из ячейки памяти, адрес которой записан во втором и третьем байтах команды, в аккумулятор;

STA_B2_B3 – пересылка из аккумулятора в ячейку памяти, адрес которой указан во втором и третьем байтах команды;

SPHL – пересылка данных из регистровой пары HL в указатель стека;

PCHL – пересылка данных из регистровой пары HL в счетчик команд;

LHLD_B2_B3 – пересылка данных из ячеек памяти с адресами, записанным во втором и третьем байтах команды и на единицу большем, в регистровую пару HL;

SHLD_B2_B3 – пересылка данных из регистровой пары HL в ячейки памяти с адресами, записанными во втором и третьем байтах команды и на единицу большем;

MVI R_B2 – загрузка второго байта команды в регистр R;

LXI B (D,H)_B2_B3 – загрузка второго и третьего байтов команды в регистровую пару BC (DE,HL).

Команды обмена данными:

XCHG – обмен данными между парами регистров HL и DE;

XTHL – обмен данными между парой регистров HL и вершиной стека (L) ⟷ [УС], (H) ⟷ [УС+1].

Команды операций со стеком:

PUSH B(D,H) – запись содержимого регистровой пары BC (DE,HL) в стек

(B,D,H) → [УС–1], (C,E,L) → [УС–2], (УС)= (УС)–2;

PUSH PSW – запись слова состояния в стек я

(A) → [УС–1], (F) → [УС–2], (УС)=(УС)–2;

POP B(D,H) – запись из стека в регистровую пару BC(DE,HL)

[УС] → (C,E,L), [УС+1] → (B,D,H), (УС)=(УС)+2;

POP PSW – запись слова состояния из стека

[УС] → (F), [УС+1] → (A), (УС)=(УС)+2.

Группа № 2. Логические команды

В составе группы 4 двухместных логических операций над байтами (таб. 3.1.2):

ANA, ANI Логическое И

XRA, XRI Логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

ORA, ORI Логическое ИЛИ

CMP, CPI Сравнение

В этих командах один из операндов содержится в аккумуляторе A, который одновременно служит приемником результата. Источником второго операнда служат:

1) регистр src;

2) ячейка памяти (HL);

3) константа, заданная непосредственно в команде.

Логические операции выполняются поразрядно. Операция сравнения выполняется методом вычитания, но результат вычитания никуда не заносится, а операнды сохраняются без изменения.

Две команды STC и CMC дают возможность манипулировать флагом CY, устанавливая и инвертируя его.

Сброс флага CY может быть выполнен командой (При этом состояние флагов тоже изменяется):

ORA A; A ← A | A

Таблица 3.1.2

Логические команды

Мнемоника	Код	Число циклов BM80A	Число тактов BM80A	Флаги: CY, Z, M, P, C, AC	Содержание
ANA src	10100SSS	1	4	0 + + + 0	A ← A & src
XRA src	10101SSS	1	4	0 + + + 0	A ← A ^ src
ORA src	10110SSS	1	4	0 + + + 0	A ← A | src
CMP src	10111SSS	1	4	+ + + + +	A = src
ANA M	A6	2	7	0 + + + 0	A ← A & (HL)
XRA M	AE	2	7	0 + + + 0	A ← A ^ (HL)
ORA M	B6	2	7	0 + + + 0	A ← A | (HL)
CMP M	BE	2	7	+ + + + +	A = (HL)
ANI data	E6	2	7	0 + + + 0	A ← A & data
XRI data	EE	2	7	0 + + + 0	A ← A ^ data
ORI data	F6	2	7	0 + + + 0	A ← A | data
CPI data	FE	2	7	+ + + + +	A = data
RLC	07	1	4	+ – – – –	A7←A6←... ←A0←A7
RRC	0F	1	4	+ – – – –	A0←A1←... ←A7←A0
RAL	17	1	4	+ – – – –	A7←A6←...← A0←CY←A7
RAR	1F	1	4	+ – – – –	A0←A1←...← A7←CY←A0
CMA	2E	1	4	– – – – –	A ← !A
CMS	3E	1	4	+ – – – –	CY ← !CY
STC	37	1	4	1 – – – –	CY ← 1

Логические команды включают также и подгруппу сдвигов вправо (RRC, RAR) и влево (RLC, RAL). Определены операции циклического (RRC, RLC) и расширенного (RAR, RAL) сдвигов.

Описание функционала команд

Логические команды:

ANA R – операция "И" между содержимым аккумулятора и содержимым регистра $(A) \leftarrow (A) \wedge (R)$;

ANI_B2 – операция "И" между содержимым аккумулятора и вторым байтом команды $(A) \leftarrow (A) \wedge B2$;

ORA R – операция "ИЛИ" между содержимым аккумулятора и содержимым регистра R $(A) \leftarrow (A) \vee (R)$;

ORI_B2 – операция "ИЛИ" между содержимым аккумулятора и вторым байтом команды $(A) \leftarrow (A) \vee B2$;

XRA R – операция "ИЛИ–НЕ" между содержимым аккумулятора и содержимым регистра R $(A) \leftarrow (A) \oplus (R)$;

XRI_B2 – операция "ИЛИ–НЕ" между содержимым аккумулятора и вторым байтом команды $(A) \leftarrow (A) \oplus B2$.

Команды сравнения:

CMP R – операция (A) – (R); если (A) = (R), то Z = 1; если (A)<(R), то C=1;

CPI_B2 – операция (A) – B2; если (A) = B2, то Z = 1; если (A)< B2, то C=1.

Команды сдвига:

RLC – сдвиг влево (каждый бит сдвигается на один разряд влево, а 7 бит переносится в 0 и одновременно записывается в признак (C)) Dm→Dm+1, D7→D0, D7→C;

RRC – сдвиг вправо (каждый бит сдвигается на один разряд вправо, а 0 бит переносится в 7 и одновременно записывается в признак (C)) Dm+1→Dm, D0→ D7,D0→C;

RAL – сдвиг влево через перенос (каждый бит сдвигается на один разряд влево, 7 бит записывается в признак (C), а бит из (C) записывается в 0 бит) Dm→Dm+1,D7→C, C→D0;

RAR – сдвиг вправо через перенос (каждый бит сдвигается на один разряд вправо, 0 бит записывается в признак (C), а бит из (C) записывается в 7 бит) Dm+1→Dm,D0→C, C→D7.

CMA – Команда инверсии содержимого аккумулятора $(A) \leftarrow (\bar{A})$

Группа № 3. Команды арифметической обработки

ADD, ADC, ADI, ACI, DAD	–	Сложение
SUB, SBB, SUI, SBI	–	Вычитание
INR, INX	–	Инкремент на 1
DCR, DCX	–	Декремент на 1
DAA	–	Десятичная коррекция

Предусмотрены операции как над байтами, так и над словами.

Во всех байтовых операциях сложения и вычитания аккумулятор используется как источник операнда и приемник результата. В качестве источника второго операнда применяется либо регистр src, либо ячейка памяти M, либо литерал data. В команде сложения DAD роль аккумулятора выполняет регистровая пара HL. Эта команда очень важна при организации таблиц и списков.

С помощью команд инкремента/декремента довольно просто реализовать счетчики, часто необходимые в практике программирования (таб. 3.1.3).

Для хранения десятичных чисел в памяти обычно используется двоично–десятичный код (2/10 код BCD – Binary Code Decimal). Различают два формата 2/10 – кода: упакованный и распакованный. В 2/10 – коде упакованного формата каждая десятичная цифра от 1 до 9 представляется 4–разрядным двоичным эквивалентом, например:

9272=1001 0010 0111 0010 BCD

В 2/10 – коде распакованного формата каждая десятичная цифра занимает 1 байт:

9272=00001001 00000010 00000111 00000010 BCD

Распакованный формат, в сравнении с упакованным форматом, требует в два раза больше памяти, однако он очень хорошо согласуется с текстовым представлением десятичных цифр в коде

КОИ–7. (Для перевода распакованного 2/10 – кода в текстовый формат достаточно в старшую тетраду каждой цифры записать код 0011В).

Таблица 3.1.3

Команды арифметической обработки

Мнемоника	Код	Число циклов КМ80А	Число тактов	Флаги: CY, Z, M, P, C, AC	Содержание
ADD src	10000SSS	1	4	+ + + + +	A ← A + src
ADC src	10001SSS	1	4	+ + + + +	A ← A + src + CY
SUB src	10010SSS	1	4	+ + + + +	A ← A – src
SBB src	10011SSS	1	4	+ + + + +	A ← A – src – CY
ADD M	86	2	7	+ + + + +	A ← A + (HL)
ADC M	8E	2	7	+ + + + +	A ← A + (HL) + CY
SUB M	96	2	7	+ + + + +	A ← A – (HL)
SBB M	9E	2	7	+ + + + +	A ← A – (HL) – CY
ADI data	C6	2	7	+ + + + +	A ← A + data
ACI data	CE	2	7	+ + + + +	A ← A + data + CY
SUI data	D6	2	7	+ + + + +	A ← A – data
SBI data	DE	2	7	+ + + + +	A ← A – data – CY
INR dst	00DDD100	1	5	– + + + +	dst ← dst + 1
DCR dst	00DDD101	1	5	– + + + +	dst ← dst – 1
INR M	34	3	10	– + + + +	(HL) ← (HL) + 1
DCR M	35	3	10	– + + + +	(HL) ← (HL) – 1
DAA	27	1	4	+ + + + +	A ← 2/10 корр–я A
DAD B/D/H/SP	09/19/29/39	3	10	+ – – – –	HL ← HL+BC/DE/HL/SP
INX B/D/H/SP	03/13/23/33	1	5	– – – – –	R16 ← R16 + 1
DCX B/D/H/SP	0B/1B/2B/3B	1	5	– – – – –	R16 ← R16 – 1

Арифметические операции над 10–чными числами выполняются в 2 этапа: сначала производится обычная двоичная операция над 10–чными числами, затем десятичная коррекция результата.

Описание функционала команд

Команды сложения:

ADD R – сложение содержимого аккумулятора с содержимым регистра R (A) ← (A) + (R);

ADI_B2 – сложение содержимого аккумулятора со вторым байтом команды (A) ← (A) + B2;

ADC R – сложение содержимого аккумулятора с содержимым регистра R и признаком (C) (A) ← (A) + (R) + (C);

ACI_B2 – сложение содержимого аккумулятора со вторым байтом команды и признаком (C) (A) ← (A) + B2 +(C);

DAD B (D, H, SP) – сложение содержимого пары регистров HL с содержимым пары регистров BC (DE или HL или указателем стека) и запись результата в пару HL.

Команды вычитания :

SUB R – вычитание из содержимого аккумулятора содержимого регистра R (A) ← (A) – (R);

SUI_B2 – вычитание из содержимого аккумулятора второго байта команды (A) ← (A) – B2;

SBB R – вычитание из содержимого аккумулятора содержимого регистра R и признака (C) (A) ← (A) – (R) – (C);

SBI_B2 – вычитание из содержимого аккумулятора второго байта команды и признака (C) (A) ← (A) – B2 – (C).

Операции инкремента и декремента:

INR R – увеличение содержимого регистра на 1: (R) + 1 → (R);

INX B (D, H, SP) – увеличение содержимого пары регистров BC (DE), (HL), (SP) на 1 ;

DCR R – уменьшение содержимого регистра R на 1: (R) – 1 → (R);

DCX B (D, H, SP) – уменьшение содержимого пары регистров BC (DE), (HL), (SP) на 1 .

Группа № 4. Команды передачи управления

Эти команды содержат три основные операции, типичные для большинства МП, и организуют безусловный переход (таб. 3.1.4):

JMP	Переход
CALL	Вызов подпрограммы
RET	Возврат из подпрограммы

Таблица 3.1.4

Команды передачи управления

Мнемоника	Код	Число циклов	Число тактов	Флаги: CY, Z, M, P, C, AC	Содержание
PCHL	E9	1	5	-----	PC ← HL
JMP addr	C3	3	10	-----	PC ← addr
JC addr	DA	3	10	-----	if (CY) PC ← addr
JNC addr	D2	3	10	-----	if (!CY) PC ← addr
JZ/JNZ addr	CA/C2	3	10	-----	if (Z / !Z) PC ← addr
JM/JP addr	FA/F2	3	10	-----	if (M / !M) PC ← addr
JPE/JPO addr	EA/E2	3	10	-----	if (P / !P) PC ← addr
CALL addr	CD	3	10	-----	-(SP) ← PC ← addr
CC/CNC addr	DC/D4	3/5	11/17	-----	if (CY / !CY) CALL addr
CZ/CNZ addr	CC/C4	3/5	11/17	-----	if (Z / !Z) CALL addr
CM/CP addr	FC/F4	3/5	11/17	-----	if (M / !M) CALL addr
CPE/CPO addr	EC/E4	3/5	11/17	-----	if (P / !P) CALL addr
RET	C9	3	11	-----	PC ← (SP)+
RC/RNC	D8/D0	1/3	5/11	-----	if (CY / !CY) RET
RZ/RNZ	C8/C0	1/3	5/11	-----	if (Z / !Z) RET
RM/RP	F8/F0	1/3	5/11	-----	if (M / !M) RET
RPE/RPO	E8/E0	1/3	5/11	-----	if (P / !P) RET

Для поддержки условной передачи управления на их базе построены три соответствующие модификации базовых операций:

Jcc Условный переход
Ccc Условный вызов ПП
Rcc Условный возврат из ПП

Каждая операция обеспечивает проверку 8 условий, в соответствии с результатами которой меняются значения поля СС:

C	Carry	CY=1	M	Minus	M=1
NC	Not Carry	CY=0	P	Positive	M=0
Z	Zero	Z=1	PE	Parity Even	P=1
NZ	Not Zero	Z=0	PO	Parity Odd	P=0

Передача управления осуществляется в любую точку 64 килобайтовой области пространства памяти.

Наличие команды PCHL решает важную проблему передачи управления по вычисляемому адресу.

Описание функционала команд

Команды безусловного перехода:

JMP_B2_B3 – записать информацию из второго и третьего байта команды в счетчик команд;

PCHL – записать содержимое пары регистров HL в счетчик команд.

Команды условного перехода:

JNC – если признак c = 0
JC – если признак c = 1
JNZ – если признак z = 0
JZ – если признак z = 1
JPO – если признак p = 0
JPE – если признак p = 1
JP – если признак s = 0
JM – если признак s = 1

то перейти по адресу, записанному во втором и третьем байтах команды, иначе перейти к следующей команде.

Команда безусловного вызова подпрограммы :

CALL_B2_B3 – вызов подпрограммы с начальным адресом, записанным во втором и третьем байтах команды.

Команды условного вызова подпрограммы:

CNZ – если z = 0
CZ – если z = 1
CNC – если c = 0
CC – если c = 1
CPO – если p = 0
CPE – если p = 1
CP – если s = 0
CM – если s = 1

то перейти к подпрограмме с начальным адресом, указанном во втором и третьем байтах команды, иначе перейти к следующей команде

Команда безусловного возврата из подпрограммы – RET.

Команды условного возврата из подпрограмм:

RNZ – если z = 0
RZ – если z = 1
RNC – если c = 0
RC – если c = 1
RPO – если p = 0
RPE – если p = 1
RP – если s = 0
RM – если s = 1

то возвратиться к команде, следующей за командой вызова подпрограммы, иначе продолжить выполнение подпрограммы.

Группа № 5. Команды управления процессором

Сюда входят две команды ввода-вывода с прямой адресацией порта IN, OUT. Команды EI и DI разрешают и запрещают прием запросов на прерывания, сбрасывая и устанавливая маску прерывания.

Однобайтовая команда RST n, n=0..7, представляющая собой укороченный вариант команды CALL addr при addr=8*n, обеспечивает возможность программной инициализации процедур обслуживания прерываний и вызова операционной системы или ее специальных средств (таб. 3.1.5).

Таблица 3.1.5
Команды управления процессором

Мнемоника	Код	Число циклов	Число тактов	Флаги: CY, Z, M, P, C, AC	Содержание
IN port	DB	3	10	– – – – –	A ← IOSEG (port)
OUT port	D3	3	10	– – – – –	IOSEG (port) ← A
RST n	11NNN111	3	11	– – – – –	–(SP) ← PC ← 8*n, n=0…7
EI	FB	1	4	– – – – –	Разрешить прерывание
DI	F3	1	4	– – – – –	Запретить прерывание
RIM*	20	–	–	– – – – –	Чтение маски
SIM*	30	–	–	– – – – –	Установка маски
HLT	76	1	4	– – – – –	Останов
NOP	00	1	7	– – – – –	Нет операции

Команда HLT приостанавливает работу МП, который, однако, сохраняет возможность обслуживания запросов на прерывания при сброшенной маске. МП может быть выведен из состояния останова двумя путями: перезапуском и по сигналу прерывания.

Команда NOP может быть полезна для организации коротких пауз и установки "заплат" на объектный код при его модификации.

Принцип работы МПКР580ВМ

К серьезным недостаткам системы команд ВМ80 следует отнести отсутствие относительной адресации, что привело к невозможности создания перемещаемого объектного кода. Второй недостаток – отсутствие средств защиты памяти.

С целью упрощения понимания принципа работы МП, дадим словесное описание его функционирования во время выполнения команд программы:

1. Перед выполнением очередной команды МП содержит ее адрес в программном счетчике PC;
2. МП обращается к памяти по адресу, содержащемуся в PC, и считывает из памяти первый байт очередной команды в регистр команд IR;
3. Дешифратор команд DCU декодирует содержащийся в IR код команды и в результате его декодирования, в частности, "узнает":
   - какова длина этой команды (1, 2 или 3 байта);
   - где хранятся ее операнды;
   - какие действия нужно выполнить над операндами;
4. В соответствии с полученной от DCU информацией устройство управления вырабатывает упорядоченную во времени последовательность микроопераций, реализующих предписания команды, в том числе:
   - извлекает операнды из регистров и памяти;
   - выполняет над ними предписанные кодом команды арифметические, логические или другие операции;
   - в зависимости от длины команды модифицирует содержимое PC на 1, 2 или 3 (при линейном алгоритме);
   - передает управление очередной команде, адрес которой снова находится в программном счетчике PC.

Рассмотрим, например, команду сложения содержимого аккумулятора и регистра B, имеющую мнемоническое обозначение ADD B. Команда ADD B – однобайтовая и имеет код операции 80h

В начале выполнения этой команды МП выставляет на шину адреса адрес команды, считывает из памяти ее код 80h и помещает его в регистр команд IR. После декодирования команды устройство управления (УУ) вырабатывает предписанную командой последовательность управляющих сигналов, приводящую к следующим действиям:

– содержимое аккумулятора копируется в RGa;

– содержимое регистра B копируется RGb;

– производится суммирование RGa+RGb и результат сложения помещается в аккумулятор A;

– в зависимости от результата операции модифицируется содержимое регистра флагов RS;

Содержимое программного счетчика PC увеличивается на 1, так как команда ADD B является однобайтовой, и теперь программный счетчик содержит адрес следующей команды программы и т.д.

Программная модель микропроцессорной системы

Система команд микропроцессора – это полный перечень элементарных действий, которые может выполнить микропроцессор. Управляемый командами микропроцессор выполняет очень простые действия, однако с помощью этих действий (команд) можно запрограммировать любую сложную операцию.

Проектировщику необходимо запомнить весь перечень команд и хорошо представлять действия, которые будет выполнять микропроцессор при их обработке.

Программная модель системы, построенной на базе микропроцессора серии КР580, состоит из следующих элементов:

– программно–доступных регистров и триггеров разрешения прерывания;

– программно–доступных восьмиразрядных ячеек памяти;

– программно–доступных восьмиразрядных регистров.

Программно – доступные регистры микропроцессора – это регистры общего назначения, регистр указателя стека, регистр

признаков и регистр счетчика команд. Разряды регистров нумеруются справа налево целыми числами, начиная с нуля.

Регистр общего назначения (РОН) — это один из шести восьмиразрядных регистров микропроцессора, обозначенных буквами B, C, D, E, H, L, или аккумулятор, обозначенный буквой A. Регистры B и C, D и E, H и L в некоторых командах рассматриваются как шестнадцатиразрядные регистры, называемые регистровыми парами. Регистры B, D и H образуют старшие восемь разрядов регистровых пар, а регистры C, E и L – младшие.

Регистр указатель стека (УС) — это шестнадцатиразрядный регистр, который содержит адрес вершины стека. *Стек* – это динамическая последовательная структура данных в ОЗУ, организованная таким образом, что очередная запись данных всегда осуществляется в вершину (начало) стека. Максимальный размер стека равен адресуемой емкости памяти. В вершину стека могут записываться только шестнадцатибитные данные. При записи данных в стек содержимое регистра указателя стека уменьшается на 2, а при считывании — увеличивается на 2.

Регистр признаков (F) — это восьмиразрядный регистр, содержащий признаки результата выполнения команды.

7	6	5	4	3	2	1	0	
			S					**S** – признак знака;
			Z					**Z** – признак нуля;
			O					**V** – признак
			V					дополнительного
			O					переноса;
			P					**P** – признак
			1					четности;
			C					**C** – признак
Распределение признаков по разрядам								переноса.

Признаки устанавливаются следующим образом:

– признак **S** — единица, если седьмой разряд результата равен единице, в противном случае — ноль;

– признак нуля **Z** — единица, если во всех разрядах результата нули, в противном случае — ноль;

– признак дополнительного переноса **V** — единица при переносе из третьего разряда или при заеме в третий разряд результата, в противном случае — ноль;

– признак четности **P** — единица, если результат в двоичном коде содержит четное количество единиц, в противном случае — ноль;

– признак переноса **C** — единица при переносе из седьмого разряда или при заеме в седьмой разряд результата, в противном случае — ноль.

Аккумулятор и регистр признаков образуют слово состояния процессора, обозначенное буквами PSW. Аккумулятору соответствуют восемь старших разрядов, а регистру признаков — младшие.

Регистр счетчика команд (СК)— это шестнадцатиразрядный регистр, указывающий адрес следующей команды, которая должна быть выполнена микропроцессором.

Триггер разрешения прерывания используется для управления прерываниями микропроцессора. Если триггер установлен в единицу, то прерывание разрешается, если триггер установлен в ноль, то прерывание запрещается.

Программно–доступные восьмиразрядные ячейки памяти используются в качестве памяти микропроцессорной системы (МПС) . Разряды ячейки памяти нумеруются справа налево целыми числами, начиная с ноля. Максимальная емкость памяти, реализуемой запоминающим устройством, равна 65 536 байт.

Программно–доступные восьмиразрядные регистры используются для ввода и вывода. Максимальное число регистров для ввода данных составляет 256, для вывода данных — столько же.

Практическая часть

В таблице 3.1.6 приведены действия, с которыми нужно оперировать в практической работе № 5

Таблица 3.1.6

Математические операции выполняемые на ассемблере

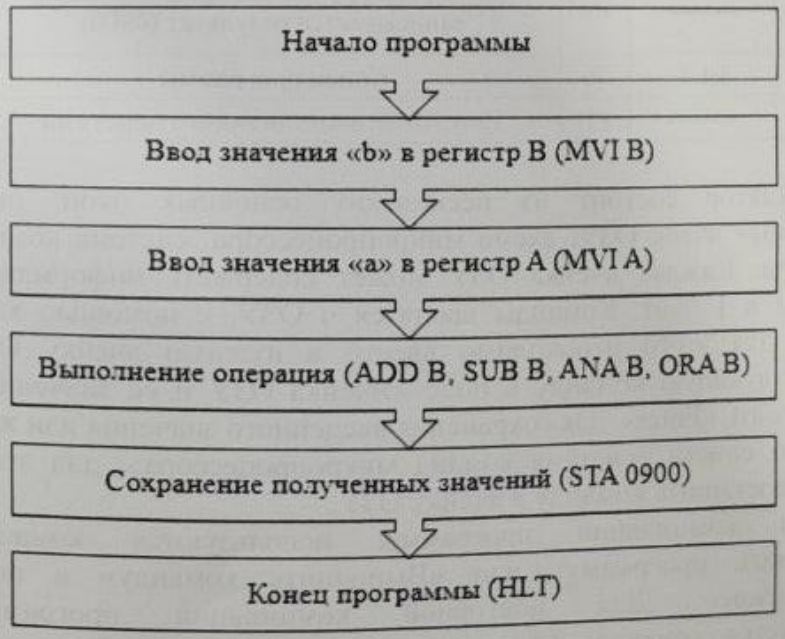
Команда	Действие	Операция	Наименование
ADD B	«+»	$A + B$	Плюс
SUB B	«—»	$A - B$	Минус
ANA B	«↑»	$A \uparrow B$	Логическое И
ORA B	«↓»	$A \downarrow B$	Логическое ИЛИ

Пример:

Найти a - b, где $a = 5_{10}$, $b = 2_{10}$.

В первую очередь, нужно построить блок. Для ввода данных значений необходимо перевести их из десятичной системы счисления в двоичную. Далее выполнить программу приведенную в таблице 3.1.7. В таблице приведен пример выполнения программы на эмуляторе МП системы на базе КП580ВМ80.

Блок схема:



Для выполнения арифметических и логических операций, в этом практической работе, на эмуляторе КР580ВМ, необходимо

задействовать аккумулятор и один из регистров общего назначения. Далее выполнить одну из операций приведенную в таблице 3.1.6.

Таблица 3.1.7

Пример программы, для выполнения практической работы

Адрес	Мнемокод	Код	Комментарии
0000	MVI B	06	Ввод в регистр B
0001	--------	05	Значение загружаемое в регистр B
0002	MVI A	3E	Ввод в аккумулятор A
0003	--------	02	Значение загружаемое в аккумулятор A
0004	ADD B SUB B ANA B ORA B	80 90 A0 B0	Операция суммы, результат в A Операция вычитания, результат в A Операция лог И, результат в A Операция лог. ИЛИ, результат в A
0005	STA 0900	32	Запись в ОЗУ по адресу 0900
0006	--------	00	Запись в 2 последующих байтах после STA, числа формирующий адрес, в который записывается результат (0900)
0007	--------	09	
0008	HLT	76	Конец программы
0900	--------	FD	Результат выполненного действия

Эмулятор состоит из нескольких основных окон: окно содержимых ячеек ОЗУ, схема микропроцессора, система команд для ввода. Каждая ячейка ОЗУ может содержать информацию размером в 1 байт. Команды вводятся в ОЗУ, с помощью кода команд, для этого необходимо ввести в нужную ячейку ОЗУ соответствующую команду в поле «Ячейка ОЗУ и её значения», после нажать «Enter», для сохранения введенного значения или же с помощью списка «система команд микропроцессора», для этого нужно перетащить команду в ячейку ОЗУ.

Для компиляции программы используются команды «Выполнить программу» или «Выполнить команду» в поле «Выполнение». Для повторной компиляции программы используется команда «Сброс регистров» в поле «Сброс» или команда «Сброс ОЗУ» для полного очищения ячеек ОЗУ.

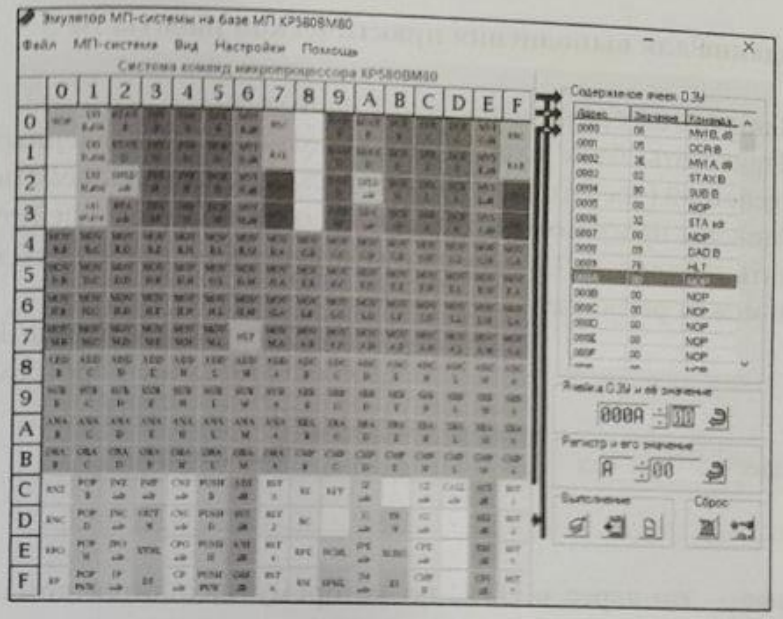


Рисунок 3.1.3 – Пример ввода программы в эмулятор

Перейдя по адресу «0900» можно увидеть результат записанный в ячейку памяти ОЗУ (рис 3.1.4).

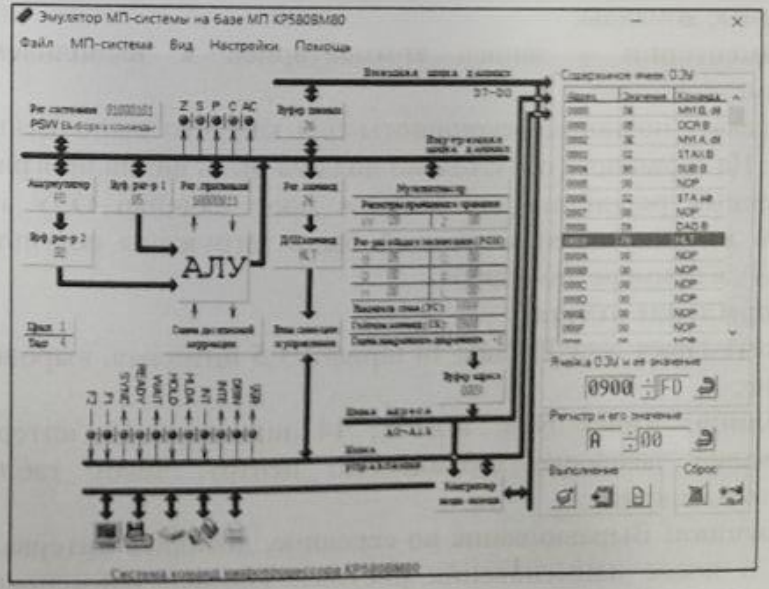


Рисунок 3.1.4 – Пример компиляции программы в эмуляторе

Задания для выполнения практической работы № 5

1. Написать алгоритм выполнения программы.

2. Нарисовать блок схему для выполнения приведенных в таблице действий (достаточно одной блок схемы, для всех действий)

3. Написать программы для выполнения приведенных действий и заполнить таблицу № 3.1.8, при этом можно использовать только операции между регистрами:

Таблица 3.1.8

Таблица для записи программы

Адрес	Мнемокод	Код	Комментарии

Адрес – это адрес в ОЗУ, по которому находится команда. Максимальный адрес которого можно достигнуть в данном эмулятор – FFFF

Код (Значение) –столбец, в который записывается код команды или вводимые значения.

Мнемокод (Команда) – в данном столбце приведены выполняемые команды.

Комментарии – запись комментариев к выполняемым операциям.

4. Сделать поэтапные скриншоты для каждого выполненного действия. На скриншоте обязательно должна быть видна программа и полученный результат, для этого в поле «ячейка ОЗУ и её значение» выбрать адрес в который были загружены результаты (рис. 3.1.4), в примере это «0900».

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 5.1 – …»

Контрольные вопросы

1. Какова структура ассемблерной программы?
2. Дайте определение языку ассемблера
3. В чем отличие инструкции от директивы?
4. Каковы правила оформления программ на языке ассемблера?
5. Каковы этапы получения выполняемого файла?
6. Для чего нужен этап отладки программы?
7. Опишите состав микропроцессора
8. Что включает в себя блок регистров рассматриваемого микропроцессора?
9. Какие функции выполняет схема управления и синхронизации?
10. Из сколько групп состоит система команд рассматриваемого МП?
11. Какие команды являются командами управления процессором?
12. Какие команды являются командами пересылки?
14. Какие команды являются командами арифметической обработки?

Таблица 3.1.9

Варианты для выполнения практической работы

№	A+B	A–B	$A \uparrow B$	$A \downarrow B$
1	a = 01 b =12	a = 01 b =12	a = 01 b =12	a = 01 b =12
2	a = 02 b =22	a = 02 b =22	a = 02 b =22	a = 02 b =22
3	a = 03 b =32	a = 03 b =32	a = 03 b =32	a = 03 b =32
4	a = 04 b =42	a = 04 b =42	a = 04 b =42	a = 04 b =42
5	a = 05 b =52	a = 05 b =52	a = 05 b =52	a = 05 b =52
6	a = 06 b =62	a = 06 b =62	a = 06 b =62	a = 06 b =62
7	a = 07 b =72	a = 07 b =72	a = 07 b =72	a = 07 b =72
8	a = 08 b =82	a = 08 b =82	a = 08 b =82	a = 08 b =82
9	a = 09 b =92	a = 09 b =92	a = 09 b =92	a = 09 b =92
10	a = 0A b =A2	a = 0A b =A2	a = 0A b =A2	a = 0A b =A2
11	a = 0B b =B2	a = 0B b =B2	a = 0B b =B2	a = 0B b =B2

12	a = 0C b =C2	a = 0C b =C2	a = 0C b =C2	a = 0C b =C2
13	a = 0D b =D2	a = 0D b =D2	a = 0D b =D2	a = 0D b =D2
14	a = 0E b =E2	a = 0E b =E2	a = 0E b =E2	a = 0E b =E2
15	a = 0F b =F2	a = 0F b =F2	a = 0F b =F2	a = 0F b =F2
16	a = 0E b =E2	a = 0E b =E2	a = 0E b =E2	a = 0E b =E2
17	a = 0D b =D2	a = 0D b =D2	a = 0D b =D2	a = 0D b =D2
18	a = 0C b =C2	a = 0C b =C2	a = 0C b =C2	a = 0C b =C2
19	a = 0B b =B2	a = 0B b =B2	a = 0B b =B2	a = 0B b =B2
20	a = 0A b =A2	a = 0A b =A2	a = 0A b =A2	a = 0A b =A2
21	a = 09 b =92	a = 09 b =92	a = 09 b =92	a = 09 b =92
22	a = 08 b =82	a = 08 b =82	a = 08 b =82	a = 08 b =82
23	a = 07 b =72	a = 07 b =72	a = 07 b =72	a = 07 b =72
24	a = 06 b =62	a = 06 b =62	a = 06 b =62	a = 06 b =62
25	a = 05 b =52	a = 05 b =52	a = 05 b =52	a = 05 b =52
26	a = 04 b =42	a = 04 b =42	a = 04 b =42	a = 04 b =42
27	a = 03 b =32	a = 03 b =32	a = 03 b =32	a = 03 b =32
28	a = 02 b =22	a = 02 b =22	a = 02 b =22	a = 02 b =22
29	a = 01 b =12	a = 01 b =12	a = 01 b =12	a = 01 b =12
30	a = 0F b =B2	a = 0F b =B2	a = 0F b =B2	a = 0F b =B2

4.2. Знакомство с основными операторами языка ассемблер. Выполнение сложных арифметических операций

Цель работы: Выполнение сложных арифметических операций на языке ассемблер.

Теоретическая часть

Любой алгоритмический язык программирования, в том числе и ассемблер, имеет операторы следующих типов:

Исполнительные операторы. Данные операторы преобразуются транслятором в машинные инструкции. Один исполнительный оператор ассемблера преобразуется в одну машинную инструкцию. А один исполнительный оператор языка высокого уровня транслируется в несколько машинных инструкций.

Все исполнительные операторы языка программирования делятся на операторы обработки данных и операторы передачи управления. Операторы обработки данных влияют на содержимое ячеек памяти (ячейки ОП, регистры, флаги), а операторы передачи управления изменяют ход выполнения программы.

Псевдо-операторы определения данных. В отличие от исполнительного оператора псевдо-оператор ни в какие машинные инструкции не транслируется, а представляет собой указание транслятору со стороны программиста. Псевдо-оператор определения данных требует от транслятора выделить область памяти заданной длины. Кроме того, он может попросить транслятор поместить в выделенную область какие-то первоначальные данные. Впоследствии на этапе выполнения сама программа может менять содержимое этой области.

Другие псевдо-операторы. Они информируют транслятор о структуре программы, помогая транслятору и редактору связей правильно преобразовать исполнительные операторы в машинные инструкции.

Макрооператоры. Каждый такой оператор заменяется транслятором на несколько обычных операторов языка программирования (в том числе, возможно, и псевдо-операторов).

Комментарии. Это любые сообщения в исходной программе, предваряемые специальным символом. В рассматриваемом языке ассемблера это символ ";". Комментарии игнорируются транслятором и никак не влияют на текст машинной программы.

Операторы обработки данных делятся на:

1. арифметические операторы;
2. логические операторы;
3. операторы передачи данных;
4. операторы манипуляций флажками;
5. операторы сдвигов;
6. цепочечные (строковые) операторы.

Подробное описание операторов языка Ассемблер для КР580ВМ80 предоставлены в прошлой теме.

Арифметические команды рассматривают коды операндов как числовые двоичные или двоично–десятичные коды. Эти команды могут быть разделены на пять основных групп:

– команды операций с *фиксированной запятой* (сложение, вычитание, умножение, деление);

– команды операций с *плавающей запятой* (сложение, вычитание, умножение, деление);

– команды очистки;

– команды инкремента и декремента;

– команда сравнения.

Команды операций с *фиксированной запятой* работают с кодами в регистрах процессора или в памяти как с обычными двоичными кодами. Команда сложения (ADD) вычисляет сумму двух кодов. Команда вычитания (SUB) вычисляет разность двух кодов. Команда умножения (MUL) вычисляет произведение двух кодов (разрядность результата вдвое больше разрядности сомножителей). Команда деления (DIV) вычисляет частное от деления одного кода на другой. Причем все эти команды могут работать как с числами со знаком, так и с числами без знака.

Команды операций с *плавающей запятой* (точкой) используют формат представления чисел с порядком и *мантиссой* (обычно эти числа занимают две последовательные ячейки памяти). В современных мощных процессорах набор команд с *плавающей запятой* не ограничивается только четырьмя арифметическими действиями, а содержит и множество других более сложных команд, например, вычисление тригонометрических функций, логарифмических функций, а также сложных функций, необходимых при обработке звука и изображения.

Команды инкремента (увеличения на единицу, INC) и декремента (уменьшения на единицу, DEC) также бывают очень удобны. Их можно в принципе заменить командами суммирования с единицей или вычитания единицы, но инкремент и декремент выполняются быстрее, чем суммирование и вычитание. Эти команды требуют одного входного операнда, который одновременно является и выходным операндом.

Наконец, команда сравнения (обозначается *CMP*) предназначена для сравнения двух входных операндов. По сути, она вычисляет разность этих двух операндов, но выходного операнда не формирует, а всего лишь изменяет биты в регистре состояния процессора (*PSW*) по результату этого вычитания. Следующая за

командой сравнения команда (обычно это *команда перехода*) будет анализировать биты в регистре состояния процессора и выполнять действия в зависимости от их значений.

Микропроцессор – центральное устройство (или комплекс устройств) ЭВМ (или вычислительной системы), которое выполняет арифметические и логические операции, заданные программой преобразования информации, управляет вычислительным процессом и координирует работу устройств системы (запоминающих, сортировальных, ввода — вывода, подготовки данных и др.). В вычислительной системе может быть несколько параллельно работающих процессоров; такие системы называют многопроцессорными. Наличие нескольких процессоров ускоряет выполнение одной большой или нескольких (в том числе взаимосвязанных) программ. Основными характеристиками микропроцессора являются быстродействие и разрядность. Быстродействие – это число выполняемых операций в секунду. Разрядность характеризует объём информации, который микропроцессор обрабатывает за одну операцию: 32–разрядный процессор за одну операцию обрабатывает 32 бит информации, 64–разрядный – 64 бита. Скорость работы микропроцессора во многом определяет быстродействие компьютера. Он выполняет всю обработку данных, поступающих в компьютер и хранящихся в его памяти, под управлением программы, также хранящейся в памяти. Основные функции центрального процессора это обработка данных по заданной программе путем выполнения арифметических и логических операций.

Параметры определяющие производительность:

1. Тактовая частота (Частота ядра) (Internal clock) – это количество электрических импульсов в секунду.

2. Объем Кэш–памяти (Cache) – Кэш–память быстрая память, используемая процессором для ускорения операций, требующих обращения к памяти. На общую производительность влияет размер кэша L2. Чем больше L2, тем дороже процессор, т.к. память для кэша еще очень дорога.

3.Разрядность – максимальное количество разрядов двоичного кода, которые могут обрабатываться или передаваться одновременно (32 или 64 бита обычно)

4. Быстродействие микропроцессора – это число элементарных операций, выполняемых микропроцессором в единицу времени (операции/секунда).

На базовом уровне компьютеры оперируют только числами. Даже в прикладных программах на всех языках внутри много чисел и операций над ними.

В этих командах один из операндов содержится в аккумуляторе A, который одновременно служит приемником результата. Источником второго операнда служат:

1) регистр src;

2) ячейка памяти (HL);

3) константа, заданная непосредственно в команде.

Особым отличием операторов языка ассемблера для КР580ВМ80 является отсутствие операторов умножения и деления которые используются в микропроцессорах Intel8086, используемый в эмуляторе Emu8086.

Пример выполнение операции умножения:

Так как операция умножения не предусмотрена арифметико–логическим устройством, выполнить её можно используя простейшие арифметические операции и цикл (рис. 3.2.1). Нужно оба числа записать в регистры общего назначения (РОН). Далее, добавить один регистр к аккумулятору, а ко второму применить операцию декремента, для организации цикла, который будет продолжаться пока регистр не будет равен нулю.

Таблица 3.2.1

Умножение чисел

Адрес	Метка	Мнемокод	Код	Комментарии
0000		MVI B 05h	06 05h	B ← 05h
0002		MVI E 06h	1E 06h	A ← 06h
0004	M_1	ADD B	80	A ← A+B
0005		DCR E	1D	E ← E–1
0006	(M_1)	JNZ 0004	C2 04 00	If(не 0) → 0004h (M_1)
0009		STA 0900	32 00 09	Запись в ОЗУ 0900
000C		HLT	76	Конец программы
0900			1F	Результат умножения

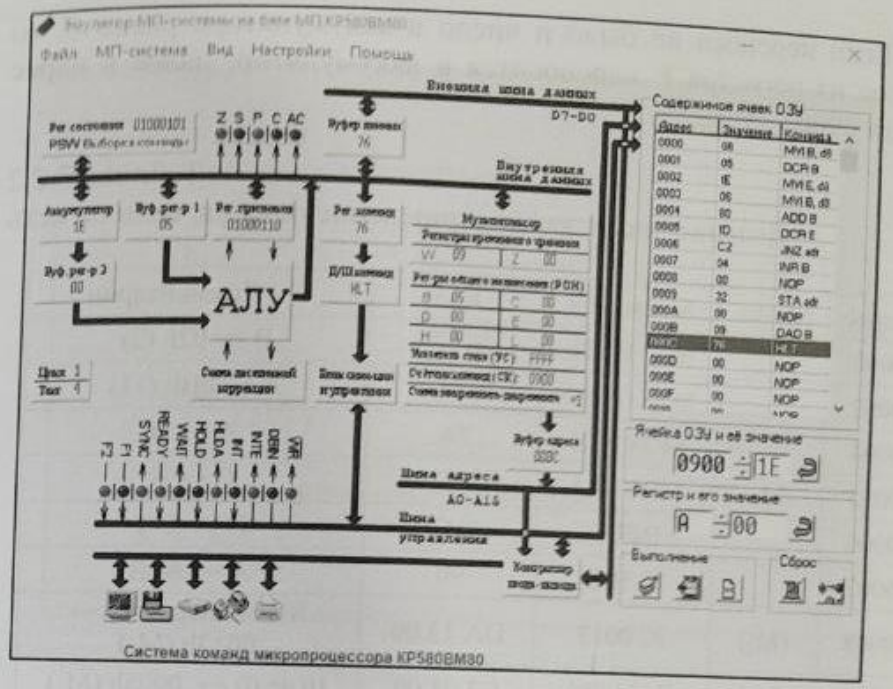


Рисунок 3.2.1 – Операция умножения на эмуляторе КР580

Пример выполнения операции деления:

Так как операция деления не предусмотрена арифметико–логическим устройством, выполнить её можно используя простейшие арифметические операции и циклы. В первом примере (рис. 3.2.2, таб. 3.2.2) предоставлено деление с остатком (программа может вычислить и деление без остатка), вторая программа осуществляет деление без остатка (таб. 3.2.3).

В первой программе организуется цикл, при котором выполняется операция вычитания делителя от делимого, для счета количества проведенных операций вычитания используется счетчик образованный в регистре «E», далее число проверяется на наличие переноса, потом проверяется на 0, если переноса не было, число не равно 0, то цикл продолжается.

Если перенос был, то в аккумулятор выписывается остаток, который переносится по адресу 0901, далее выписывается число, из регистра «E», для получения частного нужно отнять от числа в аккумуляторе 1, полученный результат переносится ОЗУ по адресу 0900.

Если переноса не было и число в аккумуляторе равно 0, то частное из регистра E переносится в аккумулятор, далее в адрес памяти 0900.

Таблица 3.2.2

Деление чисел, также вычисление остатка, если он есть

Адрес	Метка	Мнемокод	Код	Комментарии
0000		MVI B 02h	06 02h	B ← 02h (2)
0002		MVI D 0Ah	16 0Bh	A ← 0Bh (11)
0004		MOV A D	7A	A ← D
0005	M_1	MOV C A	4F	C ← A
0006		INR E	1C	E ← E+1
0007		SUB B	90	A ← A–B
0008	(M_2)	JC 0013	DA 13 00	If(есть перенос) → 0013h (M_1)
000B	(M_1)	JNZ 0005	C2 05 00	If(не 0) → 0005h (M_1)
000E		MOV A E	7B	A ← E
000F		STA 0900	32 00 09	Запись в ОЗУ 0900
0012		HLT	76	Конец программы
0013	M_2	MOV A C	79	A ← C
0014		STA 0901	32 01 09	Запись в ОЗУ 0901 (Остаток)
0017		MOV A E	7B	A ← D
0018		SUI 01	D6 01	A ← A–01
001A		STA 0900	32 00 09	Запись в ОЗУ 0900 (Целая часть)
001D		HLT	76	Конец программы
0900		--------	05h	Результат деление
0901		--------	01h	Остаток

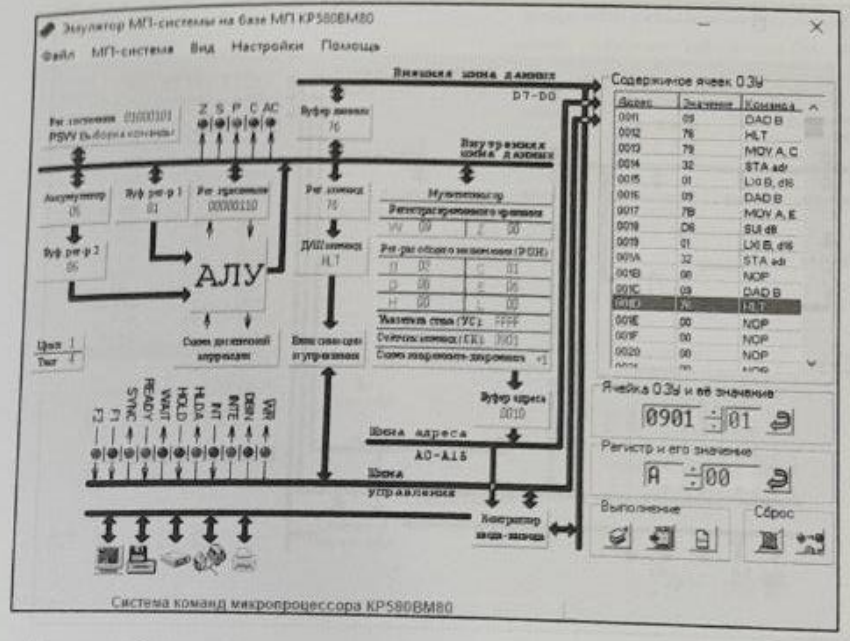


Рисунок 3.2.2 – Программа вычисляющая деление с остатком

В алгоритме выполнения деления без остатка отсутствует повторная проверка на 0, при переполнении число из счетчика «E» переносится в аккумулятор, из него вычитается 1, и переносится в адресную память по адресу 0900. Остаток отбрасывается.

Таблица 3.2.3

Деление чисел без остатка

Адрес	Метка	Мнемокод	Код	Комментарии
0000		MVI B 02h	06 02h	B ← 02h (2)
0002		MVI D 0Ah	16 10h	A ← 10h (16)
0004	M_1	MOV A D	7A	A ← D
0005		INR E	1C	E ← E+1
0006		SUB B	90	A ← A–B
0007	(M_2)	JNC 0005	D2 05 00	If(нет переноса) → 0005h (M_1)
000A		MOV A E	7B	A ← E
000B		SUI 01	D6 01	A ← A–01
000D		STA 0900	32 00 09	Запись в ОЗУ 0900
0010		HLT	76	Конец программы
0900		--------	08h	Результат деление

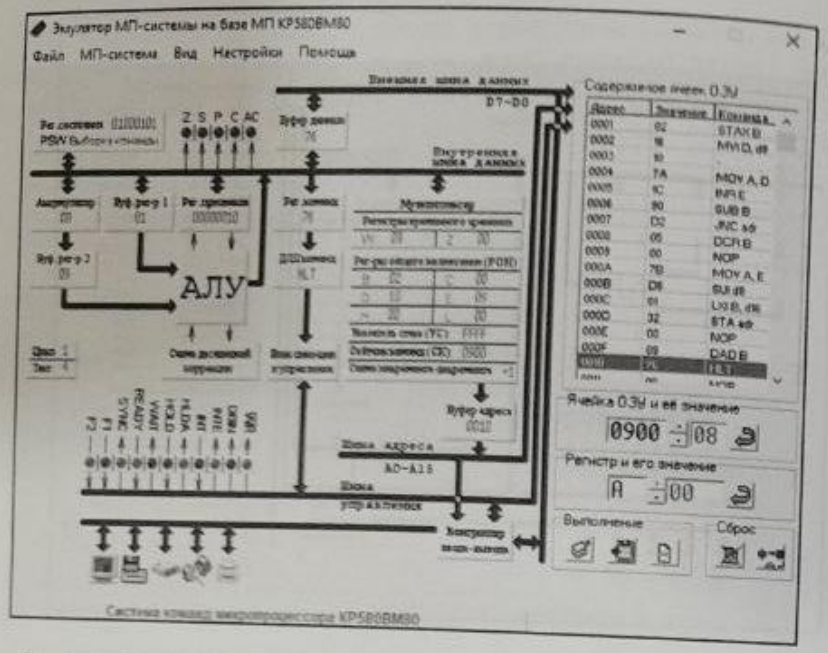


Рисунок 3.2.3 – Программа вычисляющая деление без остатком

Практическая часть

В практической работе № 6 необходимо произвести сложные арифметические и логическое операции:

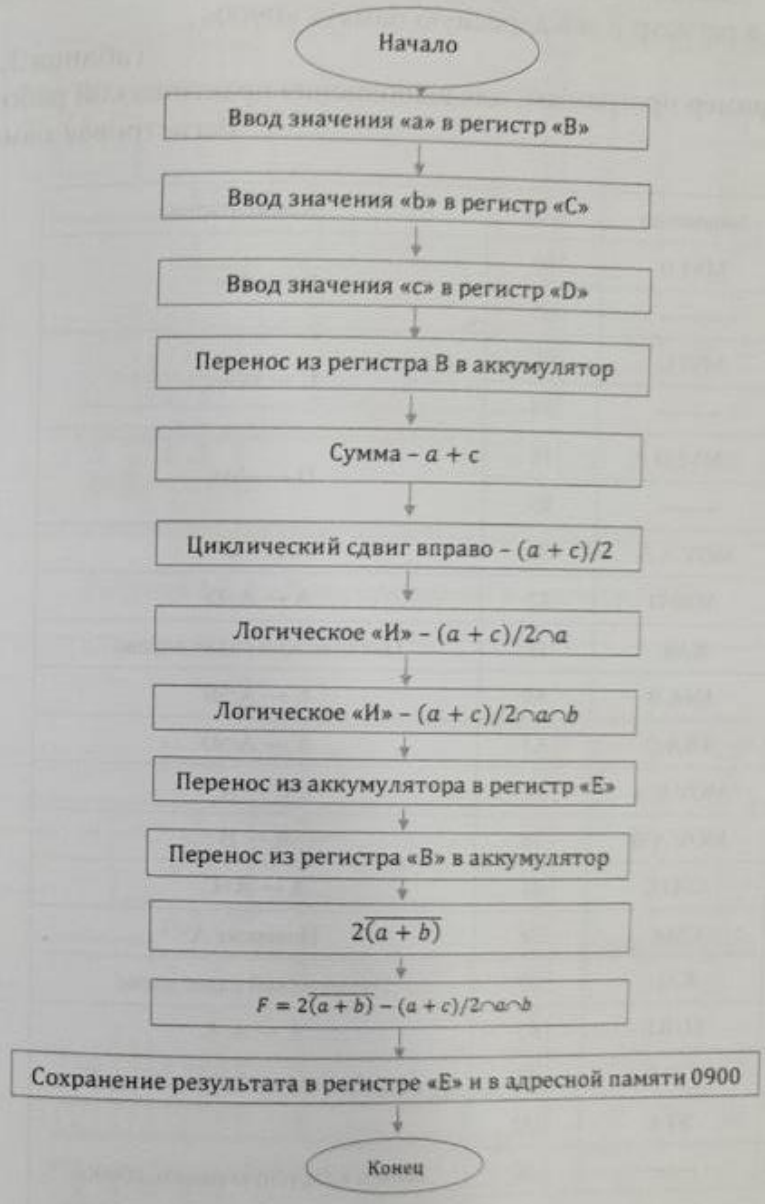
Вычислить: $F = 2\overline{(a+b)} - \frac{(a+c)}{2} \cap a \cap b$

Для того что бы выполнить вычисление приведенной функции нужно поделить её на несколько действий:

1. $\frac{(a+c)}{2}$
2. $\frac{(a+c)}{2} \cap a \cap b$
3. $2\overline{(a+b)}$
4. $2\overline{(a+b)} - (a+c)/2 \cap a \cap b$

Так как на микропроцессоре КР580ВМ команды вычитания SUB Rn имеет следующее описание: A – Rn → A, целесообразно начинать вычисления справа от знака минус, это необходимо для того что бы сэкономить количество команд требуемых для выполнения поставленной задачи.

Блок схема:



Для выполнения программы, нужно загрузить значения «a», «b» и «c» в свободные регистры общего назначения. Используя команды арифметических и логических операций выполнить

необходимые действия, для умножения на 2 – использовать команду RAL, для деления на 2 использовать команду RAR. Запись результата в регистр E и в адресную память «0900».

Таблица 3.2.4

Пример программы, для выполнения практической работы. Регистровая память

Адрес	Мнемокод	Код	Комментарии
0000	MVI B	06	B ← «05»
0001	--------	05	
0002	MVI C	0E	C ← «08»
0003	--------	08	
0004	MVI D	16	D ← «05»
0005	--------	05	
0006	MOV A B	78	A ← B
0007	ADD D	82	A ← A+D
0008	RAR	1F	Циклический сдвиг вправо
0009	ANA B	A0	A ← A∩B
000A	ANA C	A1	A ← A∩D
000B	MOV E A	5F	E ← A
000C	MOV A B	78	A ← B
000D	ADD C	81	A ← A+C
000E	CMA	2F	Инверсия A
000F	RAL	17	Циклический сдвиг влево
0010	SUB E	93	A ← A–E
0011	MOV E A	5F	E ← A. Сохранение в регистр E
0012	STA	32	Запись в адресную память «0900»
0013	--------	00	
0014	--------	09	
0015	HLT	76	Конец программы
0900	--------	3E	Результат программы

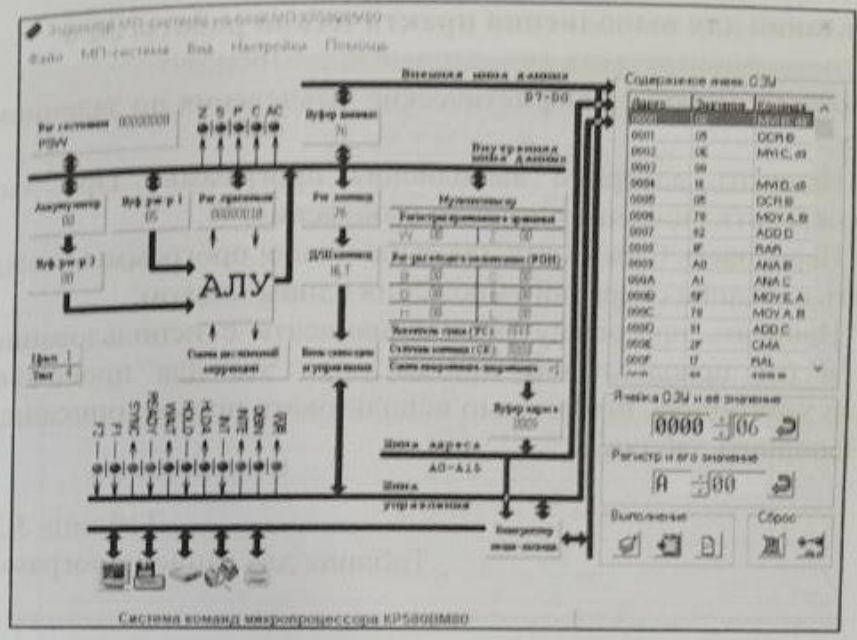


Рисунок 3.2.4 – Ввод программы в эмулятор КР580

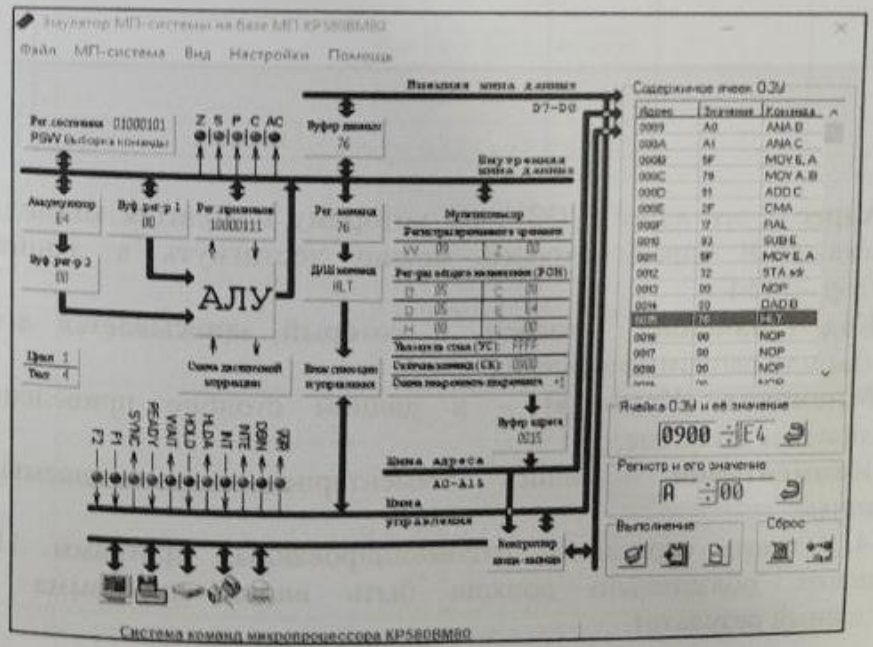


Рисунок 3.2.5 – Выполнение программы. Результат записан в аккумуляторе

Задания для выполнения практической работы № 6

Провести сложные арифметические вычисления по заданному варианту.

1. Написать алгоритм выполнения программы. При этом можно сократить описание операции умножения.

2. Нарисовать блок схему для выполнения программы, можно сократить описание операции умножения одним блоком.

3. Написать программу (можно написать с использованием циклов и без использования циклов), при это для проведения операции умножения, необходимо использовать пример описанный выше. Заполнить таблицу:

Таблица 3.2.5
Таблица для записи программы

Адрес	Мнемокод	Код	Комментарии

Адрес – это адрес в ОЗУ, по которому находится команда. Максимальный адрес которого можно достигнуть в данном эмулятор – FFFF

Код (Значение) –столбец, в который записывается код команды или вводимые значения.

Мнемокод (Команда) – в данном столбце приведены выполняемые команды.

Комментарии – запись комментариев к выполняемым операциям.

4. Сделать скриншоты откомпилированных программ. На скриншоте обязательно должна быть видна программа и полученный результат.

Таблица 3.2.6
Варианты для выполнения практической работы № 6

1	$F = \left(\frac{a+bc}{2}\right) + \left(\frac{a}{2}+b\right)(a-b)$
2	$F = \left(\frac{a+c}{4}a\right) + \left(\frac{a}{2}+\frac{b}{4}\right)(a-b-c)$
3	$F = \left(\frac{a+b-c}{2}\right)\left(\frac{b+c}{4}\right) - bc$
4	$F = \left(\frac{a+bc}{2}\right)\left(\frac{a+b+c}{4}\right) - abc$
5	$F = \left(\frac{a+bc}{8}\right)\left(\frac{a+b}{2}\right) + (a-b-c)$
6	$F = \left(\frac{ab+c}{4}\right)\left(\frac{a+b}{2}\right)$
7	$F = (a+b) + \frac{bc}{4} - \overline{ac}$
8	$F = a + \frac{ab}{4} + \frac{\overline{bc}}{2}ac$
9	$F = \overline{ab} + \frac{bc}{4} - c + \left(\frac{a+b}{2}\right)$
10	$F = \left(\left(\frac{ab}{2}\right) + \left(\frac{bc}{2}\right)\right) - ac$
11	$F = \overline{(a+\overline{bc})}bc + \overline{(abc)}$
12	$F = \overline{(a+bc)} + (\overline{abc})bc$
13	$F = (a+\overline{bc}) + \overline{(a+bc)} - abc$
14	$F = \overline{(\overline{ab}+bc)} + \overline{(ab+c)} - abc$
15	$F = (ab+bc) + \overline{(ac+\overline{bc})}$
16	$F = (abc) + \overline{(\overline{ab}+c)}ac$
17	$F = (a\overline{bc}) + (a+bc)ac$
18	$F = \overline{(abc)} + (ab+bc)(b-c)$
19	$F = \left(\overline{a} + \frac{ab}{2}\right) + \left(\overline{a} + \frac{ac}{2}\right)$
20	$F = 2\overline{(a+b)} + \frac{(a\uparrow b)}{2}ab$
21	$F = \left(\frac{a+b}{2}\right) - 2\overline{(a+c)}bc$
22	$F = \left(\left(\frac{a+b}{2}\right) - \left(\frac{a+c}{2}\right)\right) * 2(a-bc)$

23	$F = (a + b) * 2 - \overline{(a + c) * 2} + ab$
24	$F = \overline{(a + b) * 2} + (a + c) * 2 + (a - c)(b - a)$
25	$F = \overline{(a + b - c) * 2} - \overline{(a + c) * 2}$
26	$F = (a + ab) + \left(a + \frac{ac}{2}\right)$
27	$F = \left(a + \frac{ab}{2}\right) + (ab + ac)$
28	$F = \left(ac + \frac{ab}{2}\right) + \left(ab + \frac{ac}{2}\right)$
29	$F = (a + \overline{ab}) + \left(a + \frac{ac}{2}\right)$
30	$F = \left(\overline{ac} + \frac{ab}{2}\right) + (a + \overline{ac})$

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 6.1 – …»

Контрольные вопросы

1. Каковы правила оформления программ на языке ассемблера?
2. Каковы этапы получения выполняемого файла?
3. Для чего нужен этап отладки программы?
4. Что влияет на быстродействие микропроцессора?
5. Какие команды входят в состав арифметических команд ?
6. Какие команды входят в состав логических команд ?
7. Как вызвать подпрограмму на языке ассемблер, для чего нужны подпрограммы?
8. Как организовать задержку используя программные прерывания?

ГЛАВА IV
ИЗУЧЕНИЕ ВИДОВ ПАМЯТИ. ПРОГРАММНЫЕ СПОСОБЫ МАСКИРОВАНИЯ ДАННЫХ

4.1. Виды памяти и их характеристики. Использование регистровой памяти и памяти озу при программировании на ассемблере

Цель: Работа с разными типами памяти микропроцессора КР580 на языке Ассемблера.

Теоретическая часть

Компьютерная память (устройство хранения информации, запоминающее устройство) — часть вычислительной машины, физическое устройство или среда для хранения данных, используемая в вычислениях систем в течение определённого времени. Память, как и центральный процессор, неизменно является частью компьютера с 1940–х годов. Память в вычислительных устройствах имеет иерархическую структуру и обычно предполагает использование нескольких запоминающих устройств, имеющих различные характеристики.

В персональных компьютерах «памятью» часто называют один из её видов — динамическую память с произвольным доступом (DRAM), — которая используется в качестве ОЗУ персонального компьютера.

Задачей компьютерной памяти является хранение в своих ячейках состояния внешнего воздействия, запись информации. Эти ячейки могут фиксировать самые разнообразные физические воздействия. Они функционально аналогичны обычному электромеханическому переключателю и информация в них записывается в виде двух чётко различимых состояний — 0 и 1 («выключено»/«включено»). Специальные механизмы обеспечивают доступ (считывание, произвольное или последовательное) к состоянию этих ячеек.

Процесс доступа к памяти разбит на разделённые во времени процессы — операцию записи (сленг. прошивка, в случае записи ПЗУ) и операцию чтения, во многих случаях эти операции

Коды символов в ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
9	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
A	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я

FF – пробел

Таблица 5.2.4

Коды цветов эмулятора КР580

Код	Цвет	Код	Цвет
00	Чёрный	40	Синий
10	Коричневый	48	Сиреневый
18	Оранжевый	50	Фиолетовый
20	Зелёный	58	Розовый
28	Светло зелёный	60	Голубой
30	Салатовый	68	Бирюзовый
38	Жёлтый	79	Белый

3. *Исходные данные (заложенные в цикле)*

Таблица 5.2.5

Исходные данные заданные циклом, к программе выше

Адрес	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	200A
Значение	8E	E0	A3	A0	AD	A8	A7	A0	E6	A8	EF
Буква	«О»	«р»	«г»	«а»	«н»	«и»	«з»	«а»	«ц»	«и»	«я»
Адрес	200A	200B	200C	200D	200E	200F	2010	2011	2012	2013	2014
Значение	FF	8A	AE	AC	AF	EC	EE	E2	A5	E0	A0
Буква	« »	«К»	«о»	«м»	«п»	«ь»	«ю»	«т»	«е»	«р»	«а»

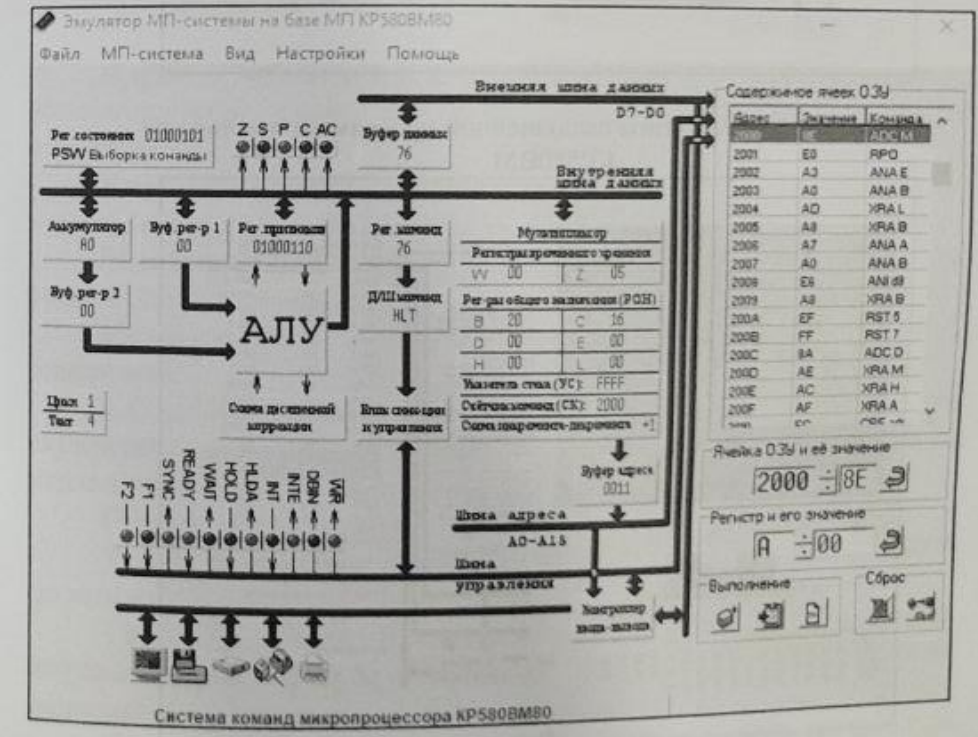


Рисунок 5.2.4 – Ввод данных массива. Коды букв

4. Результат программы

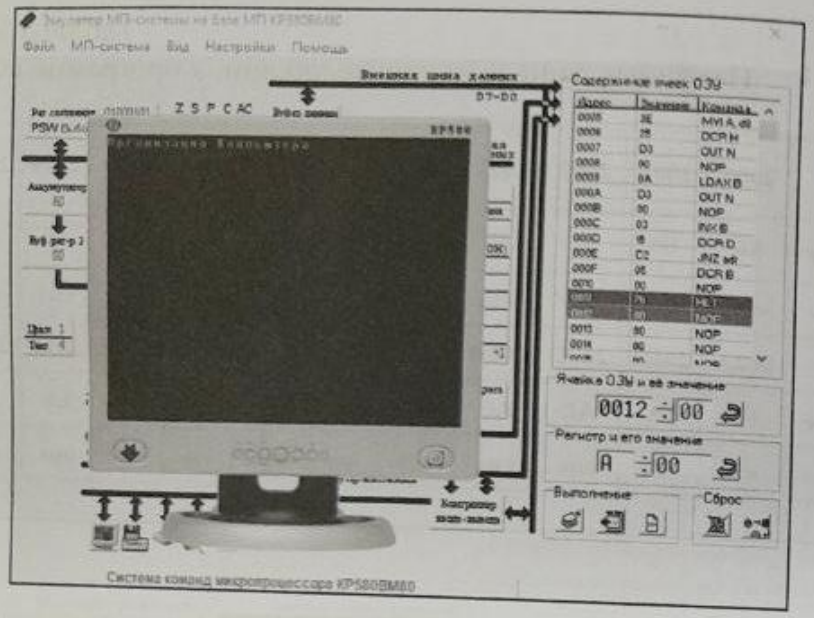


Рисунок 5.2.5 – Результаты выполненной программы в мониторе КР580ВМ

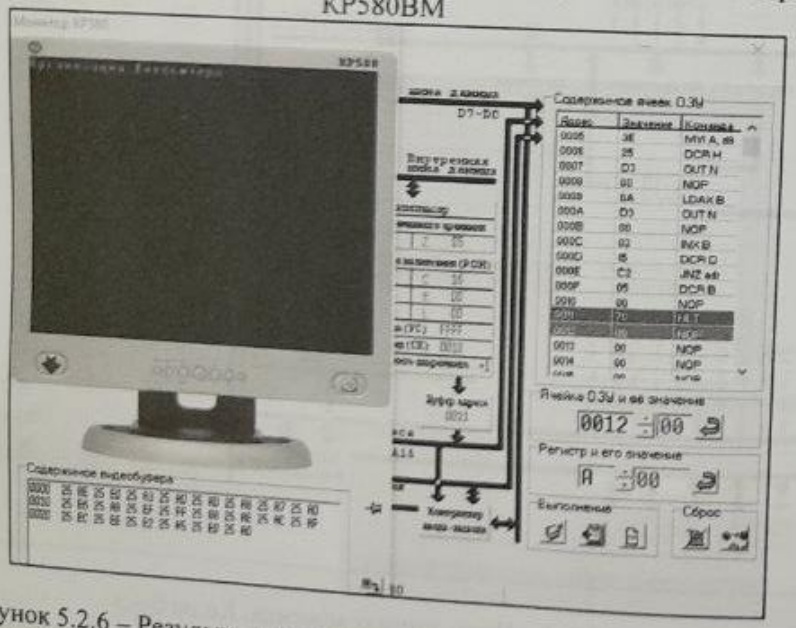


Рисунок 5.2.6 – Результаты выполненной программы в подробном виде монитора КР580ВМ

Задания для выполнения практической работы № 10

Написать программу вывода фамилии и имени на монитор КР580.

1. Написать алгоритм выполнения программы
2. Нарисовать блок схему для выполнения программы;
3. Написать программу (можно написать с использованием циклов и без использования циклов) и заполнить таблицу:

Таблица 5.2.6

Таблица для записи программы.

Адрес	Мнемокод	Код	Комментарии

Адрес – это адрес в ОЗУ, по которому находится команда. Максимальный адрес которого можно достигнуть в данном эмулятор – FFFF

Код (Значение) –столбец, в который записывается код команды или вводимые значения.

Мнемокод (Команда) – в данном столбце приведены выполняемые команды.

Комментарии – запись комментариев к выполняемым операциям.

4. Сделать скриншоты откомпилированной программы. На скриншоте обязательно должна быть видна программа, массив по указанному адресу и полученный результат.

Оформление отчета:

Текст: Times New Roman; 14 шрифт; 1,5 интервал, выровнять по ширине;

Таблица: Times New Roman; 14 шрифт; 1,15 интервал, выравнивание заголовка таблицы по центру, текст таблицы выровнять по ширине;

Картинка: Выравнивание по середине, добавить интервал до рисунка и после наименования рисунка. Пример наименования «Рисунок 10.1 – …»

Контрольные вопросы

1. На какие группы можно разделить внешние интерфейсы ПУ ?
2. Дайте определение шине ввода–вывода.
3. Опишите специализированные интерфейсы.
4. Опишите универсальные интерфейсы.
5. Опишите интерфейс RS–232.
6. Опишите интерфейс Centronics.
7. Опишите интерфейс EPP.
8. Опишите интерфейс SCSI.
9. Опишите интерфейс USB.
10. С помощью чего реализуется цифровой аудиоканал?
11. Опишите дискретный и аналоговый интерфейсы.
12. Дайте определение периферийным устройствам.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. A.Clements. "Computer Organization and Architecture" Australia Cengage Learning 2018. – 816 p.
2. Абель П. Ассемблер. Язык и программирование для IBM – 5–е изд. / СПб.: КОРОНА–Век, 2009. – 734 с.
3. Бабич Н. П. Компьютерная схемотехника. Методы построения и проектирования: учебное пособие/Киев : МК–Прогресс, 2004.–576 с.
4. Богданов Д. С. Архитектура компьютера / Интернет ресурс – URL : http://asm–pc.ru
5. Воеводин В.В., Капитонова А.П. Методы описания и классификации вычислительных систем: Учебное пособие/ Москва: Издательство МГУ,1994. – 73 с.
6. Гук М. Ю. Аппаратные средства IBM PC : энциклопедия – 3–е изд. / СПб.: Питер, 2006. – 1072 с.
7. Докторов А. Е., Докторова Е. А. Архитектура ЭВМ. Методические указания к лабораторным работам / УлГТУ, 2008. – 32 с.
8. Ефимушкина Н.В., Орлов С.П. Вычислительные системы и комплексы: учеб. пособие для вузов/ Москва: Машиностроение–1, 2006. – 268 с.
9. Кобайло А. С. Арифметические и логические основы цифровых вычислительных машин: Учебно–методическое пособие / Минск, 2014 – 76 с.
10. Коваль А. С., Сычев А. В. Архитектура ЭВМ и систем : учебно–методическое пособие для вузов / Воронеж, 2007. – 87 с.
11. Магда Ю. С. Ассемблер для процессоров Intel Pentium – СПб. : Питер, 2006. – 410 с.
12. Максимов Н.В., Партыка Т.Л., И.И. Попов. Архитектура ЭВМ и вычислительных систем: учебник для вузов / Москва:ФОРУМ:ИНФРА–М, 2005. – 512 с.
13. Манохин Ю. П. Архитектура ЭВМ и систем : учебно–методический комплекс / Москва : РГТУ, 2010. – 45 с.
14. Марек Р. Ассемблер на примерах. Базовый курс. – СПб. : Наука и техника, 2005. – 240 с.

15.Мелехин В.Ф., Павловский Е.Г. Вычислительные машины, системы и сети: учебник для вузов/ Москва: Издательский центр «Академия», 2007. – 560 с.

16.Мусаев М.М. Процессоры современных компьютеров. Олий ўқув юртлари учун қўлланма / Т.: "Алоқачи", 2020. – 512 б.

17.Описание простейших команд ассемблера. – Интернет ресурс URL : http://students.uni–vologda.ac.ru/pages/it10/2.php

18.Орлов С.А., Силкер Б.Я. Организация ЭВМ и систем: Учебник для вузов. 2–е изд. — СПб.: Питер, 2011. — 688 с

19.Поворознюк А. И. Архитектура компьютеров. Архитектура микропроцессорного ядра и системных устройств : учебное пособие / Харьков: Торнадо, 2004. – 355 с.

20.Пятибратов А.П., Гудыно Л.П., Кириченко А.А. Вычислительные системы, сети и коммуникации: учеб. пособие/ Москва: Финансы и статистика, Инфра–М, 2008. – 736 с.

21.Сайфуллаева Н.А., Яхшибоев Р.Э. «Организация компьютера». Методические указания по выполнение практических работ для студентов по всем направлениям ТУИТ имени Мухаммада аль–Хоразмий, Ташкент, 2021, 55с.

22.Столингс У. Структурная организация и архитектура компьютерных систем. Москва: Вильямс, 2002.– 896 с.

23.Таненбаум Э., Остин Т. Архитектура компьютера // 6–е издание. СПб.: Питер, 2013. — 811 с

24.Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин ; перевод с английского Е. Матвеева. — 6–е изд. — Санкт–Петербург [и др.] : Питер, 2019

25.Толстобров А. П. Архитектура ЭВМ : учебное пособие / Воронеж, 2004. – 95 с.

26.Харрис, Д. М. Цифровая схемотехника и архитектура компьютера. Дополнение по архитектуре ARM / Д. М. Харрис, С. Л. Харрис; перевод с английского А. А. Слинкин. — Москва : ДМК Пресс, 2019

27.Хорошевский В.Г. Архитектура вычислительных систем: учеб. пособие для вузов / Москва: Изд–во МГТУ им. Н.Э. Баумана, 2005. – 512 с.

28.Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: учебник для вузов/ СПб.: Питер, 2004. – 586 с.

СОДЕРЖАНИЕ

БОТИРОВ С.Р.

ОРГАНИЗАЦИЯ КОМПЬЮТЕРА

Учебное пособие

Ташкент - "METODIST NASHRIYOTI" - 2024

Muharrir: Bakirov Nurmuhammad

Texnik muharrir: Tashatov Farrux
Musahhih: Saidova Nurshoda
Dizayner: Ochilova Zarnigor

Bosishga 11.06.2024.da ruxsat etildi.
Bichimi 60x90. "Times New Roman" garniturasi.
Ofset bosma usulida bosildi.
Shartli bosma tabog'i 13. Nashr bosma tabog'i 13.
Adadi 300 nusxa.

***"METODIST NASHRIYOTI"** MCHJ matbaa bo'limida chop etildi.*
Manzil: Toshkent shahri, Shota Rustaveli 2-vagon tor ko'chasi, 1-uy.

+99893 552-11-21

ISBN 978-9910-03-230-1
9 789910 032301