МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
СУМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

В.Т. Баравой

КОНСПЕКТ ЛЕКЦИЙ ПО КУРСУ

«МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА»
РАЗДЕЛ 1 МИКРОПРОЦЕССОРНЫЙ МОДУЛЬ

для студентов специальности 6.09 14 01 «Системы управления и автоматики» дневной и заочной форм обучения

СУМЫ ИЗД-ВО СУМГУ 2004

ПРЕДИСЛОВИЕ

Конспект лекций соответствует рабочей программе по дисциплине «Микропроцессорные устройства» и представляет собой первый раздел этой дисциплины «Микропроцессорный модуль» на базе микропроцессорного комплекта КР580. Вторая и третья части будут выпущены под названиями соответственно «Электронная память» и «Устройства ввода/вывода».

Курс лекций написан для студентов физико-технических специальностей дневной и заочной форм обучения. Объем каждой лекции рассчитан на два академических часа.

Теоретический материал требует от обучаемого знания аналоговой и цифровой техники, которая изучается в курсе «Электроника и микросхемотехника». В приложении приведены система команд микропроцессора КР580, а также примеры написания ассемблерных программ, которые дадут возможность обучаемым приобрести практические навыки по созданию простых программ на языке Ассемблера.

Дополнительно приводится материал по микропроцессору К1810 и микроконтроллеру КМ1816, где предлагается материал по архитектуре и функционированию данных устройств

Лекции содержат тщательно подобранный материал, имеют большое количество иллюстрации, таблиц, структурных и функциональных схем. Подробно рассматриваются отдельные микросхемы, их сигналы, функциональное назначение.

Для углубленного изучения отдельных лекции приводится список литературы.

Подробное описание рассматриваемых микропроцессорных устройств позволит обучаемому ознакомиться с компьютерными технологиями, осмыслить физические процессы, происходящие при функционировании компьютера.

ВВОДНАЯ ЛЕКЦИЯ

История развития современных микропроцессоров (МП)

В 1972 году фирма Intel (США) выпустила первый в мире МП Intel 8008. На базе этого МП был создан персональный компьютер Micral (Франция, 1973г). В (1974, 1978, 1979, 1982) годах были выпущены МП соответственно Intel 8080 (аналог КР580), 8088, 80286.

В 1985 году фирма Intel представила первый 32-разрядный МП i80386, аппаратно совместимый снизу вверх со всеми предыдущими МП этой фирмы. Он имел 32-разрядную адресную шину и мог адресовать до 4 Гбайт оперативной памяти, был введен новый режим работы МП- режим виртуального i8086, который обеспечивал большую эффективность программ, разработанных для i8086, и их параллельную работу. Ещё одно важное нововведение- поддержка страничной организации оперативной памяти, что позволило иметь виртуальное пространство памяти размером до 4 Тбайт. Это был первый МП, в котором использовалась параллельная обработка команд. Так, одновременно осуществлялись: доступ к памяти и УВВ; размещение команд в очереди для их выполнения; декодирование команд; преобразование линейного адреса в физический; страничное преобразование адреса.

Вскоре после МП i386 появился МП i486. Дальнейшее развитие получила идея параллельной обработки команд. Был организован пятиступенчатый конвейер, на котором в различной стадии исполнения могло находиться до 5 команд. Появилась КЭШ-память 1-го и 2-го уровней. В систему команд были добавлены новые команды, повысилась тактовая частота до 133МГц. Появилась возможность строить многопроцессорные конфигурации. Всё это позволило увеличить скорость выполнения программ.

С 1993 года фирма Intel выпустила МП Pentium. Идея

параллельной обработки команд получила развитие. В устройстве декодирования и исполнения команд был добавлен второй конвейер. Теперь два конвейера (называемых u и v) могли исполнять две инструкции за один такт. Внутренний КЭШ был увеличен вдвое - до 8 Кбайт для кода и 8 Кбайт для данных. Процессор стал более интеллектуальным. Была добавлена возможность предсказания ветвлений, внутри МП используются 128 и 256- разрядные шины передачи данных. Внешняя шина данных была увеличена до 64 бит.

В МП Pentium Pro были реализованы самые передовые технологии. В частности был добавлен ещё третий конвейер обработки команд (три инструкции за один такт). Усовершенствована организация КЭШ-памяти второго уровня, теперь она расположена на одном кристалле с МП, что повысило производительность МП. Реализована 36- разрядная шина адреса, что позволило адресовать до 64 Гбайт оперативной памяти. Осуществляется динамическое исполнение команд.

В дальнейшем совершенствуются модели МП Pentium за счёт включения новых команд, для поддержки которых несколько изменили программную модель МП. Эти команды, получившие название ММХ-команды (Multi Media eXtention-мультимедийное расширение системы команд), позволили одновременно обрабатывать несколько единиц однотипных данных, а также обрабатывать информацию от звуковых и графических устройств. МП Pentium ММХ имеют большие объёмы КЭШ- памяти, достигающие в некоторых моделях 1Мбайт.

Pentium II являются высокопроизводительными МП, они объединяют достижения архитектуры и системы команд МП Pentium Pro, Pentium ММХ и уже не укладываются в отдельную микросхему, а поставляется в виде картриджа (закрытого блока), имеющего свой вентилятор.

Pentium II работают с внешней частотой 100 МГц и выше и выполняют внутреннее умножение частоты с коэффициентом

3,5 и 4 (Pentium III -350 МГц и Pentium II - 400 МГц соответственно). Имеет новые конструктивные решения, в частности, их корпус выполнен в соответствии с новой технологией изготовления корпусов. Микропроцессором поддерживается несколько режимов энергосбережения.

Pentium III появились на рынке в 1997году, они сочетают в себе архитектуру Pentium II и Pentium MMX. Удвоен размер первичного КЭШ (16И6 Кбайт), размер вторичного КЭШ варьируется от нуля до
2 Мбайт. Использована новая технология корпус-картридж с печатным краевым разъёмом.

Pentium 4- последняя новинка фирмы Intel.

Хронология развития МП

1971► Начало истории развития микропроцессоров. Корпорация Intel выпустила первый в мире микропроцессор Intel 4004. Процессор имел 2300 транзисторов. Тактовая частота — 108 КГц, производительность — 60 тыс. операций в секунду. Процессор использовался в производстве микрокалькуляторов.

1972►Состоялся выпуск первого в мире 8-разрядного процессора Intel 8008. Тактовая частота — 200 КГц, 3500 транзисторов. Производительность - 60 тыс. операций в секунду. На базе этого процессора был создан первый в мире персональный компьютер Micral(Франция, 1973г.).

1974► Корпорация Intel выпустила 8-разрядный процессор 8080. Он использовал 6000 транзисторов, работал с тактовой частотой 2 МГц, выполнял 640 тыс. операций в секунду и мог адресоваться к 64 Кбайт оперативной памяти. На базе этого процессора в США был создан первый в мире персональный компьютер со встроенным языком программирования — Altair 8800.

1976► Компания Zilog выпустила 8-разрядный процессор Z-80 (тактовая частота — 2,5 МГц). В основу системы команд положен расширенный набор команд процессора Intel 8080. На базе этого процессора в 90-е годы только в России было собрано несколько миллионов бытовых компьютеров системы Sinclair.

1978► Компания Intel представила 16-разрядный процессор Intel 8086. Тактовая частота — 4,77 МГц. Адресуемое пространство 1 Мбайт, 29000 транзисторов; производительность — 330 тыс. операций в секунду. Последующие модели работали с частотой 8 МГц (660 тыс. операций в секунду) и 10 МГц (750 тыс. операций в секунду).

1979► Компания Motorola выпустила свой 16-разрядный процессор Motorola 68000. Процессоры этого семейства

легли в основу известных компьютеров Macintosh.

1979► Выпущен процессор Intel 8088.По внутренней архитектуре он 16-разрядный, как и процессор Intel 8086, но работает с 8-разрядной шиной данных. Это позволяло использовать его в работе с широко распространенными в то время 8-разрядными устройствами.

1982► Выпуск процессора 80286 (16 разрядов, тактовая частота 6 МГц, 134 тыс. транзисторов, адресуемое пространство 16 Мбайт). Производительность — 900 тыс. операций в секунду. Последующие модели имели частоту 10 МГц (1,5 млн. операций в секунду) и 12 МГц (2,66 млн. операций в секунду).

1984► Компания Motorola выпустила свой первый 32-разрядный процессор Motorola 68020.

1985► Первый 32-разрядный процессор серии x86 — процессор 80386DX. Он имел 32-разрядные регистры и 32-разрядную шину данных, содержал 275 тыс. транзисторов и работал с частотой 16 МГц. Адресуемое пространство — 4 Гбайт. Последующие модели работали с частотами 25, 33 и 40 МГц.

1989►Создан микропроцессор Intel 80486DX. В нем объединены процессор Intel 80386DX и сопроцессор Intel 80387 в одном корпусе и добавлена кэш-память. Процессор работал с частотой 25 МГц и содержал 1,2 млн. транзисторов. Последующие версии процессора 486DX2 и 486DX4 имели внутреннее умножение частоты и работали с частотами до 40 МГц. Производительность достигла 20млн. операций в секунду.

1993► Выпущены процессоры Pentium 60 (100 млн. операций в секунду) и Pentium 66 (112 млн. операций в секунду). Процессоры содержали 3,1 млн. транзисторов и могли напрямую адресоваться к 4 Гбайт оперативной памяти. Процессоры считаются 32-разрядными, хотя имеют 64-разрядную шину данных.

1994► Второе поколение процессоров Intel Pentium — Pentium 75, 90 и 100. Рабочее напряжение снижено с 5 В до 3,3 В, введено внутреннее умножение частоты.

1995► Модели Intel Pentium 120 и 133 (3,2 млн. транзисторов, 220 млн. операций в секунду).

1995► Процессор нового поколения Intel Pentium Pro с рабочей частотой (внутренней) 150 МГц. Количество транзисторов — 5,5 млн. Впоследствии были выпущены модели для работы с частотой 166, 180 и 200 МГц.

1996► Модели Intel Pentium 150, 166 и 200.

1997► Процессоры Intel Pentium MMX. Выпускаются с тактовыми частотами 166, 200 и 233 МГц.

1997► Процессоры Intel Pentium II Выпускаются с тактовыми частотами 233, 266 и 300 МГц.

1998► Процессоры Intel Pentium II с тактовыми частотами 350 и 400 МГц.

В дальнейшем развитие процессоров шло в направлении увеличения тактовой частоты, расширения набора команд, увеличения разрядности шин, уменьшения напряжения питания (до 1.5 В), увеличения числа транзисторов на кристалле (технология 0,18 мкм).

Процессор Pentium 4 (одна из последних новинок) является однокристальным по-настоящему. На одном кристалле размещено около 42 млн. транзисторов, выполненных по технологии с разрешением 0.18 мкм, частота ядра первых моделей составляла 1,4- 1,5 ГГц, разрядность шины данных 256бит (32 байта). Что обеспечивает пропускную способность 32х1,4=44,8 Гбайт в секунду.

Расширенная система команд ориентирована на задачи:

- редактирование видеоизображений;
- трёхмерная визуализация;
- обработка видеосигнала в качестве источника данных;
- связь с телевидением высокой частоты;
- Интернет-телефония.

РАЗДЕЛ 1 МИКРОПРОЦЕССОРНЫЙ МОДУЛЬ КОМПЛЕКТА БИС КР580

ЛЕКЦИЯ 1 ОСНОВНЫЕ ПОНЯТИЯ О МИКРОПРОЦЕССОРНЫХ УСТРОЙСТВАХ

1 Основные составляющие вычислительного процесса в микропроцессорных устройствах (МПУ).

2 Структура микропроцессоров, их свойства и показатели. Пример построения микроЭВМ].

Перед тем как рассматривать основы построения микропроцессорной техники, необходимо напомнить о составляющих, посредством которых осуществляется вычислительный процесс. Основной составляющей этого процесса является информация, которая передаётся в виде **сигналов**, а запоминается в виде **кодов**. Код, используемый при этом, называется двоичным. Его наименьшая единица- бит. В виде тезисов представим свойства информации и правила её обработки.

Информация обрабатывается группами, по 8 бит в каждой группе. Такая группа называется **байтом**.

Информация хранится группами байтов, которые называются **файлами**. Каждый файл имеет имя.

По имени файла можно установить, какой вид информации в нём хранится и как её код надо обрабатывать.

Обработка данных в компьютере производится с помощью специальной микросхемы, которая называется **микропроцессором или просто процессором.**

Внутри процессора данные содержатся в специальных ячейках, которые называются **регистрами**.

В регистрах процессора данные могут храниться:

- в виде отдельных байт;
- слов (8 бит);
- двойных слов (16 бит);
- учетверённых слов (групп из 32 взаимосвязанных бит).

ВАЖНО! Чем выше разрядность процессора, тем больше бит данных он обрабатывает за одну операцию.

Данные, участвующие в операции, называются **операндами**, которые могут находиться в регистрах процессора, храниться в оперативной памяти или поступать от внешних устройств.

Операция состоит в том, что к операндам применяется инструкция. **Инструкция** - это одна из команд процессора, которая записывается одним или несколькими байтами.

Каждый процессор имеет свой набор инструкций, с помощью которых происходит обмен данными между регистрами, между регистрами и оперативной памятью, а также преобразование данных в регистрах.

Полный список инструкций процессора называется **системой команд** процессора. Разные процессоры имеют разные системы команд.

Если процессоры относятся к одному семейству, то более новые модели могут выполнять инструкции своих предшественников. Это называется **совместимостью сверху вниз.**

Инструкции для работы с данными процессор получает от программ. **Программы, как и данные,** представлены байтами и хранятся на жёстком диске в виде файлов. Во время работы программы размещаются в оперативной памяти.

Выполнение одной инструкции происходит в несколько приёмов (этапов). Отдельные этапы работы процессора называются **тактами**.

Количество тактов инструкций, выполняемых процессором в единицу времени, называется **тактовой частотой,** которая измеряется в мегагерцах (МГц). Чем выше тактовая частота процессора, тем выше его производительность.

Обычно процессоры работают с целыми положительными числами. Для арифметических операций, требующих учёта знака числа, применяются специальные инструкции дополнительной двоичной арифметики. Действительные числа в компьютере

хранятся в десятибайтной форме. Для работы с действительными числами используется математический сопроцессор.

Разрядность процессора и его тактовая частота - это основные характеристики процессора, от которых зависит его производительность.

Микропроцессор- это сердце компьютера. На его основе строятся микропроцессорные системы.

Микропроцессор (МП) - это функционально законченное программно-управляемое устройство обработки информации, выполненное в виде одной или нескольких больших интегральных схем (БИС).

МП выполняет над информационными данными арифметические и логические операции и осуществляет программное управление вычислительным процессом.

МП в процессе обработки данных обращается к оперативной памяти с помощью шины адреса (ША), а данные получает по шине данных (ШД).

МикроЭВМ (микрокомпьютер) — это устройство обработки данных, содержащее один или несколько МП, БИС оперативного запоминающего устройства (ОЗУ), БИС постоянного запоминающего устройства (ПЗУ), БИС управления вводом/ выводом.

Особенность микроЭВМ состоит в том, что в ней нет периферийных устройств (дисплея, печатающих устройств, запоминающих устройств на жестких или гибких дисках).

Такие микроЭВМ применяются в качестве встраиваемых в различные станки, машины и используются в технологических процессах в качестве управляющих устройств.

Микроконтроллер - это микроЭВМ с небольшими вычислительными ресурсами и упрощённой системой команд, ориентированными не на производство вычислений, а на выполнение процедур логического управления различным оборудованием.

Структура МП, их свойства и показатели

На основе МП можно построить следующие системы: микроЭВМ контрольно-измерительных систем; микроЭВМ для управления технологическим процессом; контроллеры периферийных устройств, бытовых приборов, игровых автоматов и т. д. На структурной схеме (рис. 1.1) представлены функции МП.

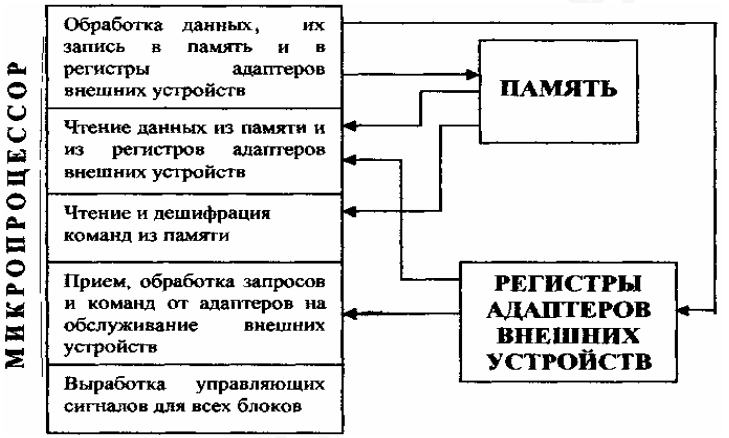


Рисунок 1.1- Структурная схема МП, его функции

К основным свойствам МП, как правило, относят:

универсальность применения, которая достигается разнообразием применения и обеспечивается программным управлением;

высокую производительность, которая обеспечивается использованием быстродействующих БИС и СИС (сверх больших ИС), а также специальных архитектурных решений: разнообразные способы адресации, конвейерная обработка данных, использование различных видов памяти (стековая, буферная, ОЗУ, ПЗУ, КЭШ-память), гибкая система команд;

технологичность — модульный принцип конструирования, т. е. все устройства реализуются в виде набора функционально законченных БИС, которые объединены в вычислительные устройства, машины.

Трудно найти сферу человеческой деятельности, где нельзя было бы использовать это замечательное изобретение человечества -микропроцессор.

Малые размеры и низкая стоимость микропроцессора позволяют легко устанавливать его во многие изделия, начиная от космического корабля и кончая бытовыми приборами. При этом включение микропроцессора в состав технического устройства совершенно преобразовывает действие последнего.

Рассмотрим единую типовую структуру (рис.1.2) микроЭВМ.

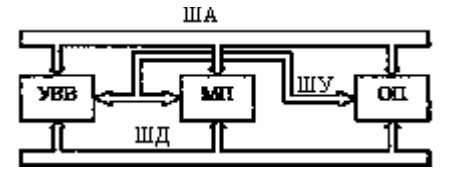


Рисунок 1.2- Типовая структура микроЭВМ

Типовая структура - это схема соединения модулей микроЭВМ при помощи многопроводных шин адреса (ША), данных (ШД) и управления (ШУ).

ША - однонаправленная шина. Выдачу сигналов осуществляет лишь МП или схема прямого доступа к памяти. Из-за малой нагрузочной способности ША требует применения буфера ША.

ШД формирует двунаправленный обмен данными между МП и памятью или внешними устройствами (ВУ). Нагрузочная способность ШД при этом невелика, что не позволяет непосредственно подключать к ней системы большой емкости памяти широким набором внешних устройств. Для увеличения

нагрузочной способности ШД необходимо использовать буфер ШД.

ШУ состоит из пяти линий, сигналы на которых формируются из записанного слова МП и его выходных сигналов ПРИЕМ и ЗАПИСЬ.

Показатели МП

Тип микроэлектронной технологии, используемой при изготовлении БИС МП.

Количество кристаллов, образующих МП.

Количество транзисторов на кристалле, размеры кристалла.

Количество выводов корпуса кристалла.

Длина слова (количество разрядов), обрабатываемого МП. Длина слова равна 8 разрядам (1 байт), двойное слово -16 разрядам (2 байта).

Быстродействие МП - это время выполнения команд основных операций, тактовая частота.

Емкость адресуемой памяти.

Параметры используемого сигнала.

Мощность рассеивания БИС МП.

Число входящих в МП набор дополнительных БИС и выполняемые ими функции.

Наличие и доступность для пользователя аппаратно-програмных средств.

Эффективность системы команд (количество команд, способы адресации, наличие команд работы со стеком, число уровней прерывания, возможность прямого доступа к памяти, пропускная способность интерфейса ввода/вывода).

Рассмотрим пример построения микроЭВМ

Архитектура простой микроЭВМ приведена на рис. 1.3. Микропроцессор является центром всех операций. Ему необходимы питание и тактовые импульсы. Генератор тактовых импульсов (ГТИ) может быть отдельным устройством или входить в состав кристалла МП. Типовой МП может содержать 16 адресных линий, которые составляют однонаправленную шину

адреса, а также 8 линий данных, которые определяют двунаправленную шину данных.

Архитектура микроЭВМ представляется двумя типами

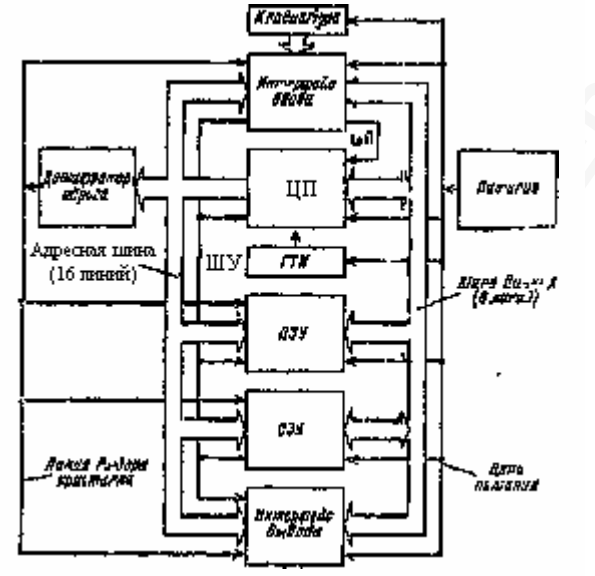


Рисунок 1.3- Архитектура микроЭВМ

полупроводниковой памяти. Постоянное запоминающее устройство (ПЗУ) - это постоянная память, которая содержит **программу-монитор**. ПЗУ содержит адресные входы, а также входы активизации чтения, выбора кристалла и тристабильные выходы, подсоединяемые на шины. ПЗУ, очевидно, имеет также сеть питания, которая на рис. 1.3 не показана.

Второй тип памяти представляет собой оперативное запоминающее устройство (ОЗУ)- устройство для временного размещения данных, которое имеет адресные входы, а также входы выбора кристалла и активизации чтения/записи. ОЗУ имеет 8 выходов с тремя состояниями, подсоединенными к шине данных. Имеется цепь питания ОЗУ.

Для сопряжения микропроцессора с устройствами ввода

информации имеется интерфейс ввода с клавиатуры. В задачу интерфейса входит размещение данных и управление их вводом с клавиатуры. В нужный момент интерфейс клавиатуры прерывает микропроцессор по специальной линии прерывания. Сигнал прерывания заставляет микропроцессор:

- закончить выполнение текущей программы;
- поддерживать свою работоспособность;
- перейти к выполнению специальной группы команд в своём мониторе, по которым ведётся управление вводом данных, исходящих с клавишного устройства.

Система интерфейса ввода с клавиатуры снабжена адресными входами, линиями выбора кристалла и команд активизации устройства.

Интерфейс вывода на семисегментные индикаторы служит для приёма данных, поступающих с шины данных, и управления индикаторами, на которых высвечиваются эти данные. Для активизации интерфейса вывода необходима информация с адресной шины, шины управления и шины данных.

Адресная линия содержит 65536 различных состояний 0 и 1 Для того чтобы активизировать требуемое устройство, дешифратор адреса считывает данные с адресной шины. Комбинационной логикой дешифратора адреса активизируется линия выбора кристалла, указывая на выбранное устройство.

МикроЭВМ можно классифицировать по их конфигурации таким образом:

-встраиваемые микроЭВМ, конструктивно приспособленные для встраивания в технологическое или другое оборудование (станок, прибор, машина и др.) в качестве сборной единицы и не имеющие индивидуального пульта управления, источника питания и декоративного конструктивного оформления;

- портативные сервисные микроЭВМ, содержащие небольшой дисплей, размещённые в небольшом чемоданчике;

- микроЭВМ настольной конструкции со встроенными в единую конструкцию дисплеем, пультом, устройством печати.

ЛЕКЦИЯ 2 МИКРОПРОЦЕССОРНЫЙ КОМПЛЕКТ КР580

1 Назначение, технические характеристики, состав микропроцессорного комплекта КР580, его модули.

2 Назначение выводов микропроцессора БИС КР580ВМ80А. его основные технические характеристики.

Микропроцессорный комплект (МПК) серии КР580 является самым первым образцом микропроцессорной техники, который был выпущен в начале 80-х годов в нашей стране. Зная его основы, можно изучить любой, даже самый сложный процессор. В табл.2,1 представлены технические характеристики МПК различных образцов, которые являются основными для любого микропроцессорного устройства. Следует заметить, что напряжение питания, а также технология изготовления образцов одинаковы, что является их достоинством при согласовании с ИС ТТЛ. Все остальные характеристики МПК не имеют идентичности, они достаточно разнообразны. Заслуживает внимания МПК К18IO, который имеет один мегабайт памяти и 16-ти разрядную ШД.

В состав базового комплекта серии КР580 входят следующие БИС:

- 8-зарядный параллельный центральный процессор КР580ВМ80А;
- программируемый параллельный интерфейс КР580ВВ55;
- программируемый последовательный интерфейс КР580ИК51;
- программируемый таймер КР580ВИ53;
- программируемый контроллер прямого доступа к памяти КР580ВТ57;
- программируемый контроллер прерываний КР580ВН59.

Выпишем основные характеристики МПК КР580:

- число БИС в МПК -19 шт.;
- разрядность данных - 8 бит;
- быстродействие- 2 мкс;
- число команд- 78 .

Таблица 2.1-Основные технические характеристики МПК

Обозначение серии	Число БИС в МПК	Обозначение МП	Разрядность, бит	Быстродействие, операций/с, (мкс)	Число команд	Емкость адресуемой памяти, Кбайт	Число РОН	Число уровней прерывания	Технология	Напряжение питания, В
К580	19	ВМ80А	8	(2,0)	78	64	6	1	n-МОП	±5; +12
К1801/ 109	14	ВМ1 ВМ2 ВМ3	16 16 16	$5\cdot10^5$ $1\cdot10^6$ $2\cdot10^6$	66 74 74	64 64 $4\cdot10^3$	8 8 8	4 2 4	n-МОП n-МОП n-МОП	+5 +5 +5
К1806	3	ВМ2	16	$5\cdot10^5$	77	64	8	2	КМОП	+5
К1808	9	ВМ1	8	$f_{т.и.}$=20ч100 кГц	-	-	-	-		+5
К1810	12	ВМ86 ВМ87 ВМ88 ВМ89	16 16 8 16; 16	(0,4) (0,5) (0,45) (2,4)	135 68 135 46	$1\cdot10^3$ $1\cdot10^3$ $1\cdot10^3$ $1\cdot10^3$	8 8 8 4	2 2 	n-МОП n-МОП n-МОП n-МОП	+5 +5 +5 +5
К1821	3	ВМ85А	8	(0,8)	80	64	6	2	КМОП	+5

- ёмкость адресуемой памяти- 64 Кбайт;
- число регистров общего назначения (РОН)- 6 ;
- число уровней прерывания-1;
- технология - n-МОП;
- напряжения питания - (+5, -5, +12В).

Следует отметить, что МПК КР580 является аналогом МП Intel 8080 (США), который был выпущен в 1972году.

Как ужё было сказано, МП обязательно должен работать совместно с памятью и устройствами ввода/вывода (УВВ) данных. При этом комплект, состоящий из МП, памяти и УВВ, называют микроЭВМ, а составляющие части этого комплекта - модулями.

Комбинируя тип модулей и их количество, можно создавать разнообразные микроЭВМ, начиная от простейших контроллеров для управления различными устройствами и кончая сложнейшими персональными компьютерами.

Комплект КР580 состоит из трёх модулей (рис. 2.1):

-микропроцессорного модуля;

-модуля памяти;

-модуля устройства ввода/вывода.

В микропроцессорном модуле (МПМ) реализуются:

-алгоритм обработки информации в соответствии с набором функций, выполняемых микропроцессорной системой;

-управление работой устройств системы осуществляется в соответствии с принципом программного управления. Как правило, в состав МПМ (табл.2.2) включают:

- центральный процессор;

-генератор тактовых импульсов (ГТИ);

-системный контроллер;

-шинные формирователи (это усилители мощности, посредством которых представляется возможность подключать к выходу работающей микросхемы n-го количества других нагрузочных элементов);

-буферный регистр (это силовой элемент, который также

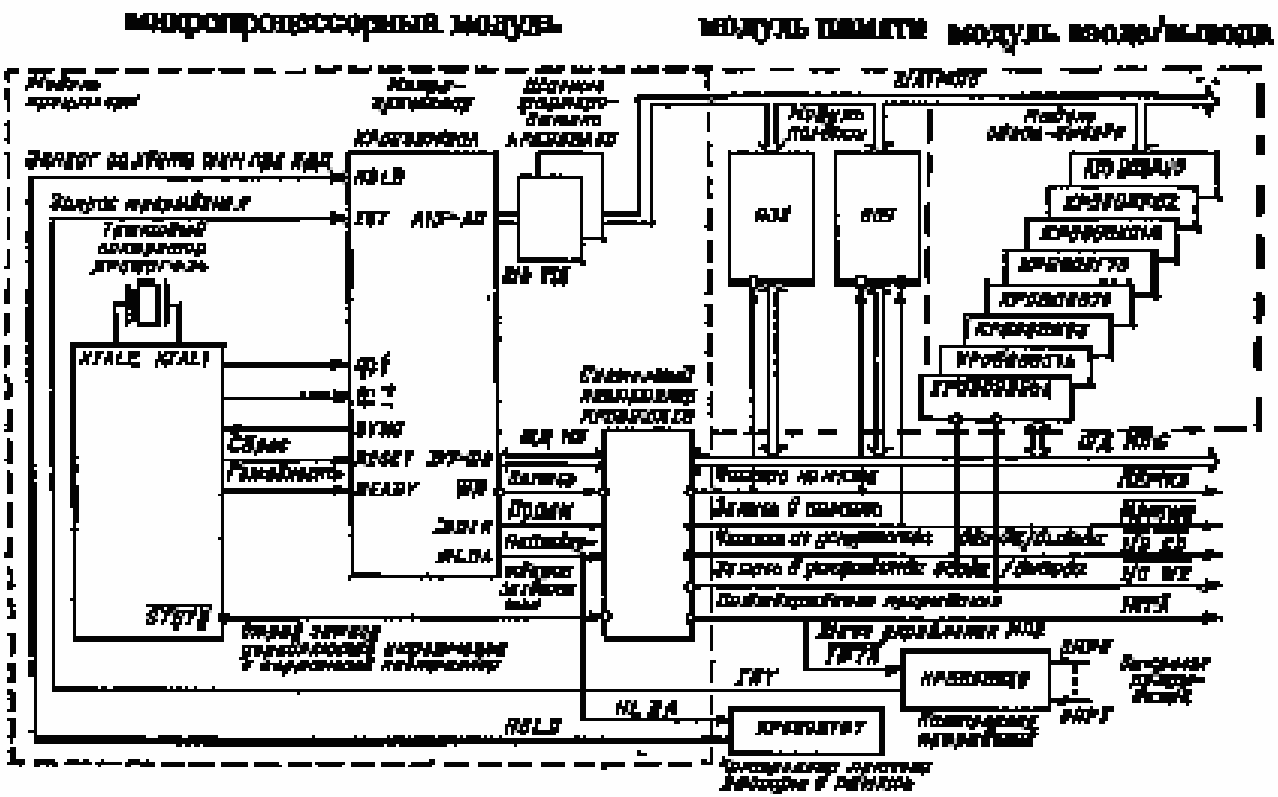


Рисунок 2.1 - Модульная организация МПС на базе МПК БИС 580-й серии

Таблица 2.2-Состав микропроцессорного модуля КР580

Обозначение	Назначение	n, бит	F_T МГц	$E_{пит}$,В
КР580ВМ80А	Центральный процессор	8	2,5	5, 12, 5
КР580 ГФ24	Генератор тактовых импульсов	-	27	12,5
КР580ВА86	Шинные формирователи	8	(30)	5
КР580ВК28	Системный контроллер	8	(60)	5
КР580ИР82	Буферный регистр	8	(45)	5

Таблица 2.3- Состав модуля ввода/вывода

Обозначение	Назначение	n,бит	F_T МГц	$E_{пит}$,В
КР580ВВ55	Программируемый параллельный интерфейс	8,4,1	2	5
КР580ВВ51	Программируемый последовательный интерфейс	8	2	5
КР580ВИ53	Программируемый таймер	8,16	2	5
КР580ВТ57	Программируемый контроллер прямого доступа к памяти	8,16	2	5
КР580ВН59	Программируемый контроллер приоритетных прерываний	8	2	5

является усилителем мощности и устанавливается для взаимосвязи между ШД, ША и устройствами ввода/ вывода).

Модуль памяти представляется набором микросхем, состоящих из запоминающих устройств (ЗУ), которые получили названия: оперативные запоминающие устройства (ОЗУ), являющиеся основным видом памяти; постоянные запоминающие устройства (ПЗУ), выполненные так, что данные в них записываются однократно при их изготовлении и в процессе эксплуатации данные можно только считывать.

Модуль устройства ввода/вывода предназначен для сопряжения микропроцессорного модуля с внешней средой. Он представляет собой совокупность каналов ввода/вывода, каждый из которых обслуживает отдельное внешнее устройство.

В состав модуля включены ряд микросхем, которые имеют свои функциональные особенности и получили названия (табл.2.3):

-программируемый параллельный интерфейс;

-программируемый последовательный интерфейс;

-программируемый контроллер приоритетных прерываний;

- программируемый контроллер прямого доступа к памяти;

-программируемый контроллер таймера.

Рассмотрим одну из основных микросхем микропроцессорного модуля, которая относится к классу больших микросхем (БИС)- КР580ВМ80А.

Назначение выводов микропроцессора БИС КР580ВМ80А

В связи с тем, что название «микропроцессор типа КР580ВМ80А» будет часто повторяться в тексте нашего дальнейшего изложения материала, условимся называть его «МП-К580».

МП-К580 (рис 2.2) является восьмиразрядным процессором, служащим для ввода, обработки и вывода

восьмиразрядных двоичных чисел, и имеет 10 управляющих выводов. К этим выводам подключается шина управления, по которой сигналы управления передаются из МП и в МП для согласования совместных действий с внешними устройствами, а также для изменения режимов их работы. Рекомендуем использовать рис.2.1.

Сигналы управления можно разделить на два вида:

-входные сигналы;

-выходные сигналы (формируются в МП и отправляются к другим устройствам).

Входные сигналы:

Ф1 и **Ф2** — это входы для подачи тактовых сигналов. Они определяют тактовую частоту работы МП.

СБР (RESET)— сброс (вход). Сюда поступают входные сигналы с ГТИ, которые сбрасывают счетчик команд в начальное положение, в результате чего МП начинает свою работу с обращения к ячейке памяти с нулевым адресом.

ГТ (READY) – готовность (вход). Сигнал «Готовность» информирует о готовности внешнего устройства (ВУ) к обмену информацией (обмен данными) с МП. Сигнал поступает с ГТИ. При наличии уровня «О» (т.е. при сигнале низкого уровня) на выходе ГТ МП останавливает свою работу и находится в состоянии «Ожидания» (WAIT).

ЗХ (HOLD) — захват (вход). Сюда поступает сигнал от ВУ о запросе разрешения на использование ША и ШД, подключённых к МП. Этот сигнал переводит МП в состояние «Захват», в котором ША и ШД переходят в 3-е состояние (высокое сопротивление). Шины ША и ШД отключаются от МП. Этот захват необходим для прямого доступа внешних устройств к памяти, минуя МП. В модуле ввода-вывода имеется контроллер прямого доступа к памяти (КПДП), с которого поступает сигнал на вход МП.

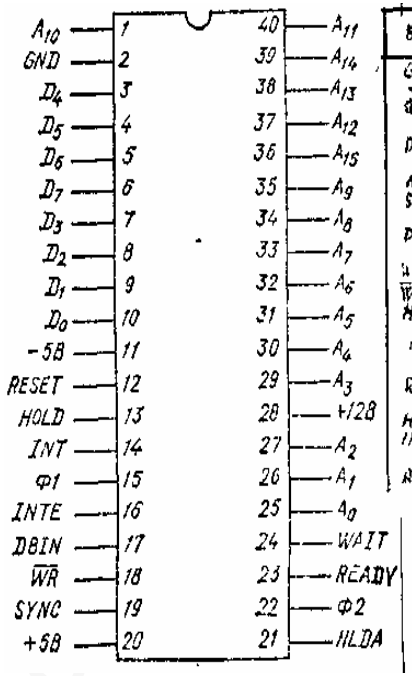
ЗПР (INT) -запрос прерывания (вход). Сюда поступает сигнал от внешних устройств (контроллер приоритетного

прерывания) с требованием о прерывании основной программы, которую выполняет МП, и перехода на выполнение подпрограмм обслуживания прерывания.

Вне очереди обслуживаются более важные и нужные программы ВУ. Если МП работает в режиме «Захвата» или в режиме «Ожидания», то МП находится в нулевом состоянии и сигнал ЗПР не воспринимается.

Выходные сигналы:

ПЗХ (HLDA) - подтверждение захвата шин (выход). На этот вывод МП выдаёт сигнал, разрешающий внешним устройствам использовать шины ШД и DJA, а сам отключается от этих шин. ПЗХ появляется в ответ на ЗХ.



Выводы	Назначение	Вход или выход
GND, +5 В, −5 В, +12 В	Питание	Входы
Ф1, Ф2	Тактовые импульсы	Входы
D_0—D_7	Линии данных	Двунаправленные
A_0—A_{15}	Линии адресов	Выходы
SYNC	Синхронизация	Выход
DBIN	Строб входных данных	Выход
WAIT	Ожидание	Выход
WR	Строб записи	Выход
HLDA	Подтверждение захвата	Выход
INTE	Разрешение прерывания	Выход
READY	Готовность ввода данных	Вход
HOLD	Захват	Вход
INT	Требование прерывания	Вход
RESET	Сброс	Вход

Рис. 2.2 - Микросхема БИС МП- К580 и её выводы

С (SYNC)—синхронизация (выход). Используется Mil для выдачи сигнала о начале нового машинного цикла работы и о том, что МП приступил к очередному этапу выполнения

команды. (Сигнал формируется в начале каждого машинного цикла).

ОЖ (WAIT)- ожидание (выход). МП извещает о том, что МП прервал выполнение своей программы и готов выполнить внеочередную программу внешнего устройства.

ПМ (DBIN) -приём (выход). МП выдаёт сигнал на системный контроллер. Этот управляющий сигнал указывает на то, что МП находится в режиме приёма, и может принять данные по ЩЦ из памяти или внешних устройств.

ЗП (WR)- запись (выход) или ВД (выдача). Этот сигнал появляется, когда МП может выдать данные по ШД в память или ВУ. Сигнал поступает на системный контроллер.

ПЗП (INTA) - подтверждение запроса прерывания (выход) формируется в ответ на принятый запрос прерывания, выполняет функцию сигнала RD в цикле подтверждения прерывания.

На все выводы управления МП выдаёт или получает от внешних устройств управляющие сигналы в виде высокого H-уровня или низкого L-уровня.

А0–А15 - адресная шина (это выходы с тремя состояниями), обеспечивает адресацию к любой из 256 восьмиразрядных ячеек памяти или внешнего устройства.

Д0–Д7 - двунаправленная шина данных, используется для обмена информации с памятью или ВУ.

Следует отметить, что основные технические характеристики МПК в основном зависят от характеристик микропроцессора.

ЛЕКЦИЯ 3 АРХИТЕКТУРА МИКРОПРОЦЕССОРА (МП) КР580 ВМ80А

1 Структурная схема МП, ее состав.
2 Арифметическо- логический блок.
3 Блок регистров, устройство управления.

1 Структурная схема МП, ее состав.

Архитектура МП - это его логическая организация, определяемая возможностями МП по аппаратной или программной реализации функций, необходимых для построения микроЭВМ.

Понятие **архитектуры** МП включает: его структуру, способы представления и форматы данных, набор операций, выполняемых МП, способы указания (адресации) данных, участвующих в операциях, форматы управляющих слов, поступающих в МП извне, характеристики и назначение вырабатываемых МП управляющих сигналов, реакцию МП на внешние сигналы.

Рассмотрим структурную схему МП (рис. 3.1). Первое, на что надо обратить внимание при рассмотрении данной схемы, - это наличие **внутренней шины данных**, посредством которой осуществляются логические и электрические связи между всеми элементами МП. Не обойти вниманием различные латинские буквы, которые являются обозначениями **регистров,** выполняющих основные операции приема, хранения, передачи данных.

В центре схемы - **вычислительное устройство** МП, обозначенное АЛБ (арифметическо- логический блок). Конечно, читатель внизу схемы обнаружит **устройство управления**, которое вырабатывает управляющие сигналы. Таким образом, структурную схему можно представить в виде трёх блоков: арифметическо- логического блока, блока регистров, устройства управления.

2. Арифметическо-логический блок

(рис.3.2) представляет собой:

-арифметическо- логическое устройство (АЛУ);

-регистр временного хранения;

- аккумулятор.

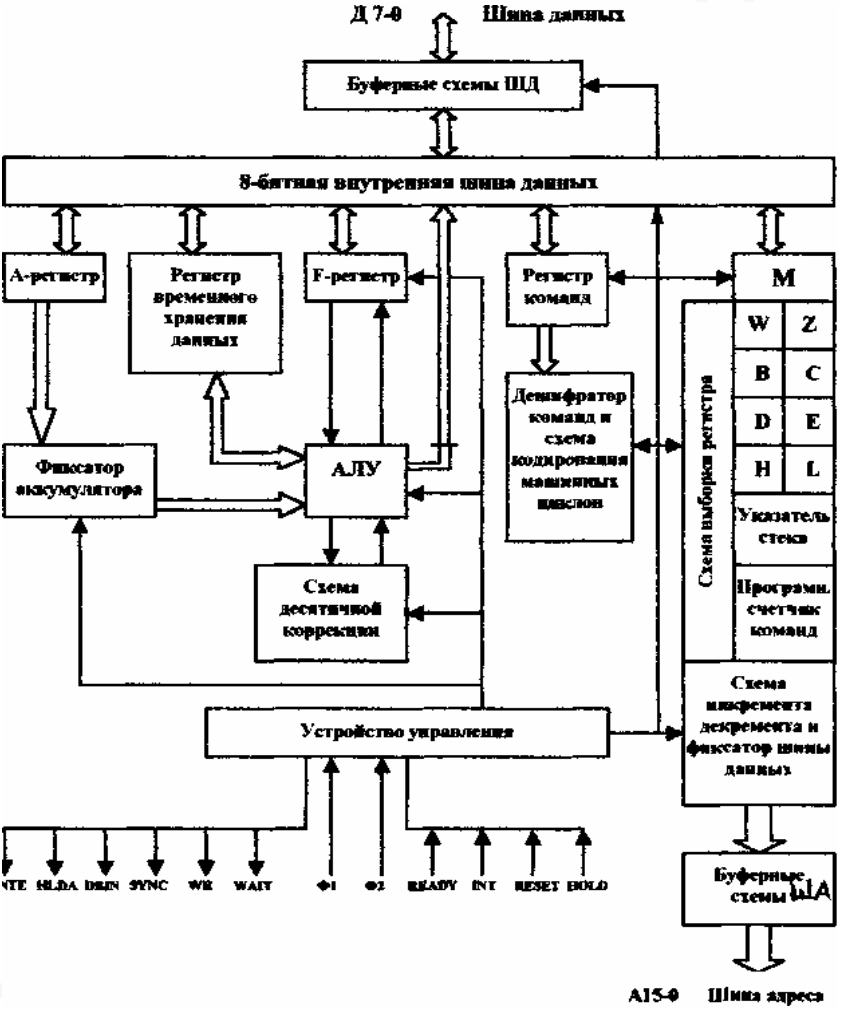


Рисунок 3.1 – Структурная схема МП

АЛУ - осуществляет операции по арифметической и логической обработке данных, операции сдвига. АЛУ состоит из сумматора, регистра сдвига, регистра состояния.

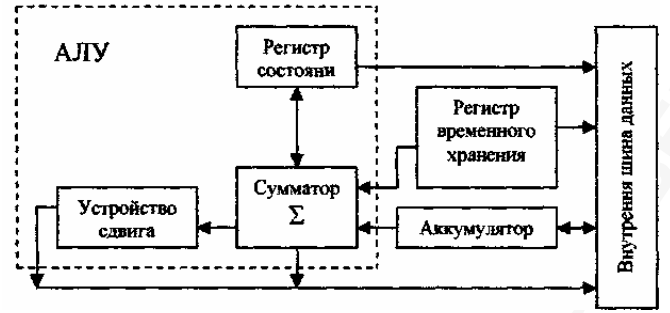


Рисунок 3.2- Арифметический логический блок

Арифметическими операциями являются: сложение, вычитание, сравнение, преобразование, увеличение, уменьшение числа на 1.

Более сложные операции (умножение, деление, вычисление элементарных функций и др.) выполняются по подпрограммам.

Логические операции - их 16. Основные - И, ИЛИ, НЕ и их комбинации.

При сложении двух чисел одно из них должно находиться в аккумуляторе, который является регистром. Другое слагаемое находиться в регистре временного хранения. Результат вычисления заносится в аккумулятор поверх одного из слагаемых. Например, рассмотрим сложение двух чисел 00001010 + 00000101. После операции сложения получим число 00001111, которое запишется в аккумулятор поверх числа 00001010 (рис.3.3).

Аккумулятор - это 8-разрядный регистр, здесь концентрируются результаты выполнения команд, его работа неразрывно связана с АЛУ.

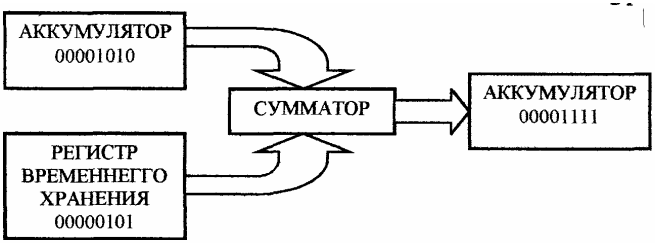


Рисунок 3.3- Сложение чисел в АЛУ

3. Блок регистров, устройство управления.

Блок регистров можно разбить на следующие группы по их функциональному назначению:

-регистры общего назначения (РОН);
-регистр признаков F(флагов);
-регистры временного хранения W, Z, T;
-счетчик команд;
-указатель стека;
-регистр команд.

Регистры B, C, D, E, H, L образуют группу регистров общего назначения (РОН) (рис.3.4).

Особенностью РОН является их доступность для пользователя. С помощью определённой команды можно поместить данные в любой регистр РОН или извлечь их из регистров, кроме того, переслать из одного регистра в другой.

Регистры РОН - 8-разрядные, но они допускают попарную работу, поэтому позволяют вводить и запоминать 16-разрядные числа (BC, DE, HL). При этом из пары регистров выделяют старший и младший разряды. РОН выполняют следующие функции:

-временно хранят данные, подлежащие обработке;
-осуществляют промежуточные вычисления;
-осуществляют счет числа циклов работы МП;
-хранят указатели адресов ячеек памяти двоичных чисел.

B	**C**
D	**E**
H	**L**
Старший разряд	Младший разряд

Рисунок 3.4- Регистры общего назначения

Регистр признаков F (флагов).

Процесс выполнения программы можно поставить в зависимость от значения результата выполнения предыдущей операции. Для обращения к информации о результатах вычислений в состав МП введен набор триггеров, которые устанавливаются в «1» или сбрасываются в «О» в зависимости от результата произведенных вычислений.

Каждый из триггеров хранит какой-то один бит условия, а в совокупности эти биты образуют регистр бит условий (рис.3.5). Программа может проверить значения четырех бит условий: переноса (CY), знака (S), нуля (Z), четности (P), используя одну из команд условного перехода, условного вызова подпрограммы или условного возврата из подпрограммы. Пятый бит - бит вспомогательного переноса (AC).

Бит переноса обычно используется для индикации переполнения при сложении или заема при вычитании, возникающих в седьмом разряде аккумулятора. Кроме того, команды, сдвигающие содержимое аккумулятора влево или вправо на один разряд, используют бит переноса как девятый бит аккумулятора. Например, при сложении двух однобайтных чисел AE_{16} и 74_{16}, возникает перенос «1» из седьмого разряда аккумулятора в бит переноса:

```
  10101110=AE
+ 01110100=74
--------------
CY= 1 00100010
```

Если при выполнении операции сложения осуществляется перенос из седьмого разряда аккумулятора, то бит переноса устанавливается в «1», если переноса нет, то бит переноса устанавливается в «0».

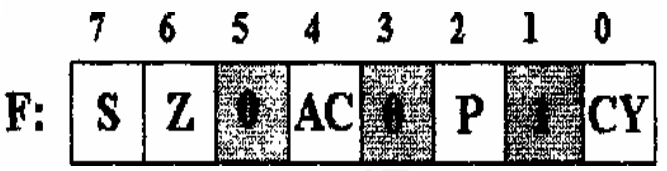


Рисунок 3.5- Регистр признаков (флагов)

Бит знака. Седьмой разряд результата выполнения операции в аккумуляторе может быть интерпретирован как знак результата. Команды, использующие бит знака, устанавливают его эквивалентным значению седьмого разряда аккумулятора. Ноль в седьмом разряде говорит о положительном значении результата, а единица — об отрицательном. Это значение дублируется в бит знака регистра бит условий и используется в командах условных переходов для реализации ветвления в зависимости от знака полученного результата.

Бит нуля. Ряд команд МП устанавливают бит нуля в «1», что соответствует равенству нулю всех бит аккумулятора. Если результат в аккумуляторе отличен от нуля, бит нуля будет сброшен в «0». Например, в результате сложения двух чисел $A7_{16}$ и 59_{16} будут установлены в «1» биты переноса и нуля:

```
  10100111=A7
+ 01011001=59
--------------
CY=1 00000000=0
 Z=1
```

Бит четности. Четность результата выполнения команды определяется подсчетом количества единиц в аккумуляторе после выполнения команды. Бит четности устанавливается в «1» для четного количества единиц и сбрасывается в «0» для нечетного их количества. Например, если результат вычислений в аккумуляторе представляет собой значение 01001110, то происходит установка в «1» бита четности в регистре бит условий. Обычно признак четности используют для контроля на четность данных в процессе их передачи.

Бит вспомогательного переноса указывает на наличие переноса из третьего разряда результата в аккумуляторе. Этот бит нельзя опросить в программе непосредственно и он используется лишь в команде DAA (десятичная настройка аккумулятора) для преобразования чисел из двоичной системы счисления в двоично-десятичную. Покажем, например, как бит дополнительного переноса и команда DAA позволяют выполнить сложение кодов 19_{10} и 8_{10} с получением двоично-десятичного результата. Значение 00011001_2 будет эквивалентно 19_{10}. Сложение 19_{10} с 8_{10} даст шестнадцатеричное число 21_{16}, и будет установлен бит вспомогательного переноса:

$$\begin{array}{r} 00011001 = 19_{10} \\ +\ 00001000 = 8_{10} \\ \hline AC=1\ \ 00100001 = 21_{16} \end{array}$$

Регистры временного хранения данных T, W, Z.

Это 8-разрядные регистры. Они недоступны программисту. Используются для запоминания двухбайтовых и трёхбайтовых команд перехода, передаваемых с внутренней ШД в счётчик команд.

Программный счётчик команд - это программно-доступный регистр и состоит из 16 двоичных разрядов и позволяет адресовать память объемом 2^{16} = 65536 бит. Это

средство, с помощью которого МП переходит от одной команды к другой при хранении их в памяти.

Счетчик команд содержит адрес байта команды, которая будет следующей считана из памяти (ПЗУ или ОЗУ) для выполнения. После считывания очередной команды счетчик команд увеличивает свое значение на 1, 2 или 3 в зависимости от длины команды. Этот процесс длится до тех пор, пока выполнение команд происходит по последовательной ветви команд.

Если текущая команда (перехода или вызова подпрограммы) изменит последовательность выполнения программы, МП занесет в счетчик команд не адрес следующей команды, а адрес команды, выполняемой в настоящий момент.

Счетчик команд доступен программно и всегда содержит 16-разрядный адрес. Он может быть сброшен устройством управления, инкрементирован, изменен командой перехода.

Указатель стека представляет собой 16- разрядный регистр и служит для адресации стековой памяти.

Стек — это группа последовательно пронумерованных регистров или ячеек памяти, снабжённых указателем стека, в котором автоматически при записи и считывании устраняется адрес последней занятой ячейки стека (вершина стека).

При записи в стек слово помещается в следующую по порядку свободную ячейку стека, а при считывании из стека извлекается последнее из него слово.

Стековая память является производственной зоной в ОЗУ и осуществляет режим обработки прерываний.

Действия МП при выполнении прерываний следующие:

-обслуживание подпрограммы прерывания;

-запоминание результата незавершённых операций;

-восстановление своёго состояния;

-доведение до конца прерванной операции.

Регистр команд — это 8-разрядный регистр, содержащий первый байт команды, принимает и хранит код очередной команды, адрес которой был установлен в программном

счетчике команд.

Устройство управления и синхронизации (УУС) выполняет следующие функции:

-получает сигналы с ШД и определяет природу выполняемой команды;

-передаёт сигналы во все устройства системы для координации выполнения команд;

-вырабатывает серию управляющих сигналов, которые передаются на узлы, блоки и элементы МП для координации их работы и согласования действий с внешними устройствами, например, с памятью.

Всё что происходит в МП, подчинено УУС.

На рис.3.1 показаны 10 управляющих выводов. К ним подключается шина управления, по которой сигналы передаются из МП и в МП для согласования совместных действий с внешними устройствами и изменения режима их работы.

Двунаправленная шина данных предназначена для организации связи между отдельными блоками микропроцессора, для связи с другими микросхемами и микроЭВМ. Она включает в себя внутреннюю шину данных и буфер данных (БД), соединенный с внешней шиной данных Д7–Д0. Двунаправленный с тремя состояниями БД состоит из буферного регистра, формирователей и предназначен для развязки внутренней и внешней шин данных (в процессе ввода или выполнения операций, не связанных с пересылкой данных, БД отключается).

Выбор регистра, участвующего в операции, осуществляется схемой **выбора регистра (СВР).**

Адресная логика обеспечивает выдачу на **адресную шину** адресов, команд и реализуется в виде **буферных схем**, которые включает в себя адресный буфер (АБ), буферный регистр адреса (БРА) и схему инкрементации и декрементации (СИД).

Адресный буфер представляет собой 16 выходных формирователей с тремя состояниями и предназначен для

выдачи адреса на выводы адресной шины A15–A0. Третье (отключающее) состояние позволяет подключать микропроцессор непосредственно к общей системной адресной шине микроЭВМ.

Буферный регистр адреса принимает и хранит адрес с любого 16-разрядного регистра. Его выход связан со входами адресного буфера и со входами схемы СИД.

Схема СИД - схема быстрого переноса/заема. С ее помощью содержимое БРА может быть передано с изменением на единицу или без изменения через 16-разрядный мультиплексор на вход любого 16-битового регистра (пары) BC, DE, HL, SP или PC.

Схема десятичной коррекции (СДК) предназначена для преобразования двоичного кода в двоично-десятичный при обработке двоично-десятичных чисел.

Отметим основные особенности организации МП К580:

-**трёхшинная** структура с шинами данных, адреса и управления;

-**магистральный принцип** связей, реализованный в виде связывающей основные узлы МП двунаправленной шины данных;

-наличие **регистровой памяти**, организованной регистрами общего назначения и специальными регистрами (программный счётчик команд, указатель стека, указатель данных), а также регистрами временного хранения;

-наличие средств организации **стековой памяти** (регистр-указатель стека, схемы выполнения операций инкрементация-декрементация, специальных команд стековых операций;

-наличие 16-разрядной шины **адреса**, обеспечивающей возможность прямой адресации любого байта в памяти ёмкостью 64Кбайт;

-наличие операций над **двухбайтными** словами;

-использование **трёх форматов команд** (однобайтного,

двухбайтного и трёхбайтного) и разнообразных способов адресации;

-реализация векторного многоуровневого **приоритетного прерывания** путём использования контроллера прерываний;

-реализация в МП режима **прямого доступа к памяти** путём подключения контроллера прямого доступа;

-наличие **эффективных** средств работы с подпрограммами и обработки запросов прерываний (стековая память, специальные команды вызова подпрограмм и возврата из подпрограмм).

Приведём структурную схему МП Intel 8080 (рис.3.6) для сравнения с МП К580.

Рассматривая структурную схему этого МП, можно сказать, что она аналогична структуре МП К580. Остаётся лишь добавить некоторые сведения, не указанные при описании МП К580, например, об аккумуляторе.

Аккумулятор (А) используется в качестве источника одного из операндов и места, где фиксируется результат операции, В команде А в явном виде не адресуется. На использование А в операции указывает код операции команды. Можно сказать, в отношении А применяется подразумеваемая адресация, что позволяет применять одноадресные команды, имеющие короткий формат. Чтобы А мог выполнять функции регистра операнда и регистра результата операции, он строится на основе двухступенчатого триггера. Использование А и РОН позволяет при выполнении команд уменьшить количество обращений к памяти и тем самым повысить быстродействие МП.

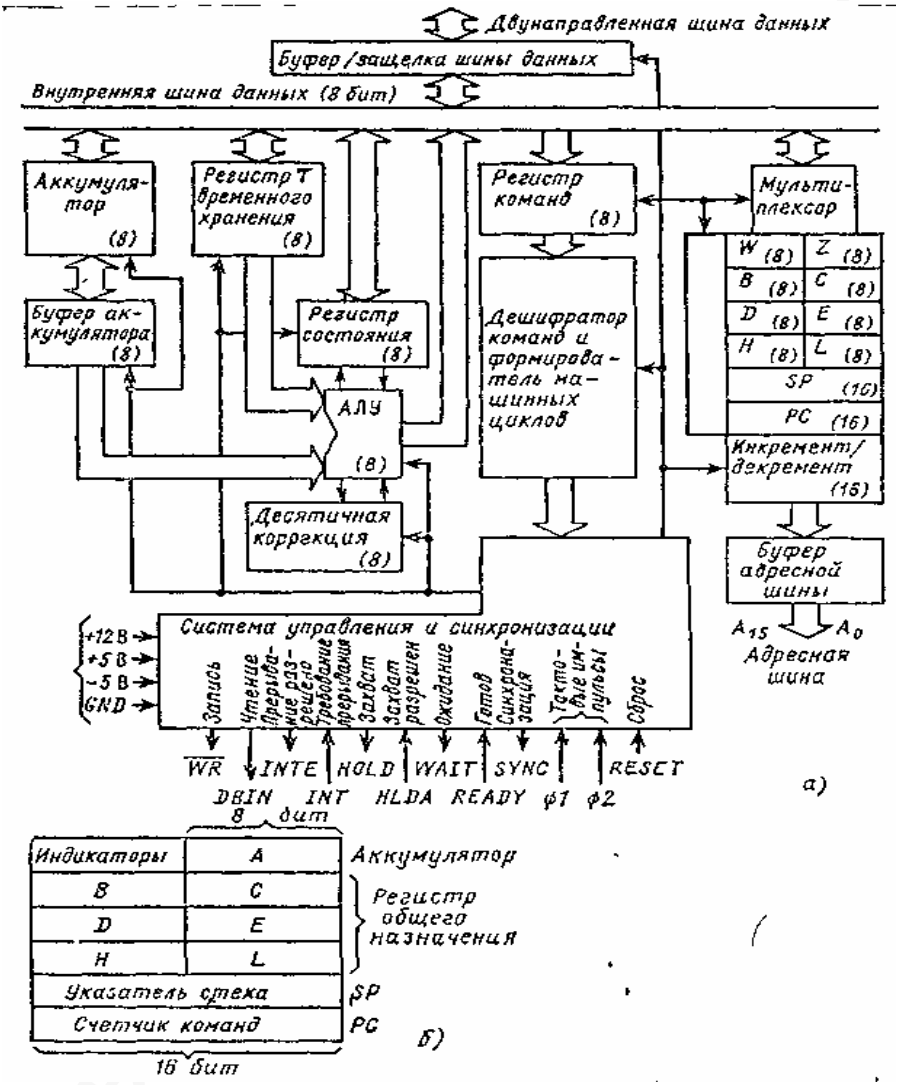


Рисунок 3.6-Структурная схема МП Intel 8080 (а), регистры, доступные программисту (б)

ЛЕКЦИЯ 4. ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ МИКРОПРОЦЕССОРА КР580ВМ80А

1. Форматы представления чисел в микропроцессоре, способы адресации памяти.
2 Програмистская модель, форматы команд, система команд микропроцессора.

1 Микропроцессор предназначен для работы с байтовой организацией памяти и имеет различные форматы представления чисел (рис.4.1).

В простейшем применении управляющих микропроцессорных систем используется способ представления целых чисел **без знака** в двоичном коде (рис.4.1 а) в диапазоне от 0 до 2^8-1, т.е. в МП могут использоваться только 256 различных чисел.

В МП, выполняющих операции сложения и вычитания, используется представление чисел **со знаком** (рис.4.1 б). При этом способе старший бит D7 отводится для знака числа (s = 1 - число отрицательное, s = 0- число положительное). Положительные числа представляются модулем, а отрицательные- дополнительным кодом. Диапазон изменения чисел от -2^7 до $+(2^7-1)$. Если используются двухбайтные числа, то знаковый разряд располагается и бите D7 старшего байта. Диапазон представления чисел расширяется от -2^{15} до $+2^{15}$-1.

При применении МП-систем, требующих выполнения операций над числами по **правилам десятичной арифметики**, используется десятичный двоично-кодированный упакованный формат (рис.4.1 в), в котором каждый байт условно разбивается на две тетрады, в каждой из которых кодируется десятичное число.

Если МП-система требует расширенного диапазона представления чисел, то в этом случае применяются многобайтные коды. Пример размещения в памяти МП- системы 32-битного целого числа со знаком представлен на рис.4.1 г. Обработка таких многобайтных чисел, расположенных в смежных ячейках,

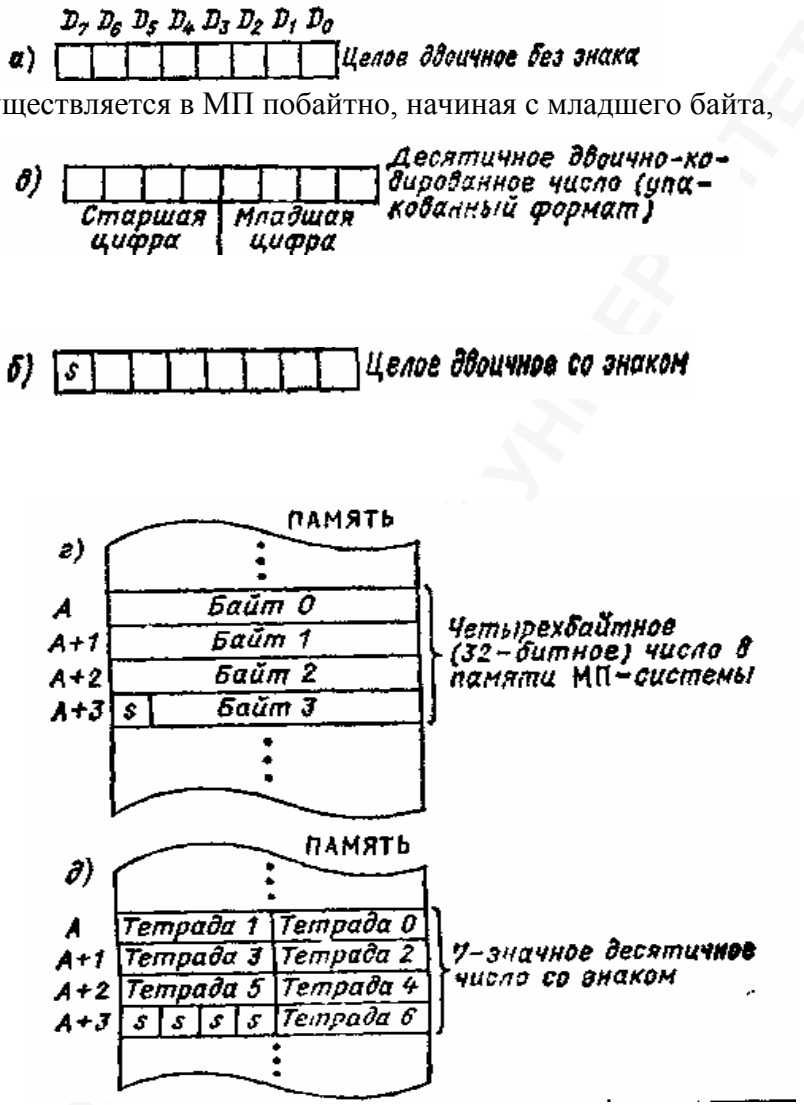


Рисунок 4.1 - Форматы представления чисел в микропроцессоре

путем вызова соответствующей подпрограммы и трёхкратной модификации (инкрементирования) указателя данных.

Если в МП-системе необходимо выполнить обработку **десятичных чисел со знаком**, то такие многобайтные числа представляются в десятичном дополнительном коде. При этом старшая тетрада старшего байта числа используется для кодирования знака числа, а остальные тетрады содержат двоичные десятичные цифры (рис.4.1 д).

Положительное число в знаковой тетраде кодируется четырьмя нулями, а для представления знака «минус» используется код 1001 (девятка). Цифровые тетрады положительного числа задают его модуль.

Для получения десятичного дополнительного кода отрицательного числа необходимо вычесть его модуль из 10^{n+1}, где п- число тетрад в модуле числа Это преобразование производят поэтапно: сначала модуль десятичного двоично-кодированного числа вычитается из 10^{n+1}-1 (код «все девятки»), а потом к результату добавляется 1 по правилам десятичной арифметики.

Для представления **дробных чисел** в МП может использоваться формат с плавающей точкой. Как правило, при этом два или три младших байта отводятся для представления мантиссы, а старший байт - для представления порядка числа и его знака.

Необходимо помнить, что использование в МП-системах многобайтных чисел приводит к резкому снижению производительности. Чем в большей степени формат обрабатываемых чисел превышает собственный формат МП (разрядность шины данных), тем значительнее снижается быстродействие обработки данных в МП-системе.

Арифметические операции выполняются по правилам двоичной арифметики над числами в дополнительном коде. При обработке двоично-десятичных чисел используется преобразование в двоичный код. Логические операции

выполняются по правилам двоичной логики. Результат операции представляется числом в соответствующем формате и байтом признаков результата (флагов) (рис.3. 5).

Форматы команд зависят от типа команд и способа адресации. Код операции представляется одним байтом. В случае прямой адресации к памяти используется адрес длиной 16 бит (т.е. максимальная емкость памяти 65536 байт). Длина команды может быть 1, 2 или 3 байта. Много-байтовые команды хранятся в соседних ячейках памяти и адресуются по первому байту, младший байт располагается по меньшему адресу. Формат команды определяется кодом операции. Адресация памяти в микропроцессоре КР580ВМ80А задается в коде операции. Используются следующие способы адресации.

Прямая - используется для адресации однобайтовых слов данных и двухбайтовых адресов, содержащихся в памяти или внешнем устройстве. Прямой адрес указывается во втором или во втором и третьем байтах команды.

Прямая регистровая - используется для адресации одно - и двухбайтовых слов, содержащихся во внутренних регистрах процессора, и указывается в байте кода операции.

Косвенная регистровая - используется для адресации байтов данных в памяти. Косвенный адрес содержится в паре регистров процессора (адресных указателях), указываемых в байте кода операции.

Непосредственная - используется в двух- или трех-байтовом формате команд. Байт 2 (или байты 2 и 3) непосредственно содержат данные (операнд или адрес, заносимый в регистр).

Стековая - используется для косвенной адресации двухбайтовых слов данных или адресов, находящихся в области памяти и отведенных под стек. Адрес определяется по содержимому регистра указателя стека (SP).

При изучении системы команд программистская модель любого МП даёт возможность лучше понять структуру МП.

Модель МП КР580 (рис.4.2) содержит только узлы, наиболее важные для программиста в процессе его работы.

Программно- доступные узлы (рис.4.2 б), адресуемые в командах в явной или неявной форме, на которые программист может воздействовать при программировании, содержат восьмиразрядные и шестнадцатиразрядные регистры. К первым из них относятся: регистр- аккумулятор, регистры общего назначения и регистр признаков, ко вторым - счётчик команд, указатель стека, регистр косвенного адреса H, L.

Программно-недоступные узлы (рис. 4.2 б) являются наиболее существенными для процесса выполнения команд. К ним относятся АЛУ, управляющее устройство (УУ), регистры временного хранения T, W, Z и регистр адреса памяти.

Системой команд называют перечень (список, набор) команд, который может выполнить данный микропроцессор. Обычно система команд оформляется в виде таблицы, указывающей действие каждой команды, её сокращённое название, двоичный код и другие данные. МП разных типов имеют свои несколько отличающиеся друг от друга системы команд.

В системе команд МП КР580 имеются однобайтные, двухбайтные и трёхбайтные команды. Форматы команд представлены на рис.4.3. Формат команды и тип адресации задаются в команде неявно кодом операции. **Адрес команды определяется адресом ее первого байта.**

В командах **условного перехода** трёхразрядный код CCC задаёт в трёхбайтной команде условие передачи управления по адресу, указанному в команде Возможны задания вариантов условия перехода: по наличию переноса, отсутствию переноса, нулевому, ненулевому, положительному, отрицательному, чётному, нечётному результатам. Систему команд МП можно разделить на группы команд в соответствии с их функциональным назначением. Выделим следующие **группы**

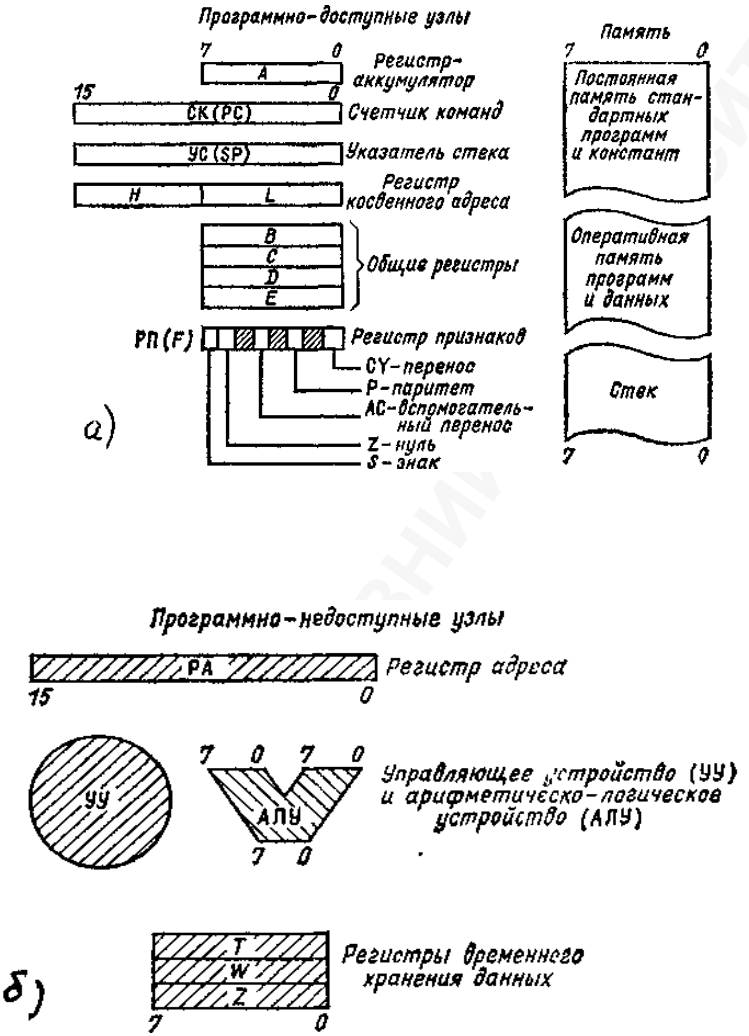


Рисунок 4.2- Программистская модель МПКР580

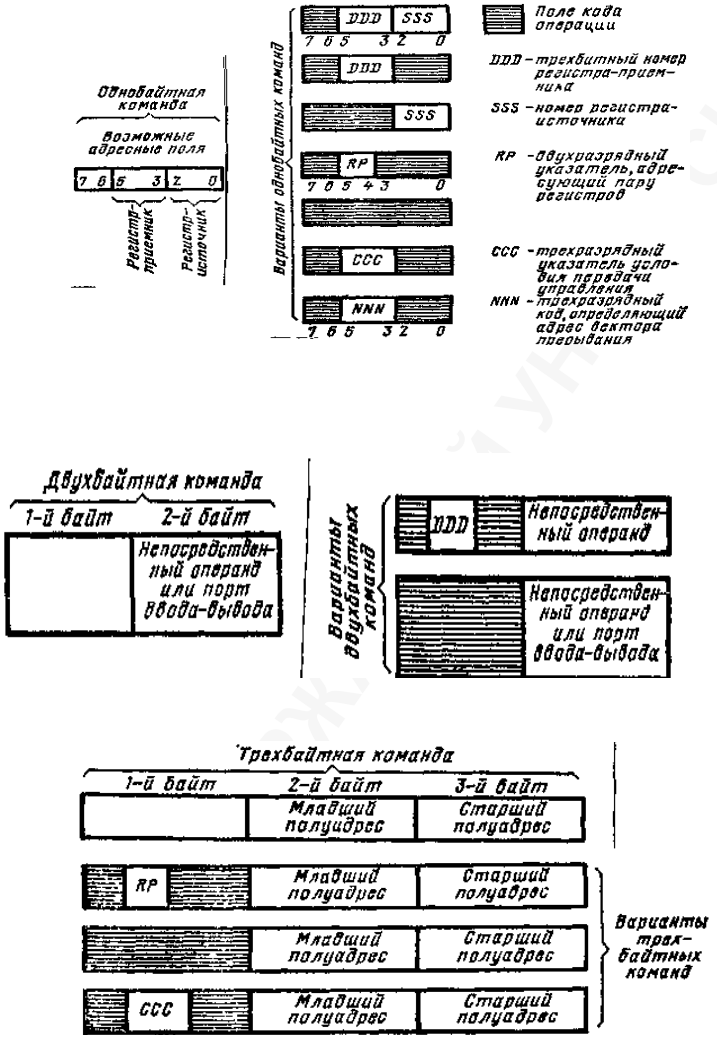


Рисунок 4.3- Форматы команд МП КР580

команд, пересылки информации, арифметических операций, логических операций, передачи управления, специальные команды. МП разных типов имеют свои несколько отличающиеся друг от друга системы команд. Для МП КР580 система команд приведена в приложении А

Команды **группы пересылок** осуществляют обмен данными между регистрами МП, ячейками памяти и устройствами ввода/вывода. Команды этого вида чаще других встречаются при составлении программ. Каждая пересылка данных всегда связана с тем местом, куда они пересылаются, а именно:с приёмником данных и тем местом, откуда эти данные поступают-источником данных. При написании операнда условились сначала указывать приёмник (место, куда помещаются данные), а затем источник (место, откуда поступают данные).

Команды **арифметических операций** выполняются только через аккумулятор. Притом один из операндов должен находиться в аккумуляторе ещё до выполнения команды. Таким образом, складывая два числа, необходимо одно из них поместить в аккумулятор, а второе можно загрузить в регистр R, в ячейку памяти М или записать во второй байт команды. Только после этого следует выполнять команду сложения.

После выполнения команды арифметической операции результат вычисления **автоматически** размещается в аккумуляторе, а число, которое там было, стирается.

МП КР580 выполняет лишь команды сложения и вычитания, а операции умножения и деления осуществляются по особым **подпрограммам** с помощью набора команд сложения, сдвига и других преобразований. К группе команд арифметических операций относятся также команды, уменьшающие или увеличивающие на единицу содержимое регистров РОН или ячеек памяти. МП КР580 может выполнять ряд арифметических вычислений с числами произвольной длины, с отрицательными и дробными числами.

Команды **логических операций** во многом похожи на ко-

манды арифметических операций. Они также оперируют с двумя восьмиразрядными числами, одно из которых должно находиться в аккумуляторе, а второе — в регистре R, в ячейке памяти M или во втором байте команды. Результат операции размещается в аккумуляторе.

МП КР580 выполняет следующие логические операции: И, ИЛИ, НЕ и ИСКЛЮЧАЮЩЕЕ ИЛИ. К группе логических команд относятся также операции сравнения двух чисел и сдвига числа, находящегося в аккумуляторе.

Логические действия выполняются над числами **поразрядно**, т.е. нулевой разряд одного числа логически сравнивается с нулевым разрядом другого числа, и результат сравнения располагается в нулевом разряде аккумулятора. Затем такое же сравнение производится над первыми разрядами чисел, потом над вторыми и т.д. до последних седьмых разрядов.

Команды передачи управления выполняются в зависимости от состояния триггеров регистра признаков. Известно, что программа работы МП представляет собой последовательность команд, расположенных одна за другой в ячейках памяти. Обычно выполнение программы начинается с нулевой ячейки и продолжается в соответствии с возрастающей последовательностью номеров (адреса) ячеек до тех пор, пока не будет выполнена поставленная задача. При этом программа может насчитывать сотни тысяч и даже миллионы команд. При этом встречаются группы одинаковых команд, выполняющих определённые, всегда одни и те же действия, например: умножение двух чисел или их деление, задержку выполнения команд на одно и то же время и т.д.

Чтобы не было повторения в программе таких команд, поступают следующим образом. Их записывают в виде короткого списка команд, называемого подпрограммой, и помещают на хранение в определённое место в памяти, известное программисту. Когда возникает необходимость выполнить подпрограмму, то пишут всего одну команду «Перейти к выполнению подпро-

граммы». Такой переход от программы к подпрограмме и составляет сущность операции перехода.

Специальные команды не производят каких-либо действий над данными. Команды не имеют операнда, имеют только код операции. Большинство из них являются служебными командами, изменяющими режим работы МП. В системе команд МП КР580 имеется две специальные команды, управляющие режимом прерывания. Команда EI - разрешает прерывание, а команда DI - запрещает.

Система команд МП КР580 (приложение А) представлена в виде таблицы, которая построена следующим образом:

-во второй колонке даётся название команды на русском языке;

-в третьей – десятой колонках приведен код команды, причём символы DDD, SSS – это 3 разрядные поля, адресующие один из регистров общего назначения; RR- разрядные поля, адресующие один из парных регистров; RSW – слово состояния программы, первый байт которой равен содержимому A, а второй- содержимому RS; NNN- двоичное представление команды RST; символ “+” означает установку и сброс флага условия; символ” –“ означает отсутствие влияния на флаг;

-колонки четырнадцать- восемнадцать отображают состояние триггеров **регистра признаков**.

Таким образом, каждому машинному коду команды ставится в соответствие мнемоническое название (мнемоника) команды.

Выполнение команд в микропроцессоре КР580

Каждая команда микропроцессора занимает командный цикл, состоящий из цикла выборки команды и цикла ее исполнения. Длительность цикла выборки команды различна для различных команд и зависит от длины команды (возможны от одного до трех обращений к памяти). С другой стороны, длительность цикла выполнения команды зависит от типа команды и способа адресации операндов. Например, для

выполнения команды ADD R требуется только одно обращение к памяти за самой командой, а операнд находится в регистре R процессора. Для выполнения команды ADD M требуется два обращения к памяти: за командой и за операндом, адресуемым парой HL. Таким образом, длительность командного цикла определяется числом обращений к памяти или внешнему устройству. Интервал времени на одно обращение к памяти или внешнему устройству определяется как **машинный цикл М**. Число машинных циклов в команде изменяется от одного для однобайтовых команд с регистровой адресацией до пяти для трехбайтовых сложных команд. Временные диаграммы и алгоритм работы МП КР580 представлены в приложении А.

Машинный цикл, в свою очередь, разбивается **на машинные такты Т**, в каждом из которых выполняется элементарное действие в процессоре - микрооперация. Число тактов в цикле определяется кодом команды и изменяется от трех до пяти. Длительность такта задается периодом синхроимпульсов F1, формируемых внешними цепями.

Для синхронизации процессора с памятью и внешними устройствами, имеющими меньшее быстродействие, для организации прямого доступа к памяти и останова процессора предусмотрены три специальных состояния: ОЖИДАНИЕ, ЗАХВАТ и ОСТАНОВ, длительность которых произвольная, но всегда кратная Т.

Последовательность действий в каждом машинном цикле определяет его тип. Имеется 10 типов машинных циклов:

- ВЫБОРКА - выборка из памяти байта кода операции команды;
- ЧТЕНИЕ ПАМЯТИ - выборка второго и третьего байтов команды и выборка операнда с косвенной адресацией;
- ЗАПИСЬ В ПАМЯТЬ - запись операндов или сохранение адресов при выполнении команд пересылок;
- ЧТЕНИЕ СТЕКА - обращение к стеку;
- ЗАПИСЬ В СТЕК - обращение к стеку;

• ВВОД - выполнение команды ВВ;
• ВЫВОД - выполнение команды ВВ;
• ПРЕРЫВАНИЕ - организация прерывания и останов процессора;
• ОСТАНОВ - при выполнении команды останова;
• ПРЕРЫВАНИЕ ПРИ ОСТАНОВЕ - если прерывание возникло после выполнения команды "останов".

В каждой команде циклы появляются в той или иной последовательности, например:

• ADDR M1 (выборка);
• ADD M Ml (выборка) - М2 (чтение памяти);
• IN <port> Ml (выборка) -М2 (чтение памяти) - М3 (ввод);
• CALL <адрес> Ml (выборка) - М2 (чтение памяти) – М3 (чтение памяти) -М4 (запись в стек) - М5 (запись в стек).

Каждый машинный цикл процессора идентифицируется байтом состояния, который выдается на шину данных в начале каждого машинного цикла и сопровождается выдачей сигнала синхронизации машинных циклов на вывод SYNC. Этот байт несет информацию о последующих действиях процессора и может быть использован для расширения функций управления в микропроцессорной системе.

Программное обеспечение для МП КР580 на языке Ассемблер приведено в приложении Б.

ЛЕКЦИЯ 5 ПРИНЦИП ДЕЙСТВИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

1 Основы работы микропроцессорных систем.

2 Пример программирования простой системы управления.

1 Мы выяснили, что **микропроцессор**-это устройство, предназначенное для обработки цифровой информации и управления процессом обработки. **Память** - это устройство, предназначенное для хранения информации, т.е. программ в закодированном виде и данных. К **внешним устройствам** относятся устройства ввода и вывода информации, а также внешние запоминающие устройства.

Информация в процессоре и памяти представляется в виде электрических сигналов в двоичной форме. Устройства ввода обеспечивают ввод данных и управляющих сигналов в удобной для человека форме и их преобразование в двоичный код. Устройства вывода обеспечивают обратное преобразование. В совокупности все эти устройства объединяются в микропроцессорную систему (МПС), которая может быть вычислительной или управляющей.

Принципы работы микропроцессорной системы были сформулированы ещё в 1937 году Дж. фон Нейманом, которые можно представить в виде вопросов и ответов на них.

Где находятся программы и данные? Программы и данные находятся в памяти Память- это совокупность ячеек. Каждой ячейке присваивается адрес, задание которого позволяет идентифицировать ячейку. Адреса присваиваются ячейкам в следующем порядке: 0,1,2 и т.д. Структура микропроцессорной системы представлена на рис.5.1.

Как выбирается требуемая ячейка памяти? Выбор требуемой ячейки памяти производится процессором путём передачи адреса в память по специальной шине, называемой шиной адреса. Эта шина является выходной по отношению к

МП. Доступная МП область адресов называется адресным пространством. Например, если адресная шина 16- разрядная, то можно сформировать 65536 ячеек памяти, т.е. адресное пространство составляет 64 Кб.

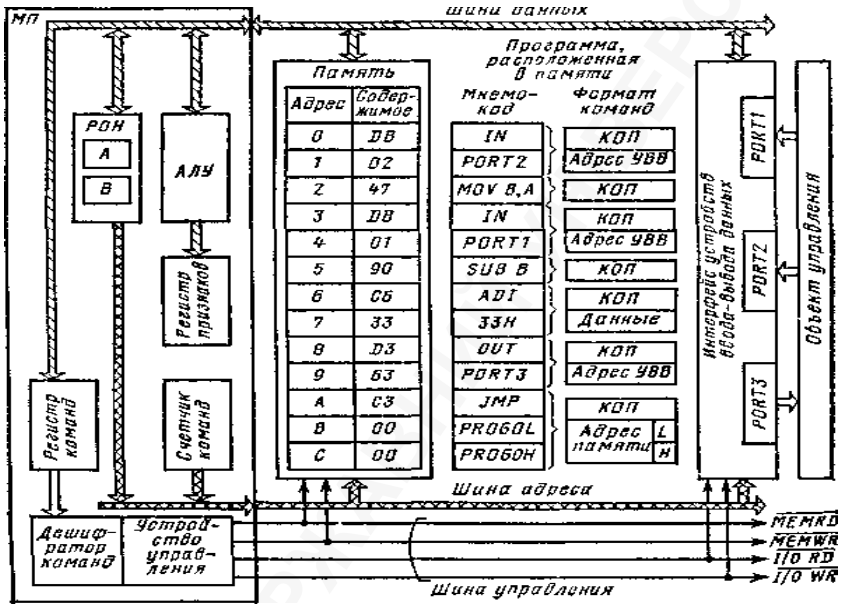


Рисунок 5.1 – Структура микропроцессорной системы

Каким образом информация поступает в микропроцессор? Информация хранится в памяти в виде одномерного массива слов. Слово состоит из нескольких бит (рис. 5.2) и может содержать 1, 2 или 4 байта информации. Разрядность слова в памяти и разрядность данных, обрабатываемых процессором, как правило одинаковы. Для передачи данных из памяти в процессор и обратно служит шина данных. Направление передачи данных определяется в большинстве случаев процессором. Передача данных снаружи внутрь процессора называется считыванием, обратный процесс - записью.

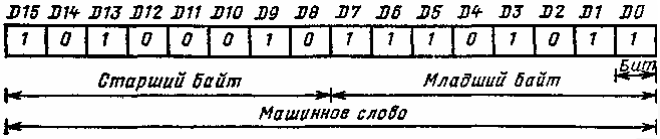


Рисунок 5.2- Машинное слово

Каким образом информация вводится в МПС и выводится из неё? В интерфейсе устройств ввода/вывода данных можно выделить отдельно адресные области, предназначенные для ввода и вывода данных (рис.5.1). Первая называется портом ввода, а вторая - портом вывода. Вводом данных называется процедура считывания информации из порта ввода, а выводом — процедура записи информации в порт вывода.

Для идентификации порта используется шина адреса. Номер порта, с которым предстоит обмен данными, выставляется процессором на шину адреса. Так как на практике число требуемых портов ввода или портов вывода вряд ли превысит 256, то для их адресации можно использовать однобайтный адрес (если шина адреса 16-разрядная, то только 8 старших или младших разрядов. Ввод/вывод данных осуществляется путем передачи информации по двунаправленной шине данных.

В чем суть магистрального принципа обмена информации МПС? Микропроцессор, память и интерфейс внешних устройств, соединенный между собой шиной данных и шиной адреса, обеспечивают три вида передачи данных:

-процессор <=> память;

-процессор <=> интерфейс устройств ввода/вывода;

-памятью <=> интерфейс устройств ввода/вывода.

В любой момент времени в операции обмена данными участвуют только два устройства. Вид передачи определяется процессором с помощью шины управления:

$\overline{\textbf{MEMRD}}$ - считывание данных из памяти;

MEMWR - запись данных в память;

$\overline{\text{I/O RD}}$ - ввод данных из устройств ввода/вывода (пор-

та ввода);

$\overline{\text{I/O WR}}$ - вывод данных в устройство ввода/вывода (порт вывода);

HLDA -разрешение прямого доступа в память со стороны УВВ для записи или считывания данных, минуя процессор.

Так, **для считывания данных из памяти** необходимо:
-выдать адрес ячейки памяти на шину адреса;
-сформировать сигнал «чтения памяти» MEMRD;
-принять данные с шины данных внутрь процессора.
Все эти операции автоматически выполняются процессором с помощью устройства управления, входящего в его состав (рис. 5.1).

Как обеспечивается строгий порядок выполнения программы? Работа компьютера состоит в последовательном извлечении команд и их выполнении. Так как явное различие между командами и данными, находящимися в памяти, отсутствует, то их идентификация производится на основе соглашений, принятых «по умолчанию».

Считается, что текущее содержимое специального регистра процессора, называемого счётчиком команд СК, указывает на адрес ячейки памяти, содержащей код очередной команды, подлежащей выполнению.

Процессор начинает свою работу с выдачи текущего содержимого счётчика команд на шину адреса и формирования сигнала MEMRD (чтение памяти). По умолчанию считанные данные интерпретируются процессором как код операции КОП и размещаются в регистр команд (РК).

После расшифровки код операции с помощью специального устройства - **дешифратора команд (ДШК)** устройство управления (УУ) формирует необходимую последовательность действий (микрокоманд) в соответствии с кодом операций, т.е. следует фаза выполнения команды.

Одновременно выполняется длина команды в байтах (сло-

вах), и содержимое счётчика команд увеличивается на эту длину.

Таким образом, СК всегда содержит адрес следующей команды. После завершения фазы выполнения команды содержимое СК вновь выставляется на шину адреса и цикл выборки-выполнения команды повторяется.

Как выполняются арифметические и логические команды? Арифметические и логические команды выполняются с помощью арифметико- логического устройства. При этом сначала на каждый из двух входов АЛУ (рис.5.3) поступают операнды (числа), участвующие в операции, а затем устройством управления на основе расшифровки кода команды, хранящегося в регистре команд, отдаётся «приказ» выполнить ту или иную операцию.

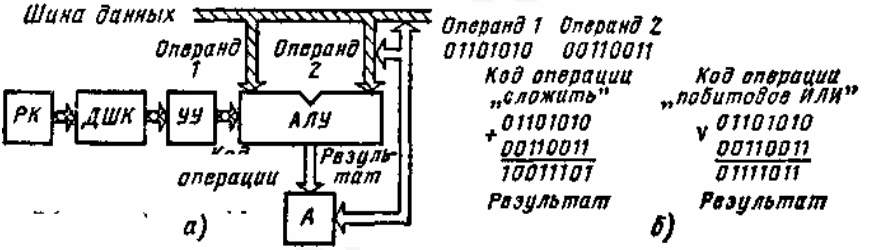


Рисунок 5.3 - Принцип действия АЛУ

Операнды поступают на входы АЛУ либо непосредственно с шины данных, либо из регистров общего назначения (РОН) - встроенной в микропроцессор сверхоперативной памяти, служащей для временного хранения информации. Результат операции в большинстве случаев записывается в специальный регистр, называемый аккумулятором (А). При сложении содержимого регистра А с содержимым регистра В результат операции замещает один из операндов в регистре А.

Таким образом, содержимое А используется и в качестве одного из операндов.

Как выполняются команды передачи управления в другую точку программы?

Последовательный характер выполнения программы может быть нарушен, если микропроцессор выполнит программу, которая запишет в счётчик команд новое значение. Такой командой может быть **команда безусловного перехода**, когда содержимое счётчика команд заменяется адресом, указанным в самой команде, либо **команда условного перехода**.

В последнем случае предварительно проанализируется один из признаков результата предыдущей операции, хранящейся в регистре признаков (флагов). Если указанное в команде условного перехода условие выполняется, то переход осуществляется, в противном случае управление передаётся следующей команде. Содержимое регистра признаков автоматически меняется после выполнения в АЛУ любой арифметической или логической операции.

Как выполняются команды передачи управления подпрограммам? Что такое стек?

Подпрограмма - это последовательность команд, которая может выполняться неоднократно в любом месте программы. Как только возникает потребность в выполнении действий, реализуемых в подпрограмме, последняя вызывается. Вызов подпрограммы осуществляется в основной программе с помощью команды вызова подпрограммы, например, CALL PPROG, где символическое имя PPROG соответствует начальному адресу размещения подпрограммы в памяти. **Вызов подпрограммы отличается от команды перехода**. Вначале текущее содержимое счётчика команд, которое представляет собой адрес следующей команды после команды вызова под программы, запоминается в специальной области памяти — **стеке. Эта операция называется запоминанием адреса возврата.** Далее содержимое счётчика команд заменяется адресом перехода и управление передаётся подпрограмме (рис.5.4). Последней командой подпрограммы является специальная команда возврата в основную программу RET. Она извлекает из стека адрес возврата и записывает его в счётчик команд.

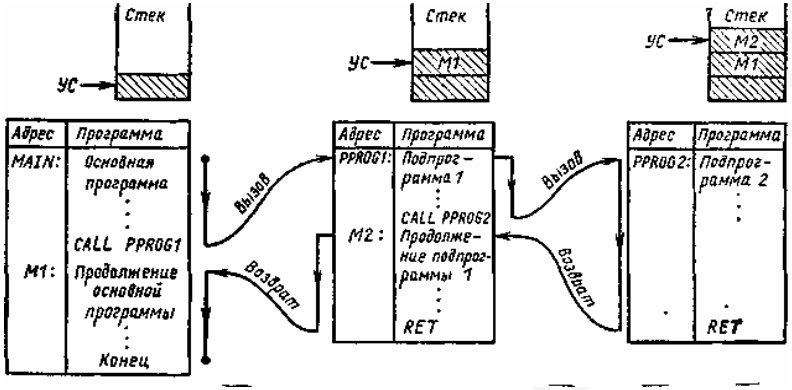


Рисунок 5.4 Использование стека для запоминания адресов возврата при обращении к подпрограммам

Тем самым осуществляется переход к следующей после команды вызова подпрограммы команде основной программы. За правильностью записи в извлечении информации из стека следит специальный регистр микропроцессора, называемый **указателем стека (УС).** Он всегда указывает на адрес последней заполненной данными ячейки памяти, относящейся к стеку.

При записи в стек адреса возврата УС предварительно декрементируется (его содержимое уменьшается на 1) для определения адреса свободной ячейки памяти и только потом производится запись. По команде возврата из ячейки памяти, адресуемой УС, производится сначала извлечение адреса возврата, а затем - инкрементирование УС.

Стек широко используется также для временного хранения информации. В любом микропроцессоре имеются команды записи данных в стек PUSH и извлечения из стека POP. При этом используется принцип «последний вошёл, первый вышел». Так, если в стек последовательно записаны числа А, Б, В, то при первом обращении к стеку считывается число В, при втором - Б и т.д. Указатель стека в этом случае автоматически

инкрементируется - декрементируется, всегда показывая на самый последний записанный элемент данных. Рассмотренный механизм применяется для сохранения в памяти текущего содержимого регистров процессора при обращении к подпрограммам, если подпрограммы используют в своей работе те же регистры, что и основная программа.

Что такое прерывание?

Часто возникает ситуация, когда внешнее устройство сигнализирует процессору о том, чтобы он остановил выполнение текущей программы и «уделил внимание» этому устройству. Например, в силовой цепи электропривода, управляемого от микропроцессорного контроллера, произошло короткое замыкание и сработал датчик максимально допустимого тока. Возникает необходимость немедленно **прервать программу управления** электроприводом в «штатном» режиме и перейти к выполнению специальной программы обслуживания аварийной ситуации- отключить питание, включить сигнализацию и т.д. Решить эту задачу можно двумя путями:

-первый- составить **управляющую программу** так, чтобы через определённые промежутки времени процессор проверял состояние датчика максимально допустимого тока и при наличии аварии передавал управление подпрограмме обслуживания аварийной ситуации. Это потребует включения одной и той же процедуры «тестирования аварии» во все фрагменты управляющей программы;

-второй - предусмотреть специальный вход (или входы) **запроса на прерывания** в самом процессоре и механизм обработки этих запросов - **механизм обслуживания прерываний**, когда появление запроса на прерывание будет вызывать останов текущей программы и передачу управления подпрограмме обслуживания прерывания. Второй путь не требует внесения каких -либо изменений в управляющую программу и принят во всех системах управления оборудованием от ЭВМ.

Обработка прерывания во многом подобна выполнению

подпрограммы. Если прерывание возникло, оно фиксируется процессором. После того как текущая команда выполнена до конца, процессор записывает адрес следующей команды (адрес возврата) в стек и переходит к выполнению подпрограммы обслуживания прерывания (рис. 5.5).

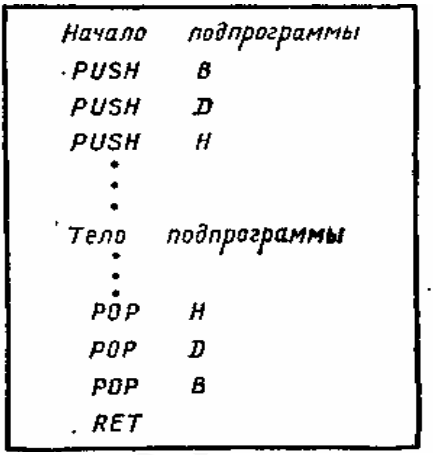


Рисунок5.5- Структура подпрограммы

Если по условиям работы объекта управления после завершения подпрограммы обслуживания прерывания возможен возврат в основную программу, то в конце программы обслуживания прерывания ставится команда RET, которая продолжает выполняться дальше, как будто ничего не произошло.

На каком языке можно написать программу для микропроцессорной системы? Реальная программа, выполняемая процессором, представляет собой последовательность двоичных кодов, хранящихся в памяти. Такую программу, которую в состоянии «понять» процессор, называют **программой на машинном языке или программой в машинном коде.** Так, для процессора К580ВМ80А код 1001 0000 соответствует команде

вычитания из содержимого регистра А (аккумулятора), содержимого регистра В (РОН) и записи результата в аккумулятор: (А)= =(А) - (В). Знак равенства используется в смысле «присвоить».

Хотя для ЭВМ **язык кодов** — прекрасный язык, он трудно воспринимается людьми. Поэтому программисты пишут программы либо на языках символического кодирования команд **-ассемблирования**, либо на языках высокого уровня (алгоритмических языках, например, Фортране, Паскале, С++ и др.).

При ассемблировании каждая машинная команда имеет аналог - **символьную цепочку**, вид которой у человека вызывает ассоциацию с тем действием, которое выполняет команда. Поясним это на указанной команде. Так, вместо приведённого выше кода 1001 0000 записывают SUB B (от английского слова subtract- вычитать), вместо кода 0000 0101 - DCR B, что означает уменьшить содержимое регистра В на 1 (от английского слова decrement- уменьшение). Естественно, что такие символические имена проще запоминать и использовать. Более того, поскольку каждой ассемблерной команде соответствует машинная команда, процесс перевода программы с символического языка на машинный можно поручить самой ЭВМ. Этот процесс называется **транслированием** и выполняется с помощью специальной программы, называемой **ассемблером**, которая преобразовывает текст программы, написанной на языке ассемблера, в программу на машинном языке. **Ассемблер** - машинно зависимый язык программирования, т.е. зависит от типа используемого процессора. Это очень важное обстоятельство, т.к. переход при проектировании на другой процессор потребует освоения и нового языка программирования. К счастью, если вы знаете один ассемблер, освоить другой — существенно более простая задача.

Ассемблер используется при разработке **программного обеспечения** для микропроцессорных контроллеров, в части программ, где обеспечивается связь между различными аппаратными средствами (ЦАП, АЦП и т.д.). Те части программ, которые связаны с расчётами, время которых некритично, лучше

писать на языке высокого уровня.

Рассмотрим пример программирования простой системы управления. Пусть микропроцессорная система управления формирует управляющее воздействие в соответствии с законом

$$U_у=(X_з-X_{факт})+51$$

и построена на базе микропроцессора К580ВМ80А, где код 51 формируется для компенсации предварительно измеренного возмущающего воздействия. Для ввода 8-разрядных целых двоичных чисел $X_з$ и $X_{факт}$ служат порты PORT1 и PORT2 соответственно. Управляющее воздействие выдаётся в порт PORT3 (рис. 5.1). Адреса портов - 0, 1,2 соответственно. На рис. 5.6 при- V веден алгоритм решения задачи, а справа от него — программа на ассемблере.

Алгоритм решения задачи следующий:

-значение переменной $X_{факт}$ считывается из порта 2 в аккумулятор с помощью команды ввода данных IN PORT2;

-для того чтобы освободить аккумулятор для приёма переменной $X_з$, ранее считанное значение $X_{факт}$ пересылается в регистр общего назначения В с помощью команды пересылки содержимого А в регистр В MOV B, A;

-в аккумулятор А считывается переменной $X_з$ с помощью команды ввода IN PORT1;

-из содержимого аккумулятора А ($X_з$) вычитается содержимое регистра В ($X_{факт}$) с помощью команды вычитания SUB B, результат записывается в аккумулятор;

-к содержимому аккумулятора прибавляется константа 51с помощью команды сложения с операндом, заданным непосредственно в команде ADI 51;

-рассчитанное значение управляющего воздействия выводится в порт 3 с помощью команды вывода OUT PORT3;

-осуществляется возврат к началу программы с помощью команды безусловного перехода по адресу размещения первой команды JMP PROG 0. Далее процесс повторяется.

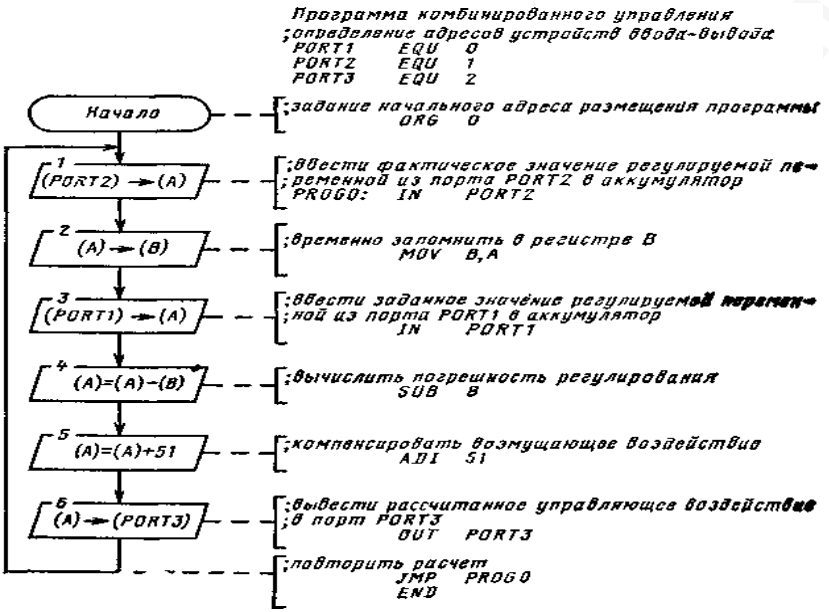


Рисунок 5.6 - Схема алгоритма и программа на ассемблере

Интерес представляет команда ADI33H, которая кроме КОП содержит данные (второй байт). Команда перехода занимает в памяти три байта: КОП, младший байт 16-разрядного адреса перехода и старший байт 16-разрядного адреса перехода. Можно сделать вывод, что разные команды имеют разную длину и, следовательно, разное время их выполнения.

Рассмотрим, как работает процессор, выполняя программу (рис. 5.6).

1. При подаче питания на микропроцессорную систему счётчик команд обнуляется.

2. Адрес 0 выставляется на шине адреса, вырабатывается сигнал считывания памяти MEMRD.

3. Код DB поступает в регистр команд. В процессе расшифровки команды определяется, что она двухбайтная, и счётчик команд получает приращение на 2 (указывает на следующую

команду).

4.Для считывания второго байта команды код, выставленный ранее на шину адреса, инкрементируется (увеличивается на 1 и поступает вновь на шину адреса.

5. По сигналу MEMRD второй байт команды 02 вводится в процессор и временно запоминается в одном из регистров общего назначения Z. На этом заканчивается фаза считывания команды и начинается **фаза её выполнения**.

6. Адрес порта из регистра Z поступает на шину адреса, формируется сигнал чтения УВВ FORD и текущее значение переменной $X_{факт}$ вводится а аккумулятор. **Первая команда выполнена.**

7. Микропроцессор переходит к следующему циклу считывания - **выполнения**.

8. Текущее содержимое счётчика команд (02) выдается на шину адреса, формируется сигнал чтения памяти MEMRD и новый КОП передаётся по шине данных в регистр команд. Его расшифровка даёт процессору информацию о том, что текущая команда - однобайтная. Поэтому содержимое счётчика команд увеличивается на 1. Следует фаза выполнения команды, и содержимое аккумулятора поступает на внутреннюю шину данных микропроцессора и далее - в регистр B. Выполнена вторая команда. Обучаемым предлагается продолжить описание работы микропроцессора по выполнению программы.

Программа, приведённая на рис. 5.6, содержит несколько специальных команд, так называемых **псевдокоманд**. Эти команды не выполняются микропроцессором. Их функция - дать необходимую справочную информацию программе-транслятору с языка ассемблер в машинный код. Приведём роль этих псевдокоманд в данной программе:

-EQU (от английского слова equate- равнять) определяет значение символического имени, которое затем будет использоваться в программе;

-PORT1 EQU 0 означает, что символическое имя PORT 1

в процессе трансляции будет автоматически замещено фактическим адресом 0, где бы оно ни встречалось.

-ORG (от английского слова origin- начало) указывает программе ассемблеру начальный адрес размещения программы в памяти.

-ORG 0 означает, что следующая за ней команда IN PORT 2 будет размещена, начиная с адреса 0.

Перед любой командой в программе может стоять **метка**, отделённая от неё двоеточием. Она имеет смысл текущего адреса размещения команды. В нашем случае метка PROG 0 получит значение 0.

Псевдокоманда END ставится в конце программы и даёт указание транслятору закончить процесс ассемблирования.

Так как в примере рассматривается встроенная система управления, то полученная программа в объектном коде подлежит записи в ПЗУ. Рассмотрим содержимое ПЗУ для нашей программы (рис. 5.1).

Адреса и коды - шестнадцатеричные. Справа даны символические имена этих кодов и форматы команд. Первая команда двухбайтовая. Она содержит код операции (КОП) - это первый байт и адрес порта ввода - второй байт. Адреса портов МП КР580 однобайтовые. Вторая команда - однобайтовая и содержит код операции и адреса регистров A и B.

ЛЕКЦИЯ 6 ЭЛЕКТРИЧЕСКАЯ СХЕМА МИКРОПРОЦЕССОРНОГО МОДУЛЯ

1 Структура простейшего микропроцессорного модуля.

2 Состав и принцип действия простейшего микропроцессорного модуля по электрической схеме.

Рассмотрим простейшую структурную схему (рис.6.1) микропроцессорного модуля и на её основе построим электрическую схему (рис. 6,2). Структурная схема состоит из микропроцессора (МП), генератора тактовых импульсов (ГТИ), шинных формирователей (ШФА, ШФД), устройства формирования управляющих сигналов (ФУ) и многоконтактного штыревого разъёма ХР, с помощью которого модуль подключается к разъёму общей шины.

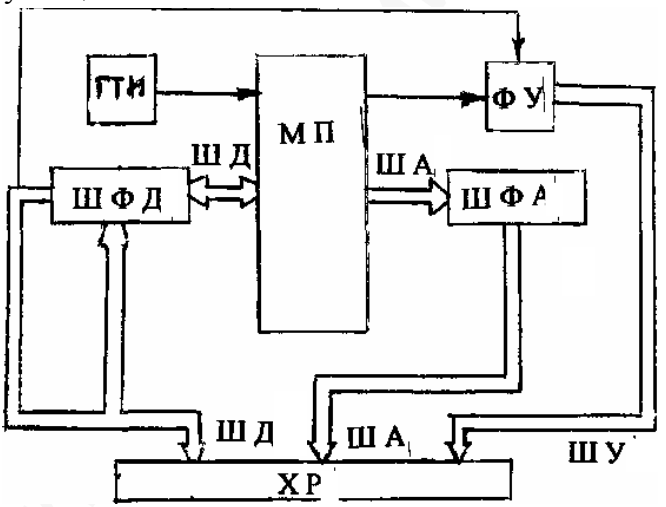


Рисунок 6.1- Структурная схема простейшего микропроцессорного модуля

Основное назначение микропроцессорного модуля - обеспечение соединений выводов МП с общей шиной.

Микропроцессор устроен так, что его выходные выводы (адресные, данных и управления) нельзя подключать

непосредственно к проводам общей шины, так как на всех выходах микропроцессора установлены маломощные транзисторы, которые не могут обеспечить нужную силу тока при значительной нагрузке, подключённой к общей шине.

Каждый выходной вывод МП-К580 можно подключать только к одному входному выводу микросхем типа ТТЛ (например, серии 155 или 133). Если же требуется подключение к нескольким входам ТТЛ, то приходится дополнительно устанавливать усилители мощности **(шинные формирователи)**.

Помимо этого микропроцессорный модуль осуществляет такие действия:

-создаёт тактовые сигналы Ф1 и Ф2 для микропроцессора;

-выдаёт управляющие сигналы ЧТЗУ, ЗПЗУ, ЧТВВ, ЗПВВ;

-отключает выводы адреса, данных и управления МП от общей шины при запросе внешним устройством использования (захвата) общей шины;

-обеспечивает двунаправленное действие шины данных при выполнении МП режима чтение/ запись;

-осуществляет включение в нужный момент времени сигналов ГТ и СВР, поступающих от внешних устройств на одноимённые выводы МП.

В соответствии со структурной схемой (рис.6.1) приведена принципиальная электрическая схема микропроцессорного модуля (рис. 6.2), в состав которой входят следующие микросхемы:

DD1 - К5 80ГФ24- генератор тактовых импульсов (ГТИ); DD2- КР580ВВ80А- микропроцессор;

DD3.1...DD3.5- К155ЛН1- логический элемент ИЛИ-НЕ;

DD4- К155ЛЛ1- логический элемент 4ИЛИ;

DD5- DD12- К589АШ6- шинные формирователи;

DD13- К155ТМ7- триггер;

DD14- К155ЛА3- логический элемент 4И-НЕ.

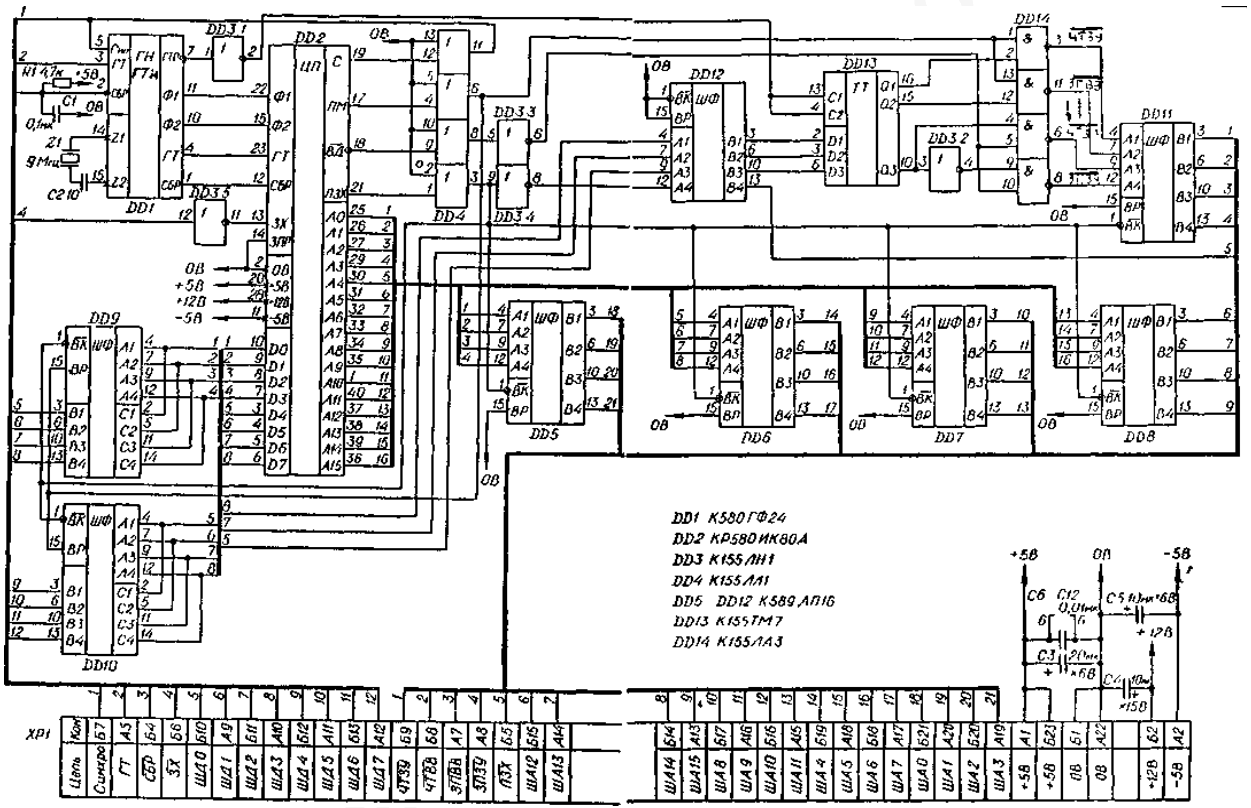


Рисунок 6.2- Принципиальная электрическая схема МП модуля

Поясним назначение вышеуказанных микросхем:

DD1 - генерирует сигналы Ф1,Ф2;

- обеспечивает выдачу сигналов ГТ (ГОТОВНОСТЬ) и СВР (СБРОС) в нужный момент времени.

DD2 -выполняет арифметические и логические операции над данными;

- осуществляет программное управление процессом обработки информации;

- организовывает взаимодействие всех устройств, входящих в систему.

DD3.1-DD3.5 - инвертируют сигналы управления. DD4- усиливает сигналы по мощности. DD5-DD12- усиливают сигналы по току;

- отключают выводы В1-В4 от общей шины при поступлении сигнала ПЗХ (ПОДТВЕРЖДЕНИЕ ЗАХВАТА) на управляющие выводы ВК (ВЫБОР КРИСТАЛЛА);

- обеспечивают двунаправленную передачу данных при осуществлении их записи или чтения.

DD13 - запоминает и выдаёт на выход управляющий сигнал при С1 = С2=1.

DD14 - формирует сигналы управления:

-ЧТЗУ (ЧТЕНИЕ ИЗ ПАМЯТИ);

-ЧТВВ (ЧТЕНИЕ ИЗ ПОРТА);

-ЗПВВ (ЗАПИСЬ В ПОРТ);

-ЗПЗУ (ЗАПИСЬ В ПАМЯТЬ).

Поясним последовательно работу каждого элемента принципиальной электрической схемы.

Образование тактовых сигналов Ф1, Ф2.

Для формирования этих сигналов в схеме используется генератор тактовых импульсов (ГТИ) типа К580ГФ24 (DD1).

К выводам (14 и 15 ГТИ) подключается кварцевый резонатор Z1. На выводах (Ф1, Ф2 ГТИ) формируются сигналы тактовой частоты FT = 1МГц. ГТИ выдаёт на выводы (23, 12

МП) сигналы ГТ и СБР.

Конденсатор С2 обеспечивает устойчивую работу кварцевого резонатора.

Цепочка R1,C1- служит для автоматического возврата в исходное состояние программного счетчика команд МП при каждом включении напряжения питания,

В схеме используются **шинные формирователи (DD5 - DD8, DD11 типа К580АП16**, каждый из которых обеспечивает подключение четырёх адресных выводов МП к адресной шине. Сигнал ПЗХ подаётся на вывод (1, ВК) всех пяти ШФ через микросхему DD4.

DD4 - усиливает сигнал ПЗХ по мощности, что позволяет к выводу 3 DD4 подключать уже 7 входов (DD5-DD10) и один вход DD3.4.

На выводы (15, ВР DD5-DD8, DD11) подаётся "О" (ОВ-общий вывод), поэтому DD5-DD8, DD11 не изменяют направление передачи сигнала (выводы А1-А4 будут соединяться с В1-В4).

Рассмотрим схему соединения выводов данных DO- D7 микропроцессора с разъёмом ХР1. Эти соединения выполняются при помощи двух **шинных формирователей (DD9 - DD10) типа К589АП16**, которые выполняют тройную роль:

-усиливают сигналы по мощности, снятые с выводов DO-D7МН;

-отключают выводы В1-В4 от общей шины, когда на ВК поступает сигнал ПЗХ;

-обеспечивают двунаправленную передачу данных при чтении и записи.

Рассмотрим, как **формируются режимы чтения и записи.**

Таблица слово состояний микропроцессора К580

Ном. опер.	Сообщение о предстоящем действии МП К580 или его состоянии	Логические уровни на выводах данных МП К580							
		D0	D1	D2	D3	D4	D5	D6	D7
1	Начало первого машинного цикла. Будет выбираться первый байт команды	0	1	0	0	0	1	0	1
2	Будет выполняться чтение из памяти (ЧТЗУ)	0	1	0	0	0	0	0	1
3	Будет выполняться чтение из порта (ЧТВВ)	0	1	0	0	0	0	1	0
4	Будет выполняться чтение из стека	0	1	1	0	0	0	0	1
5	Будет выполняться запись в память (ЗПВВ)	0	0	0	0	0	0	0	0
6	Будет выполняться запись в порт (ЗПВВ)	0	0	0	0	1	0	0	0
7	Будет выполняться запись в стек	0	0	1	0	0	0	0	0
8	Подтверждение режима прерывания	1	1	0	0	0	1	0	0
9	Подтверждение режима останова	0	1	0	1	0	0	0	1
10	Подтверждение прерывания при останове	1	1	0	1	0	1	0	0

Режим чтения. Когда МП принимает данные, на выводе 17 - (ПМ - ПРИЁМ) образуется уровень логической единицы.

Срабатывает усилитель DD4, сигнал поступает на вывод (15, ВР) DD9-DD10. Под воздействием этой единицы выводы (B1-B4 DD9-DD10) соединяются с (C1-C4 DD9-DD10) и данные поступают из общей шины в МП для чтения.

Режим записи. На выводе (17, ПМ) образуется ноль, срабатывает DD4, сигнал поступает на вывод (15, ВР DD9-DD10), выводы (A1-A4 DD9-DD10) соединятся с выводами (B1-B4 DD9-DD10) и данные из МП подаются на общую шину для записи в память или устройство вывода.

Формирование сигналов ЧТЗУ, ЗПЗУ, ЧТВВ, ЗПВВ осуществляется с помощью специальных БИС КР 580ВК28 и КР 580ВК38.

Формирование этих сигналов осуществляется путём использования **сигналов состояния**, их восемь и называются они **словосостояниями** (табл.6.1).

Для формирования сигналов ЧТЗУ, ЗПЗУ, ЧТВВ, ЗПВВ используем номера 2, 3, 5, 6 словосостояния таблицы, а в этих словах - 4,6,7 разряды, т.е. выводы (D4, D6, D7 DD2).

Сигналы (D4, D6, D7 DD2) нельзя непосредственно подавать на входы (Dl, D2, D3 DD13), т.к. ко входам (DO-D7 DD2) разрешено подключать только один вход микросхемы типа ТТЛ, а к ним уже подключены DD9 и DD10. Поэтому необходим усилитель сигналов состояния DD12. Выводы (D4, D6, D7 DD2) соединены со входами (A1-A3 DD12), а сигнал с выходов (B1-B3 DD12) передаётся на входы (D1-D3 DD13). DDB-это три триггера. На выходе DD13 появляются сигналы Ql, Q2, Q3, когда C1= C2= 1, формируемые (DD1 СТР):

Q1-сообщает о чтении данных из памяти (ЧТЗУ);

Q2-сообщает о чтении данных из порта (ЧТВВ);

Q3-сообщает о записи данных в порт (ЗПВВ).

Необходимо использовать сигналы ПМ (ПРИЁМ) и ВД (ВЫДАЧА) DD2, передаваемые на DD14. На выходах DD14

формируются сигналы ЧТЗУ, ЗПВВ, ЧТВВ, ЗПЗУ.

Эти сигналы поступают на (А1 А4 DD11), а с его выходов на общую шину. Управляющие выводы (ВК и ВР DD11) соединены с (ВК и ВР DD5-DD10), что позволяет с приходом сигнала ПЗХ отключать от общей шины выводы данных и адреса МП, ЧТЗУ, ЗПВВ, ЧТВВ, ЗПЗУ.

Сигнал ПЗХ управляет шинными формирователями и другими узлами микроЭВМ (контакт Б5 разъёма ХР1 общей шины).

Из общей шины (Б6 разъёма ХР1 общей шины) приходит сигнал (ЗАХВАТ), запрашивающий у МП разрешения на захват шин Для инвертирования сигнала используется DD3.5. Другим способом рассмотрения принципа действия МП являются временные диаграммы (рис. 6,3), в которых рассматривается поведение МП в каждом такте.

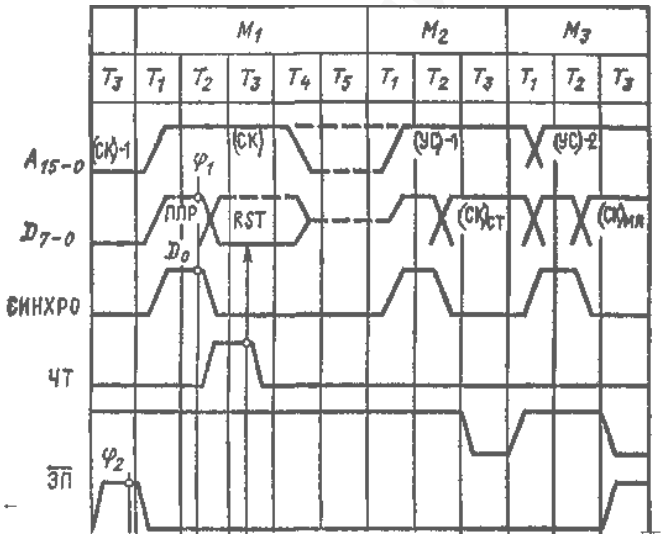


Рисунок 3 - Временная диаграмма работы МП

ЛЕКЦИЯ 7 ВСПОМОГАТЕЛЬНЫЕ ИНТЕГРАЛЬНЫЕ МИКРОСХЕМЫ

1. Шинные формирователи, буферные регистры, их структура, использование в МПС.
2. Системный контроллер: назначение, структура, действие.

1 При работе МП совместно с памятью возникает проблема соединения шины данных с памятью (рис.7.1). В режиме записи, когда процессор желает разместить в память какие-то данные, шину данных приходится подключать ко входам ячеек памяти, а при чтении, когда надо получать данные,- к выходам. Подключение должно выполняться автоматически и с большой скоростью. Решение этой проблемы осуществляется с помощью шинных **формирователей (ШФ).**

На рис.7.2 приведена учебная схема ШФ, где представлены три группы восьмиразрядных выводов: КО-К7, ЛО-Л7, МО-М7 и два управляющих вывода ВР (Выбор режима) и $\overline{ВК}$ (Выбор кристалла). К выводам КО-К7 подключаются выходы ячеек памяти, к выводам ЛО-Л7- входы ячеек памяти, а к выводам МО-М7 подключается шина данных.

Работа схема на рис.7.2 определяется состоянием управляющих сигналов:

-если оба управляющих сигнала ВР и $\overline{ВК}$ равны логической единице (рис.7.2 а), то ШФ переходит в **высокоимпедансное** состояние т.е. все выводы ЛО-Л7 и МО-М7 отключаются от шины данных и выводов памяти. В литературе такое состояние называют **третьим состоянием**, считая первым- наличие логической единицы, вторым- наличие логического нуля;

-если управляющие сигналы ВР и $\overline{ВК}$ равны нулю, то выводы КО-К7 соединяются с МО-М7 и выходы ячеек памяти соединятся с шиной данных микропроцессора, реализуется режим чтения (рис. 7.2 б);

-при значениях управляющих сигналов, равных ВР = 1 и $\overline{ВК}$ =0, шина данных микропроцессора соединяется с входами ячеек памяти, осуществляется режим записи (рис. 7.2 в).

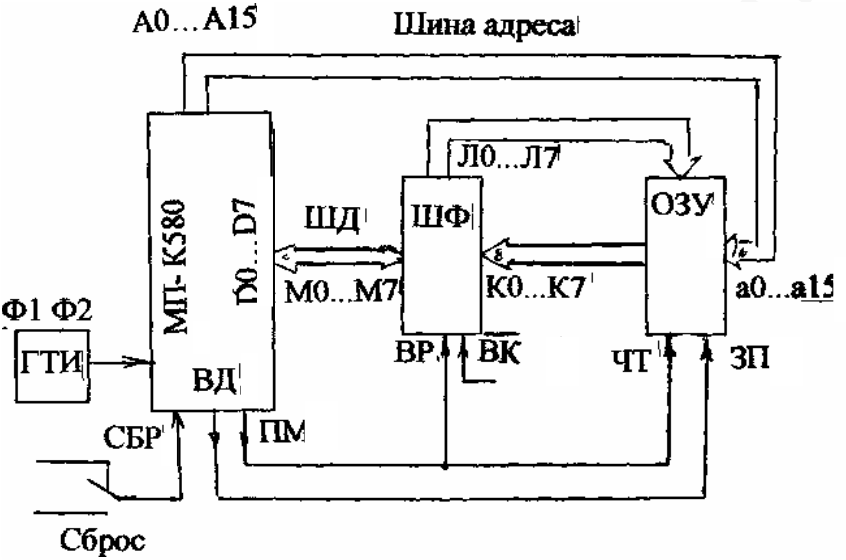


Рисунок 7.1- Структурная схема соединения ШФ с памятью

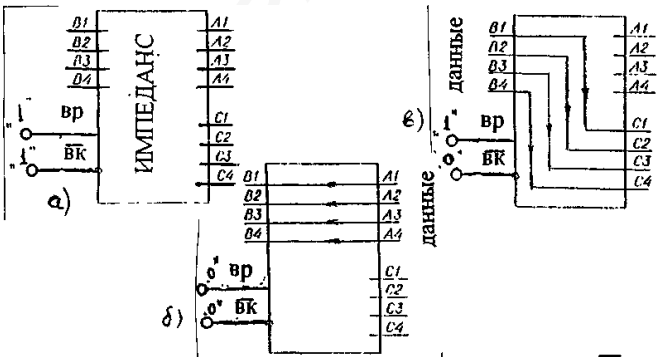


Рисунок 7.2- Учебная схема ШФ: а) управляющие сигналы $\overline{ВК}$ =1 и ВР=1; б) управляющие сигналы $\overline{ВК}$ =0 и ВР=0; в) управляющие сигналы $\overline{ВК}$ =0 и ВР=1

Принцип построения схем, отключающихся от нагрузки, используемых не только в ШД, но и в портах, и в шинных буферах МП, поясним на примере буферного усилителя транзисторно-транзисторной логики (ТТЛ) (рис.7.3), которая часто называется **схемой с тремя состояниями**, основой её является магистральный усилитель. Действие схемы можно пояснить следующим образом:

-при нулевом сигнале на управляющем входе V усилитель передаёт входную переменную X на выход У с инверсией;

-при V=1 транзистор Т2, включённый параллельно с транзистором Т2, откроется, шунтируя транзистор Т2, и формирует в точке К потенциал, близкий к нулю. В результате этого выходные транзисторы Т3 и Т4 окажутся в режиме отсечки. Выход У перейдёт в высокоимпедансное состояние (не 0, не 1), т.е. электрическое сопротивление выхода будет равно бесконечности, и схема отключается от нагрузки под воздействием управляющего сигнала V.

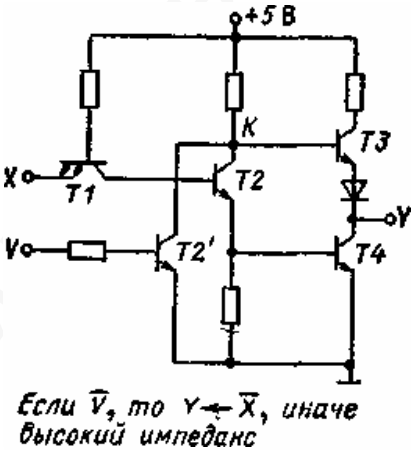


Рисунок 7.3 - Принципиальная электрическая схема магистрального усилителя ТТЛ с тремя состояниями выходного сигнала.

Возможные способы использования ШФ для подключения к системной шине процессора, памяти и устройств ввода/вывода (УВВ) представлены на рис.7.4.

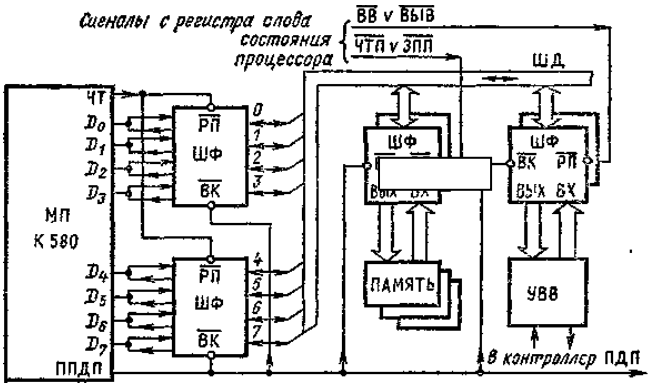


Рисунок 7.4 - Организация интерфейса микропроцессора, памяти и УВВ с использованием ШФ.

В микропроцессорном комплекте КР580 используются восьмиразрядные ШФ КР580ВА86 и КР580ВА87. Они отличаются большой выходной мощностью и простотой управления, что позволяет использовать их для построения двунаправленных согласующих буферов либо простых усилительных каскадов.

Формирователь состоит из восьми одинаковых функциональных блоков с общими сигналами управления Т и ОЕ. Функциональные блоки состоят из двух усилителей- формирователей с высокоимпедансными состояниями на выходах, схема включения которых обеспечивает разнонаправленную передачу. Условное графическое обозначение ШФ представлено на рис. 7.5. Шинный формирователь ВА86 не инвертирует данные, ВА87-инвертирует.

Назначение выводов ШФ

A7-A0- вход/выход линий данных. В зависимости от состояния входа Т они могут быть входными, если на Т-сигнал высокого уровня, и выходными, если на Т- сигнал низкого уровня.

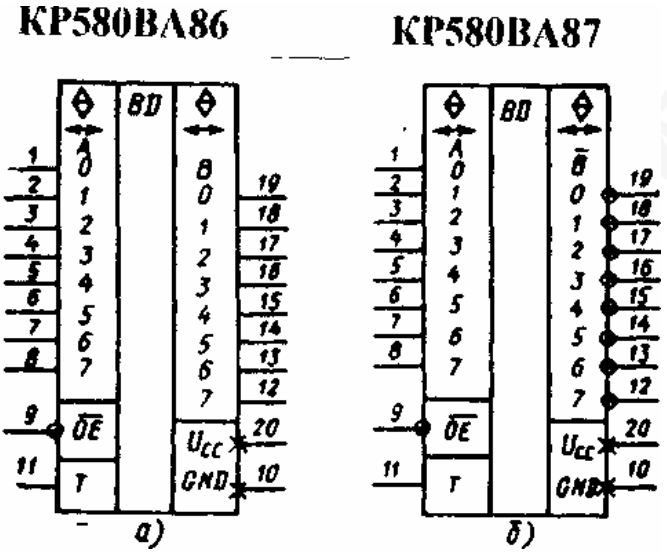


Рисунок 7.5 - Условное графическое обозначение ШФ

B7-B0 - вход/выход линий данных. Они являются входными, если на Т- сигнал низкого уровня, и выходными, если на Т- сигнал высокого уровня.

Т - входной сигнал управления направлением передачи. При Т= 0 осуществляется передача от В к А (режим В → А). При Т= 1 осуществляется передача от А к В (режим А ← В). Сигнал Т выбирает один из двух усилителей- формирователей, разрешая соответствующую передачу.

$\overline{OE}$ - входной сигнал разрешения передачи. При ОЕ= О снимается высокоимпедансное состояние с выхода усилителя-формирователя, выбранного по входу Т. Информация о выводах ШФ преставлена в табл.7.1, а функциональная схема - на рис. 7.6.

Дадим краткую оценку ШФ.

Это восьмиразрядная микросхема, она не имеет внутренней памяти, представляет собой двунаправленный приёмопередатчик с тремя стабильными состояниями, имеет

высокую нагрузочную способность (величина токов нагрузки может достигать до 32 mA), что позволяет увеличить нагрузочную способность МП. Имеет два канала, которые являются двунаправленными, передача информации возможна при поступлении сигнала Т.

Таблица 7.1- Выводы ШФ

Вывод	Обозначение	Тип вывода	Функциональное назначение выводов
1-8	*A0-A7*	Вход/Выход	Информационная шина
9	$\overline{OE}$	Вход	Разрешение передачи (управление 3-м состоянием)
10	*GND*	-	Общий
11	*T*	Вход	Выбор направления передачи
12-19	*B7-B0* ($\overline{B7} - \overline{B0}$ *для КР580ВА87)*	Вход/Выход	Информационная шина
20	U_{CC}	-	Напряжение питания 5В±5%

К основным достоинствам ШФ можно отнести большой выходной ток при малом входном токе и отсутствие шума на выходе при переключениях.

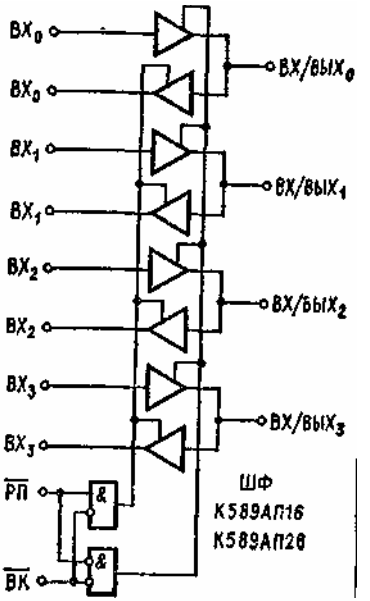


Рисунок 7.6 - Функциональная схема ШФ

Буферные регистры (на рис.3.1-буферные схемы ШД и ША) используются для организации запоминающих буферов, адресных защёлок, портов ввода/вывода, мультиплексоров и т.д. Они состоят из восьми триггеров с выходными схемами с тремя состояниями (рис.7.7), имеющими общие сигналы записи информации STB и управления выходными схемами $\overline{OE}$.

Буферные регистры **в шине данных** двунаправленные, предназначены для логического и электрического разделения внутри процессорной шины данных и внешней, системной шины ШД. Наличие в МП буферных регистров, отключающихся от общей системной шины, обеспечивают магистральный принцип межмодульных связей в МПС.

Работу буферного регистра **в шине данных** можно представить следующим образом. В **режиме ввода информации** внутренняя ТТТД подсоединяется к регистру-защёлке буфера, загрузку которого из внешней шины производит буферная схема под управлением команды. В **режиме вывода информации** буферная схема передаёт в ШД содержимое буферного регистра-защёлки, на вход которого по внутренней шине с одного из регистров (чаще всего из аккумулятора) загружен код, подлежащий выдаче. Во время выполнения операций, не связанных с обменом информации по отношению к МП, буферная схема отключается от ШД, т.е. переходит в высокоимпедансное состояние.

Буферные регистры ША - однонаправленные, обеспечивают передачу адресов команд и данных, а также номеров периферийных устройств от МП в систему. Выход буфера адреса, точно так же, как и буфера данных, может переходить в отключённое состояние. Такой режим необходимо иметь в МПС, в которой к памяти могут обращаться по системной шине адреса не только МП, но и некоторые из периферийных устройств, например, контроллер прямого доступа к памяти и др.

В комплекте КР580 используются восьмиразрядные буферные регистры КР580ИР82 и КР580ИР83, функциональные схемы которых представлены на рис.7.7, где триггеры (Т) и

выходные схемы (SW) обведены штрих-пунктирными линиями. Выходные схемы - это буферные усилители с тремя состояниями.

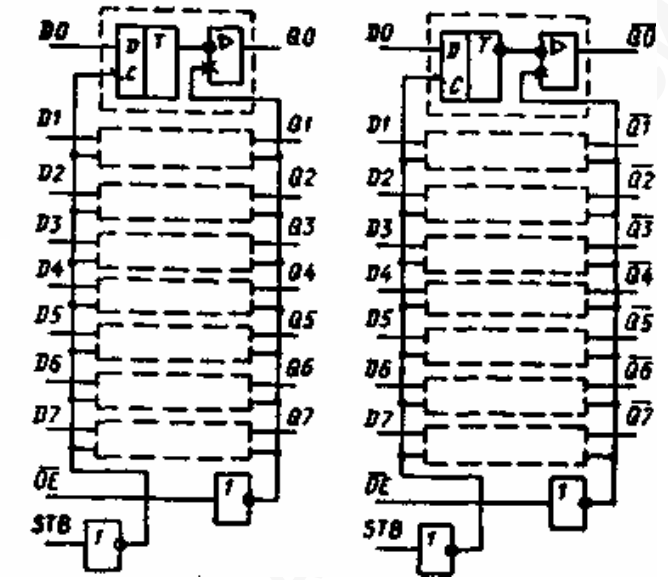


Рисунок 7.7 – Функциональная схема буферных регистров:
а) КР580ИР82; б) КР580ИР83

Условное графическое обозначение МС буферного регистра представлено на рис. 7.8, а назначение выводов - в табл.7.2. Работа выводов буферных регистров заключается в следующем. При сигнале высокого уровня на входе STB состояние входных линий DO-D7 передаётся на выходные линии Q0-Q7- Запоминание (защёлкивание) в информационных триггерах осуществляется при переходе сигнала STB от высокого уровня к низкому. Сигнал OE управляет выходными буферами: при OE= 0 буфер открывается, при OE=1 он устанавливается в высокоимпедансное состояние. Сигнал OE не влияет ни на состояние информационных триггеров, ни на функцию записи. Малый входной ток и достаточно большой выходной позволяют использовать эти элементы в качестве развязывающих буферов- защёлок либо ШФ.

Таблица 7.2 - Назначение выводов МС буферных регистров

Вывод	Обозначение	Тип вывода	Функциональное назначение выводов
1-8	*D0-D7*	Вход	Информационная шина
9	$\overline{OE}$	Вход	Разрешение передачи (управление 3-м состоянием)
10	*GND*	-	Общий
11	*STB*	Вход	Стробирующий сигнал
12-19	*Q7-Q0* ($\overline{Q7} - \overline{Q0}$ *для КР580ВА87)*	Выход	Информационная шина
20	U_{CC}	-	Напряжение питания 5В±5%

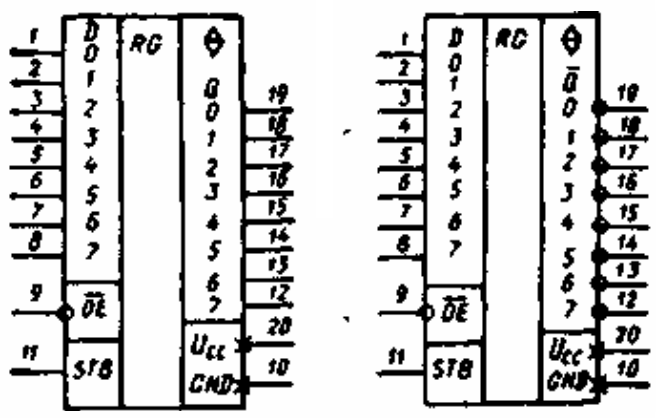


Рисунок 7.8 - Условное графическое обозначение МС буферных регистров

Системный контроллер КР580ВК28 (38) (рис.2.1) предназначен для формирования магистральных данных, управления микропроцессорной системой и выполняет дополнительную функцию умощнения ШД в целях увеличения её нагрузочной способности, т.е. возможности одновременного подключения к ШД нескольких устройств.

Поясним назначение данной микросхемы более подробно. Процессор формирует лишь два сигнала управления внешними устройствами: приём DBIN и выдачу - $\overline{WR}$, Поэтому недостающая управляющая информация о том, «что собирается делать микропроцессор» в данном машинном цикле, выдаётся в первом такте цикла на шину данных и по сигналу строб записи $\overline{STSTB}$ фиксируется в специальном регистре системного контроллера. На основе этой информации системный контроллер формирует сигналы управления памятью или устройством ввода/вывода.

Условное графическое обозначение микросхемы и назначение выводов представлены соответственно на рис.7.9 и 7.10.

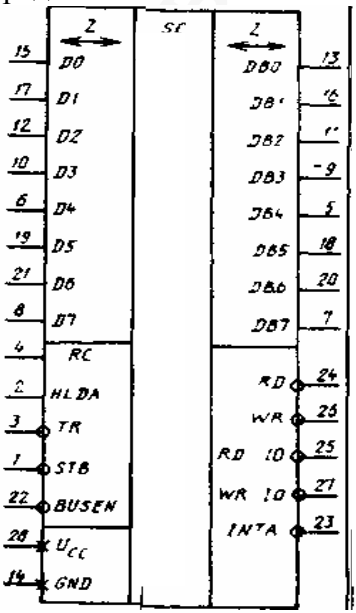


Рисунок 7.9- Условное графическое обозначение микросхемы

В состав микросхемы системного контроллера входит также двунаправленный шинный формирователь, который увеличивает нагрузку шины данных, т.е. повышает нагрузочную способность.

Вывод	Обозначение	Тип вывода	Функциональное назначение выводов
1	$\overline{STB}$	Вход	Стробирующий сигнал состояния
2	*HLDA*	Вход	Подтверждение захвата
3	$\overline{TR}$	Вход	Выдача информации
4	*RC*	Вход	Прием информации
5, 7, 9, 11 13, 16, 18, 20	*DB4, DB7, DB3, DB2, DB0, DB1, DB5, DB6*	Выход/Вход	Канал данных системы
6, 8,10, 12, 15, 17, 19, 21	*D4, D7, D3, D2, D0, D1, D5, D6*	Вход/Выход	Канал данных микропроцессора
14	*GND*	-	Общий
22	$\overline{BUSEN}$	Вход	Управление передачей данных и выдачей сигналов
23	$\overline{INTA}$	Выход	Подтверждение запроса прерывания
24	$\overline{RD}$	Выход	Чтение из ЗУ
25	$\overline{RD\_IO}$	Выход	Чтение из УВВ
26	$\overline{WR}$	Выход	Запись в ЗУ
27	$\overline{RD\_IO}$	Выход	Запись в УВВ
28	U_{CC}	Вход	Напряжение питания +5В

Рисунок 7.10 - Выводы микросхемы системного контроллера

Действие микросхемы КР580ВК28 (38) поясним посредством структурной схемы (рис.7.11). Дешифратор управляющих сигналов формирует один из управляющих сигналов в каждом

машинном цикле: при чтении ЗУ- $\overline{RD}$, а при записи в ЗУ- $\overline{WR}$, при чтении т УВВ- $\overline{RD}$ IO, при записи в УВВ- $\overline{WR}$ IO, при подтверждении запроса прерывания — сигнал $\overline{\text{INTA}}$.

Асинхронный сигнал $\overline{\text{BUSEN}}$ управляет выдачей данных с буферной схемы и управляющих сигналов с дешифратора: при напряжении низкого уровня на входе $\overline{\text{BUSEN}}$ буферная схема передает данные и формирует один из управляющих сигналов; при напряжении высокого уровня все выходы микросхемы переводятся в высокоимпедансное состояние.

Напряжение высокого уровня на входе $\overline{\text{HLDA}}$ переводит выходы RD, RD, IO, INTA в пассивное состояние (напряжение высокого уровня) и блокирует передачу информации через буферную схему данных.

Управляющие сигналы $\overline{\text{WR}}$ и $\overline{\text{WR IO}}$ формируются в цикле записи в микросхеме КР580ВК28 по сигналу TR, в микросхеме КР580ВК38- по сигналу $\overline{\text{STB}}$. На рис.7.12 представлена структурная схема процессора комплекта КР580 с многопользовательской системной и однопользовательской локальной шинами.

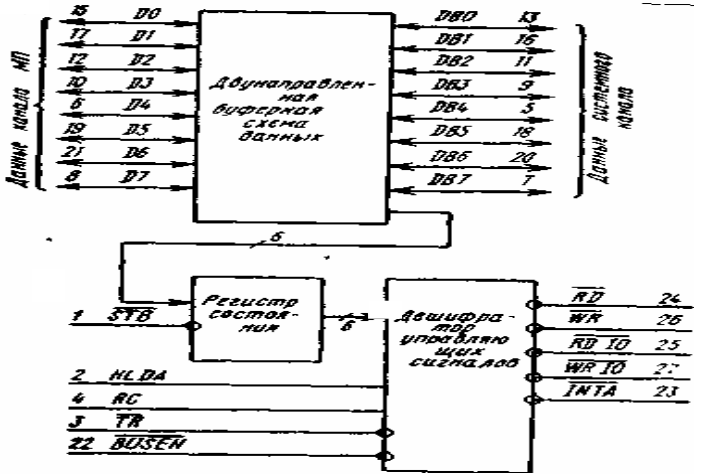


Рисунок 7.11- Структурная схема микросхемы системного контроля

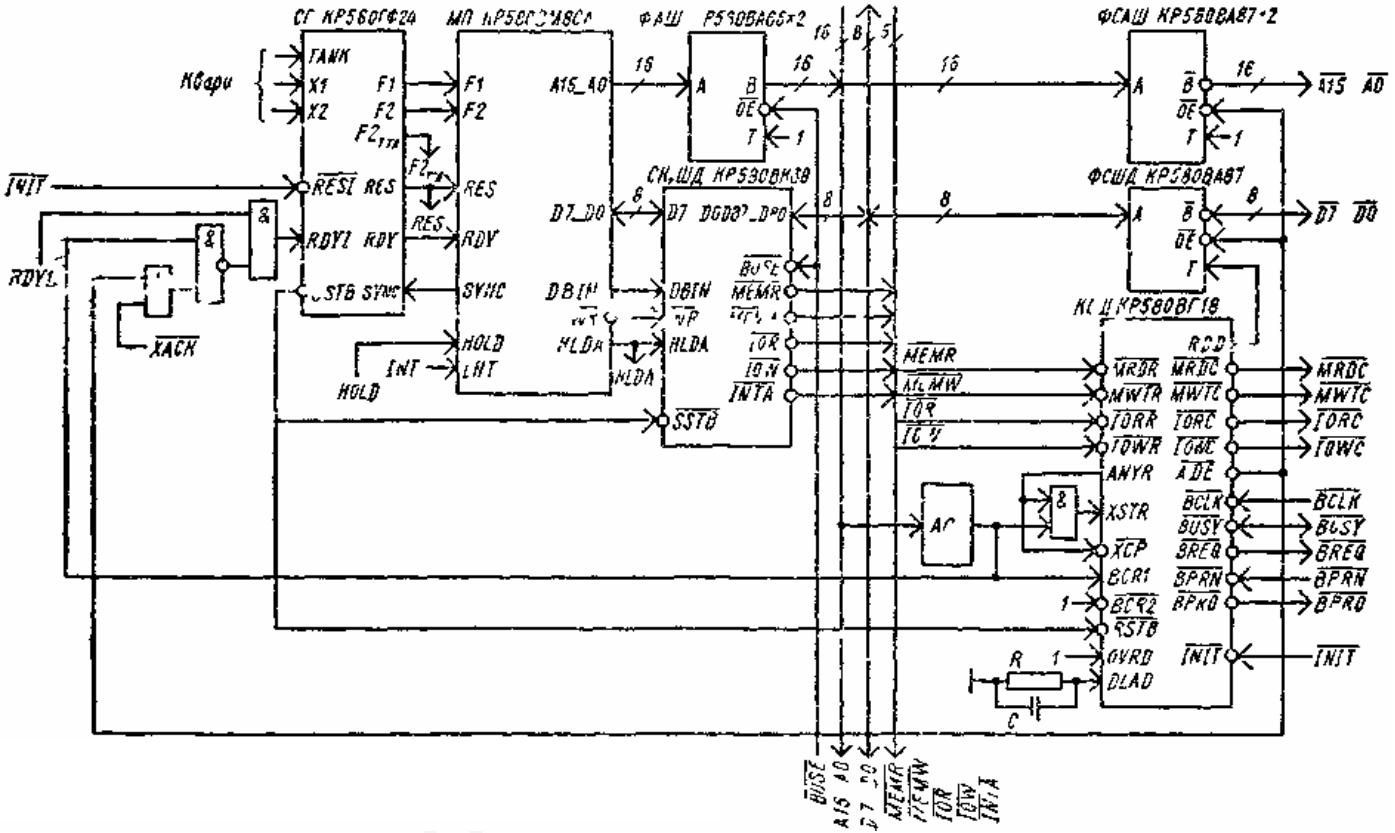


Рисунок 7.12 - Структурная схема процессора комплекта КР580 с однопользовательской системной и однопользовательской локальной шинами

ЛЕКЦИЯ 8 ШЕСТНАДЦАТИРАЗРЯДНЫЙ МП К1810ВМ86

1. Достоинства, состав, технические характеристики МП К1810ВМ86.
2. Особенности построения МП К1810ВМ86. Сигналы управления МП К1810ВМ86. Организация МПС.

Отметим, что МП К1810ВМ86 имеет расширенные функциональные возможности по сравнению с МП К580, что стало возможным благодаря:

-повышению степени интеграции БИС;

-увеличению разрядности шин

-повышению быстродействия.

Производительность данного МП на порядок выше МП К580. Это достигнуто за счет специализированных МП:

-арифметического сопроцессора К1810ВМ87;

-процессора ввода-вывода К181ОВМ89.

К1810ВМ87 выполняет операции над числами с фиксированной и плавающей запятой, обеспечивая высокую точность и быстродействие.

К1810ВМ89 осуществляет пересылку данных между ВУ и памятью системы, производит преобразование информации, освобождая ЦП от второстепенной работы, повышая его производительность по основной программе.

Необходимо также отметить, что в рассматриваемом комплекте имеется большое количество МС БИС, что упрощает разработку МПС и делает ее более экономной и компактной. Программы, составленные для МП К580, относительно легко переделать для выполнения на МП К1810, так как система команд МП 580 представляет собой подмножество команд МП К1810, что и обеспечивает программную совместимость этих МП.

На базе К1810 выпущены различные вычислительные средства:

-одноплатные управляющие микроЭВМ;
-микроконтроллеры;
-универсальные микроЭВМ;
-персональные микроЭВМ;
-высокопроизводительные МПС.

ЦП К1810ВМ86 представляет собой однокристальный 16-битовый МП, выполненный по высококачественной n-МОП технологии, питается от источника питания +5 В и синхронизируется однофазными импульсами с частотой 5 МГц. Он имеет разрядность данных - 16 бит, разрядность адреса - 20 бит.

Состав МПК серии К1810 приведен в табл.8.1.

Таблица 8.1- Состав МПК К1810

Тип БИС	Назначение
К1810ВМ86	Центральный процессор
К1810ВМ88	Процессор с 8-битовой внешней ШД
К1810ВМ87	Арифметический сопроцессор
К1810ВМ89	Специализированный процессор ВВ
К1810ГФ84	Генератор тактовых сигналов
К1810ВГ88	Системный контроллер
К1810ВБ89	Арбитр системной шины
К1810ВТ02	Контроллер динамической памяти (16К)
К1810ВТОЗ	Контроллер динамической памяти (64К)
К1810ВИ54	Интервальный таймер
К1810ВТ37	Контроллер прямого доступа к памяти
К1810ВН59	Программируемый контроллер прерываний
К1810ИР82/83	Регистр-защелка
К1810ВА86/87	Шинный формирователь

Отдельные технические характеристики МП К1810ВМ86:

-однокристальный;
-16-разрядный;
-быстродействие 2,5 млн операций/сек;
-емкость адресуемой памяти 1 Мбайт;
-система команд совместима с К580 на Ассемблере;

К1810ВМ87:

-однокристальный;

-16-разрядный;

-фиксированная и плавающая запятая;

К1810ВМ88:

-8-разрядный;

-быстродействие в 5 раз выше, чему К580;

-емкость адресуемой памяти 1 Мбайт;

-система команд аналогична К1810ВМ86.

МПК К1810 используется совместно с контроллером клавиатуры, индикации и интерфейсными БИС К580.

Достоинства комплекта

1 Совместим с К580.

2 Имеет высокую эффективность работы с языками высокого уровня.

3 Имеет гибкую и мощную систему команд.

4 Может быть использован как 16-, так и 8-разрядный МП.

5 Выполняет арифметические действия, в том числе умножение и деление.

6 Имеет гибкую структуру аппаратных и программных прерываний (до 256).

Отметим сразу следующие особенности МП:

-развитая регистровая структура, уменьшающая число обращений к памяти;

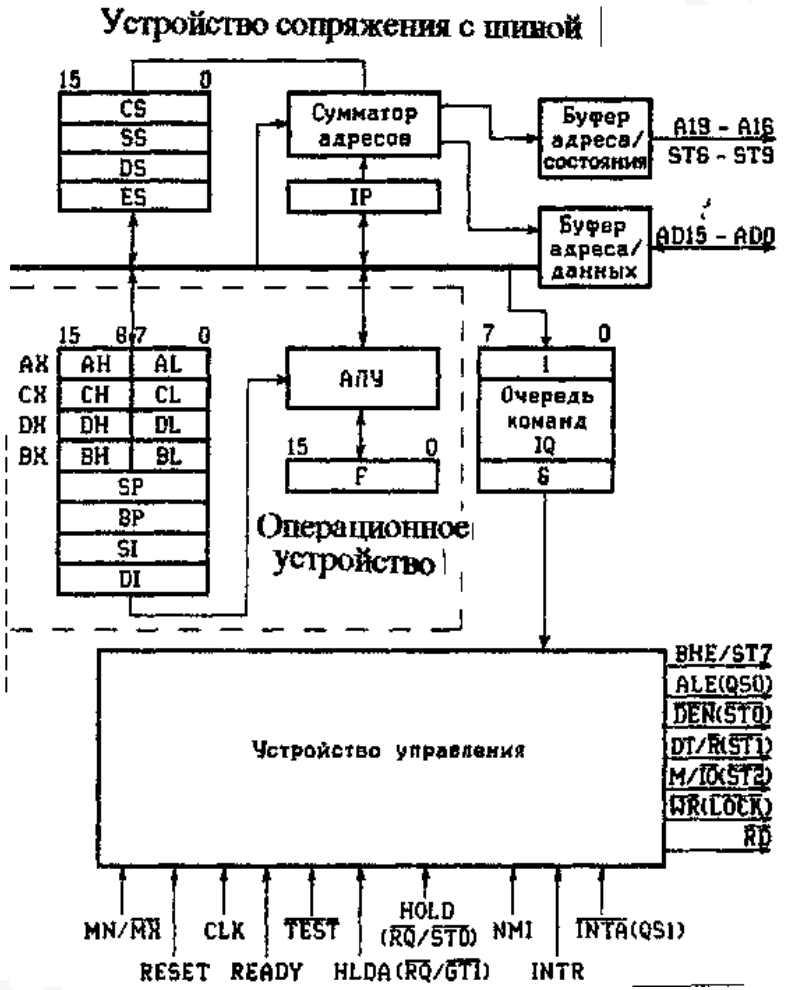
-конвейерная обработка команд с предварительной выборкой (увеличена пропускная способность системной шины);

-мультиплексированная шина А/Д. В первом такте машинного цикла по шине А/Д передаются младшие 16 бит адреса памяти или УВВ, во втором, третьем и четвертом тактах передаются данные:

-способность координировать взаимодействие нескольких МП (многопроцессорность);

-многофункциональное использование выводов, позво

позволяющее приспособить МП к уровню сложности. разрабатываемых систем.



8.1- Структурная схема 1810ВМ86

Сопроцессор анализирует команды, вырабатываемые независимым МП, и выполняет те команды, на которые распространяется его специализация.

Структурная схема

Структурная схема представлена на рис.8.1.Здесь параллельно выполняются функции выборки и выполнения команд, поэтому имеются две части, работающие раздельно (асинхронно):

-устройство сопряжения с шиной (УСШ);

-операционное устройство (ОУ).

УСШ формирует 20-разрядный адрес памяти, осуществляет выборку команд и операндов из памяти, организовывает очередность выполнения команд, запоминает результаты выполнения команд в памяти.

В его состав входят:

-шесть 8-разрядных регистров очереди команд;

-четыре 16-разрядных сегментных регистра;

-16-разрядный сумматор адреса;

-16-разрядный указатель команд (счетчик);

-16-разрядный регистр обмена.

Регистры очереди команд

Из них ОУ последовательно извлекает очередную команду побайтно. Как только освобождается два регистра, УСШ выбирает из памяти программ следующие 2 байта.

Сегментные регистры

К ним относятся сегмент данных (DS), сегмент стека (SS), сегмент кода (CS), сегмент промежуточных данных (ES). Они предназначены для динамического перемещения программ и данных в памяти. Для динамического перемещения достаточно модифицировать содержимое сегментных регистров.

Всё пространство памяти К18IOBM86 ёмкостью I Мбайт представляет собой набор сегментов, определяемых программным путем. Каждый сегмент представляет собой набор ячеек памяти, является независимым с адресуемой памятью ёмкостью 64

Кбайт. Каждому сегменту программой назначается начальный (базовый) адрес, являющийся адресом первого байта сегмента в пространстве памяти. Начальные адреса сегментов записываются в регистры.

Сегментные регистры инициализируются в начале программы путём засылки в них соответствующих констант. Если загрузить сегментные регистры нулями, то получим организацию памяти, характерную для МП К580, который не имеет сегментной памяти.

В сегментном регистре хранится 16 старших бит 20-битового начального адреса сегмента. Четыре младших бита адреса принимаются равными нулю и дописываются справа к содержимому сегментного регистра при вычислении физических адресов ячеек памяти. Поэтому начальные адреса сегментов всегда кратны 16.

Как известно, стек организуется в ОЗУ, и его положение определяется содержимым регистров SS и SP. Регистр SS хранит базовый адрес текущего сегмента стека, а регистр SP указывает на вершину стека, т.е. содержит смещение вершины стека в стековом сегменте. При каждом обращении к стеку пересылается одно слово, причём содержимое SP модифицируется автоматически: при записи (включении) в стек оно уменьшается на два, при чтении (извлечении)из стека -увеличивается на два.

ОУ извлекает команды из очереди команд и реализует операции в 16-разрядном АЛУ.

В его состав входят:

- блок РОН; АЛУ;
- регистр признаков.

Регистры общего назначения

В состав блока РОН входят четыре 16-разрядных регистра **AX**, **BX**, **CX** и **DX**, допускающие независимую адресацию старших (H) и младших (L) половин, что позволяет прямо обрабатывать как байты, так и двухбайтные слова.

Все регистры блока РОН участвуют в выполнении ариф-

метических и логических операций.

Регистр **AX** выполняет функции аккумулятора (с ним связаны операции умножения и деления, преобразования и десятичной коррекции).

Регистр **AL** соответствует аккумулятору МП К580, а регистр AH является его расширением.

Регистр **BX** является источником базового адреса (соответствует регистровой паре HL МП К580).

Регистр **CX** используется в качестве счетчика в командах сдвига и циклов.

Регистр **DX** неявным образом адресуется в командах умножения и деления и содержит адрес порта ВВ при косвенно-регистровой адресации.

Блок РОН также содержит два 16-разрядных указательных (SP и BP) и два 16-разрядных индексных регистра, которые участвуют в выполнении арифметических операций над двухбайтными словами, вычисляют исполнительный адрес и обеспечивают косвенную адресацию.

Регистр признаков

Формат 16-разрядного регистра признаков представлен на рис.8.2. Его младший байт FL соответствует регистру признаков

МП К580. В дополнение к этому в регистре признаков фиксируются некоторые признаки, предназначенные для управления вычислительным процессом в МП:

TF - признак прослеживания; при TF=1 МП переходит в режим исполнения программы по командам (пошаговый режим), что необходимо при отладке программ.

IF - признак прерывания; используется в механизме маскирования прерываний (при IF=1 прерывания разрешены, т.е. МП воспринимает запросы на прерывание).

DF - признак направления; при DF=0 выполняется инкрементирование, а при DF=1- декрементирование содержимого регистра-указателя массива.

OF - признак переполнения.

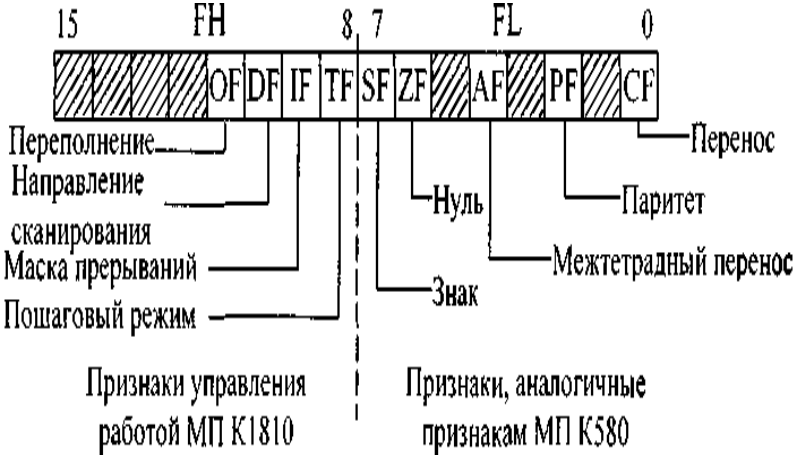


Рисунок 8.2 - Формат регистра признаков

Устройство управления

Осуществляет управление УСШ и ОУ и периферийным оборудованием системы, обеспечивая обмен данными с использованием механизмов прерывания и прямого доступа к памяти Дешифрует команды, а также воспринимает и вырабатывает необходимые управляющие сигналы. На базе УУ реализован блок микропрограммного управления.

Сигналы управления МП К1810ВМ86

ЦП осуществляет общую обработку данных и управление блоками системы в соответствии с заданной программой. Особенностью рассматриваемого МП является возможность частичной реконфигурации аппаратной части для обеспечения работы в двух режимах: минимальном и максимальном.

В **минимальном режиме** для построения блока ЦП используется малое число ИС: генератор тактовых импульсов, буферные регистры и шинные формирователи. МП формирует сиг-

налы для управления внутрисистемным интерфейсом и используется для построения однопроцессорных контроллеров и микроЭВМ.

В **максимальном режиме** МП используется для построения многопроцессорных систем, где сигналы управления шиной вырабатываются системным контроллером на основании кода, сформированного МП.

Условное графическое обозначение МП К1810ВМ86 представлено на рис.8.3.

Рассмотрим функциональное назначение сигналов МП.

AD15-AD0 — 16-разрядная, мультиплексная (совмещенная) двунаправленная шина адреса/данных (ШАД). Разделенные во времени по ней передаются адресная информация и данные. В первом такте цикла шины (цикла обращения к ЗУ или ВУ) МП выдает на эту шину младшие 16 бит адреса памяти или полный адрес ВУ. В остальных трех тактах цикла шины по этим линиям передаются данные.

A19/S6-A16/S3 — 4-разрядные, мультиплексные выходные линии адреса/состояния. В первом такте на эти линии выдаются старшие 4 бит адреса памяти, а при адресации ВУ - нули. В остальных тактах цикла шины МП выдает на эти линии сигналы состояния S6 S3. Код на линиях **S3** и **S4** указывает сегмент памяти, к которому обращается МП в текущем цикле (табл. 8.2). Сигнал S5 указывает на состояние флага разрешения прерываний IF: 0 - прерывания запрещены; 1 - прерывания разрешены. Сигнал **S6** не используется и всегда равен нулю.

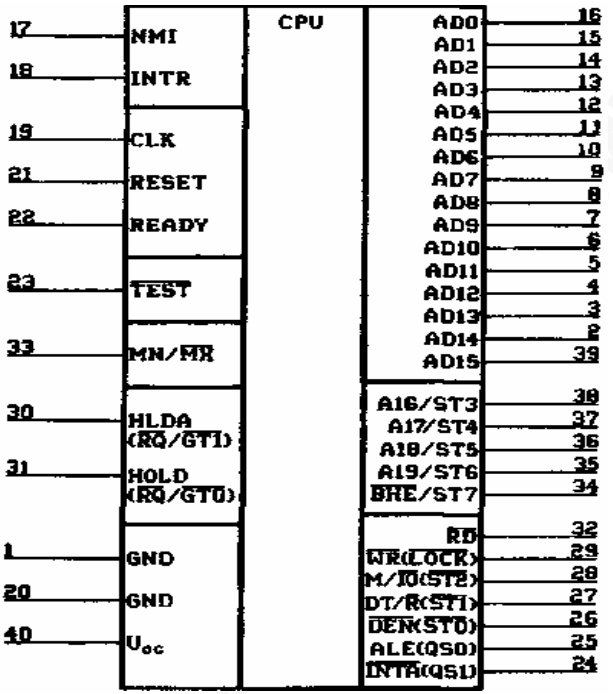


Рисунок 8.3 - Условное графическое обозначение МП К181ОВМ86

$\overline{\text{BHE}}$ **/S7** — разрешение старшего байта. Формируется в первом такте цикла одновременно с адресной информацией. Сигнал низкого уровня означает, что по старшей половине AD15-AD8 ШАД передаются 8-битные данные. В остальных тактах машинного цикла на этом выходе присутствует признак **S7** состояния МП, не имеющий определенного значения. Совместное использование $\overline{\text{BHE}}$ и младшей линии адреса А0 для дешифрации адресов позволяет осуществлять передачу слов и отдельных байт по ШАД (табл. 8.3).

$\overline{\text{RD}}$ — чтение. Нулевой сигнал на этом выходе указывает на то, что МП принимает информацию из памяти или ВУ.

$\overline{\text{WR}}$ — запись. Нулевой сигнал указывает на то, что МП выдает информацию для записи в память или ВУ.

Таблица 8.2- Код S4 и S3

S4	S3	Сегментный регистр
0	0	ES
0	1	SS
1	0	CS
1	1	DS

Таблица 8.3- Код BHE и A0

$\overline{\text{BHE}}$	A0	Разрядность данных
0	0	Все слово (оба байта)
0	1	Старший байт D15-D8, нечетный адрес
1	0	Младший байт D7-D0, четный адрес
1	1	Нет обращения

M/$\overline{\text{IO}}$ — выходной сигнал, который позволяет отличить передачу информации между МП и памятью (M/IO=1) от передачи между МП и ВУ (M/IO). Таким образом, данный сигнал служит для разделения адресного пространства памяти и ВУ.

DT/$\overline{R}$ — выходной сигнал, который указывает на направление передачи информации между МП и памятью или ВУ: DT/R=1 - МП выдает информацию, DT/R=0 - МП принимает информацию.

$\overline{\text{DEN}}$ — строб данных (разрешение передачи данных). Этот сигнал управляет выдачей информации из шинных формирователей при выполнении команд чтения и записи.

ALE — строб адреса. Этот выходной сигнал стробирует в начале каждого цикла шины передачу адресной информации с ШАД на другие элементы микроЭВМ. Обычно этот сигнал управляет записью адреса в адресный буферный регистр.

HOLD — запрос захвата шины от ВУ или контроллера прямого доступа к памяти.

HLDA — подтверждение захвата шины, выдается в ответ на сигнал HOLD после приостанова вычислительного процесса в МП и перевода ШАД в z-состояние. При HOLD=1 устройство, инициирующее запрос захвата, использует шину самостоятельно. Когда HOLD=0, ЦП выдает сигнал HLDA=0, возобновляет управление шиной и продолжает работу по

программе.

NMI — немаскируемое прерывание, распознается МП по завершению текущей команды независимо от состояния флага разрешения прерывания IF. Этот вход предназначен для сигнализации о некоторых критических ситуациях (например, об аварийном отключении питания).

INTR — запрос прерывания, опрашивается ЦП в конце выполнения каждой команды, если прерывания разрешены (IF=1). Если IF=0, то запрос по этому входу игнорируется. Обычно на этот вход подается запрос от программируемого контроллера прерываний.

$\overline{\text{INTA}}$ — подтверждение запроса прерывания, формируется в ответ на INTR, обеспечивает ввод информации в МП из источника прерывания, вызвавшего переход в режим прерывания.

RDY — готовность, указывает на то, что адресуемое в данном цикле устройство готово к обмену данными. Если устройство не готово к взаимодействию с МП, то оно выдает сигнал RDY=0, а МП переходит в состояние ожидания (между тактами Т3 и Т4 появляется необходимое число тактов ожидания Tw. При RDY=1 МП выходит из состояния ожидания и возобновляет работу.

$\overline{\text{TEST}}$ — проверка, используется вместе с командой ожидания WAIT, выполняя которую МП проверяет уровень данного сигнала. Если TEST = 0, то МП переходит к выполнению следующей по порядку команды. Если TEST = 1, то МП вводит холостые такты ТГ и периодически с интервалом 5Т проверяет значение сигнала TEST.

CLK — тактовая синхронизация. Сигнал синхронизации подается от генератора тактовых импульсов и предназначен для синхронизации МП. Используется серия тактовых импульсов с периодом повторения Т=200 - 500 нс.

RESET — сброс, переводит МП в начальное состояние, т.е. сброшены сегментные регистры (кроме CS, все разряды

которого равны 1), указатель команд IP, все флаги, регистры очереди команд, все триггеры в устройстве управления. RESET не влияет на состояние общих регистров — они устанавливаются в начальное состояние программным путем. При действии данного сигнала все выходы, имеющие три состояния, переходят в z-состояние, а выходы, имеющие два состояния, - в пассивное. После снятия сигнала RESET работа МП возобновляется из начального состояния.

MN/ $\overline{MX}$ — минимальный/максимальный режимы. Сигнал на этом входе определяет режим работы МП: 1 - минимальный режим, 0 - максимальный режим. В максимальном режиме изменяется назначение восьми выводов МП (на рис. 1 названия этих выводов взяты в скобки). Ниже описываются эти восемь управляющих сигналов.

$\overline{ST0} - \overline{ST2}$ — сигналы состояния МП, указывающие на один из типов выполнения цикла шины (табл. 8.4).

QS0-QS1 — сигналы, которые указывают на состояние памяти очереди команд (табл. 8.5). Данные сигналы предназначены для сопроцессора, который воспринимает команды и операнды с помощью команды ESC.

$\overline{RQ}/\overline{GTO}, \overline{RQ}/\overline{GT1}$ — запрос/предоставление локальной шины. Двунаправленные линии, предназначенные для обмена сигналами между процессорами в многопроцессорной системе, для управления процедурой использования шин. Процесс доступа к шине осуществляется в следующем порядке: 1) устройство, захватывающее шину, формирует запросный импульс длительностью один такт; 2) МП выдает ответный импульс, подтверждающий возможность доступа к шине (один такт), в следующем такте МП переводит свои шины в z-состояние; 3) по окончании работы с шиной устройство выдает импульс, указывающий на окончание захвата. В следующем такте МП возобновляет управление шиной и продолжает работу. Функционально эти две линии одинаковы, но $\overline{RQ}/\overline{GTO}$ имеет

более высокий приоритет, чем $\overline{RQ}/\overline{GTI}$.

Таблица 8.4 –Сигналы состояния МП

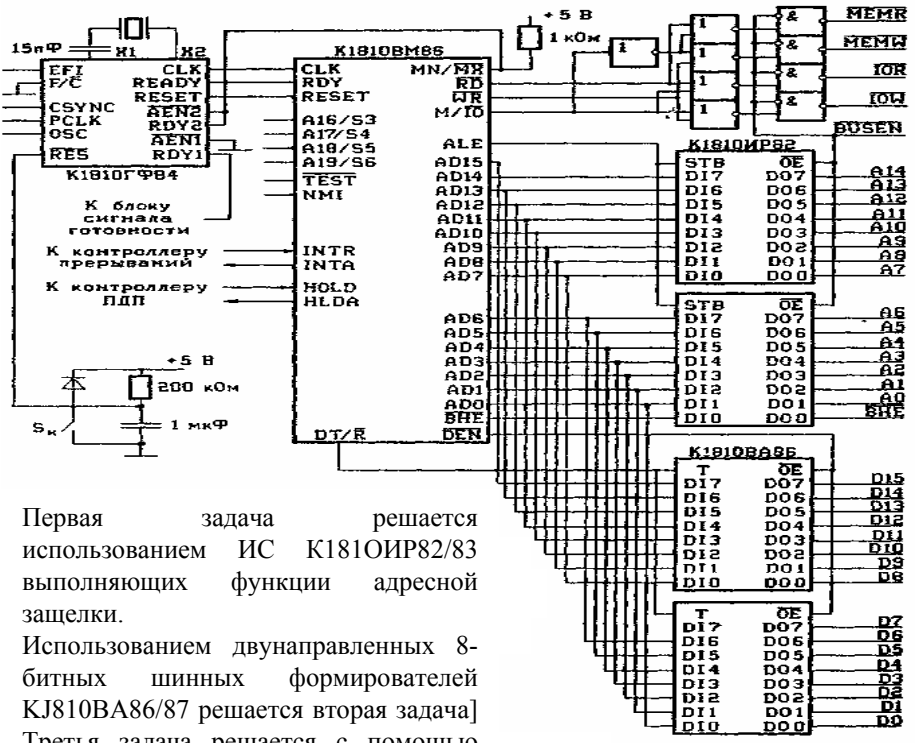
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Тип цикла шины
0	0	0	Подтверждение прерывания
0	0	1	Чтение ВУ
0	1	0	Запись ВУ
0	1	1	Останов
1	0	0	Выборка команды
1	0	1	Чтение ЗУ
1	1	0	Запись ЗУ
1	1	1	Цикла шины нет

Таблица 8.5 –Сигналы состояния памяти

QS1	QS0	Операция над очередью
0	0	Операции нет, в последнем такте не было выборки из очереди
0	1	Из очереди выбран первый байт команд
1	0	Очередь пуста, была опустошена командой передачи управления
1	1	Из очереди выбран следующий байт команды

$\overline{LOCK}$ — блокировка шины. Если на этом выходе сигнал низкого уровня, то другие устройства, в частности, другие процессоры, не могут захватить системную шину.

При организации МПС (рис. 8.4) возникают задачи разделения (демультиплексирования) шины адреса/ данных (ШАД), буферирования шин адреса (ША) и шин данных (ШД), а также формирования системных управляющих сигналов для блоков памяти и внешних устройств.



Первая задача решается использованием ИС К181ОИР82/83 выполняющих функции адресной защелки.

Использованием двунаправленных 8-битных шинных формирователей KJ810BA86/87 решается вторая задача] Третья задача решается с помощью комбинационных устройств, которые формируют управляющие сигналы на основе сигналов RD, WR и M/IO, вырабатываемых МП.

Рисунок 8.4- Организация МПС

ЛЕКЦИЯ 9 АДРЕСНОЕ ПРОСТРАНСТВО ПАМЯТИ

1 Организация взаимодействия ОУ и УСШ.

2 Адресация памяти.

1 Как нам уже известно, МП К1810ВМ86 по своей структуре представляет собой два блока УСШ и ОУ. УСШ предназначено для извлечения из памяти кода команд и их операндов, а также записи результата в память. ОУ извлекает команды из очереди команд и реализует операции в 16-разрядном АЛУ.

Эти два блока работают независимо друг от друга, следовательно, процессы преобразования и передачи информации в них могут выполняться параллельно. На рис.9.1 для сравнения

а)

б)

Рисунок 9.1 - Процесс выполнения команд в микропроцессоре с последовательным (а) и с параллельным (б) включением этапов в блоках ОУ и УСШ приведены два варианта различных способов выполнения команд одной программы в процессорах. Предполагается, что на первом этапе оба процессора выполняют первую команду. Она состоит из двух

этапов: выполнение 1 и запись результата в память (запись 1). Вторая команда - получение кода (код 2), выполнение 2. Третья команда - получение кода (код 3), получение операнда (операнд 3) выполнение 3. Четвертая команда - получение кода (код 4), выполнение 4. Как видно из рисунка, в МП К1810ВМ86 за одно и то же время количество выполненных и полученных команд программы больше, при этом шины (магистрали) микроЭВМ используются для обмена информацией более эффективно.

Как уже известно, ОУ имеет 16-разрядное АЛУ с регистром состояния и флагами управления, а также регистры общего назначения. На АЛУ поступают коды команд из конвейера команд, расположенного в УСШ. Если в результате дешифрации кода команд в АЛУ необходимо получение одного или нескольких операндов по внешним магистралям МП, то ОУ запрашивает УСШ на получение и размещение необходимых данных в УСШ. Несмотря на то, что все адреса, с которыми оперирует ОУ, 16-разрядные, УСШ производит необходимое преобразование адресов так, чтобы ОУ имело возможность обращаться ко всему возможному адресному пространству (1 Мбайт) микропроцессорной системы.

УСШ производит все необходимые пересылки данных и

.

УСШ организует получение нового кода команды, как только 2 байта в конвейера команд будут переданы в ОУ.

Как правило, в УСШ находится хотя бы одна команда и ОУ не ждет, пока очередная команда будет извлечена из памяти. Коды команд подаются в ОУ последовательно так, как они записаны в программе. Если ОУ выполняет команду передачи

управления в другое место программы, то УСШ очищает конвейер команд, получает код команды из нового адреса, передает его в ОУ и начинает заполнять конвейер заново. Если ОУ требует обращения к памяти или ВУ, то УСШ приостанавливает процесс получения команд в конвейер и организует необходимый цикл обмена данными.

Чтобы легче было понять метод адресации, считаем, что любая ячейка памяти (ЯП) имеет два типа адресов: **физический и логический.**

Физический адрес - это 20-разрядное число, которое однозначно определяет любую из 1 Мбайт ЯП. В 16-разрядной системе счисления адреса лежат в диапазоне от 0 до FFFFF. Весь обмен информацией между МП БИС и памятью осуществляется с помощью физических адресов.

Логический адрес позволяет записывать команду без предварительного знания места, где эта команда будет размещена в памяти (это дает возможность программисту распределять ресурсы памяти при выполнении программ).

Логический адрес состоит из двух составляющих, которые представляются 16-разрядными числами:

- значения базы сегмента (базовый адрес);
- значения смещения в сегменте.

Как только УСШ обращается к памяти, базовый адрес формирует физический адрес из логического по принципу:

-значение базы сегмента смещается на четыре разряда влево;

-полученное 20-разрядное число (с четырьмя нулями в младших четырех разрядах) складывается со значением смещения в сегменте (рис. 9.2).

Таким образом, база сегмента (с четырьмя нулями в качестве младших разрядов) задает для памяти сегмент длиной в 64Кбайта, а значение смещения в сегменте - расстояние от начала сегмента до искомого адреса памяти.

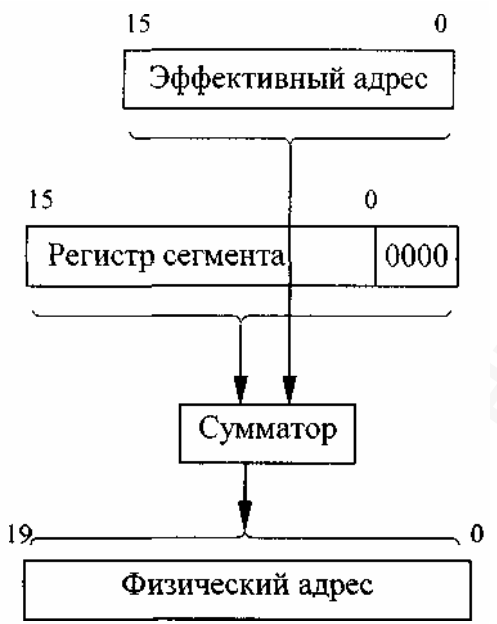


Рисунок 9.2 - Вычисление адреса памяти

Максимально возможное смещение в сегменте равно 64 байта. В любой момент времени программа может осуществить доступ к одному из четырех сегментов:

- сегменту текущего кода CS *(Current Code Segment);*
- сегменту текущих данных DS *(Current Date Segment);*
- сегменту текущего значения стека SS *(Current Stack Segment);*
- дополнительному сегменту текущих значений ES *(Current Extra Segment).*

Сегменты могут быть заданы в программах произвольно по усмотрению программиста и могут пересекаться в адресном пространстве.

В зависимости от команд УСШ получает информацию о логическом адресе памяти из различных регистров МП БИС. Коды команд всегда извлекаются из адреса памяти,

определяемого содержимым регистра текущего кода и регистра указателя команд IP. В регистре IP записано смещение в сегменте текущего кода CS. Команды со стеком всегда используют для адресации SS и регистр-указатель стека SP, где записано смещение в сегменте текущего значения стека. Данные или переменные в командах извлекаются из адресов памяти, расположенных в сегменте текущих данных DS.

При вычислении физического адреса смещение в сегменте задается в ОУ, т.к. в нем происходят дешифрация кода команд и определение сегмента памяти, с которым будет работать МП БИС к текущий момент времени.

Организация ввода/вывода. Ввод/вывод данных может осуществляться двумя способами: с использованием адресного пространства ввода/вывода и с использованием общего с памятью адресного пространства, т.е. с отображением на память.

При **первом** способе применяются специальные команды ПЧ (ввод) и OUT (вывод), которые обеспечивают передачу данных между аккумуляторами AL или AX и адресуемыми портами.При выполнении этих команд вырабатывается сигнал $M/\overline{IO}$ =0, который идентифицирует выбор пространства

сформировать системные сигналы IOW и IOR для управления операциями записи данных в порт и чтения из порта.

При **втором** способе адреса портов размещаются в об ем адресном простра об-
ращения к ячейкам памяти. Это повышает гибкость программирования, так как для ввода/вывода можно использовать любую команду с обращением к памяти при любом способе адресации.

Так, например, MOV позволяет передавать данные между любым общим регистром или ячейкой памяти и портом ввода/вывода, а логические команды AND, OR, XOR, TEST позволяют манипулировать битами в регистре порта.

ЛЕКЦИЯ 10. ОРГАНИЗАЦИЯ И ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ОДНОКРИСТАЛЬНЫХ МИКРОКОНТРОЛЛЕРОВ

1 Основные положения, модификации, технические характеристики, применение микроконтроллеров серии КМ1816

2 Особенности построения однокристальных 8-разрядных микроконтроллеров серии КМ1816

Несмотря на непрерывное развитие и появление всё новых и новых 16- и 32- разрядных микроконтроллеров и микропроцессоров, наибольшая доля мирового микропроцессорного рынка и по сей день остаётся за 8-разрядными устройствами.

В настоящее время среди всех 8-разрядных микроконтроллеров семейство MCS-51 является несомненным чемпионом по количеству разновидностей и количеству компаний, выпуск
представителя этого семейства **- микроконтроллера 18051,** выпущенного в 1980 году компанией Intel. Микроконтроллер представлял довольно сложное устройство: в кристалле было 128 тыс. транзисторов, что в четыре раз превышало количество транзисторов в 16-разрядном микропроцессоре i8086.

В настоящее время имеется более 200 модификаций микроконтроллеров семейства i8051, выпускаемых почти 20 компаниями. Эти модификации включают в себя кристаллы с широчайшим спектром периферии: от простых 20- выводных устройств с одним таймером и 1К программной памяти до сложнейших 100-выводных кристаллов с 10- разрядными АЦП, массивами таймеров-счётчиков, аппаратными 16-разрядными умножителями и 64К программной памяти на кристалле.

Основными направлениями развития микроконтроллеров являются: увеличение быстродействия (повышение тактовой частоты и переработка архитектуры), снижение напряжения питания и потребления, увеличение объёма ОЗУ и FLASH-памяти на кристалле, введение в состав периферии

микроконтроллера сложных устройств типа системы управления приводами и т. д.

Все микроконтроллеры (МК) из семейства MCS имеют общую систему команд. Наличие дополнительного оборудования влияет только на количество регистров специального назначения

Основными производителями клонов 51 семейства в мире являются фирмы Intel, AMD, Philips, Siemens и др. Основные технические характеристики отдельных однокристальных микроконтроллеров приведены в табл. 10.1.

В рамках СССР производство микроконтроллеров i8051 осуществлялось в Киеве (1816ВЕ31/51), Воронеже (1830ВЕ31/51), Минске (1834ВЕ31/51), Новосибирске (1850ВЕ31).

Микроконтроллеры данного семейства выпускаются в DIP QFP, PLCC корпусах и могут работать при следующих температурных режимах: коммерческий (0°С - +70°С); расширенный $НО^0$ С - +85° С); для военного пользования (-55 °С - +125 °С).

Примерами МК семейства MCS -51c расширенными возможностями являются 8XC51FA, 8XC51GB и др.

Все аппаратные средства МК лучше всего приспособлены для решения задач управления и регулирования в сравнительно несложных объектах, приборах и технологических процессах.

МК этой серии разработаны для устройств, требующих короткие программы, небольшой объём оперативной памяти и имеющих ограниченные возможности ввода/вывода.

Микроконтроллеры обычно классифицируют по разрядности обрабатываемых чисел: четырехразрядные-самые простые и дешевые; восьмиразрядные - наиболее многочисленная группа (оптимальное сочетание цены и возможностей), шестнадцати-разрядные - MCS-96 (intel).

Микроконтроллеры данного семейства размещены в 40-выводном корпусе и имеют два вывода питания +5 В, один из которых Ucc подводит питание к МК, а другой Uss – к внутреннему ОЗУ, что позволяет сохранять в нем данные во время сбоя питания. Микросхема МК представлена на рис. 10.1. Она состоит из 4 портов, каждый из которых выполняет определенные функции по обработке информации.

Таблица 10.1 - Основная характеристика однокристальных МК.

Обозначения	Резидентная память программ	Резидентное ОЗУ	Таймеры/счетчик	Послед. порт	Аналог. выходы	Число линий В/В	Тактовая частота	Тип корпуса	Секретность	Ключевые особенности
Серия 8051										
8031AH	–	128	2	1	0	32	12	D,N,P	нет	Процессор булевых функций
8051AH	4К ROM	128	2	1	0	32	12	D,N,P	P	Процессор булевых функций
8751 H	4К EPROM	128	2	1	0	32	12	D	L1	Один уровень блокировки ЗУ
8751 BH	4К OTP ROM	128	2	1	0	32	12	N,P	L2	Два уровня блокировки ЗУ
Серия 8052										
8032AH	–	256	3	1	0	32	12	D,N,P	нет	Три таймера-счетчика
8052AH	8К ROM	256	3	1	0	32	12	D,N,P	нет	Три таймера-счетчика
8752BH	8К OTP/EPROM	256	3	1	0	32	12	D,N,P	L2	Два уровня блокировки ЗУ
Серия 80C51										
80C31BH	–	128	2	1	0	32	12,16	D,N,P,S	нет	Режимы управления потреблением
80C51BH	4К ROM	128	2	1	0	32	12,16	D,N,P,S	P	Режимы управления потреблением
87C51	4К OTP/EPROM	128	2	1	0	32	12,16,20,24i	D,N,P,S	L3	Три уровня блокировки ЗУ

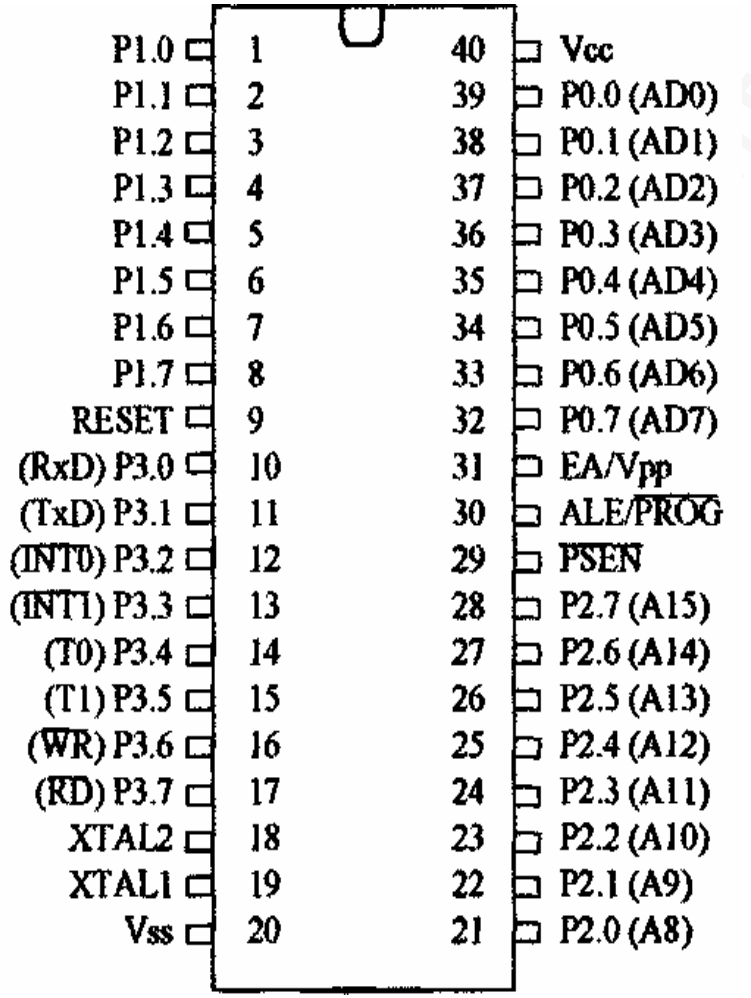


Рисунок 10.1 - Назначение **выводов МК**

Приведём обозначения сигналов на рис.10.1.

Uss- потенциал общего провода (земля);

Ucc- основное напряжение питания (+5В);

X1Д2- выводы для подключения кварцевого резонатора;

RST- ввод общего сброса МК;

PSEN- разрешение внешней памяти программ (выдается только

при обращении к внешнему ПЗУ);

ALE- строб адреса внешней памяти;

EA- отключение внутренней программной памяти (при уровне сигнала 0 МК выполняет программу только внешнего ПЗУ).

Порты МК:

P1 - восьмибитный двунаправленный порт ввода/ вывода; каждый разряд порта может быть запрограммирован как на ввод, так и на вывод данных;

P2- восьмибитный двунаправленный порт, аналогичный P1; выводы этого порта используются для выдачи адресной информации при обращении к внешней памяти программ или данных;

P3- восьмибитный двунаправленный порт, аналогичный P1; выводы этого порта выполняют ряд функций, используемых при работе таймера, порта последовательного ввода/вывода, контроллера прерываний, внешней памяти программ и данных;

P0- восьмибитный двунаправленный порт ввода/вывода информации; по линиям порта в режиме временного мультиплексирования выдается адрес внешней памяти, после чего осуществляется передача или приём данных; порт работает с внешними ОЗУ и ПЗУ.

Особенностью архитектуры МК является физическое и логическое разделение памяти на программную и память данных, которая может расширяться путём подключения к ней внешних БИС: программная - на 4 Кбайта, данных - на 128 байт. Возможно также расширение средств ввода/вывода за счёт использования интерфейсных БИС серии КР580.

Однокристальный 8-разрядный микроконтроллер КМ1816ВЕ51, для краткости называемый МК 1816, представляет собой БИС, имеющую в своём составе все атрибуты небольшой микроЭВМ, Иногда в публикациях МК 1816 называют однокристальной микроЭВМ (ОЭВМ),

На одном кристалле, изготовленном по n- МОП-технологии, установлены:

-8 разрядный центральный процессор;

-оперативное ЗУ данных, постоянное ЗУ программ;

-система прерываний;

-тактовый генератор;

-интерфейсные схемы ввода/вывода информации.

Незначительный объём памяти программ и данных, простой набор команд и ограниченные возможности ввода/вывода информации определяют основные формы его использования:

-в качестве специализированного вычислителя, включаемого в контур управления объектом или процессом;

-в качестве устройства решения задач управления и регулирования в несложных объектах, приборах и технологических процессах.

Можно сказать, что микроконтроллер КМ1816ВЕ51- это СБИС со встроенной перепрограммируемой памятью (ППЗУ), со стиранием информации ультрафиолетовым излучением, изготовленную по n- МОП- технологии. Данная СБИС может быть использована в контрольно-измерительной аппаратуре, роботах, бытовой технике.

Наличие ШВУ с ультрафиолетовым стиранием позволяет, пользователю самостоятельно задавать функции ОЭВМ путём записи своей программы в ППЗУ. Это особенно становится полезным для систем, требующих периодической модернизации, и в мелкосерийном производстве.

Организация МК и его система команд допускают в случае необходимости расширение функционально-логических возможностей контроллера. С использованием внешних дополнительных БИС адресное пространство памяти программ и данных может быть расширено.

Однокристальный МК К1816 имеет следующие аппаратные особенности:

-внутреннее ОЗУ объемом 128 байт;

-четыре двунаправленных побитно настраиваемых;

-восьмиразрядных порта ввода-вывода;

-два 16-разрядных таймера-счетчика;

-встроенный тактовый генератор;

-адресация 64 КБайт памяти программ и 64 Кбайт памяти данных;

-две линии запросов на прерывание от внешних устройств;

-интерфейс для последовательного обмена информацией с другими микроконтроллерами или персональными компьютерами;

-ПЗУ с ультрафиолетовым стиранием объемом 4Кбайта.

Структурная схема К1816 во многом напоминает схему микропроцессора серии КР580. МК К1816, в отличие от КР580, характерен тем, что:

-увеличен объем внутренней сверхоперативной памяти данных;

-введена память команд; введен аппаратный таймер;

-используется мультиплексирование данных и адреса в порту ввода/вывода;

-увеличено число сигналов логического воздействия на микроЭВМ.

Элементы структурной схемы могут быть объединены в четыре блока:

- центральное процессорное устройство;
- блок программного управления;
- устройство ввода-вывода;
- блок оперативных регистров.

Рассмотрим подробнее структурную схему (рис. 10.2).

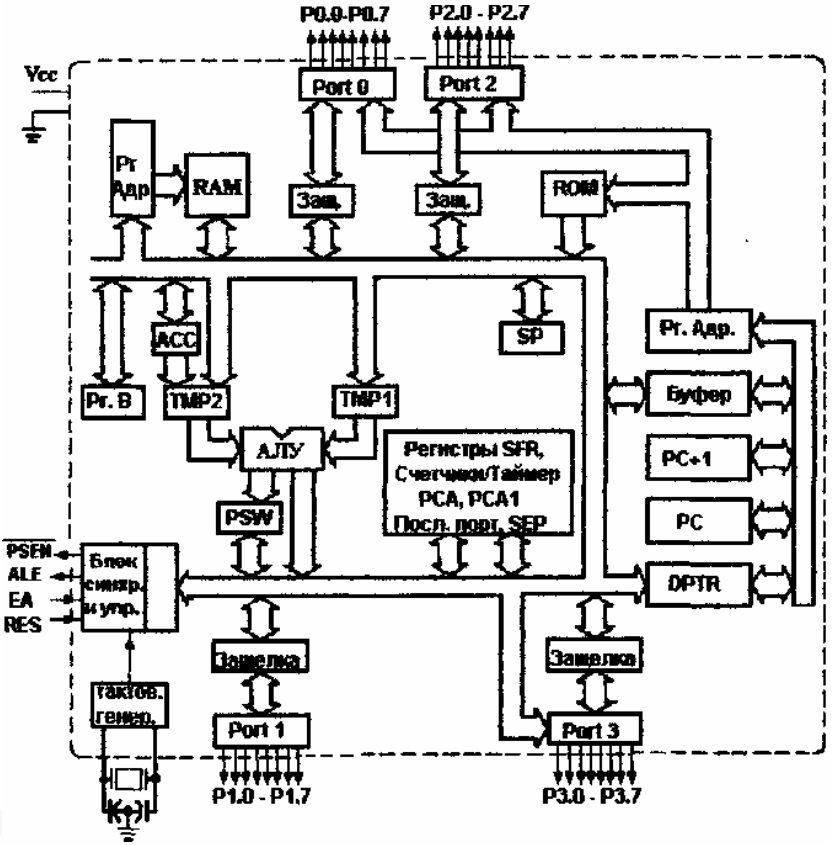


Рисунок 10.2 - Структурная схема К1816

Организация портов ввода

Количество портов - 4. Название – P0–P3, они адресуются как регистры специальных функций. Разрядность - 8 с возможностью побитной адресации разрядов. Каждый из портов содержит регистр-защелку (SFR P0 — SFR P3), выходную цепь и входной буфер.

Направление обмена информацией через порты - все порты двунаправленные, причем имеется возможность в каждом порту часть разрядов использовать для ввода данных, а часть для вывода.

Альтернативные функции. Из-за ограниченного количества выводов корпуса ИМС микроконтроллера большинство выводов используемся для выполнения двух функций : в качестве линий портов и для альтернативных функций.

Порты P0 и P2 используются при обращении к внешней памяти. При этом на выходах P0 младший байт адреса внешней памяти мультиплексируется с вводимым/выводимым байтом. Выходы P2 содержат старший байт адреса внешней памяти, если адрес 16-разрядный. При использовании восьмиразрядного, адреса порт P2 можно применять для ввода-вывода информации обычным образом. При обращении к внешней памяти в P0 автоматически заносятся во все биты. Информация в P2 при этом остается неизменной.

Порт P3 помимо обычного ввода и вывода информации используется для формирования, приема специальных управляющих и информационных сигналов. Разряды порта (все или частично) при этом могут выполнять альтернативные функции. Альтернативные функции могут быть активированы только в том случае, если в соответствующие биты порта P3 предварительно занесены 1. Неиспользуемые альтернативным образом разряды могут

работать как обычно. Выводы порта Р3 и их функциональное назначение представлено на табл. 10.2.

Таблица 10.2- Выводы порта Р3, их назначение

Вывод порта	Альтернативная функция
P3.0	RXD - вход последовательного порта
P3.1	TXD - выход последовательного порта
P3.2	INT0 - внешнее прерывание 0
P3.3	INT1 - внешнее прерывание 1
P3.4	T0 - вход таймера-счетчика 0
P3.3	T1 - вход таймера-счетчика 1
P3.6	IWR - строб записи во внешнюю память данных
P3.7	RD - строб чтения из внешней памяти данных

Память программ (ПЗУ). Как и у большинства микроконтроллеров, у микроконтроллеров семейства 8051 память программ и память данных являются самостоятельными и независимыми друг от друга устройствами, адресуемыми различными командами и управляющими сигналами.

Объем встроенной памяти программ, расположенной на кристалле микроконтроллеров 8051 и 8751, равен 4 Кбайт. При обращении к внешней памяти программ все микроконтроллеры семейства 8051 всегда используют 16-разрадный адрес, что обеспечивает им доступ к 64 Кбайт ПЗУ. Микроконтроллер обращается к программной памяти при чтении кода операции и операндов, используя счетчик команд PC, а также при выполнении команд переноса байта из памяти программ в аккумулятор. При выполнении команд переноса данных адресация ячейки памяти программ, из которой будут прочитаны данные, может осуществляться с использованием как счетчика PC, так и специального двухбайтового регистра-указателя данных DPTR.

Память данных (ОЗУ). Объем расположенной на кристалле памяти данных—128 байт. Объем внешней памяти

данных может достигать 64 Кбайт. Первые 32 байта организованы в четыре банка регистров общего назначения, обозначаемых соответственно банк 0 — банк 3. Каждый из них состоит из восьми регистров RO — R7. В любой момент программе доступен только один банк регистров, номер которого содержится в третьем и четвертом битах слова состояния программы PSW.

Оставшееся адресное пространство может конфигурироваться разработчиком по своему усмотрению: в нем располагаются стек, системные и пользовательские области данных. Обращение к ячейкам памяти данных возможно двумя способами. Первый способ — прямая адресация ячейки памяти. В этом случае адрес ячейки является операндом соответствующей команды. Второй способ — косвенная адресация с помощью регистров RO или R1. перед выполнением соответствующей команды в один из них должен быть занесен адрес ячейки, к которой необходимо обратиться.

Для обращения к внешней памяти данных используется только косвенная адресация с помощью регистров RO и R1 или с помощью 16-разрядного регистра-указателя DPTR. Он относится к группе регистров специальных функций и с его помощью можно адресовать все 64 Кбайта внешней памяти.

Часть памяти данных представляет собой так называемую битовую область, в ней имеется возможность при помощи специальных битовых команд адресовываться к каждому разряду ячеек памяти. Адрес прямоадресуемых битов может быть записан либо в виде (Адресбайта - Разряд), например, выражение 21.3 означает третий разряд ячейки памяти с адресом 21H, либо в виде абсолютного битового адреса.

Аккумулятор ACC является источником операнда и местом фиксации результата при выполнении ряда операций.

Только с использованием аккумулятора могут быть выполнены операции сдвига, проверки на нуль и ряд других.

Регистр-указатель стека SP в микроЭВМ рассматриваемого семейства — восьмибитный. Он может адресовать любую область внутренней памяти данных. В отличие от микропроцессора КР580ВМ80 у микроЭВМ семейства 1816 стек «растет вверх», т.е. перед выполнением команды PUSH или CALL содержимое SP инкрементируется, после чего производится запись информации в стек. Соответственно при извлечении информации из стека регистр SP декрементируется после извлечения информации. В процессе инициализации микроЭВМ после сигнала сброса или при включении питающего напряжения в SP заносится код 07Н. Это означает, что первый элемент стека будет располагаться в ячейке памяти с адресом 08Н.

Центральное процессорное устройство включает в себя устройство синхронизации и управления, АЛУ и схему условных переходов, которая организует условные переходы по битам регистра флажков АЛУ и внешним сигналам управления ТО и Т1. АЛУ строится аналогично МП К580 — оно выполняет операции накапливающего типа в двоичной и десятичной арифметике.

При выполнении ряда команд в арифметико-логическом устройстве (АЛУ) формируются признаки операций — флаги, которые фиксируются в **регистре PSW.**

Регистры специальных функций. К адресному пространству памяти данных примыкает адресное пространство регистров специальных функций (SFR) (табл. 10.3).

Таблица 10.3- Адреса памяти данных

Адрес	Символ	Наименование
OEOH	*ACC	Аккумулятор
OFOH	*B	Регистр расширитель аккумулятора
ODOH	*PSW	Слово состояния программы
080H	*PO	Порт 0 (SFR PO)
090H	*P1	noprl(SFRPl)
OAOH	*P2	Порт 2 (SFR P2)
OBOH	*P3	nopr3(SFRP3)

Продолжение табл. 10.3

081H	SP	Регистр указатель стека
083H	DPH	Старший байт регистра указателя данных DPTR
082H	DPL	Младший байт регистра указателя данных DPTR
08CH	THO	Старший байт таймера 0
08AH	TLO	Младший байт таймера 0
08DH	TH1	Старший байт таймера 1
08BH	TL1	Младший байт таймера 1
089H	TMOD	Регистр режимов таймеров счетчиков
088H	*TCON	Регистр управления статуса таймеров
OB8H	*IP	Регистр приоритетов
OA8H	*IE	Регистр маски прерывания
087H	PCON	Регистр управления мощностью
098H	*SCON	Регистр управления приемопередатчиком
099H	SBUF	Буфер приемопередатчика

Устройство управления и синхронизации К1816. К выводам XI и X2 микроЭВМ подключают кварцевый резонатор, вырабатываемые им сигналы синхронизуют всю работу микроЭВМ.

Устройство управления К1816 формирует машинный цикл фиксированной длительности, равной 12 периодам колебаний кварцевого резонатора или шести состояниям управляющего устройства. Каждое состояние содержит две фазы сигналов резонатора (Р1 и Р2). В фазе Р1, как правило, выполняется операция в АЛУ, а в фазе Р2 — межрегистровая передача. Весь машинный цикл состоит из 12 фаз. Команды различной степени сложности выполняются за разное количество циклов, возможны, в частности, и следующие варианты:

1 байт/1 цикл, например, INC A;

2 байта/1 цикл, например, ADD A, #data;

1 байт/2 цикла, например, INC DPTR;

1 байт/4 цикла, например, DIV AB и другие.

Кроме того, устройство управления и синхронизации формирует сигналы управления внешней памятью команд и данных.

Регистр-указатель данных DPTR чаще всего используют для фиксации 16-битного адреса в операциях обращения к внешней памяти программ и данных. С точки зрения программиста он может выступать как в виде одного 16-битного регистра, так и в виде двух независимых регистров DPL и DPH.

Рассмотрим более подробно ALU (рис.10.2). Это восьмиразрядное устройство, предназначенное для выполнения арифметических операций (сложение, вычитание, умножение и деление), логических операций (И, ИЛИ, исключающее ИЛИ), а также операций циклического сдвига, сброса, инвертирования и т.д. Ко входам ALU подключены программно-недоступные регистры T1 и T2, предназначенные для временного хранения операндов, схема десятичной коррекции (DCU) и схема формирования признаков результата операции (PSW).

Простейшая операция сложения используется в ALU для инкрементирования содержимого регистров, продвижения регистра-указателя данных (RAR) и автоматического вычисления следующего адреса резидентной памяти программ (RPM). Простейшая операция вычитания используется в ALU для декрементирования регистров и сравнения переменных.

Простейшие операции автоматически образуют «тандемы» для выполнения таких операций, как, например, инкрементирование 16- битных регистровых пар. В ALU реализуется механизм каскадного выполнения простейших операций для реализации сложных команд. Так, например, при выполнении одной из команд условной передачи управления по результату сравнения в ALU трижды инкрементируется счетчик команд (PC), дважды производится чтение из резидентной памяти данных (RDM), выполняется арифметическое сравнение двух переменных, формируется 16-битный адрес перехода и принимается решение о том, делать или не делать переход по программе. Все перечисленные операции выполняются за 2 мкс.

Важной особенностью ALU является его способность оперировать не только байтами, но и битами. Отдельные программно-доступные биты могут быть установлены, сброшены, инвертированы, переданы и использованы в логических операциях.

Приведём структурную схему (рис.10.3) и список сигналов однокристального 8-разрядного микроконтроллера КМ1816ВЕ48, в котором, в отличие от КМ1816ВЕ51, все обозначения сигналов и блоков приведены на русском языке, что облегчает понимание аббревиатур английского языка.

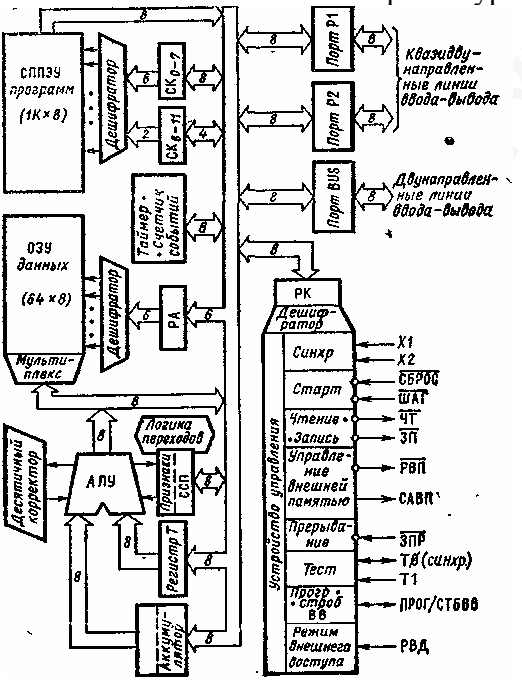


Рисунок 10.3 - Структурная схема микроконтроллера КМ1816ВЕ48

Список сигналов однокристального МК КМ1816ВЕ48

ЗЕМЛЯ (20) —потенциал земли.

$U_{осн}$(40)—основное напряжение питания +5 В подеется во время работы и при программировании С11ПЗУ.

$U_{доп}$ (26) —дополнительное напряжение питания -}-5 В во время работы МК обеспечивает электропитание только для СППЗУ; на этот вывод при программировании СППЗУ подается уровень +25 В.

ПРОГ/СТБВВ (25)—вход для подачи программирующего импульса +25 В при загрузке СППЗУ; выход стробирующего сигнала УВВ во время работы.

XI (2) —вход для подключения вывода кварцевого осциллятора или вход для сигнала от внешнего источника синхронизации.

X2 (3) — вход для подключения второго вывода осциллятора.

$\overline{\text{СБРОС}}$ (4)—вход сигнала общего сброса при запуске МК; сигнал 0 при программировании и проверке СППЗУ.

$\overline{ШАГ}$ (5)—сигнал, который совместно с сигналом САВП позволяет выполнять программу с остановом после исполнения очередной команды; используется на этапе отладки.

$\overline{РВП}$ (9) — разрешение внешней памяти; сигнал подастся на вход разрешения буфера внешней памяти только тогда, когда происходит обращение к внешней памяти программ.

САВП (11)—строб адреса внешней памяти; сигнал используется для приема и фиксации адреса внешней памяти на внешнем регистре, сигнал является идентификатором машинного цикла, так как всегда выводится из МК с частотой, в 5 раз меньшей основной частоты синхронизации.

$\overline{ЧТ}$ (8) — стробирующий сигнал при чтении из внешней памяти или УВВ.

$\overline{ЗП}$ (10) — стробирующий сигнал при записи во внешнюю память данных или УВВ.

ТО (1)—входной сигнал, опрашиваемый по командам условного перехода JTO и JNTO; используется при

программировании СППЗУ; может быть использован для вывода сигнала синхронизации по команде ENTO CLK.

T1 (39)—входной сигнал, опрашиваемый командами условного перехода JT1 и JNT1; кроме того, используется в качестве входа внутреннего счетчика внешних событий после исполнения команды STRT CNT.

ника; вызывает подпрограмму обслуживания прерывания, если прерывание разрешено ранее по команде ENI; сигнал СБРОС запрещает прерывания.

РВД (7)—режим внешнего доступа; уровень 1 на этом входе заставляет ЛЩ выполнять выборку команд только из внешней памяти программ; используется при тестировании прикладной программы и отладке МК.

Порт$P1_0$-7(27—34) —8-разрядный квазидвунаправленный порт ввода — вывода информации; каждый разряд порта может быть запрограммирован на ввод или на вывод.

Порт $P2_C$-;(21—24. 35—38)—8-разрядный квазидвунаправленный порт ввода — вывода информации; каждый разряд порта может быть запрограммирован на ввод или вывод; биты $P2_0$-з во время чтения из внешней памяти программ содержат старшие четыре разряда счетчика команд СК в-п, во время ввода — вывода используются для схемы расширения ввода — вывода и адресуют внешние порты Р4—Р7.

Порт BUS_0-7(12—19) — 8-разрядный двунаправленный порт в

нагрузки; может выполнять прием и выдачу байтов синхронно с сигналами $\overline{ЧТ}$ и $\overline{ЗП}$; при обращении к внешнему ППЗУ программ содержит 8 младших разрядов счетчика команд и затем по сигналу РВП принимает нную команду; при обращении к внешнему ОЗУ данных содержит младшие 8 разрядов адреса синхронно с сигналом САВП байт данных синхронно с сигналами $\overline{ЗП}$.

ЛЕКЦИЯ 11. ОСОБЕННОСТИ ФУНКЦИОНИРОВАНИЯ МИКРОКОНТРОЛЛЕРА

1 Работа с внешней памятью микроконтроллера. Система прерываний. Последовательный порт.

2 Функционирование таймеров/счётчиков микроконтроллера.

Работа с внешней памятью

Обращения к внешней памяти подразделяются на обращения к внешней памяти программ и обращения к внешней памяти данных. В первом случае для формирования сигнала, активирующего ПЗУ с программой, используется сигнал PSEN, во втором — сигналы RD и WR, активизирующие ОЗУ с данными.

Если используется 16-битовый адрес, старшие восемь бит выводятся через порт P2, где они сохраняются в течение всего цикла обращения к внешней памяти. Отметим, что выходные каскады порта P2 имеют внутреннюю нагрузку, несколько отличающуюся от P1 и P3, благодаря чему в SFR P2 при выводе адресной информации вовсе не обязательно защелкивать все единицы. Добавим также, что при выводе адресной информации информация из SFR P2 хотя и не присутствует на выводах микроЭВМ, но и не теряется, восстанавливаясь на них после окончания обращений к внешней памяти (если в процессе этих обращений SFR P2 не был модифицирован).

Если при обращении к внешней памяти данных используется восьмибитный адрес, то на выводах порта остается та же информация, которая там была до начала обращения к внешней памяти. Это позволяет организовать постраничную адресацию внешней памяти данных.

Как уже отмечалось, на выводах порта P0 младший байт адреса мультиплексируется с данными. Сигналы адреса/данных задействуют оба полевых транзистора выходного

каскада порта P0. Таким образом, в этом случае выводы P0 уже не являются выводами с открытым стоком и не требуют внешних нагрузочных элементов.

Сигнал ALE используется для фиксации младшего байта адреса во внешнем регистре-защелке. Адресная информация достоверна в момент окончания сигнала ALE.

Выводимый в цикле записи байт заносится в P0 непосредственно перед активизацией сигнала WR и остается неизменным до окончания этого сигнала. В цикле чтения данные на выводах P0 для достоверного считывания должны быть установившимися к моменту окончания сигнала RD.

Во время обращения к внешней памяти CPU записывает OFFH в SFR P0, уничтожая, таким образом, хранимую там информацию. Таким образом, использовать для записи порт P0 при работе с внешней памятью надо с известной долей осторожности.

Обращение к внешней памяти программ возможно в двух случаях:

• когда сигнал EA активен, т.е. имеет нулевой уровень;

• когда программный счетчик PC содержит число больше OFFH.

Следовательно, при использовании микроЭВМ, не имеющей встроенного ПЗУ или не использующей его, на входе EA должен присутствовать сигнал с нулевым уровнем.

Когда CPU работает с внешней памятью программ, все линии порта P2 используются для вывода старшего байта адреса и не могут быть использованы для обычного ввода\вывода информации. При этом, как отмечалось выше, в SFR P2 может быть занесена любая адресная информация, выводимая через P2, которая не зависит от состояния его SFR.

Система прерываний микроконтроллера 8051

Упрощенная схема прерываний микроЭВМ 8051 показана на рис.11.1.

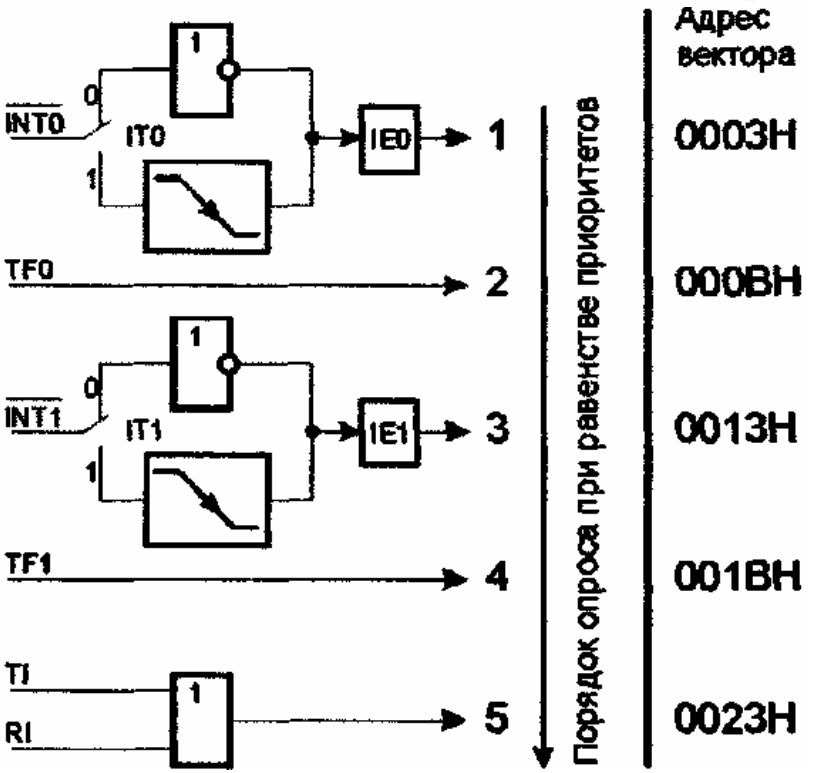


Рисунок 11.1 - Упрощённая схема прерываний

Внешние прерывания INT 0 и INT 1 могут быть вызваны либо уровнем, либо переходом сигнала из 1 в О на входах 8051 в зависимости от значений управляющих бит ITO и IT1 в регистре TCON. От внешних прерываний устанавливаются флаги IE0 и IE1 в регистре TCON, которые инициируют вызов соответствующей программы обслуживания прерывания. Сброс этих флагов выполняется аппаратно только в том случае, если прерывание было вызвано по переходу (срезу) сигнала. Если же прерывание

вызвано уровнем входного сигнала, то сбросом флага I должна управлять соответствующая подпрограмма обслуживания прерывания путем воздействия на источник прерывания с целью снятия им запроса.

Флаги запросов прерывания от таймеров TFO и TF1 сбрасываются автоматически при передаче управления подпрограмме обслуживания. Флаги запросов прерывания Rl и TI устанавливаются блоком управления приемопередатчика аппаратно, но сбрасываться должны программным путем.

Прерывания могут быть вызваны или отменены программой, так как все названные флаги программно доступны и могут быть установлены/ сброшены программой с тем же результатом, как если бы они были установлены/сброшены аппаратными средствами.

В блоке регистров специальных функций есть два регистра, предназначенных для управления режимом прерываний IE и уровнями приоритета IP. Возможность программной установки/сброса любого управляющего бита в этих двух регистрах делает систему прерываний 8051 исключительно гибкой.

В более сложных модификациях микроконтроллеров семейства MCS-51 количество периферийных устройств увеличено, что приводит к необходимости использовать один вектор прерывания для нескольких устройств (разделение подпрограмм обслуживания прерываний в этом случае необходимо реализовать программно) либо добавить еще два регистра - режима (маски) и приоритета прерываний.

Рассмотрим регистр масок прерывания (IE) и регистр приоритетов прерываний. Их символы, позиции, имя и назначение приведены в табл.11.1 и табл.11.2.

Таблица 11.1- Регистр масок прерываний

Символ	Позиция	Имя и назначение
EA	IE.7	Снятие блокировки прерывания. Сбрасывается программно для запрета всех прерываний независимо от состояний IE.4 – IE.O
	IE.6	Не используется
	IE.5	Не используется
ES	IE.4	Бит разрешения прерывания от приемопередатчика Установка/сброс программой для эазрешения/запрета прерываний от флагов TI или RI
ET1	IE.3	Бит разрешения прерывания от таймера. Установка/сброс программой для эазрешения/запрета прерываний от таймера 1
EX1	IE.2	Бит разрешения внешнего прерывания 1 Установка/сброс программой для разрешения/запрета прерывания 1
ETO	IE.1	Бит разрешения прерывания от таймера 0. Установка/сброс программой для разрешения/запрета прерываний от таймера 0
EXO	IE.O	Бит разрешения внешнего прерывания 0. Установка/сброс программой для разрешения/запрета прерывания 0

Таблица 11.2 - Регистр приоритетов прерываний (IP)

Символ	Позиция	Имя и назначение
-	IP7- IP.5	Не используется
PS	IP.4	Бит приоритета приемопередатчика. Установка/сброс программой для присваивания прерыванию от приемопередатчика высшего/низшего приоритета
PT1	IP.3	Бит приоритета таймера 1. Установка/сброс программой для присваивания прерыванию от таймера 1 высшего/низшего приоритета
PX1	IP.2	Бит приоритета внешнего прерывания 1. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию INT1
PTO	IP.1	Бит приоритета таймера 0. Установка/сброс программой для присваивания прерыванию от таймера 0 высшего/низшего приоритета
PXO	IP.O	Бит приоритета внешнего прерывания 0. Установка/сброс программой для присваивания высшего/низшего приоритета внешнему прерыванию NTO

Выполнение подпрограммы прерывания

Система прерываний формирует аппаратный вызов (LCALL) соответствующей подпрограммы обслуживания, если она не заблокирована одним из следующих условий:

• в данный момент обслуживается запрос прерывания равного или высокого уровня приоритета;

• текущий машинный цикл — не последний в цикле выполняемой команды;

• выполняется команда RETI или любая команда, связанная с обращением к регистрам IE или IP.

Отметим, что если флаг прерывания был установлен, но по одному из указанных выше условий не получил обслуживания и к моменту окончания блокировки уже сброшен, то запрос прерывания теряется и нигде не запоминается.

По аппаратно сформированному коду LCALL система прерывания помещает в стек только содержимое счетчика команд (PC) и загружает в него адрес вектора соответствующей подпрограммы обслуживания. По адресу вектора должна быть расположена команда безусловной передачи управления (JMP) к начальному адресу подпрограммы обслуживания прерывания. В случае необходимости она должна начинаться командами записи в стек (PUSH) слова состояния программы (PSW), аккумулятора, расширителя, указателя данных и т.д. и должна заканчиваться командами восстановления из стека (POP). Подпрограммы обслуживания прерывания должны завершаться командой RETI, по которой в счетчик команд перезагружается из стека сохраненный адрес возврата в основную программу. Команда RET также возвращает управление прерванной основной программе, но при этом не снимает блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

Последовательный порт микроконтроллера 8051

Через универсальный асинхронный приемопередатчик UART (Universal Asynchronous Receiver-Transmitter) осуществляются прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена. В состав приемопередатчика, называемого часто **последовательным**

портом, входят принимающий и передающий сдвигающие регистры, а также специальный буферный регистр (SBUF) приемопередатчика.

Кроме того, работой последовательного порта управляют два служебных регистра:

• Регистр управления/статуса приемопередатчика SCON.

• Бит SMOD регистра управления мощностью PCON.

Запись байта в буфер приводит к автоматической переписи байта в сдвигающий регистр передатчика и инициирует начало передачи байта. Наличие буферного регистра приемника позволяет совмещать операцию чтения ранее принятого байта с приемом очередного. Но если к моменту окончания приема байта предыдущий не был считан из SBUF, то он будет потерян.

Последовательный порт 8051 может работать в четырех различных режимах.

- **Режим 0.** Информация и передается, и принимается через вывод входа приемника (RXi TXi). Принимаются или передаются 8 бит данных. Через вывод выхода передатчика (TXD) выдаются импульсы сдвига, которые сопровождают каждый бит. Частота передачи бита информации равна 1/12 частоты кварцевого резонатора
- **Режим 1.** В этом режиме передаются через вывод TXD или принимаются через RXD: 10 бит информации: старт-бит (0), 8 бит данных и стоп-бит (1). При приеме информации в бит RB8 регистра управления/статуса приемопередатчика SCON заносится стоп-бит. Скорость приема/передачи — величина переменная и задается таймером.
- **Режим 2.** В этом режиме через вывод TXD передаются или через RXD принимаются 11 бит информации: старт-бит, 8 бит данных, программируемый девятый бит и стоп-бит. При передаче девятый бит данных может

принимать значение 0 или 1 или, например, для повышения достоверности передачи путем контроля по четности в него может быть помещено значение признака паритета из слова состояния программы (PSW.O). При приеме девятый бит данных помещается в бит RB8 SCON, а стоп-бит, в отличие от режима 1, теряется. Частота приема/передачи выбирается программой и может быть равна либо 1/32, либо 1/64 частоты резонатора в зависимости от управляющего бита SMOD.

- **Режим 3** совпадает с режимом 2 во всех деталях, за исключением частоты приема/передачи, которая является величиной переменной и задается таймером.

Во всех случаях передача инициализируется инструкцией, в которой данные перемещаются в SBUF. Прием инициализируется при обнаружении перепада из 1 в 0 на входе приемника. При этом в режиме 0 этот переход должен сопровождаться выполнением условий R1 = 0 и REN= 1, а для остальных режимов - REN = 1.

Регистр управления/статуса приемопередатчика SCON

Управление режимом работы приемопередатчика осуществляется через специальный регистр с символическим именем SCON. Этот регистр содержит не только управляющие биты, определяющие режим работы последовательного порта, но и девятый бит принимаемых или передаваемых данных (RB8 и TB8) и биты прерывания приемопередатчика (R1 и T1).

Прикладная программа путем загрузки в старшие биты регистра SCON двухбитного кода определяет режим работы приемопередатчика. Во всех четырех режимах работы передача инициализируется любой командой, в которой буферный регистр SBUF указан как получатель байта. Как уже отмечалось, прием в режиме О осуществляется при

условии, что R1 = 0 и REN = 1, в остальных режимах - при условии, что REN = 1.

В бите TB8 программно устанавливается значение девятого бита данных, который будет передан в режиме 2 или 3. В бите RB8 в этих режимах фиксируется девятый принимаемый бит данных. В режиме 1 в бит RB8 заносится стоп-бит. В режиме 0 бит RB8 не используется.

Флаг прерывания передатчика TI устанавливается аппаратно в конце периода передачи стоп-бита во всех режимах. Соответствующая подпрограмма обслуживания прерывания должна сбрасывать бит TL.

Флаг прерывания приемника RI устанавливается аппаратно в конце периода приема восьмого бита данных в режиме 0 и в середине периода приема стоп-бита в режимах 1, 2 и 3. Подпрограмма обслуживания прерывания должна сбрасывать бит RI.

Приведём функциональное назначение регистра управления/ статуса приёмопередатчика (табл. 11.3).

Таблица 11.3 - Назначение бит регистра управления/статуса

<table>
<tr><td>Символ</td><td>Позиция</td><td colspan="3">Имя и назначение</td></tr>
<tr><td>SMO</td><td>SCON.7</td><td colspan="3">Биты управления режимом работы приемопередатчика.
Устанавливаются/сбрасываются программно</td></tr>
<tr><td rowspan="4"></td><td rowspan="4">SCON.6</td><td>SMO</td><td>SM1</td><td>Режим работы приемопередатчика</td></tr>
<tr><td>0</td><td>0</td><td>Сдвигающий регистр расширения ввода/вывода</td></tr>
<tr><td>0</td><td>1</td><td>8 битовый приемопередатчик, изменяемая скорость передачи</td></tr>
<tr><td>1</td><td>0</td><td>9 битовый приемопередатчик. Фиксированная скорость передачи</td></tr>
</table>

		1	1	9 битовый приемопередатчик, изменяемая скорость передачи
SM2	SCON.5	Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение 0		
REN	SCON.4	Бит разрешения приема. Устанавливается/сбрасывается программно для разрешения/запрета приема последовательных данных		
TB8	SCON. 3	Передача бита 8. Устанавливается/сбрасывается программно для задания девятого передаваемого бита в режиме 9-битового передатчика		
RB8	SCON.2	Прием бита 8. Устанавливается/сбрасывается аппаратно для фиксации девятого принимаемого бита в режиме 9-битового приемника		
TI	SCON. 1	Флаг прерывания передатчика. Устанавливается аппаратно при окончании передачи байта. Сбрасывается программно после обслуживания прерывания		
RI	SCON.O	Флаг прерывания приемника. Устанавливается аппаратно при приеме байта. Сбрасывается программно после обслуживания прерывания		

Скорость приема/передачи информации через последовательный порт

Скорость приема/передачи, т.е. частота работы приемопередатчика в различных режимах, определяется различными способами.

В режиме 0 частота передачи зависит только от резонансной частоты кварцевого резонатора $f_{РЕЗ}$:

$$f=f_{РЕЗ}/12.$$

За машинный цикл последовательный порт передает один бит информации. В режимах 1, 2 и 3 скорость приема/передачи зависит от значения управляющего бита SMOD в регистре специальных функций PCON (табл.11.4).

Таблица 11.4- Регистр управления мощностью PCON

Символ	Позиция	Наименование и функция
SMOD	PCON.7	Удвоенная скорость передачи. Если бит установлен в 1 , то скорость передачи вдвое больше, чем при SMOD = 0. По сбросу SMOD = 0.
	PCON.6	Не используется
	PCON.5	Не используется
	PCON.4	Не используется
GF1 GFO	PCON.3 PCON.2	Флаги, специфицируемые пользователем (флаги общего назначения)
PD	PCON1	Бит пониженной мощности. При установке бита в 1 микроЭВМ переходит в режим пониженной потребляемой мощности
IDL	PCON.O	Бит холостого хода. Если бит установлен в 1 , то микроЭВМ переходит в режим холостого хода

Примечание. При одновременной записи 1 в PD и IDL бит PD имеет преимущество. Сброс содержимого PCON выполняется путем загрузки в него кода OXXXOOOO.

Таймеры/счетчики микроконтроллеров семейства 8051

В базовых моделях семейства имеются два программируемых 16-битных таймера/счетчика (Т/СО и Т/С1), которые могут быть использованы как в качестве таймеров, так и в качестве счетчиков внешних событий, В первом случае содержимое соответствующего таймера/счетчика (далее для краткости Т/С) инкрементируется в каждом машинном цикле, т.е. через каждые 12 периодов колебаний кварцевого резонатора, во втором-оно инкрементируется под воздействием перехода из 1 в О внешнего входного сигнала, подаваемого на соответствующий (ТО,Т1) вывод микроЭВМ 8051. Так как на распознавание периода требуются два машинных цикла, максимальная частота подсчета входных сигналов равна 1/24 частоты резонатора. На длительность периода входных сигналов ограничений сверху нет. Для гарантированного прочтения входной сигнал должен удерживать значение 1, как минимум, в течение одного машинного цикла микроЭВМ.

Для управления режимами работы Т/С и для организации их взаимодействия с системой прерываний используются два регистра специальных функций (TMOD и TCON), описание которых приведено ниже (табл.11.5, табл. 11.6).

Таблица 11.5 - Регистр режима работы таймера/счетчика TMOD

Символ	Позиция	Имя и назначение
GATE	TMOD.7 для Т/С1 и TMOD.3 для ТС/О	**Управление блокировкой.** Если бмт установлен, то таймер/счетчик “х” разрешен до тех пор, пока на входе “INTx” высокий уровень и бит управления “TRx” установлен. Если бит сброшен, то Т/С разрешается
С/Т	TMOD.6 для Т/С1 и TMOD.2 для Т/СО	**Бит выбора режима таймера или счетчика событий**. Если бит сброшен, то работает таймер от внутреннего источника сигналов синхронизации. Если установлен, то работает счетчик от внешних сигналов на входе “Тх”

		Режим работы		
M1	TMOD.5 для T/C1 и TMOD.1 для T/CO	**M1**	**M0**	
		0	0	Таймер BE48. "TLx" работает как 5-битный предделитель
MO	TMOD.4 для T/C1 и TMOD.O для T/CO	0	1	16 битный таймер/счетчик. THx" и "TLx" включен последовательно
		1	0	8-битный авто перезагружаемый таймер/счетчик. “THx”ранит значение, которое должно быть перезагружено в “TLx?” каждый раз по переполнению
		1	1	Таймер/счетчик 1 останавливается. Таймер/счетчик 0: TLO работает как 8-битный таймер/счетчик, и его режим определяется управляющими битами таймера 0. THO работает только как 8 битный таймер, и его режим определяется управляющими битами таймера 1

Таблица 11.6-Регистр управления/статуса таймера TCON

Символ	Позиция	Имя и назначение
TF1	TCON.7	**Флаг переполнения таймера 1.** Устанавливается аппаратно при переполнении таймера/счетчика. Сбрасывается при обслуживании прерывания аппаратно
TR1	TCON.6	**Управления таймера 1.** Устанавливается/сбрасывается программой для пуска/останова
TF0	TCON.5	флаг переполнения таймера 0. Устанавливается аппаратно. Сбрасывается при обслуживании прерывания

TR0	TCON.4	**Бит управления** анавливается / сбрасывается программой для пуска/останова таймера/сче
IE1	TCON.3	апп атн к сигнала Т обслуживан
IT1	TCON.2	**Бит управл** Ус ав а для ец и уровень)
	TCON.1	**Флаг фронта прерывания 0.** Устанавливается по срезу сигнала INTO. Сбрасывается при обслуживании прерывания
IT1	TCON.0	**Бит управл** Устанавлива но для ец и ий уровень)

Режимы работы таймеров-счетчиков

Как следует из описания управляющих бит TMOD, для обоих Т/С режимы работы 0, 1 и 2 одинаковы. Режимы 3 для Т/СО и Т/С работу Т/С в каждом из режимов (рис.11.2).

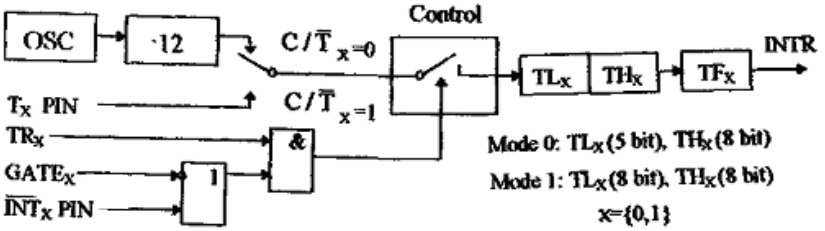


Рисунок 11.2а - Логика работы Т/СО я Т/С 1 в режимах 0 и 1

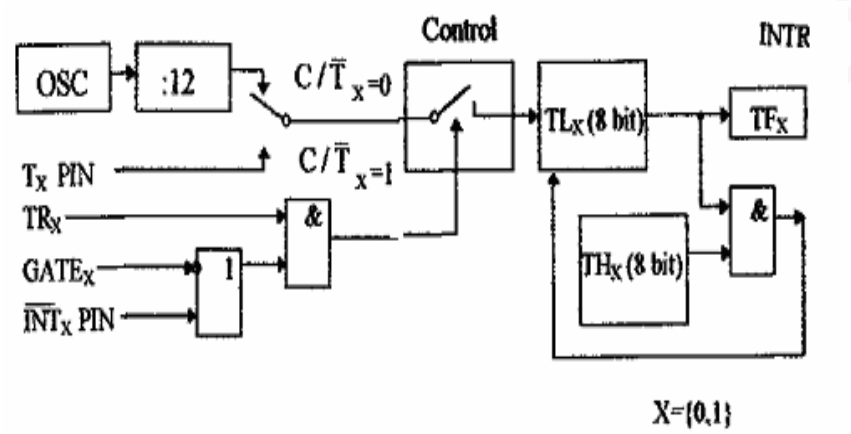


Рисунок 11.26 - Логика работы T/CO и T/C1 в режимах 0 и 1

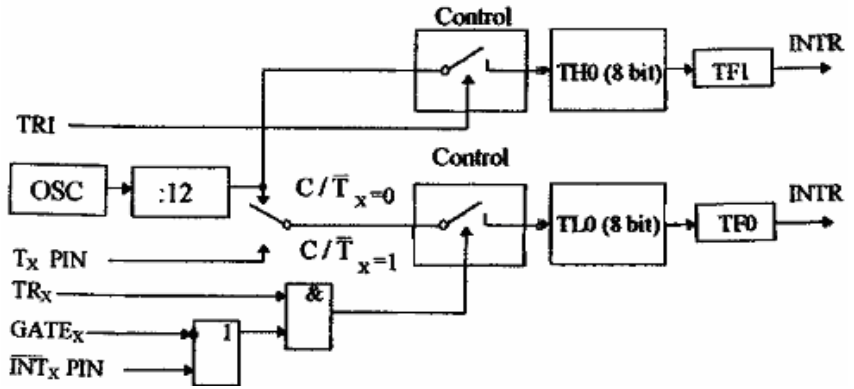


Рисунок 11.2 в - Логика работы T/CO и T/C1 в режимах 0, 1, 2 и 3

Режим 0. Перевод любого Т/С в режим 0 делает его похожим на таймер КМ1816ВЕ48 (восьмибитный счетчик), к входу которого подключен пяти-битный определитель частоты на 32. Работу Т/С в режиме О на примере Т/С1 иллюстрирует рис 11.2 а. В этом режиме таймерный регистр имеет разрядность 13 бит. При переходе из состояния "все единицы" в состояние "все нули" устанавливается флаг прерывания от таймера TF 1. Входной синхросигнал таймера 1 разрешен (поступает на вход Т/С1), когда управляющий бит TR1 установлен в 1 либо управляющий бит GATE (блокировка) равен 0, либо на внешний вывод запроса прерывания INT1 поступает уровень 1. Отметим попутно, что установка бита GATE в 1 позволяет использовать таймер для измерения длительности импульсного сигнала, подаваемого на вход запроса прерывания.

Режим 1. Работа любого Т/С в этом режиме такая же, как и в режиме 0, за исключением того, что таймерный регистр имеет разрядность 16 бит.

Режим 2. В этом режиме работа организована таким образом, что переполнение (переход из состояния "все единицы" в состояние, "все нули") восьмибитного счетчика TL1 приводит не только к установке флага TF1 (см. рис.11.26), но и автоматически перезагружает в TL1 содержимое старшего байта (TH 1) таймерного регистра, которое предварительно было задано программным путем. Перегрузка оставляет содержимое TH1 неизменным. В режиме 2 Т/СО и Т/С1 также работают совершенно одинаково.

Режим 3. В режиме 3 Т/СО и Т/С1 работают по-разному. Т/С1 сохраняет неизменным свое текущее содержимое. Иными словами, эффект такой же, как и при сбросе управляющего бита TR1 в 0. Работу Т/СО иллюстрирует рис.11.2 в. В режиме 3 TLO и THO функционируют как два независимых восьмибитных

счетчика. Работу TLO определяют управляющие биты T/CO (C/т, GATE TRO), входной сигнал INTO и флаг переполнения TFO. Работу THO, который может выполнять только функции таймера (подсчёт машинных циклов микроЭВМ), определяет управляющий бит TR1. При этом THO использует флаг переполнения TF1. Режим 3 используется в тех случаях, когда требуется наличие дополнительного восьмибитного таймера или счетчика событий. Можно считать, что в этом режиме микроЭВМ 8051 имеет в своем составе три таймера/счетчика. В случае же, если T/CO используется в режиме 3, T/C1 может быть или выключен, или переведен в режим 0, 1 или 2, или может быть использован последовательным портом в качестве генератора частоты передачи.

В модернизированных моделях микроконтроллеров семейства MCS-51 могут быть третий таймер/счетчик T/C2 и (или) блок программных счетчиков PCA, которые тоже могут быть использованы для отсчета временных интервалов.

ЛЕКЦИЯ 12 СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА

1 Общая характеристика системы команд, типы команд, группы команд, обозначения команд.

2 Краткое описание основных групп команд.

МикроЭВМ рассматриваемого семейства являются типичными микропроцессорными устройствами с архитектурой SISC - со стандартным набором команд. Поэтому их система команд довольно обширна и включает в себя 111 основных команд. Их длина - один, два или три байта, причем большинство из них (94%) -одно- или двухбайтные. Все команды выполняются за один или два машинных цикла (соответственно 1 или 2 мкс при тактовой частоте 12 МГц), исключение составляют команды умножения и деления, которые выполняются за четыре машинных цикла (4 мкс). МикроЭВМ семейства 8051 используют прямую, непосредственную, косвенную и неявную адресацию данных.

В качестве операндов команд микроЭВМ семейства 8051 могут использовать отдельные биты, четырехбитные цифры, байты и двухбайтные слова.

Все эти черты обычны для набора команд любого SISC-процессора и по сравнению с RISC -набором команд обеспечивают большую компактность программного кода и увеличение быстродействия при выполнении сложных операций.

В то же время набор команд семейства 8051 имеет несколько особенностей, связанных с типичными функциями, выполняемыми микроконтроллерами **управлением,** для которого типичным является оперирование с одноразрядными двоичными сигналами; большое число операций ввода/ вывода и ветвлений программы.

Наиболее существенная особенность системы команд рассматриваемых микроЭВМ -это возможность адресации отдельных бит в резидентной памяти данных. Кроме того, как отмечалось, некоторые регистры блока регистров специальных функций также допускают адресацию отдельных бит. Карты адресов отдельных бит находятся в резидентной памяти данных и в блоке регистров специальных функций.

Рассмотрим типы команд

Всего микроЭВМ выполняют 13 типов команд, они приведены в таблице 12.1. Как следует из нее, первый байт команды всегда содержит код операции (КОП), а второй и третий (если они присутствуют в команде) -адреса операндов или их непосредственные значения.

Таблица 12.1-Типы команд

Типы команд	Первый байт D7–D0	Второй байт D7–D0	Третий байт D7–D0
тип 1	коп		
тип 2	коп	#d	
тип 3	коп	ad	
тип 4	коп	bit	
тип 5	коп	rel	
тип 6	коп	a7–a0	
тип 7	коп	Ad	#d
тип 8	коп	Ad	rel
тип 9	коп	ads	add
тип 10	коп	#d	rel
тип 11	коп	bit	rel
тип 12	коп	ad16h	ad16I
тип 13	коп	#d16h	#d16I

Типы операндов

Состав операндов включает в себя операнды четырёх типов: биты, 4-битные цифры, байты и 16-битные слова.

Микроконтроллер имеет 128 программно-управляемых флагов пользователя. Имеется также возможность адресации отдельных битов блока регистров специальных функций и портов. Для адресации битов используется прямой 8-битный адрес (bit). Косвенная адресация битов невозможна. Карты адресов отдельных битов представлены в табл.12.2.

Четырёхбитные операнды используются только при операциях обмена SWAP и XCHD.

Восьмибитными операндами могут быть: ячейки памяти программ (ПП), резидентной памяти данных (РПД) или внешней памяти данных (ВПД); константы (непосредственные операнды); регистры специальных функций; порты ввода/вывода. Порты и регистры специальных функций адресуются только прямым способом. Байты памяти могут адресоваться также и косвенным способом через адресные регистры RO, R1, DPTR и PC.

Двухбайтные операнды - это константы и прямые адреса, для представления которых используются второй и третий байты команды.

Таблица 12.2- Карта адресов отдельных бит

Адреса	(D_7)							(D_0)
7FH								
2FH	7F	7E	7D	7C	7B	7A	79	78
2EH	77	76	75	74	73	72	71	70
2DH	6F	6E	6D	6C	6B	6A	69	68
2CH	67	66	65	64	63	62	61	60
2BH	5F	5E	5D	5C	5B	5A	59	58
2AH	57	56	55	54	53	52	51	50
29H	4F	4E	4D	4C	4B	4A	49	48
28H	47	46	45	44	43	42	41	40
27H	3F	3E	3D	3C	3B	3A	39	38
26H	37	36	35	34	33	32	31	30
25H	2F	2E	2D	2C	2B	2A	29	28
24H	27	26	25	24	23	22	21	20
23H	1F	1E	1D	1C	1B	1A	19	18
22H	17	16	15	14	13	12	11	10
21H	0F	0E	0D	0C	0B	0A	09	08
20H	07	06	05	04	03	02	01	00
1FH 18H	Банк 3							
17H 10H	Банк 2							
0FH 08H	Банк 1							
07H 00H	Банк 0							

Группы команд

Система команд семейства MCS-51 содержит 111 базовых команд, которые по функциональному признаку можно подразделить на пять групп:

- пересылки данных;
- арифметических операций;
- логических операций;
- операций над битами;
- передачи управления.

Форматы команд - одно-, двух- и трехбайтовые, причем большинство команд (94) имеют формат один или два байта. Первый байт любого типа и формата всегда содержит код операции, второй и третий байты содержат либо адреса операндов, либо непосредственные операнды.

Состав операндов включает в себя операнды четырех типов: биты, ниблы (4 разряда), байты и 16-битные слова. Время исполнения команд составляет 1, 2 или 4 машинных цикла. При тактовой частоте 12 МГц длительность машинного цикла составляет 1 мкс, при этом 64 команды исполняются за 1 мкс, 45 команд - за 2 мкс и 2 команды (умножение и деление) - за 4 мкс.

Набор команд MCS-51 поддерживает следующие режимы адресации.

Прямая адресация (Direct Addressing).Операнд определяется 8-битным адресом в инструкции. Эта адресация используется только для внутренней памяти данных и регистров SFR.

Косвенная адресация (Indirect Addressing) .В этом случае инструкция адресует регистр, содержащий адрес операнда. Данный вид адресации может применяться при обращении как к внутреннему, так и внешнему ОЗУ. Для указания 8-битных адресов могут использоваться регистры

RO и R1 выбранного регистрового банка или указатель стека SP.

Для 16-битной адресации используется только регистр "указатель данных" (DPTR - Data Pointer).

Регистровая адресация (Register Instruction). Данная адресация применяется для доступа к регистрам RO-R7 выбранного банка. Команды с регистровой адресацией содержат в байте кода операции трехбитовое поле, определяющее номер регистра. Выбор одного из четырех регистровых банков осуществляется программированием битов селектора банка (RS1, RSO) в PSW.

Непосредственная адресация (Immediate constants). Операнд содержится непосредственно в поле команды вслед за кодом операции и может занимать один или два байта (datae, dataie).

Индексная адресация (Indexed Addressing). Индексная адресация используется при обращении к памяти программ и только при чтении. В этом режиме осуществляется просмотр таблиц в памяти программ. 16-битовый регистр (DPTR или PC) указывает базовый адрес требуемой таблицы, а аккумулятор указывает на точку входа в нее. Адрес элемента таблицы находится сложением базы с индексом (содержимым аккумулятора).

Другой тип индексной адресации применяется в командах "перехода по выбору" (Case Jump). При этом адрес перехода вычисляется как сумма указателя базы и аккумулятора.

Неявная адресация (Register-Specific Instructions). Некоторые инструкции используют индивидуальные регистры (например, операции с аккумулятором, DPTR), при этом данные регистры не имеют адреса, указывающего на них; это заложено в код операции

Таблица 12.3- Команды передачи данных

Название команды	Мнемокод	КОП
Пересылка в аккумулятор из регистра (n=0-7)	MOVA,Rn	11101ПТ
Пересылка в аккумулятор прямоадресуемого байта	MOVA.ad	11100101
Пересылка в аккумулятор байта из РПД (i=0,1)	MOVA. @Ri	11100111
Загрузка в аккумулятор константы	MOVA,#d	01110100
Пересылка е регистр ия аккумулятора	MOVRn.A	11111ПТ
Пересылка в регистр прямоадресуемого байта	MOVRn.ad	10101m-
Загрузка в регистр константы	MOVRn.ffd	01111ПТ
Пересылка по прямому адресу аккумулятора	MOVad.A	11110101
Пересылка по прямому адресу регистра	MOVad.Rn	10001m-
Пересылка прямоадресуемого байта по прямому адресу	MOVadd.ads	10000101
Пересылка байта из РПД по прямому адресу	MOVad,@Ri	10000111
Пересылка по прямому адресу константы	MOVad,#d	01110101
Пересылка в РПД из аккумулятора	MOV@Ri,A	11110111
Пересылка в РПД прямоадресуемого байта	MOV@Ri,ad	0110011i
Пересылка в РПД константы	MOV@Rj,*d	0111011i
Загрузка указателя данных	MOVDPTR,*d16	10010000
Пересылка в аккумулятор байта из ПП	MOVC A,©A+DPTR	10010011
Пересылка в аккумулятор байта из ПП	MOVCA,@A+PC	10000011
Название команды	Мнемокод	КОП
Пересылка в аккумулятор байта из ВПД	MOVXA.@Ri	1110001i
Пересылка в аккумулятор байта из расширенной ВПД	MOVXA,@DPTR	11100000
Пересылка в ВПД из аккумулятора	MOVX@Ri,A	1111001i
Пересылка в расширенную ВПД из аккумулятора	MOVX ©OPTRA	11110000
Загрузка в стек	PUSH ad	11000000
Извлечение из стека	POP ad	11010000
Обмен аккумулятора с регистром	XCHA.Rn	11001rrr
Обмен аккумулятора с прямоадресуемым байтом	XCHA,ad	11000101
Обмен аккумулятора с байтом из РПД	XCHA,®Ri	1100011i
Обмен младших тетрад аккумулятора и байта РПД	XCHDA.@Ri	1101011i

По команде MOV выполняется пересылка данных из второго операнда в первый. Эта команда не имеет доступа ни к внешней памяти данных, ни к памяти программ. Для этих целей предназначены команды MOVX и MOVC соответственно. Первая из них обеспечивает чтение/запись байт из внешней памяти данных, вторая - чтение байт из памяти программ.

По команде XCH выполняется обмен байтами между аккумулятором и ячейкой РПД, а по команде XCHD - обмен младшими тетрадами (битами 0 - 3).

Команды PUSH и POP предназначены соответственно для записи данных в стек и их чтения из стека. Размер стека ограничен лишь размером резидентной памяти данных. В процессе инициализации микроЭВМ после сигнала сброса или при включении питающего напряжения в SP заносится код 07H. Это означает, что первый элемент стека будет располагаться в ячейке памяти с адресом 08H.

Группа команд пересылок микроконтроллера имеет следующую особенность - в ней нет специальных команд для работы со специальными регистрами PSW, таймером, портами ввода-вывода. Доступ к ним, как и к другим регистрам специальных функций, осуществляется заданием соответствующего прямого адреса, т.е. это команды обычных пересылок, в которых вместо адреса можно ставить название соответствующего регистра.

Например, чтение PSW в аккумулятор может быть выполнено командой MOV A, PSW, которая преобразуется Ассемблером к виду:

MOVA,ODOh(E5DO),

где E5 - код операции, а DO - операнд (адрес PSW).

Кроме того, следует отметить, что в микроЭВМ аккумулятор имеет два различных имени в зависимости от способа адресации: A - при неявной адресации (например, MOV A, RO) и ACC - при использовании прямого адреса. Первый способ предпочтительнее, однако не всегда применим.

Команды арифметических операций

В данную группу входят 24 команды (табл. 12.4). Из нее следует, что микроЭВМ выполняет достаточно широкий набор

команд для организации обработки данных, включая команды умножения и деления.

Таблица 12.4- Арифметические операции

Название команды	Мнемокод	КОП
Сложение аккумулятора с регистром (n=0-7)	ADDA,Rn	00101rrr
Сложение аккумулятора с прямо адресуемым байтом	ADD A, ad	0010010
Сложение аккумулятора с байтом из РПД (i=0,1)	ADDA,@Ri	0010011
Сложение аккумулятора с константой	ADDA,#d	0010010
Сложение аккумулятора с регистром и переносом	ADDCA,Rn	00111 rrr
Сложение аккумулятора с прямо адресуемым байтом и переносом	ADDCA,ad	0011010
Сложение аккумулятора с байтом из РПД и переносом	ADDCA,@Ri	0011011
Сложение аккумулятора с константой и переносом	ADDCA,#d	0011010
Десятичная коррекция аккумулятора	DA A	1101010
Вычитание из аккумулятора регистра и заема	SUBBA,Rn	10011 rrr
Вычитание из аккумулятора прямо адресуемого байта и заема	SUBBA,ad	1001010
Вычитание из аккумулятора байта РПД и заема	SUBBA, @Ri	1001011
Вычитание из аккумулятора константы и заема	SUBBA,d	1001010
Инкремент аккумулятора	INCA	0000010
Название команды	**Мнемокод**	**КОП**
Инкремент регистра	INCRn	00001 rrr
Инкремент прямо адресуемого байта	INCad	0000010
Инкремент байта в РПД	INC@Ri	0000011
Инкремент указателя данных	INCDPTR	1010001
Декремент аккумулятора	DEC A	0001010
Декремент регистра	DECRn	00011m
Декремент прямо адресуемого байта	DEC ad	0001010
Декремент байта в РПД	DECQRi	0001011
Умножение аккумулятора на регистр B	MULAB	1010010
Деление аккумулятора на регистр B	DIVAB	1000010

По результату выполнения команд ADD, ADDC, SUBB, MUL и DIV устанавливаются флаги PSW.

Флаг C устанавливается при переносе из разряда D7, т. е. в случае, если результат не помещается в восемь разрядов; флаг AC устанавливается при переносе из разряда D3 в командах сложения и вычитания и служит для реализации десятичной арифметики. Этот признак используется командой DAA.

Флаг OV устанавливается при переносе из разряда D6, т. е. в случае, если результат не помещается в семь разрядов и восьмой не может быть интерпретирован как знаковый. Этот признак служит для организации обработки чисел со знаком.

Наконец, флаг P устанавливается и сбрасывается аппаратно. Если число единичных бит в аккумуляторе нечетно, то P = 1, в противном случае P = 0.

Команды логических операций микроконтроллера

В этой группе 25 команд, их краткое описание приведено в табл. 12.5. Нетрудно видеть, что эти команды позволяют выполнять операции над байтами: логическое И (Л), логическое ИЛИ (V), исключающее ИЛИ (+), инверсию (NOT), сброс в нулевое значение.

Таблица. 12.5 - Логические операции

Название команды	Мнемокод	КС
Логическое И аккумулятора и регистра	ANLA,Rn	0101
Логическое И аккумулятора и прямоадресуемого байта	ANLA,ad	010K
Логическое И аккумулятора и байта из РПД	ANLA,@Ri	010K
Логическое И аккумулятора и константы	ANLA.M	010K
Логическое И прямоадресуемого байта и аккумулятора	ANLad.A	010K
Логическое И прямоадресуемого байта и константы	ANLad.fd	010K
Логическое ИЛИ аккумулятора и регистра	ORLA.Rn	0100
Логическое ИЛИ аккумулятора и прямоадресуемого байта	ORLA,ad	0100)
Логическое ИЛИ аккумулятора и байта из РПД	ORLA,@Ri	01 ОС

Продолжение табдицы 12.5

Логическое ИЛИ аккумулятора и константы	ORLA,#d	0100
Логическое ИЛИ прямоадресуемого байта и аккумулятора	ORLad.A	0100
Логическое ИЛИ прямоадресуемого байта и константы	ORLad,#d	0100
Исключающее ИЛИ аккумулятора и регистра	XRLA,Rn	0110
Исключающее ИЛИ аккумулятора и прямоадресуемого байта	XRLA,ad	0110
Исключающее ИЛИ аккумулятора и байта из РПД	XRLA,@Ri	0110
Исключающее ИЛИ аккумулятора и константы	XRLA.5M	0110
Исключающее ИЛИ прямоадресуемого байта и аккумулятора	XRLad.A	0110
Исключающее ИЛИ прямоадресуемого байта и константы	XRLad,#d	0110
Сброс аккумулятора	CLRA	1110
Инверсия аккумулятора	CPLA	1111
Сдвиг аккумулятора влево циклический	RLA	0010
Сдвиг аккумулятора влево через перенос	RLCA	0011
Сдвиг аккумулятора вправо циклический	RRA	0000
Сдвиг аккумулятора вправо через перенос	RRCA	OOOK
Обмен местами тетрад в аккумуляторе	SWAP A	1100

Команды операций над битами микроконтроллера

Группа состоит из 12 команд, краткое описание которых приведено в табл. 12.6. Эти команды позволяют выполнять операции над отдельными битами: сброс, установку, инверсию бита, а также логические И (Л) и ИЛИ (V). В качестве "логического" аккумулятора, участвующего во всех операциях с двумя операндами, выступает признак переноса С (разряд D7 PSW), в качестве операндов могут использоваться 128 бит из резидентной памяти данных и регистры специальных функций, допускающие адресацию отдельных бит.

Таблица 12. 6-Операции с битами

Название команды	Мнемокод	КОП	Т	Б	L
Сброс переноса	CLRC	11000011	1	1	1
Сброс бита	CLRbit	11000010	4	2	1
Установка переноса	SETBC	11010011	1	1	1
Установка бита	SETBbit	11010010	4	2	1
Инверсия переноса Инверсия бита	CPLC CPLbit	10110011 10110010	1 4	1 2	1 1
Логическое И битв и переноса	ANLC.bit	10000010	4	2	2
Логическое И инверсии бита и переноса	ANLC./bft	10110000	4	2	2
Логическое ИЛИ бита и переноса	ORLC.bit	01110010	4	2	2
Логическое ИЛИ инверсии бита и переноса	ORLC./bft	10100000	4	2	2
Пересыпка бита в перенос	MOVC.brt	10100010	4	2	1
Пересылка переноса в бит	MOVbitC	10010010	4	2	2

Команды передачи управления микроконтроллера

Группа представлена командами безусловного и условного переходов, командами вызова подпрограмм и командами возврата из подпрограмм(табл. 12.7).

Таблица 12. 7- Команды передачи управления

Название команды	Мнемокод	КОП	т	Б
Длинный переход в полном объеме ПП	LJMPad16	00000010	12	3
Абсолютный переход внутри страницы в 2Кб	AJMPad11	$a_{10}a_9a_8$00001	6	2
Короткий относительный переход внутри страницы в 256 байт	SJMPrel	10000000	5	2
Косвенный относительный переход	JMP@A+DPTR	01110011	1	1
Переход, если аккумулятор равен нулю	JZrei	01100000	5	2

Продолжение таблицы 12.7

Переход, если аккумулятор не равен нулю	JNZrel	01110000	5	2
Переход, если перенос равен единице	JCrel	01000000	5	2
Переход, если перенос равен нулю	JNCrel	01010000	5	2
Переход, если бит равен единице	JBbit, rel	00100000	11	3
Переход, если бит равен нулю	JNBbrt,rel	00110000	11	3

Название команды	**Мнемокод**	**КОП**	**Т**	**Б**
Переход, если бит установлен, с последующим сбросом бита	JBC bit, rei	00010000	11	3
Декремент регистра и переход, если не нуль	DJNZRn.rel	11011ПТ	5	2
Декремент прямоадресуемого байта и переход, если не нуль	DJNZad, rel	11010101	8	3
Сравнение аккумулятора с прямоадресуемым байтом и переход, если неравно	CJNEA, ad, rel	10110101	8	3
Сравнение аккумулятора с константой и переход, если не равно	CJNEA,fd, re)	10110100	10	3
Сравнение регистра с константой и переход, если не равно	CJNE Rn, *d, rel	10111m-	10	3
Сравнение байта в РПД с константой и переход, если не равно	CJNE @Ri,#d,re	10110111	10	3
Длинный вызов подпрограммы	LCALLadl6	00010010	12	3
Абсолютный вызов подпрограммы в пределах страницы в 2 Кб	ACALLad11	$a_{10}a_9a_8$000 01	6	2
Возврат из подпрограммы	RET	00100010	1	1
Возврат из подпрограммы обработки прерывания	RETI	00110010	1	1
Пустая операция	NOP	00000000	1	1

Команда безусловного перехода LJMP (L - long - длинный) осуществляет переход по абсолютному 16-битному адресу, указанному в теле команды, т. е. команда обеспечивает переход в любую точку памяти программ.

Действие команды AJMP (A - absolute -абсолютный) аналогично команде LJMP, однако в теле команды указаны лишь 11 младших разрядов адреса. Поэтому переход осуществляется в пределах страницы размером 2 Кбайта. При этом надо иметь в виду, что сначала содержимое счетчика команд увеличивается на 2 и только потом заменяется 11 разрядами адреса.

В отличие от предыдущих команд в команде SJMP (S - short - короткий) указан не абсолютный, а относительный адрес перехода. Величина смещения rel рассматривается как число со знаком, а, следовательно, переход возможен в пределах - 128...+127 байт относительно адреса команды, следующей за командой SJMP.

Команда косвенного перехода JMP @A+DPTR позволяет вычислять адрес перехода в процессе выполнения самой программы.

Командами условного перехода можно проверять следующие условия:

- JZ — аккумулятор содержит нулевое значение;
- JNZ — аккумулятор содержит ненулевое значение;
- JC — бит переноса C установлен;
- JNC — бит переноса C не установлен;
- JB — прямоадресуемый бит равен 1;
- JNB — прямоадресуемый бит равен 0;
- JBC — прямоадресуемый бит равен 1 и сбрасывается в нулевое значение при выполнении команды.

Все команды условного перехода, рассматриваемых микроЭВМ, содержат короткий относительный адрес, т.е. переход может осуществляться в пределах (от -128 до +127) байт относительно следующей команды.

Команда DJNZ предназначена для организации программных циклов. Регистр Rn или байт по адресу ad, указанные в теле команды, содержат счетчик повторений цикла, а смещение rel — относительный адрес перехода к началу цикла. При выполнении команды содержимое счетчика уменьшается на 1 и проверяется на 0. Если значение содержимого счетчика не равно 0, то осуществляется переход на начало цикла, в противном случае выполняется следующая команда.

Команда CJN удобна для реализации процедур ожидания внешних событий. В теле команды указаны "координаты" двух байт и относительный адрес перехода rel. В качестве двух байт могут быть использованы, например, значения содержимого аккумулятора и прямо адресуемого байта или косвенно адресуемого байта и константы. При выполнении команды значения указанных двух байт сравниваются, и в случае, если они не одинаковы, осуществляется переход. Например, команда

WAIT: CJNE A, PO, WAIT

будет выполняться до тех пор, пока значения на линиях порта PO не совпадут со значениями содержимого аккумулятора.

Действие команд вызова процедур полностью аналогично действию команд безусловного перехода. Единственное отличие состоит в том, что они сохраняют в стеке адрес возврата.

Команда возврата из подпрограммы RET восстанавливает из стека значение содержимого счетчика команд, а команда возврата из процедуры обработки прерывания RETI, кроме того, разрешает прерывание обслуженного уровня. Команды RET и RETI не различают, какой командой - LCALL или ACALL -была вызвана подпрограмма, так как и в том, и в другом случаях в стеке сохраняется полный 16-разрядный адрес возврата.

В заключение следует отметить, что большинство Ассемблеров допускают обобщенную мнемонику JMP -для команд безусловного перехода и CALL - для команд вызова подпрограмм.

ПРИЛОЖЕНИЕ А

(обязательное)

Система команд микропроцессора КР580ВМ80А

Таблица А.1- Команды пересылки

Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MOV R1, R2	Передача из R2 в R1	0	1	D	D	D	S	S	S	-	-	-	-	-
MOV M, R	Передача из R в память	0	1	1	1	0	S	S	S	-	-	-	-	-
MOV R, M	Передача из памяти в R	0	1	D	D	D	1	1	0	-	-	-	-	-
MVI R, byte	Передача байта в регистр	0	0	D	D	D	1	1	0	-	-	-	-	-
MVI M, byte	Передача байта в память	0	0	1	1	0	1	1	0	-	-	-	-	-
LXI RP, date	Загрузка парных регистров B-C, D-E, H-L, SP	0	0	R	R	0	0	0	1	-	-	-	-	-
LDAX RP	Загрузка аккумулятора по адресу в рег.B-C или D-E	0	0	R	R	1	0	1	0	-	-	-	-	-
STAX RP	Занесение содержимого аккумулятора по адресу в рег.B-C или D-E	0	0	R	R	0	0	1	0	-	-	-	-	-
LDA adr	Загрузка аккумулятора по адресу, указанному в команде	0	0	1	1	1	0	1	0	-	-	-	-	-
STA adr	Загрузка содержимого аккумулятора по адресу, указанному в команде	0	0	1	1	0	0	1	0	-	-	-	-	-

Продолжение приложения А

Продолжение таблицы А.1- Команды пересылки

LHLD adr	Загрузка рег.LH из двух соседних, начиная из адреса, указанного в команде	0	0	1	0	1	0	1	0	-	-	-	-	-
Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SHLD adr	Занесение содержимого рег.LH в две соседние ячейки, начиная из адреса указанного в команде	0	0	1	0	0	0	1	0	-	-	-	-	-
XCHG	Обмен данных между парами регистров H-L, D-E	1	1	1	0	1	0	1	1	-	-	-	-	-

Продолжение приложения А

Таблица А.2-Команды арифметических и логических операций

Мнемоника	Описание команды	Код команды									Флаги условий			
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADD R	Сложение рег. и аккумулятора	1	0	0	0	0	S	S	S	+	+	+	+	+
ADC R	Тоже, но с учетом переноса CY	1	0	0	0	1	S	S	S	+	+	+	+	+
ADD M	Сложение ячейки памяти и аккумулятора	1	0	0	0	0	1	1	0	+	+	+	+	+
ADC M	Тоже, но с учетом переноса CY	1	0	0	0	1	1	1	0	+	+	+	+	+
ADI, bite	Сложение байта с содержимым аккумулятора	1	1	0	0	0	1	1	0	+	+	+	+	+
ACI, bite	Тоже, но с учетом переноса CY	1	1	0	0	1	1	1	0	+	+	+	+	+
DAD RP	Сложение содержимого пар рег. B-C, H-L, D-E, SP с содержимым H-L	0	0	R	R	1	0	0	1	-	-	-	-	-
SUB R	Вічитание содержимого рег. из аккумуляторов	1	0	0	1	0	S	S	S	+	+	+	+	+
SBB R	Тоже, но с заемом	1	0	0	1	1	S	S	S	+	+	+	+	+
SUB M	Вычитание содержимого ячейки памяти из аккумулятора	1	0	0	1	0	1	1	0	+	+	+	+	+
SBB M	Тоже, но с заемом	1	0	0	1	1	1	1	0	+	+	+	+	+
SUI, bite	Вычитание байта из содержимого аккумулятора	1	1	0	1	0	1	1	0	+	+	+	+	+
SBI, bite	Тоже, но с учетом заема	1	1	0	1	1	1	1	0	+	+	+	+	+
INR R	Увеличение рег. на 1	0	0	D	D	D	1	0	0	+	+	+	+	-
INR M	Увеличение ячейки памяти на 1	0	0	1	1	0	1	0	0	+	+	+	+	-

Продолжение приложения А

Продолжение таблицы А.2- Команды арифметических и логических операций.

Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DCR R	Уменьшение рег. на 1	0	0	D	D	D	1	0	1	+	+	+	+	-
DCR M	Уменьшение ячейки памяти на 1	0	0	1	1	0	1	0	1	+	+	+	+	-
INX RP	Увеличение парных рег. B-C, H-L, D-E, SP на 1	0	0	R	R	0	0	1	1	-	-	-	-	-
DCX RP	Уменьшение парных рег. B-C, H-L, D-E, SP на 1	0	0	R	R	1	0	1	1	-	-	-	-	-
ANA R	Поразрядное лог. умножение рег. и аккумулятора	1	0	1	0	0	S	S	S	+	+	0	+	0
ANA M	Поразрядное лог. умножение содержимого ячейки памяти аккумулятора	1	0	1	0	0	1	1	0	+	+	0	+	0
ANI, bite	Поразрядное лог. умножение содержимого аккумулятора и байта	1	1	1	0	0	1	1	0	+	+	0	+	0
XRA R	Поразрядное исключающее ИЛИ над содержимым рег. или аккумулятора	1	0	1	0	1	S	S	S	+	+	0	+	0
XRA M	Поразрядное исключающее ИЛИ над содержимым ячейки памяти или аккумулятора	1	0	1	0	1	1	1	0	+	+	0	+	0

Продолжение приложения А

Продолжение таблицы А.2- Команды арифметических и логических операций.

Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
XRI, bite	Поразрядное исключающее ИЛИ над содержимым аккумулятора и байтом	1	1	1	0	1	1	1	0	+	+	0	+	0
ORA R	Поразрядное логическое сложение содержимого рег. и аккумулятора	1	0	1	1	0	S	S	S	+	+	0	+	0
ORA M	Поразрядное логическое сложение содержимого ячейки памяти и аккумулятора	1	0	1	1	0	1	1	0	+	+	0	+	0
ORI, bite	Поразрядное логическое сложение содержимого аккумулятора и байта	1	1	1	1	0	1	1	0	+	+	0	+	0
CMP R	Сравнение содержимого рег. и аккумулятора	1	0	1	1	1	S	S	S	+	+	+	+	+
CMP M	Сравнение содержимого ячейки памяти и аккумулятора	1	0	1	1	1	1	1	0	+	+	+	+	+
CPI, bite	Сравнение байта с содержимым аккумулятора	1	1	1	1	1	1	1	0	+	+	+	+	+
RLC	Циклический сдвиг содержимого аккумулятора влево	0	0	0	0	0	1	1	1	-	-	-	-	+
RRC	Тоже, но вправо	0	0	0	0	1	1	1	1	-	-	-	-	+
RAL	Циклический сдвиг содержимого аккумулятора влево через перенос	0	0	0	1	0	1	1	1	-	-	-	-	+

Продолжение приложения А

Продолжение таблицы А.2- Команды арифметических и логических операций.

Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	d0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RAR	Тоже, но вправо	0	0	0	1	1	1	1	1	-	-	-	-	+
CMA	Инвертирование аккумулятора	0	0	1	0	1	1	1	1	-	-	-	-	-
STC	Установка флага переноса CY в 1	0	0	1	1	0	1	1	1	-	-	-	-	1
CMC	Инвертирование флага переноса	0	0	1	1	1	1	1	1	-	-	-	-	C
DAA	Двоично-десятичное коррекция	0	0	1	0	0	1	1	1	+	+	+	+	+

Продолжение приложения А

Таблица А.3-Команды управления

Мнемоника	Описание команды	Код команды								Флаги условий				
		D7	d6	d5	d4	d3	D2	d1	D0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
JMP, adr	Безусловный переход при наличии переноса	1	1	0	0	0	0	1	1	-	-	-	-	-
JC, adr	Переход при наличии переноса	1	1	0	1	1	0	1	0	-	-	-	-	-
JNC, adr	Переход при отсутствии переноса	1	1	0	1	0	0	1	0	-	-	-	-	-
JZ, adr	Переход при нуле	1	1	0	0	1	0	1	0	-	-	-	-	-
JNZ, adr	Тоже при отсутствии нуля	1	1	0	0	0	0	1	0	-	-	-	-	-
JP, adr	Тоже при плюсе	1	1	1	1	0	0	1	0	-	-	-	-	-
JM, adr	Тоже при минусе	1	1	1	1	1	0	1	0	-	-	-	-	-
JPE, adr	Тоже при четности	1	1	1	0	1	0	1	0	-	-	-	-	-
JPO, adr	Тоже при нечетности	1	1	1	0	0	0	1	0	-	-	-	-	-
PCHL, adr	Занесение в счетчик команд содержимого H-L	1	1	1	0	1	0	0	1	-	-	-	-	-
PUSH RP	Ввод содержимого регистров B-C, D-E, H-L в стек	1	1	R	R	0	1	0	1	-	-	-	-	-
XTHL	Обмен данными между H-L и SP	1	1	1	0	0	0	1	1	-	-	-	-	-
SPHL	Занесение содержимого в рег. H-L в SP	1	1	1	1	1	0	0	1	-	-	-	-	-
PUSH PSW	Ввод PSW в стек	1	1	1	1	0	1	0	1	-	-	-	-	-
POP RP	Выдача данных из стека в рег. B-C, D-E, H-L	1	1	R	R	0	0	0	1	-	-	-	-	-

Продолжение таблицы А.3-Команды управления

Мнемоника	Описание команды	Код команды								Флаги условий				
		D7	d6	d5	d4	d3	D2	d1	D0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
POP PSW	Выдача данных из стека в аккумулятор и рег. признаков	1	1	1	1	0	0	0	1	+	+	+	+	+
CALL, adr	Вызов подпрограммы	1	1	0	0	1	1	0	1	-	-	-	-	-
CC, adr	Тоже при переносе	1	1	0	1	1	1	0	0	-	-	-	-	-
CNC, adr	Тоже при отсутствии переноса	1	1	0	1	0	1	0	0	-	-	-	-	-
CZ, adr	Тоже при нуле	1	1	0	0	1	1	0	0	-	-	-	-	-
CNZ, adr	Тоже но при отсутствии нуля	1	1	0	0	0	1	0	0	-	-	-	-	-
CP, adr	Вызов подпрограммы при плюсе	1	1	1	1	0	1	0	0	-	-	-	-	-
CM, adr	То же, но при минусе	1	1	1	1	1	1	0	0	-	-	-	-	-
CPE, adr	То же, при четности	1	1	1	0	1	1	0	0	-	-	-	-	-
CPO, adr	То же, но при нечетности	1	1	1	0	0	1	0	0	-	-	-	-	-
RET	Возврат	1	1	0	0	1	0	0	1	-	-	-	-	-
RC	Возврат при переносе	1	1	0	1	1	0	0	0	-	-	-	-	-
RNC	Возврат при отсутствии переноса	1	1	0	1	0	0	0	0	-	-	-	-	-
RZ	Возврат при нуле	1	1	0	0	1	0	0	0	-	-	-	-	-
RNZ	Возврат при отсутствии нуля	1	1	0	0	0	0	0	0	-	-	-	-	-
RP	Возврат при плюсе	1	1	1	1	0	0	0	0	-	-	-	-	-
RM	Возврат при минусе	1	1	1	1	1	0	0	0	-	-	-	-	-
RPE	Возврат при четности	1	1	1	0	1	0	0	0	-	-	-	-	-
RPO	Возврат при нечетности	1	1	1	0	0	0	0	0	-	-	-	-	-

Продолжение таблицы А.3-Команды управления

Мнемоника	Описание команды	Код команды								Флаги условий				
		d7	d6	d5	d4	d3	d2	d1	D0	S	Z	AC	P	CY
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RST	Повторный запуск	1	1	N	N	N	1	1	1	-	-	-	-	-
IN	Ввод	1	1	0	1	1	0	1	1	-	-	-	-	-
OUT	Вывод	1	1	0	1	0	0	1	1	-	-	-	-	-
EI	Разрешение прерывания	1	1	1	1	1	0	1	1	-	-	-	-	-
DI	Запретить прерывания	1	1	1	1	0	0	1	1	-	-	-	-	-
NOP	Отсутствие операции	0	0	0	0	0	0	0	0	-	-	-	-	-
HLT	Останов	0	1	1	1	0	1	1	0	-	-	-	-	-

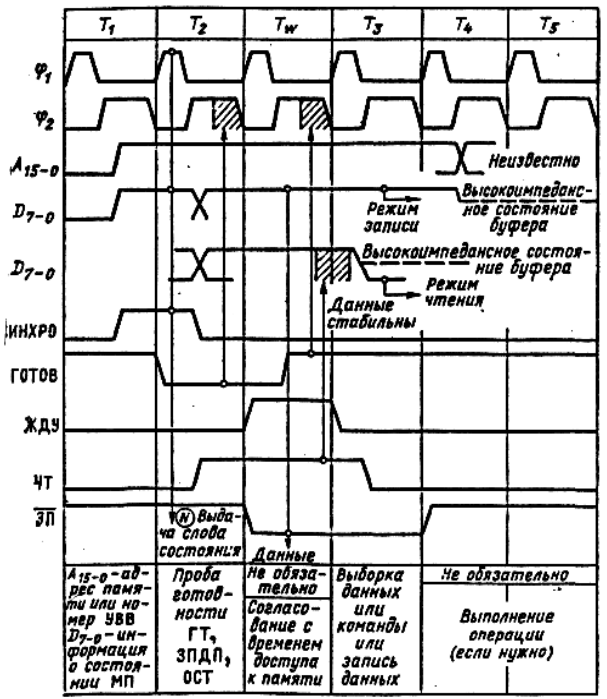
Примечание: 1 DDD, SSS – 3- разрядные поля, адресующие один из регистров общего назначения. 2 RR – 2-разрядные поля адресующие один из парных регистров. 3 RSW – слово состояния программы, первый байт которого равен содержимому A, а второй -содержимому RS. 4 NNN – двоичное представление номера команды RST. 5 Установка и сброс флага условия; б) - - отсутствие влияния на флаг.

Кодировка адресов регистров

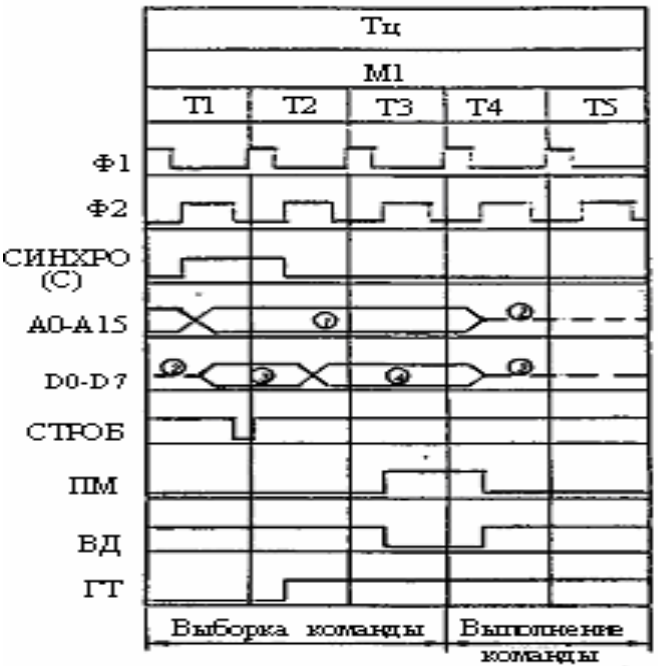
Рег.R	Код SSS,DDD	Рег.R	Код SSS,DDD	Парный рег.RP	Рег.RR
B	000	H	100	B-C	00
C	001	L	101	D-E	01
D	010	M	110	H-L	10
E	011	A	111	S-P	11

Продолжение приложения А

Временные диаграммы работы МП КР580



Машинные такты работ МП КР580



(1) Адрес

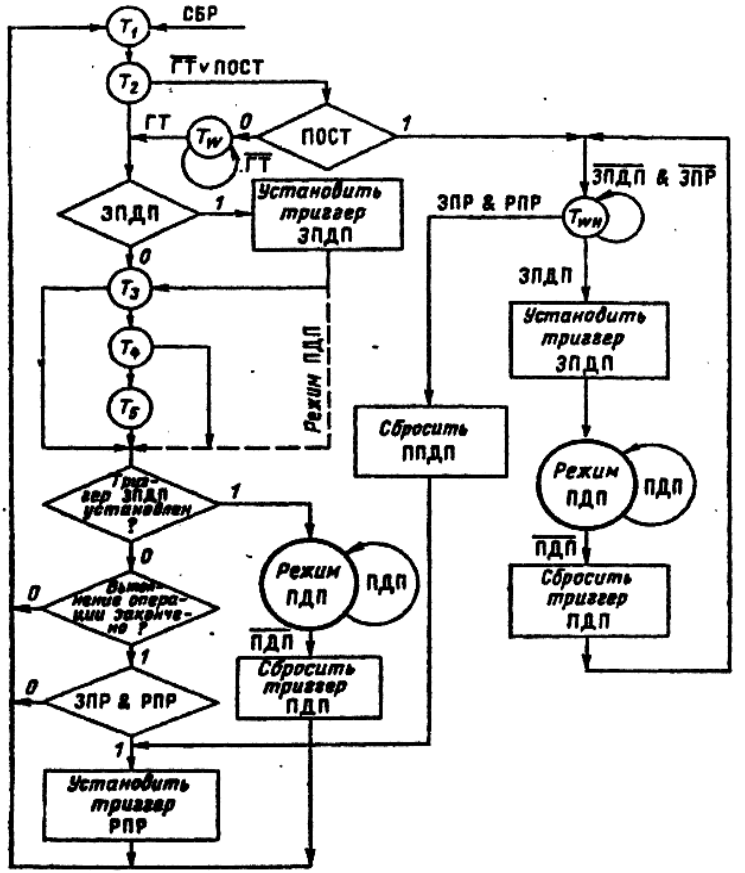
(2) Высокоимпедансное состояние

(3) Сигналы о состоянии микропроцесора

(4) Данные

Продолжение приложения А

Временные диаграммы и алгоритм работы ПМ КР580



ПРИЛОЖЕНИЕ Б

(обязательное)

Программное обеспечение. Ассемблер КР580 *I* Указания по изучению языков программирования. Общие понятия об Ассемблере

Все языки программирования можно подразделить на три группы:проблемно-ориентированные, машинно-
ориентированные и машинные.

Машинные языки - это языки цифр. Они занимают самый нижний уровень в иерархии языков программирования и весьма неудобны для программиста при написании и отладке программ.

К проблемно-ориентированным языкам, или языкам высокого уровня, относятся БЭЙСИК, ФОРТРАН, ПЛ/М, ПАСКАЛЬ, АДА и др. Они обладают относительной машинной независимостью, и программа для них пишется, исходя из существа задачи. Проблемно-ориентированны ми языками для микроЭВМ являются БЭЙСИК и ПЛ/М.

Машинно-ориентированный язык мнкроЭВМ называется языком Ассемблера. Поскольку микроЭВМ, производимые различными изготовителями, не похожи друг на друга, отличаются и языки Ассемблера, но незначительно. Свое название языки Ассемблера получили от имени программы* транслятора» преобразующей исходную программу (написанную на языке Ассемблера) в объектную (программу на машинном языке» понятную ЭВМ). Каждый язык Ассемблера является машинно-зависимым и отражает аппаратурные особенности (в частности,
состав программно* доступных регистров) той микроЭВМ, для которой он создан.

Ассемблер-транслятор, сам являющийся программой, загружается в ЭВМ вместе с исходной программой. При этом последняя выступает в роли данных, обрабатываемых Ассемблером-транслятором.

Для трансляции языков высокого уровня используются **программы-компиляторы.**

2 Общее понятие о программном обеспечении (ПО) микро-ЭВМ

Любая современная вычислительная система (ВС) состоит из двух главных взаимосвязанных и взаимодействующих компонентов - аппаратуры и ПО. ПО включает в себя ПО пользователя и системное ПО, называемое также ОС. ПО пользователя разрабатывается самим пользователем и содержит исходную, объектную и загрузочную программы, ОС поставляется вместе с машиной и выступает посредником между ЭВМ и ее пользователями. Она является основой ПО и предназначена для управления всеми аппаратурными и программными ресурсами *ВС,* облегчения создания и отладки новых программ, автоматизации их прохождения через вычислительную машину, управления файлами, повышения пропускной способности ВС и производительности труда обслуживающего персонала. Чаще всего ОС микроЭВМ состоит из управляющей программы-монитора, находящегося с момента загрузки и запуска ОС в оперативной памяти микроЭВМ, а также набора системных программ и данных.

Монитор предназначен для организации взаимодействия пользователей с микроЭВМ, запуска и контроля выполнения системных программ пользователей, слежения за выполнением операций ввода-вывода, управления распределением

оперативной памяти микроЭВМ и манипулирования файлами.

В группу системных программ и данных могут входить трансляторы языков программирования, редактор текстов, редактор связей программных модулей, отладчик программы, библиотека макрокоманд и т.д.

Важной задачей организации ОС микроЭВМ является выбор машинного носителя для постоянного хранения ее компонентов, а также программ и данных пользователя.

3 Этапы разработки программ

Разработка программы включает следующие основные этапы:

1) постановку задачи;
2) проектирование программы;
3) кодирование;
4) тестирование и отладка;
5) документирование.

На этапе постановки задачи определяются входные и выходные данные, временные ограничения, требования к процессу обработки данных, точность, ограничения по памяти, метод обработки ошибок.

На этапе проектирования программы создается программный алгоритм в соответствии с поставленной задачей. При этом используются такие методы, как проектирование сверху вниз, структурное программирование, модульное программирование, создание блок-схемы.

Нисходящее проектирование, или проектирование сверху вниз, - это метод, при котором разрабатываемая задача расчленяется сначала на несколько обобщенных подзадач, которые, в свою

очередь, детализируются далее. Этот процесс продолжается до тех пор, пока полу-
чаемые подзадачи не оказываются в виде, пригодном для простой реализации на ЭВМ

Противоположным данному методу является проектирование программы снизу вверх, при котором сначала кодируются все подзадачи, а затем они объединяются в более крупные части общего проекта.

Метод структурного программирования заключается в том, что при написании программ используются только определенные типы программных операторов, являющиеся простыми логическими структурами.

Модульное программирование представляет собой метод разработки программ, при котором большие программы пишутся, тестируются и отлаживаются небольшими частями -модулями, - которые затем объединяются в единый комплекс.

Выбранный для решения поставленной задачи метод может быть изображен в виде блок-схемы алгоритма программы, представляющей собой упорядоченную совокупность соединенных между собой блоков. Стрелки на схеме показывают направление развития вычислительного процесса

На этапе кодирования (собственно программирования) с помощью выбранного языка программирования пишется текст спроектированной

программы с последующей трансляцией его на машинный язык. При этом при написании текста программы для удобства ее отладки и последующего сопровождения рекомендуется придерживаться определенной последовательности. Заголовком программы должен быть текст комментариев, включающий в себя название программы, дату ее разработки и назначение, а также краткие сведения о требованиях к памяти и организации ввода-вывода.

Затем следуют псевдокоманды определения символических констант и имен, резервирования элементов памяти для переменных, стек, таблицы переходов, определяющие обращение к блокам программы и подпрограммам ввода-вывода. Далее в текст заносят основную программу. Подпрограммы размещаются вслед за основной программой в порядке обращения к ним. В заключение приводятся таблицы данных

Тестирование и отладка программ тесно взаимосвязаны. Тестирование - это, по-существу, следующий этап отладки. На этапе отладки проводится поиск и исправление программных ошибок, а на этапе тестирования проверяется корректность программы. В теории программирования отладка и тестирование программ часто называется соответственно верификацией и проверкой работоспособности.

На этапе документирования разрабатывается программная документация, которая дает возможность тем, кто должен использовать программу, разобраться в ее работе.

4 указания по написанию ассемблерных программ

Программа, написанная на языке Ассемблера, состоит из последовательности предложений (операторов) и называется исходной. Каждое предложение, как правило, содержит 4 поля, расположенные в следующем порядке: I) поле метки (имени); 2) поле кода; 3) поле операндов; 4) поле комментариев. Обязательным является лишь поле кода, а остальные могут отсутствовать.

Ассемблер допускает свободный формат, при котором различные поля могут отделяться друг от друга произвольным числом пробелов.

Поле метки позволяет задать адрес команды в символическом виде. Метка - последовательность букв и цифр,

начинающаяся с буквы и содержащая не более б символов. После метки ставится двоеточие (:). Метка позволяет при необходимости сослаться на команду и облегчает чтение программы. Использование в качестве меток команд и псевдо-команд Ассемблера недопустимо.

Поле кода содержит мнемонический код команды, указывающий, какую машинную операцию надо выполнить (сложение, вычитание, переход и т.п.).

Мнемонический код содержит 2-4 буквы, но иногда может быть длиннее. В поле кода могут быть псевдокоманды, которые предназначены для указаний Ассемблеру, и задания значения меток в Ассемблерной программе. В Ассемблере имеются следующие псевдокоманды:

ORC - установить начальное значение счетчика адреса;

DS - задать область памяти;

EQU - задать значение символического имени;

END - конец исходной записи;

EOF - конец исходного файла.

Поле кода необязательно отделять от поля метки пробелами.

В поле операндов содержится вся информация, которая нужна для полного определения выполняемой команды. Оно может включать в себя символический адрес памяти, обозначение общего регистра, непосредственно данные или процедуру вычисления. В зависимости от содержимого поля кода поле операндов может отсутствовать или содержать один или два операнда, разделенных запятой (,). Если операционный код требует наличия операндов, то поле кода отделяется от поля операндов, по крайней мере, одним пробелом.

Код метки, команды и операнда называется Ассемблерным.Поле комментариев дает возможность программисту снабдить предложение любыми пояснениями, облегчающими чтение и понимание программы. Обычно единственное требование к этому полю состоит в том, чтобы ему предшествовал ограничитель - точка с запятой (;),так как

это поле игнорируется в процессе трансляции исходной программы в объектную.

Команды программы после загрузки располагаются в памяти последовательно одна за другой. Память состоит из набора восьмиразрядных ячеек, каждая из которых может хранить 1 байт информации. В зависимости от выполняемой операции команды могут быть одно-, двух- или трехбайтными и занимать при этом соответственно одну, две или три ячейки памяти. Каждая ячейка памяти имеет свой шестнадцатиразрядный адрес.

При написании программ целесообразно перед каждой командой указывать ее адрес, представленный для удобства в шестнадцатеричной системе счисления.

Представлению чисел в той или иной системе счисления соответствуют следующие буквенные обозначения, приводимые непосредственно после числа:

D - десятичное число;

B - двоичное число;

H - шестнадцатеричное число.

5 Примеры ассемблерных программ МП КР580

5. I Пересылка информации

В любой ЭВМ при решении различных задач часто возникает необходимость пересылки информации из одной области памяти в другую, из памяти – в регистр или между регистрами.

Рассмотрим простейшие примеры программ пересылки данных.

Пример 1 Число длиной 3 байта, находящееся в регистрах *C, D*, *E*, переслать в регистры *B*, *H*, *L* .

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MOV B, C*	; Переслать содержимое *C* в *B*
01	*MOV H,D*	; Переслать содержимое *D* в *H*
02	*MOV L, E*	; Переслать содержимое *E* в *L*
03	*END*	; Конец.

Как видно из примера, после команды *MOV* (пересылка) указывается регистр, в который будет производиться пересылка, а затем через запятую - регистр, из которого пересылаются данные.

Пример 2 Переслать данные фиксированного формата, занимающие 2 байта, из памяти в регистровую пару *DE*.

Программа:

```
Адрес   Ассемблерный код          Комментарий
00      LXI  H, ADR        ; Установить непосредственный
01                         ; адрес, по которому хранится
02                         ; 2-й байт данных, в
                           ; регистровую пару HL
03      MOV E ,M           ; Переслать в E содержимое
                           ; ячейки памяти, адресуе-
                             мой парой HL
04      INX   H            ; Увеличить на 1 адрес,
                             находящийся в HL
05      M0V   D ,M         ; Переслать в D содержимое
                             ячейки памяти, адрес ко-
                             торой находится в HL
06      END                ; Конец.
```

Пример 3 Переслать данные фиксированного формата, занимающие 2 байта, из регистровой пары *DE* в память.

Программа:

```
Адрес   Ассемблерный код          Комментарий
00      LKI  H, ADR        ; Установить непосредст-
                             венный
01                         ;адрес в регистровую
02                         ; пару HL
03      M0V  M, E          ; Переслать содержимое E
                             в ячейку памяти с адресом,
                           находящимся в HL
04      INX  H             ; Увеличить на 1 адрес в HL
05      MOV  M, D          ; Переслать содержимое D в
                             ячейку памяти, адресуе-
                             мую парой HL
06      END                ; Конец.
```

Пример 4 Переслать данные фиксированного формата длиной 2 байта из памяти в пару *HL*.

Программа:

Адрес	Ассемблерный код	Комментарий
00	*LXI H, ADR*	; Установить непосредственный
01		; адрес в регистровую
02		; пару *DE*
03	*LDAX D*	; Загрузить *A* содержимым ячейки памяти с адресом, находящимся в *DE*
04	*MOV L, A*	; Переслать содержимое *A* в *L*
05	*INX D*	; Увеличить на 1 адрес, содержащийся в *DE*
06	*LDAX D*	; Загрузить *A* содержимым ячейки памяти, адресуемой парой *DE*
07	*MOV H, A*	; Переслать содержимое *A* в *H*
08	*E ND*	; Конец.

Пример 5 Переслать данные фиксированного формата длиной 2 бай та из пары *HL* в память.

Программа:

Адрес	Ассемблерный код	Комментарий
00	*LXI D, ADR*	; Установить непосредствен-
01		; ный адрес в регистровую
02		; пару *DE*
03	*MOV A, L*	; Переслать содержимое регистра *L* в *A*
04	*STAX D*	; Записать в ячейку памяти, адресуемую парой *DE*, содержимое *A*
05	*INX D*	; Увеличить на 1 адрес в *DE*
06	*MOV A, H*	; Переслать содержимое регистра *H* в *A*
07	*STAX D*	; Записать в ячейку памяти, адресуемую парой *DE*, содержимое *A*

```
08          END                 ; Конец.
```

Пример 6 Переслать число в плавающем формате, которое занимает 3 байта, из памяти в регистры *C* , *D* , *E* .

Программа:

```
Адрес   Ассемблерный код        Комментарий
00      LXI  H, ADR             ;Загрузить в регистровую
                                  пару
01                              ; HL непосредственный адрес,
02                              ; по которому хранится 1-й
                                  байт информации
03      MOV  C, M               ; Переслать в C содержимое
                                ; ячейки памяти, адресуемой
                                    регистрами H, L
04      INX  H                  ; Увеличить на 1 содержимое
                                    пары HL
05      MOV  E, M               ; Переслать в D содержимое
                                  ячейки памяти, адресуемой
                                    HL
06      INX  H                  ;Увеличить на 1 содержимое
                                  пары HL
07      MOV  E, M               ; Переслать в E содержимое
                                  ячейки памяти, адресуемую
                                    HL
08      END                     ; Конец.
```

Пример 7 Переслать число в плавающем формате, которое занимает 3 байта, из регистров C , *D* , *E* в память.

Программа:

```
Адрес   Ассемблерный код        Комментарий
00      LXI  H, ADR             ; Установить
01                              ; в регистровой паре HL
02                              ; непосредственный адрес
03      MOV  M, C               ;Переслать в ячейки памяти,
                                    адресуемую парой H L,
                                    содержимое регистра C
04      INX  H                  ; Увеличить на 1 адрес,
                                    содержащийся в HL
```

```
05      MOV   M, D          ;Переслать в ячейки памяти,
                              адресуемую парой H L,
                              содержимое регистра D
06      INX  H              ; Увеличить на 1 адрес,
                              содержащийся в HL
07      MOV   M, E          ;Переслать в ячейки памяти,
                              адресуемую парой H L,
                              содержимое регистра E
08      END                 ; Конец.
```

Пример 8 Переслать число в плавающем формате, которое занимает 3 байта, из памяти в регистры *B* , *H* , *L* .

Программа:

```
Адрес   Ассемблерный код        Комментарий
00      LXI  D,  ADR        ;Загрузить непосредственный
01                          ;  адрес в регистровую
02                          ;  пару DE
03      LDAX  D             ; Загрузить  A  содержимым
                              ячейки памяти с адресом,
                              находящимся в паре DE
04      MOV  B,  A          ;Переслать содержимое  A в B
05      INX   D             ; Увеличить на 1 адрес,
                              содержащийся в DE
06      LDAX D              ; Загрузить  A  содержимым
                              ячейки памяти, адресуемой
                              парой DE
07      MOV   H, A          ;Переслать содержимое A в H
08      INX   D             ; Увеличить на 1 адрес,
                              содержащийся в DE
09      LDAX D              ; Загрузить  A  содержимым
                              ячейки памяти, адрес
                              которой находится в DE
0A      MOV   L, A          ; Переслать содержимое A в L
0B      END                 ; Конец.
```

Пример 9 Записать содержимое плавающего формата регистров *B* , *H* , *L* в память.

Программа:

Адрес	Ассемблерный код	Комментарий
00	*LXI D, ADR*	; Загрузить непосредственный
01		; адрес в регистровую
02		; пару *DE*
03	*MOV A, B*	; Переслать содержимое регистра *B* в *A*
04	*STAX D*	; Записать в ячейку памяти адресуемую парой *DE*, содержимое *A*
05	*INX D*	; Увеличить на 1 адрес, содержащийся в *DE*
06	*MOV A, H*	; Переслать содержимое регистра *H* в *A*
07	*STAX D*	; Записать в ячейку памяти адресуемую парой *DE*, содержимое *A*
08	*INX D*	; Увеличить на 1 адрес, содержащийся в *DE*
09	*MOV A, L*	; Переслать содержимое регистра *L* в *A*
0A	*STAX D*	; Записать в ячейку памяти адресуемую парой *DE*, содержимое *A*
0B	*END*	; Конец.

5.2 Арифметика одинарной точности

Одна из основных функций любой ЭВМ - выполнение арифметических операций, так как без них невозможно решить почти ни одной задачи. Максимальное значение целого числа определяется числом бит, отведенных для его хранения. Если целое число хранится в одной, двух, трех и более ячейках памяти, то считается, что оно имеет соответственно одинарную, двойную, тройную и большую точность.

Пример 10. Сложить два числа 10100011 и 00011110.

Далее приводится два варианта решения задачи.

1-й вариант составления программы:

Адрес	Ассемблерный код	Комментарий

00	*MVI A 10100011 B*	;Пересылка непосредственных
01		данных в *A*
02	*ADI A 00011110 B*	; Сложение непосредственных
03		данных с содержимым *A*
04	*END*	; Конец.

2-й вариант составления программы:

Адрес	Ассемблерный код	Комментарий
00	*MVI B 10100011 B*	;Переслать непосредственные
01		данные в регистр *B*
02	*MOV A, B*	; Переслать данные из *B* в *A*
03	*MVI B 10100011 B*	;Переслать непосредственные
04		данные в регистр *B*
05	*ADD B*	;Сложить содержимое A и *B*
06	*END*	;Конец.

Первый вариант по сравнение со вторым более предпочтителен, так как программа в первом случае занимает меньший объем памяти.

Пример 11 Сложить два числа 10100011 и 00011110 и результат записать в ячейку памяти 1101.

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI B 10100011 B*	;Переслать непосредственные
01		данные в регистр B
02	*MOV A, B*	;Переслать содержимое *B* в *A*
03	*ADI B 10100011 B*	; Сложить непосредственные
04		данные с содержимым *A*
05	*IXI B 00000000*	;Загрузить пару регистров *BC*
06	*00001101B*	;непосредственными данными
07		
08	*STAX B*	; Записать содержимое *A* в ячейку памяти, адресуемую содержимым регистром *BC*
09	*END*	; Конец.

Пример 12 Сложить два числа 10100011 и 00011110 и результат записать по адресу, находящемуся в двойном регистре *HL* .

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI 10100011 B*	;Пересылка непосредственных
01		данных в *A*
02	*ADI A 00011110 B*	;Сложение непосредственных
03		данных с содержимым *A*
04	*HCHG*	; Обмен содержимым регистров *H* и *D* , *E* и *L*.
05	*STAX D*	; Запись содержимого *A* в ячейку памяти, адресуемую содержимым регистров D и E
06	*END*	; Конец.

Пример 13 Сложить содержимое ячеек 40*H* и 41*H*. Резуль-тат поместить в ячейку 42*H*.

Программа

Адрес	Ассемблерный код	Комментарий
00	*LDA 40H*	; Загрузка *A* содержимым
01		; ячейки *40H*
02		;
03	*MOV B, A*	; Пересылка в регистр *B* содержимого *A*
04	*LDA 41H*	; Загрузка *A* содержимым
05		; ячейки *41H*
06		
07	*ADD B*	; Сложение содержимого *B* с содержимым *A*
08	*STA 42H*	; Пересылка содержимого *A* в
09		; ячейку *42 H*
0A		;
0B	*END*	; Конец.

Пример 14 Сложить число *IEH* с числами, находящимися в ячейках памяти с адресами 2000 *H* и 2005 *H* .

Программа:

Адрес	Ассемблерный код	Комментарий
00	*LXI H, 2000H*	; Установить адрес 2000*H*
01		; в *HL*
02		;
03	*MOV A, M*	;Переслать содержимое ячейки памяти, адрес которой находится в *HL* , в *A*
04	*ADI IEH*	;Сложить число, содержащееся
05		; в *A*, с числом *IEH*
06	*LXI H,2005H*	; Загрузить адрес 2005 *H*
07		; в *HL*
08		;
09	*MOV B, M*	; Переслать число из ячейки памяти с адресом, находящимся в *HL* , в *B* .
0A	*ADD B*	; Сложить содержимое *A* с числом, находящимся в *B*
0B	*LXI H REZ*	; Установить в *HL* адрес,
0C		; определяемый меткой *REZ.*
0D		;
0E	*MOV M, A*	; Переслать результат из *A* в ячейку памяти с адресом в *HL*
0F	*REZ: DS OIH*	; Зарезервировать 1 байт памяти для хранения результата
10	*END*	; Конец.

Пример 15 Из числа *A3H* вычесть число *1EH* . Результат занести в ячейку памяти с адресом *1030H.*

Программа:

Адрес	Ассемблерный код	Комментарий
00	*MVI A, 043H*	; Загрузить в *A* число *OA3H*
01		

```
02          SUI   I E H        ; Вычесть из числа OA3H
03                             ; число IEH
04      LXI   H,  1030H        ;Установить адрес I030H
05                               в HL
06
07          MOV   M, A         ; Переслать результат из A
                               в ячейку памяти, адресуемую
                                парой HL
08      END                    ; Конец.
```

5.3 Арифметика многократной точности

Пример 16 Из числа, хранящегося в ячейках памяти 0120 - 0121, вычесть сумму двух чисел: младшие и старшие биты первого слагаемого находятся соответственно в ячейках 0110 и 0111, второе слагаемое - в ячейках 0112 и 0113. К разности прибавить число 6 и результат занести в ячейки памяти 0070 и 0071. Ячейкам памяти, в которых располагаются младшие биты чисел, соответствуют следующие метки: 0110 - *NOS* ; 0120 - *TOTL* ; 0070 - *ANS* .

Программа:

```
Адрес   Ассемблерный код        Комментарий
 00          LXI   H,   NOS     ; Загрузить адрес NOS в
 01                             ; регистровую пару HL
 02                             ;
 03          MOV     C, M       ; Содержимое по адресу HL
                                  переслать в C
 04          INX   H            ;Увеличить содержимое HL на 1
 05       MOV  B,  M            ; Содержимое по адресу HL
                                  переслать в B
 06          INX     H          ;Увеличить содержимое HL  на1
 07          MOV     A, M       ; Содержимое по адресу HL
                                    переслать в A
 08          ADD     C          ; Сложить содержимое  A  и  C
 09          MOV     C, A       ; Переслать в   C   сумму
                                    младших байт
```

```
1A      1HX     H           ;Увеличить содержимое HL на 1
1B      MOV     A, M        ; Содержимое по адресу ML
                              переслать в A
1C      ADC     B           ; Сложить содержимое A и B
                              с учетом переноса из
                              младшего разряда
1D      MOV  B, A           ; Переслать в B сумму
                              старших байт
1E      LXI  H  TOTL        ; Загрузить адрес TOTL в
1F                          ; регистровую пару HL
10                          ;
11      MOV  A,  M          ; Содержимое по адресу HL
                              переслать в A
12      SUB     C           ; Вычесть из содержимого A
                              содержимое C
13      MOV     C, A        ; Переслать в C младший
                              байт разности
14      INX     H           ;Увеличить содержимое HL на 1
15      MOV     A, M        ; Содержимое по адресу HL
                              переслать в A
16      SBB     B           ; Вычесть из A содержимое B
                              с учетом переноса
17      MOV     B, A        ; Переслать в B старший байт
                              разности
18      MOV     A, C        ; Переслать в A младший байт
                              разности
19      ADI     6           ; Прибавить к содержимому A
2A                          ; число 6
2B      STA     ANS         ;Запомнить младший байт
2C                          ;разности по адресу AMS
2D
2E      MVI  A, O           ; Сбросить A
2F
30      ADS     B           ; Передать содержимое B в A
```

```
                                  с учетом переноса
31    STA     ANS+1               ; Запомнить старший байт
32                                ; разности по адресу ANS+1
33                                ; Конец.
```

5.4 Логические операции

В системах команд всех ЭВМ есть команды логических операций. Они выполняют поразрядные операции инвертирования, И, ИЛИ, исключительно ИЛИ (сложения по *mod 2*), а также циклический сдвиг аккумулятора. При этом инвертирование заменяет каждый бит его дополнением; операция И дает в результате бит, равный единице, если соответствующие биты обоих операторов содержат единицу; операция ИЛИ - бит, равный единице, если, соответствующие биты в одном или обоих операнда содержат единицы; операция исключающего ИЛИ - бит, равный единице, если только соответствующий бит одного операнда содержит единицу.

Большинство применений логических операций связано с изменением, обнулением /сбросом/ и проверкой бит в байте. Эти действия выполняются с помощью логических операций над нужным байтом и вторым байтом - маской. Такая операция называется маскированием.

Пример 17 Выделить младшие 4 бита (шестнадцатеричную цифру) из содержимого ячейки *40 H* и переслать их по адресу 41 *H*.

Программа:

```
Адрес  Ассемблерный код      Комментарий
 00       LXI   H, 40H       ; Загрузить непосредственный
 01                          ; адрес в регистровую пару HL
 02                          ;
 03       MOV   A, M         ;Переслать данные,
                             хранящиеся по адресу HL , в A
 04     ANI   00001111B      ; Маскировка старших 4 бит
 05
```

```
06      INX  H          ;Увеличить на 1 содержимое HL
07      MOV  M, A       ; Переслать содержимое A по
                                адресу HL
08      END             ; Конец.
```

При написании этой программы, если данные хранятся в смежных ячейках памяти, целесообразно использовать команды *MOV A, M* и *MOV M, A,* а не *LDA* и *STA,* так как в первом случае уменьшаются затраты времени и памяти. Следует также обратить внимание на то, что только команда *ANI* изменяет состояние признаков; команда *INX* на их состояние влияния не оказывает.

Пример 18 Из содержимого ячейки *40H* выделить младшие 4 бита (младшую шестнадцатеричную цифру) и переслать их по адресу *41H*. Старшие 4 бита (старшую шестнадцатеричную цифру) содержимого ячейки *40H* переслать в младшие разряды ячейки *42H*.

Программа:

```
Адрес   Ассемблерный код      Комментарий
 00       LXI  H, 40H        ; Загрузить непосредственный
 01                          ; адрес в регистровую пару
 02                          ; HL
 03       MOV  A, M          ; Загрузить в A данные по
                               адресу HL
 04       MOV  B,  A         ; Запомнить данные в регистре
                              B
 05       ANI  00001111B     ; Маскировка старшей цифры
 06                          ;
 07       INX  H             ; Увеличить на 1 содержимое
                                  HL      .  .
 08       MOV  M,  A         ; Запомнить младшую цифру
 09       MOV  A,  B         ; Переслать исходное число
                                 из  B  в  A
 0A       RRS                ; Четырехкратный
 0B       RRS                ; циклический
```

```
0C          RRS              ; сдвиг
0D          RRS              ; вправо
0E      ANI  00001111B       ; Маскировка младшей цифры
0F                           ;
10        INX   H            ; Увеличить на 1 содержимое
                                 HL
11        MOV  M, A          ; Запомнить старшую цифру
12          END              ; Конец.
```

Пример 19 В ячейке 0040 находится слово, биты 2 и 3 которого необходимо селективно установить, бит 6 сбросить, а бит 5 - изменить, а затем проверить. В ячейках 0030-0032 хранятся маски, соответствующие кодам *OC, BF* и 20.

Ячейки 0030, 0031, 0032, 0040 соответственно имеют следующие метки*: MASKS, MASKS+1, MASKS+2* и *CTRL* .

Программа:

```
Адрес   Ассемблерный код        Комментарий
 00       LXI   H, MASKS       ; Загрузить непосредственный
 01                             ;адрес маски в регистровую
 02                             ; пару HL
 03        LDA    (CTRL)        ; Загрузить в A   содержимое
 04                             ;  ячейки CTRL
 05                             ;
 06         ORA   M             ; Установить бит 2 и 3
 07         INX   H             ; Инкремент до MASKS+1
 08         ANA   M             ; Сбросить бит 6
 09         INX   H             ; Инкремент до MASKS+2
 0A         XRA  M              ; Изменить бит 5
 0B        STA    CTRL          ;  Запомнить в CTRL
 0C                             ;
 0D                             ;
 0E         ANA   M             ;  Проверить бит 5
 0F           *                 ; Конец.
```

5.5 Организация циклов и обработка массивов

Реальные задачи, решаемые на ЭВМ, не сводятся к обработке отдельного элемента данных с помощью одной операции. Напротив, они требуют обработки многих элементов данных (например, массива или блока данных) либо отдельных элементов, которые занимают несколько ячеек памяти. Программа может выполнять одну и ту же операцию над содержимым нескольких ячеек или элементов данных. Многократное выполнение, группы команд осуществляется с помощью программных циклов. Числом повторений цикла управляют счетчики, а указатели показывают, какой именно элемент данных обрабатывается при данном проходе цикла. На рис.5.1 показана обобщенная блок-схема циклической программы.

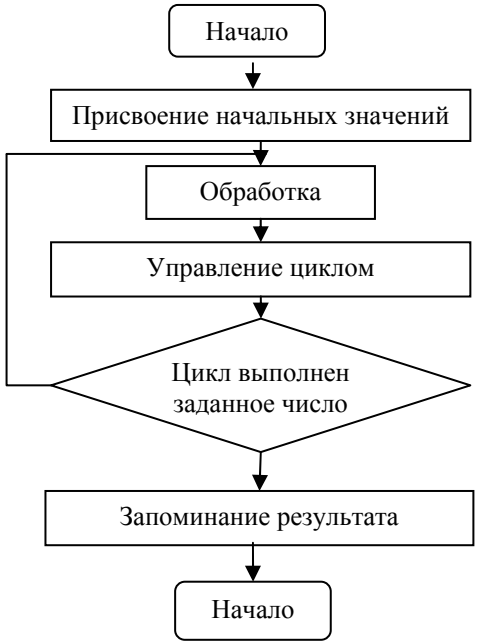


Рисунок 5.1- Обобщенная блок-схема циклической программы

Как видно из рис.5.1, циклическая программа содержит следующие четыре основных блока:

1) присвоения начальных значений переменным, счетчикам и указателям данных. Указатели представляют собой адреса данных;

2) обработки, который фактически выполняет требуемые вычисления;

3) управления циклом, в котором изменяются значения счетчиков и указателей перед выполнением следующей операции, а также проверяются числа выполнения цикла;

4) заключительный, запоминающий результат.

Первый и четвертый блоки выполняются только один раз, поэтому большая часть машинного времени затрачивается на выполнение второго и третьего блоков.

О изация циклов осуществляется с помощью команд условных переходов.

Рассмо использованием циклов.

5.5.1 Пересылка блока информации

Приме в памяти, начиная с начинающуюся с ячейки *51H*. в ячейке *40H*. Конкретно

Таким двух элементов, расположен льтате пересылки получим

Програ

Адрес тарий

00 пару *HL*

01 адрес

03 *MOV B, H* ; Счетчик = длина массива

04

05 посредственный адрес

06 ;

07 *TRANS: INX D* ; Увеличить содержимое

```
                                  DE на 1
08          INX  H                ; Увеличить содержимое
                                  HL  на 1
09          MOV  A, M             ; Загрузить данные из
                                    исходного массива
0A          STAX  D               ; Послать данные в
                                    результирующий массив
0B          DSR  B                ;Уменьшить на 1 содер-
                                    жимое счетчика
0C          JNS   TRANS           ; Переход при ненулевом
0D                                ; состоянии счетчика
0E
0F          END                   ; Конец.
```

В рассмотренном примере в качестве счетчика, указывающего, сколько раз (задается длиной массива) должен выполняться цикл, используется регистр B. Но иногда при выполнении операций цикла все регистры заняты, поэтому счетчик цикла хранят в памяти. Как и другие данные, используемые или получаемые в ходе выполнения программы, счетчик цикла удобно располагать в памяти сразу после программы. С этой целью выделяется область памяти с помощью псевдокоманды *DS* . Установку счетчика, хранимого в памяти, выполняет в виде подпрограммы для облегчения отладки всей программы в целом.

5.5.2 Арифметика многократной точности

Выполнение большинства задач арифметики многократной точности удобно организовывать с применением циклов.

Пример 21 Сложить два 24-битных числа *50A429* и *2837FB* . Результат поместить в ячейки, где хранится первое число. Длина чисел (в байтах) задана в ячейке *40H*. Первое число располагается, начиная с ячейки *41H*, второе - с ячейки *51H*, причем сначала идут младшие разряды чисел, т.е.

(40) = 03;
(41) = 29;
(42) = A4;

(43) = 50;
(51) = FB;
(52) = 37;
(53) = 28.

Результат будет получен при суммировании в такой последовательности:

(41) = 29 + FB;
(42) = A4 + 37 + ПЕРЕНОС;
(43) = 50 + 29 + ПЕРЕНОС.

Составляем программу:

Адрес	Ассемблерный код	Комментарий
00	*SUB A*	; Обнулить *A*
01	*LXI H, 40H*	; Загрузить пару регистров
02		; *HL* непосредственными
03		; данными
04	*MOV B, M*	; Переслать в *B* содержимое ячейки памяти, адресуемую регистрами *HL*
05	*LXI D, 50H*	; Загрузить пару *DE*
06		; непосредственными
07		; данными
08	*MRADD: INX D*	;Увеличить содержимое *HL* на1
09	*INX H*	; Увеличить содержимое пары *HL* на 1
0A	*LDAX D*	; Загрузить A содержимым ячейки памяти, адресуемой регистровой парой *DE*
0B	*ADC H*	; Сложить с содержимым *A* содержимое ячейки памяти, адресуемой *HL*, и содержимое триггера переноса
0C	*MOV H, A*	; Переслать содержимое *A* в ячейку, адресуемую *HL*
0D	*DCR B*	;Уменьшить содержимое *B* на 1
0E	JNS *MPADD*	; Переход при ненулевом

результате

```
0F
10                          ;
11          END             ; Конец.
```

Пример 22Просуммировать ряд чисел, хранящихся начиная с ячейки *42H.* Результат поместить в ячейку памяти с адресом *40H* . Предполагается, что сумма не превышает 256. Длина суммируемого ряда хранится в ячейке *41H.* Таким образом,

(41) = 03;
(42) = 35;
(43) = 72;
(44) = ID.

Ряд содержит три числа, так как в ячейке 41 находится число 3. Сумма (42) + (43) + (44) = 35 +72 + ID = C4.

В программе суммирования ряда чисел в качестве счетчика используется регистр *B*, а в качестве адресного указателя данных - регистровая пара *HL.*

Программа:

```
Адрес   Ассемблерный код       Комментарий
00       SUB   A               ; Обнулить A
01      LXI  H, 41H            ; Загрузить непосредственный
02                             ; адрес в регистровую пару HL
03
04      MOV   B, H             ; Загрузить в счетчик число
                                   повторений цикла
05    SUMB:  INX   H           ;Увеличить на1содержимое HL
06        ADD  M               ; Сложить содержимое A и
                                   содержимое, адресуемое
                                   регистровой парой HL.
07          DCR  B             ; Уменьшить на I содержимое
                                   счетчика
08       JNZ   SVHB            ; Переход при
09                             ; ненулевом состоянии
0A                                 счетчика
0B          STA   40H          ; Запомнить сумму
0C                             ; в ячейке
```

0D ; с адресом *40H*
0E *END* ; Конец.

5.5.3 Десятичная арифметика

Работа всей вычислительной техники основывается на использовании двоичной системы счисления. Но для человека эта система неудобна, поэтому иногда желательно, чтобы вычислительная машина работала с десятичными числами. Для сочетания простоты двоичной системы с удобствами десятичной используют несколько существующих способов кодирования десятичных цифр двоичными. Наиболее часто применяется двоично-десятичный код 8421 *BCD (Binari Coded Decimal),* в котором десятичные цифры кодируются первыми десятью четырехразрядными двоичными числами, т.е. восьмибитное двоичное число рассматривается как две десятичные четырехбитные двоично-кодированные цифры. Для работы таким форматом предусмотрена команда *DAA*, которая корректирует результат операции по следующим правилам:

-если значение младших 4 бит больше 9 или признак вспомогательного переноса *AC* равен 1, то к младшему разряда добавляется 6;

-если значение старших 4 бит больше 9 или признак переноса *CY* равен 1, то 6 прибавляется в старший разряд.

Например:

I Десятичный код Двоично-десятичный код:

$_{+}24$ $_{+}00100100$

37 00110111

61 $_{+}01011011$

110 прибавление 6,

поскольку

011≥1001($11_{10}>9_{10}$)

01100001 – десятичное 61.

2 Десятичный код Двоично-десятичный код:

$_{+}58$ $_{+}01011000$

69 01101001

127 $_{+}11000001$

110 прибавление 6,

поскольку возник
перенос
+11000111
110 прибавление 6,
поскольку
1100>1001 (12_{10}>9_{10})
100100111– десятичное 127.
перенос

Пример 23 Сложить два десятичных числа 376529 и 224388. Результат поместить в ячейки, где хранится первое число. Длина чисел (в байтах) задана в ячейке *40H*. Первое располагается, начиная с ячейки *4IH* , второе - с ячейки *5IH*, причем вначале идут младшие разряды, т.е.:

(40) = 03;
(41) = 29;
(42) = 65;
(43) = 37;
(51) = 88;
(52) = 43;
(53) = 22.

Результатом сложения является число 600917:

(41) = 17;
(42) = 09;
(43) = 60.

Составляем программу:

Адрес	Ассемблерный код	Комментарий
00	*SUB A*	; Обнулить *A*
01	*LXI H, 40H*	; Загрузить пару
02		; регистров *HL*
03		; непосредственными данными
04	*MOV B, M*	; Установить счетчик
05	*LXI 50H*	; Установить
06		; указатели
07		; на
08	*DCADD: INX D*	; второе и
09	*INX H*	; первое число
0A	*LDAX*	; Взять две цифры второго

```
                                  числа
0B      ADC   M                   ; Прибавить две цифры
                                    первого числа
0C      DAA                       ; Десятичная корректировка
0D    MOV    M, A                 ; Послать сумму на место
                                    первого числа
```

Продолжение приложения Б

```
0E      DCR     B                 ; Уменьшить содержимое
                                            счетчика на 1
0F    JNZ    DCADD                ; Переход
10                                ;  при ненулевом
11                                ; состоянии счетчика
12.           END                 ;  Конец.
```

5.5.4 Реализация операции умножения

Умножение двоичных чисел на 2, 4, 8 и т.д. (т.е. на степень числа 2) соответствует сдвигу разрядов множимого соответственно на одну, две, три и т.д. позиций влево. Например:

1) $10100 \times 10 = 101000$ $(20_{10} \times 2_{10} = 40_{10})$
2) $10100 \times 100 = 100000$ $(20_{10} \times 4_{10} = 80_{10})$

Так как деление - это операция, противоположная умножению, то оно выполняется сдвигом делимого на соответствующее количество разрядов вправо.

При умножении двоичного числа, например, на 5 сначала необходимо сдвинуть число на два разряда влево, что эквивалентно умножению на 4, а затем исходное число сложить с результатом сдвига ($A \times 5 = A \times 4 + A$). Например:

$1101 \times 101 = 1101 \times 100 + 1101 = 1000001$ ($13_{IO} \times 5_{10} = 65_{10}$).

В ЭВМ операции умножения реализуются программно с помощью команд сложения и сдвига. Существуют различные способы умножения, которые организуются в виде цикла.

Один из способов с одинарной точностью заключается в сбросе регистровой пары, предназначенной для умножения, и в последовательном анализе бита множителя, начиная с

младшего. Если бит равен 1, множимое прибавляется к регистровой паре - произведению; в противном случае сложение не производится. Затем множимое сдвигается на один бит влево и проверяется следующий бит множителя. Этот процесс продолжается до конца проверки всех бит множителя (рис.5.2).

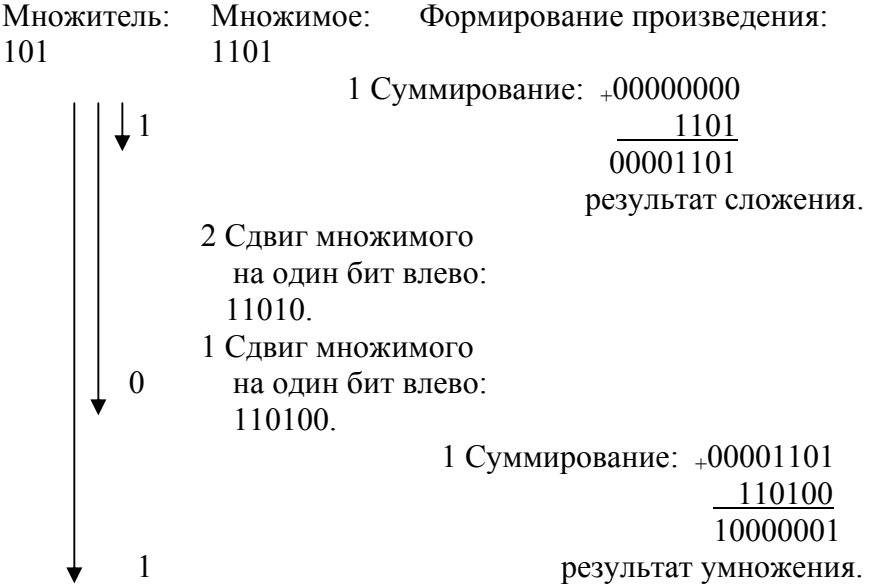


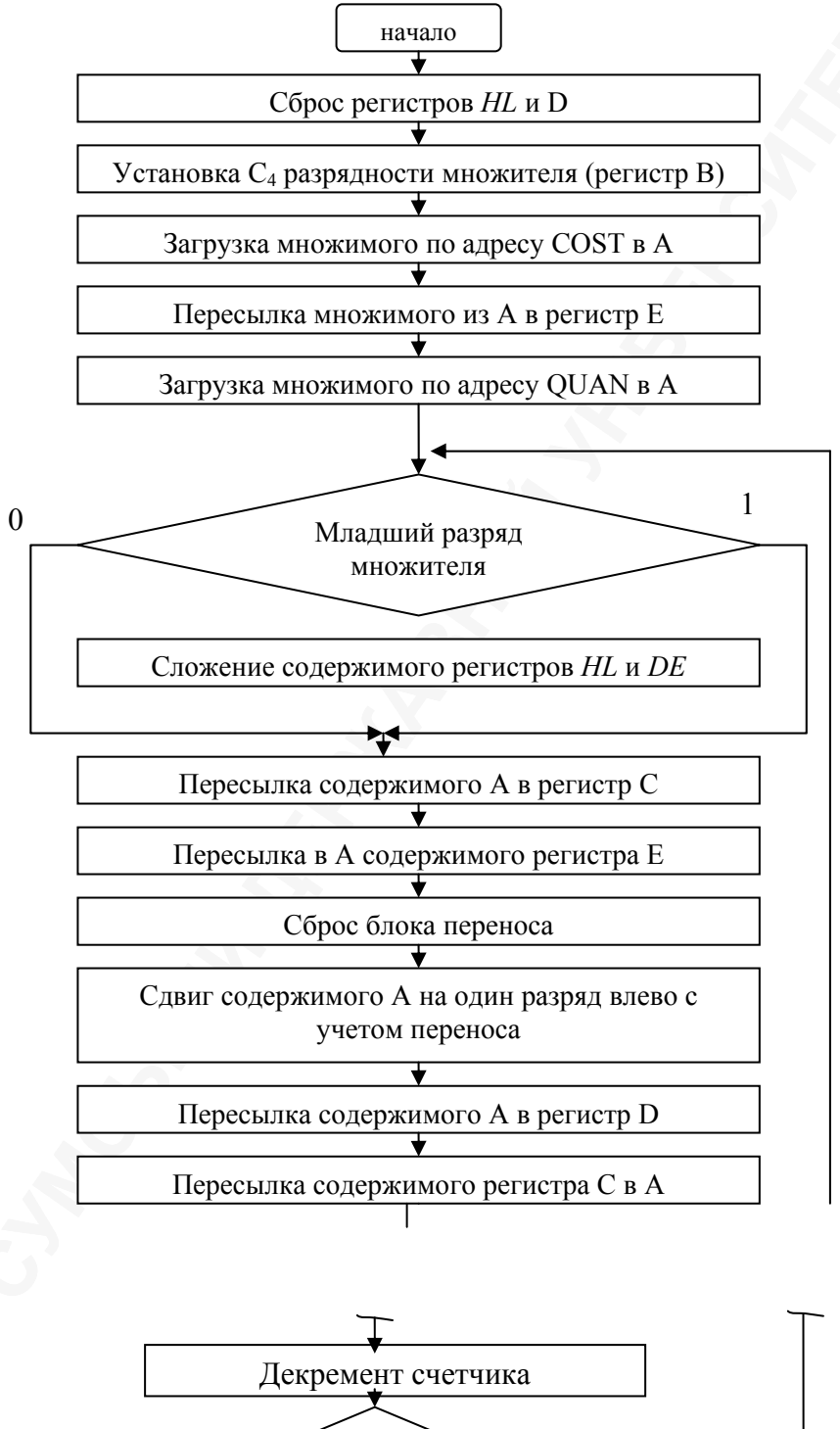
Рисунок 5.2-Способ умножения, организуемый в виде цикла

Рассмотрим пример программы реализации операции умножения с помощью данного способа.

Пример 24 Найти произведение содержимого ячеек памяти 0140 и 0150. С ячейкой 0140 ассоциирует *QUAN* , а с ячейкой 0150 - *COST* . Произведение образуется в регистровой паре *HL* .

Чтобы облегчить написание более сложных программ, последовательность выполняемых действий вначале представляется в виде блок-схемы алгоритма решения поставленной задачи.

Для данного примера составляем следующую блок-схему алгоритма (рис.5.3).


начало
Сброс регистров *HL* и D
Установка C_4 разрядности множителя (регистр B)
Загрузка множимого по адресу COST в A
Пересылка множимого из A в регистр E
Загрузка множимого по адресу QUAN в A
0
Младший разряд множителя
1
Сложение содержимого регистров *HL* и *DE*
Пересылка содержимого A в регистр C
Пересылка в A содержимого регистра E
Сброс блока переноса
Сдвиг содержимого A на один разряд влево с учетом переноса
Пересылка содержимого A в регистр D
Пересылка содержимого регистра C в A
Декремент счетчика

Нет

Да

Рисунок 5.3- Блок-схема алгоритма

В соответствии с блок-схемой составляем программу:

Программа:

```
Адрес    Ассемблерный код          Комментарий
00       LXI   H,  OH          ; Сброс пары HL
01                               ;
02                               ;
03       MVI   B,  08H         ; Пересылка числа бит,
04                               ; равного разрядности
                                   множителя, в регистр B
05       MVI   D  OH           ; Сброс регистра D
06                               ;
07       LDA     COST          ; Загрузка в A  множимого
08                               ;
09                               ;
0A       MOV    E,  A          ; Пересылка множимого из A
                                   в E
 0B      LDA    QUAN           ; Загрузка в A   множителя
0C                               ;
0D                              ;
0E    LOOP:   RRC              ; Переход, если
0F    JNC     NOADD            ; младший бит
10                               ; равен  0,   и сложение
11                              ; содержимого  E  с HL  ,
12          DAD                 ; если младший бит равен 1
13  NOADD:  MOV  C, A           ;Пересылка содержимого A в C
```

```
15      ORA   A          ; Сброс переноса
16      RAL              ; и сдвиг содержимого
17      M0V   E, A       ; пары DE
18      MOV   A, D       ; на один
19      RAL              ; разряд .
1A      MOV   D, A       ; влево
1B      MOV   A, C       ; Пересылка содержимого C
                           в A
1C      DCR   B          ; Декремент счетчика
1D    JNZ     LOOP       ; Переход по
1E                       ; ненулевому
1F                       ; результату
                         ;Конец.
```

Умножение можно организовать также путем анализа младшего разряда множителя и сдвига суммы частичных произведений и множителя вправо. Например, умножение чисел 011 и 101 производится следующим образом:

+000000 - обнуление суммы частичных произведений;
011 - множимое;
011000 - результат сложения;
001100 - первый сдвиг;
+000110 - второй сдвиг;
011 - множимое;
011110 - результат сложения
001111 - третий сдвиг, умножение закончено.

Схема алгоритма данного способа реализации операции умножения следующая (рис.5.4).

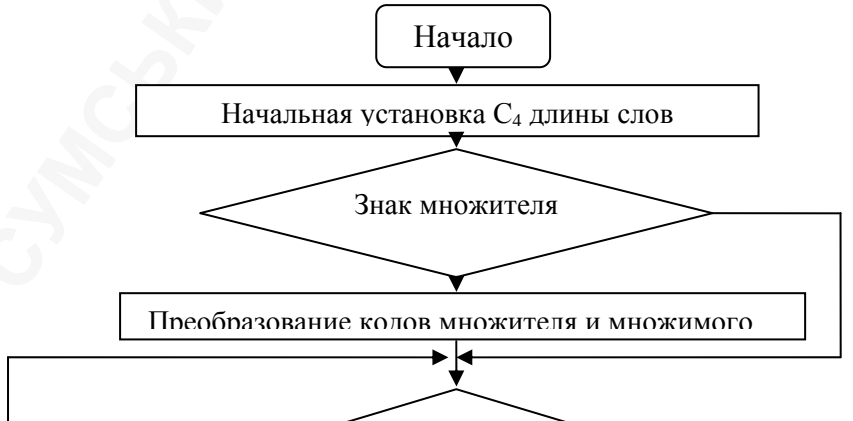
0 1

Нет

Да

Рисунок 5.4-Блок-схема алгоритма операции умножения

Числа со знаком умножается по следующему алгоритму (рис.5.5)



Нет

Да

Рисунок 5.5- Блок-схема алгоритма операции умножения числа со знаком

При умножении чисел по такому алгоритму не имеет значения, является ли множимое положительном или отрицательным числом, поскольку в первом случае формируются положительные суммы частичных произведений, во втором - отрицательные (после прибавления к дополнительному коду суммы частичных произведений отрицательного множимого, представленного также в дополнительном коде, будет получено правильное значение результата /следующей суммы частичных произведений/ в дополнительном коде. Однако множитель всегда должен быть в прямом коде.

Если множитель отрицательный (т.е. представлен в дополнительном коде), то для получения его прямого кода и правильного значения произведения следует преобразовать коды обоих сомножителей независимо от того, какой знак имеет

множимое. Это эквивалентно умножению обоих сомножителей на -1.

При выполнении сдвига суммы частичных произведений вправо (блок 6 алгоритма) освобождающиеся разряды должны заполняться при положительной сумме нулями, а при отрицательной - единицами, так как отрицательные числа представляются в дополнительном коде.

5.6 Подпрограммы

Во многих программах часто приходится сталкиваться с необходимостью в нескольких ее местах решать одну и ту же задачу. В этом случае обычно пишут подпрограмму (п/п), решающую эту задачу, и обращаются к ней по мере необходимости, не повторяя одни и те же команды в разных точках программы. При использовании п/п необходимо, нарушить обычную последовательность выполнения команд, перейти к п/п, а также обеспечить запоминание адреса возврата в основную программу. Для запоминания адреса возврата в основную программу и для других целей в области оперативного запоминающего устройства (ОЗУ) отводится так называемая стековая область, или стек. Стек - это совокупность регистров, которые принимают и выдают информацию по принципу "последним записан - первым считан" (*last input first Output-LIFO*), т.е. доступ к стеку возможен только с одного конца. Для приема и хранения адреса ячейки стека, к которой было последнее обращение, предназначен шестнадцатиразрядный указатель стека. Содержимое указателя стека уменьшается на единицу при поступлении данных в стек и соответственно увеличивается при выборке из него.

Командами, оперирующими с п/п, являются команды типа *CALL* (вызов) и *RET (RETURN* - возврат). Оба типа этих программ могут быть как безусловными, т.е. исполняемыми всегда, так и условными, т.е. исполняемыми при наличии определенных условий.

К безусловным командам относятся трехбайтная команда безусловного вызова п/п *CALL ADR* (символ *ADR* соответствует обозначению шестнадцатиразрядного адреса и

используется для большей наглядности), во втором и третьем байтах которой указывается адрес вызываемой п/п, и однобайтная команда безусловного возврата из п/п *RET*. Выполнение команды *CALL ADR* начинается с побайтовой засылки в стек адреса следующей после этой команды ячейки памяти. Такой адрес называется адресом возврата из п/п. Он необходим для того, чтобы по окончании выполнения п/п вернуться к продолжению выполнения основной программы. После записи в стек адреса возврата из п/п в счетчик команд PC микропроцессора загружается величина *ADR,* т.е. адрес первой команды, вызываемой п/п. Выполнение п/п всегда заканчивается командой возврата из п/п *RET* . При этом содержимое стека, т.е. адрес возврата из п/п, передается из стека в счетчик команд PC МП.

Условные команды, осуществляющие взаимодействие с п/п, включают в себя восемь команд условного вызова п/п и восемь команд условного возврата из п/п, действие которых определяется состоянием регистра признаков. Если условие для выполнения команды отсутствует, то вызов п/п или возврат из нее не выполняется.

Принцип вызова п/п и возврата в основную программу показан на рис.5.6.

В качестве подпрограмм могут использоваться рассмотренные ранее программы пересылки данных, умножения, сложения и т.д.

Рассмотрим простейший пример использования п/п пересылки двухбайтного числа из регистровой пары в память.

Пример 25 Записать шестнадцатеричное число *7ABH* в ячейку памяти с адресом *3050H*. Расположить программу в памяти с адресом *2000H*.

Программа:

Адрес	Объектный код	Ассемблерный код	Комментарий
		ORG 200QH	; Установить на ;начальное значение счетчика адреса равным *2000H*
2000	*11 AB 07*	*LXI D, 7ABH*	; Загрузить в пару

```
                                        DE число 7ABH
2003   21 50 30     LXI H, 3050H        ; Установить в пару
                                          HL адрес 3050H
2006   CD 09 20     CALL DERAM          ; Вызвать п/п, находя-
                                         ; щуюся по адресу,
                                         ;который соответствует
                                        имени DERAM
2009   73   DERAM:  MOV M, E            ; Переслать содержимое
                                          E в ячейку памяти с
                                          адресом, находящимся
                                          в HL
200A   23           INX H               ; Увеличить на 1 адрес,
                                          содержащийся в HL
200B   72           MOV  M, D           ; Переслать содержимое
                                          D в ячейку памяти,
                                          адресуемую парой HL
200C   23           INX  H              ; Увеличить на 1 адрес
                                          в HL
200D   C9            RET                 ; Возврат
200E                 END                 ;Конец исходной записи
```

В данном примере приводятся объектные коды команд, получаемые в результате трансляции Ассемблерного кода. Объектный код команды *LXI D, 7ABH* содержит три байта, из которых первый байт (11) - мнемонический код команды *LXI D,* второй байт *(AB)* представляет собой младший байт числа *7ABH,* а третий (07) - старший байт этого же числа. Из примера видно, что при трансляции команды *CALL DERAH* вместо символического имени фигурирует адрес 2009 (см. байты 2, 3 объектного кода), который соответствует имени *DERAM,* причем сначала идет младший байт адреса - 09, а затем старший - 20.

Псевдокоманды в объектные коды не транслируются, что наглядно иллюстрирует приведенный пример.

При использовании п/п в системах на базе МП К580 ИК80 только с ПЗУ (без ОЗУ), где внешний стек нельзя применять из-за невозможности записи в память, адрес возврата в основную программу запоминается в паре *HL* с помощью

команды *LXI* до перехода к подпрограмме обычной командой *JMP* . Для возврата используется команда *PCHL* . При таком способе стек не нужен.

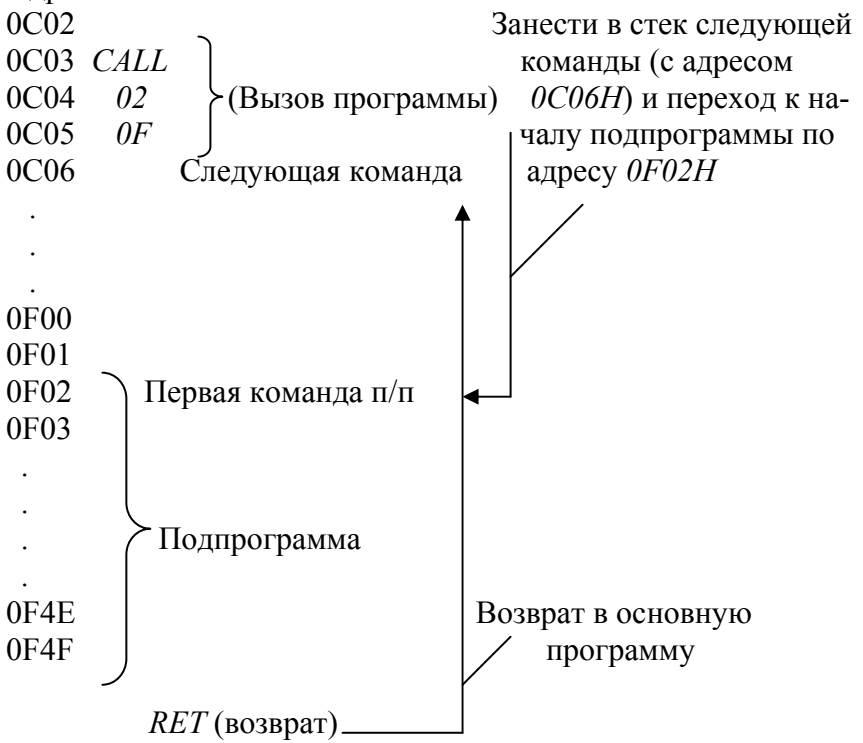


Рисунок 5.6- Принцип вызова подпрограмм и возврата в основную программу

5.7. Примеры программ, часто встречающихся на практике

Пример 26 Тумблер присоединен ко входу $в_3$ порта ввода 5. Если тумблер разомкнуть, то $в_3$=1, если замкнут, то $в_3$=0 (рис.5.7).

Требуется написать программу, которая проверит значение сигнала на входе $в_3$ порта ввода 5 и осуществит переход по программе к части *A* в том случае, если $в_3$=0, и к части *B*, если $в_3$=1.

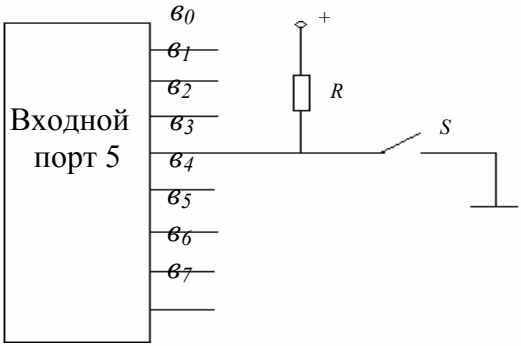


Рисунок 5.7- Схема входного порта и написание программы для него

Назовем данную программу *TESTSW* (тест ключа) и соста-вим ее блок-схему (рис.5.8).

Символическое имя *TESTSW* будем использовать в качестве метки, указывающей на адрес первой команды программы. При необходимости можно обращаться к этой программе в процессе выполнения другой.

Программа:

```
Адрес     Ассемблерный код          Комментарий
00    TESTSW:  IN   05 H    ; Ввод из порта 5 в A
01                          ;
02          ANI  08 H       ; Маскирование в1, в6, в5,
03                          ;  в4, в2, в1, в0
04           JZ  A          ;  Переход к сегменту A ,
05                          ;  если  в3 = 0, иначе
06                          ; переход к следующей команде
07                    B:          ; Первая команда сегмента
08                          ;   B  программы
•
•
7A                    A:          ; Первая команда сегмента
                                    A  программы
•
•
```

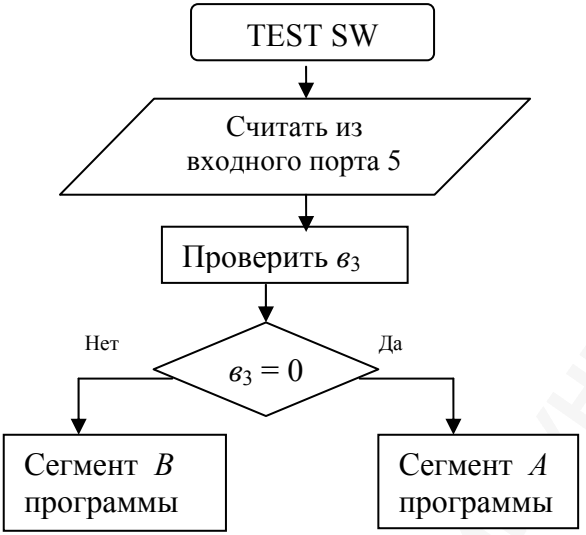


Рисунок 5.8- Блок-схема программы TEST SW

Пример 27 Дана следующая схема (рис.5.9).

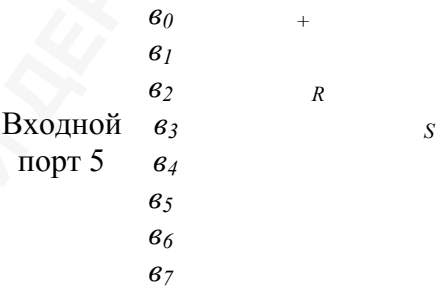


Рисунок 5.9- Схема для написания программы

Требуется написать подпрограмму, которая постоянно опрашивает состояние $в_5$ до тех пор, пока оно не станет равным нулю, т.е. пока тумблер не будет замкнут. Осуществить возврат в основную программу, если тумблер замкнут ($в_5 = 0$).

Назовем данную подпрограмму WAITSW (*Wait switsh* - ожидание включения) и используем это символическое имя в качестве метки, указывающей на первую команду этой подпрограммы. Составляем блок-схему решения поставленной задачи (рис.5.10).

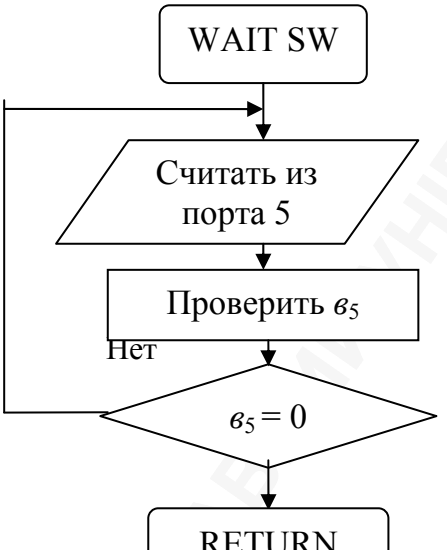


Рисунок 5.10- Блок-схема подпрограммы WAIN SW

Программа:

Адрес	Ассемблерный код	Комментарий
30	*WAITSW: IN 05 H*	;Ввод из порта 5 в *A*
31		;
32	*ANI 20H*	;Маскирование $в_1$, $в_6$, $в_4$,
33		; $в_3$, $в_2$, $в_1$, $в_0$
34	*JN2 WAITSW*	; Переход к циклу
35		; ожидания, если $в_5 = 1$
36		;
37	*RET*	; Возврат к основной
38		; программе, если $в_5 = 0$

Пример 28 Дана следующая схема (рис.5.11).

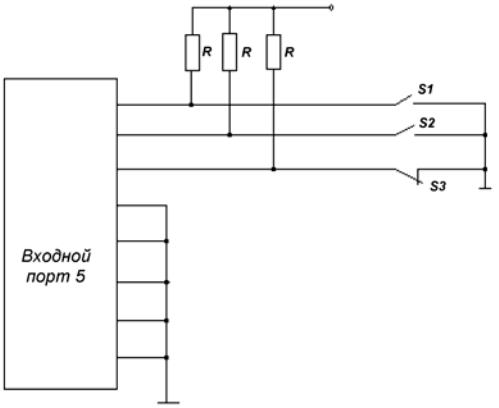


Рисунок 5.11- Схема порта с тремя тумблерами

Три $в_0, в_1, в_2$ входного соответствующем входе 0, $_7$ заземлены, т.е. на них постоянно подается нуль. Таким образом, с помощью трех тумблеров имеется возможность представить восемь различных состояний процесса:

00000000B = 00H;
00000001B = 01H;
00000010B = 02H;
00000011B = 03H;
00000100B = 04H;
00000101B = 05H;
00000110B = 06H;
00000111B = 07H.

Требуется написать программу перехода к одной из восьми подпрограмм (от A до H) в зависимости от комбинации сигналов входного порта 2.

Предположим, что соответствующие подпрограммы расположены в памяти следующим образом (рис.5.12):

Рисунок 5.12- Схема расположения подпрограммы в памяти

Первая команда подпрограммы *A* находится в ячейке с адресом 0300H, подпрограммы B - в ячейке с адресом 0400H и т.д.

Адреса первых команд подпрограмм хранятся в другой части памяти в соответствии с рис.5.12. Для записи каждого шестнадцатиразрядного адреса используются две ячейки памяти. Адрес первой команды подпрограммы *A* хранится в ячейках памяти с адресами 0200H и 0201H и т.д.

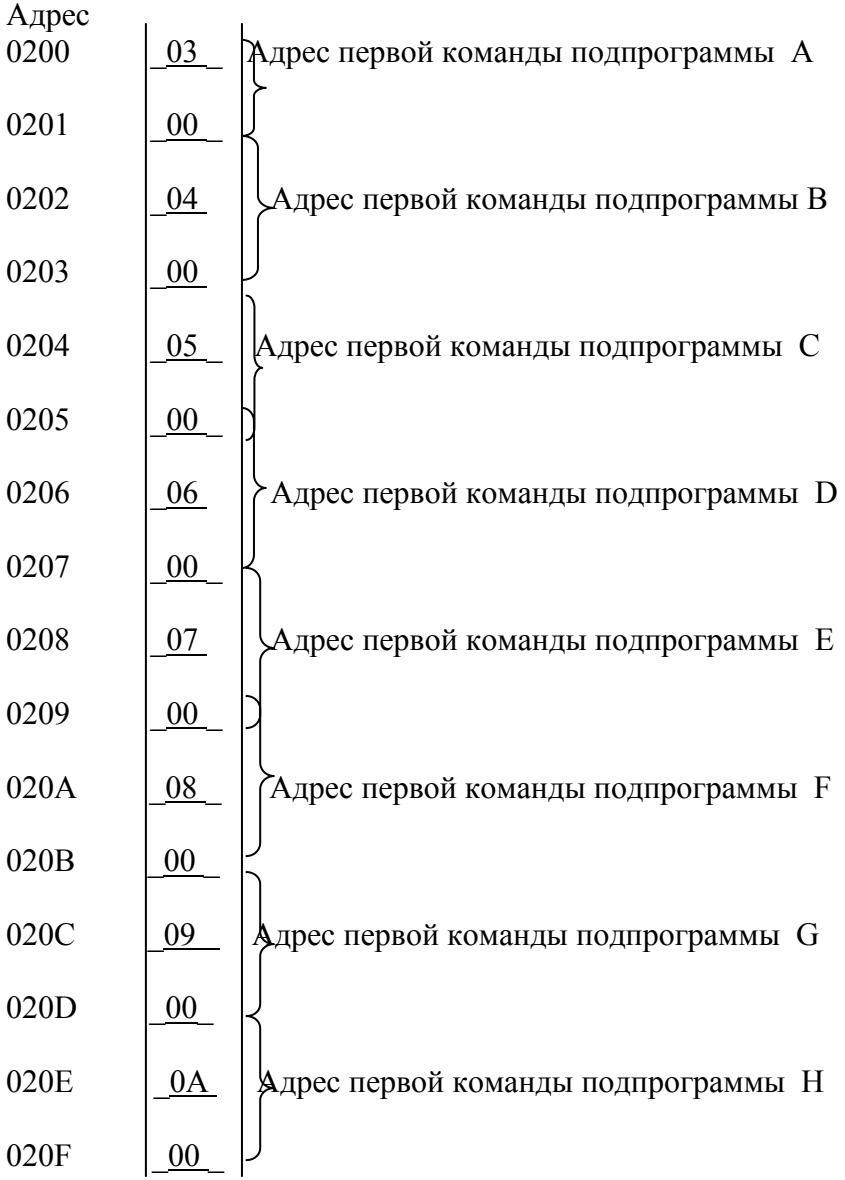


Рисунок 5.13- Вариант расположения подпрограммы в памяти

Для определения адреса ячейки памяти, в которой хранится первая команда соответствующей подпрограммы, необходимо входное слово порта 2 умножить на 2 и прибавить к числу 0200H . Например, если тумблеры S*1* и *S2* разомкнуты, а тумбле замкнут, на входе порта присутствует код 00000011 = 03H. Это означает, что должен быть выполнен переход подпрограмме *D*. Для нахождения адреса ячейки , в которой хранится первый байт адреса первой команды подпрограммы, необходимо входное слово 03H умножить на 2 прибавить к коду 0200H. В результате получаем (03H × 2 +

+0200H)0206H. Увеличение кода 0206H на 1 дает адрес ячейки памяти, в которой хранится второй байт адреса.

Программа должна выполнять следующие действия:

1) принять информацию со входа порта 2 и заслать ее в аккумулятор;

2) умнож ть на 2 содержимое аккумулятора и прибавить к числу 0200H;

3) поместить в счетчик команд содержимое ячейки памяти с полу адресом и содержимое ячейки памяти, непосредственно следующей за ней.

Присвоим этой программе имя *JUMPTA (JUMP TABLE*), так как в процедуры лежит операция сравнения содержимого адресов подпрограмм с кодом входного слова.

Составляем блок-схему алгоритма выполнения данной программы (рис.5.14).

Составляем программу, причем если это независимая программа, то ее следует начать с директивы Ассемблера *ORG*, которая сообщит Ассемблеру, что адрес 0000H ассоциирует с меткой *JUMPTA* .

Рисунок 5.14- Блок-схема алгоритма выполнения программы

Программа:

Адрес	Ассемблерный код	Комментарий
	OKG : 0000H	;Адрес первой команды *0000H*
00	*JUMPTA: IN 02H*	; Ввод из порта 2 в
01		; аккумулятор
02	*RAL*	; Умножение содержимого *A* на 2
03	*LXI H 0200H*	; Загрузка 0200*H* в
04		; регистровую пару *HL*
05		;
06	*ADD*	;Сложение содержимого *L* и *A*
07	*MOV L, A*	; Адрес в *HL*
08	*MOV D, M*	; Пересылка содержимого, адресуемого *HL* в *D*
09	*INX H*	; Инкремент содержимого *HL*
0A	*MOV E, M*	; Пересылка адреса 1-й команды подпрограммы в

DE

0B

DE и *HL*,
команд

с
HL

0C

которая
100 мкс.
Временная программой
микроЭВМ команд не
выполняет время.
применяется
метод регистр
загруж последовательно
уменьшается на единицу до тех пор, пока не станет равным нулю. Чем большую задержку требуется получить, тем большее число нужно занести в регистр.

Блок-схема, описывающая этот метод формирования временных задержек, аналогична рис.5.15.

Назовем данную подпрограмму *DELAY* (задержка) и это символическое имя будем использовать в качестве метки, указывающей начальный адрес подпрограммы.

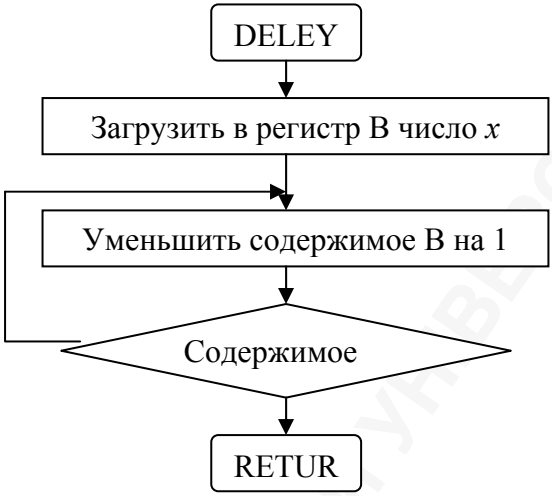


Рисунок *5*.15- Блок-схема формирования временных задержек

Занесем в регистр *B* число *X* (блок 2). Его значение зависит от требуемой задержки времени. Определить точное значение *X* можно только после написания программы, когда будет известно, сколько команд содержится в программе и какова продолжительность выполнения каждой команды. Из содержимого регистра *B* вычитается 1 (блок 3). Затем проверяется (блок 4) содержимое регистра *B*.Признак состояния НУЛЬ установится в 1, если результат вычитания станет равным нулю.

Составляем подпрограмму:

Ассемблерный код	Комментарий
DELAY: MVI B, x	; Загрузить в регистр *B* число *x*
FORWARD: DOR B	; Декрементировать содержимое регистра *B*
JNZ FORWARD	; Повторить процесс, если результат не равен 0
RPT	; Если результат нулевой

то возврат в основную

не будет
задано в этой
подпрограмме каждой
команды данная
чтобы
получить
один раз,
команды в
зависимости ь время
выполнения которой
программа команд
определяем

MVI B
DCR B = 2,5;
JNZ FORWARD = 5,0;
RET = 5,0.

Команды, которые выполняются один раз, длятся 8,5 + 3,5 + +5 = 17 мкс. Чтобы получить задержку времени, равную 100 мкс, микроЭВМ должна выполнить команды *DCR B* и *JNZ FORWARD* столько раз, что бы на этот процесс затрачивалось 100 - 17 = 83 мкс. Время выполнения этих команд равно 2,5 + 5,0 = 7,5 мкс. Таким образом, число *x* (число повторений данной пары команд) будет равно 10, и это обеспечит задержку 75 мкс, т.е. на 8 мкс меньше, чем требуется (83 -75 = 8 мкс). Можно добавить эти 8 мкс, четырехкратно использовав команду *NOP (N0 OPERATION),* длительность которой 2,0 мкс.

Окончательно подпрограмма будет иметь следующий вид:

Адрес	Ассемблерный код	Комментарий
30	*DELAY: MVI B, 10*	; Загрузить в регистр B
31		число 10
32	*FORWARD : DCR B*	; Декремент содержимого *B*
33	JNZ *FORWARD*	; Повторять процесс,
34		; если результат не
35		равен 0

```
36      NOP         ; Пустая операция
37      NOP         ;
38      NOP         ;
39      NOP         ;
3A      RET         ;
```

Пример 30 Светодиодный индикатор соединен с выходным портом 2 (рис.5.16). Светодиод загорается, если $в_0$ = = 1. Написать программу включения включения и выключения индикатора. Частота включений не имеет принципиального значения.

Можно заставить индикатор зажигаться и гаснуть, формируя на выходе $в_0$ выходного порта 2 сигналы соответственно 1 и 0 . Занесем в аккумулятор (А) число 01Н и по команде *OUT 02H* передадим его в выходной порт 2. После заданной временной задержки сделаем содержимое А равным 00Н. Если снова воспользуемся командой *OUT 02H,* то индикатор погаснет. После заданной временной выдержки снова зашлем в аккумулятор *A01H* число и опять отправим его содержимое в выходной
порт 2.

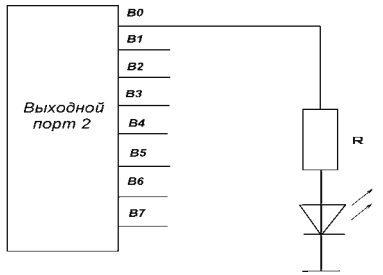


Рисунок *5.*16- Схема включения светодиода

Для получения временной задержки можно использовать подпрограмму DELAY, описанную в примере 29. Но поскольку

период 110 мкс слишком мал для различения сигнала светодиода, необходимо сформировать большую задержку, взяв соответственно большое значение *x*. Составляем блок-схему решения задачи (рис. 5.17).

Назовем данную подпрограмму *ALARM,* используя это символическое имя в качестве метки адреса первой команды.

Подпрограмма *DELAY* вызывается по команде *CALL DELAY* (блок 4) и используется для того, чтобы светодиод горел в течение определенного времени. Содержимое *A* изменяется (блок 5) с помощью операции ИСКЛЮЧАЮЩЕЕ ИЛИ и операнда 00000001B=01H.

Рисунок 5.17- Блок-схема алгоритма решения задачи

Подпрограмма:

Адрес	Ассемблерный код	Комментарий
	ORG 0000H	; Первую команду поместить по адресу *0000H*
00	*ALARM: MVI A,01H*	; Загрузить в *A 01H*
01		;
02	*FLASH: OUT 02H*	; Содержимое *A* выдать
03		; в порт 2

```
04      CALL DELAY      ; Задержка
05                      ;
06                      ;
07      XRI   01H       ; Изменить значение в0
08                      ; в A
09      JMP FLASH       ; Переход
0A                      ;
0B                      ;
```

СПИСОК ЛИТЕРАТУРЫ

1. Бессекерский В.А. и *др.* Микропроцессорные системы автоматического управления. - Л.: Машиностроение,1988.

2. Брябрин В.М. Программное обеспечение персональных ЭВМ. - М.: Наука, 1988.

3. Казаринов Ю.М. Микропроцессорный комплект К1810. Структура. Программирование. Применение.- М.: Высшая школа, 1990.

4. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: В двух томах.- М,: Радио и связь, 1988.

5. Микропроцессоры: В трёх книгах/ Под ред. Л. Н. Преснухина -М.: Высшая школа, 1986.

6. Сташин В.В. и др. Проектирование цифровых устройств на однокристальных микроконтроллерах. -М.: Энергоатомиздат, 1990.

7. Хвощ С.Т. и др. Микропроцессоры и микроЭВМ в системах автоматического управления: Справочник. -Л.: Машиностроение, 1987.

8. Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник.- М: Радио и связь, 1993.

9. Бродин В.Б., Шаругин М.И. Справочник. Микроконтроллеры: архитектура, программирование, интерфейс. - М.:ЭКОМ, 1999.

10. Документация на микроконтроллеры фирмы Intel MCS-51/151.

11. Описание микроконтроллеров семейства MCS-51.