# ВВЕДЕНИЕ

Задачей курса компьютерная схемотехника являются: освоение основных элементов ЭВМ.

Микропроцессорные системы по своей сложности, требованиям и функциям могут значительно отличаться надежностными параметрами, объемом программных средств, могут быть однопроцессорными и многопроцессорными, построенными на одном типе микропроцессорного набора или нескольких, и т.д ..

Для системного программиста важно совокупность программно - доступных средств, и аппаратная составляющая системы. В этом случае приобретает важное значение операции необходимой для выполнять системе, ее возможности.

# 1. Микропроцессорные комплекты

ЭВМ получили широкое распространение, начиная с 50-х годов. Ранее это были очень большие и дорогие устройства, используемые только в государственных учреждениях и крупных фирмах. Размеры и форма цифровых ЭВМ неузнаваемо изменились в результате разработки новых устройств, называемых микропроцессорами.

Микропроцессор (МП) - это программно-управляемое электронно -цифровое устройство, предназначенное для обработки цифровой информации и управления процессом этой обработки, выполненное на одной или нескольких интегральных схемах с высокой степенью интеграции электронных элементов.

В 1970 году Маршиан Эдвард Хофф из фирмы Intel сконструировал интегральную схему, аналогичную по своим функциям центральному процессору большой ЭВМ - первый микропроцессор Intel-4004, который уже в 1971 году был выпущен в продажу.

15 ноября 1971 можно считать началом новой эры в электронике. В этот день компания приступила к поставкам первого в мире микропроцессора Intel 4004.

Это был настоящий прорыв, потому что МП Intel-4004 размером менее 3 см был производительнее гигантской машины ENIAC. Правда работал он гораздо медленнее и мог обрабатывать одновременно только 4 бита информации (процессоры больших ЭВМ обрабатывали 16 или 32 бита одновременно), но и стоил первый МП в десятки тысяч раз дешевле.

Кристалл был 4-разрядным процессором с классической архитектурой ЭВМ гарвардского типа и изготавливался по передовой p-канальной МОП технологии с проектными нормами 10 мкм. Электрическая схема прибора насчитывала 2300 транзисторов. МП работал на тактовой частоте 750 кГц при длительности цикла команд 10,8 мкс. Чип i4004 был адресный стек (счетчик команд и три регистра стека типа LIFO), блок РОНов (регистры Сверхоперативная памяти или регистровый файл - РФ), 4-разрядное параллельное АЛУ, аккумулятор, регистр команд с дешифратором команд и схемой управления, а также схему связи с внешними устройствами. Все эти функциональные узлы объединялись между собой 4-разрядной ШД. Память команд достигала 4 Кбайт (для сравнения: объем ЗУ мини-ЭВМ в начале 70-х годов редко превышал 16 Кбайт), а РФ ЦП насчитывал 16 4-разрядных регистров, которые можно было использовать и чем 8 8-разрядных. Такая организация РОНов сохранена и в последующих МП фирмы Intel. Три регистра стека обеспечивали три уровня вложения подпрограмм. МП i4004 монтировался в пластмассовый или металлокерамический корпус типа DIP (Dual In-line Package) всего с 16 выводами.

В систему его команд входило всего 46 инструкций.

Вместе с тем кристалл располагал весьма ограниченными средствами ввода / вывода, а в системе команд отсутствовали операции логической обработки данных (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ), в связи с чем их приходилось реализовывать с помощью специальных подпрограмм. Модуль i4004 не имел возможности останова (команды HALT) и обработки прерываний.

Цикл команды процессора состоял из 8 тактов задающего генератора. Была мультиплексированная ША (шина адреса) / ШД (шина данных), адрес 12-разрядный передавался по 4-разряда.

1 апреля 1972 компания Intel начала поставки первого в отрасли 8-разрядного устройства i8008. Кристалл изготавливался по р-канальной МОП-технологии с проектными нормами 10 мкм и содержал 3500 транзисторов. Процессор работал на частоте 500 кГц при длительности машинного цикла 20 мкс (10 периодов задающего генератора).

В отличие от своих предшественников МП имел архитектуру ЭВМ принстонского типа, а как память допускал применение комбинации ПЗУ и ОЗУ.

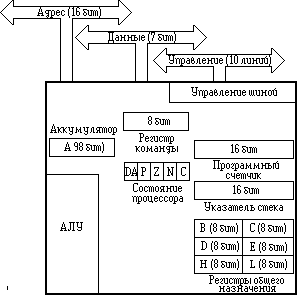
По сравнению с i4004 число РОН уменьшилось с 16 до 8, причем два регистра использовались для хранения адреса при косвенной адресации памяти (ограничение технологии - блок РОН аналогично кристаллам 4004 и 4040 в МП 8008 был реализован в виде динамической памяти). Почти вдвое сократилась продолжительность машинного цикла (с 8 до 5 состояний). Для синхронизации работы с медленными устройствами был введен сигнал готовности READY.

Система команд насчитывала 65 инструкций. МП мог адресовать память объемом 16 Кбайт. Его производительность по сравнению с четырехразрядный МП выросла в 2,3 раза. В среднем для сообщения процессора с памятью и устройствами ввода / вывода требовалось около 20 схем средней степени интеграции.

Возможности р-канальной технологии для создания сложных высокопроизводительных МП были почти исчерпаны, поэтому "направление главного удара" перенесли на n-канальную МОП технологию

# 1.1 Комплект И 8080

1 апреля 1974 МП Intel 8080 был представлен вниманию всех заинтересованных лиц. Благодаря использованию технологии П-МОП с проектными нормами 6 мкм, на кристалле удалось разместить 6 тыс. Транзисторов. Тактовая частота процессора была доведена до 2 МГц, а продолжительность цикла команд составила уже 2 мкс. Объем памяти, адресуемой процессором, был увеличен до 64 Кбайт. За счет использования 40-выводного корпуса удалось разделить ША и ШД, общее число микросхем, необходимых для построения системы в минимальной конфигурации сократилось до 6 (рис. 1).

Рисунок 1.1 Мікропроцесор Intel 8080

В РФ (регистровый файл 0 были введены указатель стека, активно используемый при обработке прерываний, а также два программно недоступных регистры для внутренних пересылок. Блок Рона был реализован на микросхемах статической памяти. Исключение аккумулятора из РФ и введение его в состав АЛУ упростило схему управления внутренней шиной.

**Новое в архитектуре МП** - использование многоуровневой системы прерываний по вектору. Такое техническое решение позволило довести общее число источников прерываний 256 (до появления БИС контроллеров прерываний схема формирования векторов прерываний требовала применения до 10 дополнительных чипов средней интеграции). В i8080 появился механизм прямого доступа в память (ПДП) (как ранее в универсальных ЭВМ IBM System 360 и др.).

ПДП открыл зеленую улицу для применения в микроЭВМ таких сложных устройств, как накопители на магнитных дисках и лентах дисплеи на ЭЛТ, которые и превратили микроЭВМ в полноценную вычислительную систему.

Традицией компании, начиная с первого кристалла, стал выпуск не отдельного чипа ЦП, а семейства БИС, рассчитанных на совместное использование.

 Микропроцессорный комплект БИС серии КР580 (I8080.) Предназначен для широкого класса средств вычислительной техники и обработки информации. На основе комплекта строятся микроЭВМ контрольно-измерительных систем, микроЭВМ для управления технологическими процессами, контроллеры периферийных устройств, битных устройств и игровых автоматов и т.д..МПК КР580 выполнен по П-МОП технологии и по напряжениям логических уровней согласуется с ИС ТТЛ.

В состав базового комплекта серии КР580 входят следующие БИС:

 8-разрядный центральный процессор КР580ВМ80А (i8080)

 генератор тактовых импульсов КР580ГФ24 (I8224)

 системный контолер и шинный формирователь КР580ВК28 / 38

 (I8228 / 38)

 расширитель ввода-вывода КР580ВР43 (I8243)

 восьмиразрядный буферный регистр с тремя состояниями на выходе

 (Что неинвертирующий) КР580ИР82 (I8282)

 восьмиразрядный буферный регистр с тремя состояниями на выходе

 (Инвертующий) КР580ИР83 (I8283)

 восьмиразрядный двунаправленный шинный формирователь (неин-

 вертуючий) КР580ВА86 (I8286)

 восьмиразрядный двунаправленный шинный формирователь (инвер-

 тующие) КР580ВА87 (I8287)

 программируемый последовательный интерфейс КР580ВВ51

 (I8251A)

 программируемый таймер КР580ВИ53 (I8353)

 программируемый параллельный интерфейс КР580ВВ55А (I8255A)

 программируемый контроллер прямого доступа к памяти КР580ВТ57

 (I8257)

 программируемый контроллер прерываний КР580ВН59 (I8259)

 контроллер периферийных устройств-клавиатуры и индикации

 КР580ВВ79 (I8279)

 Контроллер электронно-лучевой трубки КР580ВГ75 (I8275).

 С комплектом совместим ряд микросхем серий 555,531,1533ы т.д ..

 Нагрузочная способность каждого выхода БИС достаточное для подключения одного входа ТТЛ схем (и 1,6 mA).

 Центральный процессорный элемент КР580ВМ80А функционально законченным равно кристальным параллельным 8-разрядным микропроцессором с фиксированной системой команд. В процессоре отсутствуют возможности аппаратного наращивания разрядности обрабатываемых данных.

**Процессор характеризуется :**

1) тактовой частотой, определяющей максимальное время выполнения переключения элементов в ЭВМ; .

2) разрядностью, т.е. максимальным числом одновременно обрабатываемых двоичных разрядов.

Разрядность МП сказывается m / n / k / и включает:.

**m** - разрядность внутренних регистров определяет принадлежность к тому или иному классу процессоров; .

**n -** разрядность шины данных, определяет скорость передачи информации.

**k -** разрядность шины адреса, определяет размер адресного пространства.

Например, МП i8088 характеризуется значениями m / n / k = 16/8/20, архитектурой. Понятие архитектуры микропроцессора включает систему команд и способы адресации, возможность совмещения выполнения команд во времени, наличие дополнительных устройств в составе микропроцессора, принципы и режимы его работы

# 1.2 Понятие архитектуры ЕВМ.Типичная архитектура ЭВМ

Выделяют понятия **микроархитектуры** и **макроархитектуры**.

**Микроархитектура** процессора - это аппаратная организация и логическая структура микропроцессора, регистры, управляющие схемы, арифметико-логические устройства, запоминающие устройства и связывающие их информационные магистрали.

**Макроархитектура** - это система команд, типы обрабатываемых данных, режимы адресации и принципы работы микропроцессора.

В общем случае под архитектурой ЭВМ понимается абстрактное представление машины в терминах основных функциональных модулей, языка ЭВМ, структуры данных.

Микропроцессор координирует работу всех устройств цифровой системы с помощью шины управления (ШУ). Кроме ШУ имеется 16-разрядная адресная шина (ША), что служит для выбора определенной ячейки памяти, порта ввода или порта вывода. По 8-разрядной информационной шине или шине данных (ШД) осуществляется двунаправленное передачи данных к процессору и от процессора. Важно отметить, что МП может посылать информацию в память микроЭВМ или к одному из портов вывода, а также получать информацию из памяти или от одного из портов ввода.

Постоянное запоминающее устройство (ПЗУ) в микроЭВМ содержит некоторую программу (на практике программу инициализации ЭВМ).

Программы могут быть загружены в запоминающее устройство с произвольной выборкой (ЗУПВ) и с внешнего запоминающего устройства (ВЗУ). Это программы пользователя.

 Команда обычно разбивается на части. Первая часть команды 1 в приведенной выше программе - команда ввода данных. Во второй части команды 1 указывается, откуда нужно ввести данные (из порта 1). Первая часть команды, предлагает конкретное действие, называется кодом операции (КиП), а вторая часть - операндом. Код операции и операнд размещаются в отдельных ячейках памяти программ. КИП сохраняется в ячейке 100, а код операнда - в очаге 101 (порт 1); последний указывает откуда нужно взять информацию.

В МП выделены еще два новых блока - регистры: аккумулятор и регистр команд.

Рассмотрим прохождения команд и данных внутри микрокомпьютера с помощью занумерованных кружков на диаграмме. Напомним, что процессор - это центральный узел, управляющий перемещением всех данных и выполнением операций.

Итак, при выполнении типичной процедуры ввода-запоминания-вывода в микроЭВМ происходит следующая последовательность действий:

1. МП выдает адрес 100 на шину адреса. По шине управления поступает сигнал, устанавливающий память программ (конкретную микросхему) в режим считывания. .

2. ЗУ программ пересылает первую команду ( "Ввести данные") по шине данных, и МП получает это закодированное сообщение. Команда находится в регистр команд. МП декодирует (интерпретирует) полученную команду и определяет, что для команды нужен операнд. .

3. МП выдает адрес 101 на ША; ШУ используется для перевода памяти программ в режим считывания. .

4. Из памяти программ на ШД пересылается операнд "Из порта 1". Этот операнд находится в программной памяти в ячейке 101. Код операнда (содержащая адрес порта 1) передается по ШД к МП и направляется в регистр команд. МП теперь декодирует полную команду ( "Ввести данные из порта 1"). .

5. МП, используя ША и ШУ, связывающие его с устройством ввода, открывает порт 1. Цифровой код буквы "А" передается в аккумулятор внутри МП и запоминает. Важно отметить, что при обработке каждой программной команды МП действует согласно микропроцедур выборки-декодирования-исполнения. .

6. МП обращается к ячейке 102 по ША. ШУ используется для перевода памяти программ в режим считывания. .

7. Код команды "Запомнить данные" подается на ШД и пересылается в МП, где помещается в регистр команд. .

8. МП дешифрует эту команду и определяет, что для нее нужен операнд. МП обращается к ячейке памяти 103 и приводит в активное состояние вход считывания микросхем памяти программ. .

9. Из памяти программ на ШД пересылается код сообщения "В ячейке памяти 200". МП воспринимает этот операнд и помещает его в регистр команд. Полная команда "Запомнить данные в ячейке памяти 200" выбрана из памяти программ и декодирована. .

10. Теперь начинается процесс выполнения команды. МП пересылает адрес 200 на ША и активизирует вход записи, относящийся к памяти данных. .

11. МП направляет информацию, хранящуюся в аккумуляторе, в память данных. Код буквы "А" передается по ШД и записывается в ячейку 200 этой памяти. Выполнено вторую команду. Процесс запоминания не разрушает содержимого аккумулятора. В нем по-прежнему находится код буквы "А". .

12. МП обращается к ячейке памяти 104 для выбора очередной команды и переводит память программ в режим считывания. .

13. Код команды вывода данных пересылается по ШД к МП, который помещает ее в регистр команд, дешифрует и определяет, что нужен операнд. .

14. МП выдает адрес 105 на ША и устанавливает память программ в режим считывания. .

15. Из памяти программ по ШД к МП поступает код операнда "В порт 10", который далее помещается в регистр команд. .

16. МП дешифрует полную команду "Вывести данные в порт 10". С помощью ША и ШУ, связывающие его с устройством вывода, МП открывает порт 10, пересылает код буквы "А" (все еще находится в аккумуляторе) по ШД. Буква "А" выводится через порт 10 на экран дисплея.

В большинстве микропроцессорных систем (МПС) передача информации осуществляется способом, аналогичным рассмотренному выше. Наиболее существенные различия возможны в блоках ввода и вывода информации.

Подчеркнем еще раз, что именно микропроцессор является ядром системы и осуществляет управление всеми операциями. Его работа представляет последовательную реализацию микропроцедур выборки-дешифрования-исполнение. Однако фактическая последовательность операций в МПС определяется командами, записанными в памяти программ.

Таким образом, в МПС микропроцессор выполняет следующие функции:.

- Выборку команд программы из основной памяти; .

- - Дешифровки команд .

- - Выполнение арифметических, логических и других операций, закодированных в командах; .

- - Управление пересылкой информации между регистрами и основной памятью, между устройствами ввода / вывода; .

- - Обработка сигналов от устройств ввода / вывода, в том числе реализацию прерываний с этих устройств; .

- - Управление и координацию работы основных узлов МП.

# 1.3 Система команд

Проектирование системы команд влияет на структуру ЭВМ. Оптимальную систему команд иногда определяют как совокупность команд, удовлетворяет требованиям проблемно-ориентированных приложений таким образом, что избыточность аппаратных и аппаратно-программных средств на реализацию редко используемых команд оказувалася минимальной.

В различных программах ЭВМ частота появления команд различна; например, по данным фирмы DEC в программах для ЭВМ семейства PDP-11 наиболее часто встречается команда передачи MOV (B), на ее долю приходится примерно 32% всех команд в типичных программах. Систему команд следует выбирать таким образом, чтобы затраты на редко используемые команды прошлого были минимальными.

При наличии статистических данных можно разработать (выбрать) ЭВМ с эффективной системой команд. Одним из подходов к достижению данной цели является разработка команд длиной в одно слово и кодирования их таким образом, чтобы разряды таких коротких команд использовать оптимально, что позволит сократить время реализации программы и ее длину.

Другим подходом к оптимизации системы команд является использование микроинструкций. В этом случае отдельные биты или группы бит команды используются для кодирования нескольких элементарных операций, выполняемых в одном командном цикле. Эти элементарные операции не требуют обращения к памяти, а последовательность их реализации определяется аппаратной логикой.

Сокращение времени выполнения программ и емкости памяти достигается за счет увеличения сложности логики управления.

Важной характеристикой команды является ее формат, определяющий структурные элементы команды, каждый из которых интерпретируется определенные образом при ее выполнении. Среди таких элементов (полей) команды выделяют следующие:

1. код операции, определяющий выполняемое действие;
2. адрес ячейки памяти, регистров процессора, внешнего устройства;
3. режим адресации;
4. операнд при использовании непосредственной адресации;
5. код анализируемых признаков для команд условного перехода.

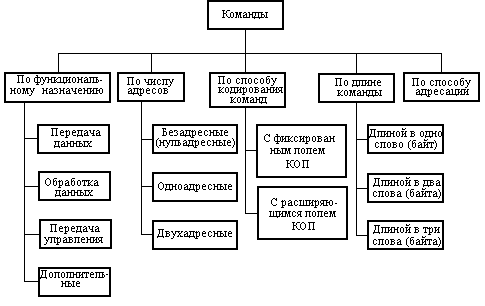
Классификация команд по основным признакам представлена ​​на рис. 2.4. Важнейшим структурным элементом формата любой команды код операции (КоП), что определяет действие, которое должно быть выполнено. Большое число КоП в процессоре очень важно, потому что аппаратная реализация команд экономит память и время. Но при выборе ЭВМ необходимо концентрировать внимание на полноте операций с конкретными типами данных, а не только на числе команд, на доступных режимах адресации. Число бит, отводимое под КИП, является функцией полного набора реализуемых команд. .

Рисунок 1.2 Классификация команд.

При использовании фиксированного числа бит под КоП для кодирования всех m команд необходимо в поле КоП выделить двоичные разряды. Однако, учитывая ограниченную длину слова мини- и микро-ЭВМ, разное функциональное назначение команд, источники и приемники результатов операций, а также то, что не все команды содержат адресную часть для обращения к памяти и периферийных устройств, в малых ЭВМ для кодирования команд широко используется принцип кодирования с переменным числом бит под поле КоП для различных групп команд.

В некоторых командах необходим только один операнд и они называются однооперандными (или одноадресной) командами в отличие от двооперандних (или двухадресный), в которых требуются два операнда. При наличии двух операндов командой обычно изменяется только один из них. Так как информация берется только из одной ячейки, очаг называется источником; ячейка, содержимое которого меняется, называется приемником.

Ниже приведен формат двухадресный (двооперандной) команды процессоров СМ.

Формат команд процессоров СМ:

а) двухадресная команда; .

б) одноадресная команда.

Пример кодирования двухадресных команд в процесорах СМ

|  |  |  |
| --- | --- | --- |
| КоП | Мнемоніка команди | Коментар |
| 0001 0010 0110 1110 | MOV CMP ADD SUB | Передача данных    Сравнение    Добавление    Вычитания |
| 0000 1000 | - - | Кодирование группы одноадресных команд |

Четирехбитний КоП (биты 15-12) кодирует ряд двухоперандных операций, приведенных в таблице 1. Биты (11-6) и (5-0) для команд данного типа определяют адреса источника и приемника данных. Как видно из таблицы, комбинации 0000 и 1000 КИП определяют группы одноадресных команд (рис 1, б). КИП 1 (биты 15-12), что соответствует кодам 0000 и 1000, определяет группу одноадресных команд, а КИП 2 (биты 11-6) кодирует конкретную операцию команд данной группы. Таким образом, команды, использующие один операнд, кодируются 10-битным КИП (биты 15-6).

Наиболее гибкая команда требует до четырех операндов. Например, команда добавления может указывать адреса складывающихся адрес результата и адрес следующей команды. Если для задания адреса требуется 16 бит, то чотирьохоперандна команда займет 8 байт памяти, не из учитывая код операции. Итак, получится медленно действующая ЭВМ с огромной памятью. Поэтому в большинстве микроЭВМ любым команд нужно не более двух операндов. Это достигается следующими приемами:.

1. Адрес следующей команды указывается только в командах переходов в других случаях очередная команда выбирается из ячеек памяти, следующих за выполненной командой. .

2. Использование его организации, в которой находится один из операндов, для запоминания результата (например, сумма запоминается в ячейки первого операнда).

Локализацию и обращение к операндам обеспечивают режимы адресации. При введении нескольких режимов адресации необходимо отвести в команде биты, указывающие режимы адресации для каждого операнда. Если предусмотрено восемь режимов адресации, то для задания каждого из них нужно три бита.

Почти во всех форматах команд первые биты отводятся для кода операции, но дальше форматы команд разных ЭВМ сильно отличаются друг от друга. Другие биты должны определять операнды или их адреса, и поэтому они используются для комбинации режимов, адресов регистров, адресов памяти, относительных адресов и непосредственных операндов. Обычно длина команды варьируется от 1 до 3 и даже 6 байт.

По форматах команд можно судить о возможностях ЭВМ.

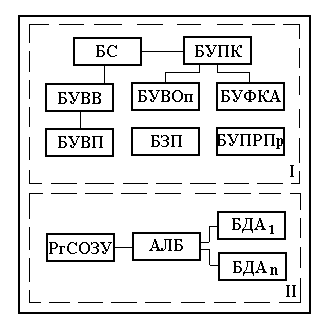
# 1.4 Логическая структура микропроцессора

Логическая структура микропроцессора, то есть конфигурация составной микропроцессора логических схем и связей между ними, определяется функциональным назначением. Именно структура задает соединение логических блоков микропроцессора и то, как эти блоки должны быть связаны между собой, чтобы полностью соответствовать архитектурным требованиям.

Срабатывания электронных блоков микропроцессора в определенной последовательности приводит к выполнению заданных архитектурой микропроцессора функций, то есть к реализации вычислительных алгоритмов. Те же функции можно выполнить в микропроцессорах со структурой, отличающимся набором, количеством и порядком срабатывания логических блоков.

Различные структуры микропроцессоров, как правило, обеспечивают их возможности, в том числе и различную скорость обработки данных. Логические блоки микропроцессора с развитой архитектурой показаны на рисунке 1.3.

Рисунок 1.3 Общая логическая структура микропроцессора.



**- I - управляющая часть,**

- БС - блок синхронизации

-БУПК - Блок управления последовательностью команд

- Бувоп - блок управления выполнением операций;

- БУФКА - блок управления формированием кодов адресов;

- БУВП - блок управления виртуальной памятью

- БЗП - блок защиты памяти;

- Бупрпр - блок управления прерыванием работы процессора;

- БУВВ - блок управления вводом / выводом;

**-II - операционная часть;**

- Ргсозу - реестровый зверхоперативний запоминающее устройство;

- Алб - арифметико-логический блок;

- БДА - блок дополнительной арифметики;

При проектировании логической структуры микропроцессоров необходимо рассмотреть:.

1) номенклатуру электронных блоков, необходимой и достаточной для реализации архитектурных требований; .

2) способы и средства реализации связей между электронными блоками; .

3) методы отбора если не оптимальных, то наиболее рациональных вариантов логических структур из возможного числа структур с соединением блоков, отличающихся, и конфигурацией связей между ними.

При проектировании процессора приводятся в соответствие,

внутренняя сложность кристалла и количество выводов корпуса. Относительный рост числа элементов по мере развития микроэлектронной технологии во много раз превышает относительное увеличение числа выводов корпуса, поэтому проектирование БИС в виде конечного автомата, а не в виде набора схем, реализующих некоторый набор логических переключающих функций и схем памяти, дает возможность получить функционально законченные блоки и устройства ЭВМ.

Использование микропроцессорных комплектов БИС позволяет создать микроЭВМ для широких областей применения вследствие программной адаптации микропроцессора к конкретной области применения: изменяя программу работы микропроцессора, изменяют функции информационно-управляющей системы. Поэтому за счет составления программы работы микропроцессоров в конкретных условиях работы определенной системы можно получить оптимальные характеристики последней.

Если уровень только программной "настройки" микропроцессоров не позволит получить эффективную систему, доступен следующий уровень проектирования - микропрограммный. За счет изменения содержания ПЗУ или программируемой логической матрицы (ПЛМ) можно "настроиться" на более специфические черты системы обработки информации. В этом случае частично за счет изменения микропрограмм затрагивается аппаратный уровень системы. Технико-экономические последствия здесь связаны лишь с ограниченным вмешательством в технологию изготовления управляющих блоков микроЭВМ.

Изменение аппаратного уровня информационно-управляющей микропроцессорной системы, включающей в себя функциональные БИС комплекта, одновременно с конкретизацией микропрограммного и программного уровней позволяет наилучшим образом удовлетворить требованиям, предъявляемым к системе.

Решение задач управления в конкретной системе чисто аппаратными средствами (аппаратная логика) дает выигрыш в быстродействии, однако приводит к сложностям при модификации системы. Микропроцессорное решение (программная логика) является более медленным, но более гибким решением, позволяющим развивать и модифицировать систему. Изменение технических требований к информационно-управляющей микропроцессорной системы ведет лишь к необходимости перепрограммирования работы микропроцессора. Именно это качество обеспечивает высокую логическую гибкость микропроцессоров, определяет возможность их широкого использования, а значит и крупносерийного производства.

# 1.5 Режимы адресации

Для взаимодействия с различными модулями в ЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода / вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Конечно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода / вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации операнда в его исполнительный адрес.

Все способы адресации памяти можно разделить на:,

1) прямой, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержания какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.); ,

2) косвенный, который предполагает, что в команде содержится значение косвенного адреса, то есть адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждой микроЭВМ реализованы только некоторые режимы адресации, использование которых, как правило, определяется архитектурой МП.

# 1.6 Типы архитектур

Существует несколько подходов к классификации микропроцессоров по типу архитектуры. Так, выделяют МП с CISC (Complete Instruction Set Computer) архитектурой, характеризуемого полным набором команд, и RISC (Reduce Instruction Set Computer) архитектурой, определяет систему с сокращенным набором команд одинакового формата, выполняемых за один такт МП.

Определяя в качестве основной характеристики МП разрядность, выделяют следующие типы МП архитектуры:,

- С фиксированной разрядностью и списком команд (однокристальные) ,

- - С наращиваемой разрядностью (секционные) и микропрограммного управлением.

Анализируя адресные пространства программ и данных, определяют МП с архитектурой фон Неймана (память программ и память данных находятся в едином пространстве и нет никаких признаков, указывающих на тип информации в ячейке памяти) и МП с архитектурой Гарвардской лаборатории ( память программ и память данных разделены, имеют свои адресные пространства и способы доступа к ним).

Мы рассмотрим более подробно основные типы архитектурных решений, выделяя связь со способами адресации памяти. ,

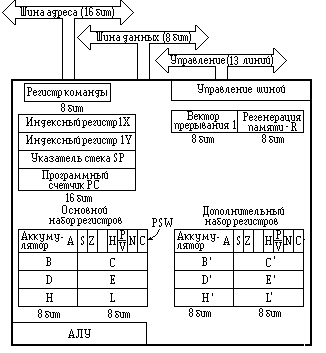
1. Регистрационная архитектура определяется наличием достаточно большого регистрового файла внутри МП. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах. Размер регистра обычно фиксирован и совпадает с размером слова, физически реализуемого в оперативной памяти. К любому регистра можно обратиться непосредственно, поскольку регистры представлены в виде массива запоминающих элементов - регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти)К данному типу архитектуры относится процессор фирмы Zilog.Процессор Z80 - детище фирмы Zilog кроме Расширенное системы команд, одного номинала питания и способности выполнять программы, написанные для i8080, имел архитектурные "изюминка". ,

Рисунок 1.4 Процессор Z80 фирми Zilog.

В дополнение к основному набору РОН, в кристалле был реализован второй комплект аналогичных регистров. Это значительно упрощало работу при вызове подпрограмм или процедур обслуживания прерываний, поскольку программист мог использовать для них альтернативный набор регистров, избегая сохранения в стеке содержимого Рона для основной программы с помощью операций PUSH. Кроме того, в систему команд был включен ряд специальных инструкций, ориентированных на обработку отдельных битов, а для поддержки регенерации динамической памяти в схему процессора введены соответствующие аппаратные средства. Z80 применялся в машинах Sinclair ZX, Sinclair Spectrum, Tandy TRS80.

Предельный вариант - архитектура с адресацией с помощью аккумуляторов (меньший набор команд). ,

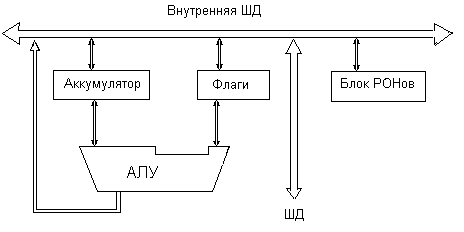


Рисунок 1.5 Процессор МСS-48 Intel с одним аккумулятором

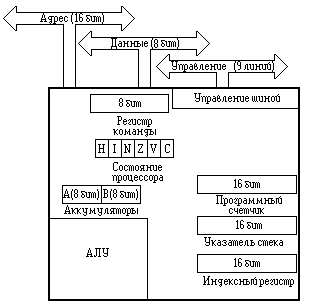
МП фирмы Motorola имел ряд существенных преимуществ. Прежде всего, кристалл МС6800 требовал для работы одного номинала питания, а система команд оказалась достаточно прозрачной для программиста. Архитектура МП также имела ряд особенностей.

Рисунок 1.6 Процессор МС 6800 фирмы Motorola.

Микропроцессор МС 6800 содержал два аккумулятора, и результат операции АЛУ мог быть помещен в каждой из них. Но самой ценной качеством структуры МС 6800 было автоматическое сохранение в стеке содержимого всех регистров процессора при обработке прерываний (Z80 требовалось для этого несколько команд PUSH). Процедура восстановления РОН из стека тоже выполнялась аппаратно.

2. Стековая архитектура дает возможность создать поле памяти с упорядоченной последовательностью записи и выборки информации. В общем случае команды неявно адресуются к элементу стека, расположенному на его вершине, или к двум верхним элементов стека.

3. Архитектура МП, ориентированная на оперативную память (типа "память-память»), обеспечивает высокую скорость работы и большую информационную емкость рабочих регистров и стека при их организации в оперативной памяти. ,

Архитектура этого типа не предполагает явного определения аккумулятора, регистров общего назначения или стека; все операнды команд адресуются к области основной памяти.

С точки зрения важности для пользователя-программиста под архитектурой в общем случае понимают совокупность следующих компонентов и характеристик:,

- Разрядности адресов и данных; ,

- - Состав, имен и назначения программно-доступных регистров; ,

- - Форматов и системы команд ,

- - Режимов адресации памяти; ,

- - Способов машинного представления данных разного типа; ,

- - Структуры адресного пространства; ,

- - Способа адресации внешних устройств и средств выполнения операций ввода / вывода; ,

- - Классов прерываний, особенностей инициирования и обработки прерываний.

# 1.7 Устройство управления

Коды операции команд программы, воспринимаемые управляющей частью микропроцессора, расшифрованы и преобразованы в ней, дают информацию о том, какие операции нужно выполнить, в памяти расположены данные, куда надо направить результат и где расположена следующая за выполняемой команда.

Управляющее устройство имеет достаточно средств для того, чтобы после восприятия и интерпретации информации, получаемой в команде, обеспечить переключение (срабатывание) всех необходимых функциональных частей машины, а также для того, чтобы подвести к ним данные и принять полученные результаты. Именно срабатывания, то есть изменение состояния двоичных логических элементов на противоположное, позволяет с помощью коммутации вентилей выполнять элементарные логические и арифметические действия, а также передавать необходимые операнды в функциональные части микроЭВМ.

Устройство управления в строгой последовательности в рамках тактовых и цикловых временных интервалов работы процессора (такт - минимальный рабочий интервал, в течение которого происходит одно элементарное действие; цикл - интервал времени, в течение которого выполняется одна машинная операция) осуществляет: выборку команды; интерпретацию ее с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов) установление номенклатуры и временной последовательности всех функциональных управляющих сигналов; генерацию управляющих импульсов и передаче их на управляющие шины функциональных частей микрокомпьютер и вентили между ними; анализ результата операции и изменение своего состояния так, чтобы определить местоположение (адрес) следующей команды.

# 1.8 Особенности программного и микропрограммного управления

В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов: программный и микропрограммный.

Выполнение операций в машине сводится к элементарным преобразований информации (передача информации между узлами в блоках, сдвиг информации в узлах, логические поразрядные операции, проверка условий и т.д.) в логических элементах, узлах и блоках под влиянием функциональных управляющих сигналов блоков (устройств) управления. Элементарные преобразования, неразложимые на более простые, выполняются в течение одного такта сигналов синхронизации и называются микрооперациями.

В аппаратных (схемных) устройствах управления каждой операции соответствует свой набор логических схем, которые производят определенные функциональные сигналы для выполнения микроопераций в определенные моменты времени. При этом способе построения устройства управления реализация микроопераций достигается за счет один раз соединенных между собой логических схем, так ЭВМ с аппаратным устройством управления называют ЭВМ с жесткой логикой управления. Это понятие относится к фиксации системы команд в структуре связей ЭВМ и означает практическую невозможность каких-либо изменений в системе команд ЭВМ после ее изготовления.

При микропрограммное реализации устройства управления в состав последнего вводится ЗП, каждый разряд исходного кода которого определяет появление определенного функционального сигнала управления. Поэтому каждой микрооперации ставится в соответствие свой информационный код - микрокоманда. Набор микрокоманд и последовательность их реализации обеспечивают выполнение любой сложной операции. Набор микроопераций называют микропрограммами. Способ управления операциями путем последовательного считывания и интерпретации микрокоманд с ЗП (наиболее часто в виде микропрограммного ЗП используют быстродействующие программируемые логические матрицы), а также кодов микрокоманд для генерации функциональных управляющих сигналов-называют микропрограммного, а микроЭВМ с таким способом управления - микропрограммное или с сохраненной (гибкой) логикой управления.

К микропрограмм выдвигают требования функциональной полноты и минимальности. Первое требование необходимо для обеспечения возможности разработки микропрограмм любых машинных операций, а второе связано с желанием уменьшить объем используемого оборудования. Учет фактора быстродействия ведет к расширению микропрограмм, поскольку осложнения последних позволяет сократить время выполнения команд программы. Преобразование информации выполняется в универсальном арифметико-логическом блоке микропроцессора. Он обычно строится на основе комбинационных логических схем.

Для ускорения выполнения определенных операций вводятся дополнительно специальные операционные узлы (например, циклические устройства сдвига). Кроме того, в состав микропроцессорного комплекта (МПК) БИС вводятся специализированные оперативные блоки арифметических расширителей.

Операционные возможности микропроцессора можно расширить за счет увеличения числа регистров. Если в реестровом буфере закрепления функций регистров отсутствует, то их можно использовать как для хранения данных, так и для сохранения адресов. Подобные регистры процессора называются регистрами общего назначения (РОН). По мере развития технологии реально осуществлено изготовление в процессоре 16, 32 и более регистров.

В целом же, принцип микропрограммного управления (ПМК) включает следующие позиции:,

1) любая операция, реализованная устройством, является последовательностью элементарных действий - микроопераций; ,

2) для управления порядком прохождения микроопераций используются логические условия; ,

3) процесс выполнения операций в устройстве описывается в форме алгоритма, представляется в терминах микроопераций и логических условий, называемого микропрограммой; ,

4) микропрограмма используется как форма представления функции устройства, на основе которой определяются структура и порядок функционирования устройства во времени.

ПМК обеспечивает гибкость микропроцессорной системы и позволяет осуществлять проблемную ориентацию микро- и мини-ЭВМ.

# 2 Процессор и его окружение

# 2.1 Процессор i8080А (КР580ВМ80А)

Микросхема I8080А (КР580ВМ80А), представляет собой 8-разрядный центральный процессорный устройство (ЦПУ) параллельной обработки данных. Устройство не имеет возможность аппаратного наращивания разрядности обрабатываемых данных, но позволяет осуществлять это программным способом. Структурная схема КР580ВМ80А представлена на рисунке 2.1.

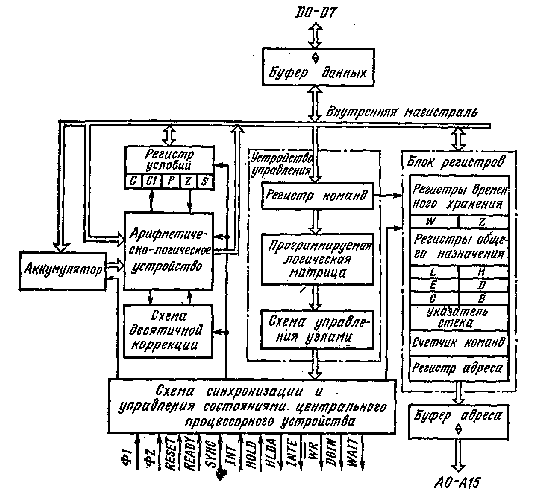
ы

Рисунок 2.1 Структурна схема КР580ВМ80А.

Назначение основных узлов и принцип их взаимодействия.

Арифметически-логическое устройство (АЛУ) обеспечивает выполнение арифметических, логических операций и операций сдвига над двоичными данными, представленными в дополнительном коде, или над двоично-десятичными данными. Устройство содержит схему десятичной коррекции, позволяет делать операции десятичной арифметики. По результатам операций в АЛУ формируется ряд признаков, записываются в регистр условий. Признак переноса С Устанавливается в единицу, если в результате выполнения команды появляется перенос из старшего разряда. Дополнительный признак переноса С1 Устанавливается в единицу при возникновении переноса из третьего разряда. Используется в командах десятичной арифметики. Признак четности Р устанавливается в единицу, если, число единиц в разрядах результата четное. Признак нуля Z устанавливается в единицу, если результат равен нулю. Признак знака S указывает знак числа и равен единице, если число отрицательное, или нулю, если число положительное.

Блок регистров делает прием, хранение и выдачу различной информации, принимающих участие в процессе выполнения программы, и содержит счетчик команд, указатель стека, регистры общего назначения, регистры временного хранения и регистр адреса. Шестнадцати разрядный счетчик команд сохраняет текущий адрес команды. Содержимое счетчика команд автоматически увеличивается после выборки каждого байта команды. Шестнадцати разрядный указатель стека содержит начальный адрес памяти, используемая для сохранения и восстановления содержимого программно-доступных регистров ЦПУ, Содержимое указателя стека уменьшается, когда данные загружаются в стек, и увеличивается, когда данные выбираются из стека. Восьми разрядные регистры общего назначения В, С, D, Е, Н, L могут применяться как накопители (обрабатываемые данные находятся в самом регистре) и указатели (16-разрядный адрес операнда определяется содержанием пары регистров). Регистры временного хранения W, Z используются для приема и временного запоминания второго и третьего байтов команд переходов, переданных с внутренней магистрали ЦПП в счетчик команд. Эти регистры являются программно-недоступными. Шестнадцати разрядный регистр адреса принимает и сохраняет в течение одного машинного цикла адрес команды или операнда и выдает его через буфер адреса на однонаправленную исходную магистраль АО-А15. Буфер адреса выполнен в виде выходных формирователей, имеющих на выходе состояние "выключено" (третье сословие),

Схема синхронизации и управления состояниями ГПК формирует машинные такты и циклы, которые координируют выполнение всех команд, и вырабатывает сигнал SYNC "Синхронизация", что определяет начало каждого машинного цикла. Для выполнения команды требуется от одного до пяти машинных циклов. Каждый цикл может состоять из 3-5 тактов (Т1-Т5), продолжительность каждого из них соответствует периоду прохождения тактовых импульсов Ф1, Ф2. Центральный процессорное устройство может находиться в трех состояниях (ожидание, восторг и остановка), продолжительность которых составляет целое число тактов и зависит от внешних управляющих сигналов.

Устройство управления формирует комплекс управляющих сигналов, организующими исполнение поступило в ЦПП команды, и состоит из регистра команд, программируемой логической матрицы (ПЛМ) и схемы управления узлами. Восьми разрядный регистр команд осуществляет прием и сохранение команды, поступающей по магистрали данных. Программируемая логическая матрица дешифрует код операции команды и формирует микрооперации согласно микропрограммы выполнения команды.

Схема управления узлами производит для различных узлов ЦПУ необходимые управляющие сигналы. Восьми разрядный буфер данных обеспечивает ввод команд и данных в ЦПП, вывод данных и состояния ЦПП через формирователи, имеющих на выходе состояние "Выключено".

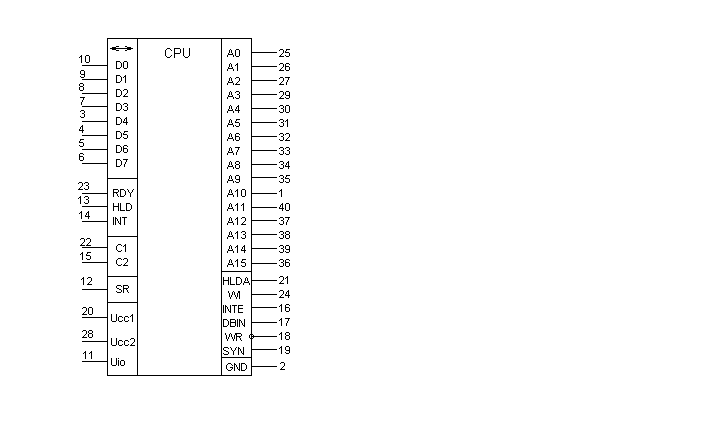


Рисунок 2.2-Интегральное исполнение ИМС И8080А

Основные технические характеристики процессора:

-розряднисть МД - 8;

-розряднисть МА - 16;

-адресний пространство - 64 Кб;

-число РОН - 6 восьми разрядных;

-организация стека - указатель стека позволяет в любой точке памяти зафиксировать вершину стека;

-организация прерываний - прерывание векторные, существует упрощенная возможность организации прерываний на восемь направлений (адресов)

-швидкодия - 500000 коротких (регистр - регистр) операций;

-тактова частота 0,5 ... 2,5 МГц;

-напряжения питания 5,12 В;

-мощность рассеяния 1,25 Вт;

-технология n-МДП;

-диапазон рабочих температур +10 ... 70 ° С;

-Uвис. р (высокого уровня) - 9 ... 13 В

-Uнизьк. р (низкого уровня) - -0,3 ... + 0,8 В

Продолжительность тактовых импульсов: С1 и (более равно) 60 нс

СО2 и 220 нс

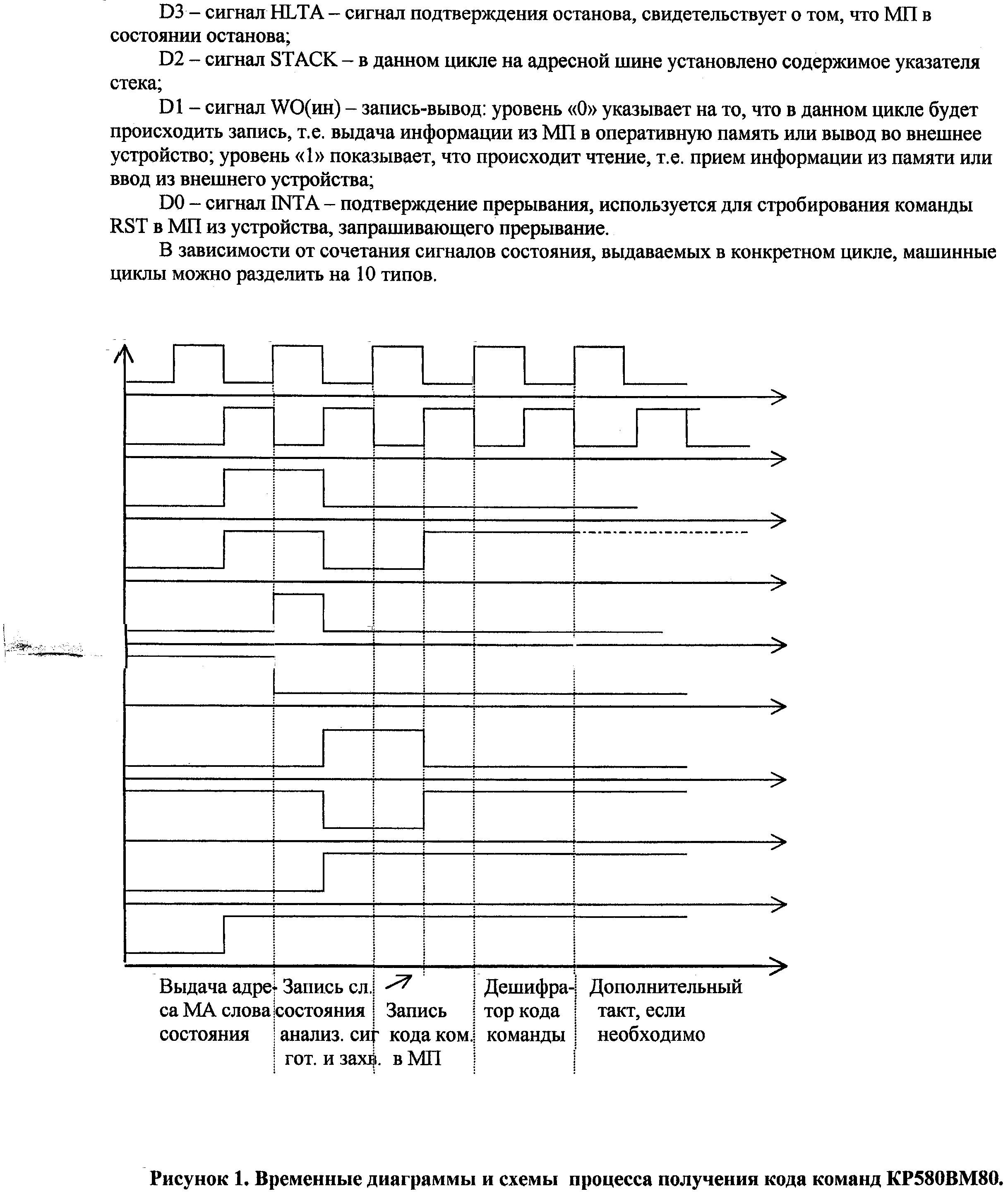


Рисунок 2.3- Временные диаграммы процесса получения кода команд И8080А

В дальнейшем в прошлом были разработаны буферные устройства, позволяющие зафиксировать информацию ША, ШД и ШУ и освободить системную шину для работы других устройств. Такие устройства называют шинимы формирователями.

**2.1 Процессор И8086А (КР580ВМ80А)**

Intel 8086 (также известный как iAPX86) - первый 16-битный микропроцессор компании Intel, разрабатывавшийся с весны 1976 года и выпущенный 8 июня 1978 года [1]. Процессор содержал набор команд, який применяется и в современных процессорах, именно от этого процессора берёт своё начало известная на сегодня архитектура x86.

Основными конкурентами микропроцессора Intel 8086 были Motorola 68000, Zilog Z8000, чипсеты F-11 и J-11 семейства PDP-11, MOS Technology 65C816. В некоторое степени, в области военных разработок, конкурентами являлись процессоры-реализации MIL-STD-1750A.

Аналогами микропроцессора Intel 8086 являлись такие разработки, как NEC V30, який был на 5% производительнее Intel 8086, но при этом был полностью с ним совместим. Советским аналогом являлся микропроцессор К1810ВМ86, входя в серию микросхем К1810.

Проект 8086 был начат в мае 1976 года, и первоначально задумывался как временная замена для амбициозно и задерживающегося проекта iAPX 432 (также известного как 8800). Это была попытка, с одной стороны, противостоять менее запаздывавшим 16-и 32-битными процессорам вторых производителей (таких как Motorola, Zilog и National Semiconductor), а с другой - борьбы с угрозой от Zilog Z80 (разработанного командой под руководством ушедшего из Интел Федерико Фаджина), который стал очень успешным. Первая версия архитектуры 8086 (система команд, прерывания, работа с памятью и вводом-выводом) была разработана с середины мая до середины августа Стивеном Морзе. Потом команда разработчиков была увеличена до четырёх Человек, который представили два основных проектных документа - «8086 Architectural Specifications» и «8086 Device Specifications». При разработке НЕ использовалось специализированных CAD-программ, а диаграммы были исполнены из текстовых символов. Использовались уже опробованные элементы микроархитектуры и физической реализации, в основном от Intel 8085.

Описание [править | правит вики-текст]

Рынок 8-разрядных микропроцессоров в конце 1970-х годов был переполнен, и Intel, оставляя попытки закрепиться на нём, выпускает свой первый 16-битный процессор. Процессор Intel 8086 представляет собой модернизированный процессор Intel 8080, и хотя разработчики не ставили перед собой цель достичь полной совместимости на программном уровне, большинство программ, написанных для Intel 8080, способны выполняться и на Intel 8086 после перекомпиляция. Новый процессор несёт в себе множество изменений, Которые позволили значительно (в 10 раз) увеличить производительность по сравнении с предыдущим поколением процессоров компании.

**Регистры**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Регистры общего назначения** | | | | | | | | | | | | | | | | |
| AH | | | | | | | | AL | | | | | | | | **AX** (primary accumulator) |
| BH | | | | | | | | BL | | | | | | | | **BX** (base, accumulator) |
| CH | | | | | | | | CL | | | | | | | | **CX** (counter, accumulator) |
| DH | | | | | | | | DL | | | | | | | | **DX** (accumulator, other functions) |
| **Индексные регистры** | | | | | | | | | | | | | | | | |
| SI | | | | | | | | | | | | | | | | **S**ource **I**ndex |
| DI | | | | | | | | | | | | | | | | **D**estination **I**ndex |
| **Указательные регистры** | | | | | | | | | | | | | | | | |
| BP | | | | | | | | | | | | | | | | **B**ase **P**ointer |
| SP | | | | | | | | | | | | | | | | **S**tack **P**ointer |
| **Регистр состояния** | | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | (bit position) |
| - | - | - | - | O | D | I | T | S | Z | - | A | - | P | - | C | Флаги |
| **Сегментные регистры** | | | | | | | | | | | | | | | | |
| CS | | | | | | | | | | | | | | | | **C**ode **S**egment |
| DS | | | | | | | | | | | | | | | | **D**ata **S**egment |
| ES | | | | | | | | | | | | | | | | **E**xtra**S**egment |
| SS | | | | | | | | | | | | | | | | **S**tack **S**egment |
| **Указатель команды** | | | | | | | | | | | | | | | | |
| IP | | | | | | | | | | | | | | | | **I**nstruction **P**ointer |

Всего в процессоре Intel 8086 было 14 16-разрядных [регистров](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0): 4 регистра общего назначения (AX, BX, CX, DX), 2 индексных регистра (SI, DI), 2 указательных (BP, SP), 4 сегментных регистра (CS, SS, DS, ES), программный счётчик или указатель команды (IP) и [регистр флагов](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D1%84%D0%BB%D0%B0%D0%B3%D0%BE%D0%B2) (FLAGS, включает в себя 9 флагов). При этом регистры данных (AX, BX, CX, DX) допускали адресацию не только целых регистров, но и их младшей половины (регистры AL, BL, CL, DL) и старшей половины (регистры AH, BH, CH, DH), что позволяло использовать не только новое 16-разрядное [ПО](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), но сохраняло совместимость и со старыми программами (правда, их необходимо было, по крайней мере, [перекомпилировать](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80)).

**Шины**[[править](https://ru.wikipedia.org/w/index.php?title=8086&veaction=edit&vesection=6) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=8086&action=edit&section=6)]

Размер [шины адреса](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0) был увеличен с 16 бит до 20 бит, что позволило адресовать 1 Мбайт (220 байт) памяти. [Шина данных](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) была 16-разрядной. Однако в микропроцессоре шина данных и шина адреса использовали одни и те же контакты на корпусе. Это привело к тому, что нельзя одновременно подавать на системную шину адреса и данные. Мультиплексирование адресов и данных во времени сокращает число контактов корпуса до 20, но и замедляет скорость передачи данных. Из-за того что выполнение отдельных команд меньше цикла ввода-вывода, в процессор был введён (впервые) буфер команд на 6 байт, это позволяло выполнять, например, команды сложения параллельно командам ввода-вывода.

**Работа с памятью**[[править](https://ru.wikipedia.org/w/index.php?title=8086&veaction=edit&vesection=7) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=8086&action=edit&section=7)]

**Первый вариант**

Для того чтобы адресовать больший, чем [Intel 8080](https://ru.wikipedia.org/wiki/Intel_8080), объём памяти, потребовалось изменить [способ адресации](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D0%B8) памяти. Ведь если использовать старые методы, когда адрес к ячейке памяти содержался в указательных регистрах, то пришлось бы увеличивать размер этих самых регистров, чтобы иметь возможность обращаться к большему объёму памяти. Поэтому для адресации 1 Мбайт памяти применили следующую схему. На шину адреса подавался физический адрес размером 20 бит, который формировался путём сложения содержимого одного из сегментных регистров (16 бит), умноженного на 24, с содержимым указательного регистра: таким образом, адресация ячейки памяти производилась по *номеру*[*сегмента*](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) и *эффективному адресу ячейки в сегменте* (называемому также [смещением](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F))). Если результат сложения оказывался больше, чем 220 − 1, то 21-й бит отбрасывался; такая процедура называется «заворачиванием» адреса ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *address wraparound*). Этот метод впоследствии (после появления защищённого режима) назвали [реальным режимом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B5%D0%B6%D0%B8%D0%BC) адресации процессора, такой режим позволяет адресовать до 1 Мбайт памяти.

**Второй вариант**

Для того чтобы адресовать 1 мегабайт памяти (20 бит адреса) с использованием 16-битных регистров используется сегментирование. Старшие 4 бит адреса выводятся на отдельные контакты корпуса, а младшие 16 выводятся на совмещённую шину адреса-данных. Но граница сегмента не жёсткая, а плавающая. Для того, чтобы адресовать нужный сегмент, используются 16-битные регистры сегмента, значение которых сдвигается на 4 бита вверх и складывается с указательным 16-битным регистром. Полученное значение — 20-битный адрес памяти или устройства выводится на контакты. Если результат сложения оказывался больше чем 1 мегабайт, выводятся только младшие 20 бит адреса, а старший, 21-й бит, отбрасывается.

[](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:8086_adress.png)

Схема, показывающая работу реального режима адресации процессора Intel 8086 и выше

Таким образом, память разделяется на сегменты, размером 64 Кбайт каждый и начинающиеся с адреса, кратного 16 (4 бита двоичного смещения вверх любого из регистра-указателей процессора), сегменты могли перекрываться или совпадать (граница [параграфа](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B3%D1%80%D0%B0%D1%84_(%D0%B5%D0%B4%D0%B8%D0%BD%D0%B8%D1%86%D0%B0_%D0%B8%D0%B7%D0%BC%D0%B5%D1%80%D0%B5%D0%BD%D0%B8%D1%8F)));

В компьютере, подобном [IBM PC](https://ru.wikipedia.org/wiki/IBM_PC), разработчики сэкономили 1 микросхему и решили не использовать разделение адресных пространств для памяти и для устройств ввода-вывода (т. н.*верхняя память* — [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *upper memory*) использовались для видеопамяти и [BIOS](https://ru.wikipedia.org/wiki/BIOS)-а, это ограничивало память, доступную пользователю, объёмом в 640 Кбайт (т. н. *обычная память* —[англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *conventional memory*; страницы 0~9).

На то время такой [режим адресации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D0%B8) обеспечивал множество преимуществ: ёмкость памяти могла составлять до 1 Мбайт, хотя команды оперировали 16-битными адресами; упрощалось использование отдельных областей памяти для программы, её данных и [стека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA); упрощалась разработка устройств, совместимых друг с другом.

**Система команд**[[править](https://ru.wikipedia.org/w/index.php?title=8086&veaction=edit&vesection=8) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=8086&action=edit&section=8)]

Система команд процессора Intel 8086 состоит из 98 команд (и более 3800 их вариаций): 19 команд передачи данных, 38 команд их обработки, 24 команды перехода и 17 команд управления процессором. Возможно 7 режимов адресации. Микропроцессор не содержал команды для работы с числами с плавающей запятой. Данная возможность реализовывалась отдельной микросхемой, называемой [математический сопроцессор](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80), который устанавливался на материнской плате. Сопроцессор вовсе не обязательно должен был быть произвёден Intel (модель [Intel 8087](https://ru.wikipedia.org/wiki/Intel_8087)), к примеру, некоторые производители микросхем, такие как [Weitek](https://en.wikipedia.org/wiki/Weitek), выпускали более производительные сопроцессоры, чем Intel.

Система команд процессора Intel 8086 включает в себя несколько очень мощных строковых инструкций. Если инструкция имеет префикс REP (повтор), то процессор будет выполнять операции с блоками — перемещение блока данных, сравнение блоков данных, присвоение определённого значения блоку данных определенной величины, и т. д., то есть, одна инструкция 8086 с префиксом REP может выполнять 4—5 инструкций, выполняемых на некоторых других процессорах. Но следует упомянуть, что подобные приёмы были реализованы и в других процессорах — Zilog Z80 имел инструкции перемещения и поиска блоков, а Motorola 68000 может выполнять операции с блоками, используя всего две команды.

В микропроцессоре Intel 8086 была использована примитивная форма [конвейерной](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B2%D0%B5%D0%B9%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80)) обработки. Блок интерфейса с шиной подавал поток команд к исполнительному устройству через 6-[байтовую](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82) очередь команд. Таким образом, выборка и выполнение новых команд могли происходить одновременно. Это значительно увеличивало пропускную способность процессора и лишало необходимости ожидать считывание команды из памяти при занятом другими операциями интерфейсе микросхемы.

Микрокомпьютеры на основе Intel 8086[[править](https://ru.wikipedia.org/w/index.php?title=8086&veaction=edit&vesection=9) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=8086&action=edit&section=9)]

Для работы процессора Intel 8086 требовался полностью новый 16-разрядный набор микросхем поддержки (шинные формирователи,[мультиплексоры](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BE%D1%80) и демультиплексоры, [интерфейсные](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) микросхемы, набор различных [контроллеров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80) и т. п.). Так как рынок на тот момент был ориентирован на 8-разрядные микропроцессоры, то производителей производящих 16-разрядные микросхемы поддержки почти не было. Это привело к тому, что данные микросхемы если и присутствовали, то в весьма ограниченном ассортименте и зачастую имели завышенную рыночную цену. Таким образом, крайне мало производителей решились использовать процессор Intel 8086 в своих персональных компьютерах. [Intel](https://ru.wikipedia.org/wiki/Intel), столкнувшись с проблемой сбыта, разработала и выпустила в [1979 году](https://ru.wikipedia.org/wiki/1979_%D0%B3%D0%BE%D0%B4) процессор [Intel 8088](https://ru.wikipedia.org/wiki/Intel_8088), который отличался от Intel 8086 в основном только 8-битной шиной данных. Однако, этот шаг позволил использовать совместно с Intel 8088 большой ассортимент 8-битных микросхем. На основе Intel 8088 было создано большое число персональных компьютеров, которые оставались популярными в [1970-е](https://ru.wikipedia.org/wiki/1970-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B) и [1980-е годы](https://ru.wikipedia.org/wiki/1980-%D0%B5_%D0%B3%D0%BE%D0%B4%D1%8B), а также определили популярность набора команд [x86](https://ru.wikipedia.org/wiki/X86) и, соответственно, дальнейшее развитие как процессоров Intel x86, так и компьютеров на основе данной линейки микропроцессоров. Знаменитый персональный компьютер [IBM PC](https://ru.wikipedia.org/wiki/IBM_PC) (модель IBM 5150) — прародитель всех современный [IBM PC-совместимых машин](https://ru.wikipedia.org/wiki/IBM_PC-%D1%81%D0%BE%D0%B2%D0%BC%D0%B5%D1%81%D1%82%D0%B8%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) — был построен именно на Intel 8088.

Тем не менее, дальнейшее развитие 16-разрядных систем, со временем, привело к выпуску большого ассортимента 16-разрядных микросхем поддержки от различных производителей по доступным ценам. Это позволило разрабатывать дешёвые и разнообразные микрокомпьютеры на процессорах Intel 8086. Однако, к [1982 году](https://ru.wikipedia.org/wiki/1982_%D0%B3%D0%BE%D0%B4) в ассортименте Intel появились более современные и производительные процессоры с 16-разрядными шинами данных, Intel 80186 и Intel 80286, и производительные компьютеры стали строиться преимущественно на процессоре [Intel 80286](https://ru.wikipedia.org/wiki/Intel_80286) и, реже, на основе [Intel 80186](https://ru.wikipedia.org/wiki/Intel_80186). Но всё же в основе некоторых микрокомпьютеров использовался Intel 8086, одним из таких является [Mycron](https://ru.wikipedia.org/w/index.php?title=Mycron&action=edit&redlink=1) 2000 — первый коммерческий микрокомпьютер на базе Intel 8086. Машина для обработки текстов [IBM](https://ru.wikipedia.org/wiki/IBM) Displaywriter, [Compaq](https://ru.wikipedia.org/wiki/Compaq) DeskPro и Wang Professional Computer также использовали Intel 8086.

Технические характеристики[[править](https://ru.wikipedia.org/w/index.php?title=8086&veaction=edit&vesection=10) | [править вики-текст](https://ru.wikipedia.org/w/index.php?title=8086&action=edit&section=10)]

* Дата анонса: 8 июня 1978 года
* Тактовая частота: от 4 до 10 МГц
  + производительность:
  + 5 (модель 8088), при частоте 4,77 ([IBM PC](https://ru.wikipedia.org/wiki/IBM_PC)) — 0,33 [MIPS](https://ru.wikipedia.org/wiki/MIPS_(%D0%B1%D1%8B%D1%81%D1%82%D1%80%D0%BE%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B5))
  + 8 (модель 8086-2 — 0,66 MIPS)
  + 10 (модель 8086-1 — 0,75 MIPS)
  + Приблизительные затраты времени на операции, процессорных циклов (EA — время, необходимое для расчета эффективного адреса памяти, которое варьируется от 5 до 12 циклов):
    - Суммирование: 3—4 (регистровое), (9—25) +EA — при операциях с памятью
    - Умножение без знака: 70—118 (регистровое), (76—139) +EA — при операциях с памятью
    - Умножение знаковое: 80—154 (регистровое), (86—160) +EA — при операциях с памятью
    - Деление без знака: 80—162 (регистровое), (86—168) +EA — при операциях с памятью
    - Деление знаковое: 101—184 (регистровое), (107—190) +EA — при операциях с памятью
    - Перемещение данных: 2 (между регистрами), (8—14) +EA — при операциях с памятью
* Разрядность [регистров](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%B0): 16 бит
* Разрядность [шины данных](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85): 16 бит
* Разрядность [шины адреса](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D0%BD%D0%B0_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0): 20 бит
* Объём адресуемой памяти: 1 Мбайт
* Адресное пространство I/O: 64 Кбайт
* Количество транзисторов: 29 000
* Техпроцесс: 3000 нм (3 мкм)
* Площадь кристалла: ~30 мм2 (по другим данным, 16 мм2)
* Максимальное тепловыделение корпуса: 1,75 Вт (фактическое потребление — 0,65 Вт)
* Напряжение питания: +5 В
* Разъём: DIP-40
* Корпус: 40-контактный керамический или пластиковый [DIP](https://ru.wikipedia.org/wiki/DIP), позже — 56-контактный [QFP](https://ru.wikipedia.org/wiki/QFP) и 44-контактный [PLCC](https://ru.wikipedia.org/wiki/PLCC)
* Поддерживаемые технологии: 98 инструкций
* Объём очереди команд: 6 байт (кэш-буфер команд)

**2 Семейство процессоров Intel x86**

Первым представителем семейства Intel x86, или, согласно официальной

классификации фирмы Intel (Integrated Electronics, США), семейства про-

цессоров IA (Intel Architecture), является микропроцессор 8086, разрабо-

танный к 1978 году. Программы, написанные для него, выполняются на

всех последующих процессорах семейства, включая все современные мо-

дели. Предшествующие процессоры - 8080, 8085 и 4004 (разработка 1967

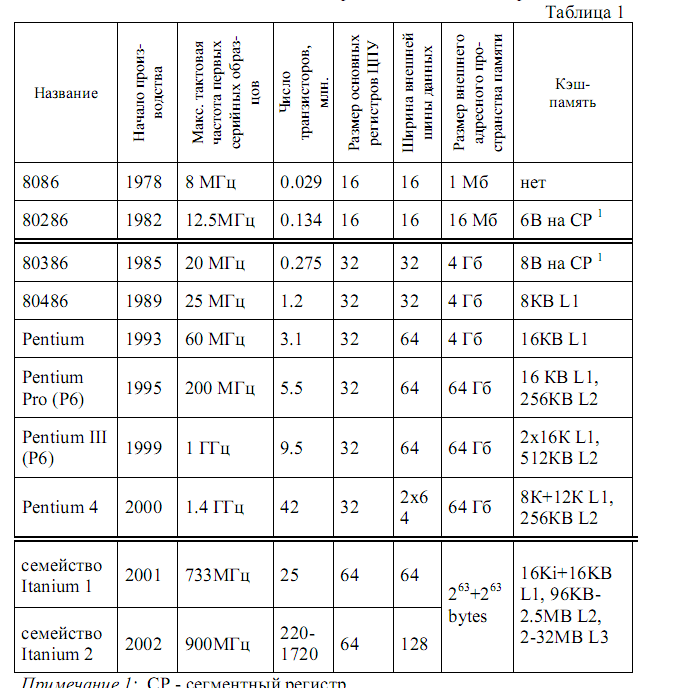
г.) из-за несовместимости по объектному коду, остаются вне семейства,

являясь, тем не менее, важными этапами на пути развития Intel x86 [3]. Се-

годня процессоры этого семейства стали стандартом де-факто для боль-

шинства персональных компьютеров (ПК) во всем мире. Ниже приводятся

основные данные наиболее известных представителей этой серии:



**2.1 Микроархитектура процессоров 8086 и Pentium Pro**

Микропроцессор 8086 ориентирован на выполнение команд параллельно с их выборкой и может быть условно разделен на две части, работающие асинхронно ( Рис. 2.1): устройство сопряжения с внешними шинами (УС) и устройст-во обработки (УО). Устройство сопряжения обеспечивает формирование 20-разрядного физического адреса памяти, выборку команд и операндов

из памяти, организацию очередности команд и запоминание результатов выполнения команд в памяти. В состав УС входит шесть 8-разрядных регистров очереди команд, четыре 16-разрядных сегментных регистра, 16-разрядный регистр обмена и 16-разрядный сумматор адреса, интерфейс с

внешними шинами. Регистры очереди команд организованы по принципу FIFO - «первым пришел - первым вышел». УС готово выполнить цикл выборки 16-разрядного слова из памяти всякий раз, когда в очереди освобождаются, по меньшей мере, два байта, а УО извлекает из очереди команды

по мере их выполнения. При выполнении команд передачи управления, например условных и безусловных переходов, очередь очищается УС и начинает заполняться заново.

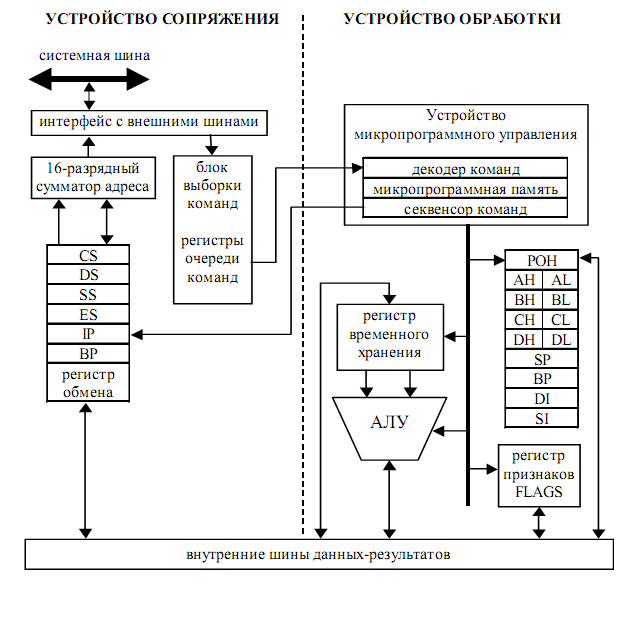


Рис. 2.1. Микроархитектура процессора 8086

Устройство обработки предназначено для выполнения операций по обработке данных и состоит из устройства микропрограммного управления (УМУ), 16-разрядного АЛУ, восьми 16-разрядных регистров общего назначения и регистра признаков. Команды из очереди, сформированной УС, поступают в УМУ, где декодируются и выполняются в 16-разрядном АЛУ согласно процедурам, записанным в памяти микропрограмм. Последова-тельное выполнение команд обеспечивается секвенсором команд, часть

которого (регистр счетчика команд IP) изображена в составе УС, т.к. именно УС записывает в IP смещение следующей команды, т.е. положение новой команды относительно начала сегмента команд. УО обменивается данными с УС через внутреннюю 16-разрядную шину и регистр обмена ( Рис. 2.1).

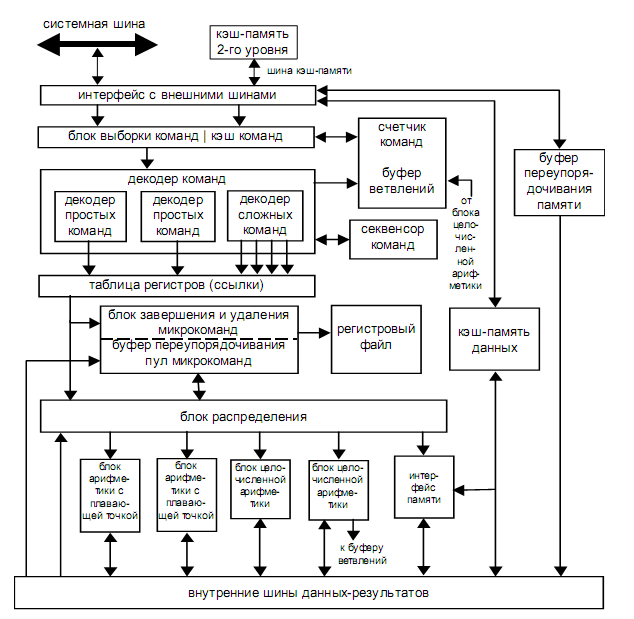


Рис. 2.2. Микроархитектура процессоров семейства Pentium Pro

Для сравнения приводится (Рис. 2.2) микроархитектура процессоров шестого поколения Pentium Pro. Как видно, архитектура со времени 8086 претерпела не только количественные (разрядность внутренних и внешних шин данных, регистров), но и существенные качественные изменения. Pentium Pro имеет суперскалярную архитектуру, т.е. может одновременно выполнять несколько команд за один такт. Эту возможность обеспечивают несколько АЛУ - два блока целочисленной арифметики и два блока с плавающей точкой. Другая важная особенность – т.н. динамическое исполнение – команды разбиваются на простейшие операции, порядок независимого исполнения которых определяется блоком «завершения и удаления микрокоманд» с буферами переупорядочивания и пулом микрокоманд

(Рис. 2.2). На Рис. 2.3 представлены обозначения микропроцессоров для принци-

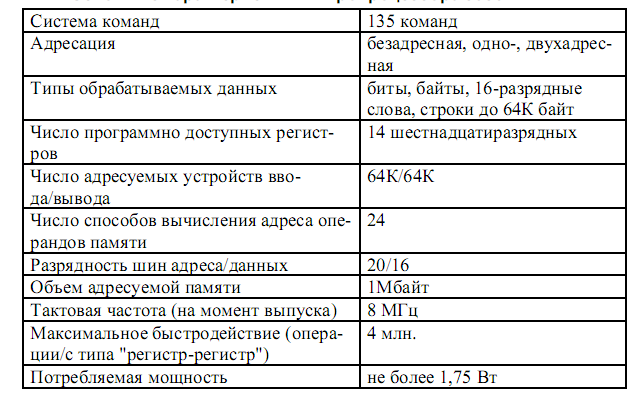
пиальных электрических схем, на которых видны внешние шины и сигналы 16-разрядного 8086 и 32-разрядного 80486.

**2.2 Система команд и методы адресации процессоров**

8086/8088

Процессоры 8086/8088 – первые в семействе x86, отличаются друг от друга шириной шины данных: в 8088 – 8-разрядная шина, в 8086 – 16разрядная. Поэтому чтение и запись 16-разрядных данных выполняются примерно в два раза медленнее для 8088, т.к. возможна передача только одного байта за один цикл, а не 16-разрядного слова как у 8086.

**2.2.1 Основные характеристики микропроцессора 8086**



2.2.2 Регистры процессора

Микропроцессор 8086 имеет 12 программно-доступных шестнадцатиразрядных регистров ( Рис. 2.1), регистр счетчика команд IP (Instruction Pointer) и регистр флагов (или регистр состояния процессора) FLAGS.

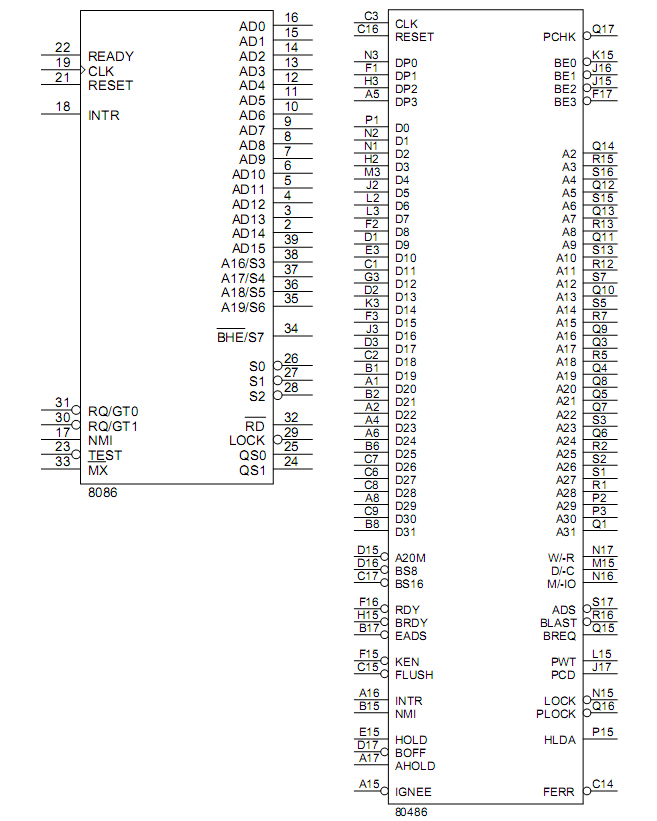


Рис. 2.3 ИС процессоров 8086 и 80486 в DIP и PGA корпусах

Среди программно-доступных регистров выделяют следующие группы (см. Рис. 2.4):

• Регистры данных: AX – аккумулятор (Accumulator); BX - базовый регистр (Base); CX - регистр счетчика (Counter); DX - регистр данных (Data).

• Регистры-указатели (индексные регистры): SI - индекс источника (Source Index); DI - индекс приемника (Destination Index); BP - указатель базы (Base Pointer); SP - указатель стека (Stack Pointer).

• Сегментные регистры: SS - сегмент стека (Stack Segment); DS - сегмент данных (Data Segment); ES - дополнительный сегмент (Extended data Segment); CS – сегмент кода (Code Segment).

16-битные регистры AX, BX, CX, DX состоят из двух 8-битных половин, к которым можно независимо обращаться по именам AH, BH, CH, DH - старшие байты и AL, BL, CL, DL - младшие байты.



Рис. 2.4 Регистры процессора 8086

Биты (или флаги) регистра признаков FLAGS разделяются на условные, отражающие результат предыдущей операции ALU, и управляющие, от которых зависит выполнение специальных функций.

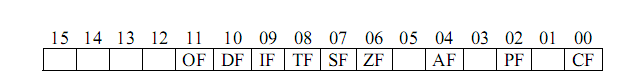


Рис. 2.5 Флаги процессора 8086

Флаги TF, IF, DF относятся к управляющим флагам, остальные - к ус-

ловным (флагам состояния). Флаги отражают следующие условия:

• флаг переноса CF устанавливается в 1 при переносе/заеме, возни-

кающем при сложении/вычитании байтов или слов, а также прини-

мает значение выдвигаемого бита при сдвигах операнда;

• флаг четности PF устанавливается в 1, если младшие 8 бит резуль-

42

тата имеют четное число единиц;

• флаг вспомогательного переноса AF устанавливается в 1, если есть

перенос/заем из младшей тетрады в старшую 8- или 16-битного ре-

зультата в операциях десятичного сложения/вычитания;

• флаг нуля ZF устанавливается в 1 при нулевом результате операции;

• флаг знака SF устанавливается в 1 при отрицательном результате

операции при использовании дополнительного кода;

• флаг переполнения OF устанавливается в 1 при потере старшего

бита результата сложения или вычитания;

• если флаг направления DF установлен в 1, то используется авто-

декрементная адресация при выполнении операции обработки строк,

если DF установлен в 0, то используется автоинкрементная адреса-

ция;

• если флаг прерывания IF установлен в 1, то внешние маскируе-

мые прерывания разрешены, иначе запрещены;

• если флаг трассировки TF установлен в 1, то процессор переходит

в состояние прерывания после выполнения каждой операции, что

позволяет проводить пошаговую отладку программ.

**2.2.3 Организация памяти**

Хотя процессор имеет 20-разрядную адресную шину, которая соединяет его с физической памятью, он оперирует с 16-битными логическими адресами, состоящими из 16-разрядного базового адреса сегмента и 16-разрядного смещения в сегменте. Физические, 20-разрядные адреса дан-

ных и команд формируются путем сложения содержимого регистров указателей и смещенного на 4 бита влево содержимого сегментных регистров. Т.н. эффективный адрес данных получается как сумма содержимого регистров BX или BP, содержимого регистров SI или DI и смещения.

Затем из эффективного адреса и содержимого сегментного регистра формируется физический адрес (Рис. 2.6). В формировании физического адреса команды участвуют IP и CS. Таким образом, адресное пространство разбивается на 4 сегмента емкостью 64К адресов по числу сегментных ре-

гистров. Регистр CS указывает на текущий сегмент кода (программы), откуда выбираются команды. Регистр DS указывает на текущий сегмент данных, в котором содержатся переменные. Регистр SS адресует текущий сегмент стека, в котором реализуются все стековые операции. Наконец,

регистр ES определяет текущий дополнительный сегмент данных. Смещенное содержимое сегментного регистра определяет положение сегмента в 20-разрядном адресном пространстве, а регистры-указатели определяют положение команды или данных внутри сегментов.

Поскольку при формировании эффективного адреса содержимое сегментного регистра сдвигается на 4 бита, сегмент всегда начинается с адреса, кратного 16, т.е. на границе 16-байтового блока памяти (параграфа). Сегменты в памяти могут располагаться как последовательно, так и с на-

ложением друг на друга. Если программа превышает 64 Кбайт, то необходимо перезагружать сегментный регистр CS новым значением базового адреса. Точно также, если данные превышают 64 Кбайт, то необходимо перезагрузить регистр DS.

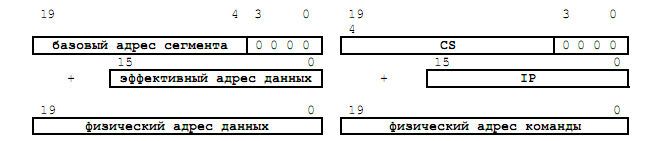


Рис. 2.6 Формирование физического адреса данных и команды

Память организована в виде одномерного массива байтов с физически-

ми адресами от 0000016 до FFFFF16. Две области адресного пространства

памяти зарезервированы для выполнения специальных функций, связан-

ных с обработкой прерываний и системным сбросом. Этими областями яв-

ляются первые 128 байт (физические адреса 00000 - 0007F) и последние

16 байт (физические адреса FFFF0 - FFFFF). Данные области использо-

вать для других целей нельзя.

Байты в памяти организуются в слова таким образом, что байту, имею-

щему меньший адрес, соответствуют менее значимые позиции разрядов в

слове. Каждый байт или слово памяти адресуется с помощью 20-битного

адреса, причем в случае адресации слова адрес указывает на его младшую

часть. Например, адрес 0000016 может обозначать и байт с этим адресом,

что условно записывается в виде [00000] = 34h, и слово с таким же адре-

сом, что записывается в виде [00000] = 1234h. Тогда старший байт слова,

[00001] = 12h. Квадратные скобки обозначают ячейку памяти, адрес кото-

рой находится в этих скобках, h – шестнадцатеричную систему счисления.

Команды, байты и слова можно размещать по любому адресу байта, одна-

ко рекомендуется размещать слова в памяти по четным адресам, так как

процессор может передавать такие слова за один цикл обращения к памя-

ти. Слово с четным адресом называется выравненным на границу слова.

Слова с нечетными адресами (невыравненные) также допустимы, однако

они считываются в два раза медленнее (требуют два цикла обращения к

памяти).

2.2.4 Форматы команд

Команды i8086 имеют переменную длину от 1 до 6 байт. По числу об-

рабатываемых операндов команды подразделяются на безадресные, одно-

адресные и двухадресные. В двухадресных командах результат всегда за-

писывается по первому адресу и только один из операндов может нахо-

диться в памяти. При этом в мнемонике, операнд-приемник записывается

слева от запятой-разделителя операндов, а операнд-источник – справа.

44

Всего существует четыре источника операндов: тело команды, регистр,

память и порт ввода/вывода. В первом случае операнд называется непо-

средственным.

Первый байт команды содержит код операции (КОП), в состав которого

могут входить специальные разряды d, s и w. При w=1 операции выполня-

ются с 16-разрядными словами, при w=0 – с байтами. Разряд d определяет

направление передачи данных в двухоперандных командах: из регистра в

регистр/память (d=0) или из регистра/памяти в регистр (d=1). S – определя-

ет расширение 8-битных непосредственных данных до полного размера

(s=1) или нет (s=0). При некоторых сочетаниях команд и методов адреса-

ции (регистровый метод адресации) положение операнда может задаваться

непосредственно в байте кода операции (см. следующий раздел), но чаще

для этого используется т.н. «постбайт».

В командах, имеющих длину 2 и более байта, второй байт называется

постбайтом. Он выполняет функции кодирования адресов операндов.

Байты 3 - 6 присутствуют в команде в зависимости от типа адреса опе-

ранда, описанного постбайтом и наличия непосредственного операнда.

Постбайт состоит из трех полей: режима - MOD, регистра - REG и ре-

гистра/памяти - R/M. Поле MOD занимает 2 бита (6 и 7) постбайта. Поле

REG занимает 3 бита (3-5) постбайта. Поле R/M занимает 3 бита (0-2) по-

стбайта. Полями MOD и R/M совместно кодируется тип адреса операнда,

находящегося в памяти или регистре. 32 значения этих полей определяют

нахождение операнда 24 возможными методами адресации либо в одном

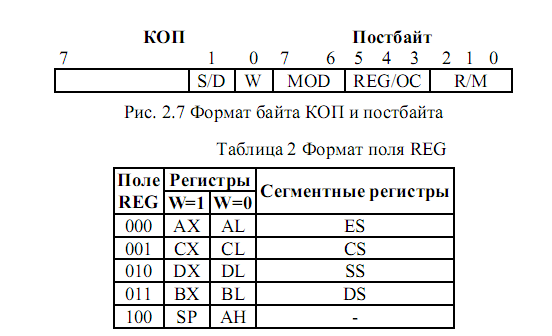
из 8 регистров.

Ниже приведена структура байта кода операции и постбайта, а в таб-

лицах показано формирование адресов регистровых операндов и адресов

операндов памяти. DISP8 и DISP16 – смещения длиной 8 и 16 бит, распо-

ложенные в команде непосредственно за байтом адресации.



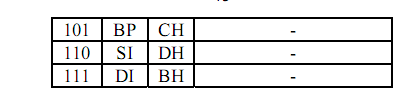
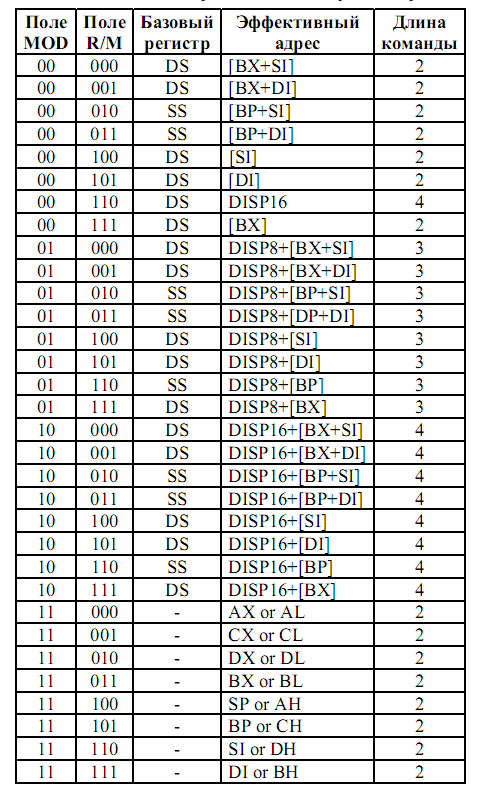


Таблица 3 Кодирование типа адреса операнда



Команде может предшествовать префикс – байт со специальным коди-

рованием, которое изменяет операцию следующей за ним команды. В сис-

теме команд процессоров 8086/8088 есть два таких префикса – REP (RE-

Peat) для повторения команд обработки строк и SEG (SEGment) для прямо-

го указания команде сегментного регистра вместо регистра, используемого

по умолчанию.

2.2.5 Методы адресации

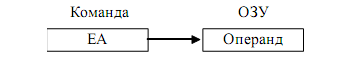
Методы адресации можно разделить на два класса: адресация данных и

адресация переходов. Все методы можно отнести к одной из следующих

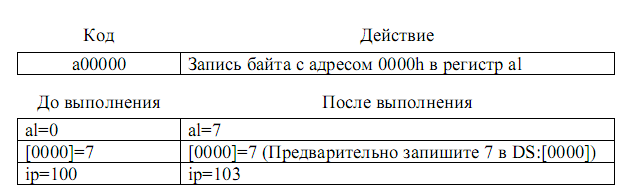
групп:

Прямая адресация.

16-битный эффективный адрес (EA) операнда является частью команды:



Пример: mov al, [0000h]



Примечание: запись в квадратных скобках обозначает ячейку, адрес кото-

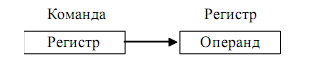
рой (смещение по отношению к DS) записан в квадратных скобках.

Регистровая адресация.

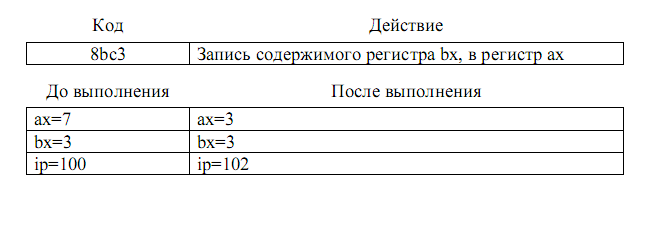
Операнд содержится в определяемом командой регистре. 16-битный

операнд может находиться в регистрах AX, BX, CX, DX, SI, DI, SP или BP,

а 8-битный - в регистрах AH, AL, BH, BL, CH, CL, DH, DL:



Пример: mov ax, bx



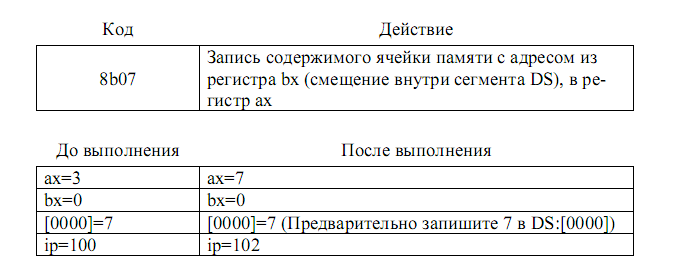
Косвенно-регистровая адресация.

Эффективный адрес операнда находится в базовом регистре BX или ин-

дексном регистре (SI или DI):



Пример: mov ax, [bx]



Физический адрес определяется парой сегмент-смещение (например

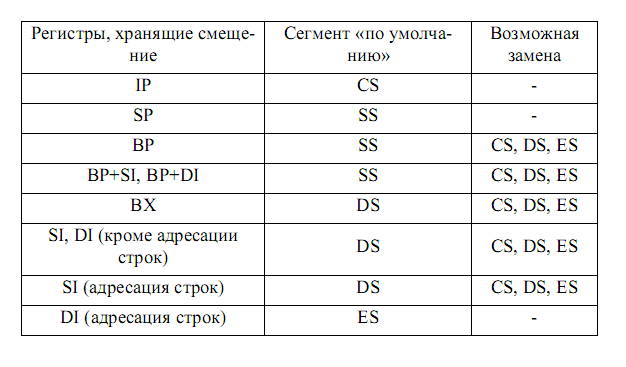
CS:IP – адрес следующей команды), и для каждого регистра, содержащего

смещение, есть сегментный регистр, заданный по умолчанию. Некоторые

сегменты разрешается принудительно переназначать, мнемоническое обо-

значение переназначения – «сегментный\_регистр:смещение», в коде ко-

манды появляется дополнительный (первый) байт – байт замены сегмента.



Пример: mov ax, cs:[bx]

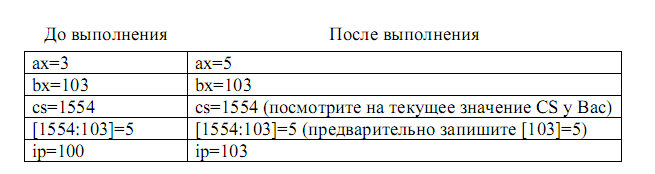
Код Действие

2e8b07

Запись содержимого ячейки памяти с адресом из

регистра bx (смещение внутри сегмента СS), в ре-

гистр ax



Физический адрес определяется парой сегмент-смещение (например

CS:IP – адрес следующей команды), и для каждого регистра, содержащего

смещение, есть сегментный регистр, заданный по умолчанию. Некоторые

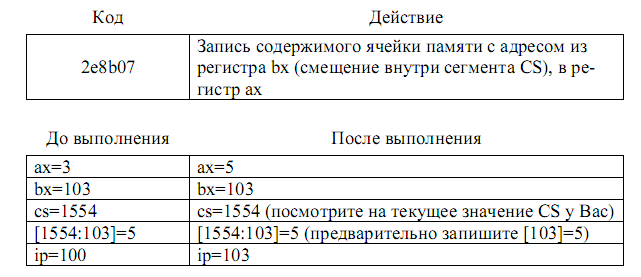
сегменты разрешается принудительно переназначать, мнемоническое обо-

значение переназначения – «сегментный\_регистр:смещение», в коде ко-

манды появляется дополнительный (первый) байт – байт замены сегмента.



Пример: mov ax, cs:[bx]



Непосредственная адресация.

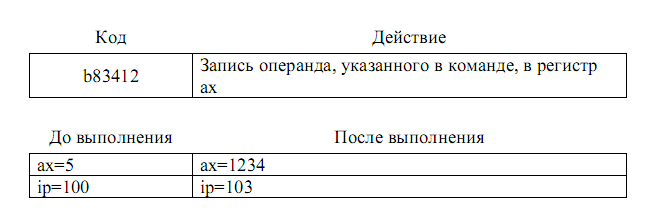
Операнд длиной байт или слово является частью команды. Операнд по-

мещается в последние байты команды, причем младший байт следует пер-

вым (находится по меньшему адресу).



Пример: mov ax, 1234h



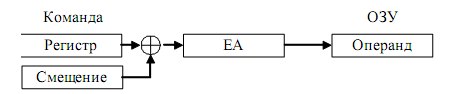
Относительная косвенно-регистровая адресация.

Эффективный адрес операнда – сумма 8- или 16-разрядного смещения и

значения одного из базовых или индексных регистров. Этот метод также

называют базовым, если используются регистры BX, BP или индексным,

при использовании SI, DI.



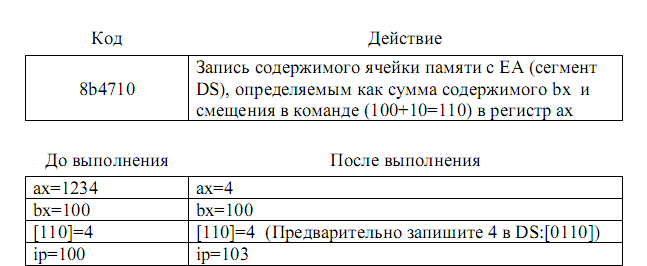
Базовая адресация обеспечивает возможность работы со структурами

49

данных, размещенными в памяти, например, с соседними ячейками памя-

ти относительно эффективного адреса в базовом регистре.

Пример: mov ax, [bx+10], другая форма записи - mov ax, 10[bx]

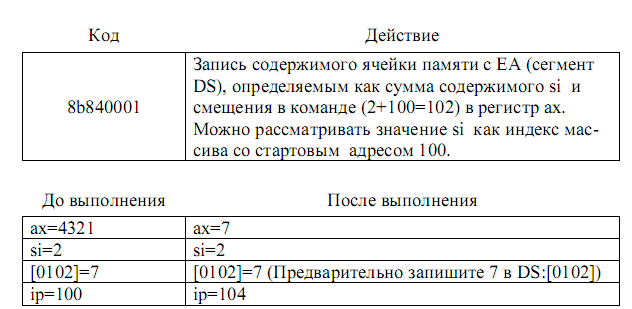


Индексный метод адресации удобен при обработке массивов, когда

смещение указывает стартовый адрес массива, а содержимое индексного

регистра соответствует индексу массива.

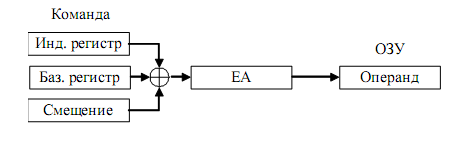
Пример: mov ax, 0100[si], другая форма записи - mov ax, [si+0100]



Базовая индексная адресация.

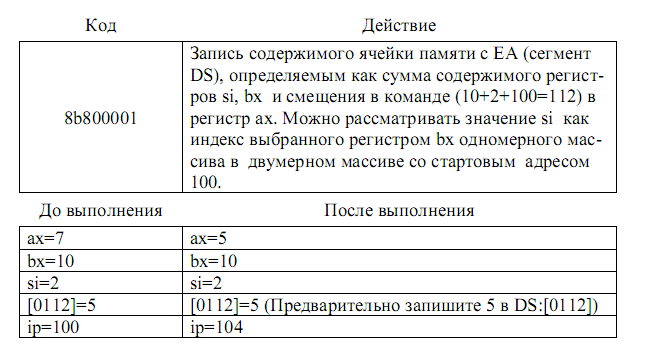
Эффективный адрес равен сумме содержимого базового (BX или BP) и

индексного (SI или DI) регистров.



50

Пример: mov ax, 100[bx][si], другая форма записи - mov ax, [bx+si+0100]



Неявная адресация.

Неявная адресация задается операцией. Например, в командах обработ-

ки строк неявно используются регистры SI, DI. В командах управления

циклами неявно используется регистр CX и т.д.

Стековая адресация.

Стековая адресация применяется в командах работы со стеком PUSH и

POP. Для этих команд адрес операнда находится в указателе стека SP и ав-

томатически уменьшается или увеличивается на два при записи в стек или

при чтении из стека. Заполнение стека происходит в направлении умень-

шения адресов ячеек памяти. Стек может обмениваться данными с регист-

рами общего назначения и сегментными регистрами. Команды обмена

данными между стеком и памятью содержат байт адресации, в котором 3-

разрядное поле REG, совместно с полем КОП, идентифицирует команду.

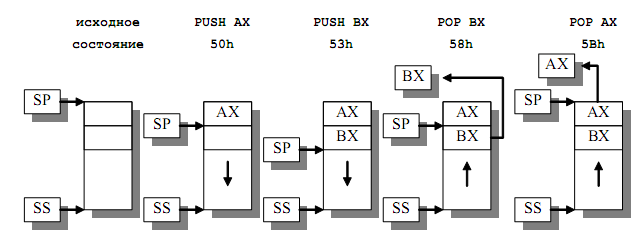


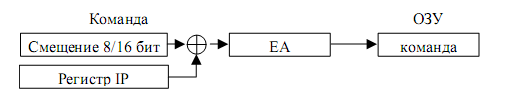
Рис. 2.8 Работа со стеком

51

Адресация переходов. Внутрисегментный прямой переход.

Адрес следующей команды определяется суммированием смещения со

знаком, представленным в двоично-дополнительном коде, и значения IP:

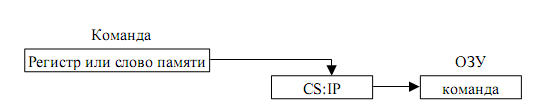


Пример: jmp 100

Адресация переходов. Внутрисегментный косвенный переход.

Содержимое регистра IP заменяется содержимым регистра или слова

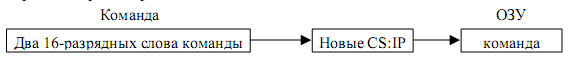
памяти, заданным любым методом адресации, кроме непосредственного:



Пример: jmp [bx]

Адресация переходов. Межсегментный прямой переход.

Содержимое регистров CS и IP заменяются словами из команды:

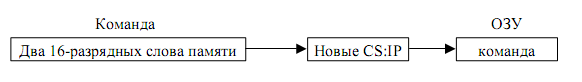


Адресация переходов. Межсегментный косвенный переход.

Содержимое регистров CS и IP заменяется словами, последовательно рас-

положенными в памяти. Слова могут быть указаны любым методом адре-

сации, кроме непосредственного:



2.2.6 Система команд

Команды передачи данных

предназначены для пересылок данных между регистрами и памятью. Эту

группу команд можно разделить на четыре подгруппы: команды пересыл-

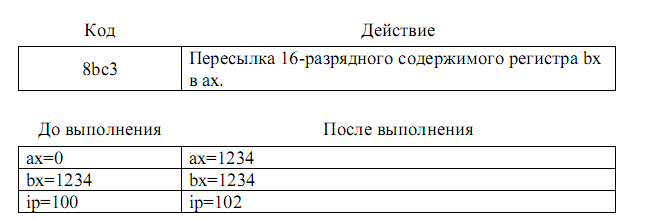
ки данных общего назначения; команды, использующие аккумулятор; ко-

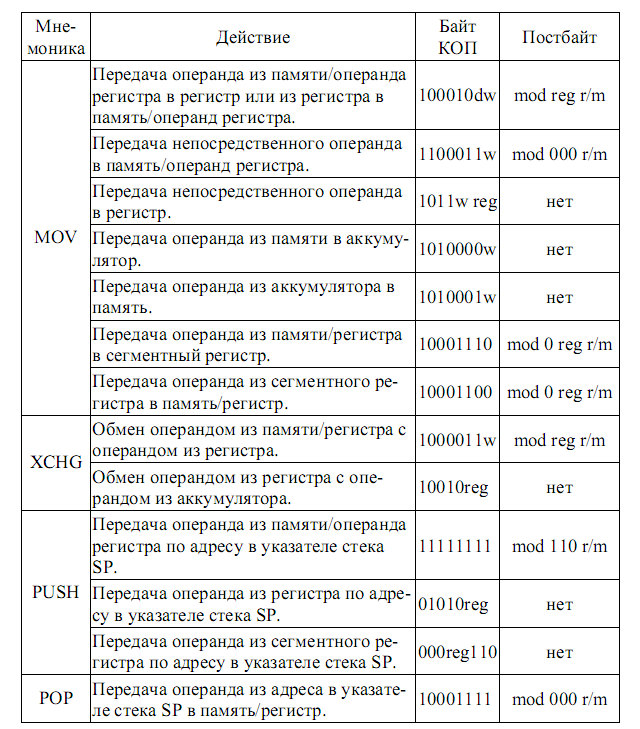
манды пересылки адресных объектов; команды пересылки флагов. Все

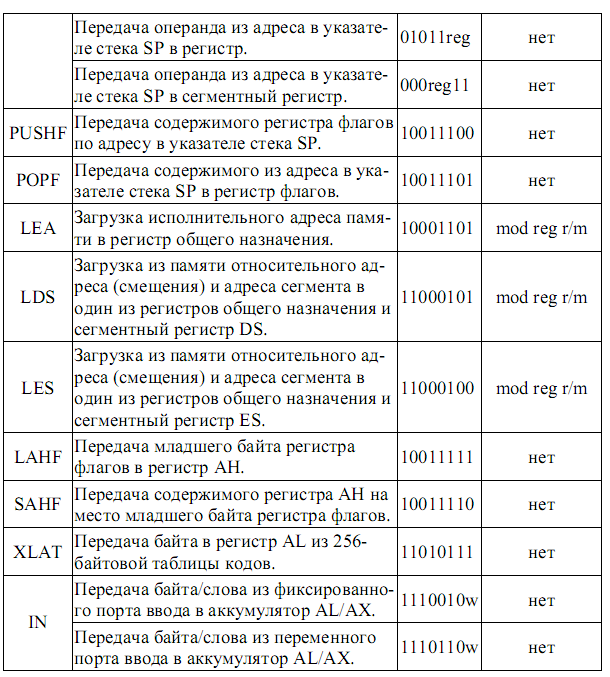
команды этой группы не влияют на установку флагов, за исключением

команд SAHF и POPF.

Пример: mov ax, bx







Арифметические команды

предназначены для выполнения четырех основных видов арифметиче-

ских действий над 8- и 16- разрядными операндами в знаковом и беззна-

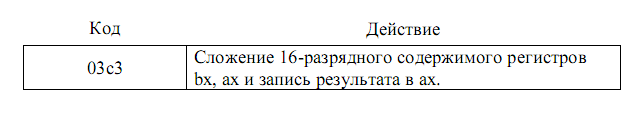
ковом представлении. Кроме основных арифметических команд имеются

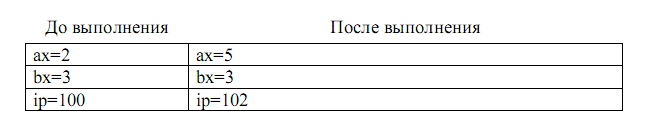
операции коррекции арифметических результатов для их перевода в

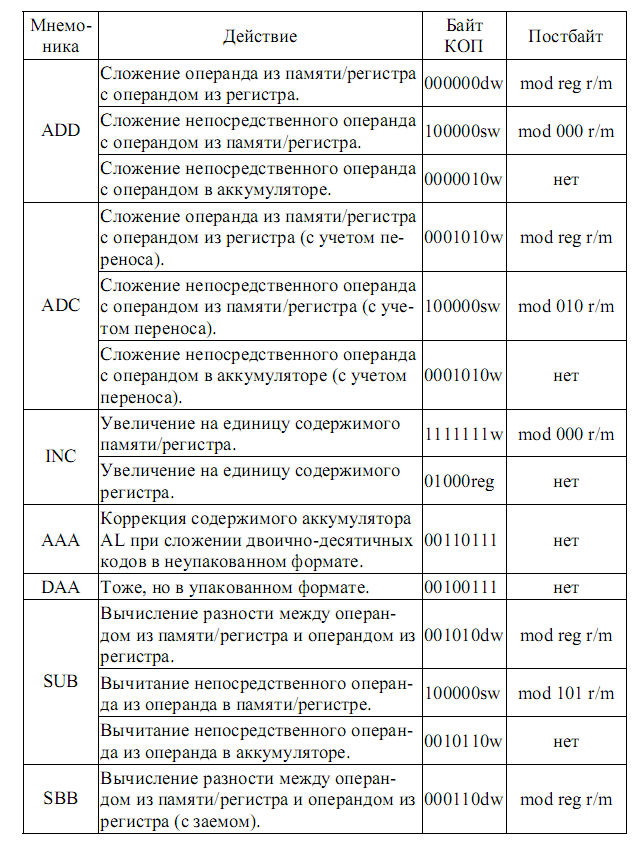
упакованную или неупакованную форму. Признаки полученного резуль-

тата отображаются в 6 битах регистра Flags (CF, AF, SF, ZF, PF и OF).

Пример: add ax, bx









56

Логические команды

предназначены для выполнения четырех логических действий над 8- и 16-

битовыми логическими структурами: получение инверсного кода, логи-

ческое произведение, логическая сумма, сумма по модулю два. Команды

AND, TEST, OR и XOR воздействуют на арифметические флаги сле-

дующим образом: флаги OF и CF всегда сбрасываются в нулевое состоя-

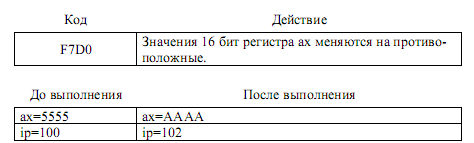
ние; состояния флагов SF, ZF, PF зависят от полученного результата и

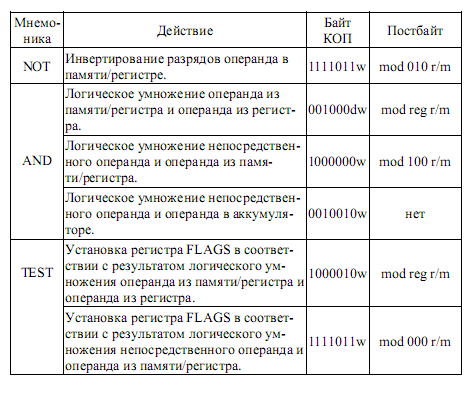
определяются по тем же правилам, что и в командах арифметических опе-

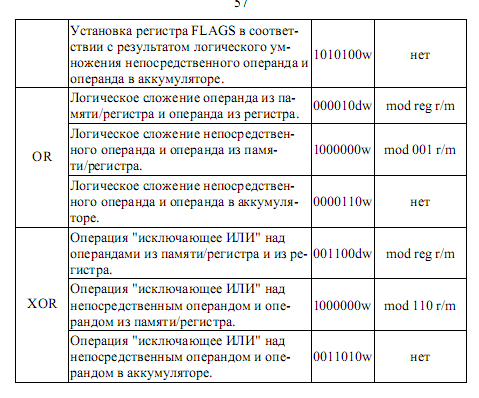
раций; состояние флага AF не определено. Команда NOT не влияет на со-

стояние флагов.

Пример: not ax







Команды сдвига

предназначены для выполнения логических, арифметических и цикли-

ческих сдвигов. Поле операнда имеет формат mem/reg, count. Здесь

mem/reg адресует регистр или ячейку памяти, а count (счет или счетчик)

определяет число сдвигов. Число сдвигов может быть указано как кон-

станта 1 (статический сдвиг) или как регистр CL. В первом случае осуще-

ствляется сдвиг на один байт, а во втором - число сдвигов определяется

содержимым регистра CL, которое должно быть беззнаковым целым дво-

ичным числом. Таким образом, число сдвигов можно задать переменной,

вычисляемой во время выполнения программы (так называемый динами-

ческий сдвиг). При выполнении команд сдвигов флаги изменяются сле-

дующим образом:

• состояние флага AF всегда не определено;

• флаг CF всегда содержит значение последнего выдвинутого бита;

• в однобитных сдвигах флаг OF=0, если операция изменила значение

старшего (знакового) бита операнда; при сдвиге на несколько бит со-

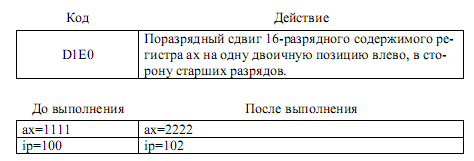
стояние флага OF не определено;

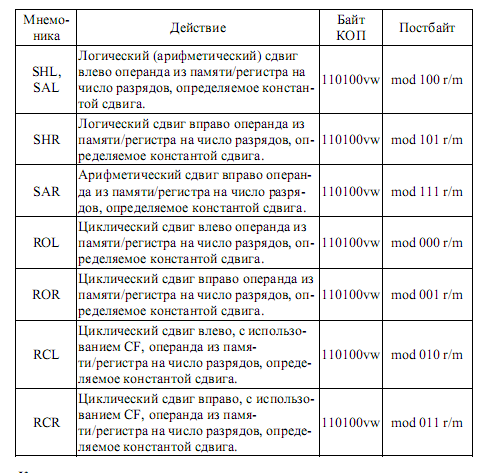
• циклические сдвиги влияют только на флаги OF и CF;

• в арифметических и логических сдвигах флаги SF, ZF и PF изменяются

в соответствии с полученным результатом.

Пример: shl ax, 1





Команды передачи управления

включают в себя безусловные переходы, команды обращения к подпро-

граммам и возврата из них, а также команды управления циклами. Выпол-

нение этих команд приводит к изменению содержимого указателя команд

IP и регистра CS. Команды безусловных переходов и обращений к подпро-

59

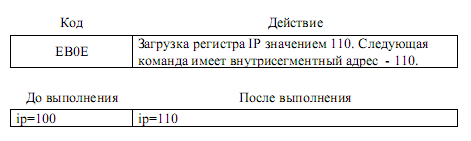
граммам обеспечивают как внутрисегментные, так и межсегментные пере-

ходы с прямой и косвенной адресацией. Команды управления циклами

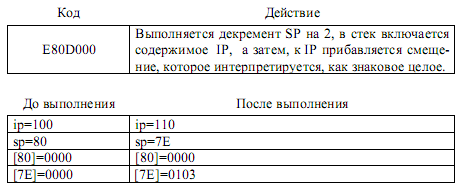
обеспечивают переход только в области текущего сегмента с коротким

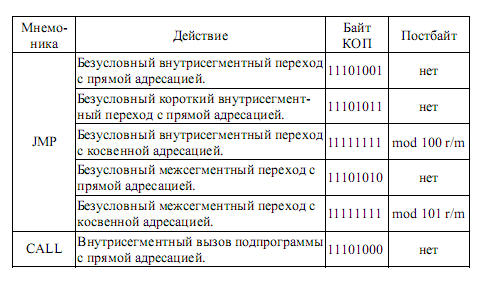
смещением.

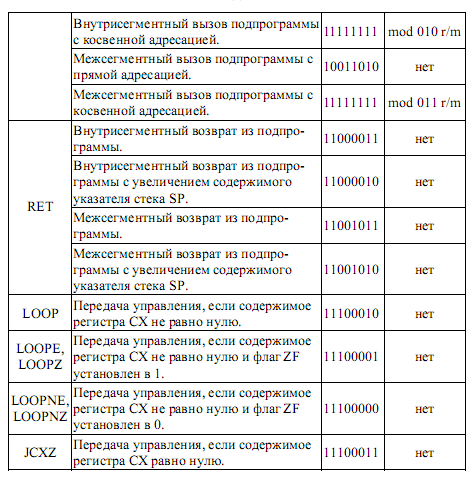
Пример: jmp 110



Пример: call 110

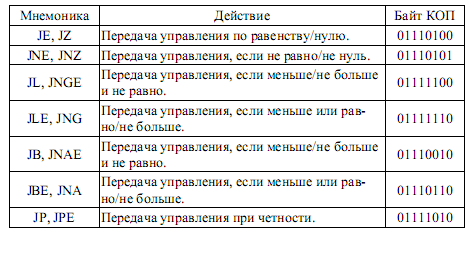






Команды условных переходов

обеспечивают только внутрисегментные переходы.





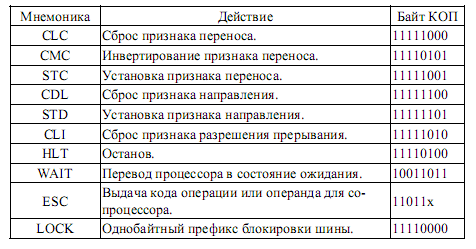
Команды управления процессором

за исключением ESC являются однобайтовыми и могут быть разделены на

две группы: команды, изменяющие содержимое регистра флагов, и коман-

ды, предназначенные для работы с внешними устройствами и не воздейст-

вующие на флаги.



Команды обработки строк

Все команды обработки строк символов имеют длину один байт. Бит 0

показывает операцию с байтом (бит 0=0) или словом (бит 0=1).

Пример: movsw

Код Действие

A5

Выполняется пересылка слова строки источника

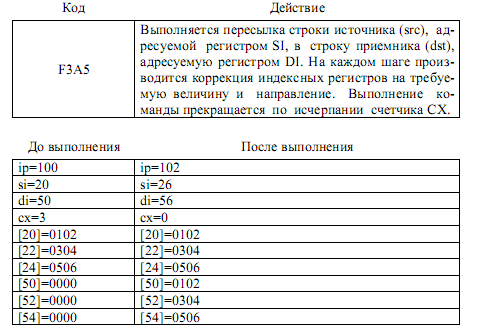
(src), адресуемой регистром SI, в строку приемни-

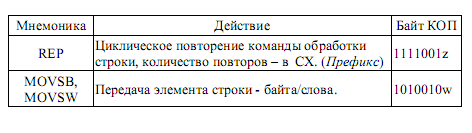
ка (dst), адресуемую регистром DI. Команда не ус-

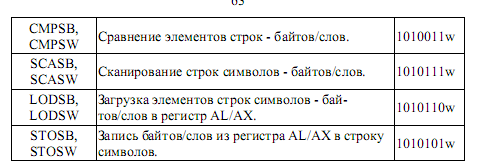
танавливает флагов.



Пример: rep movsw







2.2.7 Математический сопроцессор

Математический сопроцессор (MCP - math coprocessor) – это расшире-

ние основной архитектуры и множества команд основного процессора.

Сопроцессор дополняет возможности процессора новыми командами для

работы с вещественными числами и новыми регистрами и реализован в

виде отдельной ИС для процессоров до 80386 включительно (8087, 80287,

80387), либо непосредственно в микросхеме центрального процессора, как

внутренний модуль (Floating Point Unit, FPU). Сопроцессор поддерживает

семь типов данных: 16-, 32-, 64-битные целые числа; 32-, 64-, 80-битные

вещественные числа и 18-разрядные числа в двоично-десятичном формате.

Форматы чисел с плавающей точкой соответствуют стандартам IEEE 754,

854 и представлены на Рис. 2.9, где Di – разряды десятичного числа в дво-

ично-десятичном представлении; М – мантисса, E – порядок вещественно-

го числа, а S – знаковый разряд.

Декодирование инструкций для сопроцессора и доставка данных осу-

ществляется основным процессором, сопроцессор только исполняет свои

команды. Для хранения операндов и промежуточных данных имеется во-

семь 80-разрядных регистров данных R0-R7, в которых данные представ-

лены в расширенном вещественном формате (см. Рис. 2.9). При загрузке

регистра из памяти, данные автоматически преобразуются в этот формат.

Регистры данных образуют стек, обращение к которому возможно через

относительные имена ST(N). Пять регистров специального назначения

служат для управления вычислениями и определения состояния сопроцес-

сора (Рис. 2.10).

68

2.2.8 Организация ввода-вывода

Работа с портами ввода/вывода в процессорах x86 возможна как с ис-

пользованием специальных команд (IN, OUT), через отдельное адресное

пространство ввода/вывода, так и по схеме с отображением регистров уст-

ройств на обычное адресное пространство. В последнем случае возможно

использование обычных команд из системы команд процессора.

Для адресации портов устройств в пространстве ввода/вывода, исполь-

зуется 16-разрядный адрес, обеспечивая доступ к 64K 8-битным портам с

адресами от 0h до FFFFh. Адреса 0F8h – 0FFh– зарезервированы для сис-

темных целей. Порты с адресами 0h – 0FFh используются оборудованием

системной платы ПК (таймер, контроллер прерываний и т.п.), адреса

0100h–03FFh используют различные контроллеры: дисков, видеомонитора,

компьютерной сети.

Команды IN и OUT работают с прямой адресацией (адрес порта нахо-

дится в команде) для портов с адресами 0h - 0F7h и с косвенной адресаци-

ей (адрес порта находится в регистре DX) для любых портов: 0h - FFFFh.

Обмен данными происходит только через регистр-аккумулятор (AX, AL),

например, чтение из порта: in al,dx , запись в порт: out dx,ax

# 2.2 Генератор тактових імпульсів І8224

Мікросхема І8224 являє собою генератор тактових імпульсів (ГТІ), призначений для спільної роботи з ЦПП І8080A. Генератор формує: високорівневі тактові сигнали Ф1 і Ф2 про незбіжними фазами; тактовий сигнал Ф2Т, за рівнем сумісний із ТТЛ і синхронізований із сигналом Ф2; сигнал STSTB “Строб стану”, що, надходячи на системний контролер, фіксує стан шини даних мікропроцесора; сигнал RESET “Установка”.

Генератор опорної частоти при підключенні до висновків XTAL1 і XTAL2 кварцові резонатори забезпечує високу стабільність частоти, обумовлену основною частотою порушення кварцового резонатора.

Вихід генератора опорної частоти виведений на зовнішній висновок OSC і з'єднаний усередині мікросхеми з лічильником-дільником, що входить у склад тактового генератора. Тактовий генератор складається з лічильника-дільника на 9, логічних дешифраторів, що формують необхідні тактові імпульси, вихідних формувачів і допоміжних логічних схем і тригерів для генерації вихідних сигналів: Ф1.Ф2.Ф2Т, STSTB, Тактові імпульси Ф1 і Ф2 керують Моп-входами мікропроцесора І8080А. Тактовий імпульс Ф2Т використовується для керування Ттл-входами в режимі прямого звертання до пам'яті.

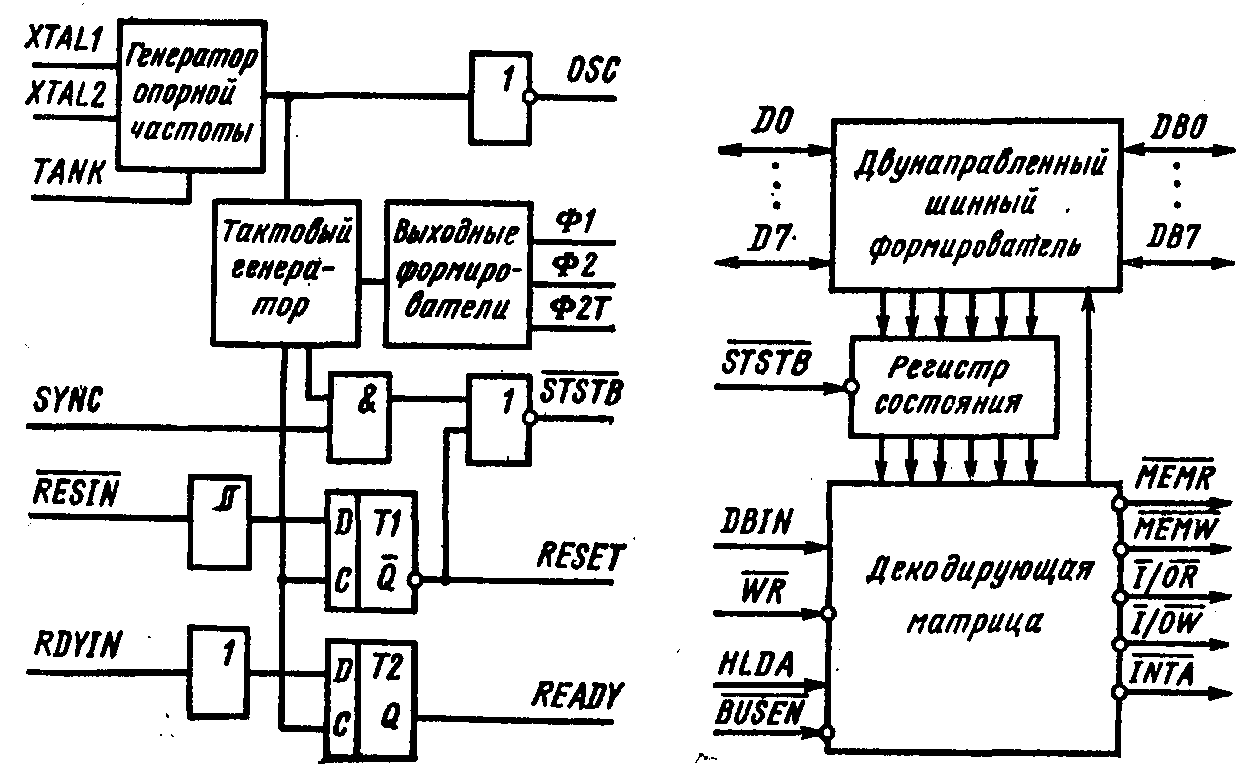
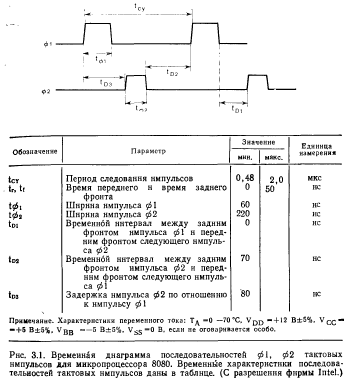


Рисунок 2.4 Структурна схема І8224.

Таблиця 2.1- Призначення виводів І8224.

|  |  |  |
| --- | --- | --- |
| **Номер виводу** | **Позначення** | **Призначення** |
| 1 | RESET | Установка (вихід) |
| 2 | RESIN | Установка (вхід) |
| 3 | RDYIN | Готовність (вхід) |
| 4 | READY | Готовність (вихід) |
| 5 | SYNC | Синхронізація |
| 6 | Ф2Т | Фаза 2 з рівнем ТТЛ |
| 7 | STSTB | Строб стану |
| 8 | GND | Загальний |
| 9 | Ucci | +12 В |
| 10, 11 | Ф2,Ф1 | Фаза 2, 1 |
| 12 | OSC | Вихід осцилятора |
| 13 | TANK | Вхід коливального контуру |
| 14, 15 | XTAL2, XTAL1 | Кварцовий резонатор |
| 16 | Ucc | +5В |



Негативний сигнал STSTB, тривалість якого дорівнює од--ному періодові частоти опорного генератора, формується мікросхемою І8224 при надходженні на її вхід з мікропроцесора І8080А сигналу SYNC “Синхронізація”, що свідчить про початок машинного циклу.При надходженні вхідного сигналу RESIN мікросхема І8224 за допомогою тригера Шмітта і тригера Т1 виробляє сигнал RESET, синхронізований з тактовим сигналом Ф2, По сигналі RESET здійснюється установка у вихідний стан різних пристроїв мікропроцесорної системи. Наявність у мікросхемі тригера Шмітта дозволяє подавати на вхід RESIN сигнал з положистим фронтом. За допомогою тригера Т2 здійснюється стробування вхідного сигналу RDYIN “Готовність” тактовим сигналом Ф2.

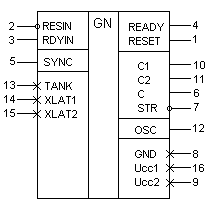
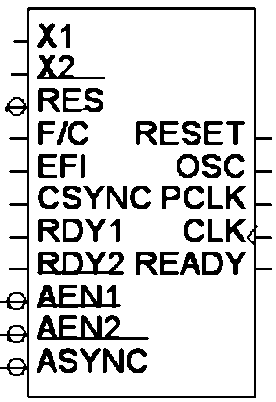


Рисунок 2.5-інтегральне виконання ІМС І8224

**2.1 Генератор синхронизации 8284**

Специализированный кристалл – генератор синхронизации 8284 разработан для синхронизации работы микропроцессора. Графическое представление этого кристалла приведено на рис.3.1.



**Рис. 3.1. Внешний генератор 8284 для систем на основе микропросессора 8086**

Выводы **X1** и **X2** обеспечивают подключение параллельного **резонансного контура** для внутреннего генератора (**F/C** имеет низкий уровень). **EFI** может соединяться с внешним генератором (**F/C** имеет высокий уровень). Частота генератора внутренне делится на три, чтобы

получить **синхросигнал** **(CLK)**. Вывод **RESET** указывает, что централь­ный процессор устанавливается в исходное состояние, и может исполь­зо­вать­ся как сигнал сброса системы. Его активный уровень – высокий, и сигнал синхронизован с синхросигналом процессора **CLK** и продолжа­ет­ся целое число тактов, соответствующих длине сигнала **RES**. Этот сигнал может быть асинхронным. **RES** внутренне синхронизирован.

Асинхронные входные сигналы готовности (**RDY1**, **RDY2**) сооб­щают процессору, что адресуемая память или устройство ввода/вывода закончили передачу или прием данных. Вход **READY** изменяет свое состояние синхронно с **CLK** и принимает активное высокое значение в соответствии с сигналом **RDY2** или **RDY1**. Сигналы **RDY1** и **RDY2** используются, если сигналы **AEN1** и **AEN2** соответственно имеют низкий уровень. Соединение **RDY** с уровнем логической 1 (**AEN** должен иметь низкий уровень) будет всегда утверждать состояние готовности для центрального процессора. Если одна из линий **RDY** не использована, она должна быть подключена к низкому логическому уровню, чтобы управление осуществлялось через другой вход **RDY**.

# 2.3 Буферні регістри

Специальные внешние регистры-защелки должны использоваться для разделения информации на шине адреса/данных. Для этих целей выпускаются специальные восьмибитовые регистры, например, 82C82. Микропроцессорный сигнал **ALE** обеспечивает строб для фиксации физической адресной информации. Адрес представлен на мультиплекс­ной шине адреса/данных в течение состояния T1. Срез импульса **ALE** происходит в середине T1 и обеспечивает правильную фиксацию адреса.

В большинстве случаев требуется буферизации шины данных системы. Следующие случаи требуют дополнительных внешних приемопередатчиков на шине данных:

• емкостная нагрузка на шине адреса/данных становится слишком большой;

• текущая нагрузка на шине адреса/данных превышает предельные технические параметры устройства;

• память или устройства ввода/вывода не могут обеспечить перевод своих выводов в высокоомное состояние, чтобы предотвратить конфликтную ситуацию при обращении к шине.

Микропроцессор генерирует два сигнала управления **DEN** и **DT/R** для управления двунаправленными буферами или приемопередат­чика­ми. Линии связи между процессором и приемопередатчиками носят название **локальная шина**. Линии связи между приемопередатчиками и памятью или устройствами ввода/вывода называются **буферизирован­ной шиной**. Полностью буферизированная система не имеет никаких устройств, подключенных к локальной шине. Частично буферизирован­ная система имеет устройства как на локальной, так и на буферизирован­ной шине. В полностью буферизированной системе, **DEN** непосредственно управляет выводом приемопередатчика. Частично буферизированная система требует, чтобы **DEN** был использован совместно с другим сигналом, чтобы предотвратить включение приемопередатчика при доступе к локальной шине. **DT/R** всегда соединяется непосредственно с приемопередатчиком. Однако может потребоваться инвертор, если полярность **DT/R** не соответствует полярности, требуемой приемопередатчику. **DT/R** переходит в состояние низкого уровня (0) только для чтения памяти и устройств ввода/вывода, выбора команды и циклов подтверждения прерывания.

Мікросхема І8282/83 (КР580ІР82/83) являє собою 8-розрядний буферний регістр, призначений для фіксації інформації і може використовуватися в

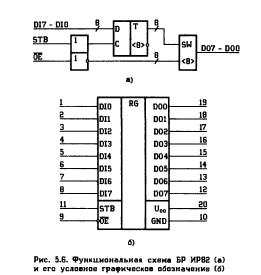


Рисунок 2.6- Структурна схема ІМС 18282

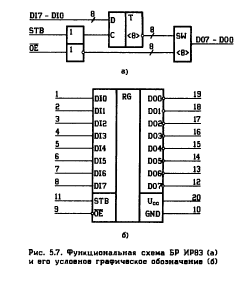


Рисунок 2.7- Структурна схема ІМС І8283

системах , побудованих на мікросхемах різних серій. В системах з мікропроцесором І8080А регістр може використовуватись для зберігання слова стану мікропроцесора, а в системах з І8086 – для виділення адреси, яка поступає по мультиплексованій шині адреси даних

Структурна схема мікросхеми І8282 (КР580ІР82) представлена на рис.2.6.

Якщо на вході /ОЕ надходить сигнал дозволу низького рівня , а на вході STB - сигнал високого рівня , то інформація зі входів мікросхеми передається на виходи. Після переходу сигналу на вході STB з високого рівня на низький інформація, записана до буферного регістру, зберігається до появи сигналу дозволу на вході STB . Сигнал високого рівня на вході /ОЕ переводе виводи DO0-DO7 у високоімпедансний стан.

Електричні параметри регістру:

1. Uжив. (напруга живлення) - 5 В

2. Вихідна напруга живлення низького рівня ( Uвих низ р ): < 0.45 В

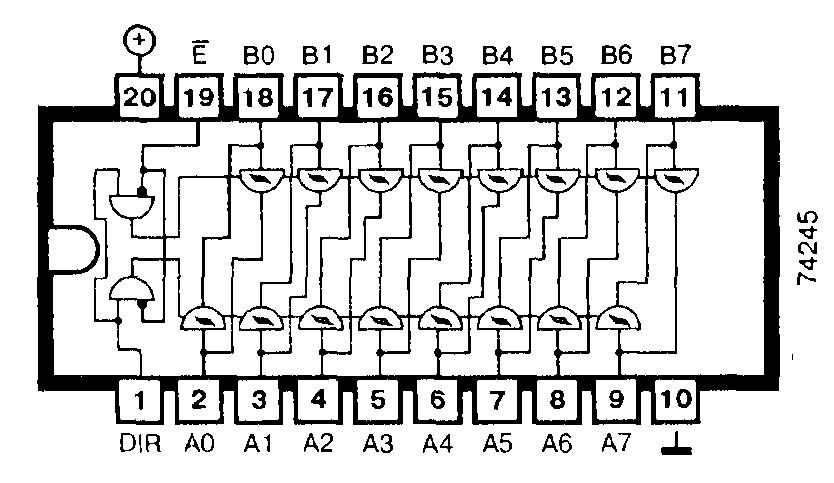
3. Вихідна напруга живлення високого рівня ( Uвих вис р ): > 2.4 В

4. t-затр (Час затримки поширення інформаційного сигналу на виході щодо інформаційного сигналу на вході < 30 нс

Таблиця 2..2- Призначення виводів І8282/83

|  |  |  |  |
| --- | --- | --- | --- |
| Номер виводу | Позначення | Тип виводу | Функціональне  значення |
| 1-8 | DI0—DI7 | Входи регістра | Сигнали, які подаються на виводи 1-8, надходять на входи D тригерів буферного регістру |
| 9 | /ОЕ | Вхід | Дозвіл видачі змусту буферного регісту. Сигнал низького рівня на вході 9 дозволяє видачу на виводи DO0-DО7 змісту буферного регістру, а сигнал високого рівня переводе ці виводи у високоімпенансний стан |
| 10 | GND |  | Загальний |
| 11 | STB | Вхід | Строб. По сигналу високого рівня , який поступає на вивід 11 , інформація , яка подається на входи DI0—DI7, записується в тригери буферу |
| 19—12 | DO0-DО7  (І8282)  / DO0-/ DО7  (І8283) | Виходи з висо-коімпедансним станом | На 19-12 виводиться стан тригерів буферного регісту, якщо на вхід /ОЕ поданий сигнал дозволу, в противному випадку виводи 19-12 знаходяться у високоімпедансному стані |
| 20 | Ucc | +5 В | Живлення |

**Микросхема 74245** применяется как двунаправленный буфер для шин данных и драйвер для 8-разрядных шин данных и адресных шин. Производится следующая номенклатура микросхем: 74ALS245, 74AS245, 74F245, 74L245, 74LS245.



Микросхема **74245** содержит восемь неинвертирующих двунаправленных буферов шины с выходами, имеющими три состояния

Восемь буферов шины микросхемы 74245 позволяют осуществить асинхронную двунаправленную связь между двумя 8-разрядными шинами.

С помощью входа DIR (направление передачи) микросхемы 74245 можно установить направление передачи данных от шины А к шине В, для чего на вход DIR подается напряжение высокого уровня, или от шины В к шине А — на вход DIR поступает напряжение низкого уровня. При этом для передачи информации на управляющем входе E (разрешение) должно быть напряжение низкого уровня. Если же на входе E напряжение высокого уровня (на вход DIR может подаваться напряжение любого уровня), то все выходы переходят в третье (высокоомное) состояние и обе шины изолированы друг от друга.

Волновое сопротивление подключенных к микросхеме 74245 шин должно быть не менее 133 Ом. На всех входах есть формирователи цифровых сигналов на триггерах Шмитта, передаточная характеристика которых имеет гистерсзис около 0,4 В, поэтому микросхема очень хорошо подходит для приема сигналов на зашумленных каналах (только для микросхем серии 74LS245).

При напряжении высокого уровня на выходах микросхемы 74245 величина выходного тока может быть 15 мА, а при напряжении низкого уровня ток достигает 24 мА (для серии 74LS245).

Технические данные

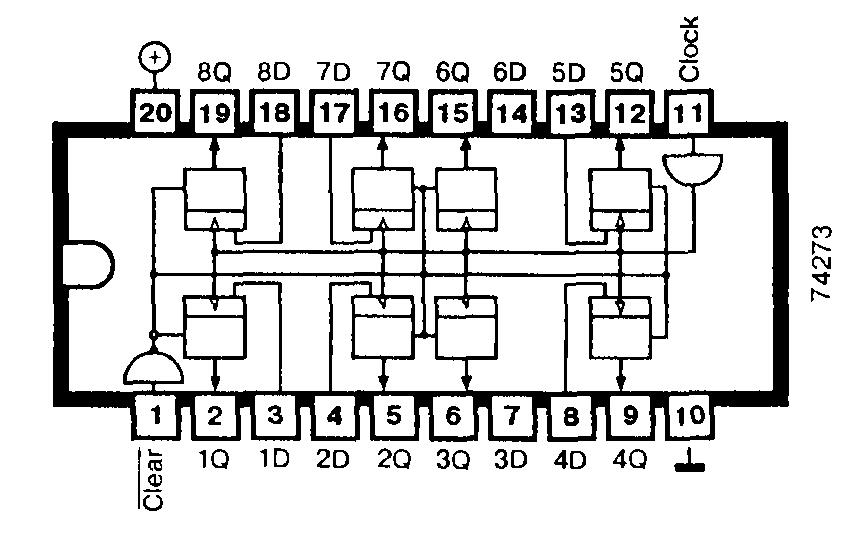
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Тип микросхемы** | **74ALS245** | **74AS245** | **74F245** | **74LS245** |
| Время задержки прохождения сигнала, нс | 6,6 | 5,5 | 3,8 | 8 |
| Ток потребления, мА | 35 | 59 | 100 | 62 |

Состояние микросхемы

|  |  |  |
| --- | --- | --- |
|  | | |
| **Управляющие входы** | | **Режим работы** |
| **Е** | **DIR** |
| 0 | 0 | Передача данных шины В к А |
| 0 | 1 | Передача данных шины А к В |
| 1 | X | Шины А и В изолированы друг от друга |

**Регистры 74273**

Буферные регистры, регистры сдвига, генераторы двоичного кода. Производится следующая номенклатура микросхем: 74273, 74ALS273, 74AS273, 74F273, 74LS273, 74S273.



Микросхема **74273** служит для одновременного хранения восьми бит информации.

В основном режиме работы на вывод 1 (Clear) микросхемы 74273 подаётся напряжение высокого уровня. Данные поступают в регистр через входы D. При перепаде тактового импульса на входе Clock с низкого уровня на высокий (положительный фронт импульса) информация записывается в регистр и появляется на соответствующих выходах Q.

Если на вывод 1 (Clear) микросхемы 74273 подается кратковременный сигнал низкого уровня, то и на всех выходах устанавливается напряжение низкого уровня.

Технические данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| **Тип микросхемы** | **74273** | **74ALS273** | **74F273** | **74LS273** | **74S273** |
| Максимальная тактовая частота, МГц | 30 | 35 | 145 | 30 | 95 |
| Время задержки прохождения сигнала, нс | 17,5 | 16 | 7,5 | 17,5 | 10 |
| Ток потребления, мА | 62 | 15 | 66 | 17 | 109 |

Состояние микросхемы

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| **Входы** | | | **Выход Q** |
| **Clear** | **Clock** | **D** |
| 0 | X | X | 0 |
| 1 | фронт тактового импульса | 1 | 1 |
| 1 | фронт тактового импульса | 0 | 0 |
| 1 | 0 | X | Нет изменений |
| 1 | 1 | X | Нет изменений |
| 1 | спад тактового импульса | X | Нет изменений |

# 2.4 Двонаправленний шинний формувач І8286/87

Шинний формувач І8286/87 призначений в першу чергу для використання в мікропроцесорних системах , які побудовані на ВІС серії І8080А та І8086, але він може використовуватися і в інших системах сумісно з мікропроцесорами інших серій.

Шинний формувач призначений для збільшення навантажувальної здібності виводів мікропроцесорних ВІС. Крім цього, він забезпечує відключення модулів системи в потрібний час від загальної шини.

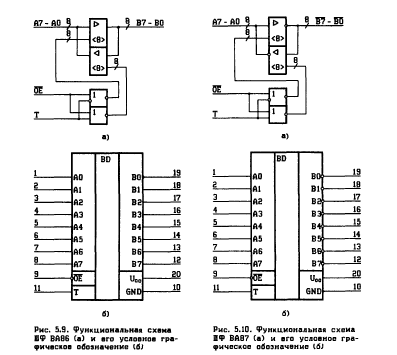


Рисунок 2.8 Шинний формувач

Восьмирозрядний шинний формувач в залежності від сигналів на входах Т і /ОЕ (коли Т- високий і /ОЕ-низький) забезпечується підключнггя входів А0-А7 до виходів В0-В7, в іншому випадку (якщо на Т-низький)-входи В0-В7 підключаються до виходів А0-А7. Сигнал високого рівня на вході /ОЕ переводить виводи В0-В7 (якщо на Т-високий) або виводи А0-А7 (якщо на Т-низький) у високоімпедансний стан.

Таблиця 2.3- Призначення виводів І8286/87

|  |  |  |  |
| --- | --- | --- | --- |
| Номер виводу | Позначення | Тип виводу | Функціональне  значення |
| 1-8 | А0—А7 | Входи визоди з високо-імпедансним станом | На виводи 1-8 можуть надходити сигнали,або зніматися в залежності від сигналу на вході Т |
| 9 | /ОЕ | Вхід | Дозвіл виходу. Коли на вивід 9 надходить сигнал високого рівня в залежності від сигналу на вході Т виводи А0-А7 або В0-В7 знаходяться у високоімпенансному стані |
| 10 | GND |  | Загальний |
| 11 | T | Вхід | Напрямок передачі інформації. Якщо на вхід Т подається сигнал високого рівня, то формувач налаштовується на передачу від входів А0-А7 до виводів В0-В7, в іншому випадку, якщо на вхід Т подається сигнал низького рівня , то інформація передається від виводів В0-В7 до виводів А0-А7 |
| 19—12 | В0-В7  (І8286)  /В0-/В7  (І8287) | Виходи з висо-коімпедансним станом | На виводи 19-12 можуть вводитись або виводитись в залежності від сигналу на вході Т стан ( в І8287 сигнали інвертуються) |
| 20 | Ucc | +5 В | Живлення |

# 2.5 Системний контролер І8228/38 (КР580ВК38)

Мікросхема І8228/38 виконує функцію системного контролера і шинного формувача, здійснює формування керуючих сигналів звертання до ОЗП або до пристроїв введення/виводу (ПВВ) і забезпечує прийом і передачу 8-розрядної інформації між шиною даних мікропроцесора і системною шиною.

Формування сигналів I/OW, MEMW у даній мікросхемі відбувається щодо сигналу STSTB “Строб стану”, що дозволяє при застосуванні в мікропроцесорній системі мікросхеми І8238 використовувати ЗП і ПВВ із більш широким діапазоном швидкодії. . Двонаправлений шиний формувач здійснює буферирування 8-розрядної шини даних і автоматичний контроль напрямку передачі даних.

Підключення системного контролера до шини даних мікропроцесора здійснюється за допомогою двонаправлених виводів DO-D7, до системного з допомогою двонаправлених висновків DBO-DB7. При необхідності за допомогою сигналу BUSEN “Керування системною шиною” виводи DBO—DB7 системного контролера можуть бути переведені в стан “Виключене”.

Таблиця 2.4 Призначення висновків ІМС І8228/38 (КР580ВК38)

|  |  |  |
| --- | --- | --- |
| Номер виводу | Позначення | Призначення |
| 6, 8, 10, 12, 15,17,19, 21 | DO—D7 | Шина даних |
| 5, 7, 9, 11, 13, 16, 18, 20 | DBO—DB7 | Системна шина |
| 1 | /STSTB | Строб стану |
| 2 | HLDA | Підтвердження захоплення |
| 3 | WR | Запис |
| 4 | DBIN | Прийом |
| 14 | GND | Загальний |
| 22 | BUSEN | Керування системною шиною |
| 23 | INTA | Підтвердження переривання |
| 24 | /MEMR | Читання пам'яті |
| 25 | /I/OR | Читання УВВ |
| 26 | /MEMW | Запис у пам'ять |
| 27 | /I/OW | Запис в УВВ |
| 28 | Ucc | +5 В |

Регістр стану виконаний на шести D-тригерах і призначений для збереження інформації про стан мікропроцесора, що надходить по шині даних DO-D7. Запис у регістр стану здійснюється по сигналі STSTB, що надходить на початку кожного машинного циклу. Декодувальна матриця в залежності від режиму роботи мікропроцесора, зафіксованого в регістрі стану, і вхідних керуючих сигналів HLDA, WR, DBIN формує сигнал INTA “Підтвердження переривання” або сигнали читання/запису при звертанні до ОЗП або ПВВ.

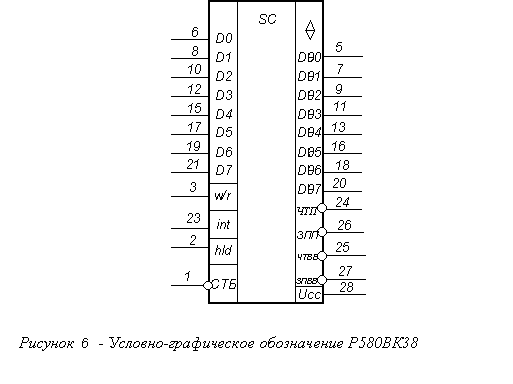


Рисунок 2.8- Інтегральне виконання ІМС КР580ВК38.

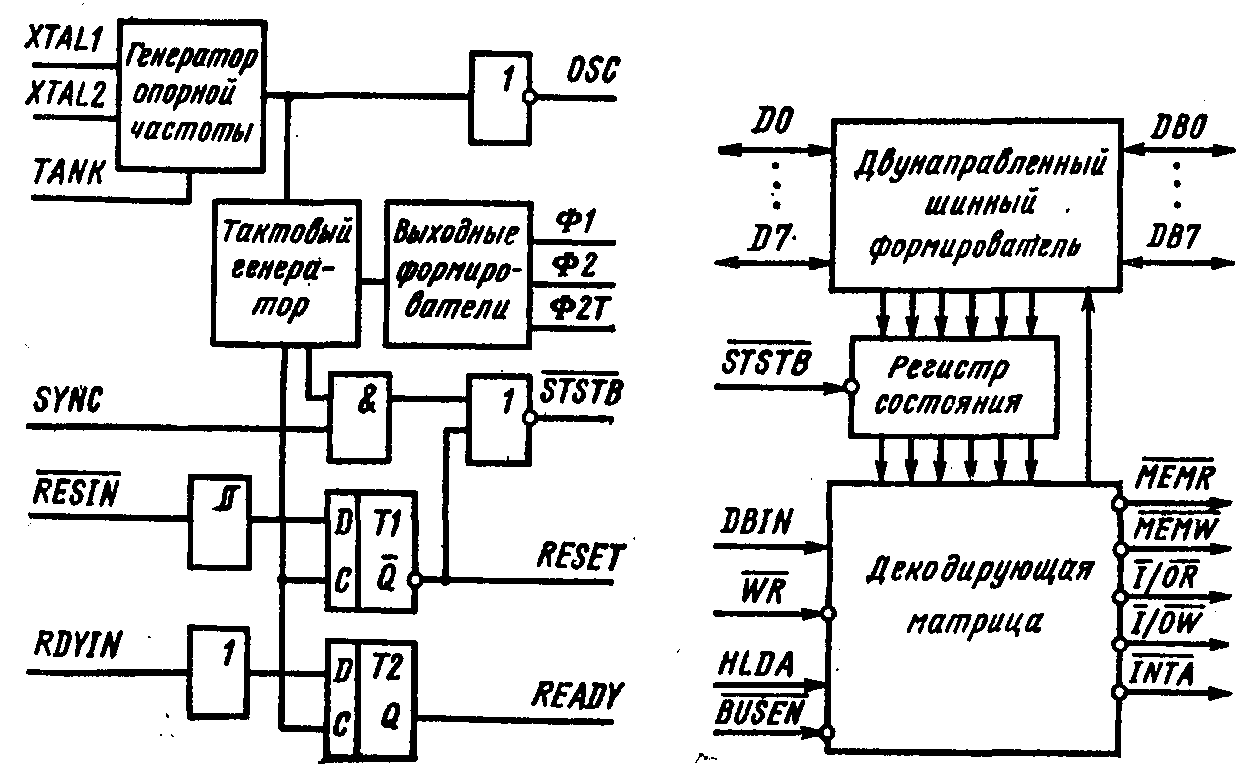
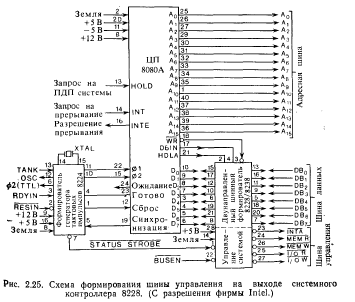


Рисунок 2.9- Структурна схема ІМС І8228/38.



# Шинная архитектура

# 3.1 Главные шины

Архитектура систем с 3 шинами.

Архитектура с 3 шинами является наиболее общей для микропроцессорных систем. Шинной системой называют физическую группу линий передачи сигналов, имеющих схожие функции в рамках системы. Например, некоторая группа линий может использоваться для передачи сигналов адреса памяти. Эту группу линий можно назвать адресной шиной. Все три шины являются специализированными с точки зрения их функций. Эти шины именуются так:

 1. Адресная шина системы

 2. Шина данных системы.

 3. Шина управления системы.

Технически проще использовать однонаправленные шины, но тогда их число должно увеличиться, то есть по две шины для операции "Чтение" (Введение) и "Запись" (Вывод). Это приводит к существенному увеличению числа контактов разъема модуля МП или непосредственно самой БИС МП, а также числа проводников ОМ. Между тем любое увеличение числа проводников ОМ всегда приводит к увеличению стоимости ЭВМ, а в ряде случаев вообще невозможно в силу технических ограничений.

# 3.2 Типы шинной архитектуры

Самым очевидным способом сократить число выводов БИС и проводников ОМ является объединение однонаправленных шин в одну двунаправленную, управляемую соответствующими сигналами - запись / чтение (READ / WRITE) для модулей памяти и ввода / вывода (INPUT / OUTPUT) для модуля ППУ. Ниже рассматриваются 5 вариантов структур ОМ только с двунаправленными шинами.

Раздельные шины (Рис.7.2, а)

Использование отдельных двунаправленных шин упрощает обмен процессора с модулями памяти и ППУ и ​​дает принципиальную возможность вести его в перекрываются интервалы времени. При этом адресные пространства ячеек памяти и регистров ППУ могут перекрываться. Основным недостатком такой структуры является большое количество проводников общей магистрали и контактов модуля МП.

Изолированные шины (Рис.7.2, б)

Сходство процессов обмена процессор - память и процессор - регистры ППУ позволяет использовать в обоих случаях одни и те же проводники ША и ШД. Это приводит к структуре с изолированными шинами. Адресные пространства элементов памяти и регистров ППУ, как и при использовании предыдущей структуры, могут перекрываться, то есть они изолированы. Для того чтобы занять шины для обмена с памятью, процессор выдает сигналы READ / WRITE, а для обмена с ПУ - INPUT / OUTPUT.

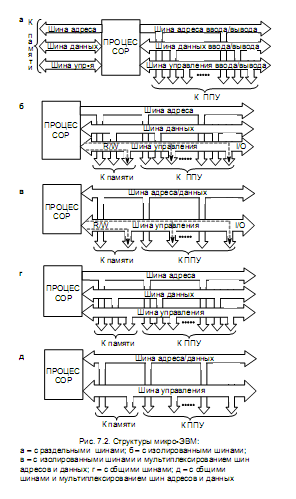
По сравнению с предыдущей структурой число проводников ОМ (по модуля МП) уменьшилась, но исчезла принципиальная возможность вести параллельный обмен с памятью и ПУ.

Изолированные шины и мультиплексирования ША и ШД (рис. 7.2, в)

 В этом случае ША и ШД совмещены. В результате передача адресов и данных идет в разные моменты времени. Адресные пространства элементов памяти и регистров ППУ изолированные.

По сравнению с предыдущими структурами уменьшилось число проводников общей магистрали и выводов модуля МП, но адреса и данные могут передаваться только в неперекрывающиеся моменты времени. Это затрудняет возможность конвейеризации процесса выполнения команд и удлиняет цикл обмена процессор - память.

Общие шины (рис. 7.2, г)

 В данном случае команды ввода / вывода (INPUT / OUTPUT) вообще исключены, что упрощает структуру модуля МП и общей магистрали, хотя количество проводников примерно соответствует структуре с изолированными шинами. Ячейки памяти и регистры ППУ лежат в общем адресном пространстве, и для обращения к ним используются одни и те же команды**.**

В ряде случаев это является преимуществом, однако при возникновении определенных сбоев в работе ПУ и их некорректной обработки со стороны операционной системы возможны «зависания» вычислительного процесса.

Общие шины и мультиплексирования ША и ШД (рис. 7.2, д)

Недостатки и преимущества данной структуры по сравнению с предыдущей ( "общие шины") аналогичные изложенным выше для структуры, показанной на рис. 7.2, ст.

Современные МП, практически все имеют команды ввода / вывода, то есть дают возможность организовать структуру с изолированными шинами. При этом все они допускают обращение к регистрам ППУ как к ячейкам памяти, то есть позволяют реализовать структуру с общими шинами.

Следует отметить, что структура магистрали типа "общие шины" является

широкое распространение в реальных устройствах. Понятие "общая магистраль (ОМ)" и "общая шина (ОШ)" в литературе часто используются как синонимы, хотя согласно приведенной выше классификации ОШ является частным случаем структуры ОМ. Ниже, при изложении материала, понятие ОШ и ОМ также будут использоваться как синонимы, за исключением особо оговоренных случаев.

Рассмотрены структуры ОМ во многом определяют внутреннюю структуру конкретной микроЭВМ. Однако структура микроЭВМ определяется также и множеством вопросов, связанных с формой представления информации и способами ее передачи внутри микрокомпьютер, алгоритмами взаимодействия отдельных модулей.

3.3 Функциональная схема ЭВМ на базе процессора И8080А

Логическое состояние этих трех шин описывает коммуникационный тракт системы в любой момент времени. Коммуникационной тракт это путь, который данные, представленные в виде электрических сигналов, проходящих в системе от одной точки к другой.

### Адресная шина системы

По адресной шине системы передаются только выходные сигналы, поступающие из выводов в корпусе процессора. Эта шина предназначена для того, чтобы открывать или выбирать правильный тракт для электрического соединения в пределах микропроцессорной системы.

Для удобства будем в дальнейшем считать, что все электрические соединения в микропроцессорной системе осуществляется между микропроцессором и устройством, открытым с помощью адресной шины. В качестве устройства здесь выступает любая электрическая Cxeмa, принимающий данные от микpoпроцесора или производит данные для него. После того как понятие описываемого здесь коммуникационного тракта дано, легче объяснить особенности других коммуникационных трактов, имеющихся в микропроцессорная системе.

Другой важной характеристикой адресной шины системы является ее емкость. Емкость шины определяется числом входящих в нее отдельных электрических линий. Для процессоров 8080, 8085. Z80, 6800 характерна 16 разрядные адресная шина. Это означает, что адресная шина систем, построенных на базе этих процессоров, компонуется с 16 физических линий /

### Шина данных системы

Шина данных системы является двунаправленной шиной. Это означает, что передача данных может осуществляться в обоих направлениях. В некоторых случаях данные генерируются микропроцессором и передаются от него к определенному устройству системы. Это устройство открывается с помощью заданного логического состояния линий адресной шины и получает данные с шины данных.

В других случаях данные генерируются каким источником и передаются процессору посредством шины данных. В качестве источника выступает то устройство системы, которое открывается с помощью адресной шины. Подобный режим называется вводом данных в микропроцессор.

Хотя передача данных по шине данных может осуществляться в обоих направлениях, однако в каждый заданный момент времени она осуществляется только в одном направлении. Это означает, что для передачи данных в систему и их приема из системы микропроцессор переводится в соответствующий режим. Более того, во всех разрядах шины в каждый момент времени данные передаются только в одном направлении, то есть в любой момент по всем линиям шины они могут или только вводиться или только выводиться.

Для процессоров 8080, 8085. Z80 и 6800 шина данных является 8-разрядной Поэтому говорят, что емкость шины данных равна 8 разрядов и параллельно могут передаваться только 8 бит информации. По этой причине перечисленные процессоры относятся к классу 8-разрядных микропроцессоров.

### Шина управления системы

На шине управления действует 4 следующих типа сигналов:

1. Чтение из памяти активизирована.

2. Запись в память активизирована.

3. Чтение устройства ввода активизирована.

4. Запись на устройство ввода активизирована

Позже для этой шины мы введем некоторые дополнительные сигналы. Однако для понимания сути процессов пока необходимо ограничиться указанным списком сигналов. После того как станут ясны функции этих четырех сигналов, будет легче изучать функции других сигналов.

Шина управления используется только для вывода сигналов, то есть однонаправленной и работает в режиме вывода, напротив, шину данных мы рассматриваем как двустороннюю.

Термин «активизирован» означает, что при наступлении события, запрашиваемой соответствующей линией шины управления, эта линия имеет активный сигнал логического уровне I или 0. В микропроцессорных системах активное состояние линий шины управления может быть либо логической 1. или логическим 0. При этом разные линии системы могут быть активными при различных уровнях логического сигнала. Например, линия управления ЧТЕНИЕ ИЗ ПАМЯТИ может быть активной при логическом уровне 1, а линия управления ЗАПИСЬ В ПАМЯТЬ - при уровне 0.

Распознавание и инициирования типа электрического соединения для шины данных системы является функцией сигнала шины управления. Необходимо заметить, что линия управления может быть активизированной и при уровне логической 1, и при уровне логического 0.

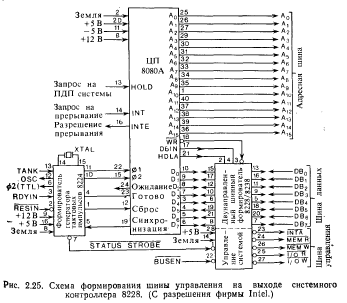


Рисунок 3.1

На рис. 3.1 архитектура систем с тремя шинами показана в виде блок-схемы, на которой видно, что стрелки, отвечаю адресной шины и шине управления, указывают только на одно направление. Это говорит о том, что эти шины однонаправленные. Для шины данных на рис. 3.1 стрелки указывают на два направления, соответствует двусторонней шине. Принятые нами обозначения часто используются в литературе для описания этих шин.

Использование архитектуры с 3 шинами

Теперь мы обсудим общие принципы передачи информации в микропроцессорной системе, имеет архитектуру с 3 шинами. Прежде всего необходимо объяснит »основные функции, реализованные микропроцессорной системой. После этого можно будет перейти к рассмотрению особенностей их выполнения.

Для начального знакомства с процессорами достаточно рассмотреть только пять функций, описанных ниже, позже список этих функций может быть расширен. Такие функции хорошо отражают возможные операции, выполняемые в микропроцессорной системе. К ним относятся:

 1. Запись данных в память системы.

 2. Чтение данных из памяти системы.

 3. Запись данных на устройство ввода-вывода.

 4. Чтение данных с устройства ввода-вывода.

 5. Выполнение операций с содержанием внутренних регистров процессора.

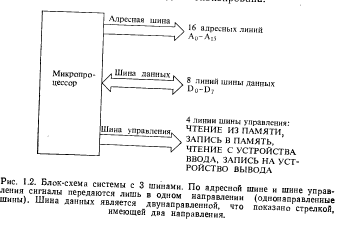
 Указанные пять возможных типов функций микропроцессорной системы позволяют создавать большое количество различных средств.

Рисунок 3.2

# 3.4 Функциональная схема ЭВМ на базе процессора И8086А

Рассмотрим систему, в которой микропроцессор работает в минимальном режиме. В этом режиме он генерирует все необходимые сигналы управления. Блок-схема системы на основе микропроцессора 8086 показана на рис.3.2. Система состоит из микропроцессора, генератора синхронизации, 3-х регистров-защелок, 2-х приемопередатчиков, устройства памяти и устройств ввода/вывода.

В системе, представленной на блок-схеме, микропроцессор выпол­няет все функции по обработке информации и формированию всех необходимых сигналов управления. Внешний генератор обеспечивает формирование синхронизирующего сигнала **CLK** и сигналов начального сброса **RESET** и готовности **READY,** связанных с ним. **READY** форми­руется при включении питания системы, потому что конденсатор заряжается через резистор некоторое время. В это время входной сигнал **RES** имеет низкое значение, и **RESET** активен. По сигналу **RESET** все составляющие системы устанавливаются в начальное состояние. Адрес­ные выводы микропроцессора связаны с входами регистров-фиксаторов. Поскольку число адресных сигналов 21 (**AD0** … **AD15**, **AD16** … **AD19**, **BHE**), требуется 3 регистра. Адрес сохраняется в регистре по сигналу **ALE** от микропроцессора. Тогда в течение всего цикла шины адрес находится в регистрах и доступен для составляющих системы. Выводы регистров-фиксаторов формируют буферизированную адресную шину.

Кроме того, линии шины адреса/данных (**AD0**…**AD15**) микропро­цессора связаны со входами буферных приемопередатчиков с высокой нагрузочной способностью. Приемопередатчики 8 разрядные, поэтому требуется 2 кристалла. Приемопередатчики управляются сигналами микропроцессора **DEN** и **DT/R**. В течение цикла чтения шины, сигнал **DT/R** имеет низкий уровень, и данные передаются от выводов **B** к выводам **А**. Сигнал **DEN** разрешает работу приемопередатчиков.

**Рис. 3.2. Блок-схема системы с микропроцессором в минимальном режиме**

В течение цикла записи сигнал **DT/R** имеет высокий уровень, и данные передаются от выводов **А** к выводам **B.**

**RD**, **WR**, **M/IO** – сигналы шины управления микропроцессорной системы. Однако память и устройства ввода/вывода в большинстве случаев требуют других сигналов. Это сигналы **MWTC**, **MRDC**, **IOWC** и **IORC**. **MWTC** и **MRDC** – это сигналы, разрешающие запись в память и чтение из памяти, соответственно. **IOWC** и **IORC** разрешают запись и чтение данных из портов ввода/вывода. Сигналы могут быть сформиро­ваны из сигналов микропроцессора при помощи дополнительных логических схем. Пример такой схемы показывается на рис.3.3.

**Рис. 3.3. Схема формирования управляющих сигналов**

# 4 ПАМЯТЬ микропроцессорной системы

# s4.1 РАЗНООБРАЗИЯ ПАМЯТИ

Основные характеристики запоминающих устройств

Запоминающие устройства (ЗУ) характеризуются рядом параметров, определяющих возможные области применения различных типов таких устройств. К основным параметрам, по которым проводится наиболее общая оценка ЗУ, принадлежат их

* информационная емкость (E),
* время обращения (T) и
* стоимость (C).

Под информационной емкостью ЗУ понимают количество информации, измеряемая в байтах, килобайтах, мегабайтах или гигабайтах, которое может храниться в запоминающем устройстве.

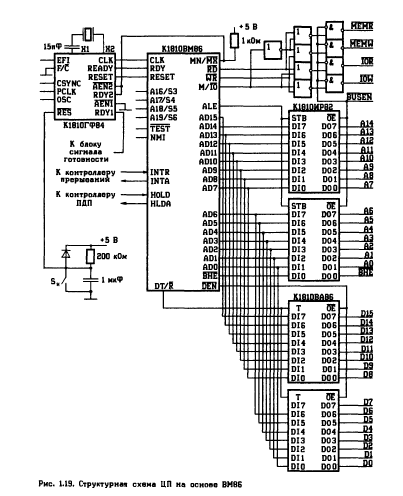
Как известно, приставки кило-, мега- и гига-допускают неоднозначную трактовку в связи с различиями их понимание в общенаучном и специфическом при использовании двоичной системы счисления смыслах. Так, в общем смысле приставка «кило» соответствует 103, "мега" - 106, а "гига" - 109 (на подходе "тера", "пента" и "гекса"). В то же время, близкие по звучанию и смыслу двоичные аналоги этих величин: К-, М и Г-обозначают 210 (1024), 220 (1048576) и 230 (1073741824), что только примерно соответствует перечисленным выше степенями 10. Поэтому при указанию емкости одного и того же устройства памяти например жесткого диска, в Гбайт и миллиардах байт, могут наблюдаться определенные различия.

Рисунок 3.3-Структурная схема ЭВМ на i8086А с шиной адреса данных и управления

Обычно информационная емкость учитывает только полезный объем информации, который не включает объем памяти, расходуемый на служебную информацию, контрольные разряды или байты, резервные области (например, интервал между концом дорожки диска и ее началом), дорожки синхросигналов и пр.

Время обращения к ЗУ различных типов определяется по-разному. В качестве примера можно рассмотреть оперативные ЗП и жесткие диски.

Оперативные ЗУ обычно реализуются как ЗУ с произвольным доступом. Это означает, что доступ к данным, физически организованным в виде двумерного массива (матрицы элементов памяти), производится с помощью схем дешифрирования, выбирают нужные строку и столбец массива по их номерам (адресам). Поэтому время Tоб обращения к ним определяется, в случае отсутствия дополнительных этапов (таких, например, как передача адреса за два такта), время срабатывания схем дешифрации адреса и собственно время записи или считывания данных.

# 4.2. Классификация запоминающих устройств

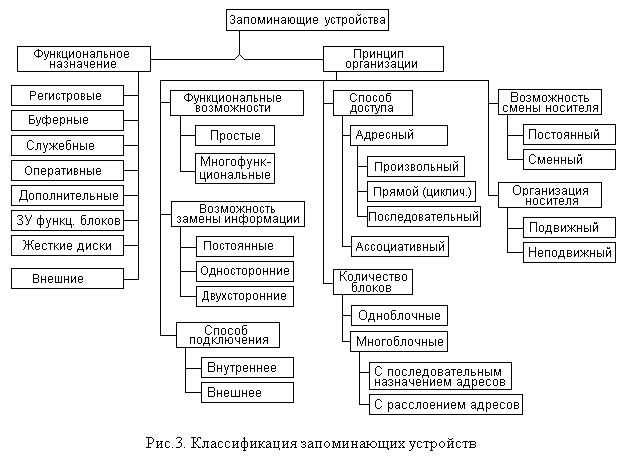
В настоящее время существует большое количество различных типов ЗУ, используемых в ЭВМ и системах. Эти устройства различаются рядом признаков:p

* принципом действия;
* логической организацией;
* конструктивной и технологической реализацией;
* функциональному назначению и т.д.

Большое количество существующих типов ЗУ обусловливает различия в структурной и логической организации (систем) памяти ЭВМ. Необходимые характеристики памяти достигаются не только за счет применения ЗУ с соответствующими характеристиками, но в значительной степени за счет особенностей ее структуры и алгоритмов функционирования.

Память ЭВМ почти всегда "узким местом", что ограничивает производительность компьютера. Поэтому в ее организации используется ряд приемов, улучшающих временные характеристики памяти и, следовательно, повышают производительность ЭВМ в целом.

Классификация запоминающих устройств и систем памяти позволяет выделить общие и характерные особенности их организации, систематизировать базовые принципы и методы, положенные в основу их реализации и использования.

Один из возможных вариантов классификации ЗУ представлен на рис.3. В нем устройства памяти делятся по двум основным критериям: по функциональному назначению (роли или места в иерархии памяти) и принципа организации.

# 4.2.1. Классификация ЗП по функциональному назначению

При разделении ВСУ по функциональному назначению иногда рассматривают два класса: внутренние и внешние ЗУ ЭВМ. Такое разделение сначала основывался на различном конструктивном расположении их в ЭВМ. В настоящее время, например, накопители на жестких магнитных дисках, которые традиционно относят к внешним ЗУ, конструктивно располагаются непосредственно в основном блоке компьютера. Поэтому разделение на внешние и внутренние ЗУ имеет в ряде случаев относительный, условный характер. Обычно к внутренним ЗУ относят устройства, непосредственно доступны процессору, а к внешним - такие, обмен информацией которых с процессором происходит через внутренние ЗУ.

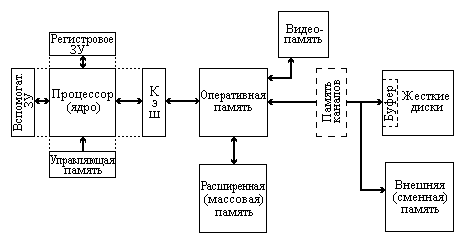
Общий вид иерархии памяти ЭВМ представлен на рис.4.2 На нем показаны различные типы ЗУ, причем поскольку рисунок обобщенный, то не все из представленных на нем ЗУ обязательно входят в состав ЭВМ, а характер связей между устройствами может отличаться от показанного на рисунке.

Рисунок 4.2- Возможный состав системы памяти ЭВМ

1. Верхнее место в иерархии памяти занимают регистровые ЗУ, которые входят в состав процессора и часто рассматриваются не как самостоятельный блок ЗУ, а просто как набор регистров процессора. Такие ЗУ в большинстве случаев реализованы на том же кристалле, что и процессор, и предназначены для хранения небольшого количества информации (до нескольких десятков слов, а в RISC-архитектурах - до сотни), которая обрабатывается в текущий момент времени или часто используется процессором. Это позволяет сократить время выполнения программы за счет использования команд типа регистр-регистр и уменьшить частоту обменов информацией с более медленными ЗУ ЭВМ. Обращение к этим ЗУ проводится непосредственно по командам процессора.

2. Следующую позицию в иерархии занимают буферные ЗУ. Их назначение состоит в сокращении времени передачи информации между процессором и более медленными уровнями памяти компьютера. Буферная память может устанавливаться на разных уровнях, но здесь речь идет именно об указанном ее местоположение. Ранее такие буферные ЗУ в отечественной литературе называли сверхоперативных, сейчас это название практически полностью вытеснил термин "кэш-память" или просто кэш.

Принцип использования буферной памяти во всех случаях сводится к одному и тому же. Буфер представляет собой более быстрое (а значит, и более дорогое), но менее емкое ЗУ, чем то, для ускорения работы которого он предназначен. При этом в буфере размещается только та часть информации с более медленного ЗУ, которая используется в данный момент.

Обмен информацией между кэш-памятью и более медленными ЗУ для улучшения временных характеристик выполняется блоками, а не байтами или словами. Руководят этим обменом аппаратные средства процессора и операционной системы, и вмешательство приложения не требуется. Причем непосредственно командам процессора кэш-память недоступна, то есть программа не может явно указать чтение или запись в кэш-памяти, которая является для нее, как иногда говорят, "прозрачной" (прямой перевод используемого в англоязычной литературе слова transparent).**3. Следующим уровнем иерархии памяти является оперативная память. Оперативное ЗУ (ОЗУ) является основным запоминающим устройством ЭВМ, в котором хранятся выполняются в данный момент процессором программы и обрабатываемые данные, резидентные программы, модули операционной системы и т.п. Название оперативной памяти также несколько изменялось во времени. В некоторых семьях ЭВМ ее называли основной памятью, основной оперативной памятью и пр. В англоязычной литературе также используется термин RAM (random access memory), что означает память с произвольным доступом.**

**Эта память используется в качестве основного запоминающего устройства ЭВМ для хранения программ, которые выполняются или готовых к выполнению в текущий момент времени, и относящихся к ним данных. В оперативной памяти располагаются и компоненты операционной системы, необходимые для ее нормальной работы. Информация, находящаяся в ОЗУ, непосредственно доступна командам процессора, при условии соблюдения требований защиты.**

**Оперативная память реализуется на полупроводниках (интегральных схемах), стандартные объемы ее составляют (в начале 2000-х годов) сотни мегабайт - единицы гигабайт, а времена обращения - единицы ÷ десятки наносекунд.**

**4. Еще одним уровнем иерархии ЗУ может быть дополнительная память, которую иногда называли расширенной или массовой. Сначала (1970-е годы) эта ступень использовалась для наращивания емкости оперативной памяти до величины, соответствующей адресного пространства (например, 24-битного адреса) команд, с помощью подключения более дешевого и емкого, чем ОПГ, запоминает.**

**Затем, в ранних моделях ПЭВМ, дополнительная память также использовалась для наращивания емкости ОЗУ и представляла собой отдельную плату с микросхемами памяти. А еще позже срок дополнительная память (extended или expanded memory) стал обозначать область оперативного ЗУ с адресами выше одного мегабайта. Конечно, этот термин применим только к IBM PC совместимым ПЭВМ.**

**5. В состав памяти ЭВМ входят также ЗУ, принадлежащих отдельным функциональным блокам компьютера. Формально эти устройства непосредственно не обслуживают основные потоки данных и команд, проходящих через процессор. Их назначение обычно сводится к буферизации данных, с любых устройств и поступающих в них.**

**Типичным примером такой памяти является видеопамять графического адаптера, которая используется как буферной памяти для снижения нагрузки на основную память и системную шину процессора.**

**Другими примерами таких устройств могут служить буферная память контроллеров жестких дисков, а также память, которая использовалась в каналах (процессорах) ввода-вывода для организации одновременной работы нескольких внешних устройств.**

**Емкости и быстродействие этих видов памяти зависят от конкретного функционального назначения обслуживаемых ими устройств. Для видеопамяти, например, объем может достигать величин, сравнимых с оперативным ЗУ, а быстродействие - даже превосходить быстродействие последних.**

**6. Следующей ступенью памяти, которая стала фактически стандартом для любых ЭВМ, есть жесткие диски. В этих ЗУ сохраняется практически вся информация, которая используется более или менее активно, начиная от операционной системы и основных приложений и заканчивая редко используемыми пакетами и справочными данными.**

**Емкость этой ступени памяти, которая может включать в свой состав до десятков дисков, обеспечивая хранение очень большого количества данных, зависит от области применения ЭВМ. Типичная емкость жесткого диска, составляющая на начало 2000-х годов десятки гигабайт, удваивается примерно каждые полтора года.**

**Со временем обращения дело обстоит несколько иначе: компоненты этого времени, обусловленные перемещением блока головок чтения-записи уменьшаются сравнительно медленно (примерно вдвое за 10 лет). Компонента, обусловленная временем подведения сектора и зависимая от скорости вращения шпинделя диска, также уменьшается с ростом этой скорости примерно такими же темпами. А скорость передачи данных возрастает значительно быстрее, что связано с увеличением плотности записи информации на диски.**

**7. Все остальные запоминающие устройства можно объединить с точки зрения функционального назначения в одну общую группу, охарактеризовав ее как группу внешних ЗУ. Под словом "внешние" следует понимать, что информация, хранящаяся в этих ЗУ, в общем случае находится на носителях не является частью собственно ЭВМ. Под это определение подпадают дискеты, компакт диски, накопители на сменных магнитных дисках и магнитооптические диски, твердотельные (флэш) диски и флэш-карты, стримеры, внешние винчестеры и др Естественно, что параметры этих устройств весьма различны. Функциональное назначение их обычно сводится либо к архивному хранению информации, либо к переносу ее от одного компьютера к другому.**

**Некоторые сомнения в принадлежности к данной категории могут вызвать сменные диски, устанавливаемые в санки (rack). Такие диски, действительно, лучше отнести к предыдущей (седьмой) группе.**

# **4.2.2. Классификация ЗП ПО ПРИНЦИПУ ОРГАНИЗАЦИИ**

Особенности организации ЗП определяются, в первую очередь, используемым технологиями, логикой их функционирования, а также некоторыми другими факторами. Эти особенности и Соответствующие разновидности ЗП перечисляются ниже.

1. По функциональным возможностям ЗУ можно разделять:

- На простые, допускающие только хранение информации;

- Многофункциональные, которые позволяют не только хранить, но и перерабатывать сохраненную информацию без участия процессора непосредственно в самих ЗП.

Подход, используемый во второй группе ЗУ, в принципе, позволяет создать производительные системы с параллельной обработкой данных. В частности, подобные подходы используются в различных частях видеотракта компьютера.

-по Возможности изменения информации различают ЗП:

- Постоянные (или однократной записью)

- Односторонние (с перезаписью или перепрограммированы)

- Двусторонние.

В постоянных ЗП (ПЗУ) информация заносится или при изготовлении, либо с помощью записи (или, как иначе называют эту процедуру, программирования или прожигания), которая может быть выполнена только один раз. В ходе такой записи меняется сам носитель информации, например, пережигается проводники в микросхемах ПЗУ или формируются лунки в отражает слое CD-ROM.

Односторонними называют ЗП, которые имеют существенно различные времена записи и считывания информации. Наиболее распространенными типами таких ЗУ является перепрограммированы постоянные ЗУ или компакт-диски с перезаписью - CD-RW. Записи в устройствах этих типов значительно превышает время считывания информации.

К односторонним ЗУ можно отнести и ЗУ на приборах с зарядовой связью (ПЗС), в которых записи (формирование изображения), вообще говоря, заметно меньше времени считывания (передачи изображения).

Двусторонние ЗП имеют близкие значения времен чтения и записи. Типичными представителями таких ЗП есть оперативные ЗП и ЗУ на жестких дисках.

2. По способу доступа различают ЗП:

- С адресным доступом;

- С ассоциативным доступом.

При адресной доступе для записи или чтения местоположение информации в ЗУ определяется ее адресу. Логично адрес может иметь различную структуру. Например, в оперативных ЗУ адрес представляет собой двоичный код, одна часть разрядов которого указывают строку матрицы элементов памяти, а другая - столбец этой матрицы. На пересечении заданных строки и столбца находится искомая информация. В ЗУ на магнитных дисках адрес может представлять собой или комбинацию номеров цилиндра, головки и сектора (так называемая CHS-геометрия), или логический номер сектора (LBA-адресация). Возможны и другие варианты.

В любом случае, заданный адрес отрабатывается схемами доступа ЗУ (дешифратором, блоком позиционирования головок и т.п.) таким образом, что в операции участвует соответствующая адресу пиксели элементов памяти, памятной среды или носителя информации. При этом, в зависимости от того, как работает механизм доступа, различают следующие виды адресной доступа:

- Произвольный;

- Прямой (циклический)

- Последовательный.

Термин "память с произвольным доступом" (random access memory - RAM) применяют к ЗУ, в которых выбор места хранения информации проводится непосредственным подключением входов и выходов элементов памяти (через буферы, усилители и логические элементы) к входным и выходным шинам ЗП. Это наиболее быстрый вид адресного доступа, применяемый в оперативных ЗП и кэш-памяти.

При прямом (циклическом) доступе непосредственной коммутации связей оказывается недостаточно. В таких ЗП обычно происходит еще и перемещение данных о механизме чтения / записи, механизма чтения / записи по данным или и то и другое. Физически это может быть как механическое перемещение, например, в жестких дисках, перемещения областей намагниченности, как в ЗУ на магнитных доменах, перенос зарядов и др ..

С логической точки зрения такие ЗП можно сопоставить набора сдвиговых регистров, информация в которых сдвигается циклически и может вводиться в регистр или выводиться из него только в одном из разрядов. Термины "циклический" и "прямой" доступ близкие по смыслу, хотя "прямой доступ" - имеет более широкий смысл.

Последовательный доступ характерен для ЗП, использующих в качестве носителя информации (памятной среды) магнитную ленту, например, для стримеров. В таких ЗУ для доступа к блоку данных необходимо переместить носитель так, чтобы участок, на котором располагается необходим блок данных, оказался под блоком головок чтения / записи.

Кроме того, при всех формах адресного доступа адресуются элементом может быть не только байт или слово (как в оперативной памяти и кэш-памяти), но целый блок данных. Это обычно связано либо с конструктивными особенностями ЗП, или с большим временем доступа.

При ассоциативном доступе место хранения информации при чтении и записи определяется не адресу, а значением некоторого ключа поиска. Каждое записанное и сохраненное в ассоциативной памяти слово имеет поле ключа. Значение этого ключа сравнивается со значением ключа поиска при чтении данных из памяти. В случае совпадения сравниваемых значений информация считывается из памяти.

Ассоциативная память эффективна для решения задач, связанных с поиском данных. Однако ее использование ограничено в силу сравнительно высокой ее сложности.

 Действительно, с аппаратной точки зрения сам поиск может быть организован по-разному: последовательно по разрядам ключевых полей или параллельно по всем ключам во всем массиве памяти. Второй способ, конечно, более быстрый, но требует соответствующей организации (ключевой части) памяти, которая должна иметь для этого в ключевой части каждого сохраняется слова схемы сравнения. Именно поэтому такая память существенно дороже, чем оперативная и используется в основном для решения задач, требующих быстрого поиска в небольших объемах информации.

 Одним из частых применений ассоциативной памяти является быстрое преобразование логических (линейных) адресов данных в физические (то есть адреса ячеек памяти), что выполняется, например, так называемым буфером трансляции адресов. Другой близкой задачей является определение того, является ли необходимая информация в верхних уровнях ЗП или необходимо ее подкачка из более медленных ЗП.

# 4.3 ПОЛУПРОВОДНИКОВЫЕ запоминающее устройство

Полупроводниковые ЗУ в настоящее время представляют собой обширный класс запоминающих устройств, различных по своим функциональным и техническим характеристикам, широко используются как внутренних ЗП ЭВМ. Но этим их использование не ограничивается. Подавляющее большинство электронной и бытовой техники переходит на цифровые методы представления данных (не только текстовых, но и аудио, графических и видео) и управления (использование микроконтроллеров).

Различные области применения накладывают свои особенности на реализацию полупроводниковых ЗП, однако это чаще касается их конструктивных особенностей, а принципы построения одинаковы.

Высокое быстродействие полупроводниковых ЗП обуславливает то, что большинство из них имеет организацию с произвольным доступом. Хотя такие ЗУ, как флэш-память и ЗУ с переносом зарядов (используемые, например, в фото- и видеокамерах), организованы несколько иначе.

Это же высокое быстродействие определяет и основные сферы применения полупроводниковых ЗП в ЭВМ: кэш-память и оперативная память.

Причем надо отметить, что термин "ЗП с произвольным доступом" (Random Access Memory - RAM) не соответствует в точности термина "оперативная память", поскольку первый из них указывает на способ доступа, а второй - на функциональное назначение. И действительно, кэш-память и постоянные ЗУ также ЗУ с произвольным доступом. Однако, согласно принятой в русскоязычной литературе терминологии, термин "оперативные ЗУ" ниже иногда используется как синоним ЗП с произвольным доступом.

В этом разделе рассматриваются основные виды полупроводниковых ЗУ: статическая и динамическая память с произвольным доступом, постоянная и флэш-память, а также приводятся некоторые сведения по другим видам памяти.

# 4.4 постоянное запоминающее устройство

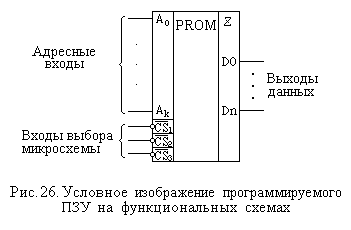
Постоянные запоминающие устройства (ПЗУ или Read Only Memory - ROM), которые также часто называют энергонезависимыми (или Non Volatile Storage), обеспечивающих сохранение записанной в них информации и при отсутствии напряжения питания. Конечно, под такое определение подпадают и память на жестких и гибких дисках, и компакт диски, и некоторые другие виды ЗУ.

 Однако, говоря о постоянных ЗУ, обычно имеют в виду устройства памяти с произвольным адресным доступом. Такие ЗУ могут строиться на различных физических принципах и обладать различными характеристиками не только по вместимости и времени обращения к ним, но и по возможности замены записанной в них информации.

# 4.4.1. Разновидности ПОСТОЯННЫХ ЗП

К началу 2000-х годов наибольшее распространение получили полупроводниковые ПЗУ, элементы памяти которых используют различные модификации диодов и транзисторов и изготавливается по интегральной технологии.

Непосредственными предшественниками таких ЗП были магнитные (трансформаторные) ПЗУ, информация в которые записывалась соответствующей прокладкой (прошивкой) проводников ферритовых сердечников, что обеспечивало при необходимых в то время емкостях высокую надежность этих ЗП в тяжелых (в электромагнитном отношении) условиях.

Известны также емкостные и индуктивные ПЗУ, в которых использовались проводники специальной формы, образующие емкостные или индуктивные связи.

В настоящее время исследуются и другие принципы реализации постоянных ЗП, в некотором смысле возвращаются к магнитным и конденсаторным схемам, но на другом уровне развития технологий.

Запись информации в постоянные ЗП, как правило, существенно отличается от чтения по способу и времени выполнения. Процесс записи для полупроводниковых постоянных ЗУ получил также название "прожигания" или программирования, первое из которых связано с образом записи, сводится к разрушению (расплавлению, прожига) соединительных перемычек в чистом ЗУ.

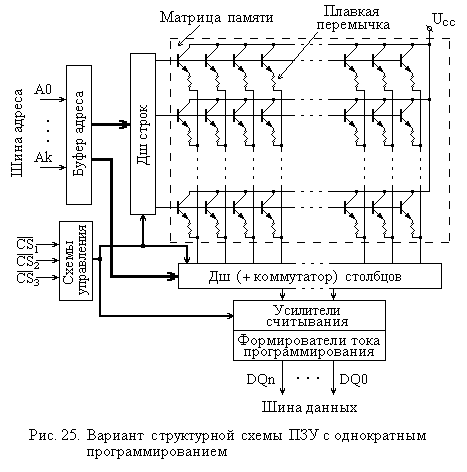
В полупроводниковых ПЗУ как элементы памяти, точнее, в качестве нелинейных коммутирующих и усилительных элементов обычно используются транзисторы. Они объединены в матрицу, выборка данных из которой производится по строкам и столбцам, соответствующим указанному адресу, так же, как и в других ЗУ с произвольным доступом. Один из возможных вариантов структурной схемы полупроводникового ПЗУ, представленный на рис. 25. Строго говоря, непосредственно запоминания информации в этом ПЗУ осуществляется плавкой перемычкой, а транзисторы выполняют роль ключей-усилителей. Плавка перемычка может быть изготовлена ​​из нихрома, поликристаллического кремния или других материалов. В зависимости от того, как работает усилитель считывания (в режиме повторителя или инвертора), наличие перемычки соответствует записи "1" или "0". Разрушение перемычки (импульсом сильного тока) приводит к записи значения, обратного исходному.

Рисунок 4.2-

Различают две большие группы ПЗУ: программируемые производителем и программируемые пользователем.

ЗП первой группы, называемые иначе масочным, обычно выпускаются большими партиями. Информация в них заносится в процессе изготовления этих ЗП на заводах: с помощью специальной маски в конце технологического цикла на кристалле формируется соответствующая конфигурация соединений. Такие ЗП оказываются самыми дешевыми при массовом изготовлении. Их обычно используют для хранения различных постоянных программ и подпрограмм, кодов, физических констант, постоянных коэффициентов и пр.

В ПЗУ, программируемые пользователем, информация записывается после их изготовления самими пользователями. При этом существуют два основных типа таких ЗУ: однократно программируемые и перепрограммированы.

Нетрудно вспомнить, что аналогичные виды есть и в CDROM, которые, по сути, является ПЗУ (ROM), изготавливаемых на основе другого физического принципа.

Наиболее простыми являются однократно программируемые ПЗУ. В этих ЗП запись как раз и проводится с помощью разрушения соединительных перемычек между выводами транзисторов и шинами матрицы (хотя есть и несколько иные технологии). Изображение программируемого ПЗУ на функциональной схеме показано на рис. 26.

Рисунок 4.3-Умовне зображення ПЗП

Перепрограммированные ПЗУ позволяют производить в них запись информации многократно. Конечно, в таких ЗП должен использоваться другой принцип, чем разрушение перемычек в процессе записи. Распространены технологические варианты используют МОП-транзисторы со сложным затвором (составляющим или "плавающим"), который способен накапливать заряд, снижает пороговое напряжение отпирания транзистора, и хранить этот заряд при выключенном питании. Программирование таких ПЗУ и заключается в создании зарядов на затворах тех транзисторов, где должны быть записаны данные (обычно "0", так как в исходном состоянии в таких микросхемах записаны все "1").

Перед повторной записи нужно сделать стирания ранее записанной информации. Оно проводится или электрически, подачей напряжения обратной полярности, или с помощью ультрафиолетового света. В микросхем последнего типа было круглое окошко из кварцевого стекла, через которое и освещался кристалл при стирании.

Параметры постоянных ЗУ соответствуют технологическим нормам своего времени. В начале 2000-х годов типичные емкости микросхем постоянной памяти с масочным программированием составляли около 32-128 Мбит, а времена обращения превышали аналогичные показатели оперативной памяти и для различных модификаций достигали доя 100 нс.

# 4.5 Оперативная память

Оперативная память или оперативное запоминающее устройство (ОЗУ) применяется для хранения оперативной информации, требуемой в процессе обработки. ОЗУ, в отличие от ПЗУ, позволяет как вводить информацию в адресуются ячейку (операция Зп), так и выводить (операция Чт) Так как обращение возможно в любой адресуется организации в произвольном порядке, то такие ЗУ называют запоминающими устройствами с произвольной выборкой (ЗУПВ). Возможны различные способы обмена информацией между ОЗУ и МП. Чаще всего используются запоминающие устройства с произвольной выборкой, позволяющие обращаться с адресом в произвольном порядке. Различают статические и динамические ОЗУ.

Статический ОЗУ реализуется на триггерах с непосредственной связью, которые при включенном питании могут хранить информацию неограниченно долго без дополнительных управляющих сигналов.

Большинство ЗУПВ изготавливаются на основе МОП транзисторов и имеют так называемую матричную организацию. Функциональная схема ЗУПВ 256х1 с матричной организацией типа 16х16 приведена на рисунке 4.4. После дешифровки младших битов адреса А0-А3 возникает сигнал Xi, подключает i-я строка матрицы. Столбец матрицы Yj выбирается дешифратором старших битов адреса А4-А7 и элемент памяти ЕПij подключается к схеме вывода или ввода информации в зависимости от значения сигнала СЧ / / Зп. При СЧ / / Зп = 1 происходит считывание информации из памяти; при СЧ / / Зп = 0 - запись в память. Разрешающий сигнал ВМ (выбор микросхемы) активизирует данную ИС. При запрещающем значении сигнала ВМ матрица изолирована от выходов дешифратора и шины ввода-вывода.

Статические ЗУПВ могут быть построены на полупроводниковых приборах различного типа. В настоящее время наиболее распространенными являются ИС статической памяти на МОП транзисторах. КМОП запоминающий элемент состоит из 5-ти транзисторов, четыре из которых (VT1-VT4) образуют триггер. Управление триггером для записи и считывания осуществляется с помощью ключа - транзистора VT5 (рис.4.5).

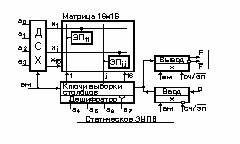
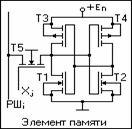
Рисунок 4.4

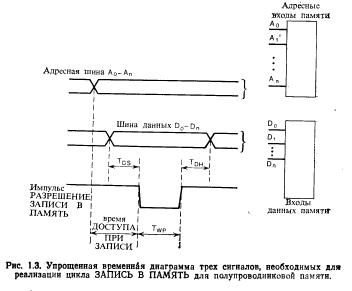
Рисунок 4.5-Елемент пам’яті

В динамических ОЗУ информация хранится в виде некоторого заряда на емкости между затвором информационного МОП-транзистора и общей точкой схемы - землей и паразитной емкости. При комнатной температуре на этой емкости заряд сохраняется в течение десятков миллисекунд. С увеличением температуры ток разряда через МОП транзисторы быстро увеличивается и при 100 0С допустимое время сохранения заряда составляет около 2 мс. Для восстановления заряда на памятной емкости ее нужно периодически подключать к источнику питания, то есть производить регенерацию хранимых данных, что значительно усложняет схему управления. Но количество транзисторов в запоминающем элементе уменьшается до 2-3. Это позволяет значительно увеличить плотность упаковки и уменьшить потребляемую мощность, особенно в режиме хранения. Динамическая память применяется в системах, требующих больших объемов информации, хранимой информации.

# 4.6 ПОДКЛЮЧЕНИЕ ЗП К ШИН

Так как шина данных (ШД) является общей для многих устройств, то при их одновременной работе неизбежен «конфликт». Необходимо, чтобы в каждый момент времени в ШД было подключено только одно устройство - источник информации. приемников же информации может быть одновременно подключено несколько. Важно только, чтобы мощность выхода источника информационных сигналов была достаточной для подключения нескольких приемников. Оперативное подлючение и одключение выхода любого устройства осуществляется с помощью схемы выхода, имеющий 3 возможных состояния: 1, 0 и «отключено».

**Запись данных и память**

 Чтобы понять, каким образом в микропроцессорной системе осуществляется запись данных в память, необходимо выяснить особенности передачи данных в память от любого внешнего источника. С этой целью приведем временную диаграмму общего процесса записи данных в полупроводниковую память, после рассмотрения которой будет легко показать особенности использования системы с 3 шинами для выполнения этой операции.

 Рассмотрим рис. 1.3. Отметим, что адресные входы памяти на рисунке маркируются как А0-А11,

Рисунок 4.6-временных диаграмма цикла записи в памяти

где А n характеризует максимальное число адресных линий, необходимых для данного блока памяти. Например, если память организована как 1024X1. То для обеспечения доступа к любой ячейки памяти необходимо 10 линий, и адресные линии будут обозначаться как А0-А9.

Линии данных маркируются как Do-Do. Где D характеризует максимальное чисто линий данных памяти. Например, если намять организована как 256Х4 бит. Необходимо четыре линии данных. Линии данных такой памяти сказываются D0-D8.

Заметим также, что как адресные входы, так и входы данных должны быть активизированы после выдачи сигнала разрешении записи. Термин «активизировать» здесь означает подачу на адресные линии и линии данных уровня напряжения, соответствующего логическим 1 или 0 и удовлетворяет принятым для используемого семейства логических схем параметрам. Например, если в системе используются схемы семейства TTL, то активизации логической I соответствует интервалу 2,4-5.0 В. а логического 0-0.0-0.4 В.

Интервал времени, в течение которого должны храниться активизированным адресные и информационные входы в выдачи сигнала разрешения записи для различных устройств разный. Например, для устройств памяти типа МОП (металл - окисел-полупроводник) этот интервал составляет 200 нс, а для памяти на схемах TTL - 30 нс. Для точного определения временных соотношении следует ознакомиться с технической документацией на соответствующие изделия.

Анализируя рис. 1.3, важно подчеркнуть, что независимо от конкретных временных соотношений сигналов для заданного устройства памяти микропроцессорная система должна удовлетворить всех существующих временных параметров для устройства памяти данного типа. Если же эти временные соотношения не соблюдаются, то надежной взаимосвязи между микропроцессором и памятью системы обеспечить не удастся.

Для того чтобы успешно реализовать обработку сигнала записи в память в системе с 3 шинами, на шинах системы необходимо выполнить следующие действия:

1. На адресной шине A0-А15 должен быть активизирован адрес памяти (то есть адрес ячейки, куда записываются данные, генерируемые процессором, с шины данных).

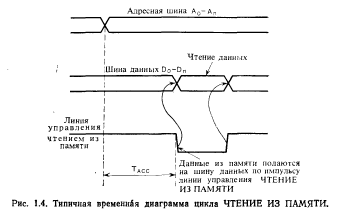
2. На шину данных D0-D8 должны поступить данные из микропроцессора. (Эти данные необходимо записать в ячейку, адрес которой содержится в адресной шине.)

3. После осуществления действий I и 2 на линию записи в память шины управлении должен поступить соответствующим образом синхронизирован импульс разрешения записи необходимого уровня напряжения. При этом осуществляется передача данных. Таким образом, шина управления проводит управление системой с помощью соответствующим образом синхронизированных импульсов.

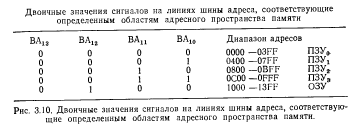
Однако существует ряд дополнительных действий, которым также необходимо уделить внимание. Схемы, реализующие их, проще описывать при более детальном ознакомлении с системой.

Теперь ясно, каким образом сигналы разного логического уровня на каждой из шин {адресной, данных, управления) взаимодействуют между собой при формировании необходимой исходной информации при записи данных в память системы. Шины системы обеспечивают надежное соединение процессора с

памятью системы путем формирования необходимой комбинации сигналов на входах памяти.

**Чтение данных из памяти**

Рассмотрим, каким образом в системе с 3 шинами реализуется чтения данных из памяти. Коротко рассмотрим основные особенности чтения данных с любой полупроводниковой памяти.

На рис. 1.4 приведена типичная временная диаграмма выполнения операции чтения данных из памяти. Отметим, что в соответствии с рис 1.4 адресные линии А0-А n должны быть активизированы с сохранением стабильного уровня сигнала до момента времени, когда данные из памяти поступают на шину данных по сигналу соответствующей линии шины управления. Вспомним, что данные в системе передаются с одного устройства на другое с помощью шины данных. Для выполнения этой операции шина данных переводится а такой режим, при котором логические уровни на этой шине соответствуют данным, сохраненным в памяти. И наконец, сигнал шины управления определяет нужный момент выдачи данных из памяти на шину данных. Таким образом, чтобы реализовать операцию чтения данных на памяти системы необходимо выполнить следующие действия:

1. Обеспечение стабильных уровней сигналов на адресной колючке А0-А n.

2. Подготовка шины данных для приема данных (то есть она переводится в режим приема данных в микропроцессор).

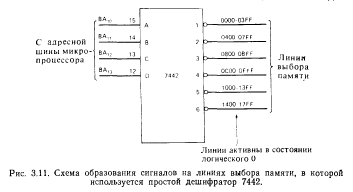
3. После реализации шагов 1 и 2 активизация шиной управления линии управления чтением из памяти. При этом данные из памяти поступают на шину данных и могут быть восприняты микропроцессором. Как видим, реализация операции чтения из памяти предусматривает взаимодействие сигналов соответствующих логических уровней на всех трех шипах.

Рисунок 4.7- Временная диаграмма цикла чтения из памяти

Рисунок 4.8-Использование дешифратора для создания сигналов выбора микросхемы памяти с заданной емкостью

### Управление памятью со стороны процесора

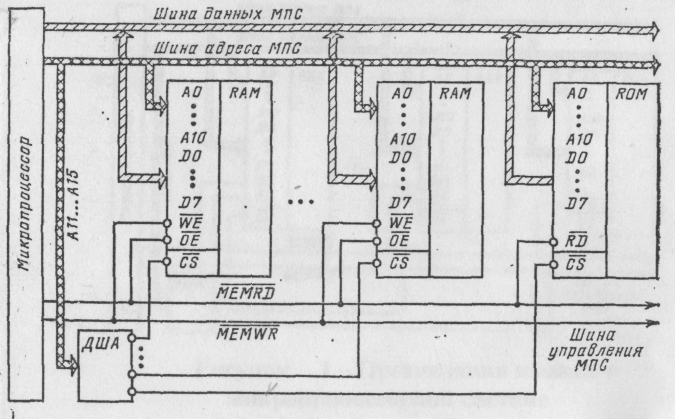
В зависимости от требуемого объема память МПС может состоять из нескольких кристаллов ОЗУ (RAM) и нескольких кристаллов ПЗУ (ROM) (рис. 2.1). Одноименные разряды шины адреса МПС и шин адреса БИС памяти со­единяются между собой. Шины ввода - вывода БИС памяти соединяются с шиной данных МПС.

рисунок - организация памяти в микропроцессорной системе

Для управления записью данных в память служит сигнал шины управления *,* чтением из памяти – сигнал *.* Выбор конкретного кристалла памяти осуществляется с помощью дешифратора адреса памяти ДША, на вход которого поступают старшие разряды с шины адреса МПС (те, которые не подаются на собственно шины адреса БИС). Эти разряды определяют номер кристалла, к которому будет обращение со стороны процессора.

Выходы дешифратора адреса соединяются для этого со входами выбора кристалла отдельных БИС. В качестве дешифратора используются ИС средней степени интеграции -дешифраторы двоичного кода в десятичный.

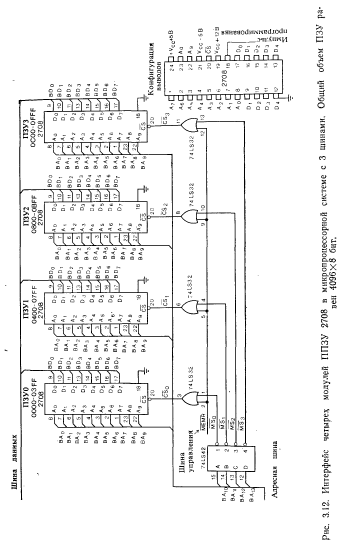
Обмен данными между микропроцессором и памятью можно проиллюстрировать на временных, диаграммах (рис.3.2), из которых видно, как формируются сигналы считывания и запи­си информации.

Нам уже известно, что минимальный интервал времени, в течение которого микропроцессор выполняет те или иные дейст­вия называется тактом. Он определяется тактовой частотой гене­ратора, входящего в микропроцессорную систему.

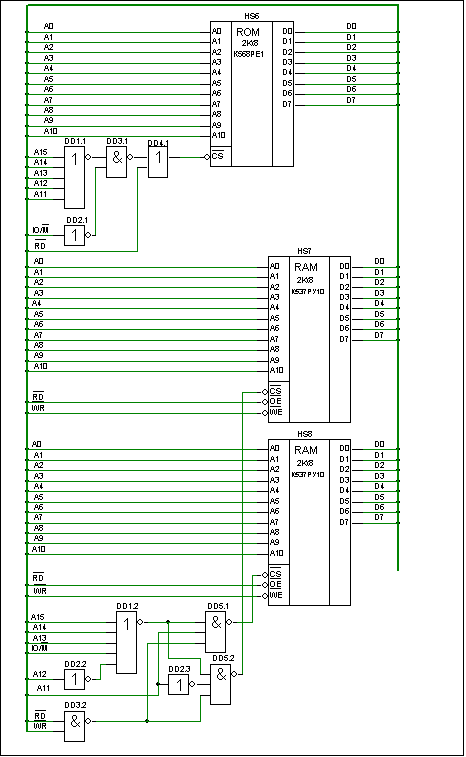
Несколько тактов, в течение которых следует одно обра­щение к памяти или устройству ввода/вывода, называется ма­шинным циклом.

Работа с памятью микропроцессора І8086

**Первый вариант**

Для того чтобы адресовать больший, чем [Intel 8080](https://ru.wikipedia.org/wiki/Intel_8080), объём памяти, потребовалось изменить [способ адресации](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%81%D0%BE%D0%B1_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D0%B8) памяти. Ведь если использовать старые методы, когда адрес к ячейке памяти содержался в указательных регистрах, то пришлось бы увеличивать размер этих самых регистров, чтобы иметь возможность обращаться к большему объёму памяти. Поэтому для адресации 1 Мбайт памяти применили следующую схему. На шину адреса подавался физический адрес размером 20 бит, который формировался путём сложения содержимого одного из сегментных регистров (16 бит), умноженного на 24, с содержимым указательного регистра: таким образом, адресация ячейки памяти производилась по номеру[сегмента](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%BD%D0%B0%D1%8F_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8) и эффективному адресу ячейки в сегменте (называемому также [смещением](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D1%8F))). Если результат сложения оказывался больше, чем 220 − 1, то 21-й бит отбрасывался; такая процедура называется «заворачиванием» адреса ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) address wraparound). Этот метод впоследствии (после появления защищённого режима) назвали [реальным режимом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B5%D0%B6%D0%B8%D0%BC) адресации процессора, такой режим позволяет адресовать до 1 Мбайт памяти.

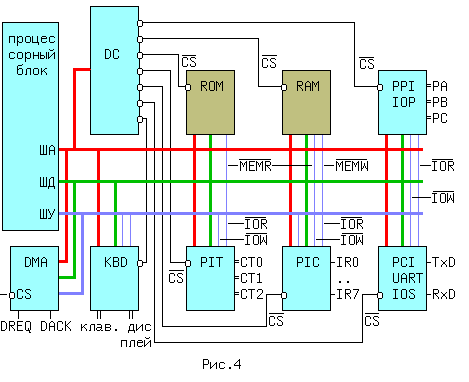
**Второй вариант**

[](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:8086_adress.png)Для того чтобы адресовать 1 мегабайт памяти (20 бит адреса) с использованием 16-битных регистров используется сегментирование. Старшие 4 бит адреса выводятся на отдельные контакты корпуса, а младшие 16 выводятся на совмещённую шину адреса-данных. Но граница сегмента не жёсткая, а плавающая. Для того, чтобы адресовать нужный сегмент, используются 16-битные регистры сегмента, значение которых сдвигается на 4 бита вверх и складывается с указательным 16-битным регистром. Полученное значение — 20-битный адрес памяти или устройства выводится на контакты. Если результат сложения оказывался больше чем 1 мегабайт, выводятся только младшие 20 бит адреса, а старший, 21-й бит, отбрасывается.Схема, показывающая работу реального режима адресации процессора Intel 8086 и выше

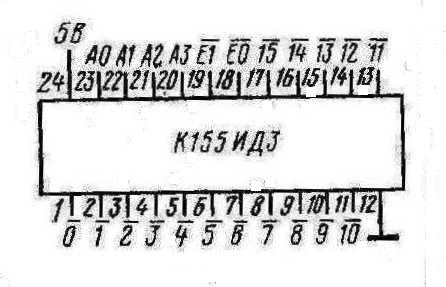
Таким образом, память разделяется на сегменты, размером 64 Кбайт каждый и начинающиеся с адреса, кратного 16 (4 бита двоичного смещения вверх любого из регистра-указателей процессора), сегменты могли перекрываться или совпадать (граница [параграфа](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B3%D1%80%D0%B0%D1%84_(%D0%B5%D0%B4%D0%B8%D0%BD%D0%B8%D1%86%D0%B0_%D0%B8%D0%B7%D0%BC%D0%B5%D1%80%D0%B5%D0%BD%D0%B8%D1%8F)));

В компьютере, подобном [IBM PC](https://ru.wikipedia.org/wiki/IBM_PC), разработчики сэкономили 1 микросхему и решили не использовать разделение адресных пространств для памяти и для устройств ввода-вывода (т. н.верхняя память — [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) upper memory) использовались для видеопамяти и [BIOS](https://ru.wikipedia.org/wiki/BIOS)-а, это ограничивало память, доступную пользователю, объёмом в 640 Кбайт (т. н. обычная память —[англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) conventional memory; страницы 0~9).

На то время такой [режим адресации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D0%B8) обеспечивал множество преимуществ: ёмкость памяти могла составлять до 1 Мбайт, хотя команды оперировали 16-битными адресами; упрощалось использование отдельных областей памяти для программы, её данных и [стека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA); упрощалась разработка устройств, совместимых друг с другом.

Подключение ПЗУ и ОЗУ к системной шине

**Дешифраторы адреса устройств ввода-вывода и памяти**

Работой всех устройств подключаемых к [процессорному блоку](http://de.ifmo.ru/--books/electron/cpu-obu.htm#ПРОЦЕССОРНЫЙ БЛОК (ПБ )) управляет [дешифратор](http://de.ifmo.ru/--books/electron/kombin.htm#ДЕШИФРАТОР) DC, к входам которого подводятся линии шины адреса. Обычно дешифраторов бывает несколько. Если используется не все адресное пространство для памяти и ВУ, то на дешифратор заводятся не все линии адреса, чаще всего несколько старших разрядов ША. Например, если на DC завести 4 линии A19..A16, то все адресное пространство будет разбито на неперекрывающиеся блоки по 2^20 / 2^4 = 64Кб, принадлежащие каждому из 16-ти (2^4 = 16) устройств ЗУ или ВУ, подключенных к шинам (на рис.4 показаны 7 устройств). Часть из них могут использовать все отводимое им адресное пространство, например [ПЗУ](http://de.ifmo.ru/--books/electron/zu-obu.htm#ПЗУ) и [ОЗУ](http://de.ifmo.ru/--books/electron/zu-obu.htm#ОЗУ), другие только несколько адресов.

Типовая схема МПС

**Дешифратор 74154**

Микросхема К155ИД3 (74154) — дешифратор, позволяющий преобразовать четырехразрядный код, поступивший на входы АО — А3 в напряжение низкого логического уровня, появляющееся на одном из шестнадцати выходовО — 16. Дешифратор К155ИД3 (74154) имеет два выхода разрешения дешифрации EO и Е1. Эти входы можно использовать как логические, когда дешифратор К155ИД3 (74154) служит демультиплексором данных. Тогда входы АО — А3 используются как адресные, чтобы направить поток данных, принимаемых входами ЕО и E1, на один из выходов О — 15. На второй, неиспользуемый в этом включении вход Е, следует подать напряжение низкого уровня.

По входам EO и E1 даются сигналы разрешения выходов, чтобы устранять текущие выбросы, которыми сопровождается дешифрация кодов, появляющихся не строго синхронно (например, поступающих от счетчика пульсаций). Чтобы разрешить прохождение данных на выходы, на входы ЕО и E1 следует дать напряжение низкого уровня согласно таблице. Эти входы необходимы также при наращивании числа разрядов дешифрируемого кода. Когда на входахЕО и E1 присутствуют напряжения высокого уровня, на выходах О — 16 появляются высокие уровни.

Дешифратор К155ИД3 (74154) потребляет ток 56 мА. Время задержки распространение сигнала для цепи вход А - выход составляет 36 нс; для цепи вход E — выход 30 нс,

Состояние дешифратора К155ИД3 (74154)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Вход** | | | | | | **Выход** | | | | | | | | | | | | | | | |
| **E0** | **E1** | **A3** | **A2** | **A1** | **A0** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| Н | Н | Н | Н | Н | Н | Н | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | Н | Н | B | B | H | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | Н | B | H | B | B | H | B | B | B | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | Н | B | B | B | B | B | H | B | B | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | B | H | H | B | B | B | B | H | B | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | B | H | B | B | B | B | B | B | H | B | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | B | В | Н | B | B | B | B | B | B | H | B | B | B | B | B | B | B | B | B |
| Н | Н | Н | B | B | B | B | B | B | B | B | B | B | H | B | B | B | B | B | B | B | B |
| Н | Н | B | Н | Н | H | B | B | B | B | B | B | B | B | H | B | B | B | B | B | B | B |
| Н | Н | B | Н | Н | B | B | B | B | B | B | B | B | B | B | H | B | B | B | B | B | B |
| Н | Н | B | Н | B | H | B | B | B | B | B | B | B | B | B | B | H | B | B | B | B | B |
| Н | Н | B | Н | B | B | B | B | B | B | B | B | B | B | B | B | B | H | B | B | B | B |
| Н | Н | B | B | Н | H | B | B | B | B | B | B | B | B | B | B | B | В | Н | B | B | B |
| Н | Н | B | B | Н | B | B | B | B | B | B | B | B | B | B | B | B | B | B | H | B | B |
| Н | Н | B | B | B | H | B | B | B | B | B | B | B | B | B | B | B | B | B | B | H | B |
| Н | Н | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | H |
| Н | B | X | X | X | X | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| B | Н | X | X | X | X | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |
| B | B | X | X | X | X | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B |

# 5 ИНТЕРФЕЙС МП С устройствами ввода-вывода

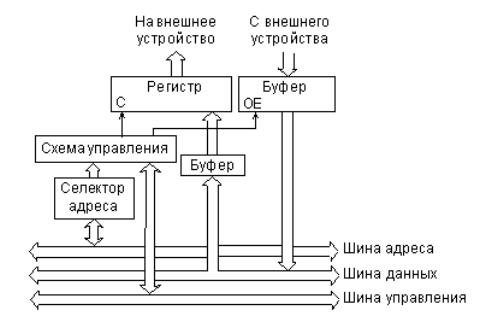
# 5.1 ОРГАНИЗАЦИЯ ввода / вывода в микропроцессорные системы

Вводом / выводом (ВВ) называется передача данных между ядром ЭВМ, включающий в себя процессор и оперативную память, и внешними устройствами (ВУ). Это единственный способ взаимодействия ЭВМ с "внешним миром", и архитектура ВР (режимы работы, форматы команд, особенности прерываний, скорость обмена и т.д.) непосредственно влияет на эффективность всей системы. За время эволюции ЭВМ подсистема ВВ претерпела наибольшие изменения благодаря расширению сферы применения ЭВМ и появлению новых внешних устройств. Особенно важную роль средства ВВ играют в управляющих ЭВМ. Разработка аппаратных средств и программного обеспечения ВВ является наиболее сложным этапом проектирования новых систем на базе ЭВМ, а возможности ВВ серийных машин представляют собой один из важных параметров, определяющих выбор машины для конкретного применения.

**Функции устройств ввода / вывода**

Устройства ввода / вывода обмениваются информацией с магистралью по тем же принципам, что и память. Наиболее существенное отличие с точки зрения организации обмена заключается в том, что модуль памяти имеет в адресном пространстве системы много адресов (до нескольких десятков миллионов), а устройство ввода / вывода обычно имеет немного адресов (обычно до десяти), а иногда и всего один адрес. Но модули памяти системы обмениваются информацией только с магистралью, с процессором, а устройства ввода / вывода взаимодействуют еще и с внешними устройствами, цифровыми или аналоговыми. Поэтому разнообразие устройств ввода / вывода неизмеримо больше, чем модулей памяти. Часто используются еще и другие названия для устройств ввода / вывода: устройства сопряжения, контроллеры, карты расширения, интерфейсные модули и т.д.

Объединяют все устройства ввода / вывода общие принципы обмена с магистралью и, соответственно, общие принципы организации узлов, осуществляющих соединение с магистралью. Упрощенная структура устройства ввода / вывода (точнее, его интерфейсной части) приведена на рис. 2.21. Как и в случае модуля памяти, она обязательно содержит схему селектора адреса, схему управления для обработки стробов обмена и буферы данных.

Простейшие устройства ввода / вывода выдают на внешнее устройство код данных в параллельном формате и принимают с внешнего устройства код данных в параллельном формате. Такие устройства ввода / вывода часто называют параллельными портами ввода / вывода. Они наиболее универсальны, то есть удовлетворяют потребности сообщения с большим числом внешних устройств, поэтому их часто вводят в состав микропроцессорной системы в качестве стандартных устройств. Параллельные порты обычно в составе микроконтроллеров. Именно через параллельные порты микроконтроллер связывается с внешним миром.

Входной порт (порт ввода) в простейшем случае является параллельный регистр, в который процессор может записывать информацию. Выходной порт (порт вывода) обычно просто однонаправленный буфер, через который процессор может читать информацию от внешнего устройства. Именно такие порты показаны для примера на рис. 2.21. Порт может быть и двунаправленным (входным / выходных). В этом случае процессор пишет информацию в внешнее устройство и читает информацию с внешнего устройства по одному и тому же адресу в адресном пространстве системы. Входные и выходные линии для связи с внешним устройством при этом могут быть объединены поразрядно, образуя двунаправленные линии.

Рис. 5.1 - Структура простого устройства ввода / вывода

При обращении со стороны магистрали селектор адреса распознает адрес, приписанный Устройство ввода / вывода. Схема управления выдает внутренние стробы обмена в ответ на магистральные стробы обмена. Входной буфер данных обеспечивает электрическое согласование шины данных с этим устройством (буфер может и не быть). Данные из шины данных записываются в регистр по сигналу С и выдаются на внешнее устройство. Выходной буфер данных передает входные данные с внешнего устройства на шину данных магистрали в цикле чтения из порта.

Более сложные устройства ввода / вывода (устройства сопряжения) имеют в своем составе внутреннюю буферную оперативную память и даже могут иметь микроконтроллер, на который возложено выполнение функций обмена с внешним устройством.

Каждому устройству ввода / вывода отводится свой адрес в адресном пространстве микропроцессорной системы. Дублирование адресов должно быть исключено, за этим должны следить разработчик и пользователь микропроцессорной системы.

Устройства ввода / вывода кроме программного обмена могут также поддерживать режим обмена по прерываниям. В этом случае они превратят поступает от внешнего устройства сигнал запроса на прерывание в сигнал

запроса прерывания, необходимый для данной магистрали (или в последовательность сигналов при векторном прерывании). Если нужно использовать режим ПДП, устройство ввода / вывода должно выдать сигнал запроса ПДП на магистраль и обеспечить работу в циклах ПДП, принятых для данной магистрали.

В составе микропроцессорных систем, как правило, выделяются три специальные группы устройств ввода / вывода:

• устройства интерфейса (ввод информации пользователем и вывода информации для пользователя)

• устройства ввода / вывода для длительного хранения информации;

• таймерные устройства.

К устройствам ввода для интерфейса относятся контроллеры клавиатуры, тумблеров, отдельных кнопок, мыши, трекбола, джойстика и т.д. К устройствам вывода для интерфейса относятся контроллеры светодиодных индикаторов, табло, жидкокристаллических, плазменных и електронно9лучевих экранов и т.д. В простых случаях управляющих контроллеров или микроконтроллеров эти средства могут отсутствовать. В сложных микропроцессорных системах они обязательно. Роль внешнего устройства в данном случае играет человек.

Устройства ввода / вывода для длительного хранения информации обеспечивают сообщения микропроцессорной системы с дисководами (компакт дисков или магнитных дисков), а также с накопителями на магнитной ленте. Применение таких устройств существенно увеличивает возможности микропроцессорной системы по хранению выполняемых программ и накопления массивов данных. В простейших контроллерах эти устройства отсутствуют.

Таймерные устройства отличаются от других устройств ввода / вывода тем, что они могут не иметь внешних выводов для подключения к внешним устройствам. Эти устройства предназначены для того, чтобы микропроцессорная система могла выдерживать заданные временные интервалы, следить за реальным временем, создавать импульсы и т.д. В основе любого таймера лежит кварцевый тактовый генератор и многоразрядные двоичные счетчики, которые могут перезапускать друг друга. Процессор может записывать таймер коэффициенты деления тактовой частоты, количество отсчитываемых импульсов, задавать режим работы счетчиков таймера, а читает процессор выходные коды счетчиков. В принципе выполнить практически все функции таймера можно и программным путем, поэтому иногда таймеры в системе отсутствуют. Но включение в систему таймера позволяет решать сложные задачи и строить более эффективные алгоритмы.

Еще один важный класс устройств ввода / вывода - это устройства для подключения к информационным сетям (локальных и глобальных). Эти устройства распространены не так широко, как устройства трех перечисленных ранее групп, но их значение с каждым годом становится все больше. Сейчас средства связи с информационными сетями вводятся иногда даже в простые контроллеры.

Иногда устройства ввода / вывода обеспечивают соединение с внешними устройствами с помощью аналоговых сигналов. Это бывает очень удобно, поэтому в состав некоторых микроконтроллеров даже вводят внутренние ЦАП и АЦП.

# 5.1.2 программной модели ВНЕШНЕГО УСТРОЙСТВА

Подключение внешних устройств к системной шине осуществляется с помощью электронных схем, называемых контроллерами ВР (интерфейсами ВВ). Они согласуют уровни электрических сигналов, а также преобразуют машинные данные в формат, необходимый устройства, и наоборот. Обычно контроллеры ВВ конструктивно оформляются вместе с процессором в виде интерфейсных плат.

В процессе ввода / вывода передается информация двух видов: управляющие данные (слова) и собственно данные, или данные сообщения. Управляющие данные от процессора, называемые также командными словами или приказами, инициируют действия, не связанные непосредственно с передачей данных, например запуск устройства, запрет прерываний и т.п. Управляющие данные от внешних устройств называются словам состояния; они содержат информацию об определенных признаках, например о готовности устройства к передаче данных, о наличии ошибок при обмене и т.п. Состояние обычно представляется в декодированном форме - один бит для каждого признака.

Регистр, содержащий группу бит, к которой процессор обращается в операциях ВВ, образует порт ВР. Таким образом, наиболее общая программная модель внешнего устройства, которое может выполнять ввод и вывод, содержит четыре регистра ВВ: регистр выходных данных (выходной порт), реестр входных данных (входной порт), реестр управления и регистр состояния (рис. 3.1). Каждый из этих регистров должен иметь однозначный адрес, который идентифицируется дешифратором адреса. В зависимости от особенностей устройства общая модель конкретизируется, например, отдельные регистры состояния и управления объединяются в один регистр, в устройства ввода (вывода) является только регистр входных (выходных) данных, для ввода и вывода используется двунаправленный порт.

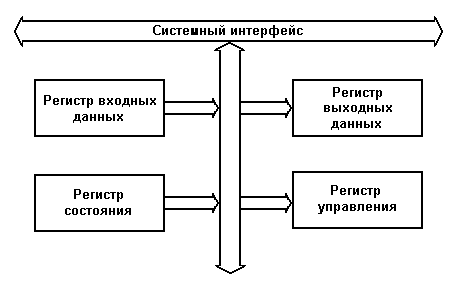


Рисунок 5.2 - Программная модель внешнего устройства

Непосредственные действия, связанные с вводом / выводом, реализуются одним из двух способов, различающихся адресацией регистров ВР.

 Интерфейс с изолированными шинами характеризуется разрешающей адресацией памяти и внешних устройств при обмене информацией. Изолированный ВВ предусматривает наличие специальных команд ввода / вывода, общий формат которых показан на рис. 5.2. При выполнении команды ввода IN содержание адресуется входного регистра PORT передается во внутренний регистр REG процессора, а при выполнении команды OUT содержание регистра REG передается в выходной порт PORT. В процессоре могут быть и другие команды, связанные с ВВ и связанные с проверкой и модификацией содержимого регистра управления и состояния.

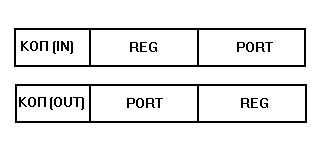


Рисунок 5.3 - Команды ввода / вывода (общий формат)

Нетрудно заметить, что в этом способе адресное пространство портов ввода и вывода изолированно от адресного пространства памяти, то есть в ЭВМ один и тот же адрес могут иметь порт ВВ и ячейку памяти. Разделение адресных пространств осуществляется с помощью сигналов, относящихся к системам ВВ и памяти (MEMRD # - считывание данных из памяти, MEMWR # - запись данных в память, IORD # - чтение порта ВВ, IOWR # - запись в порт ВВ) (# - активный низкий уровень сигналов).

В ЭВМ, рассчитанной на изолированный ВВ, нетрудно перейти к ВР, отраженном на память. Если, например, адресное пространство памяти составляет 64 Кбайт, а для программного обеспечения достаточно 32 Кбайт, то область адресов от 0 до 32 К-1 используется для памяти, от 32 К до 64 К-1 - для ввода / вывода . При этом признаком, дифференцируются обращения к памяти и портов ВВ, может быть старший бит адреса.

Таким образом, интерфейс с общими шинами (ввода / вывода с отображением на память) имеет организацию, при которой часть общего адресного пространства отводится для внешних устройств, регистры которых адресуются так же, как и ячейки памяти. В этом случае для адресации портов ВВ используются полные адресные сигналы: READ - чтение, WRITE - запись.

В операционных системах ЭВМ является набор подпрограмм (драйверов ВВ), которые управляют операциями ВВ стандартных внешних устройств. Благодаря им пользователь может не знать многих особенностей ВУ и интерфейсов ВВ, а применять четкие программные протоколы.

# 5.2 СПОСОБЫ ОБМЕНА ИНФОРМАЦИЕЙ В микропроцессорной системы

В ЭВМ применяются три режима ввода / вывода: программно-управляемый ВР (называемый также программных или нефорсированных ВВ), ВВ по прерываниям (форсированный ВВ) и прямой доступ к памяти. Первый из них характеризуется тем, что инициирование и управление ВВ осуществляется программой, выполняемой процессором, а внешние устройства играют сравнительно пассивную роль и сигнализируют только о своем состоянии, в частности, о готовности к операциям ввода / вывода. Во втором режиме ВВ инициируется НЕ процессором, а внешним устройством, генерирует специальный сигнал прерывания. Реагируя на этот сигнал готовности устройства к передаче данных, процессор передает управление подпрограмме обслуживания устройства, вызвавшего прерывание. Действия, выполняемые этой подпрограммой, определяемых пользователем, а непосредственными операциями ВВ руководит процессор. Наконец, в режиме прямого доступа к памяти, используемый, когда пропускной способности процессора недостаточно, действия процессора прекращаются, она отключается от системной шины и не участвует в передачах данных между основной памятью и быстродействующим ВУ. Заметим, что во всех вышеуказанных случаях основные действия, выполняемые на системной магистрали ЭВМ, подчиняются двум основным принципам. ,

1. В процессе взаимодействия любых двух устройств ЭВМ одно из них обязательно выполняет активную, управляющую роль и является задатчик, второе оказывается управляемым, исполнителем. Чаще всего задатчик является процессор. ,

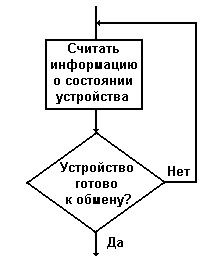
2. Другим важным принципом, заложенным в структуру интерфейса, является принцип цветения (запроса - ответа): каждый управляющий сигнал, посланный задатчик, подтверждается сигналом исполнителя. При отсутствии соответствующего сигнала исполнителя в течение заданного интервала времени формируется так называемый тайм-аут, задатчик фиксирует ошибку обмена и прекращает эту операцию.

# 5.2.1 программно-управляемых вводом / выводом

Данный режим характеризуется тем, что все действия по введению / вывода реализуются командами приложения. Наиболее простые эти действия оказываются для "всегда готовых" внешних устройств, например индикатора на светодиодах. При необходимости ВР в соответствующем месте программы используются команды IN или OUT. Такая передача данных называется синхронным или безусловным ВР.

Однако для большинства ВУ к выполнению операций ВР нужно убедиться в их готовности к обмену, то есть ВВ является асинхронным. Общее состояние устройства характеризуется флагом готовности READY, называемым также флагом готовности / занятости (READY / BUSY). Иногда состояния готовности и занятости идентифицируются отдельными флагами READY и BUSY, входящих в слово состояния

устройства одного или нескольких слов данных. Когда флаг сброшен, процессор выполняет цикл с 2-3 команд с повторной проверкой флага READY до тех пор, пока устройство не будет готов к операциям В \ В (рисунок 5/4). Данный цикл называется циклом ожидания готовности ОП и реализуется в различных процессорах по-разному.Рисунок 5.3 - Команды ввода / вывода (общий формат)

Нетрудно заметить, что в этом способе адресное пространство портов ввода и вывода изолированно от адресного пространства памяти, то есть в ЭВМ один и тот же адрес могут иметь порт ВВ и ячейку памяти. Разделение адресных пространств осуществляется с помощью сигналов, относящихся к системам ВВ и памяти (MEMRD # - считывание данных из памяти, MEMWR # - запись данных в память, IORD # - чтение порта ВВ, IOWR # - запись в порт ВВ) (# - активный низкий уровень сигналов).

В ЭВМ, рассчитанной на изолированный ВВ, нетрудно перейти к ВР, отраженном на память. Если, например, адресное пространство памяти составляет 64 Кбайт, а для программного обеспечения достаточно 32 Кбайт, то область адресов от 0 до 32 К-1 используется для памяти, от 32 К до 64 К-1 - для ввода / вывода . При этом признаком, дифференцируются обращения к памяти и портов ВВ, может быть старший бит адреса.

Таким образом, интерфейс с общими шинами (ввода / вывода с отображением на память) имеет организацию, при которой часть общего адресного пространства отводится для внешних устройств, регистры которых адресуются так же, как и ячейки памяти. В этом случае для адресации портов ВВ используются полные адресные сигналы: READ - чтение, WRITE - запись.

В операционных системах ЭВМ является набор подпрограмм (драйверов ВВ), которые управляют операциями ВВ стандартных внешних устройств. Благодаря им пользователь может не знать многих особенностей ВУ и интерфейсов ВВ, а применять четкие программные протоколы.

Однако для большинства ВУ к выполнению операций ВР нужно убедиться в их готовности к обмену, то есть ВВ является асинхронным. Общее состояние устройства характеризуется флагом готовности READY, называемым также флагом готовности / занятости (READY / BUSY). Иногда состояния готовности и занятости идентифицируются отдельными флагами READY и BUSY, входящих в слово состояния

устройства одного или нескольких слов данных. Когда флаг сброшен, процессор выполняет цикл с 2-3 команд с повторной проверкой флага READY до тех пор, пока устройство не будет готов к операциям В \ В (рисунок 5/4). Данный цикл называется циклом ожидания готовности ОП и реализуется в различных процессорах по-разному.

Рисунок 5.4- Цикл программного ожидания готовности внешнего устройства

Основной недостаток программного ВВ связан с непродуктивными потерями времени процессора в циклах ожидания. К достоинствам следует отнести простоту его реализации, не требует дополнительных аппаратных средств.

Операция ввода / информации процессора осуществляется 2-мя способами.

1. Способ ввода / по методу доступа к памяти

Команда вивода Команда вводу

STA LDA

A15

A0

МП



A15

A0



А15А0

А15А0

В данном способе внешнее устройство имеет общее адресное пространство с ЭВМ.

Преимущества этого способа заключается в том, что он не требует дополнительных аппаратных средств, и кроме того позволяет подключать большое количество внешних устройств. Программная реализация не требует дополнительных команд.

Недостатком данного способа следует считать сложность написания программы (листинга)

Команда виводу OUT Команда вводу IN

RD

WR

A15

A0 МП

I/OWR

A15

A0

I/ORD

1. Изолированный способ ввода-вывода информации, предусматривает работу с ячейками памяти, изолированные от ЭВМ. Изоляция данных ячеек осуществляется путем введения в процессор дополнительных сигналов. Сигналы I / Оrd, I / ОWR служат для чтения или записи информации данных ячеек памяти. Ячейки памяти ЭВМ активизируются с помощью сигналов D / R. Таким образом, осуществляется изоляция ячеек памяти для ввода-вывода информации различных периферийных устройств. Программно для реализации этого способа были введены дополнительные команды I / O (in, out).

недостатки:

1) Ограниченное количество устройств, подключаемых 28-216

2) Необходимы дополнительные аппаратные средства для реализации данного способа.

перева

1) Краткая адресация внешних устройств.

2) Программы более наглядными из-за наличия целевых операторов I / O.

# 5.2.2 ИНТЕРФЕЙС МП ПРИ ввода / вывода единичной ИНФОРМАЦИИ

Под интерфейсом будем понимать совокупность аппаратных и программных средств в необходимых для ввода / вывода информации в / в МП. При этом совокупность аппаратных средств, обеспечивающих доступ к информации на системную шину называют адаптером. В основе построения адаптера лежит Д-триггер, работает по следующему принципу: сигнал (тактовый) входа С подтверждает сигнал, поступающий на вход Д. Так, если наш переключатель П находится в состоянии LOW, то с приходом сигнала на С вход на выходе триггера будет Н. Если же переключатель будет находиться в Н, то с приходом такта на вход с на выходе будет единица, через системную шину поступит в аккумулятор МП. Программа по команде LDA adz 1 МП выставит адрес и сигнал RD (read) по которым через логическую схему "и" через Д-триггер будет занесена информация в аккумулятор.

В следующем программист составляет программу, позволяющую определить в каком положении находится переключатель

D- тригер

Т

S

+5 LOW

A15

D7

A0 МП

RD D0

D

H12H

C

R

D

1

Рисунок 5.5 -інтерфейс мп при виводі одиничної інформації

**Синхронизация вводом / выводом с помощью прерываний**

В предыдущем примере мы предполагали, что когда программа указывает мп ввести данные в порт, они уже имелись в наличии и произошла смена состояния тумблера. Однако, это условие в большинстве случаев не выполняется. Например, клавишный устройство, при работе с которым не возможно предсказать когда будет нажата клавиша. В этом случае проблема решается с помощью прерываний, бывают 2-х типов:

- программные прерывания;

- аппаратные прерывания.

Программное прерывание - наиболее простой метод и используется при работе с небольшим количеством устройств. Основной идеей программного прерывания является ввод данных, используется в программе цикл опроса регистра состояния объекта.

Преимущество - требует меньше аппаратных средств, и находясь в подпрограмме контроллера не требует синхронизации.

Недостатком является необходимость в очень разветвленной программе, занимающих время мп в случаях большого числа устройств опроса и необходимого значительного промежутка времени для ответа на запрос.

Аппаратное прерывание - внешнее устройство при изменении состояния аппаратно формирует сигнал, поступающий на вход int процессора, сообщая процессору о необходимости работы с ним. В этом случае мп, выполнявший текущую программу передает на хранение в стек содержимое аккумулятора счетчик команд и переходит к обслуживанию прерывания, после выполнения которой возвращается к выполнению программы.

Преимущество - быстродействие ответы, лучшее использование мп и потребности во много раз меньше программных средств.

Недостаток - в том, что в этом случае работа мп и внешних устройств происходит синхронно и нужны дополнительные аппаратные средства для решения этой проблемы.

# 5.2.3 преимущества и недостатки изолированных ввода / вывода

1) адрес порта в / в может быть короткой. В большинстве систем для адресацii порта в / в достаточно 8 разрядов, что упрощает декодуючi системы i дает возмож-весть использования коротких команд.

2) возможно легко разработать дополнительные сигналы передачи информации при в / в (стробирования, запуск).

3) программы становятся более наглядными так как операцii в / в происходят за помощью команд, которые отличаются от других.

4) разработка систем в / в может делаться отдельно от разработки моделей памяти.

5) дает возможность использования при малый степени интеграции бис.

# 5.2.4 преимущества и недостатки ввода / вывода, который адресуется как память

1) любая команда, которая работает с данными, которые находятся в памяти, может работать с данными, которые находятся в пристроi ввода или вывода. Никаких особых команд в / в ненужное i программирования многих задач упрощается.

2) не требуется отдельной системы для декодирования в / в.

3) в систему легко включить интерфейс бис и специальные контроллеры. Эти пристроi часто содержат регистры, которые настраиваются программно. Эти обстоятельства являются существенными так как управление в цiх условиях с помощью портов в / в невозможно осуществить

.

# 6 Інтерфейс МП з клавіатурою та індикацією

**Введення інформації з кнопок і клавіатур**

Загальні відомості

Введення інформації з кнопок і клавіатури, підключених до мікро-ЕОМ зазвичай пов'язаний з вирішенням наступних специфічних проблем:

1. захистом від брязкоту контактів кнопок або клавіш;

2. ідентифікацією натиснутою клавіші;

3. забезпеченням потрібного порядку спрацьовування клавіш (при натисненні або при відпуску).

Захист від брязкоту може бути забезпечена або на апаратній, або на програмному рівні.

Для ідентифікації клавіш звичайно потрібно комбінація певних апаратних і програмних засобів.

Потрібний порядок спрацьовування зазвичай забезпечується відповідним побудовою програми підтримки роботи клавіатури.

Ідентифікація натиснутою клавіші

Проблема ідентифікації натиснутою клавіші зазвичай вирішується в мікро-ЕОМ в два етапи. На першому етапі забезпечується генерація клавіатурою при натисканні кожної з клавіш унікального двійкового вода. Цей етап забезпечується або тільки апаратними засобами, або комбінацією апаратних і програмних засобів. Другий етап завжди забезпечується програмними засобами. На цьому етапі код натиснутої клавіші вводиться в мікро-ЕОМ і порівнюється із заздалегідь занесеної в пам'ять таблицею допустимих кодів. Результатом такого порівняння і є ідентифікація, впізнання натиснутою клавіші.

При малій кількості клавіш перший етап ідентифікації вирішується чисто апаратно. Для прийому сигналу з кожної з клавіш тут виділяють окремі розряди, біти в порте прямого введення. Приклад такого рішення ілюструє схема клавіатури, представлена на ріс.4.32. Комбінація натиснутих клавіш S0 ... S7 задає тут унікальний код натиснутої клавіші KNK1, який далі вводиться в мікро-ЕОМ через порт прямого введення PIKNK1. Якщо кількість клавіш більше числа розрядів порту введення, можна включити між клавіатурою і портом шифратор з відповідною кількістю входів. При використанні шифратора, восьмирозрядний порт введення може вводити інформацію максимум від 255 клавіш.

Другий етап ідентифікації натиснутою клавіші забезпечується підпрограмою IDEN2, яка:

1. вводить в мікро-ЕОМ код натиснутої клавіші KNK1;

2. перевіряє, чи міститься цей код в таблиці допустимих кодів;

1. встановлює ознака недопустимого введення Z = 1, якщо ця перевірка не успішна. Некоректний ввід можливий, наприклад, при одночасному натисканні декількох клавіш;

2. Встановлює ознака керуючої клавіші С = 1, якщо натиснута управляюча клавіша;

3. перетворює код натиснутої клавіші KNK1 в іншій - KNK2, більш зручний для подальшого використання.

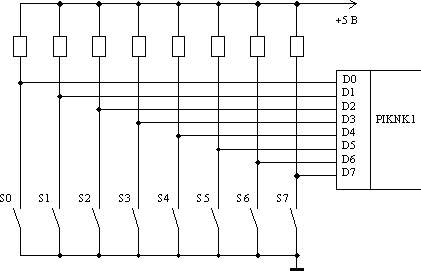


Рисунок 7.1 - Підключення клавіатури до мікро-ЕОМ

Для цифрових клавіш, коди KNK2 повинні відповідати їх маркування. Для керуючих клавіш зручно використовувати в якості кодів KNK2 послідовність парних шістнадцятиричних чисел, що починається з 00Н. Код KNK2 виходить в підпрограмі з проміжного коду CKNK2. Молодша тетрада коду CKNK2 є код KNK2 натиснутою клавіші. Старша тетрада дорівнює 0H для цифрових і 8H для керуючих клавіш. Така структура коду CKNK2 дозволяє одночасно використовувати його як для формування KNK2, так і ознаки керуючої клавіші.

У випадку, якщо на рис.1.2 клавіші S0 ... S3 - цифрові, а S4 ... S7 - керуючі, підпрограма IDEN2 може мати вигляд, представлений на рис.7.2.

Основу підпрограми становить цикл, в якому послідовно встановлюються адреси всіх восьми рядків таблиць ТАВ1 (KNK1) і ТАВ2 (CKNK2) і йде порівняння коду з таблиці ТАВ1 з кодом натиснутою клавіші. Якщо введеного коду немає в таблиці ТАВ1, тобто мав місце некоректний введення, відбувається вихід з підпрограми до встановленого некоректного введення Z = 1. Ця ознака встановлюється командою DCR B при завершенні циклу. Якщо ж введений код пізнаний в одному з кодів ТАВ1, слід вихід з циклу до мітці М1. Реєстрова пара DE містить в цьому випадку адреса коду CKNK2 в таблиці ТАВ2. Далі цей код витягується з ТАВ2 командою LDAX D.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; ПОДПРОГРАММА IDEN2 |
| PIKNK1 | EQU | … | ; ПОРТ ВВОДА PIKNK1 |
| LTAB | EQU | 08H | ; ДЛИНА ТАБЛИЦЫ КОДОВ |
| MASK | EQU | 0FH | ; МАСКА МЛАДШЕЙ ТЕТРАДЫ |
| IDEN2: | IN | PIKNK | ; ВВЕСТИ KNK1 |
|  | MVI | B,LTAB | ; ЗАГРУЗИТЬ ДЛИНУ ТАБЛИЦЫ |
|  |  |  | ; В РЕГИСТР B |
|  | LXI | H, TAB1 | ; ЗАГРУЗ. НАЧ. АДРЕС ТАБЛИЦЫ |
|  |  |  | ; КОДОВ KNK1 В ПАРУ HL |
|  | LXI | D, TAB2 | ; ЗАГРУЗ. НАЧ. АДРЕС ТАБЛИЦЫ |
|  |  |  | ; КОДОВ СKNK2 В ПАРУ DE |
| M0: | CMP | M | ; KNK1 СОВПАДАЕТ СО СТРОКОЙ |
|  |  |  | ;ТАБЛИЦЫ TAB1? |
|  | JZ | M1 | ; ДА, ПЕРЕЙТИ К М1, ИНАЧЕ |
|  | INX | H | ; МОДИФИЦИРОВАТЬ АДРЕСА |
|  | INX | D | ; КОДОВ В ТАВ1 И ТАВ2 |
|  | DCR | B | ; ПРОСМОТРЕНА ВСЯ ТАВ1? |
|  | JNZ | M0 | ; НЕТ, ПОВТОРИТЬ, ИНАЧЕ |
|  | RET |  | ; ВОЗВРАТ С Z=1 |
| M1: | LDAX | D | ; ВЫБРАТЬ CKNK2 ИЗ ТАВ2 |
|  | ANI | MASK | ; ВЫДЕЛИТЬ KNK2 |
|  | MOV | B, A | ; СОХРАНИТЬ В РЕГИСТРЕ В |
|  | MVI | A, 01H | ; УСТАНОВИТЬ |
|  | ORA | A | ; Z=0 |
|  | LDAX | D | ; ВЫБРАТЬ CKNK2 ИЗ ТАВ2 |
|  | RLC |  | ; ПЕРЕНЕСТИ СТАРШИЙ БИТ CKNK2 |
|  |  |  | ; В БИТ С РЕГИСТРА F |
|  | MOV | A, B | ; ВОССТАНОВИТЬ KNK2 В |
|  |  |  | ; РЕГИСТРЕ А |
|  | RET |  | ; ВОЗВРАТ С Z=0 И С=1 / 0 |
| TAB1: | DB | 0FEH, 0FDH, 0FBH, 0F7H | ; ЦИФРОВЫЕ КЛАВИШИ |
|  | DB | 0EFH, 0DFH, 0BFH, 7FH | ; УПРАВЛЯЮЩИЕ КЛАВИШИ |
| TAB2: | DB | 00H, 01H, 02H, 03H | ; ЦИФРОВЫЕ КЛАВИШИ |
|  | DB | 80H, 82H, 84H, 86H | ; УПРАВЛЯЮЩИЕ КЛАВИШИ |

Рисунок 7.2 - Текст підпрограми IDEN2

Особливості ідентифікації натиснутою клавіші в матричної клавіатурі

У мікро-ЕОМ часто використовують так звану матричну клавіатуру. Така клавіатура являє собою прямокутну дротову матрицю, в вузлах якої включені контакти клавіш. Принципова схема одного з реальних варіантів матричної клавіатури наведена на рис 7.3.

Роботу клавіатури підтримують порт прямого введення POKWR і порт прямого введення PIKAR. Легко бачити, що до кожної з чотирьох активних ліній порту введення тут підключено по 6 контактів клавіатури. Зрозуміло, що ці контакти не можуть бути опитані мікро-ЕОМ одночасно. Тому в процесі ідентифікації натиснутою клавіші використовується процедура послідовного опитування вертикальних рядів клавіш. Цю процедуру, здійснювану спеціальної підпрограмою, часто називають скануванням клавіатури.

Порт POKWR призначений для вибору ряду клавіш, опитуваних в даний момент часу. В цей порт виводиться код вибору ряду KWR. Одиничний біт цього коду забезпечує активацію, вибір одного з вертикальних рядів клавіш, задаючи на вертикальному провіднику матриці рівень логічного нуля. В невибраних рядах вертикальні провідники мають рівень логічної одиниці. Тому замикання контактів клавіатури в невибраних рядах не змінює одиничного стану входів D6, D5, D4 і D2 порту PIKAR. Замикання ж контактів в обраному ряду призводить до появи рівня логічного нуля на одному або декількох входах цього порту. Таким чином, формується код активного ряду KAR, який може ввести в мікро-ЕОМ через порт PIKAR.

Послідовний вибір рядів клавіш мінливим в циклі кодом KWR і введення коду KAR відповідного ряду здійснює спеціальний блок підпрограми ідентифікації IDEN. Байтові коди KWR і KAR складають разом унікальний шестнадцатіразрядний код натиснутої клавіші, що використовується на другому етапі ідентифікації. Так, наприклад, якщо цей код дорівнює 0170H, то натиснута клавіша «пробіл».

Для розглянутого варіанта клавіатури коди KWR і KAR можна об'єднати в байтовий код натиснутої клавіші KNK, використовуючи співвідношення

KNK = KWR / 2 KAR х 2.

Це співвідношення може бути реалізовано фрагментом програми, наведеним на рис.7.4. Тут передбачається, що код KWR міститься в регістрі С.

Часто потрібно визначити, натиснута якась (неважливо яка) з клавіш клавіатури. В цьому випадку слід вибрати всі шість рядів клавіш одночасно, видавши в порт POKWR код KWR = 3FH. Порт PIKAR буде при цьому брати код KAR = 74H тільки в тому випадку, якщо не натиснута жодна з клавіш.

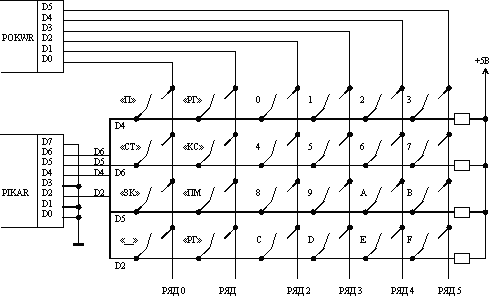


Рис.7.3 - Принципова схема матричної клавіатури

Процеси сканування матричної клавіатури і динамічного управління семісегментним дисплеєм мають багато спільного. Тому їх часто інтегрують, поєднують на апаратній (клавіатура і дисплей зазвичай мають загальний порт вибору ряду POKWR) і (або) на програмному рівні.

|  |  |  |
| --- | --- | --- |
| … |  |  |
| IN | PIKAR | ; ВВЕСТИ КОД KAR |
| RLC |  | ; KAR x 2 |
| MOV | B,A | ; СОХРАНИТЬ KAR x 2 В РЕГ. В |
| MOV | A,C | ; ЗАГРУЗИТЬ KWR В РЕГ. А |
| RAR |  | ; KWR / 2 |
| ADD | B | ; KNK = KWR / 2 + KAR х 2 |
| … |  |  |

Рис.7.4-.Об’єднання кодів KWR і KAR в код KNK

Програмна підтримка роботи клавіатури

Питання забезпечення потрібного порядку спрацьовування клавіш клавіатури, захисту від брязкоту її контактів та ідентифікації натиснутою клавіші зазвичай вирішує спеціальна підпрограма підтримки клавіатури KEY. Її типова блок-схема представлена на рис.7.5. Блоки 1 і 2 забезпечують тут спрацьовування клавіш в момент їх натискання. При запуску підпрограма спочатку очікує відпускання всіх клавіш клавіатури (блок 1). Якщо ж всі клавіші відпущені, підпрограма чекає нового натискання будь-якої з клавіш (блок 3). Відразу ж по цьому натискання йде ідентифікація натиснутою клавіші

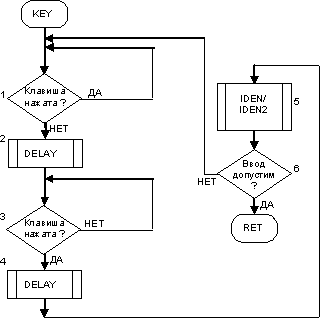


Рис.7.5 - Блок-схема підпрограми IND

(Блок 5) і відбувається вихід з підпрограми з кодом KNK2 і потрібним значенням ознаки керуючої клавіші. При неприпустимому введенні спрацьовує блок 6, повертаючи управління на початок підпрограми.

Помінявши місцями блоки 1,2 і 3,4,5, можна отримати підпрограму, що має інший порядок спрацьовування клавіш - по відпускання натиснутою клавіші.

Програмні затримки DELAY (блоки 2 і 4) тривалістю 10 ... 50 мсек призначені для захисту від брязкоту. В ході відліку цих затримок мікро-ЕОМ не реагує ні на які зміни коду, що генерується клавіатурою. Реакція відновлюється лише після того, як брязкіт закінчився. Зрозуміло, що блоки 2 і 4 можна виключити з підпрограми, якщо використовується апаратна захист від брязкоту, наприклад, активні кнопки.

Розглянуту підпрограму легко модифіковані таким чином, щоб забезпечити одночасну підтримку роботи і клавіатури і семисегментного дисплея. Для цього достатньо:

1. використовувати замість блоків 2 і 4 програмної затримки DELAY підпрограму циклу індикації IND;

2. ввести виконання підпрограми IND в цикли очікування натискання і відпускання клавіш - додаткові блоки 7 і 8.

Блок - схема отриманої після модифікації підпрограми (назвемо її KEYIND) наведена на рис.7.6.

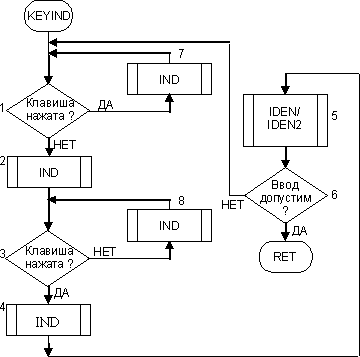


Рис.7.6 - Блок-схема підпрограми KEYIND

Введення в прикладну програму підпрограми KEY (KEYIND) ілюструє блок-схема (рис.7.7).

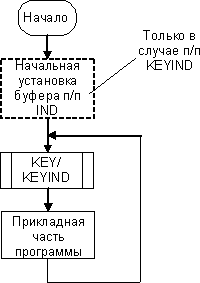


Рис.7.7 - Введення підпрограм KEY (KEYIND) в прикладну програму

Введення і виконання команд управління

Користувач-оператор зазвічай відає команди Управління мікро-ЕОМ натісканням керуючих, функціональніх клавіш клавіатурі. Тому команди Управління вводяться в мікро-ЕОМ кож Як и числа - за допомога підпрограмі KEY (KEYIND). Виконання кожної Такої команди пов'язано з запуском відповідної заздалегідь підготовленої Програми Управління.

При малому чіслі керуючих клавіш допустимо організовуваті такий запуск, вікорістовуючі порівняння (по команді CPI D8) коду натіснутої клавіші з завдання и умовно передачу Управління потрібної Програмі з Z = 1.

Якщо ж керуючих клавіш багато, застосовують табличний спосіб запуску. Цей спосіб вікорістовує таблиці стартових адресу програм, Що запускаються усіма керуючий клавішамі клавіатурі. Стартові адреси програм запісані в таблиці в порядку зростання кодів KNK2 відповідніх керуючих клавіш. Код KNK2 натіснутою функціональної клавіші віявляється при цьому Покажчиком стартового адреси запускається Програми в Цій табліці. ЯКЩО вважаті, Що код KNK2 містіться в акумуляторі, функції табличного запуску програм Може Виконати фрагмент програм, уявлень на рис.7.8.

Слід підкресліті ще раз, що таблиця стартових адресу TABST повинна містіті стартові адреси Всіх наявних на клавіатурі клавіш. ЯКЩО ж натіскання будь-яких клавіш не повинно віклікаті реакції мікро-ЕОМ, слід передбачіті Спеціальну програму-заглушку, запускається цімі клавішамі.

Користувач-оператор зазвичай видає команди управління мікро-ЕОМ натисканням керуючих, функціональних клавіш клавіатури. Тому команди управління вводяться в мікро-ЕОМ також як і числа - за допомогою підпрограми KEY (KEYIND). Виконання кожної такої команди пов'язано з запуском відповідної заздалегідь підготовленої програми управління.

При малому числі керуючих клавіш допустимо організовувати такий запуск, використовуючи порівняння (по команді CPI D8) коду натиснутої клавіші з заданим і умовну передачу управління потрібної програмі з Z = 1.

Якщо ж керуючих клавіш багато, застосовують табличний спосіб запуску. Цей спосіб використовує таблицю стартових адрес програм, що запускаються усіма керуючими клавішами клавіатури. Стартові адреси програм записані в таблицю в порядку зростання кодів KNK2 відповідних керуючих клавіш. Код KNK2 натиснутою функціональної клавіші виявляється при цьому покажчиком стартового адреси запускається програми в цій таблиці. Якщо вважати, що код KNK2 міститься в акумуляторі, функції табличного запуску програм може виконати фрагмент програми, представлений на рис.7.8.

Слід підкреслити ще раз, що таблиця стартових адрес TABST повинна містити стартові адреси всіх наявних на клавіатурі клавіш. Якщо ж натискання будь-яких клавіш не повинно викликати реакції мікро-ЕОМ, слід передбачити спеціальну програму-заглушку, запускається цими клавішами.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; ЗАПУСК ПРОГРАММ, А = KNK2 |
|  | MOV | L,A | ; ЗАГРУЗИТЬ KNK2 В РЕГ. L |
|  | MVI | H,0 | ; ОЧИСТИТЬ РЕГ. H |
|  | LXI | D,TABST | ; ЗАГРУЗИТЬ НАЧ. АДРЕС |
|  |  |  | ; ТАБЛИЦЫ СТАРТОВЫХ |
|  |  |  | ; АДРЕСОВ В ПАРУ DE |
|  | DAD | D | ; ВЫЧИСЛИТЬ АДРЕС МЛ.БАЙТА |
|  |  |  | ; СТАРТОВОГО АДРЕСА |
|  | MOV | E,M | ; ПЕРЕСЛАТЬ МЛ. БАЙТ СТ. |
|  |  |  | ; АДРЕСА В РЕГИСТР E |
|  | INX | H | ; ВЫЧИСЛИТЬ АДРЕС СТ.БАЙТА |
|  |  |  | ; СТАРТОВОГО АДРЕСА |
|  | MOV | D,M | ; ПЕРЕСЛАТЬ МЛ. БАЙТ СТ. |
|  |  |  | ; АДРЕСА В РЕГИСТР D |
|  | XCHG |  | ; ПЕРЕСЛАТЬ СТ.АДРЕС ИЗ |
|  |  |  | ; ПАРЫ DE В ПАРУ HL |
|  | PCHL |  | ; ЗАПУСТИТЬ ПРОГРАММУ |
| TABST: | DW | ST0,ST1,…,ST7 | ; СТАРТОВЫЕ МЕТКИ |
|  |  |  | ; ЗАПУСКАЕМЫХ ПРОГРАММ |

Рис.7.8 - Фрагмент програми, забезпечуючий табличний запуск управляючих програм

**Вивід інформації на індикаційні елементи**

Для виводу інформації на індикацію використовують регісти на базі тригерів або будь- який порт вводу виводу.

Виів одиничної інформації представлений нижче

D

Д0

МП

Д7

А0

А15

RD

WR

а

cs

RD

WR

A0

A15

Дешиф.

STA ard 1.

Програма по команді STA ard 1. МП видає сигнал відповідний зазначеній адресі ard 1, і так само сигнал WR (write) для виводу вмісту акумулятора по зазначеному ard 1. За допомогою адаптера виконаного на базі схем: «і», «інвектора» і Д-тригера здійснюємо подачу сигналу на світлодіод. У випадку, якщо на Д вході одиниця, то з приходом так нового сигналу на С вхід світлодіод дає сигнал (зайнятий) про наявність виходу, що може повідомляти нам про настання якого-небудь повідомлення.

Вивід на семисегментний дисплей

Особливості управління семісегментним дисплеєм

Для виведення інформації мікро-ЕОМ часто використовують багаторозрядних дисплей на семисегментних світлодіодних цифрових індикаторах. Є дві схемотехнічні різновиди таких індикаторів - із загальним анодом і з загальним катодом. Принципові схеми цих індикаторів наведено на рис.7.9. У мікро-ЕОМ зазвичай використовують індикатори із загальним анодом. Цей індикатор може управлятися портом прямого виводу за схемою (рис.7.9).

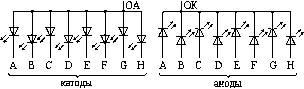


Рис.7.9-Принципові схеми семисегментних індикаторів

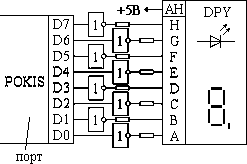


Рис.7.10 - Підключення семисегментного індикатора до порту прямого вивода

Загальний анод індикатора (вивід АН) підключається до плюса джерела живлення (зазвичай 5В), а катоди світлодіодів А ... Н до виходів інверторів, часто мають відкритий колектор. Інвертори, в свою чергу, управляються портом виведення коду | з'являтися, являтися символу POKIS.

Світлодіод запалюється при потенціалі логічного нуля на виході інвертора, тобто в тому випадку, коли у відповідний розряд порту виведений одиничний біт. Резистори R обмежують на допустимому рівні струми через світлодіоди індикатора і вихідні транзистори інверторів. Лінійка інверторів може бути виключена зі схеми, якщо порт POKIS має достатньо потужні інверсні виходи. Може використовуватися і порт з прямими виходами, але тоді в нього слід виводити не код | з'являтися, являтися символу KIS, а його інверсію.

Схему багато розрядного дисплею можна отримати, просто повторивши потрібне число раз схему управління одиничним індикатором. Така схема (її називають схемою зі статичним керуванням) повинна містити стільки портів виводу, яка розрядність дисплея. Тому на практиці її використовують рідко. Як правило, використовується схема динамічного або мультиплексного управління багато розрядним дисплеєм. Типовий варіант такої схеми для шестирозрядного дисплею представлений на рис.7.11.

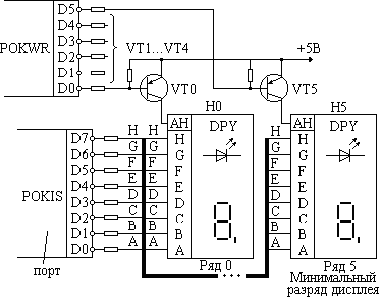


Рис.7.11 - Шестирозрядний семисегментний дисплей з динамічним керуванням

Схему утворюють шість ідентичних вертикальних рядів, кожен з яких містить по семисегментний індикаторі Н0 ... Н5 і транзисторному ключу VT0 ... VT5. Ряди пронумеровані зліва направо цифрами від 0 до 5. Схемою управляють 2 порти прямого виводу POKWR і POKIS.

В порт POKWR виводиться код вибору ряду KWR. Одиничний біт, виведений в цей порт, включає, активує відповідний ряд схеми. В порт POKIS виводиться код | з'являтися, являтися символу KIS. Цей символ буде висвітлений на індикаторі активного ряду.

Типовий режим роботи дисплею пояснюють такі тимчасові діаграми (рис.7.12).

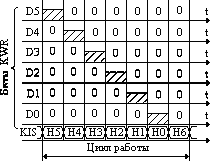


Рисунок 7.12 - Тимчасові діаграми

З діаграм видно, що повний цикл роботи дисплея складається з шести послідовних інтервалів. На кожному з них кодом KWR вибираються, активуються, включається тільки один індикатор, а решта погашені. Ім'я активного індикатора на кожному з інтервалів зазначено на діаграмі коду KIS. Неодночасність роботи індикаторів дисплея виявляється помітною користувачеві тільки при достатньо великому часу циклу. Якщо ж цей час менше 1/30 ... 1/50 сек, у користувача створюється повна ілюзія їх одночасної роботи.

Програмна підтримка роботи семисегментного дисплею з динамічним управлінням

Один повний цикл управління семісегментним дисплеєм формують за допомогою спеціальної підпрограми. Блок-схема і текст цієї підпрограми представлені відповідно на ріс.7.13 і ріс.7.14. Для безперервної індикації рядки символів, занесених до комірки S0 ... S5 досить використовувати підпрограму IND в тілі нескінченного циклу. Відповідний фрагмент програми представлений на ріс.7.13.

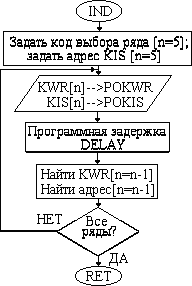


Рис.7.13.Блок-схема підпрограми IND n – номер рядку на схемі дисплею з динамічним управлінням

«Рухомий рядок»

Іноді число знакомест дисплея виявляється менше числа символів, в виведеної рядку. У цих випадках інформація може виводитися на дисплей в режимі «біжучого рядка». Блок-схема програми, що забезпечує цей режим виводу наведена на ріс.7.14.

Програма використовує буфер для семисегментних кодів, що виводяться на дисплей, що має структуру, представлену на ріс.7.28. Символи рядка, що виводиться записуються в N послідовних комірок буфера, починаючи з клітинки S0, після чого управління передається на мітку START0. Перші NN циклів індикації на дисплеї відображаються шість лівих символів рядка. Далі виконується циклічна пересилка (поворот інформації в буфері). При повороті інформація пересилається таким чином, щоб код з комірки S1 опинився в осередку S0, з S2 - в S1 і так далі. Код з осередку S0 пересилається

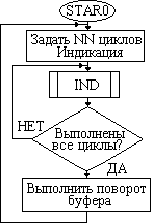


Рис.7.14.Блок-схема програми, яка забезпечує вивід в режимі «бегущей рядка»

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; ПОДПРОГРАММА IND |
| POKWR | EQU | … | ; АДРЕС POKWP |
| POKIS | EQU | … | ; АДРЕС POKIS |
| KWRS | EQU | 00100000B | ; KWR [5] |
| IND: | MVI | B, KWR5 | ; ЗАДАТЬ KWR [n=5] |
|  | LXI | H, S5 | ; ЗАДАТЬ АДРЕС KIS [n=5] |
| M0: | MOV | A, B | ; ВЫВЕСТИ KWR [N] В |
|  | OUT | POKWR | ; ПОРТ POKWR |
| MOV | A, M |  | ; ВЫВЕСТИ KIS [n] В |
| OUT | POKIS |  | ; ПОРТ POKIS |
|  | CALL | DELAY | ; ПРОГРАММНАЯ ЗАДЕРЖКА |
|  | MOV | A, B | ; НАЙТИ KWR [n = n - 1] |
|  | RRC |  | ; |
|  | MOV | B, A | ; |
|  | DCX | H | ; НАЙТИ АДРЕС KIS [n = n - 1] |
|  | JNC | M0 | ; ВСЕ РЯДЫ ? НЕТ, ПОВТОРИТЬ |
|  | RET |  | ; ИНАЧЕ ВЫЙТИ В ОСНОВНУЮ |
|  |  |  | ; ПРОГРАММУ |
| DELAY: | … |  | ; ПРОГРАММНАЯ ЗАДЕРЖКА |
| S0: | DS | 1 | ; БУФЕР ИЗ ШЕСТИ ЯЧЕЕК |
|  |  |  | ; СЕМИСЕГМЕНТНОЙ |
|  |  |  | ; ИНДИКАЦИИ |
| S1: | DS | 1 | ; |
| S2: | DS | 1 | ; |
| S3: | DS | 1 | ; |
| S4: | DS | 1 | ; |
| S5: | DS | 1 | ; |

Рис.7.15 - Текст підпрограми IND

комірку буфера з самим старшим адресою SH. Цю пересилання зручніше виконати в два етапи - спочатку переслати код з S0 в робочу комірку S00 і лише потім, коли буде пересланий код з останньої клітинки буфера SH, занести в цей осередок код з комірки S00. Поворот буфера ілюструє діаграма (рис.7.18). На ній жирним шрифтом виділено

|  |  |  |
| --- | --- | --- |
|  | … |  |
| MM: | CALL | IND | |
|  | JMP | MM | |
|  | … |  |

Рис.7.16.Фрагмент програми, яка забезпечує безперервну індикацію рядка символів

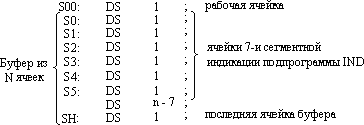


Рис.7.17.Структура буфера програми, яка забезпечує вивід в режимі бегущей рядка

символи, семисегментні коди яких знаходяться в буфері. Числа близько стрілок задають порядок пересилання кодів в процесі повороту. Після повороту буфера знову виконується NN циклів індикації. Легко бачити, що при цьому символи на дисплеї будуть послідовно зміщуватися вліво і рядок на ньому «побіжить». Швидкість її руху визначається числом циклів NN циклів індикації. Чим більше NN, тим повільніше біжить рядок.

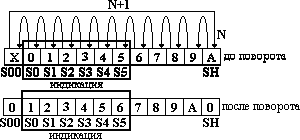


Рисунок 7.18

Для повороту буфера має N осередків можна скористатися фрагментом програми, представленим на ріс.7.19.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; ПОВОРОТ БУФЕРА |
| N | EQU | … | ; ЧИСЛО СИМВОЛОВ В СТРОКЕ |
|  | LXI | H, S0 | ; ЗАНЕСТИ В ПАРУ HL НАЧ. |
|  |  |  | ; АДРЕС ИСТОЧНИКА КОДА |
|  | LXI | D, S00 | ; ЗАНЕСТИ В ПАРУ DE НАЧ. |
|  |  |  | ; АДРЕС ПРИЕМНИКА КОДА |
|  | MVI | B, N | ; ВЫПОЛНИТЬ N ЦИКЛОВ |
| M0: | MOV | A, M | ; ПЕРЕСЛАТЬ КОД ИЗ |
|  | STAX | D | ; ИСТОЧНИКА В ПРИЕМНИК |
|  | INX | H | ; МОДИФИЦИРОВАТЬ АДРЕСА |
|  | INX | D | ; ИСТОЧНИКА И ПРИЕМНИКА |
|  | DCR | B | ; ВСЕ ЦИКЛЫ? |
|  | JNZP | M0 | ; НЕТ, ПОВТОРИТЬ, ИНАЧЕ |
|  | LDA | S00 | ; ПЕРЕСЛАТЬ КОД |
|  | STA | SH | ; ИЗ S00 В SH |
|  | … |  |  |

Рис.7.19 - Фрагмент програми, яка забезпечує поворот буфера

Перетворення двійкових кодів в семисегментні

Висновок на семисегментний дисплей результатів розрахунку або кодів, набраних на клавіатурі, зазвичай пов'язаний з переведенням двійкових кодів шістнадцятиричних цифр в семисегментні. Такий переклад виконується за допомогою підпрограми табличного перетворення. Існує кілька видів такого перетворення. Тут доцільно застосувати простий вигляд, який використовує вихідний двійковий код в якості покажчика адреси свого семисегментного еквівалента в таблиці семисегментних кодів.

Перетворення ілюструє підпрограма BIS7 (рис.7.20), яка перетворює в семисегментний код молодшу тетраду байта, що знаходиться в акумуляторі.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; ПОДПРОГРАММА BIS7 |
| MASK | EQU | 0FH | ; МАСКА МЛАДШЕЙ ТЕТРАДЫ |
| BIS7: | ANI | MASK | ; ВЫДЕЛИТЬ МЛ. ТЕТРАДУ |
|  |  |  | ; ВХОДНОГО КОДА |
|  | MVI | H, 0 | ; ОЧИСТИТЬ РЕГИСТР H |
|  | MOV | L, A | ; ЗАГРУЗИТЬ МЛ. ТЕТРАДУ ВХ |
|  |  |  | ; КОДА В РЕГИСТР L |
|  | LXI | D, TAB7 | ; ЗАГРУЗИТЬ НАЧ. АДР.ТАБЛИЦЫ |
|  |  |  | ; СЕМИСЕГМЕНТНЫХ КОДОВ |
|  |  |  | ; В ПАРУ DE |
|  | DAD | D | ; ВЫЧИСЛИТЬ АДРЕС ВЫХ. КОДА |
|  | MOV | A, M | ; ЗАГРУЗИТЬ ВЫХ. КОД В РЕГ. А |
|  | RET |  | ; ВОЗВРАТИТЬСЯ В ОСНОВНУЮ |
|  |  |  | ; ПРОГРАММУ |
| TAB7: | DB | …, …, …, …, …, …, …, … | ; ТАБЛИЦА СЕМИСЕГМЕНТНЫХ |
|  | DB | …, …, …, …, …, …, …, … | ; КОДОВ ЦИФР 0…F |

Рис.7.20 - Текст підпрограми BIS7

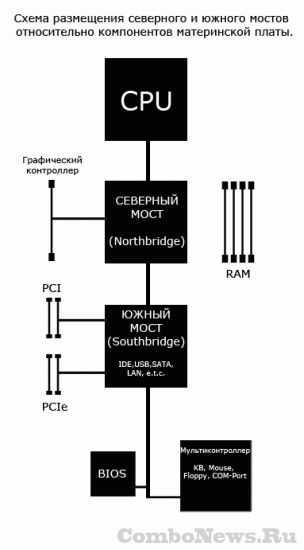
Початковий (базовий) адреса таблиці TAB7 складається в підпрограмі з вихідним кодом 0 ... F. При цьому в регістровий парі HL виходить адресу відповідного семисегментного коду. Цей код витягується з таблиці TAB7 командою MOV A, M і поміщається в акумулятор

# 7 Функціональна схема сучасної ЕОМ

На сучасному етапі шинно-бруківка архітектура в ПК організована найчастіше за принципом "Північ-Південь", з двома основними мостами - північним (Northbridge) і південним (Southbridge). У сукупності ці два мости і необхідне оточення утворюють «чіпсет» системної плати комп'ютера.



Рисунок 7.1 Архітектура сучасного ПК

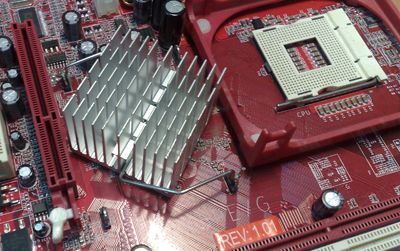


Функцією північного моста є забезпечення взаємодії ЦП, оперативної пам'яті і відеопідсистеми (як самого швидкодіючого інтерфейсного модуля), а також - інтерфейс ядра НД з системною шиною PCI. Південний міст включає контролери для зв'язку із зовнішніми пристроями по інтерфейсах ISA (міст PCI-ISA), EIDE, USB, FireWire, контролери паралельних і послідовних портів, клавіатури, портів PS / 2 і ін Північний міст також може забезпечувати зв'язок з конрроллером SCSI і іншими пристроями. Між північним і південним мостами, пов'язаними між собою шиною PCI, розташовуються фізичні інтерфейси для підключення зовнішніх плат PCI. Крім основного південного моста, розташованого на платі, до північного мосту через шину PCI можуть підключатися додаткові зовнішні південні мости, зі своїми наборами зовнішніх інтерфейсів, що робить таку архітектуру досить відкритою. Перевагою поділу «Північ-Південь» є можливість відокремити специфічну і високошвидкісну архітектуру ядра (оптимізовану під конкретні процесори, модулі пам'яті і відеопідсистему) від стандартної системної шини та зовнішніх інтерфейсів (Зауважимо, що частота «передній» шини (Front Side Bus) в сучасних системах може досягати 150-166 МГц, а для систем на базі Pentium IV - 400 Мгц!).

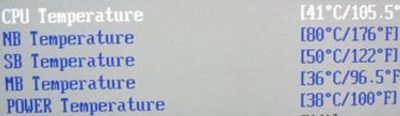
Північний і південний мости комп'ютера (а правильніше буде сказати, материнської плати) - це два основних функціональних контролера, які відповідають за роботу всіх компонентів системної плати і називаються чіпсетом (від англ. Chipset).

**Північний міст**

Північний міст (Northbridge) - це системний контролер, який є одним з елементів чіпсета материнської плати, що відповідає за роботу з оперативною пам'яттю (RAM), відеоадаптером і процесором (CPU). Північний міст відповідає за частоту системної шини, тип оперативної пам'яті і її максимально можливий об'єм. Однією з основних функцій північного моста є забезпечення взаємодії системної плати і процесора, а також визначення швидкості роботи. Частиною північного моста в багатьох сучасних материнських платах є вбудований відеоадаптер. Таким чином, функціональна особливість північного моста являє собою ще і керування шиною відеоадаптера та її швидкодією. Також північний міст забезпечує зв'язок всіх перерахованих вище пристроїв з південним мостом.



Північний міст отримав свою назву завдяки "географічному" розташуванню на материнській платі. Зовні це квадратної форми мікрочіп, розташований під процесором, але у верхній частині системної плати. Як правило, північний міст використовує додаткове охолодження. Зазвичай це пасивний радіатор, рідше - радіатор з активним охолодженням у вигляді невеликого кулера. Пов'язано це з тим, що температура північного моста приблизно на 30 градусів Цельсія завжди вище температури "південного побратима".



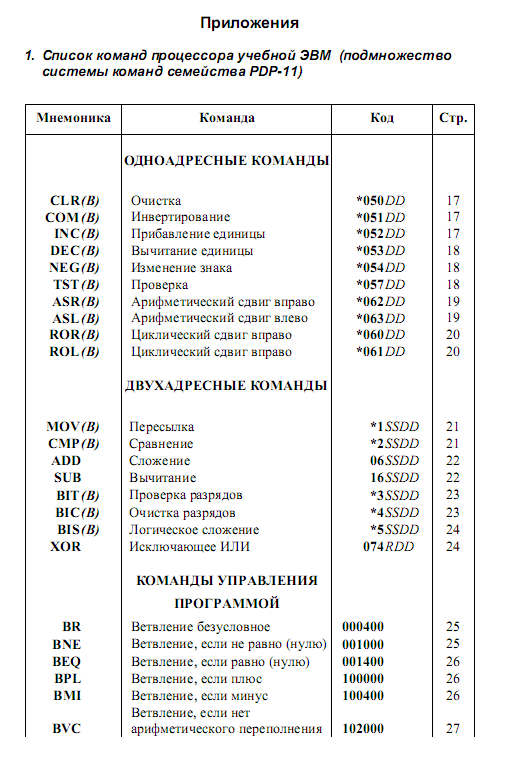
Завищена температура цілком обгрунтована. По-перше, північний міст знаходиться в безпосередній близькості від центрального процесора, по-друге, він знаходиться вище відеокарти, жорстких дисків і південного моста. Це означає, що частина тепла від вищезгаданих пристроїв доходить до північного моста. Ну і по-третє, найголовніше - північний міст відповідає за обробку команд найсильніших компонентів системи - процесор, пам'ять і графіку. Тому будемо вважати, що збільшений номінал температури є нормою для північного моста будь материнської плати.

**Південний міст**

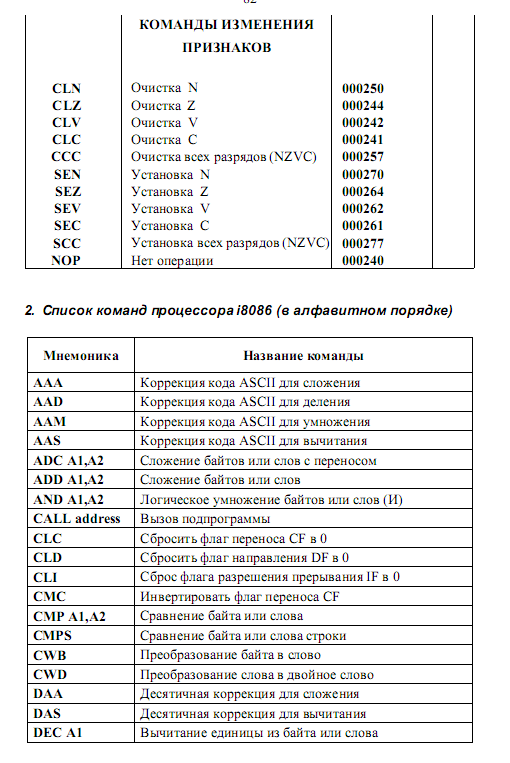
Південний міст (Southbridge) - це функціональний контролер, відомий як контролер вводу-виводу або ICH (In / Out Controller Hub). Відповідає за так звані "повільні" операції, до яких відноситься відпрацювання взаємодії між інтерфейсами IDE, SATA, USB, LAN, Embeded Audio і північним мостом системи, який, в свою чергу, безпосередньо пов'язаний з процесором та іншими важливими компонентами, такими як оперативна пам'ять або відеопідсистема. Також південний міст відповідає за обробку даних на шинах PCI, PCIe і ISA (у старих моделях системних плат).

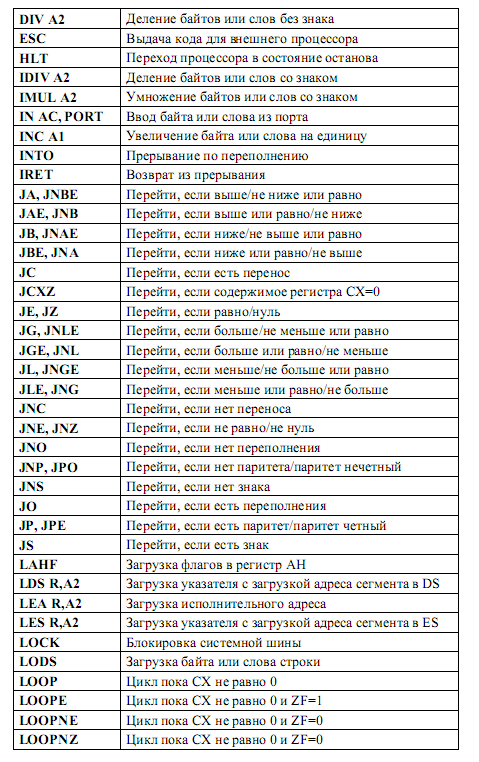


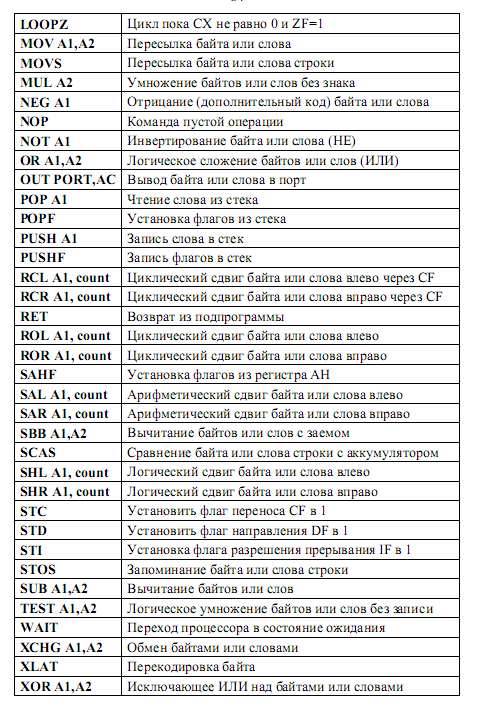
Список обслуговуються систем материнської плати південним мостом досить великий. Крім вищенаведених IDE, SATA, USB, LAN і іншого, південний міст відповідає ще й за SM шину (використовується для управління вентиляторами на платі), DMA-контролер, IRQ-контролер, системний годинник, BIOS, системи енергозабезпечення APM і ACPI, шину LPC Bridge.











# СПИСОК ЛІТЕРАТУРИ

1. 1970s Motorola History Highlights (<http://www.motorola.com/content/0,1037,121-286,00.html>)
2. 1980s Motorola History Highlights (<http://www.motorola.com/content/0,1037,122-287,00.html>)
3. 4 лучше чем 3, а 5? // КомпьютерПресс, 1992, декабрь.
4. Англо-русский словарь по вычислительной технике и программированию. © ABBYY, Е. К. Масловский. 1999.
5. Букчин Л. В., Безрукий Ю. Л. Дисковая подсистема IBM-совместимых персональных компьютеров. – М.: Бином, 1993.
6. Горбунов В. Л., Панфилов Д. И., Преснухин Д. Л. Справочное пособие по микропроцессорам и микроЭВМ. – М.: Высшая школа, 1988.
7. Григорьев В. Л. Архитектура и программирование арифметического сопроцессора. – М.: Энергоатомиздат, 1991.
8. Гук М. Аппаратные средства IBM PC. – СПб: Питер, 1996.
9. Гукин Д. IBM-совместимый персональный компьютер. – М.: Мир, 1993.
10. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. – М.: Финансы и статистика, 1992.
11. Дьяконов В. П. Справочник по алгоритмам и программам на языке бейсик для персональных ЭВМ. – М.: Наука, 1987.
12. Знакомьтесь: компьютер. Под ред. В. М. Курочкина. – М.: Мир, 1989.
13. Знакомьтесь: Персональная ЭВМ Корвет. Ахманов С. А., Персианцев И. Г., Рахимов А. Т. и др. – М.: Наука, 1989.
14. Коффрон Дж. Технические средства микропроцессорных систем. – М.: Мир, 1983.
15. Кэмпбелл Дж. Л. Операционная система OS/2. – М.: Финансы и статистика, 1991.
16. Лю Ю-Чжен, Гибсон Г. Микропроцессоры семейства 8086/8088. – М.: Радио и связь, 1987.
17. Микропроцессорный комплект К1810. Казаринов Ю. М., Номоконов В. Н., Подклетнов Г. С., Филиппов Ф. В. – М.: Высшая школа, 1990.
18. Микропроцессоры: справочное пособие для разработчиков судовой РЭА. Гришин Г. Г., Мошков А. А., Ольшанский О. В., Овечкин Ю. А. – Л.: Судостроение, 1988.
19. Нортон П. Программно-аппаратная организация IBM PC. – М.: Радио и связь, 1992.
20. Нортон П., Гудман Дж. Персональный компьютер: аппаратно-программная организация. – СПб: BHV, 1999.
21. Персональный компьютер «Ириша». Барышников В. Н., Воронов М. А., Кулаков В. Б. и др. – М.: Патриот, 1990.
22. Персональные компьютеры. Информатика для всех. – М.: Наука, 1987.
23. Ратч Э. IBM AT. Руководство для начинающих. – М.: Радио и связь, 1993.
24. Рош У. Л. Последнее слово ещё не сказано: Процессор 486 в семействе PS/2. // PC Magazine USSR, 1991, июнь.
25. Рош У. Л. Библия по модернизации персонального компьютера. – Минск: ИПП «Тивали-Стиль», 1995.
26. Скэнлон Л. Персональные ЭВМ IBM PC и XT. Программирование на языке ассемблера. – М.: Радио и связь, 1989.
27. Смит Б. Э., Джонсон М. Т. Архитектура и программирование микропроцессора Intel 80386. – М.: Конкорд, 1992.
28. Токхайм Р. Микропроцессоры: курс и упражнения. М.: Энергоатомиздат, 1988.
29. Тули М. Справочное пособие по цифровой электронике. – М.: Энергоатомиздат, 1990.
30. Фафенбергер Б., Уолл Д. Толковый словарь по компьютерным технологиям и Internet. – Киев: Диалектика, 1996.
31. Холленд Р. Микропроцессоры и операционные системы. – М.: Энергоатомиздат, 1991.
32. Элфринг Г. Программирование на языке ассемблера для микроЭВМ. – М.: Радио и связь, 1987.
33. Intel думает о том, а не назвать ли новый процессор как-нибудь по-другому. // КомпьютерПресс, 1992, сентябрь.
34. Intel Microprocessor Hall of Fame (<http://intel.com/intel/intelis/museum/exhibit/hist_micro/hof/hof_main.htm>)
35. MASM32 help library: Intel Hex Opcodes And Mnemonics. © S. L. Hutchesson 1999 – 2000
36. Processor Madness (<http://skyscraper.fortunecity.com/dos/661/main.htm>)

37. Сидоренко В.В.Курс лекцій “Мікропроцесорні ВІС”