# 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)

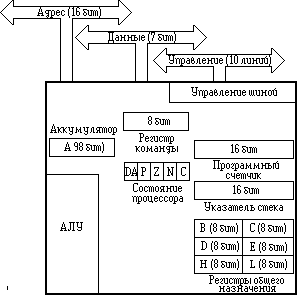
 контроллер периферийных устройств-клавиатуры и индикации

 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 ЭВМ.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.

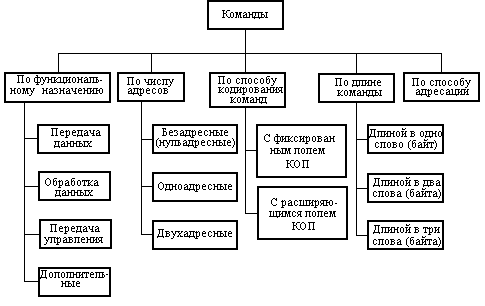
Классификация команд по основным признакам представлена ​​на рис. 2.4. Важнейшим структурным элементом формата любой команды код операции (КоП), что определяет действие, которое должно быть выполнено. Большое число КоП в процессоре очень важно, потому что аппаратная реализация команд экономит память 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;

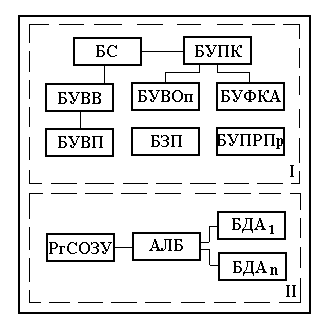
- БУВВ - блок управления вводом / выводом;

**-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

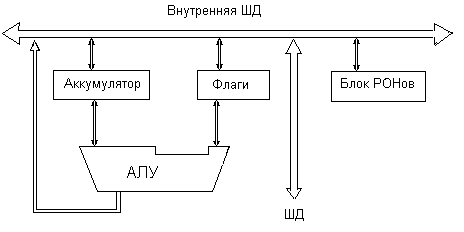
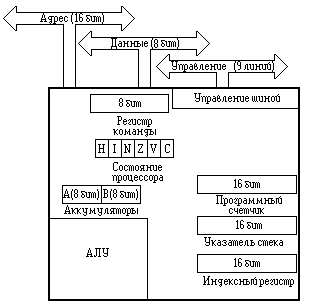
МП фирмы Motorola имел ряд существенных преимуществ. Прежде всего, кристалл МС6800 требовал для работы одного номинала питания, а система команд оказалась достаточно прозрачной для программиста. Архитектура МП также имела ряд особенностей.   


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

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

**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".

Основные технические характеристики процессора:

* -разрядность МД - 8;
* - разрядность МА - 16;
* -адресное 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

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

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 | | | | | | | | | | | | | | | | Source Index |
| DI | | | | | | | | | | | | | | | | Destination Index |
| Указательные регистры | | | | | | | | | | | | | | | | |
| **BP** | | | | | | | | | | | | | | | | Base Pointer |
| **SP** | | | | | | | | | | | | | | | | Stack Pointer |
| 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 | | | | | | | | | | | | | | | | Code Segment |
| DS | | | | | | | | | | | | | | | | Data Segment |
| ES | | | | | | | | | | | | | | | | ExtraSegment |
| SS | | | | | | | | | | | | | | | | Stack Segment |
| Указатель команды | | | | | | | | | | | | | | | | |
| IP | | | | | | | | | | | | | | | | Instruction Pointer |

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

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

**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**

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

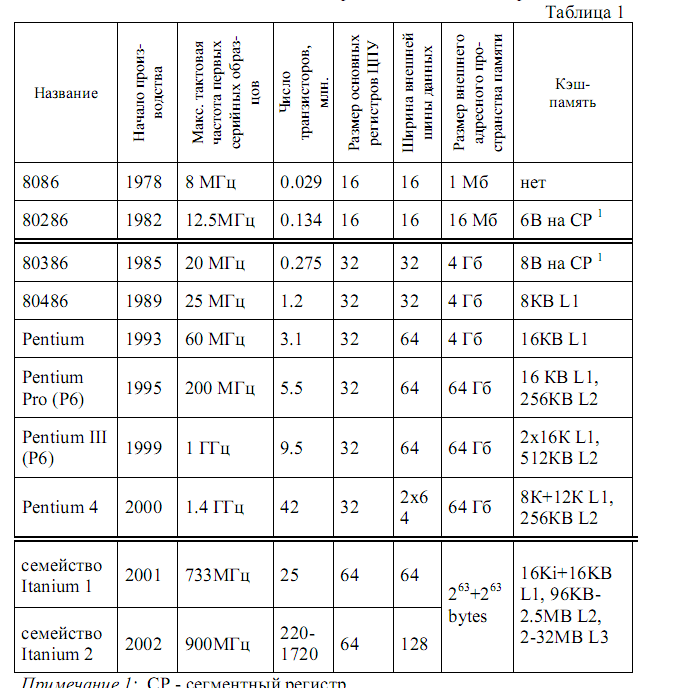
Для работы процессора Intel 8086 требовался полностью новый 16-разрядный набор микросхем поддержки (шинные формирователи,[мультиплексоры](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BF%D0%BB%D0%B5%D0%BA%D1%81%D0%BE%D1%80) и демультиплексоры, [интерфейсные](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%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

* 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 процессоров IA (Intel Architecture), является микропроцессор 8086, разработанный к 1978 году. Программы, написанные для него, выполняются на всех последующих процессорах семейства, включая все современные модели. 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]. Сегодня процессоры этого семейства стали стандартом де-факто для большинства персональных компьютеров (ПК) во всем мире. Ниже приводятся основные данные наиболее известных представителей этой серии:

# 2.1 Микроархитектура процессоров 8086 и Pentium Pro

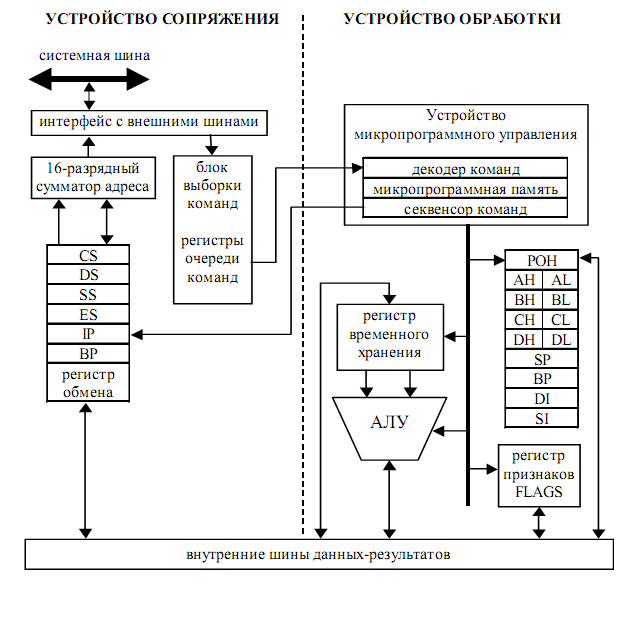
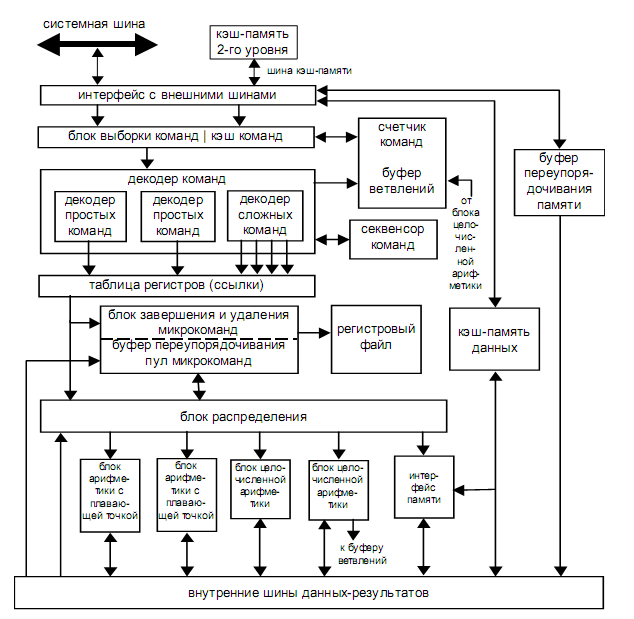
Микропроцессор 8086 ориентирован на выполнение команд параллельно с их выборкой и может быть условно разделен на две части, работающие асинхронно ( Рис. 2.1): устройство сопряжения с внешними шинами (УС) и устройст-во обработки (УО). Устройство сопряжения обеспечивает формирование 20-разрядного физического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

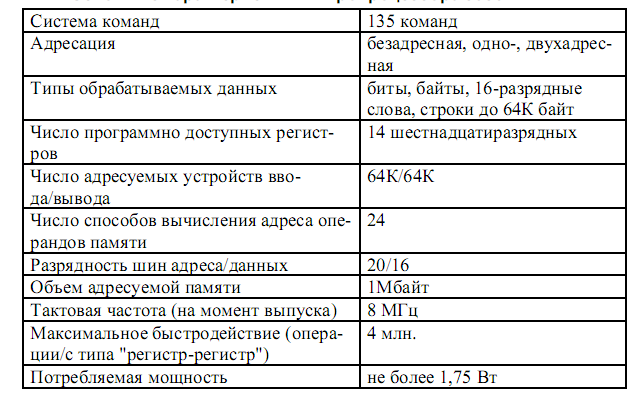
(Рис. 2.2). На Рис. 2.3 представлены обозначения микропроцессоров для принци-

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

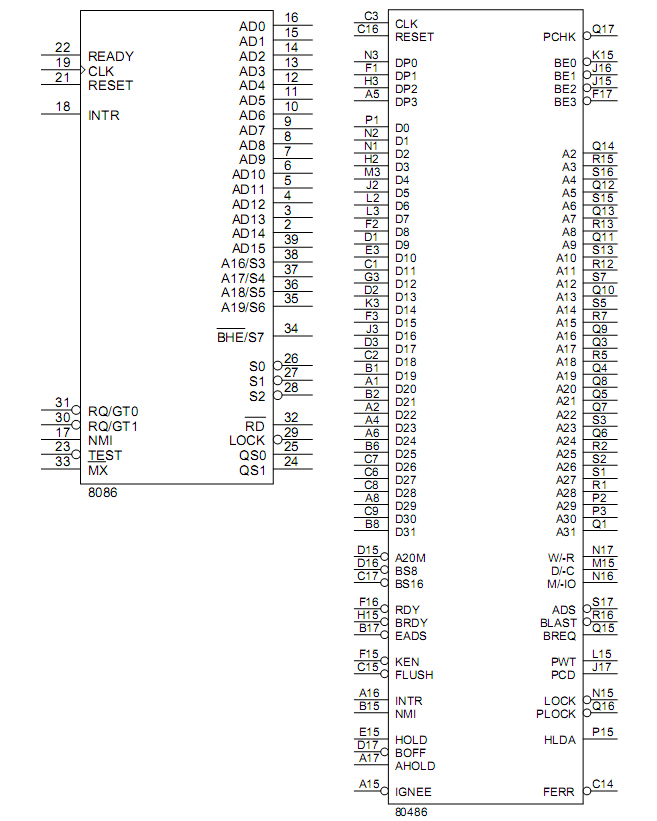
Биты (или флаги) регистра признаков FLAGS разделяются на условные, отражающие результат предыдущей операции ALU, и управляющие, от которых зависит выполнение специальных функций.

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, возникающем при сложении/вычитании байтов или слов, а также принимает 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 результата 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;

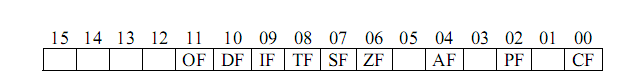
• флаг переполнения OF устанавливается в 1 при потере старшего бита результата сложения или вычитания;

• если флаг направления DF установлен в 1, то используется автодекрементная адресация при выполнении операции обработки строк, если DF установлен в 0, то используется автоинкрементная адресация;

• если флаг прерывания IF установлен в 1, то внешние маскируемые 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.

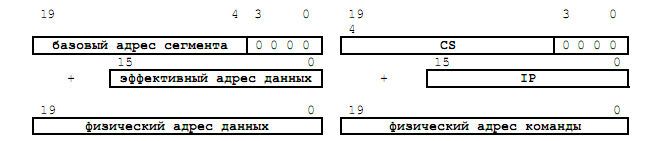


Рис. 2.6 Формирование физического адреса данных и команды

Память организована в виде одномерного массива байтов с физически-

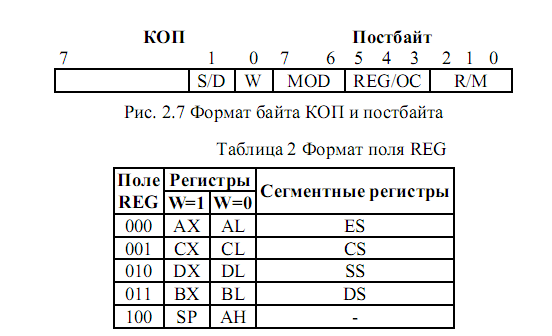
ми адресами от 0000016 до FFFFF16. Две области адресного пространства памяти зарезервированы для выполнения специальных функций, связанных с обработкой прерываний и системным сбросом. Этими областями являются первые 128 байт (физические адреса 00000 - 0007F) и последние 16 байт (физические адреса FFFF0 - FFFFF). Данные области использовать для других целей нельзя. Байты в памяти организуются в слова таким образом, что байту, имеющему 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 адресом, что записывается в виде [00000] = 1234h. Тогда старший байт слова, [00001] = 12h. Квадратные скобки обозначают ячейку памяти, адрес которой находится в этих скобках, h – шестнадцатеричную систему счисления.

Команды, байты и слова можно размещать по любому адресу байта, однако рекомендуется размещать слова в памяти по четным адресам, так как процессор может передавать такие слова за один цикл обращения к памяти. 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 обрабатываемых операндов команды подразделяются на безадресные, одноадресные и двухадресные. В двухадресных командах результат всегда записывается по первому адресу и только один из операндов может находиться 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 выполняются 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 – определяет расширение 8-битных непосредственных данных до полного размера (s=1) или нет (s=0). При некоторых сочетаниях команд и методов адресации (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 операнда, описанного постбайтом и наличия непосредственного операнда. Постбайт состоит из трех полей: режима - MOD, регистра - REG и регистра/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) постбайта. 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, расположенные в команде непосредственно за байтом адресации.

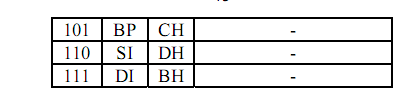
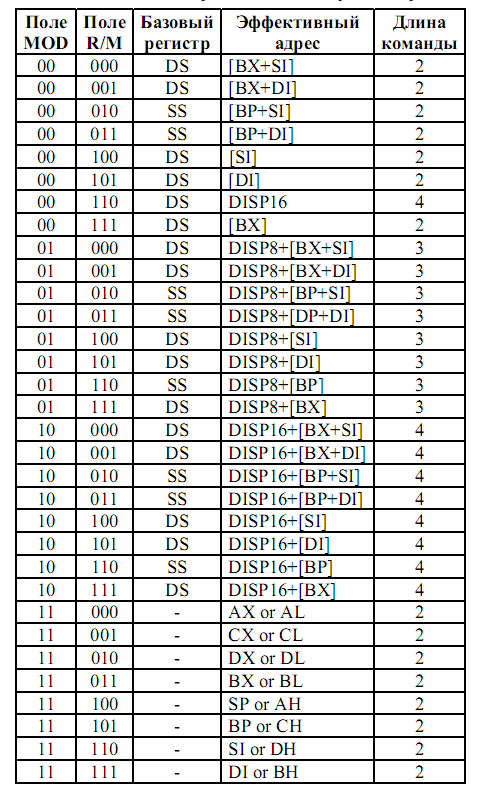


Таблица 3 Кодирование типа адреса операнда



Команде может предшествовать префикс – байт со специальным кодированием, которое изменяет операцию следующей за ним команды. В системе команд процессоров 8086/8088 есть два таких префикса – REP (REPeat) для повторения команд обработки строк и SEG (SEGment) для прямого указания команде сегментного регистра вместо регистра, используемого по умолчанию.

#### 2.2.5 Методы адресации

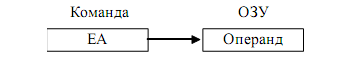
Методы адресации можно разделить на два класса: адресация данных и адресация переходов. Все методы можно отнести к одной из следующих групп:

Прямая адресация.

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

Пример: mov al, [0000h]

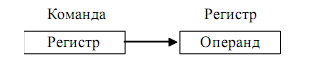
Примечание: запись в квадратных скобках обозначает ячейку, адрес которой (смещение по отношению к DS) записан в квадратных скобках.

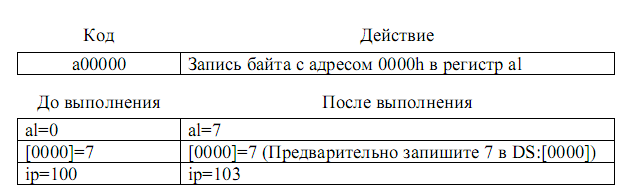
Регистровая адресация.

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:

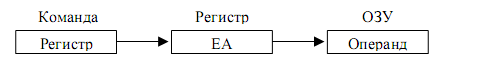




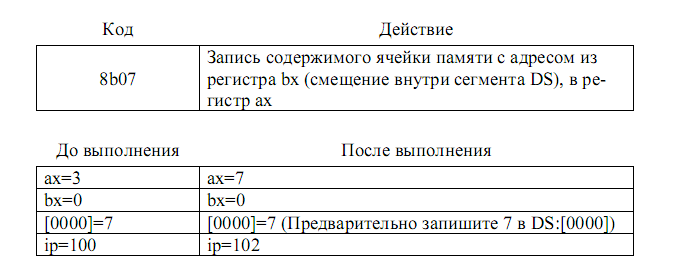
Пример: mov ax, bx

Косвенно-регистровая адресация.

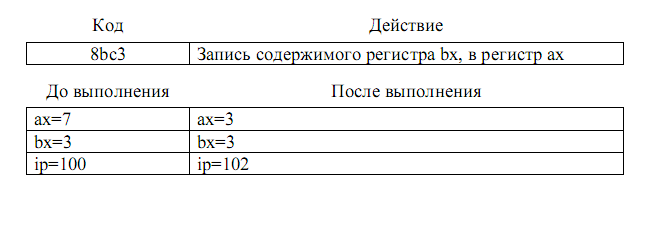
Эффективный адрес операнда находится в базовом регистре BX или индексном регистре (SI или DI):



Пример: mov ax, [bx]

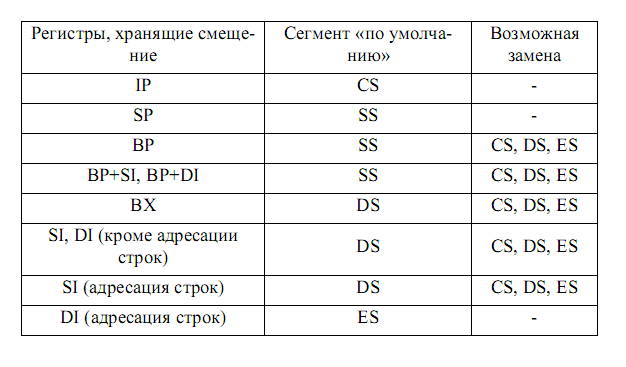


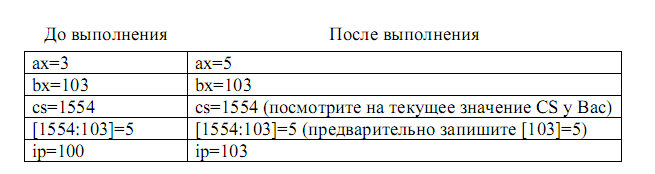
Физический адрес определяется парой сегмент-смещение (например CS:IP – адрес следующей команды), и для каждого регистра, содержащего смещение, есть сегментный регистр, заданный по умолчанию. Некоторые сегменты разрешается принудительно переназначать, мнемоническое обозначениепереназначения «сегментный\_регистр:смещение», в коде команды появляется дополнительный (первый) байт – байт замены сегмента.



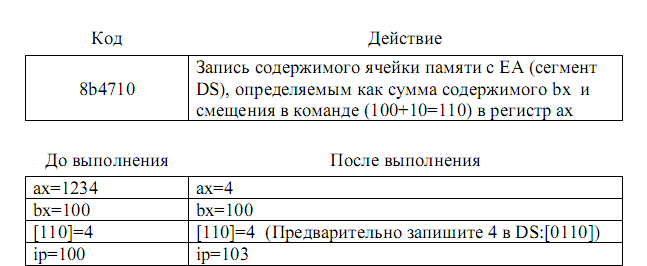
Пример: mov ax, cs: [bx]

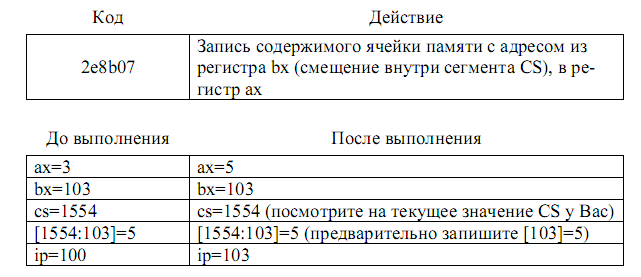
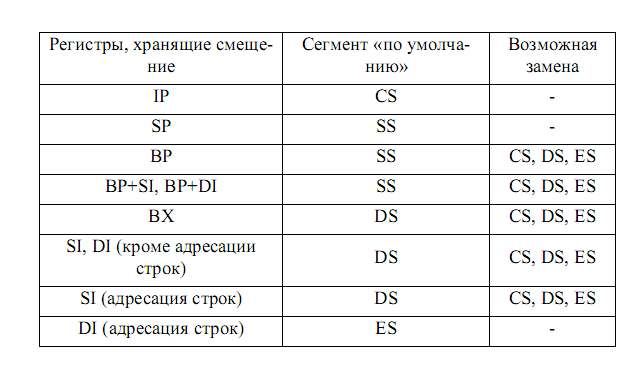
Код Действие 2e8b07 Запись содержимого ячейки памяти с адресом из регистра bx (смещение внутри сегмента СS), в регистр ax



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 обозначение переназначения – «сегментный\_регистр:смещение», в коде команды появляется дополнительный (первый) байт – байт замены сегмента.

Пример: mov ax, cs:[bx]

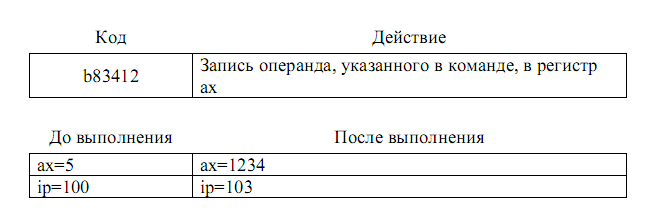




Непосредственная адресация.

Операнд длиной байт или слово является частью команды. Операнд помещается в последние байты команды, причем младший байт следует первым (находится по меньшему адресу).

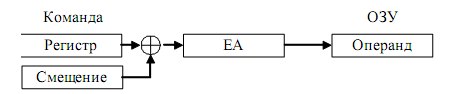
Пример: mov ax, 1234h



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,

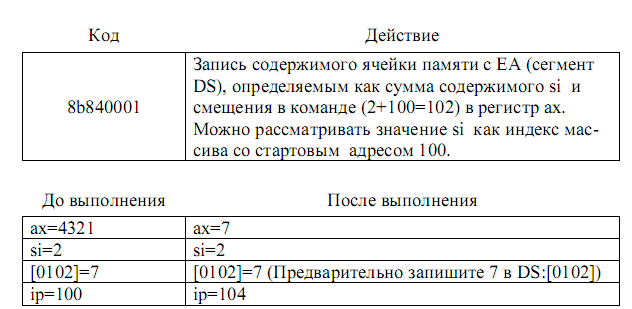
при использовании SI, DI.

Базовая адресация обеспечивает возможность работы со структурами 49 данных, размещенными в памяти, например, с соседними ячейками памяти относительно эффективного адреса в базовом регистре.

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

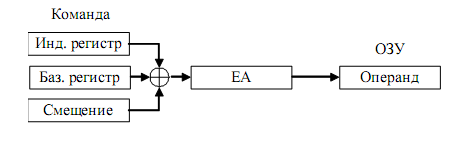
Индексный метод адресации удобен при обработке массивов, когда смещение указывает стартовый адрес массива, а содержимое индексного регистра соответствует индексу массива.

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

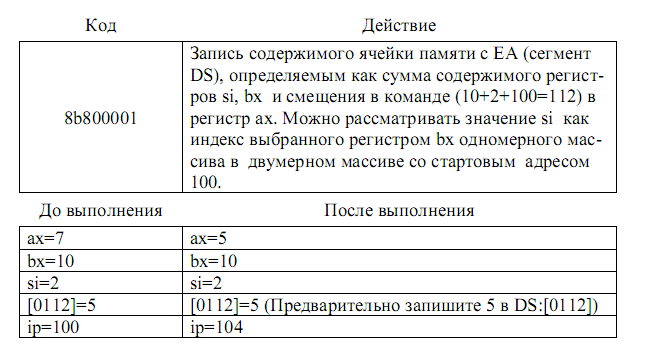


Базовая индексная адресация.

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 обработки 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 автоматически уменьшается или увеличивается на два при записи в стек или при чтении из стека. Заполнение стека происходит в направлении уменьшения адресов ячеек памяти. Стек может обмениваться данными с регистрами общего назначения и сегментными регистрами. Команды обмена данными между стеком и памятью содержат байт адресации, в котором 3разрядное поле REG, совместно с полем КОП, идентифицирует команду.

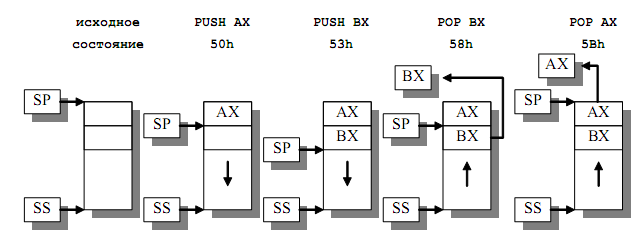
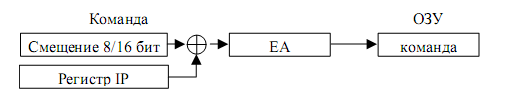


Рис. 2.8 Работа со стеком

Адресация переходов.

Внутрисегментный прямой переход.

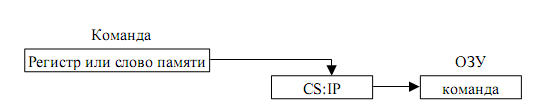
Адрес следующей команды определяется суммированием смещения со

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

Пример: jmp 100

Адресация переходов.

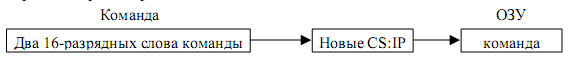
Внутрисегментный косвенный переход.

Содержимое регистра IP заменяется содержимым регистра или слова памяти, заданным любым методом адресации, кроме непосредственного:

Пример: jmp [bx]

Адресация переходов.

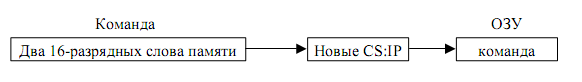
Межсегментный прямой переход. Содержимое регистров CS и IP заменяются словами из команды:



Адресация переходов.

Межсегментный косвенный переход.

Содержимое регистров CS и IP заменяется словами, последовательно расположенными в памяти. Слова могут быть указаны любым методом адресации, кроме непосредственного:

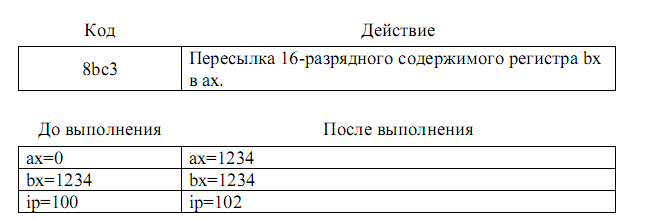


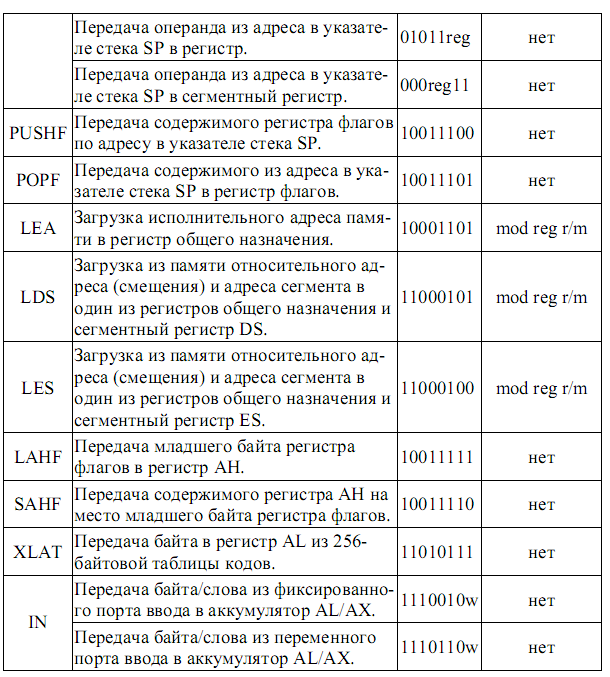
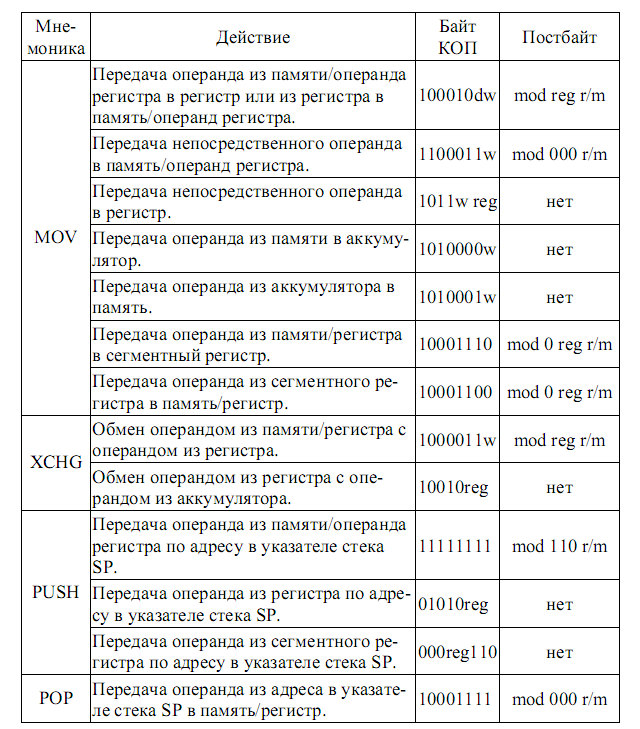
#### 2.2.6 Система команд

Команды передачи данных

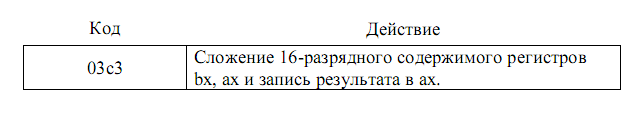
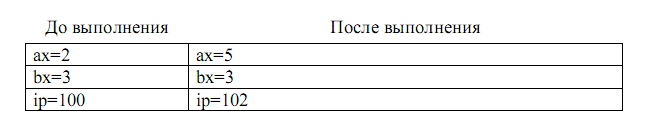
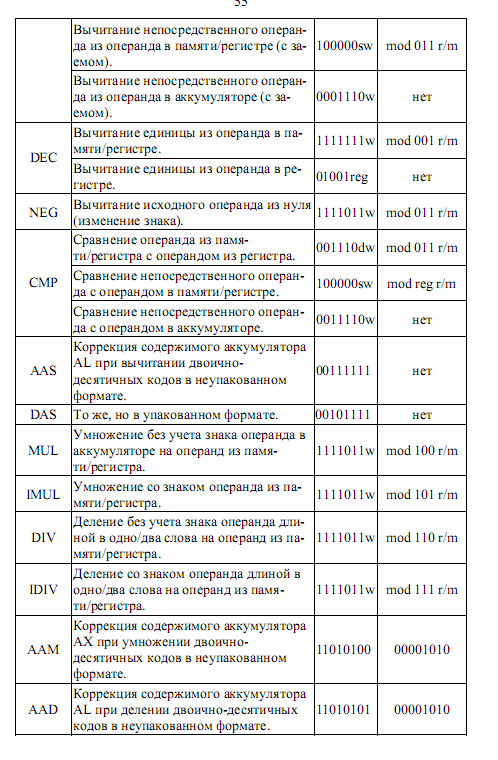
предназначены для пересылок данных между регистрами и памятью. Эту группу команд можно разделить на четыре подгруппы: команды пересылки данных общего назначения; команды, использующие аккумулятор; команды пересылки адресных объектов; команды пересылки флагов. Все команды этой группы не влияют на установку флагов, за исключением команд SAHF и POPF.

Пример: mov ax, bx





Арифметические команды

предназначены для выполнения четырех основных видов арифметических действий над 8- и 16- разрядными операндами в знаковом и беззнаковом представлении. Кроме основных арифметических команд имеются операции коррекции арифметических результатов для их перевода в упакованную или неупакованную форму. Признаки полученного результата отображаются в 6 битах регистра Flags (CF, AF, SF, ZF, PF и OF).

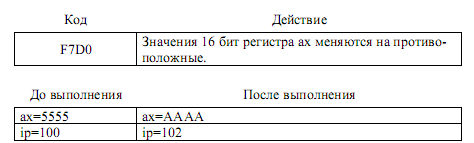
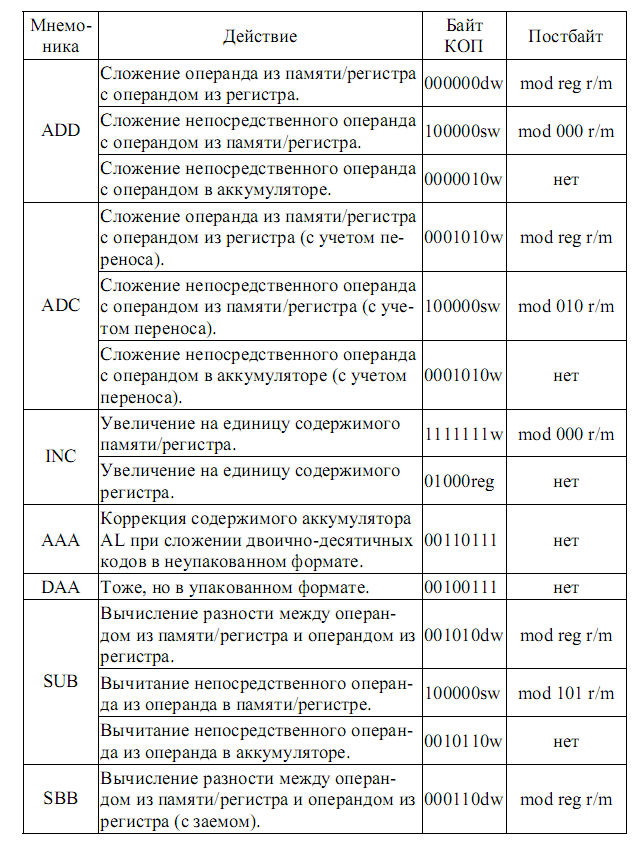
Пример: add ax, bx

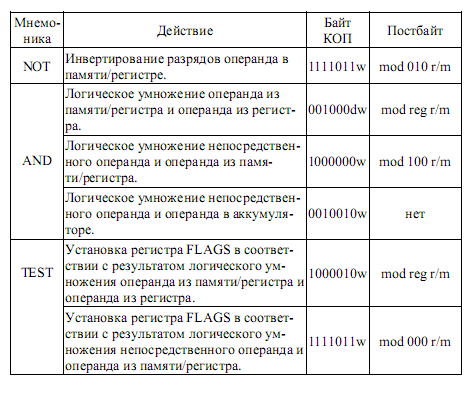
56

Логические команды

предназначены для выполнения четырех логических действий над 8- и 16битовыми логическими структурами: получение инверсного кода, логическое произведение, логическая сумма, сумма по модулю два. Команды AND, TEST, OR и XOR воздействуют на арифметические флаги следующим образом: флаги OF и CF всегда сбрасываются в нулевое состояние; состояния флагов SF, ZF, PF зависят от полученного результата и определяются по тем же правилам, что и в командах арифметических операций; состояние флага AF не определено. Команда NOT не влияет на состояние флагов.

Пример: not ax





Команды сдвига

предназначены для выполнения логических, арифметических и циклических 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 константа 1 (статический сдвиг) или как регистр CL. В первом случае осуществляется сдвиг на один байт, а во втором - число сдвигов определяется

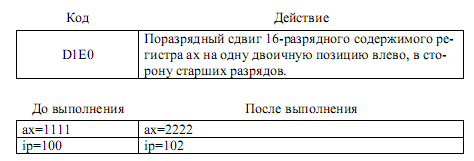
содержимым регистра CL, которое должно быть беззнаковым целым двоичным числом. Таким образом, число сдвигов можно задать переменной, вычисляемой во время выполнения программы (так называемый динами-

ческий сдвиг). При выполнении команд сдвигов флаги изменяются следующим образом:

• состояние флага AF всегда не определено;

• флаг CF всегда содержит значение последнего выдвинутого бита;

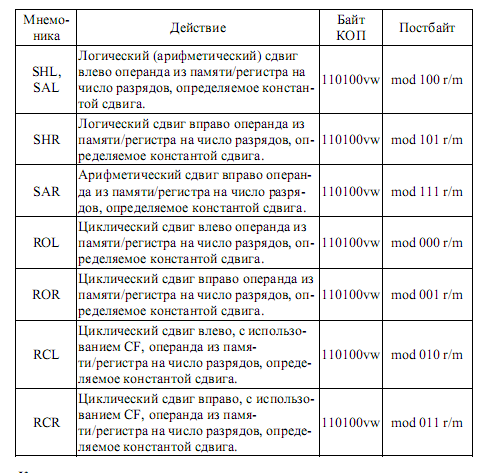
• в однобитных сдвигах флаг OF=0, если операция изменила значение старшего (знакового) бита операнда; при сдвиге на несколько бит состояние флага OF не определено;

• 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

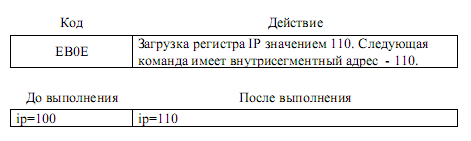


Команды передачи управления

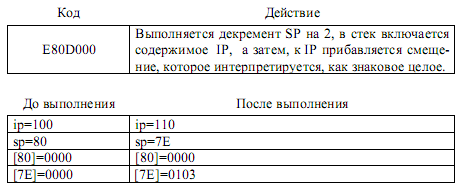
включают в себя безусловные переходы, команды обращения к подпрограммам и возврата из них, а также команды управления циклами. Выполнение этих команд приводит к изменению содержимого указателя команд

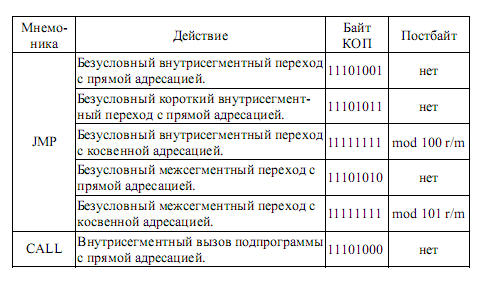
IP и регистра CS. Команды безусловных переходов и обращений к подпрограммам обеспечивают как внутрисегментные, так и межсегментные переходы с прямой и косвенной адресацией. Команды управления циклами

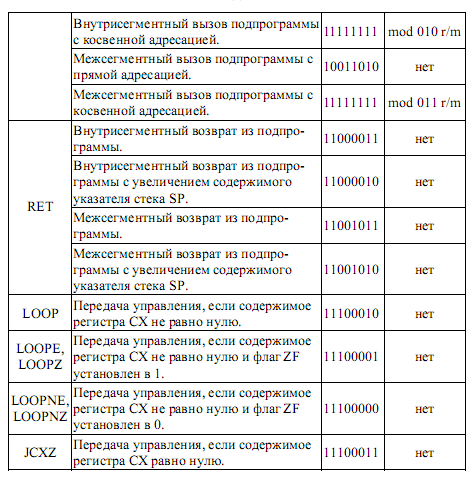
обеспечивают переход только в области текущего сегмента с коротким смещением.

Пример: jmp 110

Пример: call 110

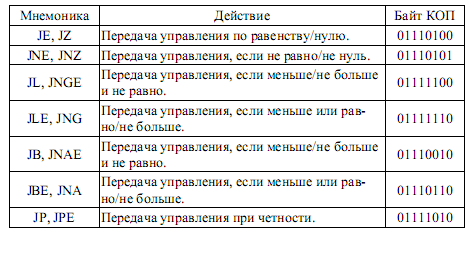


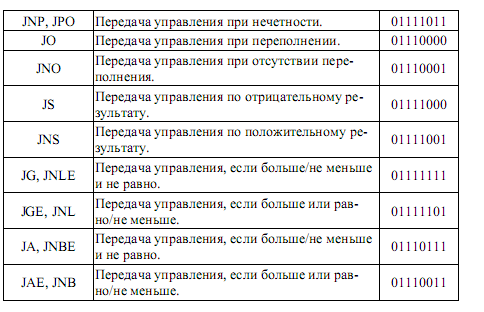




Команды условных переходов

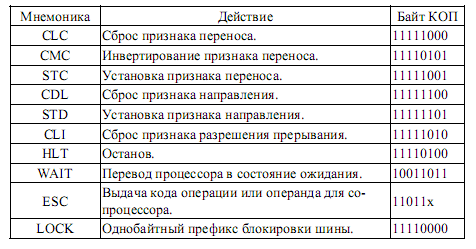
обеспечивают только внутрисегментные переходы.





Команды управления процессором

за исключением ESC являются однобайтовыми и могут быть разделены на две группы: команды, изменяющие содержимое регистра флагов, и команды, предназначенные для работы с внешними устройствами и не воздействующие на флаги.

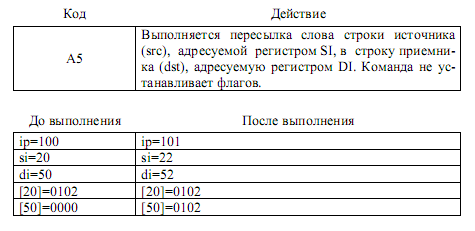


Команды обработки строк

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

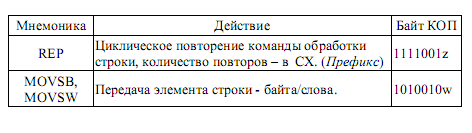
Код Действие

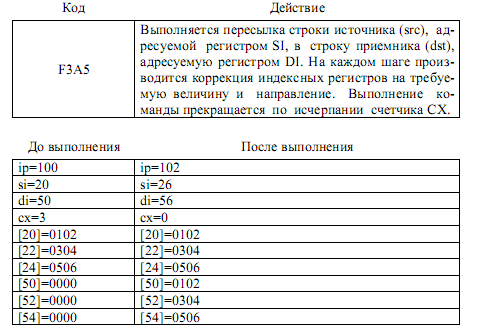
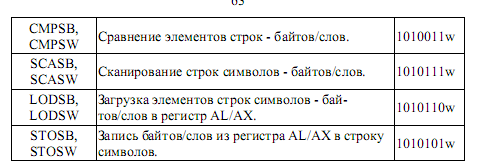
A5

Выполняется пересылка слова строки источника (src), адресуемой регистром SI, в строку приемни-

ка (dst), адресуемую регистром DI. Команда не устанавливает флагов.

Пример: rep movsw





#### 2.2.7 Математический сопроцессор

Математический сопроцессор (MCP - math coprocessor) – это расширение 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 двоично-десятичном представлении; М – мантисса, E – порядок вещественного числа, а S – знаковый разряд. Декодирование инструкций для сопроцессора и доставка данных осуществляется основным процессором, сопроцессор только исполняет свои команды. Для хранения операндов и промежуточных данных имеется восемь 80-разрядных регистров данных R0-R7, в которых данные представлены в расширенном вещественном формате (см. Рис. 2.9). При загрузке

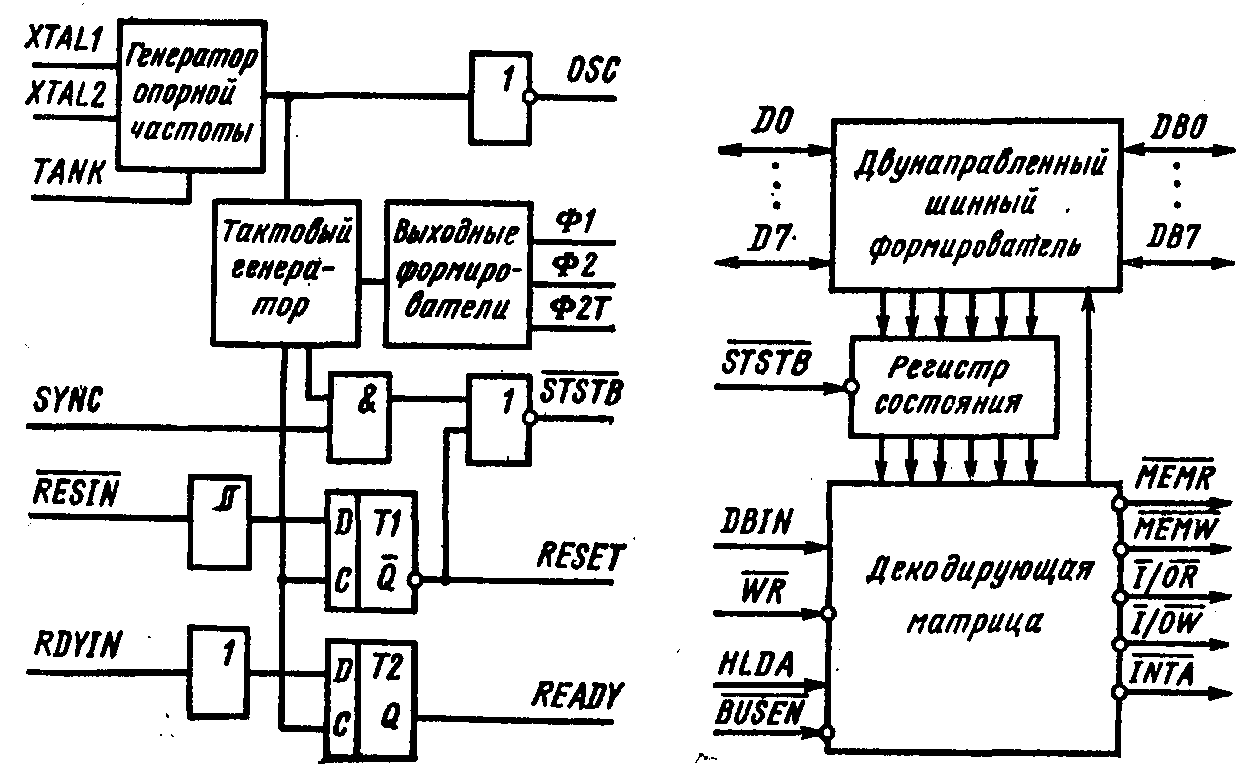
регистра из памяти, данные автоматически преобразуются в этот формат. Регистры данных образуют стек, обращение к которому возможно через относительные имена ST(N). Пять регистров специального назначения

служат для управления вычислениями и определения состояния сопроцессора (Рис. 2.10).

#### 2.2.8 Организация ввода-вывода

Работа с портами ввода/вывода в процессорах x86 возможна как с использованием специальных команд (IN, OUT), через отдельное адресное пространство ввода/вывода, так и по схеме с отображением регистров устройств на обычное адресное пространство. В последнем случае возможно использование обычных команд из системы команд процессора. Для адресации портов устройств в пространстве ввода/вывода, используется 16-разрядный адрес, обеспечивая доступ к 64K 8-битным портам с адресами от 0h до FFFFh. Адреса 0F8h – 0FFh– зарезервированы для системных 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 находится в команде) для портов с адресами 0h - 0F7h и с косвенной адресацией (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 тактовых импульсов И8224



Микросхема И8224 представляет собой генератор тактовых импульсов (ГТИ), предназначенный для совместной работы с ЦПП И8080A. Генератор формирует: высокоуровневые тактовые сигналы Ф1 и Ф2 о несовпадающими фазами; тактовый сигнал Ф2Т, по уровню совместим с ТТЛ и синхронизированный с сигналом Ф2; сигнал STSTB "Строб состояния", что, поступая на системный контроллер, фиксирует состояние шины данных микропроцессора; сигнал RESET "Установка".

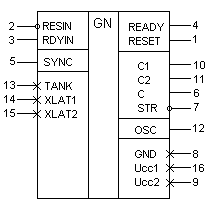
Генератор опорной частоты при подключении к выводам XTAL1 и XTAL2 кварцевые резонаторы обеспечивает высокую стабильность частоты, обусловленную основной частотой возбуждения кварцевого резонатора.

 Выход генератора опорной частоты выведен на внешний вывод OSC и соединен внутри микросхемы со счетчиком-делителем, который входит в состав тактового генератора. Тактовый генератор состоит из счетчика-делителя на 9, логических дешифраторов, формирующих необходимые тактовые импульсы, выходных формирователей и вспомогательных логических схем и триггеров для генерации выходных сигналов: Ф1.Ф2.Ф2Т, STSTB, Тактовые импульсы Ф1 и Ф2 управляют МОП-входами микропроцессора И8080А. Тактовый импульс Ф2Т используется для управления ТТЛ-входами в режиме прямого обращения к памяти.

Рисунок 2.4 Структурна схема І8224.

|  |  |  |
| --- | --- | --- |
| **Номер виводу** | **Позначення** | **Призначення** |
| 1 | RESET | Установка (выход) |
| 2 | RESIN | Установка (вход) |
| 3 | RDYIN | Готовность (вход) |
| 4 | READY | Готовность (выход) |
| 5 | SYNC | Синхронизация |
| 6 | Ф2Т | Фаза 2 с уровнем ТТЛ |
| 7 | STSTB | строб состояния |
| 8 | GND | Общий |
| 9 | Ucci | +12 В |
| 10, 11 | Ф2,Ф1 | Фаза 2, 1 |
| 12 | OSC | Выход осцилятора |
| 13 | TANK | Вход колебательного контура |
| 14, 15 | XTAL2, XTAL1 | Кварцовий резонатор |
| 16 | Ucc | +5В |

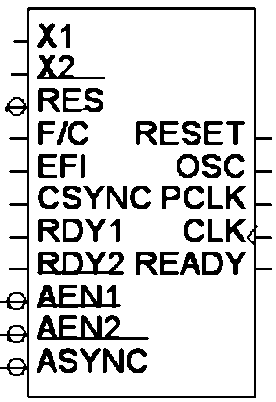
Таблица 2.1- Назначение выводов И8224.



Негативный сигнал STSTB, продолжительность которого равна от - ном периоду частоты опорного генератора, формируется микросхемой И8224 при поступлении на ее вход с микропроцессора И8080А сигнала SYNC "Синхронизация", что свидетельствует о начале машинного циклу.Пры поступлении входного сигнала RESIN микросхема И8224 с помощью триггера Шмитта и триггера Т1 вырабатывает сигнал RESET, синхронизированный с тактовым сигналом Ф2, по сигналу RESET осуществляется установка в исходное состояние различных устройств микропроцессорной системы. Наличие в микросхеме триггера Шмитта позволяет подавать на вход RESIN сигнал с пологим фронтом. С помощью триггера Т2 осуществляется стробирование входного сигнала RDYIN "Готовность" тактовым сигналом Ф2.

Рисунок 2.5-інтегральне виконання ІМС І8224

**2.2.1 Генератор синхронизации 8284**

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

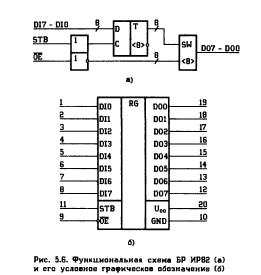
**Рис. 3.1. Внешний генератор 8284 для систем на основе микропросессора 8086**

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 буферный регистр

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.

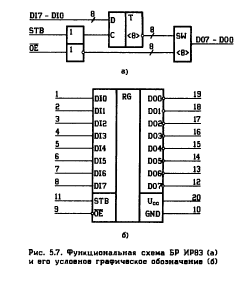
Микропроцессор генерирует два сигнала управления **DEN** и **DT/R** для управления двунаправленными буферами или

Рисунок 2.6- Структурная схема ИМС 18282

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.

Микросхема И 8282/83 (КР580ИР82 / 83) представляет собой 8-разрядный буферный регистр, предназначенный для фиксации информации и может использоваться в

Рисунок 2.7 Структурная схема ИМС И8283



системах, построенных на микросхемах различных серий. В системах с процессором И8080А регистр может использоваться для хранения слова состояния процессора, а в системах с И8086 - для выделения адреса, которая поступает по мультиплексированных шине адреса данных

Структурная схема микросхемы И8282 (КР580ИР82) представлена ​​на рис.2.6.

Если на входе / ОЕ поступает сигнал разрешения низкого уровня, а на входе STB - сигнал высокого уровня, то информация из входов микросхемы передается на выходы. После перехода сигнала на входе STB с высокого уровня на низкий информация, записанная в буферного регистра, сохраняется до появления сигнала разрешения на входе STB. Сигнал высокого уровня на входе / ОЕ переводе выводы DO0-DO7 в высокоимпедансное состояние.

**Электрические параметры регистра**:

1. U пит. (Напряжение питания) - 5 В

2. Выходное напряжение питания низкого уровня (U вых низ г) <0.45 В

3. Выходное напряжение питания высокого уровня (U вых выс г.):> 2.4 В

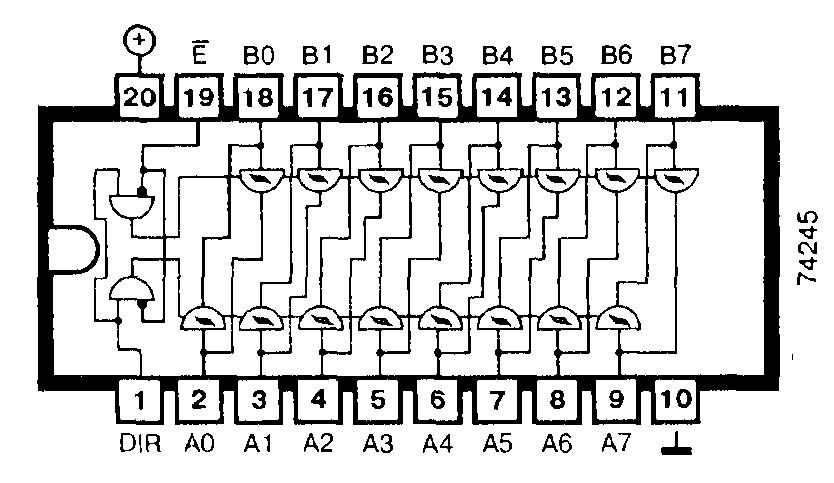
4. t-повтора (Время задержки распространения информационного сигнала на выходе относительно информационного сигнала на входе <30 нс

Таблица 2..2- Назначение выводов И8282 / 83

|  |  |  |  |
| --- | --- | --- | --- |
| Номер виводу | Позначення | Тип виводу | Функціональне  значення |
| 1-8 | DI0—DI7 | Входи регистра | Сигналы, подаваемые на выводы 1-8, поступающих на входе D триггеров буферного регистра |
| 9 | /ОЕ | Вход | Разрешение выдачи содержания буферного регистра. Сигнал низкого уровня на входе 9 разрешает выдачу на выводы DO0-DО7 содержания буферного регистра, а сигнал высокого уровня переводе эти выводы в високоимпенансний состояние |
| 10 | GND |  | Общий |
| 11 | STB | Вход | Строб. По сигналу высокого уровня, который поступает на вывод 11 информация, подается на входы DI0-DI7, записывается в триггеры буфера |
| 19—12 | DO0-DО7  (І8282)  / DO0-/ DО7  (І8283) | Выходы из высокоимпедансным состоянию | На 19-12 выводится состояние триггеров буферного регистра, если на вход / ОЭ подан сигнал разрешения, в противном случае выводы 19-12 находятся в высокоимпедансное состоянии |
| 20 | Ucc | +5 В | Питание |

### 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

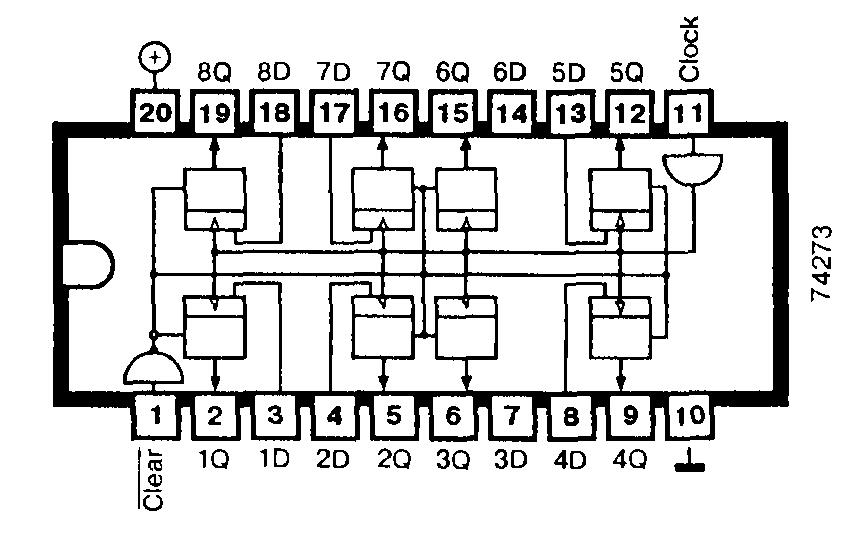
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **Тип микросхемы** | **74ALS245** | **74AS245** | **74F245** | **74LS245** |
| Время задержки прохождения сигнала, нс | 6,6 | 5,5 | 3,8 | 8 |
| Ток потребления, мА | 35 | 59 | 100 | 62 |

### 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.

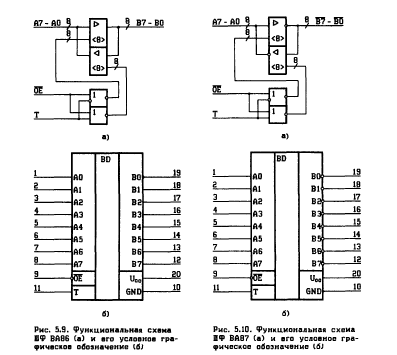
### Технические данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| **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 Двунаправленный шинный формирователь И8286 / 87

Шинный формирователь И8286 / 87 предназначен в первую очередь для использования в микропроцессорных системах, построенных на БИС серии И8080А и И8086, но он может использоваться и в других системах совместно с процессорами других серий.

Шинный формирователь предназначен для увеличения нагрузочной способности выводов микропроцессорных БИС. Кроме этого, он обеспечивает отключение модулей системы в нужное время общей шины.

Рисунок 2.8 Шинный формирователь

Восьмиразрядный шинный формирователь в зависимости от сигналов на входах Т и / ОЭ (когда Т высокий и / ОЕ-низкий) обеспечивается подключние входов А0-А7 к выходам В0-В7, в противном случае (если на Т-низкий)-вход В0- В7 подключаются к выходам А0-А7. Сигнал высокого уровня на входе / ОЕ переводит выводы В0-В7 (если на Т-высокий) или выводы А0-А7 (если на Т-низкий) в высокоимпедансное состояние.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер виводу | Позначення | Тип виводу | Функціональне  значення |
| 1-8 | А0—А7 | Входы вызод с высоко-импедансным состоянию | На выводы 1-8 могут поступать сигналы, или сниматься в зависимости от сигнала на входе  Таблица 2.3- Назначение выводов И8286 / 87 Т |
| 9 | /ОЕ | Вхід | Разрешение выхода. Когда на вывод 9 поступает сигнал высокого уровня в зависимости от сигнала на входе Т выводы А0-А7 или В0-В7 находятся в високоимпенансному состоянии |
| 10 | GND |  | Общий |
| 11 | T | Вхід | Направление передачи информации. Если на вход Т подается сигнал высокого уровня, то формирователь настраивается на передачу от входов А0-А7 к выводам В0-В7, в противном случае, если на вход Т подается сигнал низкого уровня, то информация передается от выводов В0-В7 к выводам А0- А7 |
| 19—12 | В0-В7  (І8286)  /В0-/В7  (І8287) | Виходи з висо-коімпедансним станом | На выводы 19-12 могут вводиться или вести себя в зависимости от сигнала на входе Т состояние (в И8287 сигналы инвертируются) |
| 20 | Ucc | +5 В | Питание |

# 2.5 Системный контроллер И8228 / 38 (КР580ВК38)

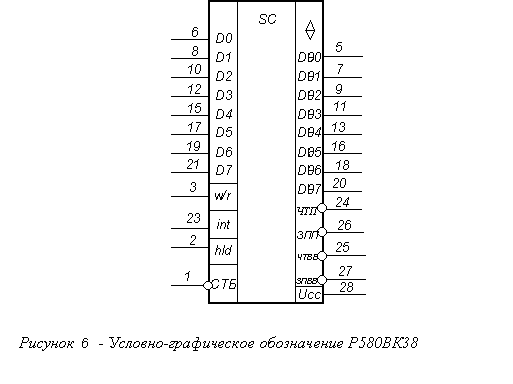
Микросхема И8228 / 38 выполняет функцию системного контроллера и шинного формирователя, осуществляет формирование управляющих сигналов обращения к ОЗУ или к устройствам ввода / вывода (ПВВ) и обеспечивает прием и передачу 8-разрядной информации между шиной данных микропроцессора и системной шиной.

Формирования сигналов I / OW, MEMW в данной микросхеме происходит по сигналу STSTB "Строб состояния", что позволяет при применении в микропроцессорной системе микросхемы И8238 использовать ЗП и ПВВ с более широким диапазоном производительности. . Двунаправленный шины формирователь осуществляет буферирування 8-разрядной шины данных и автоматический контроль направления передачи данных.

Подключение системного контроллера к шине данных микропроцессора осуществляется с помощью двунаправленных выводов DO-D7, к системному с помощью двунаправленных выводов DBO-DB7. При необходимости с помощью сигнала BUSEN ​​"Управление системной шиной" выводы DBO-DB7 системного контроллера могут быть переведены в состояние "Выключено".

Таблица 2.4 Назначение выводов ИМС И8228 / 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 |



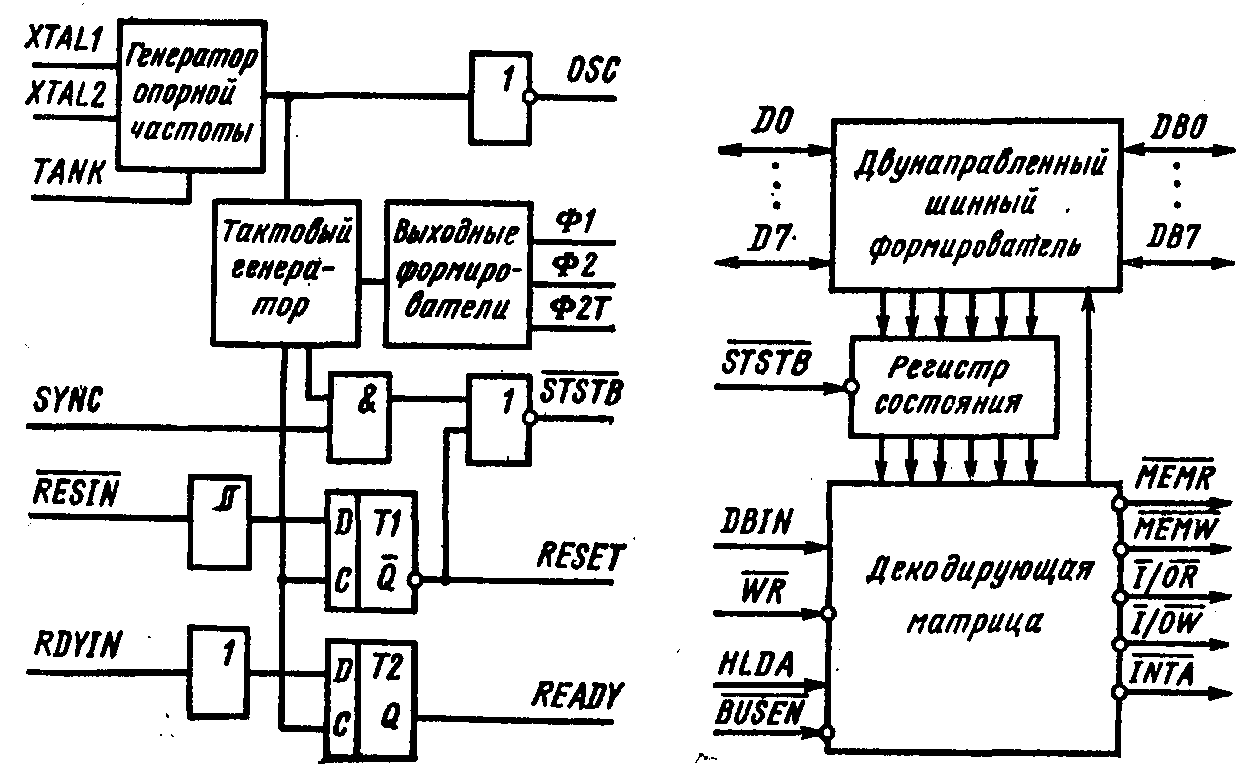
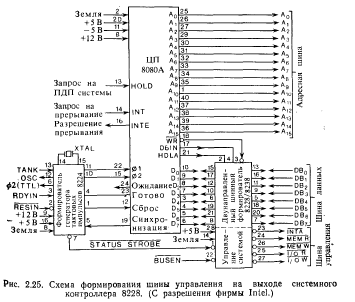
Регистр состояния выполнен на шести D-триггерах и предназначен для хранения информации о состоянии микропроцессора, поступающей по шине данных DO-D7. Запись в регистр состояния осуществляется по сигналу STSTB, поступающего в начале каждого машинного цикла. Декодирующая матрица в зависимости от режима работы процессора, зафиксированного в регистре состояния, и входных управляющих сигналов HLDA, WR, DBIN формирует сигнал INTA "Подтверждение прерывания" или сигналы чтения / записи при обращении к ОЗУ или ПВВ.

Рисунок 2.8- Интегральное исполнение ИМС 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 ВМ. Между тем любое увеличение числа проводников ВМ 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 ЭМ 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**.**

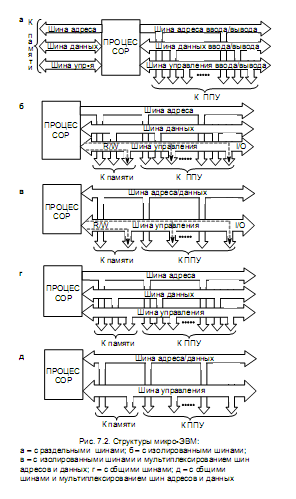
В ряде случаев это является преимуществом, однако при возникновении определенных сбоев в работе ПУ и их некорректной обработки со стороны операционной системы возможны «зависания» вычислительного процесса.

***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 i8080А

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.

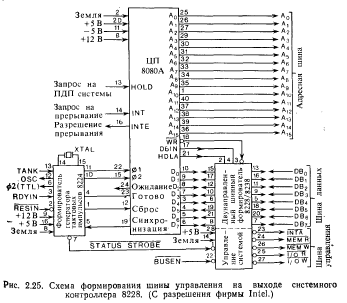


Рисунок 3.1

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.

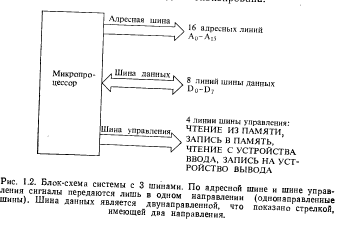
 Указанные пять возможных типов функций микропроцессорной системы позволяют создавать большое количество различных средств.

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 i8086 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 Пам’ять мікропроцесорної системи

# 4.1 Різноманітності пам’яті

Основні характеристики запам'ятовуючих пристроїв

Запам'ятовуючі пристрої (ЗП) характеризуються рядом параметрів, що визначають можливі області застосування різних типів таких пристроїв. До основних параметрів, за якими проводиться найбільш загальна оцінка ЗУ, належать їх інформаційна ємність (E), час звернення (T) і вартість (C).

Під інформаційною ємністю ЗУ розуміють кількість інформації, що вимірюється в байтах, кілобайтах, мегабайтах або гігабайтах, яке може зберігатися в запам'ятовуючому пристрої.

Як відомо, приставки кіло-, мега-і гіга-допускають неоднозначне трактування у зв'язку з відмінностями їхнього розуміння в загальнонауковому і специфічному при використанні двійкової системи числення сенсах. Так, в загальному сенсі приставка "кіло" відповідає 103, "мега" - 106, а "гіга" - 109 (на підході "тера", "пента" і "гекса"). У той же час, близькі за звучанням і змістом двійкові аналоги цих величин: К-, М-і Г-позначають 210 (1024), 220 (1048576) і 230 (1073741824), що тільки приблизно відповідає перерахованим вище ступенями 10. Тому при вказівці ємності одного і того ж пристрою пам'яті,

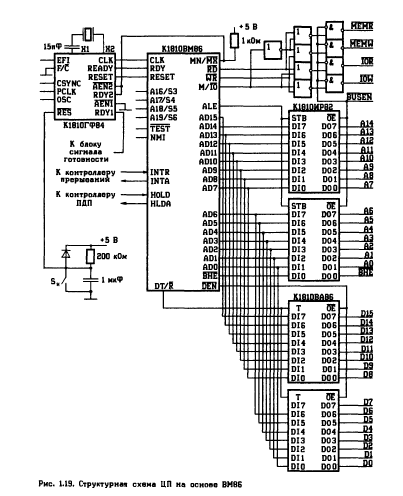


Рисунок 3.3-Структурна схема ЕОМ на І8086А з шиною адреси даних та керування

наприклад жорсткого диска, в Гбайт і мільярдах байт, можуть спостерігатися певні відмінності.

Зазвичай інформаційна ємність враховує тільки корисний об'єм інформації, який не включає обсяг пам'яті, витрачуваний на службову інформацію, контрольні розряди або байти, резервні області (наприклад, інтервал між кінцем доріжки диска і її початком), доріжки синхросигналів і пр.

Час звернення до ЗП різних типів визначається по-різному. Як приклад можна розглянути оперативні ЗП і жорсткі диски.

Оперативні ЗП зазвичай реалізуються як ЗУ з довільним доступом. Це означає, що доступ до даних, фізично організованим у вигляді двовимірного масиву (матриці елементів пам'яті), проводиться за допомогою схем дешифрування, що вибирають потрібні рядок і стовпець масиву по їх номерами (адресами). Тому час Tобр звернення до них визначається, в разі відсутності додаткових етапів (таких, наприклад, як передача адреси за два такти), часом спрацьовування схем дешифрування адреси і власне часом запису або зчитування даних.

# 4.2. Класифікація запам’ятовуючих пристроїв

В даний час існує велика кількість різних типів ЗУ, використовуваних в ЕОМ і системах. Ці пристрої розрізняються рядом ознак: принципом дії, логічною організацією, конструктивної і технологічної реалізацією, функціональним призначенням і т.д. Велика кількість існуючих типів ЗУ обумовлює відмінності у структурній і логічній організації (систем) пам'яті ЕОМ. Необхідні характеристики пам'яті досягаються не тільки за рахунок застосування ЗУ з відповідними характеристиками, але в значній мірі за рахунок особливостей її структури та алгоритмів функціонування.

Пам'ять ЕОМ майже завжди є "вузьким місцем", що обмежує продуктивність комп'ютера. Тому в її організації використовується ряд прийомів, що поліпшують тимчасові характеристики пам'яті і, отже, підвищують продуктивність ЕОМ в цілому.

Класифікація запам'ятовуючих пристроїв і систем пам'яті дозволяє виділити загальні і характерні особливості їх організації, систематизувати базові принципи і методи, покладені в основу їх реалізації та використання.

Один з можливих варіантів класифікації ЗУ представлений на рис.3. У ньому пристрої пам'яті поділяються за двома основними критеріями: за функціональним призначенням (ролі або місця в ієрархії пам'яті) і принципу організації.

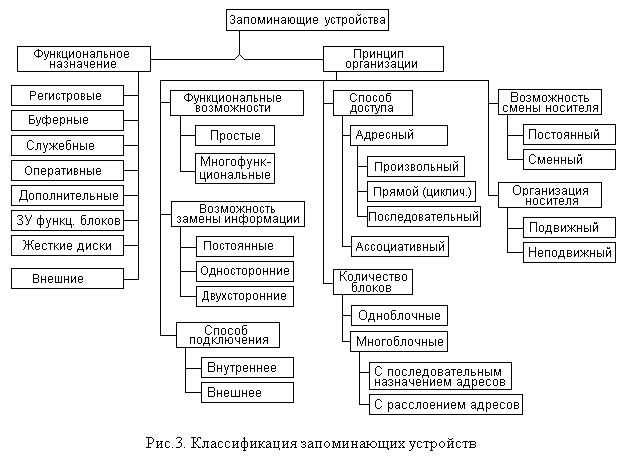


Рисунок 4.1

# 4.2.1. Класифікація ЗП по функціональному призначенню

При поділі ЗУ за функціональним призначенням іноді розглядають два класи: внутрішні та зовнішні ЗУ ЕОМ. Такий поділ спочатку грунтувався на різному конструктивному розташуванні їх в ЕОМ. В даний час, наприклад, накопичувачі на жорстких магнітних дисках, які традиційно відносять до зовнішніх ЗУ, конструктивно розташовуються безпосередньо в основному блоці комп'ютера. Тому поділ на зовнішні і внутрішні ЗУ має в ряді випадків відносний, умовний характер. Зазвичай до внутрішніх ЗУ відносять пристрої, безпосередньо доступні процесору, а до зовнішніх - такі, обмін інформацією яких з процесором відбувається через внутрішні ЗУ.

Загальний вид ієрархії пам'яті ЕОМ представлений на рис.4.2 На ньому показані різні типи ЗУ, причому оскільки малюнок узагальнений, то не всі з представлених на ньому ЗУ обов'язково входять до складу ЕОМ, а характер зв'язків між пристроями може відрізнятися від показаного на рисунку.

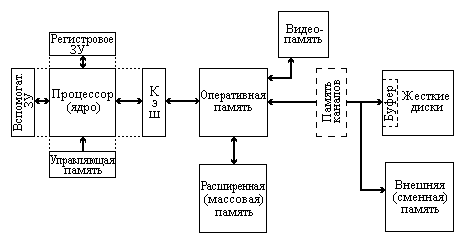


Рисунок 4.2- Можливий склад системи пам’яті ЕОМ

1. Верхнє місце в ієрархії пам'яті займають регістрові ЗУ, які входять до складу процесора і часто розглядаються не як самостійний блок ЗУ, а просто як набір регістрів процесора. Такі ЗУ в більшості випадків реалізовані на тому ж кристалі, що і процесор, і призначені для зберігання невеликої кількості інформації (до декількох десятків слів, а в RISC-архітектурах - до сотні), яка обробляється в поточний момент часу або часто використовується процесором. Це дозволяє скоротити час виконання програми за рахунок використання команд типу регістр-регістр і зменшити частоту обмінів інформацією з більш повільними ЗУ ЕОМ. Звернення до цих ЗУ проводиться безпосередньо по командам процесора.

2. Наступну позицію в ієрархії займають буферні ЗУ. Їх призначення полягає в скороченні часу передачі інформації між процесором і більш повільними рівнями пам'яті комп'ютера. Буферна пам'ять може встановлюватися на різних рівнях, але тут мова йде саме про зазначене її місце розташування. Раніше такі буферні ЗУ у вітчизняній літературі називали сверхоператівних, зараз ця назва практично повністю витіснив термін "кеш-пам'ять" або просто кеш.

Принцип використання буферної пам'яті у всіх випадках зводиться до одного й того ж. Буфер являє собою більш швидке (а значить, і більш дороге), але менш ємне ЗУ, ніж те, для прискорення роботи якого він призначений. При цьому в буфері розміщується тільки та частина інформації з більш повільного ЗУ, яка використовується в даний момент.

Обмін інформацією між кеш-пам'яттю і більш повільними ЗУ для поліпшення тимчасових характеристик виконується блоками, а не байтами або словами. Керують цим обміном апаратні засоби процесора і операційна система, і втручання прикладної програми не потрібно. Причому безпосередньо командам процесора кеш-пам'ять недоступна, тобто програма не може явно вказати читання або запис в кеш-пам'яті, яка є для неї, як іноді кажуть, "прозорої" (прямий переклад використовуваного в англомовній літературі слова transparent).

**3. Наступним рівнем ієрархії пам'яті є оперативна пам'ять. Оперативне ЗУ (ОЗУ) є основним запам'ятовуючим пристроєм ЕОМ, в якому зберігаються виконуються в даний момент процесором програми і оброблювані дані, резидентні програми, модулі операційної системи і т.п. Назва оперативної пам'яті також кілька змінювалося в часі. У деяких родинах ЕОМ її називали основною пам'яттю, основною оперативною пам'яттю і пр. В англомовній літературі також використовується термін RAM (random access memory), що означає пам'ять з довільним доступом.**

**Ця пам'ять використовується в якості основного запам'ятовуючого пристрою ЕОМ для зберігання програм, які виконуються або готових до виконання в поточний момент часу, та належних до них даних. В оперативній пам'яті розташовуються і компоненти операційної системи, необхідні для її нормальної роботи. Інформація, що знаходиться в ОЗУ, безпосередньо доступна командам процесора, за умови дотримання вимог захисту.**

**Оперативна пам'ять реалізується на напівпровідниках (інтегральних схемах), стандартні обсяги її складають (на початку 2000-х років) сотні мегабайт - одиниці гігабайт, а часи звернення - одиниці ÷ десятки наносекунд.**

**4. Ще одним рівнем ієрархії ЗУ може бути додаткова пам'ять, яку іноді називали розширеної чи масовою. Спочатку (1970-ті роки) ця щабель використовувалася для нарощування ємності оперативної пам'яті до величини, що відповідає адресного простору (наприклад, 24-бітного адреси) команд, за допомогою підключення більш дешевого і ємного, ніж ОЗУ, що запам'ятовує.**

**Потім, в ранніх моделях ПЕОМ, додаткова пам'ять також використовувалася для нарощування ємності ОЗУ і представляла собою окрему плату з мікросхемами пам'яті. А ще пізніше термін додаткова пам'ять (extended або expanded memory) став позначати область оперативного ЗУ з адресами вище одного мегабайта. Звичайно, цей термін застосовний тільки до IBM PC сумісним ПЕОМ.**

**5. До складу пам'яті ЕОМ входять також ЗУ, що належать окремим функціональним блокам комп'ютера. Формально ці пристрої безпосередньо не обслуговують основні потоки даних і команд, що проходять через процесор. Їх призначення зазвичай зводиться до буферизації даних, які з будь-яких пристроїв і вступників у них.**

**Типовим прикладом такої пам'яті є відеопам'ять графічного адаптера, яка використовується як буферної пам'яті для зниження навантаження на основну пам'ять і системну шину процесора.**

**Іншими прикладами таких пристроїв можуть служити буферна пам'ять контролерів жорстких дисків, а також пам'ять, що використовувалась в каналах (процесорах) вводу-виводу для організації одночасної роботи декількох зовнішніх пристроїв.**

**Ємності і швидкодія цих видів пам'яті залежать від конкретного функціонального призначення обслуговуваних ними пристроїв. Для відеопам'яті, наприклад, обсяг може досягати величин, порівнянних з оперативними ЗУ, а швидкодія - навіть перевершувати швидкодію останніх.**

**6. Наступною сходинкою пам'яті, яка стала фактично стандартом для будь-яких ЕОМ, є жорсткі диски. У цих ЗУ зберігається практично вся інформація, яка використовується більш-менш активно, починаючи від операційної системи та основних прикладних програм і закінчуючи рідко використовуваними пакетами і довідковими даними.**

**Ємність цьому ступені пам'яті, яка може включати в свій склад до десятків дисків, забезпечуючи зберігання дуже великої кількості даних, залежить від області застосування ЕОМ. Типова місткість жорсткого диска, складова на початок 2000-х років десятки гігабайт, подвоюється приблизно кожні півтора року.**

**З часом обігу справа йде трохи інакше: компоненти цього часу, обумовлені переміщенням блоку головок читання-запису зменшуються порівняно повільно (приблизно вдвічі за 10 років). Компонента, обумовлена часом підведення сектора і залежна від швидкості обертання шпинделя диска, також зменшується з ростом цієї швидкості приблизно такими ж темпами. А швидкість передачі даних зростає значно швидше, що пов'язано зі збільшенням щільності запису інформації на диски.**

**7. Всі інші пристрої, що запам'ятовують можна об'єднати з точки зору функціонального призначення в одну загальну групу, охарактеризувавши її як групу зовнішніх ЗУ. Під словом "зовнішні" слід розуміти те, що інформація, що зберігається в цих ЗУ, в загальному випадку розташована на носіях не є частиною власне ЕОМ. Під це визначення підпадають гнучкі диски, компакт диски, накопичувачі на змінних магнітних дисках і магнітооптичні диски, твердотільні (флеш) диски і флеш-карти, стримери, зовнішні вінчестери та ін Природно, що параметри цих пристроїв досить різні. Функціональне призначення їх зазвичай зводиться або до архівного зберігання інформації, або до переносу її од одного комп'ютера до іншого.**

**Деякі сумніви у приналежності до даної категорії можуть викликати змінні диски, що встановлюються в санчата (rack). Такі диски, дійсно, краще віднести до попередньої (сьомий) групі.**

# 4.2.2. Класифікація ЗП по принципу організації

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

1. За функціональним можливостям ЗУ можна розділяти:

- На прості, що допускають тільки зберігання інформації;

- Багатофункціональні, які дозволяють не тільки зберігати, але й переробляти збережену інформацію без участі процесора безпосередньо в самих ЗП.

Підхід, який використовується у другій групі ЗУ, в принципі, дозволяє створити продуктивні системи з паралельною обробкою даних. Зокрема, схожі підходи використовуються в різних частинах відеотракту комп'ютера.

-По можливості зміни інформації розрізняють ЗП:

- Постійні (або з однократним записом);

- Односторонні (з перезаписом або перепрограмовані);

- Двосторонні.

У постійних ЗП (ПЗП) інформація заноситься або при виготовленні, або за допомогою запису (або, як інакше називають цю процедуру, програмування або пропалювання), яка може бути виконана тільки одноразово. В ході такого запису змінюється сам носій інформації, наприклад, перепалюється провідники в мікросхемах ПЗУ або формуються лунки в відбиває шарі CD-ROM.

Односторонніми називають ЗП, які мають суттєво різні часи запису та зчитування інформації. Найбільш поширеними типами таких ЗУ є перепрограмовані постійні ЗУ або компакт-диски з перезаписом - CD-RW. Час запису в пристроях цих типів значно перевищує час зчитування інформації.

До одностороннім ЗУ можна віднести і ЗУ на приладах з зарядовим зв'язком (ПЗС), в яких час запису (формування зображення), взагалі кажучи, помітно менше часу зчитування (передачі зображення).

Двосторонні ЗП мають близькі значення часів читання і запису. Типовими представниками таких ЗП є оперативні ЗП і ЗУ на жорстких дисках.

**2. За способом доступу розрізняють ЗП:**

**- З адресним доступом;**

**- З асоціативним доступом.**

**При адресному доступі для запису або читання місце розташування інформації в ЗУ визначається її адресою. Логічно адреса може мати різну структуру. Наприклад, в оперативних ЗП адреса являє собою двійковий код, одна частина розрядів якого вказують рядок матриці елементів пам'яті, а інша - стовпець цієї матриці. На перетині заданих рядка і стовпця знаходиться шукана інформація. В ЗУ на магнітних дисках адреса може представляти собою або комбінацію номерів циліндра, головки і сектора (так звана CHS-геометрія), або логічний номер сектора (LBA-адресація). Можливі й інші варіанти.**

**У кожному разі, заданий адресу відпрацьовується схемами доступу ЗУ (дешифратором, блоком позиціонування головок і т.п.) таким чином, що в операції бере участь відповідна адресою область матриці елементів пам'яті, пам'ятною середовища або носія інформації.При цьому, залежно від того, як саме спрацьовує механізм доступу, розрізняють такі види адресного доступу:**

**- Довільний;**

**- Прямий (циклічний);**

**- Послідовний.**

**Термін "пам'ять з довільним доступом" (random access memory - RAM) застосовують до ЗУ, в яких вибір місця зберігання інформації проводиться безпосереднім підключенням входів і виходів елементів пам'яті (через буфери, підсилювачі й логічні елементи) до вхідних і вихідних шинам ЗП. Це найбільш швидкий вид адресного доступу, вживаний в оперативних ЗП і кеш-пам'яті.**

**При прямому (циклічному) доступі безпосередній комутації зв'язків виявляється недостатньо. В таких ЗП зазвичай відбувається ще й переміщення даних щодо механізму читання / запису, механізму читання / запису щодо даних або і те й інше. Фізично це може бути як механічне переміщення, наприклад, в жорстких дисках, переміщення областей намагніченості, як в ЗУ на магнітних доменах, перенесення зарядів і ін..**

**З логічної точки зору такі ЗП можна зіставити набору зсувних регістрів, інформація в яких зсувається циклічно і може вводитися в регістр або виводитися з нього тільки в одному з розрядів. Терміни "циклічний" і "прямий" доступ близькі за змістом, хоча "прямий доступ" - має більш широкий зміст.**

**Послідовний доступ характерний для ЗП, що використовують як носія інформації (пам'ятною середовища) магнітну стрічку, наприклад, для стримерів. В таких ЗУ для доступу до блоку даних необхідно перемістити носій так, щоб ділянка, на якій розташовується необхідний блок даних, опинився під блоком головок читання / запису.**

**Крім того, при всіх формах адресного доступу адресуються елементом може бути не тільки байт або слово (як в оперативній пам'яті і кеш-пам'яті), але цілий блок даних. Це звичайно пов'язано або з конструктивними особливостями ЗП, або з великим часом доступу.**

**При асоціативному доступі місце зберігання інформації при читанні і запису визначається не адресою, а значенням деякого ключа пошуку. Кожне записане і збережене в асоціативної пам'яті слово має поле ключа. Значення цього ключа порівнюється зі значенням ключа пошуку при читанні даних з пам'яті. У разі збігу порівнюваних значень інформація зчитується з пам'яті.**

Асоціативна пам'ять ефективна для вирішення завдань, пов'язаних з пошуком даних. Проте її використання обмежене в силу порівняно високою її складності.

Дійсно, з апаратної точки зору сам пошук може бути організований по-різному: послідовно за розрядами ключових полів або паралельно за всіма ключам в усьому масиві пам'яті. Другий спосіб, звичайно, більш швидкий, але вимагає відповідної організації (ключовий частини) пам'яті, яка повинна мати для цього в ключовий частини кожного зберігається слова схеми порівняння. Саме тому така пам'ять істотно дорожча, ніж оперативна, і використовується в основному для вирішення завдань, що вимагають швидкого пошуку в невеликих обсягах інформації.

Одним з частих застосувань асоціативної пам'яті є швидке перетворення логічних (лінійних) адрес даних в фізичні (тобто адреси елементів пам'яті), що виконується, наприклад, так званим буфером трансляції адрес. Інший близькою завданням є визначення того, чи є необхідна інформація у верхніх рівнях ЗП або необхідна її підкачка з більш повільних ЗП.

# 4.3 Напівпровідникові запам’ятовуючі пристрої

Напівпровідникові ЗП в даний час представляють собою великий клас запам'ятовуючих пристроїв, різних за своїм функціональним і технічним характеристикам, широко використовуються як внутрішніх ЗП ЕОМ. Але цим їх використання не обмежується. Переважна більшість електронної та побутової техніки переходить на цифрові методи представлення даних (не тільки текстових, а й аудіо, графічних і відео) і управління (використання мікроконтролерів).

Різні сфери застосування накладають свої особливості на реалізацію напівпровідникових ЗП, однак це частіше стосується їх конструктивних особливостей, а принципи побудови однакові.

Висока швидкодія напівпровідникових ЗП обумовлює те, що більшість з них має організацію з довільним доступом. Хоча такі ЗУ, як флеш-пам'ять і ЗУ з переносом зарядів (використовувані, наприклад, у фото-і відеокамерах), організовані трохи інакше.

Це ж висока швидкодія визначає і основні сфери застосування напівпровідникових ЗП в ЕОМ: кеш-пам'ять і оперативна пам'ять.

Причому треба відзначити, що термін "ЗП з довільним доступом" (Random Access Memory - RAM) не відповідає в точності терміну "оперативна пам'ять", оскільки перший з них вказує на спосіб доступу, а другий - на функціональне призначення. І дійсно, кеш-пам'ять і постійні ЗУ також є ЗУ з довільним доступом. Однак, відповідно до прийнятої в російськомовній літературі термінологією, термін "оперативні ЗУ" нижче іноді використовується як синонім ЗП з довільним доступом.

У цьому розділі розглядаються основні види напівпровідникових ЗУ: статична і динамічна пам'ять з довільним доступом, постійна і флеш-пам'ять, а також наводяться деякі відомості по іншим видам пам'яті.

# 4.4 Постійні запам’ятовуючі пристрої

Постійні запам'ятовуючі пристрої (ПЗП або Read Only Memory - ROM), які також часто називають енергонезалежними (або Non Volatile Storage), забезпечують збереження записаної в них інформації та при відсутності напруги живлення. Звичайно, під таке визначення підпадають і пам'ять на жорстких і гнучких дисках, і компакт диски, і деякі інші види ЗУ.

Однак, говорячи про постійні ЗУ, зазвичай мають на увазі пристрої пам'яті з довільним адресним доступом. Такі ЗУ можуть будуватися на різних фізичних принципах і володіти різними характеристиками не тільки по місткості і часу звернення до них, а й по можливості заміни записаної в них інформації.

# 4.4.1. Різновиди постійних ЗП

На початок 2000-х років найбільшого поширення набули напівпровідникові ПЗУ, елементи пам'яті яких використовують різні модифікації діодів і транзисторів і виготовляються за інтегральною технологією.

Безпосередніми попередниками таких ЗП були магнітні (трансформаторні) ПЗП, інформація в які записувалася відповідної прокладкою (прошивкою) провідників феритових сердечників, що забезпечувало при потрібних у той час ємностях високу надійність цих ЗП в найважчих (в електромагнітному відношенні) умовах.

Відомі також ємнісні та індуктивні ПЗП, в яких використовувалися провідники спеціальної форми, що утворюють ємнісні або індуктивні зв'язку.

В даний час досліджуються і інші принципи реалізації постійних ЗП, в деякому сенсі повертаються до магнітних і конденсаторним схемами, але на іншому рівні розвитку технологій.

Запис інформації в постійні ЗП, як правило, істотно відрізняється від зчитування за способом і часу виконання. Процес запису для напівпровідникових постійних ЗУ отримав також назву "пропалювання" або програмування, перше з яких пов'язане зі способом запису, зводиться до руйнування (розплавлення, прожига) з'єднувальних перемичок в чистому ЗУ.

В напівпровідникових ПЗП як елементи пам'яті, точніше, в якості нелінійних комутуючих і підсилювальних елементів зазвичай використовуються транзистори. Вони об'єднані в матрицю, вибірка даних з якої виробляється по рядках і стовпцях, відповідним вказаною адресою, так само, як і в інших ЗУ з довільним доступом. Один з можливих варіантів структурної схеми напівпровідникового ПЗП, представлений на рис. 25. Строго кажучи, безпосередньо запам'ятовування інформації в цьому ПЗУ здійснюється плавкою перемичкою, а транзистори виконують роль ключів-підсилювачів. Плавка перемичка може бути виготовлена з ніхрому, полікристалічного кремнію або інших матеріалів. В залежності від того, як саме працює підсилювач зчитування (в режимі повторювача або інвертора), наявність перемички відповідає запису "1" або "0". Руйнування перемички (імпульсом сильного струму) призводить до запису значення, зворотного вихідному.

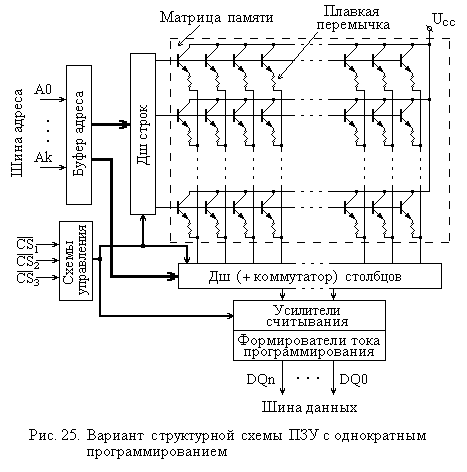


Рисунок 4.2-

Розрізняють дві великі групи ПЗП: програмовані виробником і програмовані користувачем.

ЗП першої групи, звані інакше масковим, зазвичай випускаються великими партіями. Інформація в них заноситься в процесі виготовлення цих ЗП на заводах: за допомогою спеціальної маски в кінці технологічного циклу на кристалі формується відповідна конфігурація з'єднань. Такі ЗП виявляються найбільш дешевими при масовому виготовленні. Їх зазвичай використовують для зберігання різних постійних програм і підпрограм, кодів, фізичних констант, постійних коефіцієнтів і пр.

У ПЗП, програмовані користувачем, інформація записується після їх виготовлення самими користувачами. При цьому існують два основних типи таких ЗУ: однократно програмовані і перепрограмовані.

Неважко згадати, що аналогічні різновиди є і у CDROM, які, по суті, є ПЗП (ROM), що виготовляються на основі іншої фізичної принципу.

Найбільш простими є однократно програмовані ПЗП. У цих ЗП запис якраз і проводиться за допомогою руйнування сполучних перемичок між висновками транзисторів і шинами матриці (хоча є й дещо інші технології). Зображення програмованого ПЗП на функціональній схемі показано на рис. 26.

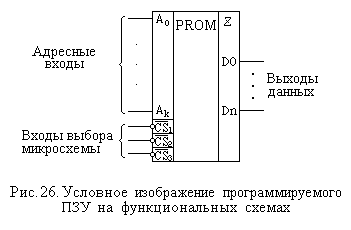


Рисунок 4.3-Умовне зображення ПЗП

Перепрограмовані ПЗП дозволяють виробляти в них запис інформації багаторазово. Звичайно, в таких ЗП повинен використовуватися інший принцип, ніж руйнування перемичок в процесі запису. Поширені технологічні варіанти використовують МОП-транзистори зі складним затвором (складовим або "плаваючим"), який здатний накопичувати заряд, що знижує порогову напругу відмикання транзистора, і зберігати цей заряд при вимкненому живленні. Програмування таких ПЗП і полягає у створенні зарядів на затворах тих транзисторів, де повинні бути записані дані (зазвичай "0", так як в початковому стані в таких мікросхемах записані всі "1").

Перед повторної записом потрібно зробити стирання раніше записаної інформації. Воно проводиться або електрично, подачею напруги зворотної полярності, або за допомогою ультрафіолетового світла. У мікросхем останнього типу було кругле віконечко з кварцового скла, через яке і висвітлювався кристал при стиранні.

Параметри постійних ЗУ відповідають технологічним нормам свого часу. На початку 2000-х років типові ємності мікросхем постійної пам'яті з масковим програмуванням складали близько 32-128 Мбіт, а часи звернення перевищували аналогічні показники оперативної пам'яті і для різних модифікацій досягали доя 100 нс.

# 4.5 Оперативна пам'ять

Оперативна пам'ять або оперативний запам'ятовуючий пристрій (ОЗП) застосовується для зберігання оперативної інформації, що вимагається в процесі обробки. ОЗП, на відміну від ПЗП, дозволяє: як вводити інформацію в адресуються клітинку (операція Зп), так і виводити (операція Чт) Так як звернення можливо до будь-якої адресується осередку в довільному порядку, то такі ЗУ називають запам'ятовуючими пристроями з довільною вибіркою (ЗУПВ ). Можливі різні способи обміну інформацією між ОЗУ і МП. Найчастіше використовуються запам'ятовуючі пристрої з довільною вибіркою, що дозволяють звертатися з будь-адресою в довільному порядку. Розрізняють статичні і динамічні ОЗП.

Статичний ОЗП реалізується на тригерах з безпосереднім зв'язком, які при включеному живленні можуть зберігати інформацію необмежено довго без додаткових керуючих сигналів.

Більшість ЗУПВ виготовляються на основі МОП транзисторів і мають так звану матричну організацію. Функціональна схема ЗУПВ 256х1 з матричної організацією типу 16х16 приведена на рисунку 4.4. Після дешифрування молодших бітів адреси А0-А3 виникає сигнал Xi, підключає i-й рядок матриці. Стовпець матриці Yj вибирається дешифратором старших бітів адреси А4-А7 і елемент пам'яті ЕПij підключається до схеми виведення або введення інформації в залежності від значення сигналу Сч / / Зп. При Сч / / Зп = 1 відбувається зчитування інформації з пам'яті; при Сч / / Зп = 0 - запис в пам'ять. Дозволяючий сигнал ВМ (вибір мікросхеми) активізує дану ІС. При заборонному значенні сигналу ВМ матриця ізольована від виходів дешифратора і шини вводу-виводу.

Статичні ЗУПВ можуть бути побудовані на напівпровідникових приладах різного типу. В даний час найбільш поширеними є ИС статичної пам'яті на МОП транзисторах. КМОП запам'ятовуючий елемент складається з 5-ти транзисторів, чотири з яких (VT1-VT4) утворюють тригер. Управління тригером для запису і зчитування здійснюється за допомогою ключа - транзистора VT5 (рис.4.5).

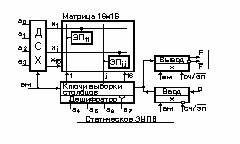


Рисунок 4.4

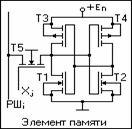


Рисунок 4.5-Елемент пам’яті

У динамічних ОЗП інформація зберігається у вигляді деякого заряду на ємності між затвором інформаційного МОП-транзистора і загальною точкою схеми - землею і паразитної ємності. При кімнатній температурі на цій ємності заряд зберігається протягом десятків мілісекунд. Зі збільшенням температури струм розряду через МОП транзистори швидко збільшується і при 100 0С допустимий час збереження заряду становить близько 2 мс. Для відновлення заряду на пам'ятною ємності її треба періодично підключати до джерела живлення, тобто виробляти регенерацію збережених даних, що значно ускладнює схему управління. Але кількість транзисторів у запам'ятовуючому елементі зменшується до 2-3. Це дозволяє значно збільшити щільність упаковки і зменшити споживану потужність, особливо в режимі зберігання. Динамічна пам'ять застосовується в системах, що вимагають великих обсягів інформації, що зберігається.

# 4.6 Підключення ЗП до шин

Так як шина даних (ШД) є спільною для багатьох пристроїв, то при їх одночасній роботі неминучий «конфлікт». Необхідно, щоб у кожен момент часу до ШД було підключено тільки один пристрій - джерело інформації. приймачів ж інформації може бути одночасно підключено декілька. Важливо лише, щоб потужність виходу джерела інформаційних сигналів була достатньою для підключення декількох приймачів. Оперативне подлюченіе і одключеніе виходу будь-якого пристрою здійснюється за допомогою схеми виходу, що має 3 можливих стани: 1, 0 і «відключено».

Запис даних а пам'ять

Щоб зрозуміти, яким чином в мікропроцесорній системі здійснюється запис даних в пам'ять, необхідно з'ясувати особливості передачі даних в пам'ять від будь-якого зовнішнього джерела. З цією метою наведемо тимчасову діаграму загального процесу запису даних в напівпровідникову пам'ять, після розгляду якої буде легко показати особливості використання системи з 3 шинами для виконання цієї операції.

Розглянемо рис. 1.3. Зазначимо, що адресні входи пам'яті на малюнку маркуються як А0-А11,

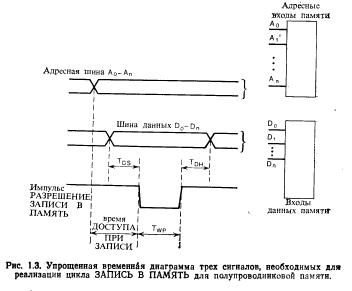


Рисунок 4.6 -Часова діаграма циклу запису до пам’яті

де Аn характеризує максимальне число адресних ліній, необхідних для даного блоку пам’яті. Наприклад, якщо пам’ять організована як 1024X1. То для забезпечення доступу до будь-якої комірки пам’яті необхідно 10 ліній, і адресні лінії будуть позначатися як А0-А9.

Лінії даних маркуються як Do-Do. Де D, характеризує максимальне чисто ліній даних пам’яті. Наприклад, якщо нам’яти організована як 256Х4 біт. Необхідно чотири лінії даних. Лінії даних такої пам’яті позначаються D0-D8.

Зауважимо також, що як адресні входи, так і входи даних повинні бути активізовані після видачі сигналу дозволі запису. Термін «активізувати» тут означає подачу на адресні лінії та лінії даних рівня напруги, відповідного логічним 1 або 0 і задовольняє прийнятим для використовуваного сімейства логічних схем параметрам. Наприклад, якщо в системі використовуються схеми сімейства TTL, то активізації логічної I відповідає інтервалу 2,4-5.0 В. а логічного 0-0.0-0.4 В.

Інтервал часу, протягом якого повинні зберігатися активізованим адресні та інформаційні входи до видачі сигналу дозволу запису, для різних пристроїв різний. Наприклад, для пристроїв пам’яті типу МОП (метал – окисел-напівпровідник) цей інтервал складає 200 нс, а для пам’яті на схемах TTL – 30 нс. Для точного визначення тимчасових співвідношенні слід ознайомитися з технічною документацією на відповідні вироби.

Аналізуючи рис. 1.3, важливо підкреслити, що незалежно від конкретних часових співвідношень сигналів для заданого пристрої пам’яті мікропроцесорна система повинна задовольнити всіх існуючих часових параметрів для пристрою пам’яті даного типу. Якщо ж ці тимчасові співвідношення не дотримуються, то надійної взаємозв’язку між мікропроцесором і пам’яттю системи забезпечити не вдасться.

Для того щоб успішно реалізувати обробку сигналу запису в пам’ять в системі з 3 шинами, на шинах системи необхідно виконати наступні дії:

1. На адресній шині A0-А15 повинен бути активізований адресу пам’яті (тобто адресу клітинки, куди записуються дані, що генеруються мікропроцесором, з шини даних).

2. На шину даних D0-D8 повинні надійти дані з мікропроцесора. (Ці дані необхідно записати в комірку, адреса якої міститься на адресній шині.)

3. Після здійснення дій I і 2 на лінію запису в пам’ять шини управлінні повинен надійти відповідним чином синхронізований імпульс дозволу запису необхідного рівня напруги. При цьому здійснюється передача даних. Таким чином, шина управління проводить управління системою за допомогою відповідним чином синхронізованих імпульсів.

Однак існує ряд додаткових дій, яким також необхідно приділити увагу. Схеми, реалізують їх, простіше описувати при більш детальному ознайомленні з системою.

Тепер ясно, яким чином сигнали різного логічного рівня на кожній з шин {адресної, даних, управління) взаємодіють між собою при формуванні необхідної вхідної інформації при записі даних в пам’ять системи. Шини системи забезпечують надійне з’єднання мікропроцесора з

пам’яттю системи шляхом формування необхідної комбінації сигналів на входах пам’яті.

Читання даних з пам’яті

Розглянемо, яким чином у системі з 3 шинами реалізується читання даних з пам’яті. Коротко розглянемо основні особливості читання даних з будь напівпровідникової пам’яті.

На рис. 1.4 наведена типова тимчасова діаграма виконання операції читання даних з пам’яті. Зазначимо, що відповідно до рис 1.4 адресні лінії А0-Аn повинні бути активізовані зі збереженням стабільного рівня сигналу до моменту часу, коли дані з пам’яті надходять на шину даних по сигналу відповідної лінії шини керування. Згадаймо, що дані в системі передаються з одного пристрою на інший за допомогою шини даних. Для виконання цієї операції шина даних переводиться а такий режим, при якому логічні рівні на цій шині відповідають даним, збереженим у пам’яті. І нарешті, сигнал шини управління визначає потрібний момент видачі даних з пам’яті на шину даних. Таким чином, щоб реалізувати операцію читання даних на пам’яті системи необхідно виконати наступні дії:

1. Забезпечення стабільних рівнів сигналів на адресної колючці А0-Аn.

2. Підготовка шини даних для прийому даних (тобто вона переводиться в режим прийому даних в мікропроцесор).

3. Після реалізації кроків 1 і 2 активізація шиною управління лінії управління читанням з пам’яті. При цьому дані з пам’яті поступають на шину даних і можуть бути сприйняті мікропроцесором. Як бачимо, реалізація операції читання з пам’яті передбачає взаємодію сигналів відповідних логічних рівнів на всіх трьох шипах.

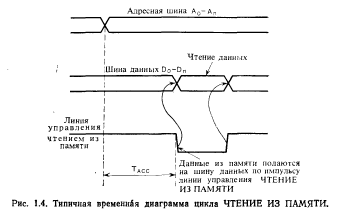
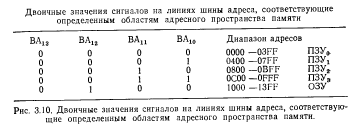


Рисунок 4.7- Часова діаграма циклу читання з пам’яті



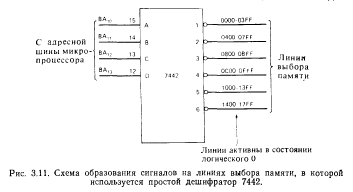


Рисунок4.8-Використання дешифратора для створювання сигналів вибору мікросхеми пам’яті з заданою ємністю

**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.

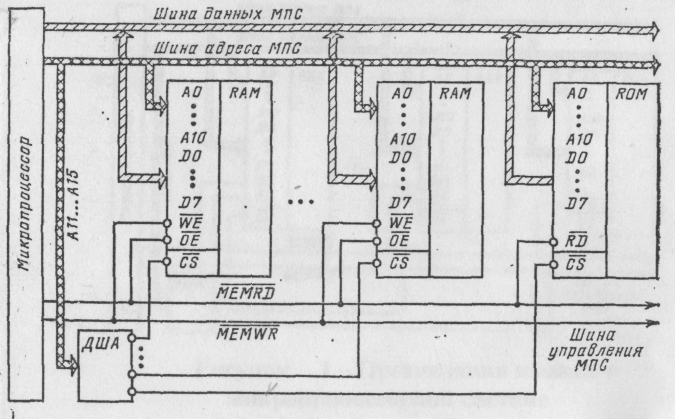


Рисунок - Организация памяти в микропроцессорной системе

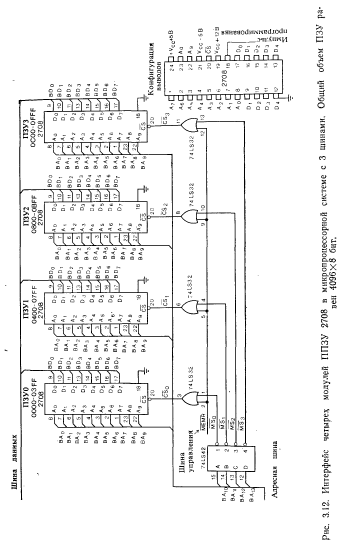
Для управления записью данных в память служит сигнал шины управления *,* чтением из памяти – сигнал *.* Выбор конкретного кристалла памяти осуществляется с помощью дешифратора адреса памяти ДША, на вход которого поступают старшие разряды с шины адреса МПС (те, которые не подаются на собственно шины адреса 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**

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.

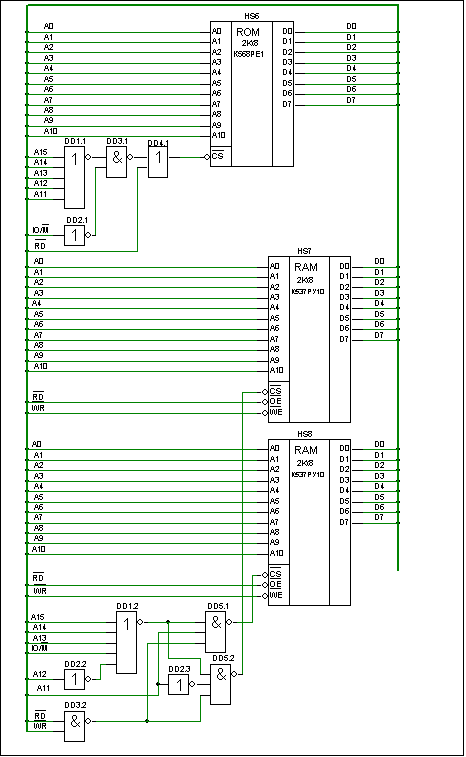
[](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:8086_adress.png)

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).

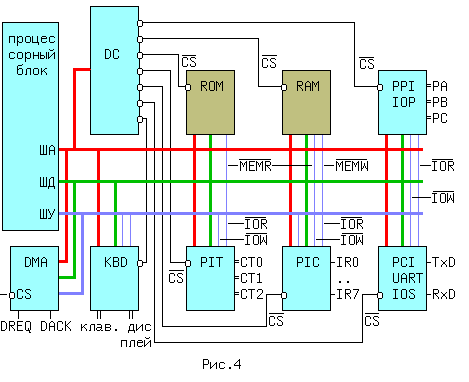
На то время такой [режим адресации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B6%D0%B8%D0%BC_%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%D1%86%D0%B8%D0%B8) обеспечивал множество преимуществ: ёмкость памяти могла составлять до 1 Мбайт, хотя команды оперировали 16-битными адресами; упрощалось использование 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.



Подключение ПЗУ и ОЗУ к системной шине

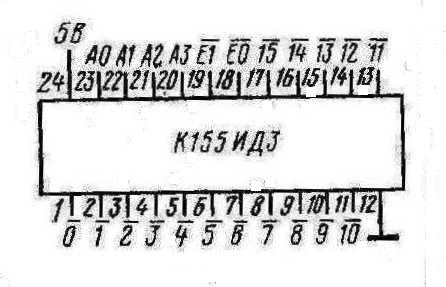
**Дешифратори адреси пристроїв вводу-виводу та пам’яті**

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 Інтерфейс МП з пристроями вводу-виводу

# 5.1 Організація вводу/виводу в мікропроцесорній системі

Введенням / виведенням (ВВ) називається передача даних між ядром ЕОМ, що включає в себе мікропроцесор і основну пам'ять, і зовнішніми пристроями (ВУ). Це єдиний засіб взаємодії ЕОМ з "зовнішнім світом", і архітектура ВВ (режими роботи, формати команд, особливості переривань, швидкість обміну тощо) безпосередньо впливає на ефективність всієї системи. За час еволюції ЕОМ підсистема ВВ зазнала найбільших змін завдяки розширенню сфери застосування ЕОМ і появі нових зовнішніх пристроїв. Особливо важливу роль засобу ВВ грають в керуючих ЕОМ. Розробка апаратних засобів і програмного забезпечення ВВ є найбільш складним етапом проектування нових систем на базі ЕОМ, а можливості ВВ серійних машин представляють собою один з важливих параметрів, що визначають вибір машини для конкретного застосування.

Функції пристроїв введення / виводу

Пристрої вводу / виводу обмінюються інформацією з магістраллю за тими ж принципами, що і пам'ять. Найбільш істотна відмінність з точки зору організації обміну полягає в тому, що модуль пам'яті має в адресному просторі системи багато адрес (до декількох десятків мільйонів), а пристрій введення / виводу зазвичай має небагато адрес (зазвичай до десяти), а іноді і всього одна адреса . Але модулі пам'яті системи обмінюються інформацією тільки з магістраллю, з процесором, а пристрої введення / виводу взаємодіють ще і з зовнішніми пристроями, цифровими або аналоговими. Тому різноманітність пристроїв введення / виводу незмірно більше, ніж модулів пам'яті. Часто використовуються ще й інші назви для пристроїв введення / виводу: пристрої сполучення, контролери, карти розширення, інтерфейсні модулі і т.д.

Об'єднують всі пристрої введення / виводу загальні принципи обміну з магістраллю і, відповідно, загальні принципи організації вузлів, які здійснюють сполучення з магістраллю. Спрощена структура пристрою введення / виводу (точніше, його інтерфейсної частини) наведена на рис. 2.21. Як і у випадку модуля пам'яті, вона обов'язково містить схему селектора адреси, схему управління для обробки стробів обміну і буфери даних.

Найпростіші пристрої введення / виводу видають на зовнішній пристрій код даних у паралельному форматі і приймають із зовнішнього пристрою код даних у паралельному форматі. Такі пристрої введення / виводу часто називають паралельними портами введення / виведення. Вони найбільш універсальні, тобто задовольняють потреби сполучення з великим числом зовнішніх пристроїв, тому їх часто вводять до складу мікропроцесорної системи в якості стандартних пристроїв. Паралельні порти зазвичай є в складі мікроконтролерів. Саме через паралельні порти мікроконтролер зв'язується із зовнішнім світом.

Вхідний порт (порт вводу) у найпростішому випадку є паралельний регістр, в який процесор може записувати інформацію. Вихідний порт (порт виводу) зазвичай є просто односпрямований буфер, через який процесор може читати інформацію від зовнішнього пристрою. Саме такі порти показані для прикладу на рис. 2.21. Порт може бути і двонаправленим (вхідним / вихідних). У цьому випадку процесор пише інформацію в зовнішній пристрій і читає інформацію з зовнішнього пристрою по одному і тому ж адресою в адресному просторі системи. Вхідні і вихідні лінії для зв'язку із зовнішнім пристроєм при цьому можуть бути об'єднані поразрядно, утворюючи двонаправлені лінії.

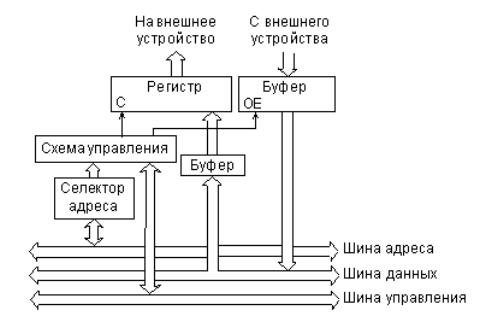


Рис. 5.1 - Структура простого пристрою вводу/виводу

При зверненні з боку магістралі селектор адреси розпізнає адресу, приписаний Цей пристрій введення / виведення. Схема управління видає внутрішні строби обміну у відповідь на магістральні строби обміну. Вхідний буфер даних забезпечує електричне узгодження шини даних з цим пристроєм (буфер може і не бути). Дані з шини даних записуються в регістр за сигналом С і видаються на зовнішній пристрій. Вихідний буфер даних передає вхідні дані із зовнішнього пристрою на шину даних магістралі в циклі читання з порту.

Більш складні пристрої введення / виводу (пристрою сполучення) мають у своєму складі внутрішню буферну оперативну пам'ять і навіть можуть мати мікроконтролер, на який покладено виконання функцій обміну з зовнішнім пристроєм.

Кожному пристрою введення / виводу відводиться свою адресу в адресному просторі мікропроцесорної системи. Дублювання адрес повинно бути виключено, за цим повинні стежити розробник і користувач мікропроцесорної системи.

Пристрої вводу / виводу крім програмного обміну можуть також підтримувати режим обміну по перериваннях. В цьому випадку вони перетворять надходить від зовнішнього пристрою сигнал запиту на переривання в сигнал

запиту переривання, необхідний для даної магістралі (або в послідовність сигналів при векторному перериванні). Якщо потрібно використовувати режим ПДП, пристрій вводу / виводу повинно видати сигнал запиту ПДП на магістраль і забезпечити роботу в циклах ПДП, прийнятих для даної магістралі.

У складі мікропроцесорних систем, як правило, виділяються три спеціальні групи пристроїв введення / виводу:

• пристрої інтерфейсу користувача (введення інформації користувачем і виведення інформації для користувача);

• пристрої введення / виводу для тривалого зберігання інформації;

• таймерні пристрої.

До пристроїв введення для інтерфейсу користувача відносяться контролери клавіатури, тумблерів, окремих кнопок, миші, трекбола, джойстика і т.д. До пристроїв виводу для інтерфейсу користувача відносяться контролери світлодіодних індикаторів, табло, рідкокристалічних, плазмових і електронно9лучевих екранів і т.д. У простих випадках керуючих контролерів або мікроконтролерів ці кошти можуть бути відсутні. У складних мікропроцесорних системах вони є обов'язково. Роль зовнішнього пристрою в даному випадку грає людина.

Пристрої вводу / виводу для тривалого зберігання інформації забезпечують сполучення мікропроцесорної системи з дисководами (ком пакт дисків або магнітних дисків), а також з накопичувачами на магнітній стрічці. Застосування таких пристроїв істотно збільшує можливості мікропроцесорної системи щодо зберігання виконуваних програм і накопичення масивів даних. У найпростіших контролерах ці пристрої відсутні.

Таймерні пристрої відрізняються від інших пристроїв введення / виводу тим, що вони можуть не мати зовнішніх виводів для підключення до зовнішніх пристроїв. Ці пристрої призначені для того, щоб мікропроцесорна система могла витримувати задані тимчасові інтервали, стежити за реальним часом, створювати імпульси і т.д. В основі будь-якого таймера лежить кварцовий тактовий генератор і багаторозрядні двійкові лічильники, які можуть перезапускати один одного. Процесор може записувати в таймер коефіцієнти ділення тактової частоти, кількість відлічуваних імпульсів, задавати режим роботи лічильників таймера, а читає процесор вихідні коди лічильників. В принципі виконати практично всі функції таймера можна і програмним шляхом, тому іноді таймери в системі відсутні. Але включення в систему таймера дозволяє вирішувати складніші завдання і будувати більш ефективні алгоритми.

Ще один важливий клас пристроїв введення / виведення - це пристрої для підключення до інформаційних мереж (локальних і глобальних). Ці пристрої поширені не так широко, як пристрої трьох перерахованих раніше груп, але їх значення з кожним роком стає все більше. Зараз засоби зв'язку з інформаційними мережами вводяться іноді навіть у прості контролери.

Іноді пристрої введення / виводу забезпечують сполучення з зовнішніми пристроями за допомогою аналогових сигналів. Це буває дуже зручно, тому до складу деяких мікроконтролерів навіть вводять внутрішні ЦАП і АЦП.

# 5.1.2 Програмна модель зовнішнього пристрою

Підключення зовнішніх пристроїв до системної шини здійснюється за допомогою електронних схем, званих контролерами ВВ (інтерфейсами ВВ). Вони погодять рівні електричних сигналів, а також перетворять машинні дані в формат, необхідний пристрою, і навпаки. Зазвичай контролери ВВ конструктивно оформляються разом з процесором у вигляді інтерфейсних плат.

В процесі введення / виводу передається інформація двох видів: керуючі дані (слова) і власне дані, або дані-повідомлення. Керуючі дані від процесора, звані також командними словами або наказами, ініціюють дії, не пов'язані безпосередньо з передачею даних, наприклад запуск пристрою, заборона переривань і т.п. Керуючі дані від зовнішніх пристроїв називаються словами стану; вони містять інформацію про певні ознаки, наприклад про готовність пристрою до передачі даних, про наявність помилок при обміні і т.п. Стан зазвичай представляється в декодованому формі - один біт для кожної ознаки.

Регістр, що містить групу біт, до якої процесор звертається в операціях ВВ, утворює порт ВВ. Таким чином, найбільш загальна програмна модель зовнішнього пристрою, який може виконувати введення та виведення, містить чотири регістри ВВ: регістр вихідних даних (вихідний порт), реєстр вхідних даних (вхідний порт), реєстр управління і регістр стану (рис. 3.1). Кожен з цих регістрів повинен мати однозначну адресу, який ідентифікується дешифратором адреси. Залежно від особливостей пристрою загальна модель конкретизується, наприклад, окремі регістри стану та управління об'єднуються в один регістр, у пристрої введення (виведення) є тільки регістр вхідних (вихідних) даних, для введення і виведення використовується двонаправлений порт.

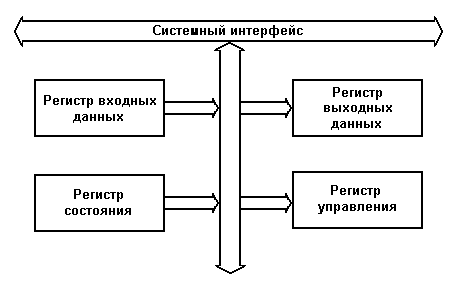


Рисунок 5.2 - Програмна модель зовнішнього пристрою

Безпосередні дії, пов'язані з введенням / виведенням, реалізуються одним із двох способів, що розрізняються адресацією регістрів ВВ.

Інтерфейс з ізольованими шинами характеризується роздільної адресацією пам'яті і зовнішніх пристроїв при обміні інформацією. Ізольований ВВ передбачає наявність спеціальних команд введення / виводу, загальний формат яких показаний на рис. 5.2. При виконанні команди введення IN вміст адресується вхідного регістра PORT передається у внутрішній регістр REG процесора, а при виконанні команди OUT вміст регістра REG передається у вихідний порт PORT. У процесорі можуть бути і інші команди, пов'язані з ВВ і пов'язані з перевіркою і модифікацією вмісту регістра управління та стану.

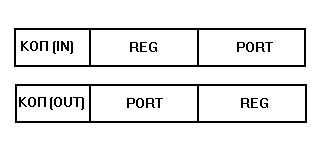


Рисунок 5.3 - Команди вводу/виводу (загальний формат)

Неважко помітити, що в цьому способі адресний простір портів введення і виведення ізольовано від адресного простору пам'яті, тобто в ЕОМ один і той же адресу можуть мати порт ВВ і осередок пам'яті. Поділ адресних просторів здійснюється за допомогою сигналів, що відносяться до систем ВВ і пам'яті (MEMRD # - зчитування даних з пам'яті, MEMWR # - запис даних в пам'ять, IORD # - читання порту ВВ, IOWR # - запис в порт ВВ) (# - активний низький рівень сигналів).

У ЕОМ, розрахованої на ізольований ВВ, неважко перейти до ВВ, відображеному на пам'ять. Якщо, наприклад, адресний простір пам'яті становить 64 Кбайт, а для програмного забезпечення досить 32 Кбайт, то область адрес від 0 до 32 К-1 використовується для пам'яті, від 32 К до 64 К-1 - для введення / виводу. При цьому ознакою, диференціюються звернення до пам'яті і портів ВВ, може бути старший біт адреси.

Таким чином, інтерфейс із загальними шинами (введення / виведення з відображенням на пам'ять) має організацію, при якій частина загального адресного простору відводиться для зовнішніх пристроїв, регістри яких адресуються так само, як і осередки пам'яті. В цьому випадку для адресації портів ВВ використовуються повні адресні сигнали: READ - читання, WRITE - запис.

В операційних системах ЕОМ є набір підпрограм (драйверів ВВ), які керують операціями ВВ стандартних зовнішніх пристроїв. Завдяки їм користувач може не знати багатьох особливостей ВУ та інтерфейсів ВВ, а застосовувати чіткі програмні протоколи.

# 5.2 Способи обміну інформацією в мікропроцесорній системі

В ЕОМ застосовуються три режими введення/виводу: програмно-керований ВВ (називаний також програмних або нефорсованим ВВ), ВВ по перериваннях (форсований ВВ) і прямій доступ до пам'яті. Перший з них характеризується тим, що ініціювання і керування ВВ здійснюється програмою, виконуваної процесором, а зовнішні пристрої грають порівняно пасивну роль і сигналізують тільки про свій стан, зокрема, про готовність до операцій уведення/висновку. В другому режимі ВВ ініціюється не процесором, а зовнішнім пристроєм, що генерує спеціальний сигнал переривання. Реагуючи на цей сигнал готовності пристрою до передачі даних, процесор передає керування підпрограмі обслуговування пристрою, що викликав переривання. Дії, виконувані цією підпрограмою, визначаються користувачем, а безпосередніми операціями ВВ керує процесор. Нарешті, у режимі прямого доступу до пам'яті, що використовується, коли пропускної здатності процесора недостатньо, дії процесора припиняються, вона відключається від системної шини і не бере участь у передачах даних між основною пам'яттю і швидкодіючим ВУ. Помітимо, що у всіх вищевказаних випадках основні дії, виконувані на системній магістралі ЕОМ, підкоряються двом основним принципам. ,

1. У процесі взаємодії будь-яких двох пристроїв ЕОМ одне з них обов'язково виконує активну, керуючу роль і є задавач , друге виявляється керованим, виконавцем. Найчастіше задавач є процесор. ,
2. Іншим важливим принципом, закладеним у структуру інтерфейсу, є принцип квітування (запиту - відповіді): кожен керуючий сигнал, посланий задавач, підтверджується сигналом виконавця. При відсутності відповідного сигналу виконавця протягом заданого інтервалу часу формується так називаний тайм-аут, задавач фіксує помилку обміну і припиняє дану операцію.

# 5.2.1 Програмно-кероване введення/виводу

Даний режим характеризується тим, що всі дії по введенню/виводу реалізуються командами прикладної програми. Найбільш прості ці дії виявляються для "завжди готових" зовнішніх пристроїв, наприклад індикатора на світлодіодах. При необхідності ВВ у відповідному місці програми використовуються команди IN або OUT. Така передача даних називається синхронним або безумовним ВВ.

Однак для більшості ВУ до виконання операцій ВВ треба переконатися в їхній готовності до обміну, тобто ВВ є асинхронним. Загальний стан пристрою характеризується прапором готовності READY, називаним також прапором готовності/зайнятості (READY/BUSY). Іноді стану готовності і зайнятості ідентифікуються окремими прапорами READY і BUSY, що входять у слово стану

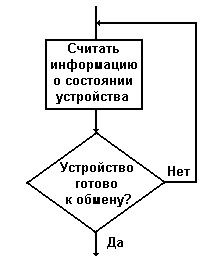
пристрою одного або декількох слів даних. Коли ж прапор скинутий, процесор виконує цикл із 2-3 команд із повторною перевіркою прапора READY доти, поки пристрій не буде готовий до операцій В\В (рисунок 5/4). Даний цикл називається циклом чекання готовності ВП і реалізується в різних процесорах по-різному.

Рисунок 5.4- Цикл програмного чекання готовності зовнішнього пристрою

Основний недолік програмного ВВ зв'язаний з непродуктивними втратами часу процесора в циклах чекання. До переваг варто віднести простоту його реалізації, що не вимагає додаткових апаратних засобів.

Операція В/В інформації процесора здійснюється 2-ма способами.

1. Спосіб В/В по методу доступу до пам’яті

Команда вивода Команда вводу

STA LDA

A15

A0

МП



A15

A0



А15А0

А15А0

У даному способі зовнішній пристрій має загальний адресний простір з ЕОМ.

*Переваги* цього способу полягають у тім, що він не вимагає додаткових апаратних засобів, і крім того дозволяє підключати велику кількість зовнішніх пристроїв. Програмна реалізація не вимагає додаткових команд.

*Недоліком* даного способу варто вважати складність написання програми (лістинга);

Команда виводу OUT Команда вводу IN

RD

WR

A15

A0 МП

I/OWR

A15

A0

I/ORD

1. Ізольований спосіб введення-виводу інформації, передбачає роботу з комірками пам’яті, що ізольовані від ЕОМ. Ізоляція даних осередків здійснюється шляхом введення в мікропроцесор додаткових сигналів. Сигнали I/ОRD, I/ОWR служать для читання або запису інформації даних комірок пам’яті. Комірки пам’яті ЕОМ активізуються за допомогою сигналів *D*/*R* . Таким чином, здійснюється ізоляція комірок пам’яті для введення-виводу інформації різних периферійних пристроїв. Програмно для реалізації цього способу були введені додаткові команди I/O (in, out).

*Недоліки:*

1. Обмежена кількість пристроїв, що підключаються, 28-216
2. Необхідні додаткові апаратні засоби для реалізації даного способу.

*Перева*

1. Коротка адресація зовнішніх пристроїв.
2. Програми є більш наочними через наявність цільових операторів I/O.

# 5.2.2Інтерфейс МП при вводі/виводі одиничної інформації

Під інтерфейсом будемо розуміти сукупність апаратних і програмних засобів у необхідних для введення/виводу інформації в/в МП. При цьому сукупність апаратних засобів, що забезпечують доступ інформації на системну шину називають адаптером. В основі побудови адаптера лежить Д-тригер, що працює по наступному принципі: сигнал (тактовий) входу З підтверджує сигнал, що надходить на вхід Д. Так, якщо наш перемикач П знаходиться в стані LOW, то з приходом сигналу на С вхід на виході тригера буде Н. Якщо ж перемикач буде знаходитися в Н, то з приходом такту на вхід З на виході буде одиниця, що через системну шину надійде в акумулятор МП. Програма по команді LDA adz 1 МП виставить адресу і сигнал RD (read) по яких через логічну схему «і» через Д-тригер буде занесена інформація в акумулятор. У наступному програміст складає програму, що дозволяє визначити в якому положенні знаходиться перемикач

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

Синхронізація введення /виводу за допомогою переривань

У попередньому прикладі ми припускали, що коли програма вказує МП увести дані в порт, вони вже малися в наявності і відбулася зміна стану тумблера. Однак, ця умова в більшості випадків не виконується. Наприклад, клавішний пристрій, при роботі з яким не можливо передбачити коли буде натиснута клавіша. У цьому випадку проблема вирішується за допомогою переривань, що бувають 2-х типів :

* + програмні переривання;
  + апаратні переривання.

*Програмне переривання* – найбільш простий метод і використовується при роботі з невеликою кількістю пристроїв. Основною ідеєю програмного переривання є введення даних, використовується в програмі цикл опитування регістра стану об’єкта.

*Перевага* – вимагає менше апаратних засобів, і знаходячись у підпрограмі контролера не вимагає синхронізації.

*Недоліком* є потреба в дуже розгалуженій програмі, що займають час МП у випадках більшого числа пристроїв опитування і необхідного значного проміжку часу для відповіді на запит.

*Апаратне переривання* – зовнішній пристрій при зміні стану апаратно формує сигнал, що надходить на вхід INT процесора, повідомляючи процесорові про необхідність роботи з ним. У цьому випадку МП, що виконував поточну програму передає на збереження в стек вміст акумулятора лічильник команд і переходить до обслуговування переривання, після виконання якої повертається до виконання програми.

*Перевага* – швидкодія відповіді, краще використання МП і потреби в багато разів менших програмних засобів.

*Недолік* – полягає в тому, що в цьому випадку робота МП і зовнішніх пристроїв відбувається синхронно і потрібні додаткові апаратні засоби для рішення цієї проблеми.

# 5.2.3 Переваги і недоліки ізольованого вводу/виводу

1) Адреса порту в/в може бути короткою. В бiльшостi систем для адресацii порта в/в достатньо 8 розрядiв,що спрощуе декодуючi системи i дає можли-вiсть використовування коротких команд.

2) Можливо легко розробити додатковi сигнали передачi iнформацii при в/в (стробування, запуск).

3) Програми стають бiльш наглядними так як операцii в/в вiдбуваються задопомогою команд, якi вiдрiзняються вiд iнших.

4) Розробка систем в/в може робитись окремо вiд розробки моделей пам'ятi.

5) Дає можливiсть використовування при малий ступенi інтеграцii ВIС.

# 5.2.4 Переваги і недоліки вводу/виводу, який адресується як пам’ять

1) Будь яка команда, яка працює з даними, якi знаходяться в пам'ятi, може працювати з даними, якi знаходяться в пристроi вводу або виводу. Нiяких особливих команд в/в непотрiбно i програмування багатьох задач спрощується.

2) Не потрiно окремiй системи для декодування в/в.

3) В систему легко включити iнтерфейс ВIС i спецiальнi контролери. Цi пристроi часто мiстять регiстри, якi настроюються програмно. Цi обставини є суттєвими так як керування в цiх умовах за допомогою портiв в/в неможливо здiйснити.

# 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.

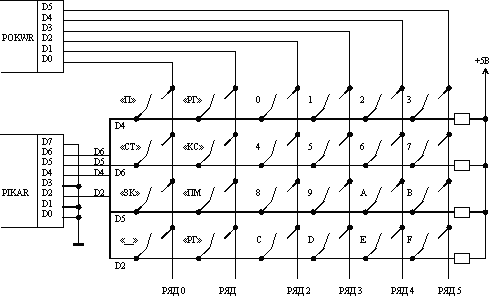


Рис.7.3 - Принципова схема матричної клавіатури

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

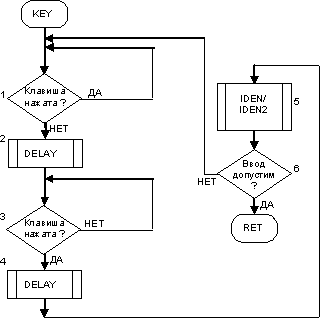


Рис.7.5 - Блок-схема підпрограми IND

(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.

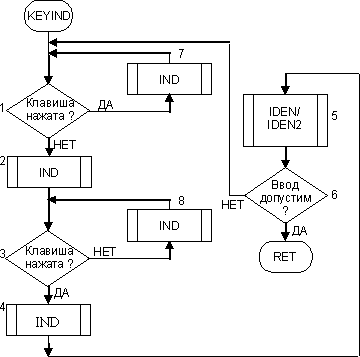


Рис.7.6 - Блок-схема підпрограми KEYIND

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

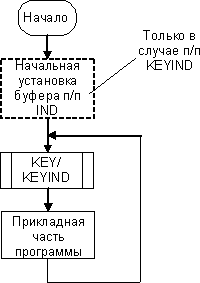


Рис.7.7 - Введення підпрограм KEY (KEYIND) в прикладну програму

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).

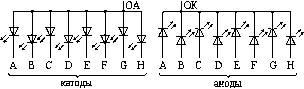


Рис.7.9-Принципові схеми семисегментних індикаторів

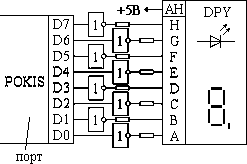


Рис.7.10 - Підключення семисегментного індикатора до порту прямого вивода

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.

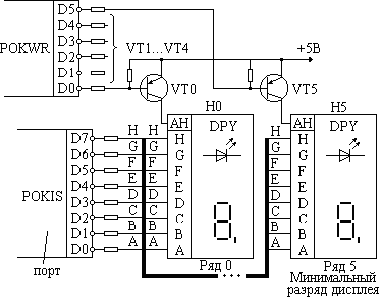


Рис.7.11 - Шестирозрядний семисегментний дисплей з динамічним керуванням

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).

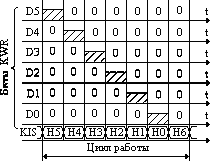


Рисунок 7.12 - Тимчасові діаграми

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.

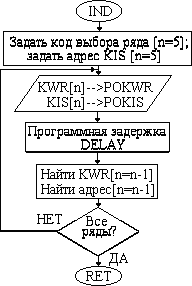


Рис.7.13.Блок-схема підпрограми IND n – номер рядку на схемі дисплею з динамічним управлінням

«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

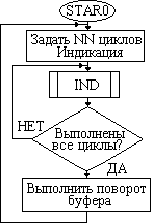


Рис.7.14.Блок-схема програми, яка забезпечує вивід в режимі «бегущей рядка»

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | ; 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

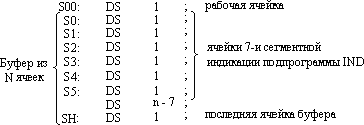


Рис.7.17.Структура буфера програми, яка забезпечує вивід в режимі бегущей рядка

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.

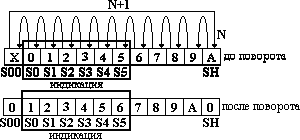


Рисунок 7.18

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

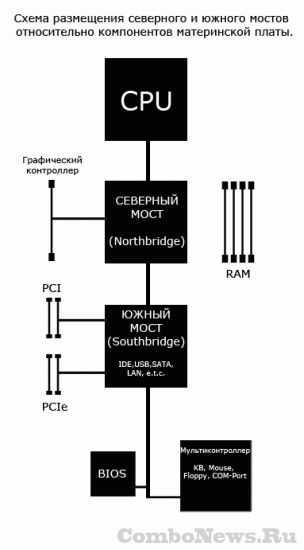
Початковий (базовий) адреса таблиці TAB7 складається в підпрограмі з вихідним кодом 0 ... F. При цьому в регістровий парі HL виходить адресу відповідного семисегментного коду. Цей код витягується з таблиці TAB7 командою MOV A, M і поміщається в акумулятор

# 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

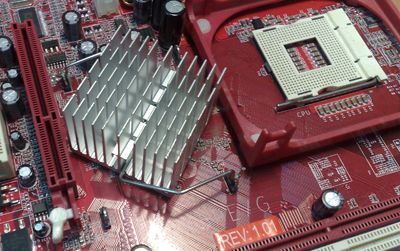


Функцією північного моста є забезпечення взаємодії ЦП, оперативної пам'яті і відеопідсистеми (як самого швидкодіючого інтерфейсного модуля), а також - інтерфейс ядра НД з системною шиною PCI. Південний міст включає контролери для зв'язку із зовнішніми пристроями по інтерфейсах ISA (міст PCI-ISA), EIDE, 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!).

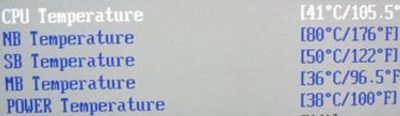
Північний і південний мости комп'ютера (а правильніше буде сказати, материнської плати) - це два основних функціональних контролера, які відповідають за роботу всіх компонентів системної плати і називаються чіпсетом (від англ. Chipset).

**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".



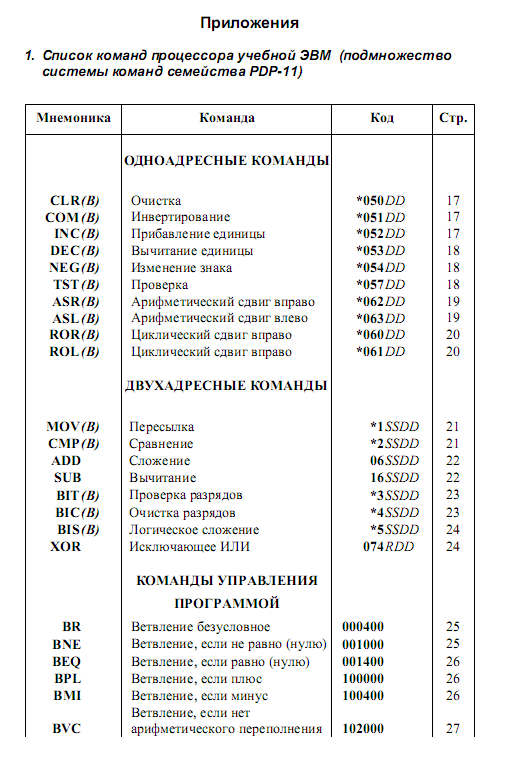
Завищена температура цілком обгрунтована. По-перше, північний міст знаходиться в безпосередній близькості від центрального процесора, по-друге, він знаходиться вище відеокарти, жорстких дисків і південного моста. Це означає, що частина тепла від вищезгаданих пристроїв доходить до північного моста. Ну 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**

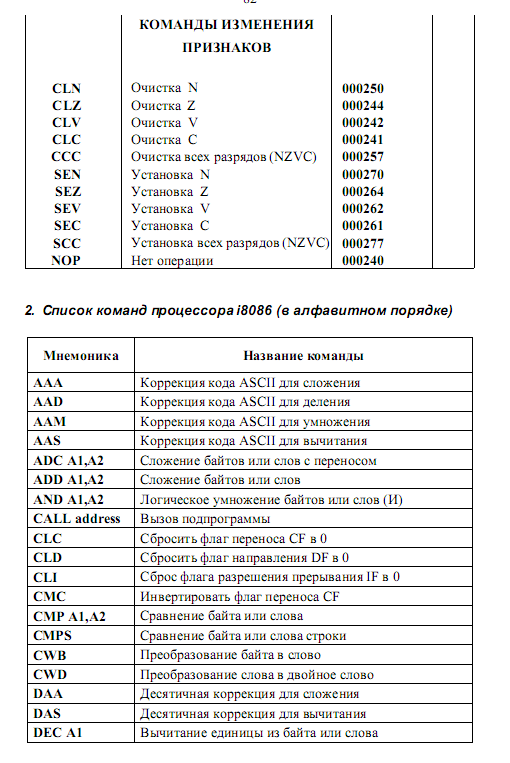
Південний міст (Southbridge) - це функціональний контролер, відомий як контролер вводу-виводу або ICH (In / Out Controller Hub). Відповідає за так звані "повільні" операції, до яких відноситься відпрацювання взаємодії між інтерфейсами IDE, SATA, USB, LAN, Embeded Audio і північним мостом системи, який, 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).

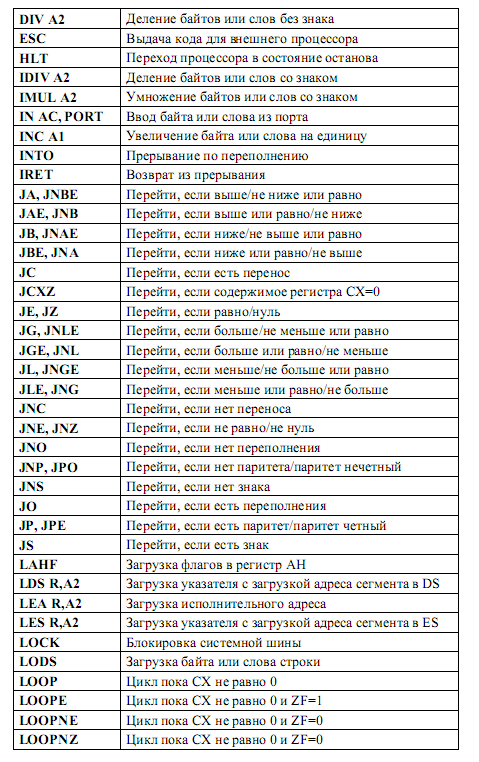


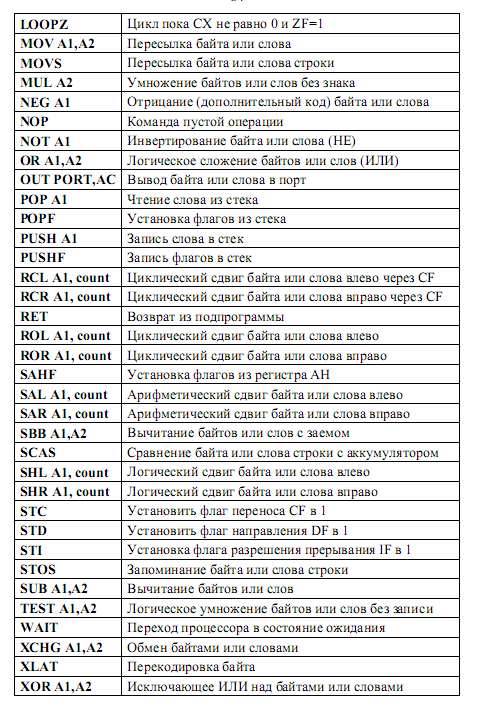
Список обслуговуються систем материнської плати південним мостом досить великий. Крім вищенаведених IDE, SATA, USB, LAN і іншого, південний міст відповідає ще й за SM шину (використовується для управління вентиляторами на платі), DMA-контролер, IRQ-контролер, системний годинник, BIOS, системи енергозабезпечення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