МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Ю.В. Китаев

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Учебное пособие
часть 1

Санкт-Петербург

2016

Китаев Ю.В. “Основы микропроцессорной техники”. Учебное пособие - СПб: Университет ИТМО, 2016., 51 с.
Даются основные способы представления данных в МП технике. Приведены примеры проектирования и программирования типовых задач ввода/вывода данных для устройств с использованием аппаратных средств МП техники.

Для студентов, обучающихся по направлениям 200100.62 Приборостроение, 200700.62 Фотоника и оптоинформатика, 210700.62 Инфокоммуникационные технологии и системы связи.

Рекомендовано к печати Советом ИФФ от 13 октября 2015г., протокол №2.

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

О Г Л А В Л Е Н И Е

КОДИРОВАНИЕ ИНФОРМАЦИИ В ЭВМ 5
СИСТЕМЫ СЧИСЛЕНИЯ 5
МАШИННОЕ ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ 5
ЧИСЛА С ФИКСИРОВАННОЙ ТОЧКОЙ 6
1.4 ДИАПАЗОН ЦЕЛЫХ ЧИСЕЛ С ФИКСИРОВАННОЙ ТОЧКОЙ 8
1.5 ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ (ВЕЩЕСТВЕННЫЕ) 8
1.6 ДИАПАЗОН ПРЕДСТАВЛЕНИЯ ВЕЩЕСТВЕННЫХ ЧИСЕЛ 9
1.7 ДВОИЧНО-ДЕСЯТИЧНЫЙ КОД 10
1.8 БУКВЕННО-ЦИФРОВОЙ КОД 10
1.9 ВОСЬМИСЕГМЕНТНЫЙ КОД 11
1.10 НЕОДНОЗНАЧНОСТЬ ПРЕДСТАВЛЕНИЯ ДВОИЧНЫХ НАБОРОВ 11
2. МИКРОПРОЦЕССОР 8086(88) 12
2.1 СТРУКТУРНАЯ СХЕМА МИКРОПРОЦЕССОРА 12
2.2 РЕГИСТР ФЛАГОВ (ПРИЗНАКОВ) 14
2.3 ОРГАНИЗАЦИЯ ПАМЯТИ И ВЫЧИСЛЕНИЕ АДРЕСА 15
2.4 ПРОЦЕССОРНЫЙ БЛОК (ПБ) Система с раздельным адресным пространством ЗУ и ВУ 16
2.5 МИКРОПРОЦЕССОРНАЯ СИСТЕМА С ШИННЫМ ИНТЕРФЕЙСОМ 19
2.5.1 МИКРОКОНТРОЛЛЕРНАЯ СИСТЕМА С ШИННЫМ ИНТЕРФЕЙСОМ Система с общим адресным пространством ЗУ и ВУ 21
2.5.2 ШИНА VS ПОРТ 24
2.6 СТЕК 26
2.7 СПОСОБЫ ВВОДА-ВЫВОДА 28

2.7.1 ПРОГРАММНЫЙ ВВОД-ВЫВОД..... 28
2.7.2 ВВОД-ВЫВОД ПО ПРЕРЫВАНИЮ..... 29
2.7.3 ПРЯМОЙ ДОСТУП К ПАМЯТИ (ПДП) И ТРАНЗАКЦИИ 31
2.8 ЗАДАЧИ И УПРАЖНЕНИЯ..... 32
3. ПРОГРАММИРОВАНИЕ НА АССЕМБЛЕРЕ..... 35
3.1 АССЕМБЛЕР. ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ. 35
3.2 ФОРМАТ КОМАНД И ИХ КЛАССИФИКАЦИЯ 37
3.3 НЕКОТОРЫЕ ОПЕРАТОРЫ, ПРЕДОПРЕДЕЛЁННЫЕ ИМЕНА, ДИРЕКТИВЫ И КОМАНДЫ АССЕМБЛЕРА 80X86(8088/86) 39
3.3.1 ПРЕДОПРЕДЕЛЕННЫЕ ИМЕНА..... 39
3.3.2 ОПЕРАТОРЫ..... 39
3.3.3 ДИРЕКТИВЫ (ПСЕВДООПЕРАТОРЫ) 40
3.3.4 КОМАНДЫ ПЕРЕСЫЛКИ 43
3.3.5 АРИФМЕТИЧЕСКИЕ КОМАНДЫ 45

КОДИРОВАНИЕ ИНФОРМАЦИИ В ЭВМ

СИСТЕМЫ СЧИСЛЕНИЯ

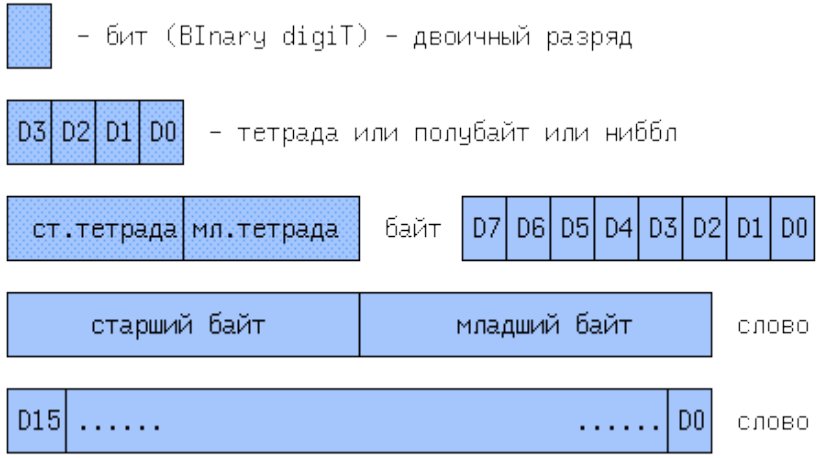
Любое неотрицательное целое число в позиционной системе счисления (СС) может быть представлено в виде:

$$D = C_{n-1}*b^{n-1} + C_{n-2}*b^{n-2} + ... + C_1*b^1 + C_0*b^0$$

где D - десятичный эквивалент числа, Ci - значение i-го разряда, b - основание системы счисления,b в степени i - вес i-го разряда и n число разрядов целой части числа. В вычислительной технике наиболее распространены: двоичная (BIN), десятичная (DEC), шестнадцатиричная (HEX) и непозиционная двоично-десятичная (BCD) системы счисления. В BCD системе вес каждого разряда равен степени 10, как в десятичной системе, а каждая цифра i-го разряда кодируется 4-мя двоичными цифрами. Восьмиричная СС (OCT) применяется реже. Двоичное число 10010011 = $1 * 2^7 + 1 * 2^4 + 1 * 2^1 + 1 * 2^0$ = 147 (DEC). Для перевода числа из двоичной системы в 16 - ную, его необходимо разбить начиная справа на группы по 4 двоичных цифры и каждую группу представить 16 - ной цифрой из таблицы. Для обратного перевода каждая HEX цифра заменяется четверкой двоичных, незначащие нули слева отб- расываются. Двоично-десятичное число можно записывать и десятичными цифрами, например 1997, и двоичными - 0001 1001 1001 0111. Каждое десятичное число можно представить в виде BCD, например 19(DEC) = 19(BCD), но их двоичные представления не равны: 19(DEC) = 10011(BIN), а 19(BCD) = 1 1001(BCD). Не каждая запись из нулей и единиц имеет двоично-десятичный эквивалент. Например, 11001001(BIN) = C9(HEX) = 201(DEC) = ?9(BCD), т.к. десятичной цифры 12 = 1100 несуществует!

МАШИННОЕ ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ

Микропроцессоры обрабатывают упорядоченные двоичные наборы. Минимальной единицей информации является один бит.



Далее следуют - тетрада (4 бита), байт (byte 8 бит), слово (Word 16 бит), двойное слово (DoubleWord 32 бита) или длинное (LongWord 32 бита) и учетверенное слова. **Младший бит обычно занимает крайнюю правую позицию.**

ЧИСЛА С ФИКСИРОВАННОЙ ТОЧКОЙ

Такие числа могут быть как целыми, так и дробными. Точка мысленно фиксируется рядом с любым разрядом. Если она располагается справа от младшего бита, то число целое, если слева от старшего - число дробное. Далее будут рассматриваться только целые числа с фиксированной точкой, для нецелых чисел чаще применяется показательная форма, о которой пойдет речь дальше. Естественным представлением целого неотрицательного числа является двоичная система счисления. Кодирование отрицательных чисел производится тремя наиболее употребительными способами, в каждом из которых **крайний левый бит - знаковый**. **Отрицательному числу соответствует единичный бит, а положительному - нулевой**. Каждый способ оценивается по скорости и затратам на выполнение сложения и изменения знака числа, т.к. вычитание есть сложение с измененным знаком одного операнда.

1. **Прямой код**. Изменение знака производится просто, путем инверсии бита знака. Пусть 00001001 = +9, тогда 10001001 = -9. Если при сложении двух чисел в этом коде знаки совпадают, то трудностей нет. Если знаки различаются необходимо найти наибольшее число, вычесть из него меньшее, а результату присвоить знак наибольшего слагаемого.

2. **Обратный код, инверсный или дополнительный "до 1"**. Изменение знака производится инверсией всех бит: 00001001 = +9, а 11110110 = -9. Сложение также выполняется просто, т.к. знаковые биты можно складывать. При переносе единицы из левого (старшего) бита, она должна складываться с правым (младшим). Например: +7 + (-5) = +2.

```
00000111 = +7
11111010 =-5 (инверсия 00000101 = +5)
1 00000001
1
00000010 = +2
```

Сложение в обратном коде происходит быстрее, т.к. не требуется принятие решения, как в предыдущем случае. Однако суммирование бита переноса требует дополнительных действий. Другим недостатком этого кода является представление нуля двумя способами, т.к. инверсия 0...00 равна 1...11 и сумма двух разных по знаку, но равных по значению чисел дает 1...11.Например: (00001001 = +9) + (11110110 = -9) = 11111111. Кстати, из этого примера понятно почему код называется дополнительным "до 1". Этих недостатков лишен --->

3. **Дополнительный "до 2" или просто дополнительный код**. Число с противоположным знаком находится инверсией исходного и сложением результата с единицей. Например, найти код числа -9.

```
00001001 = 9                    11110111 =-9
11110110 - инверсия             00001000
       + 1                             + 1
11110111 =-9                    00001001 = +9
```

Проблемы двух нулей нет. +0 = 00000000, -0 = 11111111 + 1 = 00000000 (перенос из старшего бита не учитывается).Сложение производится по обычным для неотрицательных чисел правилам.

```
   00001001 = +9
   11110111 = -9
 1 00000000
```

Из этого примера видно, что в каждом разряде двух равных по модулю чисел складываются две единицы (с учетом переноса), что и определило название способа. Этот метод применяется наиболее часто, и **когда говорят о дополнительном коде, то имеется в виду дополнительный "до 2-х" код.**

ДИАПАЗОН ЦЕЛЫХ ЧИСЕЛ С ФИКСИРОВАННОЙ ТОЧКОЙ

Беззнаковые числа: $0<=D<=2^n-1$(n - число разрядов)

Байт: 0 - 255	(DEC)	Слово: 0 - 65535
00..0 - 11..1	(BIN)	00..0 - 11..1
0 - FF	(HEX)	0 - FFFF

Числа со знаком: $-2^{n-1}<=D<=+2^{n-1}-1$(n – число разр.)

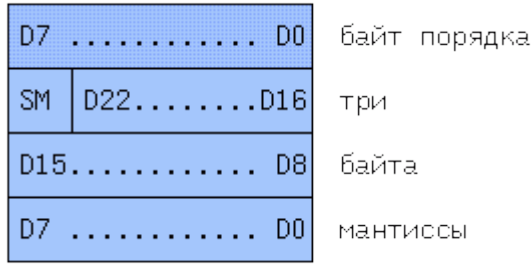
Байт: -128 - +127	(DEC)	Слово: -32768-+32767
10..0 - 01..1	(BIN)	10..0 - 01..1
80 - 7F	(HEX)	8000 - 7FFF

ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ (ВЕЩЕСТВЕННЫЕ)

Вещественные числа хранятся и используются в ЭВМ в **показательной форме**, т.е. в виде двух составляющих: **мантиссы и порядка**. Различия в способах такого представления чисел заключаются в количестве байтов отводимых под порядок и мантиссу и небольших отличиях в форме их хранения. Например в четырехбайтовом формате под мантиссу отводится 3 байта и один байт для хранения порядка (КВ - **короткий вещественный формат**).

$$D = \pm M * 2^{E-127}$$

Последовательность расположения байтов в различных ЭВМ может быть разной.



D - десятичный эквивалент числа, M - нормализованная мантисса, E - порядок, E-127 - смещенный порядок, SM - бит знака мантиссы.

ДИАПАЗОН ПРЕДСТАВЛЕНИЯ ВЕЩЕСТВЕННЫХ ЧИСЕЛ

У нормализованной мантиссы первая значащая цифра (единица) мысленно находится слева от запятой, а справа располагаются 23 разряда - 1,xx..xx. Поэтому Mmax = 1,111..11 = 1 +1/2 +1/4+ 1/8 +...= 2, а Mmin= 1,000..00 = 1 для положительных чисел (SM=0) и -1 и -2 для отрицательных, (SM=1). Порядок числа Emax = 11111110 = 254, а Emin = 00000001 = 1. Теперь нетрудно определить диапазон представления положительных чисел от +Dmax = Mmax * $2^{254-127}$ = 3,4 * 10^{38} до +Dmin = Mmin * 2^{1-127} = 1,17 * 10^{-38}. Точность определяется числом достоверных десятичных цифр. При 23 двоичных разрядах мантиссы 2^{23} примерно равно 10^{7}, т.е. достоверными являются только 6-7 значащих десятичных знаков, а не 38. Необходимо отметить, что значения порядка 11111111 и 00000000 по международному стандарту **IEEE 754 и 854** предназначены для кодирования денормализованных чисел, отрицательной и положительной бесконечностей, неопределенности и, так называемых Не-чисел.

ДВОИЧНО-ДЕСЯТИЧНЫЙ КОД

Двоично-десятичный код (ДДК) или Binary Coded Decimal (BCD) может быть **упакованным**, когда в одном байте хранятся две десятичные цифры, либо **неупакованным** - по одной цифре в байте.Упакованное число 1996 представляется в виде двух байтов: 0001 1001 и 1001 0110. Для знака числа отводится дополнительный байт, например в формате (ДД) девять байтов отводится для размещения 18-ти цифр, а в старшем бите десятого байта находится знак числа.

БУКВЕННО-ЦИФРОВОЙ КОД

Для вывода информации на устройства отображения, например дисплей или принтер, а также для ввода или передачи данных используются буквенно-цифровые коды. Буквы, цифры, математические символы, знаки препинания, символы для рисования линий, управляющие символы и некоторые другие кодируются одно- или многобайтовыми числами. Существует несколько разновидностей однобайтовых кодов, например: ASCII, КОИ7, КОИ8-R, альтернативный код ГОСТ (CP866 или DOS), основной код ГОСТ (ISO-8859-5), Windows 1251 (он же CP1251 или WIN наиболее распространен и в Windows и в web-документах) и другие. ASCII и 7-ми битный код для обмена информацией (КОИ7) отображают первые 128 символов и входят в состав остальных кодировок. Дополнительные символы и русский алфавит входят в восьмибитовые расширенные коды (КОИ8-R, альтернативный, основной и CP1251). Общее число символов в каждой из этих кодировок равно 256. Таблица некоторых 8-ми битных кодов приведена ниже. Следует отметить, что нулевой код (NULL) не кодирует цифру ноль и вообще никак не отображается. Встречаются также обозначения ANSI для "родной" кодировки Windows и OEM для кодировки DOS или Latin-1 для обеих вместе.

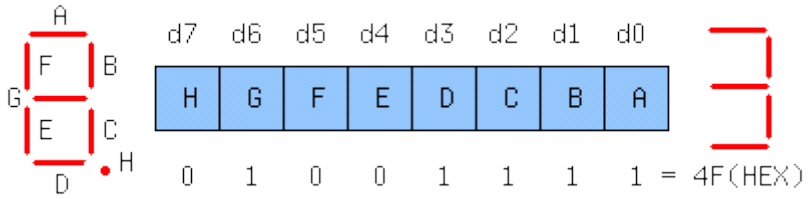
Символ	Код(HEX)	Символ	Код(HEX)	Символ	Код(HEX)
"ничего"	00	"A"	41	"А"	81
"0"	30	"B"	42	"Б"	82
"1"	31	"C"	43	"В"	83
..	..	..	..	..	..

```
"9"     ¦ 39        "Z" ¦ 5A        "Я" ¦ 9F
":"     ¦ 3A        "[" ¦ 5B        "а" ¦ A0
\______ASCII кодировка_____/
\__________альтернативная кодировка_____/
```

В настоящее время, наряду с приведенными кодировками, широко используются разновидности Уникода (Unicode): UTF-32 4-х байтовый код, а также UTF-16 и UTF-8 с переменным числом байтов и другие. Первый байт может иметь вид 11xxxxxx, а остальные 10xxxxxx.

ВОСЬМИСЕГМЕНТНЫЙ КОД

Служит для отображения образа BCD или HEX цифры высвечиваемой на индикаторе в виде набора 0 и 1. Может быть принято следующее соответствие между битами и сегментами:



Внизу приведен битовый набор для высвечивания цифры 3. В приведенном примере единицы соответствуют светящимся сегментам.

НЕОДНОЗНАЧНОСТЬ ПРЕДСТАВЛЕНИЯ ДВОИЧНЫХ НАБОРОВ

Набор единиц и нулей хранящихся в регистре или ячейке памяти (двоичный набор) для микропроцессора ничего не означает. Пусть в регистре находится набор из восьми битов **10000110**. Он может быть интерпретирован как:

1) двоичное число = 10000110, имеющее
   а) шестнадцатиричный эквивалент = 86(HEX),
   б) восьмиричный эквивалент = 206(OCT),

в) десятичный эквивалент числа без знака = 134(DEC),
2) дополнительный код отрицательного числа =-122(DEC),
3) двоично-десятичное упакованное число = 86(BCD),
4) альтернативный код буквы "Ж",
4') код КОИ-8 символа "╪",
5) восьмисегментный код цифры "1.",
6) часть вещественного числа,
7) реализация множества,включающего 3 элемента из 8-ми,
8) часть адреса ячейки памяти или внешнего устройства,
9) код операции
10) и т.д.

Поэтому **ответственность за интерпретацию двоичных наборов возлагается на программиста**. Например, попытка сложить ASCII коды "1" + "2" не даст в сумме код "3", а даст 31(HEX) + 32(HEX) = 63(HEX), что соответствует коду латинской буквы "c".

МИКРОПРОЦЕССОР 8086(88)

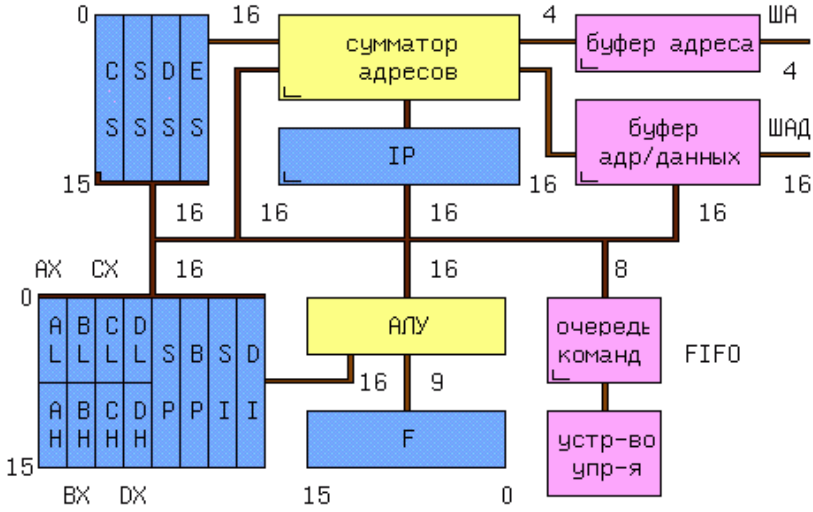
СТРУКТУРНАЯ СХЕМА МИКРОПРОЦЕССОРА

Огромное количество микропроцессоров (МП) не позволяет рассмотреть их особенности, поэтому был выбран родоначальник семейства 80x86 : МП К1810ВМ86/88 (8086/8088). Такой выбор оправдан, во-первых преобладающим числом ЭВМ с этим МП, во-вторых тем, что все МП этого семейства при включении начинают работу в реальном режиме МП 8086, и в-третьих - программной совместимостью их ассемблеров снизу вверх. Ниже на рисунке приведена структурная схема МП8086 и внешний вид типового микропроцессора.

Устройство управления декодирует байты программы и управляет работой **операционного устройства и шинного интерфейса.** Операционное устройство МП состоит из 4-х шестнадцатиразрядных **регистров общего назначения: РОН** (AX, BX, CX, DX), из 4-х регистров указателей (адресных регистров SP,BP,SI, DI) и **арифметико-логического устройства (АЛУ)** с **регистром признаков операций (флагов F).**

РОН служат для хранения промежуточных результатов операций, т.е. операндов. Помимо общих, каждый из этих регистров имеет и некоторые **специальные функции**, о которых будет сказано далее. Каждый РОН может рассматриваться, как состоящий из двух

независимых 8-ми разрядных регистров AH, AL, BH, BL, CH, CL и DH, DL.



Адресные регистры хранят 16-ти битовые указатели (адреса) на области памяти. **В SP (StackPointer) находится текущий адрес "вершины стека"** - специально организованной области памяти, которая будет описана далее. Регистр BP (Base Pointer) хранит любой **базовый адрес** в области стека. Два регистра SI и DI (Source и Destination Index) **адресуют области памяти, называемые источником и приемником данных.**

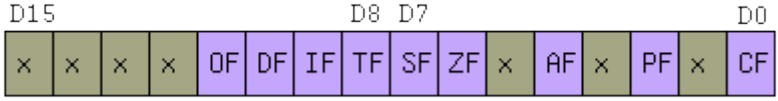
Шинный интерфейс, на рис.1 его узлы отмечены двойной чертой выполняет операции обмена между МП и памятью или внешними устройствами с помощью шин адреса и данных ША и ШАД. В **сегментных регистрах** CS, SS, DS, ES хранятся указатели на 64-х килобайтные области памяти называемые сегментами. Значения этих указателей могут перекрываться. Адрес байта в ячейке памяти получается суммированием содержимого одного из сегментных регистров и одного из регистров (SP, SI, DI, P). Например адрес текущего байта программы, извлекаемого из памяти определяется суммой адресов, хранящихся в регистре кодового сегмента (CS) и указателе команд (IP). В регистре IP хранится 16-битовый адрес байта в кодовом сегменте к которому микропроцессор дожен обратиться. Подробнее об этом будет сказано далее. **Связь с внешними устройствами осуществляется через специальные**

тристабильные схемы с повышенной нагрузочной способностью и называемые буферами.

Текущий байт программы, извлекаемый из памяти направляется в очередь команд: шесть однобайтовых регистров расположенных **конвейером (по принципу "первым вошел - первым вышел" или FIFO**). Конвейер позволяет одновременно выполнять команду из очереди и загружать следующую, повышая производительность МП.

РЕГИСТР ФЛАГОВ (ПРИЗНАКОВ)

По результатам операций АЛУ устанавливает либо сбрасывает отдельные биты в регистре флагов F**.**



x обозначает, что содержимое этого бита не имеет значения (don't care bit). Некоторые операции влияют только на отдельные флаги, а другие совсем на них не воздействуют, поэтому при описании флагов подразумевается выполнение тех команд (операций), которые влияют на эти флаги. В дальнейшем, в тексте, фраза "содержимое XX" будет обозначаться круглыми скобками - (XX).

- ZF - флаг/признак нулевого результата Zero, устанавливается в 1, если в результате выполнения команды, влияющей на этот флаг получен нулевой результат, иначе (ZF)=0.
- CF - флаг переноса Carry устанавливается, если в результате выполнения операции из старшего бита переносится или занимается 1 при сложении или вычитании, иначе (CF)=0. На CF влияют также команды сдвига и умножения.
- SF - флаг знака результата Sign равен единице, если результат отрицательный, т.е. он дублирует старший знаковый бит результата.
- PF - флаг четности Parity. (PF)=1, если сумма по модулю два всех битов результата равна нулю (число единичных битов - четное).
- AF - флаг дополнительного переноса Auxiliary устанавливается, если есть перенос из старшего бита младшей

тетрады (бит D3) в младший бит старшей тетрады (бит D4). Используется в операциях над упакованными BCD числами.

- OF - флаг переполнения Overflow устанавливается, когда результат операции превысит диапазон чисел со ЗНАКОМ, а также в некоторых других случаях. Другое определение: (OF)=1, если перенос/заем в старший бит результата не равен переносу/заему из старшего бита.

Рассмотрим в качестве примера сложение двух однобайтовых чисел: 125 + 4 = 129 выходит за пределы -128.. ..+127 чисел со знаком (для беззнаковых чисел 129 - корректный результат).

```
  0 1 1 1 1 1 0 1 = 125
  0 0 0 0 0 1 0 0 = 4
  1 0 0 0 0 0 0 1 = 129 (или -127 ???)
 / / :
0 1  :
```

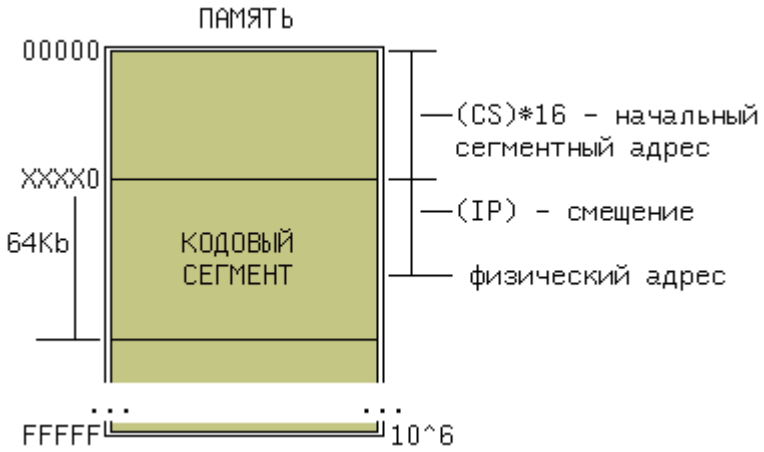
Перенос в бит D7 равен 1, а из бита D7 = 0, в результате сложения чисел (OF) = 1, (CF) = 0, (ZF) = 0, (SF) = 1, (PF) = 1, (AF) = 1. Остальные три флага будут рассмотрены далее. Содержимое регистра признаков называется также **словом состояния процессора (программы) и обозначается PSW (Processor или Program Status Word)**.

ОРГАНИЗАЦИЯ ПАМЯТИ И ВЫЧИСЛЕНИЕ АДРЕСА

МП 8086 имеет 20-ти разрядную шину адреса ША, позволяющую обращаться к 2^20 или примерно к одному миллиону ячеек памяти. 16-ти битовая шина данных ШД может пересылть информацию байтами или словами. Память обычно организована в виде линейного одномерного массива байтов, причем два соседних байта могут рассматриваться как слово. Все мегабайтное пространство памяти условно делится на 16 сегментов объемом по 64Kb. Микропроцессору доступны в каждый момент четыре - кодовый сегмент, где хранится программа, стековый сегмент, сегмент данных программы и дополнительный сегмент данных. Начальные адреса этих сегментов хранятся в регистрах CS,SS,DS и ES. Так как эти регистры 16-ти битовые, а все адресное пространство 20-ти битовое, то МП

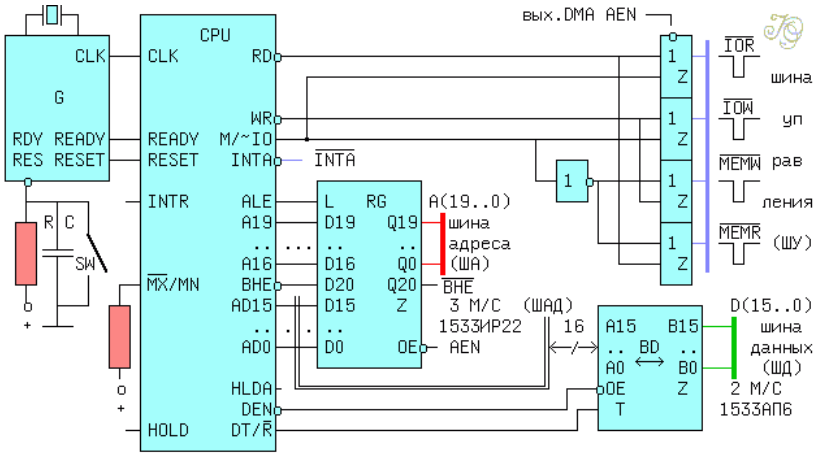
начальный сегментный адрес в 20-ти битовом сумматоре сдвигает на четыре бита влево (эквивалентно умножению на 16) и складывает с содержимым одного из регистров (IP, SP, DI, SI).

Полученное число называется физическим адресом. Например, извлекая из памяти очередной байт кода программы МП формирует физический адрес по формуле: Физический адрес = (IP) + (CS) * 16, где (IP) - смещение, эффективный или исполнительный адрес, (CS) - сегментный адрес, а (CS) * 16 - называется начальным сегментным адресом. Организация памяти приведена на рисунке.



ПРОЦЕССОРНЫЙ БЛОК (ПБ)
Система с раздельным адресным пространством ЗУ и ВУ

ПБ содержитт микропроцессор, стабилизируемый кварцем генератор импульсов, два устройства для формирования адресных и управляющих сигналов и двунаправленный буфер шины данных. Схема ПБ представлена на рис.3. Для уменьшения общего количества выводов МП, по некоторым из них в разные моменты передаются разные сигналы, поэтому младшие 16 линий адреса и шина данных совмещены (мультиплексированы).

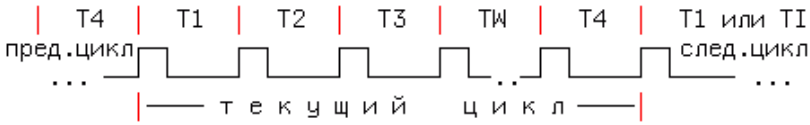


Конденсатор С до включения питания разряжен. В начальный момент времени t0 подается напряжение питания на выводы (+). Начиная с этого момента времени на входе ~RES действует логический 0 до момента t1, когда напряжение на конденсаторе С достигнет порогового значения логической 1. В течение интервала t0..t1 выходной сигнал RESET имеет активный уровень лог.1. Микропроцессор переводится в исходное состояние. При этом в регистр CS записывается код FFFFh, а в регистр IP код 0000h.

Когда С зарядится и сигнал ~RES станет "1", на входе RESET микропроцессора сигнал снизится до пассивного уровня. Начинается основной цикл работы. МП извлекает из ячейки памяти с адресом FFFF:0000 первый байт команды, который чаще всего является кодом безусловного перехода к тому месту в памяти, где находится начало программы. В процессе работы можно перезапустить МП с помощью кнопки SW. Конденсатор С разряжается до нуля и процесс запуска повторяется. МП может работать в двух режимах в максимальном и минимальном. Минимальный режим ограничивает адресуемый объем памяти до 64Kb. В большинстве приложений в приборостроении такого объема достаточно, поэтому сигнал ~MX/MN = 1. Формирователи сигналов ШУ,ША и ШД (шины управления адреса и данных), выполнены на элементах ИЛИ, регистрах и шинных формирователях и служат также для увеличения мощности этих сигналов.

Генератор G формирует последовательность импульсов CLK, называемых тактовыми. Выполнение команды производится в течение

интервалов времени, называемых циклами. Если в цикле есть обращение к памяти или к внешним устройствам, то он называется циклом шины. Цикл шины содержит 4 обязательных такта T1 ... T4.



В такте T1 микропроцессор передает по совмещенной шине адрес/данные адрес ячейки памяти или **внешнего устройства (ВУ), подключенного к шинам ШУ,ША и ШД**. В такте T2 производится выбор направления обмена данными с памятью или ВУ, а в тактах T3,T4 - передача данных. Если ЗУ (запоминающее устройство) или ВУ медленные, то на вход готовности RDY посылается сигнал RDY = 0, по которому МП вставляет циклы ожидания TW, до тех пор, пока не будет установлена готовность ВУ или ЗУ (RDY = 1). Если в цикле нет обращения к шине, то МП формирует холостые циклы TI.

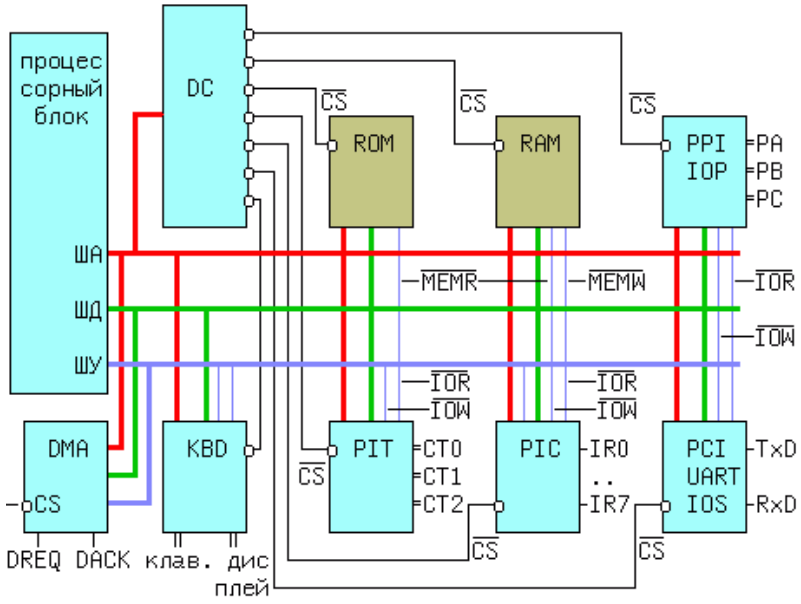
Для разделения сигналов **совмещенной шины адрес/данные ШАД(AD15..0)** их необходимо "демультиплексировать" с помощью регистра защелки адреса RG и двунаправленного буфера BD. При обращении к памяти (в том числе при выборке команды) микропроцессор передает по ШАД адрес ячейки памяти. Этот адрес записывается в D-триггеры регистра RG сигналом ALE генерируемым микрпроцессором в этот момент и поступающим на синхровходы D-триггеров. Адрес в регистре сохраняется на время последующей передачи данных. Следом по ШАД передаются, либо данные от микропроцессора к ВУ или ЗУ, либо в обратном направлении. МП должен, во-первых, обеспечить правильное направление передачи буфера BD и , во-вторых, открыть (разрешить) тристабильные элементы буфера для передачи данных. Первую задачу решает сигнал МП ~DT/R (~DT/R=0 передача данных от МП - Transmit, ~DT/R=1 прием данных МП - Receive).

Вторая задача решается генерацией МП сигнала ~DEN (Data Enable). Чтение или ввод данных в один из регистров МП осуществляется с помощью инверсных сигналов шины управления (ШУ): ~MEMR (чтение из памяти),~IOR (ввод из ВУ), называемыми еще **стробами чтения**. Запись или вывод данных из МП по шине данных сопровождается **стробами записи** ~MEMW (запись в память (ЗУ)), или ~IOW (вывод во внешнее устройство (ВУ)). Четверка стробов, которые являются основными сигналами шины управления, формируется из сигналов чтения, записи (~RD,~WR) и сигнала M/~IO,

определяющего к чему производится обращение : к ЗУ или ВУ. Формирование этих сигналов производится с помощью простой комбинационной схемы, содержащей 4 элемента ИЛИ и один инвертор, причем элементы ИЛИ выполняют в нашем случае функцию И для инверсных значений стробов.

МИКРОПРОЦЕССОРНАЯ СИСТЕМА С ШИННЫМ ИНТЕРФЕЙСОМ

Работой всех устройств подключаемых к процессорному блоку управляет 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 и ОЗУ, другие только несколько адресов.



Типовая МПС содержит:
микросхему **программируемого периферийного интерфейса ППИ (PPI или IOP)**, к которой через три 8-битовых независимых канала PA,PB и PC можно подключать периферийные устройства, например принтер, клавиатуру, 8-ми сегментный дисплей или ЦАП и АЦП. Через ППИ может производиться обмен данными с другими МПС или ЭВМ.

Ввод с клавиатуры и вывод на дисплей могут производиться специальными микросхемами.
Связь с удаленными устройствами или абонентами сетей может осуществляться с помощью **универсального асинхронного последовательного приемо-передатчика УАПП-UART (программируемый связной интерфейс ПСИ-PCI или IOS)**. К выводам RxD - приемник и TxD - передатчик через линию связи подключаются передатчик и приемник другого абонента или устройства. Если связь производится через модем, то доступны любые сети.

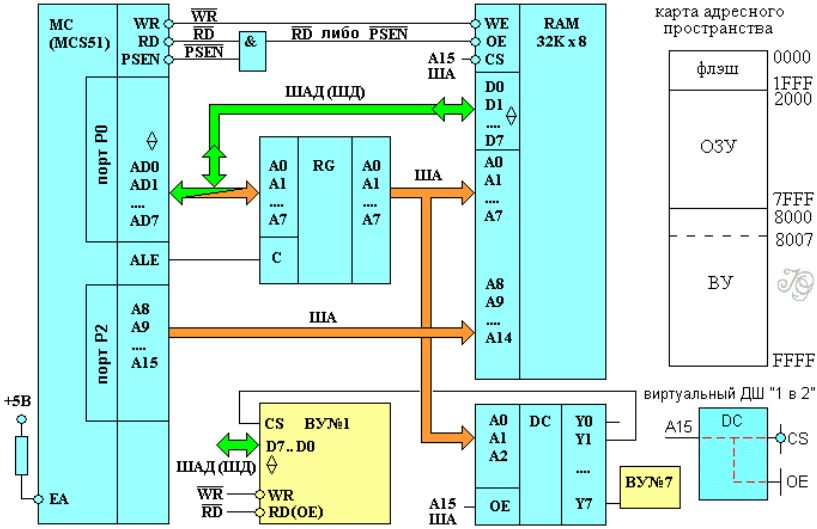
Для формирования точных, различных по частоте и длительности сигналов, используется **программируемый интервальный таймер ПИТ-PIT**, имеющий три независимых 16-ти разрядных двоичных счетчика. Задержка, длительность или частота выходного сигнала каждого счетчика кратна 3..65535 периодам входного сигнала.
Если в системе используется режим прерывания выполнения основной программы внешними устройствами, требующими безотлагательного вмешательства микропроцессора, то может применяться **программируемый контроллер прерываний ПКП - PIC** (устройство собирающее заявки на обслуживание от ВУ с входов IRi). Подробно прерывания будут рассмотрены ниже.

Обмен данными между МП и ЗУ или одним из ВУ возможен только при появлении на выходе дешифратора DC сигнала ~CSi = 0, поступающего на то устройство с которым будет производиться запись или чтение байта данных. Остальные (невыбранные) устройства будут в пассивном состоянии, т.к. их сигналы ~CSj = 1. Байт информации на ШД считывается ВУ, ЗУ или МП в строго ограниченном интервале времени во время действия одного из управляющих сигналов чтения/записи (~MEMR, ~MEMW) из памяти или в память, или во время действия одного из сигналов управления вводом/выводом (~IOR, ~IOW) в/из ВУ.
Быстрый обмен данными может производиться с помощью устройства **прямого доступа к памяти ПДП** (DMA).

МИКРОКОНТРОЛЛЕРНАЯ СИСТЕМА С ШИННЫМ ИНТЕРФЕЙСОМ

Система с общим адресным пространством ЗУ и ВУ

На рисунке ниже приведена стандартная схема микроконтроллерной системы с шинным интерфейсом на основе МК семейства MCS-51 (8051). В состав системы входят: 1) внешние устройства (ВУ), обеспечивающие ее функциональность в соответствие с техническим заданием, 2) память данных и/или программ (ОЗУ), 3) дешифратор адресов ВУ и 4) регистр-защелка адреса. Здесь же приведена карта адресного пространства 0000..FFFF(64KB) для МК ADuC812 с 8-мью килобайтами внутренней флэш памяти (адреса 0000..1FFF).

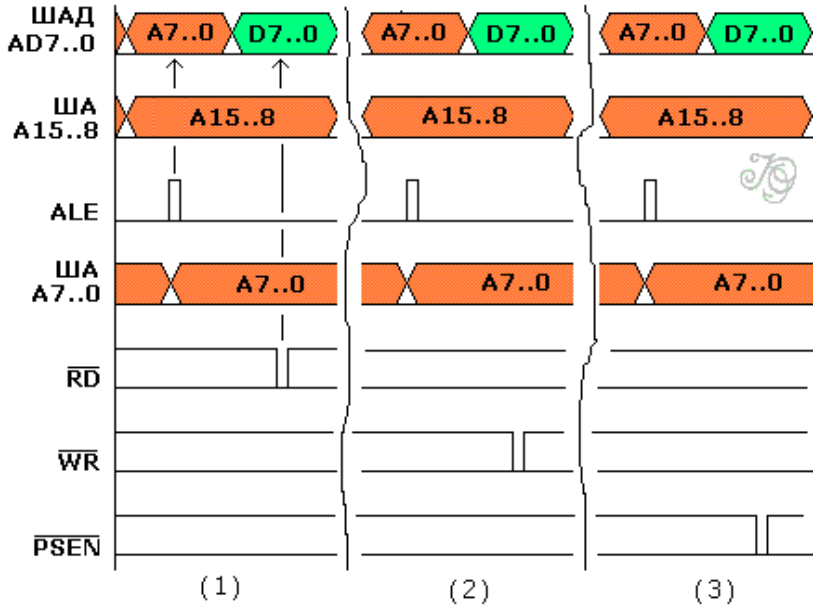


Все адресное пространство 0000..FFFF поделено на две равные области линией ША A15, которая вместе с входом ~CS микросхемы ОЗУ и входом OE дешифратора DC образует виртуальный дешифратор "1->2". Когда A15=0, то ~CS=0 и операция чтения или записи производится с памятью, сигнал OE=A15 также равен нулю, поэтому на всех выходах дешифратора пассивный нулевой уровень и операции с ВУ невозможны, т.к. CSi=Yi=0. И наоборот, когда A15=1, то ~CS=1 и обращение к памяти невозможно. Работа дешифратора ВУ разрешена по входу OE=A15=1 и в зависимости от значений A2,A1,A0

производится операция с одним из 8-ми ВУ. Из рисунка видно, что реально для ВУ задействованы 3 линии ША A2,A1,A0 на адресных входах дешифратора и линия A15=1 на входе OE, остальные 12 бит адреса не используются: 1xxx xxxx xxxx xA2A1A0(BIN). Поэтому при обращении к ВУ неопределенные биты 'x' могут иметь любые значения, например при x=0 восемь адресов ВУ будут находиться в диапазоне 1000 0000 0000 0000 .. 1000 0000 0000 0111 или 8000h .. 8007h (080000h .. 080007h в лабораторных работах №11 и №12 стр.4).

Как уже говорилось, для уменьшения общего числа выводов МК, некоторые сигналы объединены на одном выходе (мультиплексированы). Например младший байт адреса A7..0 мультиплексирован с байтом данных D7..0 в совмещенной шине адреса-данных AD7..0. В процессе выполнения программы при обращении к ВУ или внешнему ЗУ, МК помещает сначала на совмещенную шину адреса-данных младший байт адреса A7..0, который записывается адресным стробом ALE в регистр-защелку адреса. Одновременно старший байт адреса A15..8 выводится на одноименные выходы МК. Далее возможны три варианта: 1) чтение байта данных по ШД из ОЗУ или из ВУ, 2) запись байта данных в ОЗУ или в ВУ, 3) выборка (чтение) байта программы из ОЗУ. Для чтения/записи этих байтов МК генерирует один из трех стробов: чтения байта данных ~RD, записи байта данных ~WR или чтения байта программы ~PSEN. Так как у ОЗУ только один вход (~OE) для строба чтения, то инверсные стробы чтения ~RD и ~PSEN объединяются по "ИЛИ" с помощью логического элемента "И" (сигнал на выходе логического элемента "И" равен 0, если равен 0 ИЛИ один входной сигнал ИЛИ другой ИЛИ в общем случае все остальные в разных комбинациях, т.е осуществляется логическое "ИЛИ" для нулевых входных сигналов в полном соответствии с теоремой двойственности).

Ниже приведены упрощённые фрагменты временных диаграмм для циклов чтения байта данных D7..0 в МК, записи байта данных в ВУ или ОЗУ и чтения байта программы из ОЗУ.



Пример: 1) Нужно записать байт 55(HEX) в ВУ №3. 2) Байт из ВУ №3 нужно поместить в переменную "v". Код на языке С (в среде Keil uVision) и сгенерированный код на ассемблере будут выглядеть следующим образом:

исходный код на Си

```
xdata char vu3 _at_ 0x8003;
. . . . . .
vu3=0x55;
```

соответствующий фрагмент на ассемблере

```
ORG 8003h
vu3: ds 1 ; однобайтовая переменная vu3 по адресу 8003h
. . . . . .
mov dptr, #vu3
mov a, #55h
movx @dptr, a
```

Сначала в регистр DPTR (Data Pointer), указывающий на ячейку внешней памяти или внешнее устройство помещается адрес ВУ№3 (8003 HEX), а в аккумулятор операнд 55(HEX). Во время выполнения команды MOVX @DPTR, A микроконтроллер сгенерирует последовательность сигналов, соответствующую фрагменту №2 временных диаграмм и байт 55h будет записан в ВУ№3.

Во втором случае для чтения байта из ВУ микроконтроллер во время выполнения команды MOVX A, @DPTR сформирует последовательность №1.

исходный код на Cu

```
xdata char vu3 _at_ 0x8003;
data char v;
. . . . . .
v=vu3;
```

соответствующий фрагмент на ассемблере

```
ORG 8003h
vu3: ds 1 ; однобайтовая переменная vu3 по адресу 8003h
. . . . . .
MOV       DPTR,#vu3
MOVX      A,@DPTR
MOV       v,A
```

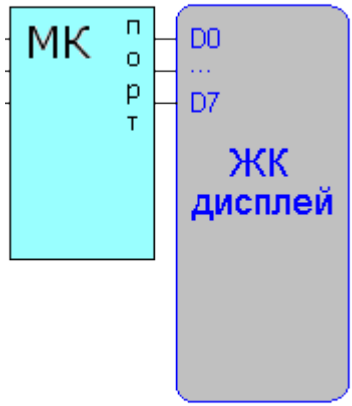
Из приведенных примеров видно, что действие команд MOVX @DPTR,A и MOVX A,@DPTR (семейство MCS-51) аналогично действию команд OUT DX,AL и IN AL,DX (семейство x86).

ШИНА VS ПОРТ

1) Обмен данными может производиться или с помощью только что рассмотренного шинного интерфейса (например, вывод байта в порт данных жидкокристаллического дисплея):

MOV DPP, #НОМЕР СТРАНИЦЫ ПАМЯТИ
MOV DPTR, #АДРЕС РЕГИСТРА ДАННЫХ ЖКД
MOV A, #БАЙТ
MOVX @DPTR,A; переслать байт в регистр данных ЖК дисплея, т.е. на входы данных ЖКД

2) Или с помощью портов ввода-вывода (например, вывод байта в порт данных жидкокристаллического дисплея):



MOV PORT, #БАЙТ; переслать байт в порт, т.е. на входы ЖКД

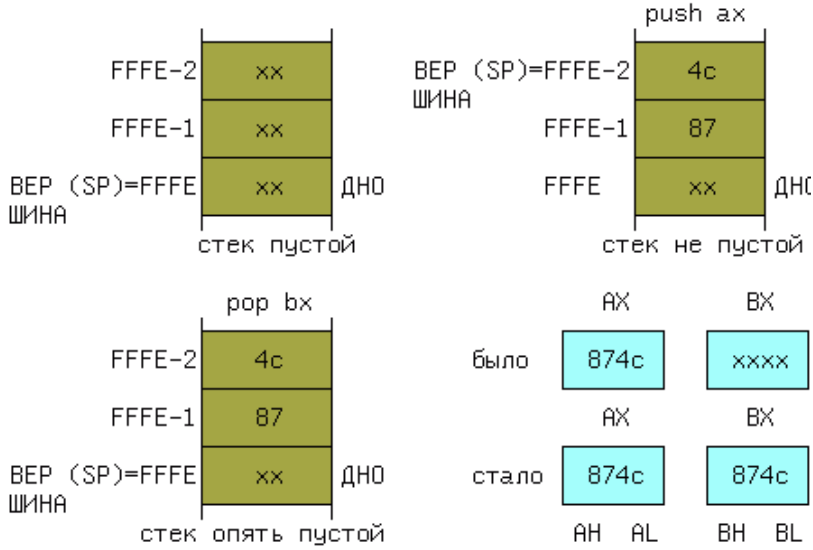
Вывод в пользу второго варианта напрашивается сам собой. И аппаратных средств меньше и цепочка команд короче. Но не все так просто. Если число требуемых внешних устройств (ЖКД, клавиатура,

8-ми сегментный дисплей, шаговый двигатель, датчики и т. д.) невелико – предпочтительнее вариант с прямым подключением ВУ к портам МК. Если имеющихся портов недостаточно, необходим первый вариант.

СТЕК

Область оперативной памяти с упрощенной схемой адресации, к которой МП обращается по принципу "последним вошел - первым вышел" (LIFO). Не путать с FIFO - то есть с очередью или конвейером. Байты программы в оперативной памяти располагаются последовательно по нарастающим адресам. Стек обычно заполняется по последовательно убывающим адресам. Во избежание перекрытия этих двух областей памяти стек располагается в старших адресах.

Начальный адрес стека, называемый дном (bottom) записывается в регистр SP командой MOV. Например: MOV SP,0fffeh. Вместо 0fffeh - адрес предпоследнего байта сегмента, может быть другое значение, но выравненное по двухбайтовым, т. е. четным адресам. Текущее значение содержимого SP называется, также адресом вершины стека (top). Если адрес вершины совпадает с адресом дна - стек считается пустым. Рассмотрим механизм помещения в стек и извлечения из него данных на примере команд PUSH AX и POP BX. Пусть начальное значение аккумулятора AX равно 874c.



Команда PUSH выполняется в четыре этапа:

- Адрес в SP уменьшается на 1: (SP) <-- (SP) - 1.
- По этому адресу помещается старший байт 87: ((SP)) <-- (AH).
- Содержимое SP снова уменьшается на 1: (SP) <-- (SP) - 1.
- По полученному адресу загружается младший байт 4c: ((SP)) <-- (AL).

Действие команды POP аналогично описанному процессу, но происходит в обратном порядке:

- (BL) <- ((SP)),
- (SP) <- (SP) + 1,
- (BH) <- ((SP)),
- (SP) <- (SP) + 1.

Байты в стек помещаются по правилу "старший байт по старшему адресу" . На рис.7 показан пустой стек до выполнения команды PUSH AX и после ее выполнения, а на рис.8 после выполнения команды POP BX.

Преимущество стека в том, что программисту не нужно заботиться об абсолютных значениях адресов переменных, но в этом таится и опасность, если текущее содержимое указателя стека будет потеряно, при неаккуратных действиях программиста, то работа компъютера станет непредсказуемой и он, как говорят в таких случаях, "зависнет". В программах стек используется для:
1) автоматического сохранения и извлечения адреса возврата из подпрограмм командами ассемблера: CALL (вызов подпрограммы), RET (возврат из подпрограммы) и IRET (возврат из подпрограммы обработчика прерывания), а также для хранения содержимого регистра флагов (PSW).
2) хранения локальных переменных (трансляторами с языков высокого уровня),
3) передачи фактических параметров подпрограммам (трансляторами с языков высокого уровня),
4) временного хранения содержимого регистров фоновой программы при ее прерывании.

СПОСОБЫ ВВОДА-ВЫВОДА

Обмен данными с участием внешних устройств (ВУ) называется вводом-выводом (ВВ). Существует четыре основных способа ВВ.

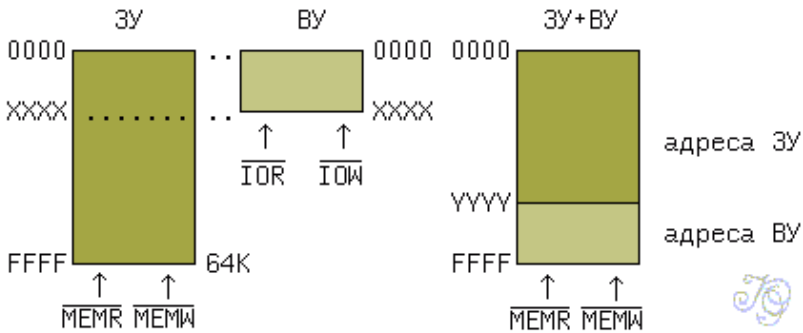
- Программный ВВ
- ВВ по прерываниям
- Прямой доступ к памяти (ПДП) или DMA
- Транзакции (MCS-96)

В первых двух случаях в обмене данными участвует микропроцессор. В режиме ПДП функции управления обменом берет на себя специальное устройство - контроллер ПДП, причем МП в это время в обмене данными не участвует. В 4-ом способе пересылки производятся параллельно с работой МП периферийным сервером транзакций.

ПРОГРАММНЫЙ ВВОД-ВЫВОД

Для внешних устрйств выделяется, либо отдельное от ЗУ адресное пространство, либо совместное с ЗУ. Следовательно программный ВВ может быть двух типов:

1) Карта адресного пространства рамером 64K в первом случае показана на рис.5 слева.



В пределах интервала 0000 ...XXXX адреса ВУ и ЗУ перекрываются. Поэтому для однозначного обращения к ячейке памяти или ВУ,

имеющих одинаковый адрес, в процессорном блоке формируются 2 пары управляющих пересылкой стробирующих импульсов - ~IOR,~IOW для ввода или вывода данных во внешнее устройство и ~MEMR,~MEMW для чтения или записи в память. Обмен данными с ВУ при этом способе производится с помощью команд ассемблера: ввод (IN) и вывод (OUT), во время выполнения которых и формируются стробы ~IOR либо ~IOW. Обращение к ячейкам памяти осуществляется с помощью обширной группы команд пересылки, например MOV [BX],AL или MOV AL,[BX], во время выполнения которых генерируются стробы ~MEMW либо ~MEMR. Емкость ЗУ для размещения программ и данных не уменьшается. Этот способ применяется в семействе МП x86.

2) Карта распределения памяти для второго случая показана на рисунке справа. Адреса ВУ и ЗУ находятся в одном адресном пространстве. Для обращения к ВУ или ЗУ нужны **только** два строба ~MEMR, ~MEMW (на самом деле достаточно стробов ~RD и ~WR см. рис.4.1). Обмен данными при втором способе производится с использованием всех команд пересылки ассемблера (MOV, LODSB,..). Емкость ЗУ уменьшается на количество адресов отводимых для ВУ. Второй способ позволяет адресоваться к ВУ с помощью команд оперирующих с памятью и применяется в семействе МК MCS-51, но может использоваться и в семействе x86.

Основное достоинство программного ВВ в простоте. Но при выполнении ввода, например с клавиатуры, МП затрачивает до 99,99..% времени на ожидание, не выполняя при этом другой полезной работы. Избавиться от этого недостатка позволяет ВВ по прерываниям.

ВВОД-ВЫВОД ПО ПРЕРЫВАНИЮ

В общем случае прерывания работы микропроцессора могут вызываться :

- внешнними устройствами (внешние или аппаратные прерывания),
- командами прерываний (программные прерывания)
- автоматически самим МП (внутренние прерывания), например при попытке деления на 0.

В этом разделе будут рассмотрены внешние прерывания микропроцессора 80x86. Работу МП можно разделить во времени между двумя независимыми программами: **фоновой**, которая выполняет основную задачу и программой ВВ данных. Когда ВУ подготовит данные для передачи или подготовится для их приема, оно посылает сигнал запроса на прерывание непосредственно на вход МП INTR или в специальное устройство - контроллер прерываний. В процессе обслуживания прерывания МП и ВУ автоматически выполняют следующие действия:

1. ВУ самостоятельно, либо через контроллер прерываний посылает запрос на прерывание INT(R) - Interrupt Request на одноименный вход МП;
2. МП завершает выполнение текущей команды и, если преывания разрешены командой ассемблера STI (т.е. флаг прерывания (IF)=1), подтверждает разрешение сигналом ШУ - INTA. Фактически этот сигнал является стробом чтения номера прерывания N;
3. Внешнее устройство самостоятельно или через контроллер прерываний по ШД передает в МП однобайтовый тип(номер) прерывания - N;
4. Содержимое PSW (флаги), а также CS,IP (адрес возврата в фоновую программу), скорректированное с учетом сброса очереди помещается в стек. Адрес возврата - адрес команды ассемблера, перед которой наступило прерывание ;
5. Сбрасываются флаги IF (флаг разрешения прерываний) и TF (флаг трассировки), причем т.к. (IF) теперь равен 0, то дальнейшие прерывания запрещаются;
6. В IP загружается содержимое двух байтов с начальным адресом 4*N, а в CS - содержимое следующих двух байтов . Эти 4 байта являются адресом подрограммы прерывания и называются **вектором (указателем) прерывания.**
7. Начинает выполняться подпрограмма - обработчик прерывания.

```
  INT_SUBR:
      STI
      PUSH AX
      ....;    здесь
      ....;    команды
      MOV AL,5; обработчика
....;  прерывания
....
```

```
POP AX
IRET
```

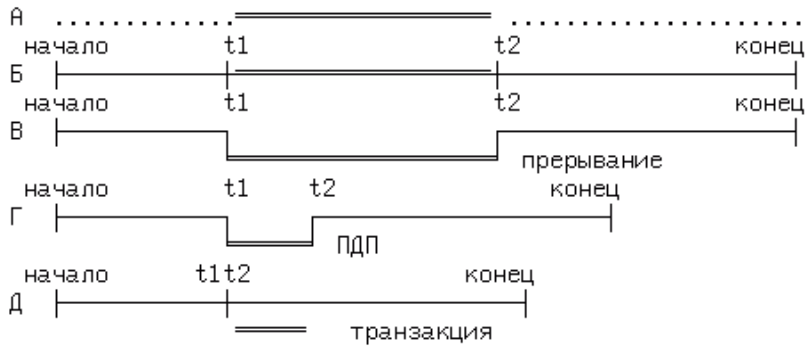
Если допускаются вложенные прерывания, то вначале помещается команда STI- разрешение преываний, запрещенных в п.5. Инструкции push и pop сохраняют и восстанавливают содержимое регистров фоновой задачи, если эти же регистры используются и обработчиком прерывания (в примере регистр AX).

8. Команда IRET извлекает из стека адрес возврата - IP, CS и содержимое PSW;

9. МП продолжает работу с адреса возврата. При выполнении программных прерываний по команде INT N действия выполняются с п.3. N находится в пределах 0 <= N <= 255, поэтому четырехбайтовые вектора прерываний занимают первые 1024 байта памяти.

ПРЯМОЙ ДОСТУП К ПАМЯТИ (ПДП) И ТРАНЗАКЦИИ

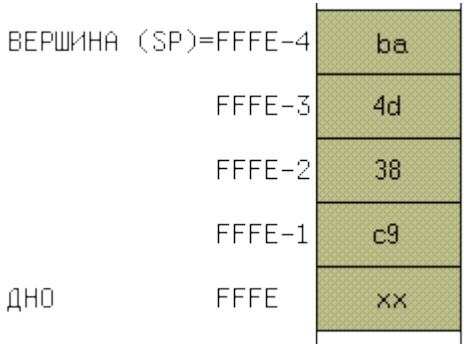
Обмен большим количеством байтов, между ВУ (например дисковым накопителем) и памятью с помощью предыдущих двух методов малоэффективен, т.к. обмен происходит по цепочке: ВУ - аккумулятор (AX или AL) - память или наоборот. В режиме ПДП при поступлении запроса от ВУ на вход HOLD, МП разрешает обмен выходным сигналом HLDA. Микропроцессор на время обмена отключается от ШУ,ШД и ША переводя их в третье состояние по входам ~OE буферных элементов сигналом ~BUSEN = 1. Специальная микросхема (контроллер ПДП) использует освободившиеся шины для высокоскоростного прямого обмена ВУ - память. Скорость обмена достигает многих мегабит/сек



На рисунке показан процесс выполнения основной (фоновой) программы - интервалы времени (начало..t1, t2..конец) и выполнение процедуры передачи массива данных, на рисунке этот отрезок времени обозначен двойной линией. На диаграмме (А) ЭВМ задействована только для передачи (отрезок t1..t2), в остальное время компьютер бездействует. Во втором варианте - диаграмма (Б), код программы передачи жестко встроен в фоновую задачу. В третьем варианте (В) передача массива оформлена в виде подпрограммы прерывания, причем если запроса на прерывание не поступит,то суммарное время на выполнение фоновой программы уменьшится на t2-t1. При использовании режима ПДП сохраняются преимущества метода прерывания, время на передачу сокращается - диаграмма (Г), но фоновая задача по прежнему прерывается. В последнем случае передача данных производится почти без нарушения хода программы параллельно во времени (Д). Транзакции реализованы в некоторых семействах однокристальных микроЭВМ, например в MCS-96.

ЗАДАЧИ И УПРАЖНЕНИЯ

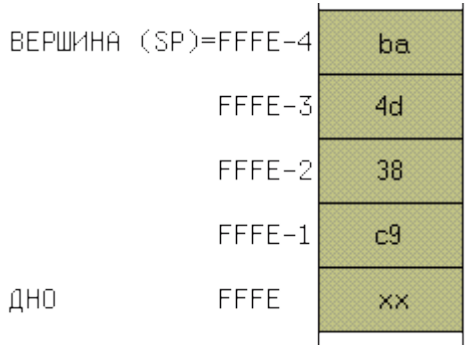
Одна подпрограмма вызывает другую. Укажите короткий адрес возврата из вложенной процедуры.



Пояснение: Вложенная подпрограмма это подпрограмма, которая вызывается из другой подпрограммы. Вызов подпрограммы сопровождается помещением в стек адреса возврата. Стек заполняется начиная с дна. Короткий адрес - двухбайтовый адрес в пределах

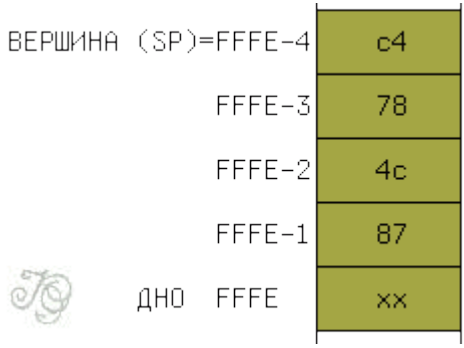
одного кодового сегмента. Байты в стек помещаются по правилу "старший байт по старшему адресу". Теперь нетрудно ответить на предложенный вопрос: **4dba.**

В тексте программы следуют подряд команды: PUSH AX; PUSH CX; POP DX; POP BX. Чему будет равно содержимое регистра BH? Рисунок стека соответствует промежуточному состоянию (до выполнения команд POP).



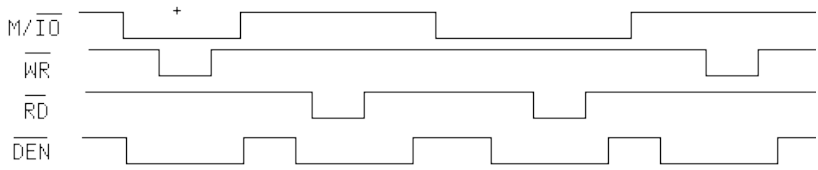
Пояснение : В стек дважды что-то помещается и дважды что-то извлекается, причем последние два байта помещаются в регистр BX. Перефразируя правило "последним вошел - первым вышел" в "последним вышел - первым вошел", придем к выводу, что содержимое AX засылается в регистр BX. Команда PUSH AX помещает два байта в стек по правилу "старшая половина двухбайтового регистра - по старшему адресу". Ответ: **c9**

В тексте программы следуют подряд команды: POP AX; POP DX. Чему будет равно содержимое регистра DL?



Ответ: 4c

На временных диаграммах внизу приведены управляющие сигналы на выходах микропроцессора (без сохранения точных временных пропорций). Что осуществляется в момент времени отмеченный знаком + ? Чтение данных из памяти, запись данных в память вывод данных в порт, ввод данных из порта, ничего из указанного выше**.**



Пояснения : Если ~DEN=0 (разрешение данных) возможно 4 случая (для приведенных диаграмм): 1) M/~IO = 0, ~RD = 0 активны сигналы ввод/вывод (~IO) и чтение-ввод (~RD). 2) M/~IO = 0, ~WR = 0 активны сигналы ввод/вывод (~IO) и запись-вывод(~WR). 3) M/~IO = 1, ~RD = 0 активны сигналы обращения к памяти(M) и чтение-ввод(~RD). 4) M/~IO = 1, ~WR = 0 активны сигналы обращения к памяти(M) и запись-вывод(~WR).

Ответ: Вывод данных в порт.

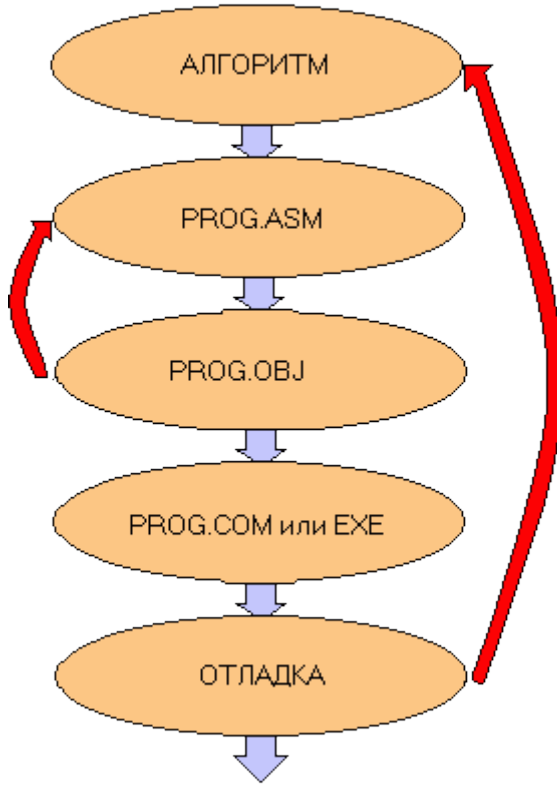
ПРОГРАММИРОВАНИЕ НА АССЕМБЛЕРЕ

АССЕМБЛЕР. ЭТАПЫ РАЗРАБОТКИ ПРОГРАММЫ.

Язык программирования наиболее полно учитывающий особенности "родного" микропроцессора и содержащий мнемонические обозначения машинных команд называется Ассемблером. Программа, написанная на Ассемблере называется исходной программой. Далее остановимся на версии, называемой Турбо Ассемблер для семейства МП x86.

Разработка программы на Ассемблере состоит из следующих этапов**:**

- 1) Составление алгоритма в виде блок-схемы или структурного описания,
- 2) Написание текста исходной программы PROG.ASM с помощью редактора текстов. Имя PROG может быть произвольным, а расширение ASM - обязательно,
- 3) Перевод (**трансляция** или ассемблирование) исходной программы в машинные коды с помощью транслятора TASM.EXE. На этом этапе получается промежуточный продукт PROG.OBJ (объектный код). Выявленные при этом синтаксические и орфографические ошибки исправляются повтором пп.2 и 3,
- 4) Преобразование с помощью программы TLINK.EXE объектного кода PROG.OBJ в выполнимый код PROG.EXE или PROG.COM. Эта операция называется компоновкой или связыванием. Для других семейств МП и МК исполнимые программы могут иметь другие расширения - BIN, HEX и др.
- 5) Выполнение программы и ее отладка начиная с п.1, если встретились логические ошибки или ошибки периода выполнения (run time errors).



Текст программы на Ассемблере содержит следующие элементы / операции:

- а) **команды (инструкции)**,
- б) **директивы (псевдооператоры)**,
- в) **операторы**,
- г) **предопределенные имена**.

Действия обусловленные операциями перечисленными в пп.б,в,г выполняются на этапе трансляции, т.е. являются командами Ассемблеру. Операции, называемые командами или инструкциями выполняются во время работы программы, т.е. являются командами микропроцессору.

ФОРМАТ КОМАНД И ИХ КЛАССИФИКАЦИЯ

Инструкция записывается на отдельной строке и включает до четырех полей, из которых необязательные выделены [ ]:

[метка:]	мнемоника_команды	[операнд(ы)]	[;комментарий]

Метка или символический адрес содержит до 31 символа из букв цифр и знаков ? @ . _ $. Причем цифра не должна стоять первой, а точка, если есть должна быть первой.

Мнемоника - сокращенное обозначение кода операции (КОП) команды, например мнемоника ADD обозначает сложение (addition).

Операндами могут быть явно или неявно задаваемые двоичные наборы, над которыми производятся операции. Операнды приводятся в одной из четырех систем счисления и должны оканчиваться символом b(B), o(O), d(D) или h(H) для 2, 8, 10 или 16-ной СС. К шестнадцатиричному числу добавляется слева ноль, если оно начинается с буквы.

Система команд может быть классифицирована по трем основным признакам -

- **длина команды или число занимаемых ею байтов,**
- **функциональное назначение и**
- **способ адресации.**

Для МП 1810ВМ86 (семейство x86) команда занимает от одного до шести байтов. Первым байтом команды всегда является код операции - КОП, например код команды INT N равен CD(HEX). Код операции может занимать и более одного байта. Следующие за КОП байты являются операндами.

По функциональному признаку инструкции можно разбить на пять больших групп:

- **1) команды пересылки данных,**
- **2) арифметические команды,**

- **3) логические команды и команды сдвига,**
- **4) команды переходов или ветвления**
- **5) команды управления.**

Существует пять основных способов адресации:

- **регистровая,**
- **непосредственная,**
- **прямая,**
- **косвенная и**
- **стековая.**

Большинство остальных способов адресации являются комбинациями или видоизменениями перечисленнных.

В первом случае операнд(ы) **располагаются в регистрах микропроцессора (МП**), например по команде MOV AX,CX пересылается содержимое CX в AX.

При **непосредственной адресации** операнд располагается в памяти непосредственно за КОП, инструкция MOV AL,0f5h записывает число 245(f5) в регистр AL.

В случае **прямой адресации** за КОП следует не сам операнд, а адрес ячейки памяти или внешнего устройства, например команда IN AL,40h вводит в аккумулятор байт данных из внешнего устройства с адресом 40h.

Косвенная адресация отличается от регистровой тем, что в регистре хранится не операнд, а его адрес. Например, по команде MOV AL,[BX] в аккумулятор al будет записано число из ячейки памяти с адресом, хранящимся в регистре BX.

Стековая адресация производится к операндам расположенным в области памяти, называемой стек.

НЕКОТОРЫЕ ОПЕРАТОРЫ, ПРЕДОПРЕДЕЛЁННЫЕ ИМЕНА, ДИРЕКТИВЫ И КОМАНДЫ АССЕМБЛЕРА 80X86(8088/86)

ПРЕДОПРЕДЕЛЕННЫЕ ИМЕНА

1. **$ -** программный счетчик. Этот символ отмечает текущий адрес в текущем сегменте. Полезен при определении длины цепочек байтов или строк.

text DB 'This string has NN letters'
NN = $ - text; во время трансляции пользовательской переменной NN будет присвоено значение равное длине строки text (количеству байтов в этой строке). Не путать часть строки '..NN..' и константу NN!

2. **@data** - адрес начала сегмента данных.

```
....
mov ax,@data
mov ds,ax;
```

в сегментном регистре DS теперь адрес сегмента данных.

3. **??date, ??time, ??filename** - эти имена во время трансляции заменяются, соответственно на текущие дату, время и имя файла в формате ASCII.

ОПЕРАТОРЫ

1. **()** - скобки, определяют порядок вычислений
2. **[]** - например [BX] означает содержимое ячейки памяти с адресом в регистре bx. Признак косвенной или прямой адресации.

```
mov al, [bx]
mov al, [my_mem1]
```

3. **+, -, *, /** - операторы сложения, вычитания, умножения и деления.
mov ax, (2 * 3 + 8 / 2) - 2; в регистр ax будет помещено число 8.
4. **MOD** - деление по модулю. Даёт остаток.
5. **SHL, SHR** - сдвиг операнда влево, вправо.

mov si, 01010101b SHR 3; в регистр SI будет загружено число 0Ah (00001010).

6. **NOT** - побитовая инверсия.

7. **AND, OR, XOR** - операции "И","ИЛИ","ИСКЛ.ИЛИ".

mov dl, (10d OR 5d) XOR 7d; (dl) будет равно 8.

8. **:** - переназначение сегмента.

mov dl, [es:bx]; поместить в dl байт данных из сегмента es и отстоящий от его начала на (bx) байтов (смещение).

9. **OFFSET** - оператор получения смещения адреса относительно начала сегмента (то есть количества байтов от начала сегмента до идентификатора адреса).

mov bx, OFFSET table; table - символическое имя, OFFSET table - адрес.

ДИРЕКТИВЫ (ПСЕВДООПЕРАТОРЫ)

1. **:** - определяет близкую метку (в пределах сегмента).

```
    jmp lbl .... ....
lbl:  ....
```

2 . = - присваивает символическому имени значение выражения (допускается переопределение).

videoram = 0B800h;

videoram = 0B000h; значение переопределено

3. **.CODE** - указывает начало кодового сегмента, то есть сегмента, где располагаются коды программы.

4. **.DATA** - определяет начало сегмента данных в программе.

5. **DB, DW** ... - директивы резервирующие один или несколько байтов: DB, или одно или несколько слов: DW. fibs, rus .. Array - произвольные имена.

```
    ....
    .DATA
fibs  DB 1,1,2,3,5,8,13
rus   DB 'Турбо Ассемблер'
buf   DB 80 DUP(0);резервируется 80 байтов,каждый обнуляется
int   DW 65535;в двух байтах располагается число FFFFh.
Array DW 100 DUP (0);резервируется 100 слов
```

6. **END** - обозначает конец программы.

```
    ....
    .CODE
MyPROG:....; точка входа (начало программы).
    ....; команды программы
    ....
    END MyPROG; MyPROG - произвольное имя
```

7. **ENDM** - окончание блока или макроопределения

8. **ENDP** - обозначает конец подпрограммы.

9. **EQU** - присваивает символическому имени или строке значение выражения (в отличие от '=' переопределение не допускается).

```
BlkSize EQU 512
BufBlks EQU 4
BufSize EQU BlkSize * BufBlks
```

10. **LABEL** - определяет метку соответствующего типа.

```
    ....
    .DATA
m_byte LABEL BYTE;  метка m_byte типа BYTE позволяет теперь
m_word DW 0;              иметь доступ отдельно к каждому байту
данных
    .CODE;m_word типа WORD
    ....
    mov [m_word],0204h
    add [m_byte],'0';теперь в m_word хранится код
    add [m_byte+1],'0';3234h,ASCII код '0' равен 30h
```

11. **LOCAL** - определяет метки внутри макроопределений, как локальные и в каждом макрорасширении вместо них ассемблер вставляет уникальные метки: ??XXXX, где XXXX = (0000...FFFF)h. Почему ??XXXX ? Да потому что никому не должно прийти в голову начинать символическое имя с ??, и транслятор смело может генерировать метки не боясь совпадений.

12. **MACRO** - задает макроопределение.

```
XchgM   MACRO a,b; a,b - параметры макро (ячейки памяти)
    mov ax,b; данное макрооопределение позволяет делать
```

```
    mov bx,a; обмен данными между ячейками памяти, в
    mov a,ax; отличие от команды xchg ;
    mov b,bx; нельзя записать mov a,b;
ENDM
```

Вызов этого макроса производится командой: XchgM m,n. Во время трансляции каждый макровызов заменяется макроопределением - макроподстановка.

13. **.MODEL** - определяет размер памяти под данные и код программы.

.MODEL tiny; под программу, данные и стек отводится один общий сегмент (64 Kb).

14. **PROC** - определяет начало подрограммы.

```
Print  PROC; Print - произвольное имя
    ;здесь команды подпрограммы
    ret; обязательная команда возврата
    ENDP
    ....
    call Print;вызов подпргаммы.
```

подпрограмма	макроопределение
My_Print PROC ;== определение подпрограммы ..... ret ENDP ..... call My_Print ;== вызов подпрограммы	My_Print MACRO ;== макроопределение (макрос) .... ENDM ..... My_Print ;== вызов макроса (макровызов)

Программа, написанная с использованием подпрограмм будет занимать меньше памяти, чем программа написанная с использованием макросов, но выполняться будет медленнее за счет выполнения двух дополнительных команд call и ret.

15. **.STACK** - задает размер стека.

.STACK 200h; выделяет 512 байтов для стека.

16. **.RADIX base** - определяет систему счисления по умолчанию, где base - основание системы счисления: 2, 8, 10, 16.

```
    .RADIX 8
oct     = 77; oct равно 63d.
```

17. **;** - начало комментария.

18. **ORG XXXXh** - определяет, что следующая часть программы и/или данных будет располагаться начиная с адреса XXXX.

КОМАНДЫ ПЕРЕСЫЛКИ

1. **MOV** DST, SRC; переслать, а точнее скопировать (SRC) в (DST), SRC - операнд источник, DST - операнд приемник. DST и SRC могут храниться в регистрах или в памяти. Исключения: а) оба операнда DST и SRC не могут одновременно находиться в памяти. б) Только операнд SRC может быть задан (адресован) непосредственно. Здесь и далее содержимое регистра, например регистра AL будет обозначаться - (AL) или (al), а пересылка в комментарии будет обозначаться знаком <-- .

```
    mov al,ch; (al) <-- (ch). Содержимое регистра CH копируется в аккумулятор.
    mov cx,dx;
    mov bh,[mems];содержимое ячейки памяти с символическим адресом mems переслать в регистр BH. Можно: mov bh,mems.
    mov al,[bx];переслать в аккумулятор содержимое ячейки памяти с адресом находящимся в регистре BX.
    mov bx,OFFSET src;поместить в BX смещение адреса ячейки памяти SRC, в текущем сегменте.
    mov al,table[bx];загрузить в аккумулятор байт из массива с символическим адресом начального элемента table и отстоящий от этого элемента на (bx) байтов. Другие варианты:
    mov al,[table + bx] или
    mov al,table + bx.
```

До выполнения команды	Регистр BX	Регистр AL	Адрес	Код

mov al,[table + bx]	0010	XX	0800 (table)	8c
			08xx	xx
			0810	58

После выполнения команды mov al,[table + bx]	**Регистр BX**	**Регистр AL**	**Адрес**	**Код**
	0010	58	0800 (table)	8c
			08xx	xx
			0810	58

2. **PUSH** RP; поместить на вершину стека содержимое пары регистров RP (например push bx).
3. **POP** RP; снять с вершины стека два байта и поместить в пару RP (например pop ax).
4. **XCHG** DST, SRC; поменять местами содержимое (DST) и (SRC). Оба операнда не могут быть одновременно содержимым ячеек памяти.
5. **XLAT** SRC; извлечь из таблицы с начальным адресом SRC байт данных имеющий номер от начала таблицы = (AL), и поместить его в AL. Адрес SRC должен находиться в регистре BX. Другой вариант: XLATB.

```
    ....
    .DATA
src   DB 15d,10h,00110101b,'A','B',166d
    .CODE
    ....
    mov al,2; в результате выполнения этих трех  команд
    mov bx,OFFSET src; в регистр AL будет загружен код
    xlatb; 00110101b = 35h = 53d = ASCII'5'
```

6. **IN** ACCUM, PORT; поместить в аккумулятор AL или AX байт или слово из порта с адресом PORT (специальная функция регистров общего назначения AX или AL). Если адрес порта <= FF то адрес

порта может указываться прямо, если адрес порта > FF, то адрес порта указывается косвенно, через содержимое регистра DX (специальная функция регистра общего назначения DX)

.

in al,0a5h; ввести в AL байт данных из ВУ с адресом A5
mov dx,379h; ввести в аккумулятор AL байт данных из
in al,dx; внешнего устройства с адресом 379

7. **OUT** PORT, ACCUM; переслать из аккумулятора AL или AX байт или слово в ВУ с символическим адресом PORT.

out 0ffh,al;
....
mov dx,37Ah; переслать слово данных из AX в ВУ с адресом порта 37Ah
out dx,ax;

При выполнении команд in al, XX и out XX,al (in al, dx и out dx, al) можно выделить 2 этапа:

а) микропроцессор помещает адрес XX на шину адреса. Этот адрес декодируется дешифратором в активный сигнал на одном из выходов, который поступает на вход ~CS (или CS) ВУ. При этом ВУ, становится готовым к обмену данными по ШД.

б) Второй этап имеет отличия. Команда **in ...**: МП генерирует на шине управления (ШУ) строб чтения ~IOR, которым байт данных считывается по ШД из адресуемого порта ВУ в аккумулятор МП. Команда **out ...**: МП генерирует на шине управления строб записи ~IOW, которым байт данных из аккумулятора микропроцессора записывается по ШД в выбранный порт ВУ. Обычно этот байт защелкивается в триггерах порта. Наглядно проследить действие этих команд можно по рисунку на стр.8 лаб. работы №2.

8. **LEA** RP,M; загрузить в регистр RP эффективный адрес (смещение) ячейки памяти с символическим адресом M.
lea di, rus; аналог этой команды - mov di, OFFSET rus.

АРИФМЕТИЧЕСКИЕ КОМАНДЫ

1. **ADD** DST, SRC; сложить содержимое SRC и DST и результат переслать в DST. SRC - операнд источник, DST - операнд приемник. Также как и в командах пересылки, DST и SRC могут храниться в регистрах или в памяти, кроме случая, когда оба операнда DST и SRC хранятся в памяти. Операнд SRC может быть задан (адресован) непосредственно.

add al, [mem_byte]; mem_byte однобайтовая ячейка памяти
add [mem_word], dx; mem_word двухбайтовая ячейка памяти
add ch,10001010b;

2. **INC** DST; увеличить (DST) на 1 (инкремент (DST)).

```
inc si; (SI) <-- (SI) + 1.
inc count; (count) <-- (count) + 1.
```

3. **SUB** DST, SRC; (DST) <-- (DST) - (SRC), вычесть (SRC) из (DST) и результат поместить в DST.

4. **DEC** DST; декремент (DST).

5. **CMP** DST, SRC; сравнить содержимое DST и SRC. Эта команда выполняет вычитание (SRC) из (DST) но разность не помещает в DST и по результату операции воздействует на флаги.

Флаги				условие
OF	SF	ZF	CF	
0/1	0	0	0	DST > SRC
0	0	1	0	DST = SRC
0/1	1	0	1	DST < SRC

0/1 - означает, что флаг может быть равен 0 или 1 в зависимости от значений операндов. Флаги OF и SF имеют смысл при операциях со знаковыми числами, CF для беззнаковых чисел. Флаг переполнения OF устанавливается в 1, если в результате операции сложения или вычитания значения переноса в старшиий двоичный разряд и из старшего двоичного разряда не совпадают. По другому определению OF принимает значение 1, если результат превышает диапазон представления соответствующих чисел. Пусть DST > SRC и оба являются однобайтовыми числами, тогда:

DST:	1.(+127)	2.(+127)
SRC:	- (+2)	- (-2)

```
------          -------
(+125) (OF)=0    (+129)? (OF)=1
```

Во втором примере результат превышает диапазон: -128 <= x < = +127. Флаг знака SF устанавливается в '1', если старший бит результата операции равен 1, т.е. при отрицательном результате. В противном случае сбрасывается. Флаг нуля ZF устанавливается в '1' при нулевом результате (!), иначе сбрасывается. Флаг переноса CF = 1, если есть перенос из старшего разряда при сложении или есть заем в младший разряд при вычитании. Иначе флаг сбрасывается. Для первого примера SF = ZF = CF = 0, для второго: SF = 1, ZF = CF = 0.

Дальнейшее изучение команд ассеблера будет продолжено во второй части этого пособия, где также будут приведены типовые примеры проектирования (программирования) базовых микропроцессорных систем.

УНИВЕРСИТЕТ ИТМО

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА СЕНСОРИКИ

Заведующий кафедрой: д.т.н., проф. Г.Н. Лукьянов.

Кафедра Сенсорики (первоначальное название “Радиотехники” затем “Электроники”) была основана в 1945 году. Первым руководителем кафедры был С.И. Зилитинкевич известный в стране и за рубежом ученый в области физической электроники и радиотехники, активный работник высшей школы, заслуженный деятель науки и техники РСФСР, доктор технических наук, профессор ЛИТМО с 1938 г., инициатор создания в ЛИТМО инженерно-физического и радиотехнического факультетов (1946г.). С.И. Зилитинкевич заведовал кафедрой с 1945 до 1978 года. Под его научным руководством аспирантами и соискателями выполнено более 50 кандидатских диссертаций, многие его ученики стали докторами наук.

В дальнейшем, с 1978 г. по 1985 г. кафедру возглавил к.т.н., доцент Е.К. Алахов, один из учеников С.И. Зилитинкевича.

С 1985 г. по 2006 г. руководителем кафедры стал д.т.н., профессор В.В. Тогатов, известный специалист в области силовой электроники и приборов для измерения параметров полупроводниковых структур.

Начиная с 2006 г. кафедрой заведует д.т.н., профессор Г.Н. Лукьянов, под руководством и при участии которого кардинально обновилось лабораторное оборудование в рамках инновационной программы развития.

В 2015 году к кафедре Сенсорики была присоединена кафедра ИТиКТ.

Основные направления кафедры связаны с разработкой приборов для лазерной и медицинской техники, приборов для измерения параметров полупроводниковых структур, а также встраиваемых цифровых и микропроцессорных устройств.

На кафедре имеются и размещены на сайте ЦДО следующие материалы для дистанционного обучения (автор Ю.В. Китаев):

- Конспект лекций по дисциплине “Электроника и микропроцессорная техника”;
- свыше 600 вопросов к обучающим и аттестующим тестам;
- 18 дистанционных лабораторных и практических работ

Кафедра оборудована следующими компьютеризированными учебными лабораториями:

- АРМС – полупроводниковые приборы;
- Устройства на полупроводниковых приборах;
- Цифровая техника;
- Микропроцессорная техника
- Моделирование электронных устройств.

Юрий Васильевич Китаев

ОСНОВЫ МИКРОПРОЦЕССОРНОЙ ТЕХНИКИ

Учебное пособие
Часть 1

В авторской редакции
Дизайн Китаев Ю.В.
Верстка Китаев Ю.В.
Редакционно-издательский отдел университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати _____
Заказ № _____
Тираж 50 экз.
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49