Коми Республикаса велӧдан, наука да том йӧз политика министерство
Министерство образования, науки и молодежной политики Республики Коми
Государственное профессиональное образовательное учреждение
«Сыктывкарский целлюлозно – бумажный техникум»

РАССМОТРЕНО
на заседании ПЦК информационных
дисциплин ГПОУ «СЦБТ»
Протокол № __
«__» _____ г.
Председатель ПЦК _____ В.А. Шулепов
Преподаватель _____ Д.Д. Расов

УТВЕРЖДАЮ
Зам.директора по УПР
ГПОУ «СЦБТ»
_____ Е.В. Соколова
«____» _____ 20__ г.

Методические указания по выполнению лабораторных работ

по учебной дисциплине МДК02.01 Микропрроцессорные системы
для специальности
09.02.01 Компьютерные системы и комплексы

Программирование микропроцессора i8086 на языке ассемблер

Сыктывкар 2018

Лабораторная работа № 4

СПОСОБЫ АДРЕСАЦИИ НА ЯЗЫКЕ АССЕМБЛЕРА

Цель работы: *Изучить основные способы адресации.*

1. Способы адресации.

Способом, или режимом адресации называют процедуру нахождения операнда для выполняемой команды. Если команда использует два операнда, то для каждого из них должен быть задан способ адресации, причем режимы адресации первого и второго операнда могут, как совпадать, так и различаться. Операнды команды могут находиться в разных местах: непосредственно в составе кода команды, в каком-либо регистре, в ячейке памяти; в последнем случае существует несколько возможностей указания его адреса. Строго говоря, способы адресации являются элементом архитектуры процессора, отражая заложенные в нем возможности поиска операндов. С другой стороны, различные способы адресации определенным образом обозначаются в языке ассемблера и являются разделом языка.

В программах, написанных на языке ассемблера термин "операнд" применим к обозначению тех физических объектов, с которыми имеет дело процессор при выполнении машинной команды и, говоря об операндах команд языка, понимают в действительности операнды машинных команд. По отношению к командам ассемблера используется термин "параметры".

В архитектуре современных 32-разрядных процессоров Intel предусмотрены довольно изощренные способы адресации; в МП 86 способов адресации меньше. Мы в настоящей лабораторной работе познакомимся с режимами адресации, используемые в МП 86.

Различают следующие режимы адресации:

- регистровый;
- непосредственный;
- прямой;
- регистровый косвенный (базовый или индексный);
- регистровый косвенный со смещением (базовый или индексный);
- базово-индексный;
- базовый индексный со смещением.

1.1 Регистровый режим

Значение операнда-источника предварительно запоминается в одном из встроенных регистров микропроцессора.

Сам регистр становится эффективным адресом. Операнд (байт или слово) находится в регистре. Этот способ применим ко всем программно-адресуемым регистрам процессора:

inc CX	**; Увеличение на 1 содержимого CX**
push DS	**; Сегментный адрес сохраняется в стеке**
xchg BX,BP	**; Регистры BX и BP обмениваются содержимым**
mov ES,AX	**; Содержимое AX пересылается в ES**

1.2 Непосредственный режим

Непосредственная адресация. Операнд (байт или слово) указывается в команде и после трансляции поступает в код команды; он может иметь любой смысл (число, адрес, код ASCII), а также быть представлен в виде символического обозначения.

```
mov AH, 40h          ; Число 40h загружается в AH
mov AL,'*'           ; Код ASCII символа "*' загружается в AL
```

```
int 21h                    ; Команда прерывания с аргументом 21h
limit equ 528              ; Число 528 получает обозначение limit
mov CX,limit               ; Число, обозначенное limit, загружается в CX
```

Команда **mov**, использованная в последнем предложении, имеет два операнда; первый операнд определяется с помощью регистровой адресации, второй - с помощью непосредственной.

Важным применением непосредственной адресации является пересылка относительных адресов (смещений), для этого используется описатель **offset** (смещение):

```
;Сегмент данных
string db “Privet"        ;Строка символов
;Сегмент команд
mov DX,offset string      ;Адрес строки засылается в DX
```

1.3 Прямой режим.

Адресуется память; адрес ячейки памяти (слова или байта) указывается в команде (обычно в символической форме) и поступает в код команды:

```
;Сегмент данных
meml dw 0          ;Слово памяти содержит 0
mem2 db 230        ;Байт памяти содержит 230
;Сегмент команд
inc meml           ;Содержимое слова meml увеличивается на 1
mov DX, meml       ;Содержимое слова с именем menu загружается в DX
mov AL,mem2        ;Содержимое байта с именем mem2 загружается в AL
```

Сравнивая этот пример с предыдущим, мы видим, что указание в команде имени ячейки памяти обозначает, что операндом является

содержимое этой ячейки; указание имени ячейки с описателем **offset** - что операндом является адрес ячейки.

Прямая адресация памяти на первой взгляд, кажется, простой и наглядной. Если мы хотим обратиться, например, к ячейке **meml**, мы просто указываем ее имя в программе. В действительности, однако, дело обстоит сложнее. Адрес любой ячейки состоит из двух компонентов: сегментного адреса и смещения. Обозначения **meml** и **mem2** в предыдущем примере, являются смещениями. Сегментные же адреса хранятся в сегментных регистрах. Однако сегментных регистров четыре: **DS**, **ES**, **CS** и **SS**. Каким образом процессор узнает, из какого регистра взять сегментный адрес, и как сообщить ему об этом в программе?

Процессор различает группу кодов, носящих название префиксов. Имеется **несколько групп префиксов**: повторения, размера адреса, размера операнда, замены сегмента. Здесь нас будут интересовать префиксы замены сегмента.

Команды процессора, обращающиеся к памяти, могут в качестве первого байта своего кода содержать **префикс замены сегмента**, с помощью которого процессор определяет, из какого сегментного регистра взять сегментный адрес. Для сегментного регистра **ES** код префикса составляет **26h,** для **SS** – 36h, для CS - 2Eh. Если префикс отсутствует, сегментный адрес берется из регистра **DS** (хотя для него тоже предусмотрен свой префикс).

В приведенном примере, по умолчанию, все данные адресуются через сегментный регистр **DS**, так что вместо **inc meml** можно было написать **inc DS: mem**. В случае замены сегментного регистра его обязательно нужно указывать явно:

inc ES: mem1

inc CS: mem2

Обращение к ячейке памяти по известному абсолютному адресу осуществляется следующим образом:

mov AL,DS: [17h]	**Загрузка в AL содержимого ячейки со смещением 17h в сегменте, определяемом содержимым DS**

1.4 Регистровый косвенный (базовый и индексный).

Адресуется память (байт или слово). Относительный адрес ячейки памяти находится в регистре, обозначение которого заключается в прямые скобки. В МП 86 косвенная адресация допустима только через регистры **BX**, **BP**, **SI** и **DI**. При использовании регистров **BX** или **BP** адресацию называют базовой, при использовании регистров **SI** или **DI** - индексной.

Если косвенная адресация осуществляется через один из регистров **BX**, **SI** или **DI**, то подразумевается сегмент, адресуемый через **DS**, поэтому при адресации через этот регистр обозначение **DS:** можно опустить:

mov es: [bx],'1' ————→ mov [bx],'1'

Кстати, этот фрагмент немного эффективнее предыдущего в смысле расходования памяти. Из-за отсутствия в коде последней команды префикса замены сегмента он занимает на 1 байт меньше места.

Регистры **BX**, **SI** и **DI** в данном применении совершенно равнозначны, и с одинаковым успехом можно воспользоваться любым из них:

Не так обстоит дело с регистром **BP**. Этот регистр специально предназначен для работы со стеком, и при адресации через этот регистр в режимах косвенной адресации подразумевается сегмент стека; другими словами, в качестве сегментного регистра по умолчанию используется регистр **SS**.

Обычно косвенная адресация к стеку используется в тех случаях, когда необходимо обратиться к данным, содержащимся в стеке, без изъятия их оттуда (например, если эти данные приходится считывать неоднократно).

Обозначение этого способа адресации:

[BX]	**(подразумевается DS: [BX])**
[BP]	**(подразумевается SS: [BP])**
[SI]	**(подразумевается DS: [SI])**
[DI]	**(подразумевается DS: [DI])**

Использование базовой адресации, на первый взгляд, снижает эффективность программы, так как требует дополнительной операции - загрузки в базовый регистр требуемого адреса. Однако команда с базовой адресацией занимает меньше места в памяти (так как в нее не входит адрес ячейки) и выполняется быстрее команды с прямой адресацией (из-за того, что команда короче, процессору требуется меньше времени на ее считывание из памяти). Поэтому базовая адресация эффективна в тех случаях, когда по заданному адресу приходится обращаться многократно, особенно, в цикле. Выигрыш оказывается тем больше, чем большее число раз происходит обращение по указанному адресу. С другой стороны, возможности этого режима адресации невелики, и на практике чаще используют более сложные способы.

Примеры:

mov SI, offset string	**; В SI загружается относительный адрес ячейки string**
mov AX, [SI]	**; Содержимое ячейки string загружается в AX**
inc [SI]	**; Увеличиваться содержимое ячейки string**
mov BX, [SI]	**; Новое содержимое ячейки string загружается в BX**
mov DI, SI	**; Относительный адрес ячейки string копируется в DI**

1.5 Регистровый косвенный режим со смещением (базовый и индексный).

Адресуется память (байт или слово). Относительный адрес операнда определяется, как сумма содержимого регистра **BX**, **BP**, **SI** или **DI** и указанной в команде константы, иногда называемой смещением. Смещение может быть числом или адресом. Так же, как и в случае базовой адресации, при использовании регистров **BX**, **SI** и **DI** подразумевается сегмент, адресуемый через **DS**, а при использовании **BP** подразумевается сегмент стека и, соответственно, регистр **SS**.

смещение = {SP, BP, DI, SI, BX} + смещение из команды

Иногда можно встретиться с альтернативными обозначениями того же способа адресации, которые допускает ассемблер. Например, вместо **4 [BX]** можно с таким же успехом написать **[BX+4]**, **4+[BX]** или **[BX]+4**. Такая неоднозначность языка ничего, кроме путаницы, не приносит, однако ее надо иметь в виду, так как с этими обозначениями можно столкнуться, например, рассматривая текст деассемблированной программы.

Рассмотрим теперь пример использования базовой адресации со смещением при обращении к стеку:

смещение = {SP, BP, DI, SI, BX} + смещение из команды

Здесь квадратные скобки [] - это тоже оператор. Он вычисляет адрес как сумму того, что находится внутри скобок с тем, что находится снаружи.

array db 0, 10, 20, 30, 40, 50, 60*;* Пусть в сегменте данных определен массив

Последовательность команд:

mov BX,5

mov AL,array [5] ; загрузит в AL элемент массива с индексом 5, то есть 50.

Тот же результат будет получен и в таких последовательностях команд:

mov BX,offset array

mov AL,5 [BX]

или

mov AL, [BX] +5

mov AL, [BX+5]

1.6 Базово-индексный режим

Адресуется память (байт или слово). Относительный адрес операнда определяется, как сумма содержимого следующих пар регистров:

смещение **[BX] [SI]**	**(подразумевается DS: *смещение* [BX] [SI])**
смещение **[BX] [DI]**	**(подразумевается DS: *смещение* [BX] [DI])**
смещение **[BP] [SI]**	**(подразумевается SS: *смещение* [BP] [SI])**
смещение **[BP] [DI]**	**(подразумевается SS: *смещение* [BP] [DI])**

Во всех этих случаях можно также написать:

смещение **[BX+SI]**
[*смещение* +BX+SI]
[BX+SI] +*смещение*

Это чрезвычайно распространенный способ адресации, особенно, при работе с массивами. В нем используются два регистра, при этом одним из них должен быть базовый (**BX** или **BP**), а другим - индексный (**SI** или **DI**). Как

правило, в одном из регистров находится адрес массива, а в другом - индекс в нем, при этом совершенно безразлично, в каком что.

1.7 Базово-индексная адресация со смещением.

Адресуется память (байт или слово). Относительный адрес операнда определяется как сумма содержимого двух регистров и смещения.

Это способ адресации является развитием предыдущего. В нем используются те же пары регистров, но полученный с их помощью результирующий адрес можно еще сместить на значение указанной в команде константы. Как и в случае базово-индексной адресации, константа может представлять собой индекс (и тогда в одном из регистров должен содержаться базовый адрес памяти), но может быть и базовым адресом. В последнем случае регистры могут использоваться для хранения составляющих индекса.

Приведем формальный пример рассматриваемого режима адресации.

Пусть в сегменте данных определен массив из 24 байт

syms db 'ЙЦУКЕНГШЩЗХЪ'
db 'йцукенгшщзхъ'

Последовательность команд

mov BX,12
mov SI,6
mov DL,syms [BX] [SI] *;* загрузит в регистр **DL** элемент с индексом **6** из второго ряда, то есть код ASCII буквы **г**

Тот же результат будет получен и в таком варианте:

mov BX,offset syms

mov SI,6

mov DL,12 [BX] [SI]

2. Порядок выполнения работы:

1. С помощью редактора эмулятора **EMU 8086** напишите программу, исходный текст которой приводится в листинге №1:

2. Создайте исполняемый файл типа **MZ**.

3. Изучите структуру программы, также изучите структуру сегмента данных программы: найдите в нем все переменные, определенные в тексте программы.

4. Переделайте программу с использованием упрощенных директив сегментации так, чтобы получить исполняемый файл типа.com и сравните размеры программ.

5. Выполните первые 5 шагов программы, анализируя и записывая состояние регистров на каждом шаге.

6. Занесите в **CX 00FFh**. Определите по способу адресации ячейку памяти в сегменте, где произойдут изменения, записать ее адрес.

7. Выполните дальнейшие шаги программы, анализируя возможные способы адресации.

8. Подготовьте отчет, который должен содержать тексты программ, адреса сегментных регистров и записи адресов ячеек памяти против соответствующих команд, а также запись содержимого этих ячеек.

9. В отчете должны содержаться ответы на следующие вопросы.

3. Контрольные вопросы

1. Как переслать содержимое X в Y?
2. Чем отличаются команды

MOV [si], cx

и

MOV si, cx?

3. К какому способу адресации относится команда **MOV dx, offset message**?

4. Какие сегменты используются при следующих вариантах адресации: [BX] [SI], [BX] [DI], [BP] [SI], [BP] [DI] ?

5. Что произойдет при выполнении инструкции

MOV AL, DS: 17h?

Чем эта команда отличается от следующей:

MOV AL, DS: [17h] ?

6. Пусть в сегменте данных определен массив

Array db 0,15,22,31,44,45,62,67,76,99

Что окажется в регистре **AL** после выполнения команд:

MOV BX, 5

MOV AL, array [BX] ?

7. Какой это способ адресации (пример вопроса 6)?

8. Укажите, какие инструкции в программе (**листинг №1**), созданной в данной лабораторной работе, относятся к инструкциям:

- с непосредственным;
- косвенным режимом адресации?

9. Укажите способ записи обращения напрямую к ячейке памяти по известному абсолютному адресу?

10. Префиксы, Виды префиксов. Префиксы замены сегмента?

11. Перечислите регистры косвенной и базовой адресации. Опишите отличия?

12. Сущность эффективности базовой адресации в сравнении с прямой?

Листинг №1.

```
TITLE MOVE2
MOVE2 SEGMENT 'CODE'
ASSUME CS: MOVE2, DS: DATA
   MYPROC PROC OUTPROC:
        MOV AX,DATA
        MOV DS,AX
        MOV AH,BH
        MOV AH,X
        MOV CH,3
        MOV AX,3
        MOV AX,Y
        MOV [SI],CX
        MOV [BP],CX
        MOV [SI],258
        MOV [BP+516],1027
        MOV BYTE PTR X,255
        MOV BYTE PTR [DI+515],4
        MOV WORD PTR [DI+515],4
        MOV [DI+BP+515],258
        MOV AX, [SI+BX+258]
        MOV AH,4CH
        INT 21H
  MYPROC ENDP
MOVE2 ENDS
DATA SEGMENT
        X DB 1
        Y DW 2
DATA ENDS
```

END MYPROC