# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №2**

# по дисциплине «Организация систем и ЭВМ»

**Тема:** «**Изучение режимов адресации основной памяти**»

Студент гр. 3388 Снигирёв А.А.

Преподаватель Фирсов М.А.

Санкт-Петербург 2024

# Цель работы.

Изучить на примере учебной программы режимы адресации памяти в ассемблере Intel 8086.

# Задание.

1. Получить у преподавателя вариант выбора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2+.dat и занести свои данные вместо значений, указанных в приведённой для образца программе.

2. Протранслировать программу с созданием файла диагностических сообщений и объяснить обнаруженные ошибки (error) и предупреждения (warning). Листинг привести в приложении к отчёту. Закомментировать операторы с ошибками в тексте программы, а операторы с предупреждениями оставить без изменения. Объяснения ошибок и предупреждений должны быть приведены в отчёте по лабораторной работе.

3. Снова протранслировать программу и скомпоновать загрузочный модуль. Учесть, что программа учебная и может выполняться только под отладчиком. В автоматическом режиме она выполняться не должна.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого **используемых** регистров и ячеек памяти до и после выполнения каждой команды. Разобраться в используемых режимах адресации и получаемых результатах. Результаты прогона программы под управлением отладчика должны быть представлены в отчёте по лабораторной работе в табличном виде, аналогичном указанному в лаб.работе №1.

Вариант 19:

vec1 32,33,34,35,39,38,37,36

19 vec2 50,70,-50,-70,60,80,-60,-80

matr 4,3,-7,-8,2,1,5,-6,8,7,-3,-4,-6,5,-1,-2

# Основные теоретические положения.

Большинство команд процессора Intel Х86 выполняются с аргументами, которые принято называть *операндами*. Операнды в программе могут задаваться следующим образом:

• в регистрах общего назначения;

• непосредственно в коде команды;

• в ячейках памяти, задаваемых в команде прямо или косвенно;

• в портах ввода-вывода.

Для указания места расположения операнда используются 8 режимов адресации.

1. **Регистровая адресация**

Операнды могут располагаться в любых регистрах общего назначения и сегментных регистрах. В этом случае в операторе программы (на языке ассемблера) указывается название соответствующего регистра.

2. **Непосредственная адресация**

Некоторые команды (пересылки, все арифметические команды, кроме деления) по­зволяют указывать один из операндов непосредственно в операторе про­граммы.

3. **Прямая адресация**

Если известен адрес операнда, располагающегося в памяти, можно ис­пользовать этот адрес. В реальных программах обычно для задания статических переменных используют директивы определения данных, которые позволяют ссылаться на статические пере­менные не по адресу, а по имени.

Если селектор сегмента данных находится в DS, имя сегментного регистра при прямой адресации можно не указывать, так как DS используется по умолчанию. Прямая адресация иногда называется ад­ресацией по смещению.

4. **Косвенная адресация**

Адрес операнда в памяти можно не указывать непосредственно, а хранить в любом регистре. Как и в случае прямой адресации, DS используется по умолчанию, но не во всех случаях: если смещение берут из регистров ESP, ЕВР или ВР, то в качестве сегментного регистра используется SS. В реальном режиме можно свободно пользоваться всеми 32-битными регистрами, надо толь­ко следить, чтобы их содержимое не превышало границ 16-битного слова.

5-6. **Базовая или индексная адресация.**

Та­кая форма адресации используется в тех случаях, когда в регистре находится адрес начала структуры данных, а доступ надо осуществить к какому-нибудь элементу этой структуры. Другое важное применение адресации по базе со сдвигом – доступ из подпрограммы к параметрам, переданным в кадре стека, используя регистр ВР (ЕВР) в качестве базы и но­мер параметра в качестве смещения.

С помощью этого метода можно организовывать доступ к одномерным массивам байт: смещение соответствует адресу начала массива, а число в регистре – индексу эле­мента массива, который надо использовать. Очевидно, что если массив состоит не из байт, а из слов, придется умножать базовый регистр на два, а если из двойных слов – на четыре. Для этого предусмотрен следующий специальный метод адресации.

7. **Индексная адресация с масштабированием**

Этот метод адресации полностью идентичен предыдущему, за исключе­нием того, что с его помощью можно прочитать элемент массива слов, двойных слов или учетверенных слов, просто поместив номер элемента в регистр

*mov ax, [esi\*2]+2*

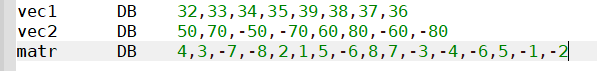
Множитель, который может быть равен 1, 2, 4 или 8, соответствует размеру элемента массива – байту, слову, двойному слову, учетверенно­му слову соответственно. Из регистров в этом варианте адресации мож­но использовать только ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI, ЕВР, ESP, но не SI, DI, ВР или SP, которые можно было использовать в предыдущих ва­риантах.

8. **Адресация по базе с индексированием и масштабированием**

Это самая полная возможная схема адресации, в которую входят все случаи, рассмотренные ранее, как частные. Смещение может быть байтом, словом или двойным словом. Если регистры ЕВР или ESP используются в роли базового регистра, селектор сегмента операн­да берется по умолчанию из регистра SS, во всех остальных случаях – из DS.

# Выполнение работы

1. Необходимые для выполнения исходные данные были получены и занесены в файл lr\_comp.asm. (см. Рисунок 1)

Рисунок 1

1. Программа была протранслирована. Были обнаружены несколько ошибок и предупреждений(см. Рисунок 2).

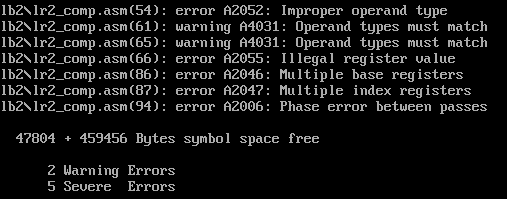


Рисунок 2

Ошибки были закомментированы, предупреждения — нет. В результате появилась ошибка — функция *Main* не завершается(см. Рисунок 3).

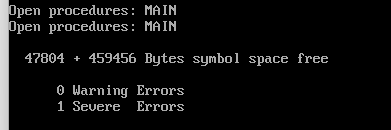


Рисунок 3

Завершение процедуры было раскомментировано, и программа успешно протранслировалась без ошибок.

**Объяснение ошибок и предупреждений:**

* *error a2052***.** Нельзя перемещать напрямую данные из разных областей памяти напрямую, нужно использовать регистр
* *warning a4031.* Возникает из-за несоответсвия типов. В нашем случае по причине попытки загрузки в 16-битный регистр cx однобайтного значения из vec2. В строке 65 — аналогично.
* *error a2055.* Возникла из-за умножения регистра на 4. В базово-индексной адресации нужно увеличить значение регистра до начала итераций.
* *error a2046.* В базированной адресации необходимо указывать базовый регистр, затем производить смещение с помощью индексного. Так как здесь оба базовые, надо сначала сложить значения регистров, затем уже передавать информацию указателю из одного регистра.
* *error a2047.* Здесь в квадратных скобках происходит сложение двух индексных регистров. Но в этом виде адресации выражение может содержать только один такой регистр.

1. Программа была заново протранслирована и слинкована. При попытке запуска в автоматическом режиме она не завершается корректно, поскольку она учебная.
2. Программа была прогнана в отладчике. Все шаги см. в Таблице 1.

Таблица 1 - Протокол пошагового исполнения lr2\_comp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес  команды | Символический  код команды | 16 - ричный  код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0000 | PUSH DS | 1E | Stack(+0) = 0000  (IP) = 0000  (SP) = 0018 | Stack(+0) = 119C (IP) = 0001  (SP) = 0016 |
| 0001 | SUB AX, AX | 2BC0 | (IP) = 0001  (AX) = 0000 | (IP) = 0003  (AX) = 0000 |
| 0003 | PUSH AX | 50 | (IP) = 0003  (SP) = 0016 Stack(+0) = 119C Stack(+2) = 0000 | (IP) = 0004  (SP) = 0014  Stack(+0) = 0000 Stack(+2) = 119C |
| 0004 | MOV AX,11AE | B8AE11 | (AX) = 0000  (IP) = 0004 | (AX) = 11AE (IP) = 0007 |
| 0007 | MOV DS,AX | 8ED8 | (DS) = 119C (IP) = 0007 | (DS) = 11AE (IP) = 0009 |
| 0009 | MOV AX,01F4 | B8F401 | (IP) = 0009 (AX) =11AE | (IP) = 000C  (AX) = 01F4 |
| 000C | MOV CX, AX | 8BC8 | (IP) = 000C  (CX)=00B6 | (IP) = 000E  (CX)=01F4 |
| 000E | MOV BL, 24 | B324 | (BX)=0000 (IP) = 000E | (BX)=0024 (IP) = 0010 |
| 0010 | MOV BH, CE | B7CE | (IP) = 0010  (BX)=0024 | (IP) = 0012  (BX)=CE24 |
| 0012 | MOV [0002], FFCE | C7060200CEFF | (IP) = 0012  (DS:0002) = 0000 | (IP) = 0018  (DS:0002) = FFCE |
| 0018 | MOV BX,0006 | BB0600 | (IP) = 0018  (BX) = CE24 | (IP) = 001B  (BX) = 0006 |
| 001B | MOV [0000], AX | A30000 | (IP) = 001B  (DS:0000)=0000 | (IP) = 001E  (DS:0000) = 01F4 |
| 001E | MOV AL, [BX] | 8A07 | (IP) = 001E  (AX) = 01F4 | (IP) = 0020  (AX) = 0120 |
| 0020 | MOV AL, [BX+03] | 8A4703 | (IP) = 0020  (AX)=0120 | (IP) = 0024  (AX)=0123 |
| 0023 | MOV CX, [BX+03] | 8B4F03 | (IP) = 0023  (CX)=01F4 | (IP) = 0026  (CX)=2723 |
| 0026 | MOV DI, 0002 | BF0200 | (IP) = 0026  (DI) = 0000 | (IP) = 0029  (DI) = 0002 |
| 0029 | MOV AL, [DI+000E] | 8A850E00 | (AX) = 0123  (IP) = 0029 | (AX) =01CE  (IP) = 002D |
| 002D | MOV CX, [DI+000E] | 8B8D0E00 | (CX) = 2723  (IP) = 002D | (CX)=BACE  (IP) = 0031 |
| 0031 | MOV BX, 0003 | BB0300 | (BX) = 0006  (IP) = 0031 | (BX)=0003  (IP)=0034 |
| 0034 | MOV AL,[BX+DI+0016] | 8A811600 | (AX) = 01CE  (IP) = 0034 | (AX)=0101  (IP)=0038 |
| 0038 | MOV CX,[BX+DI+0016] | 8B891600 | (CX) =BACE  (IP) = 0038 | (CX)=0501  (IP)=003C |
| 003C | MOV AX,11AE | B8AE11 | (AX) = 0101  (IP) = 003C | (AX)=11AE  (IP)=003F |
| 003F | MOV ES, AX | 8EC0 | (ES) = 119C  (IP) = 003F | (ES)=11AE  (IP)=0041 |
| 0041 | MOV AX, ES:[BX] | 268B07 | (AX) = 11AE  (IP) = 0041 | (AX)=00FF  (IP)=0044 |
| 0044 | MOV AX, 0000 | B80000 | (AX) = 00FF  (IP) = 0044 | (AX)=0000  (IP)=0047 |
| 0047 | MOV ES, AX | 8EC0 | (ES) = 11AE  (IP) = 0047 | (ES)=0000  (IP)=0047 |
| 0049 | PUSH DS | 1E | Stack(+0) = 0000  Stack(+2) = 119C  (IP) = 0049  (SP) = 0014 | Stack(+0) =11AE  Stack(+2)=0000  Stack(+4) = 119C  (IP) = 004A  (SP) = 0012 |
| 004A | POP ES | 07 | Stack(+0)=11AE  Stack(+2)=0000  Stack(+4) =119C  (IP) = 004A  (SP) = 0012  (ES) = 0000 | Stack(+0) = 0000  Stack(+2) = 119C  Stack(+4) = 0000  (IP) = 004B  (SP) = 0014  (ES) = 11AE |
| 004B | MOV CX,ES:[BX-01] | 268B4FFF | (CX) = 0501  (IP) = 004B | (CX)=FFCE  (IP)=004F |
| 004F | XCHG AX, CX | 91 | (AX) = 0000  (CX) = FFCE  (IP)=004F | (CX) = 0000  (AX) = FFCE  (IP)=0050 |
| 0050 | MOV DI,0002 | BF0200 | (DI) = 0002  (IP)=0050 | (DI) = 0002  (IP)=0053 |
| 0053 | MOV ES:[BX+DI], AX | 268901 | (ES:[BX+DI]) = 11AE  (IP)=0053 | (ES:[BX+DI]) = FFCE  (IP)=0056 |
| 0056 | MOV BP, SP | 8BEC | (BP) = 0000  (IP)=0056 | (BP) = 0014  (IP)=0058 |
| 0058 | PUSH [0000] | FF360000 | Stack(+0) = 0000  Stack(+2) = 119C  Stack(+4) = 0000  (IP)=0058 | Stack(+0) = 01F4  Stack(+2) = 0000  Stack(+4) = 119C  (IP)=005C |
| 005C | PUSH [0002] | FF360200 | Stack(+0) = 01F4  Stack(+2) = 0000  Stack(+4) = 119C  (IP) = 005C | Stack(+0) = FFCE  Stack(+2) = 01F4  Stack(+4) = 0000  Stack(+6) = 119C  (IP) = 0060 |
| 0060 | MOV BP, SP | 8BEC | (BP) = 0014  (IP)=0060 | (BP) = 0010  (IP) = 0062 |
| 0062 | MOV DX, [BP+02] | 8B5602 | (DX) = 0000  (IP) = 0062 | (DX) = 01F4  (IP) = 0065 |
| 0065 | RET Far | CB | Stack(+0) = FFCE | Stack(+0) = 0000 |

# Вывод.

В ходе выполнения лабораторной работы были изучены способы адресации основной памяти в ассемблере.

# ПРИЛОЖЕНИЕ А

1. lr2\_comp.ASM

­

; Учебная программа лабораторной работы №2 по дисциплине "Организация ЭВМ и С";

;

EOL EQU '$'

ind EQU 2

n1 EQU 500

n2 EQU -50

; Стек программы

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

; Данные программы

DATA SEGMENT

; Директивы описания данных

mem1 DW 0

mem2 DW 0

mem3 DW 0

vec1 DB 32,33,34,35,39,38,37,36

vec2 DB 50,70,-50,-70,60,80,-60,-80

matr DB 4,3,-7,-8,2,1,5,-6,8,7,-3,-4,-6,5,-1,-2

DATA ENDS

; Код программы

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура

Main PROC FAR

push DS

sub AX,AX

push AX

mov AX,DATA

mov DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИЙ

; Регистровая адресация

mov ax,n1

mov cx,ax

mov bl,EOL

mov bh,n2

; Прямая адресация

mov mem2,n2

mov bx,OFFSET vec1

mov mem1,ax

; Косвенная адресация

mov al,[bx]

; mov mem3,[bx]

; Базированная адресация

mov al,[bx]+3

; mov cx,3[bx]

; Индексированная адресация

mov di,ind

mov al,vec2[di]

mov cx,vec2[di]

; Адресация с базированием и индексированием

mov bx,3

mov al,matr[bx][di]

mov cx,matr[bx][di]

; mov ax,matr[bx\*4][di]

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ вариант 1

mov ax, SEG vec2

mov es, ax

mov ax, es:[bx]

mov ax, 0

; ------ вариант 2

mov es, ax

push ds

pop es

mov cx, es:[bx-1]

xchg cx,ax

; ------ вариант 3

mov di,ind

mov es:[bx+di],ax

; ------ вариант 4

mov bp,sp

; mov ax,matr[bp+bx]

; mov ax,matr[bp+di+si]

; Использование сегмента стека

push mem1

push mem2

mov bp,sp

mov dx,[bp]+2

ret

Main ENDP

CODE ENDS

END Main

2. lr2.lst

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 17:35:10

Page 1-1

; Учебная программа лабораторной работы №2 по

дисциплине "Организация ЭВМ и С";

;

= 0024 EOL EQU '$'

= 0002 ind EQU 2

= 01F4 n1 EQU 500

=-0032 n2 EQU -50

; Стек программы

0000 AStack SEGMENT STACK

0000 000C[ DW 12 DUP(?)

????

]

0018 AStack ENDS

; Данные программы

0000 DATA SEGMENT

; Директивы описания данных

0000 0000 mem1 DW 0

0002 0000 mem2 DW 0

0004 0000 mem3 DW 0

0006 20 21 22 23 27 26 vec1 DB 32,33,34,35,39,38,37,36

25 24

000E 32 46 CE BA 3C 50 vec2 DB 50,70,-50,-70,60,80,-60,-80

C4 B0

0016 04 03 F9 F8 02 01 matr DB 4,3,-7,-8,2,1,5,-6,8,7,-3,-4,-6

,5,-1,-2

05 FA 08 07 FD FC

FA 05 FF FE

0026 DATA ENDS

; Код программы

0000 CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

; Головная процедура

0000 Main PROC FAR

0000 1E push DS

0001 2B C0 sub AX,AX

0003 50 push AX

0004 B8 ---- R mov AX,DATA

0007 8E D8 mov DS,AX

; ПРОВЕРКА РЕЖИМОВ АДРЕСАЦИИ НА УРОВНЕ СМЕЩЕНИ

Й

; Регистровая адресация

0009 B8 01F4 mov ax,n1

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 17:35:10

Page 1-2

000C 8B C8 mov cx,ax

000E B3 24 mov bl,EOL

0010 B7 CE mov bh,n2

; Прямая адресация

0012 C7 06 0002 R FFCE mov mem2,n2

0018 BB 0006 R mov bx,OFFSET vec1

001B A3 0000 R mov mem1,ax

; Косвенная адресация

001E 8A 07 mov al,[bx]

; mov mem3,[bx]

; Базированная адресация

0020 8A 47 03 mov al,[bx]+3

; mov cx,3[bx]

; Индексированная адресация

0023 BF 0002 mov di,ind

0026 8A 85 000E R mov al,vec2[di]

002A 8B 8D 000E R mov cx,vec2[di]

lb2\lr2\_comp.asm(61): warning A4031: Operand types must match

; Адресация с базированием и индексированием

002E BB 0003 mov bx,3

0031 8A 81 0016 R mov al,matr[bx][di]

0035 8B 89 0016 R mov cx,matr[bx][di]

lb2\lr2\_comp.asm(65): warning A4031: Operand types must match

; mov ax,matr[bx\*4][di]

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ вариант 1

0039 B8 ---- R mov ax, SEG vec2

003C 8E C0 mov es, ax

003E 26: 8B 07 mov ax, es:[bx]

0041 B8 0000 mov ax, 0

; ------ вариант 2

0044 8E C0 mov es, ax

0046 1E push ds

0047 07 pop es

0048 26: 8B 4F FF mov cx, es:[bx-1]

004C 91 xchg cx,ax

; ------ вариант 3

004D BF 0002 mov di,ind

0050 26: 89 01 mov es:[bx+di],ax

; ------ вариант 4

0053 8B EC mov bp,sp

; mov ax,matr[bp+bx]

; mov ax,matr[bp+di+si]

; Использование сегмента стека

0055 FF 36 0000 R push mem1

0059 FF 36 0002 R push mem2

005D 8B EC mov bp,sp

005F 8B 56 02 mov dx,[bp]+2

0062 CB ret

0063 Main ENDP

0063 CODE ENDS

END Main

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 17:35:10

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0063 PARA NONE

DATA . . . . . . . . . . . . . . 0026 PARA NONE

Symbols:

N a m e Type Value Attr

EOL . . . . . . . . . . . . . . NUMBER 0024

IND . . . . . . . . . . . . . . NUMBER 0002

MAIN . . . . . . . . . . . . . . F PROC 0000 CODE Length = 0063

MATR . . . . . . . . . . . . . . L BYTE 0016 DATA

MEM1 . . . . . . . . . . . . . . L WORD 0000 DATA

MEM2 . . . . . . . . . . . . . . L WORD 0002 DATA

MEM3 . . . . . . . . . . . . . . L WORD 0004 DATA

N1 . . . . . . . . . . . . . . . NUMBER 01F4

N2 . . . . . . . . . . . . . . . NUMBER -0032

VEC1 . . . . . . . . . . . . . . L BYTE 0006 DATA

VEC2 . . . . . . . . . . . . . . L BYTE 000E DATA

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT lr2\_comp

@VERSION . . . . . . . . . . . . TEXT 510

96 Source Lines

96 Total Lines

19 Symbols

47804 + 459456 Bytes symbol space free

2 Warning Errors

0 Severe Errors

#Microsoft (R) Macro Assembler Version 5.10 9/18/24 21:06:58

Page 1-2