МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Организация ЭВМ и систем»

Тема: Изучение режимов адресации и формирования исполнительного адреса

Студент гр. 9382 _____ Михайлов Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучение и получение практических навыков по режимам адресации основной памяти в языке Ассемблер.

Постановка задачи.

Лабораторная работа 2 предназначена для изучения режимов адресации, использует готовую программу на Ассемблере, которая в автоматическом режиме выполняться не должна, так как не имеет самостоятельного функционального назначения, а только тестирует режимы адресации. Поэтому ее выполнение должно производиться под управлением отладчика в пошаговом режиме.

В программу введен ряд ошибок, которые необходимо объяснить в отчете по работе, а соответствующие команды закомментировать для прохождения трансляции.

Необходимо составить протокол выполнения программы в пошаговом режиме отладчика по типу таблицы предыдущей лабораторной работы. На защите студенты должны уметь объяснить результат выполнения каждой команды.

Выполнение работы.

Вариант №3.

Сначала режиме редактирования в текстовом редакторе NotePad++ в программе был изменён набор значений исходных данных (массивов) `vec1, vec2 и matr`, согласно заданному варианту.

```
vec1 8,7,6,5,1,2,3,4
vec2 -30,-40,30,40,-10,-20,10,20
matr -1,-2,-3,-4,8,7,6,5,-5,-6,-7,-8,4,3,2,1
```

После этого был запущен эмулятор DOSBox, создающий DOS-окружение. В нём при помощи команды:

```
> mount c:\MASM
```

Был смонтирован диск c, в который потом был совершён переход командой:

```
> c:
```

После данных операций была произведена настройка поддержки кириллицы в окружении DOS, путём ввода команды:

```
> Keyb Ru 866
```

Далее, программа была протранслирована в эмуляторе DOSBox с помощью команды masm:

```
> masm LR2_comp.ASM
```

Был создан файл листинга L1.LST, через который были просмотрены и проанализированы причины возникновения ошибок: всего при трансляции было выведено 5 сообщений об ошибках и 2 предупреждения.

Объяснение возникших ошибок:

- error A2052: Improper operand type (*Неверный тип операнда*)

  Фрагмент кода (55 стр.): `mov  mem3,[bx]`

  Нельзя читать из памяти и писать в неё одной командой. Сначала нужно перенести информацию в регистр, а потом из регистра перенести в память.

  В данном случае осуществляется попытка перемещения данных непосредственно из одного сегмента памяти в другой, без использования регистров, что неосуществимо на процессорах Intel X86.

  Пример возможного исправления:
  ```
  mov ax,[bx]
  mov mem3,ax
  ```

- warning A4031: Operands types must match (*Типы операндов должны совпадать*)

  фрагмент кода (62 стр. и 66 стр.):
  ```
  mov  cx,vec2[di]
  mov  cx,matr[bx][di]
  ```

  Размер элементов массива ‘`vec2`’ 1 байт, а ‘`cx'` - 2 байта.

  Размер элементов матрицы ‘`matr`' 1 байт, а ‘`cx`' - 2 байта.

Данное предупреждение говорит о том, что осуществляется попытка в двухбайтовый регистр `cx` записать однобайтовый элемент массива. В данном случае в регистр `cx` в младший байт записывается указанный элемент массива, а в старший байт записывается следующий за указанным элемент массива.

Пример возможного исправления:
```
mov ch,vec2[di]
mov cl,matr[bx][di]
```

CH и CL 8 битовые, т.е. 1 байтные регистры.

- error A2055: Illegal register value (*Неверное значение регистра*)

  фрагмент кода (67 стр.): `mov  ax,matr[bx*4][di]`

  Здесь используется базово-индексная адресация. При данном типе адресации надо сначала изменить значение регистра, затем уже переводить информацию.

  Пример возможного исправления:
  ```
  mov cl, 4
  imul bx,cl
  mov al,matr[bx][di]
  ```

  `imul` - умножение со знаком. Первый операнд определяет местоположение первого сомножителя. На его место впоследствии будет записан результат. Второй операнд определяет местоположение второго сомножителя.

- error A2046: Multiple base registers (*Несколько базовых регистров*)

  фрагмент кода (87 стр.): `mov  ax,matr[bp+bx]`

  Нельзя использовать несколько базовых регистров

  В данном случае используется базовая адресация. При ее использовании нельзя складывать регистры bp и bx. В базовой адресации необходимо указывать один базовый регистр, затем производить смещение. Так как здесь оба базовые, надо сначала сложить значения регистров, затем уже передавать информацию указателю из одного регистра.

  Пример возможного исправления:
  ```
  add bp,bx
  mov ax,matr[bp]
  ```

- error A2047: Multiple index registers

  фрагмент кода (88 стр.): `mov  ax,matr[bp+di+si]`

  Нельзя использовать несколько индексных регистров.

  В данном случае используется базово-индексная адресация. При ее использовании нельзя осуществлять сложение двух индексных регистров `di` и `si`, а также как и в предыдущей ошибке необходимо предварительно сложить значения этих регистров, а затем уже использовать адресацию с одним индексным регистром.

  Пример возможного исправления:
  ```
  add di,si
  mov ax,matr[bp+di]
  ```

- error A2006: Phase error between passes (*Ошибка фазы между проходами*)

  фрагмент кода (95 стр.): `Main  ENDP`

  Данная ошибка свидетельствует о том, что в функции main содержатся ошибки – неверное завершение кода программы.

После изучения всех ошибок, строки, их содержащие, были закомментированы. Далее программа была вновь протранслирована и был сформирован загрузочный модуль. Для этого использовались команды:

```
>masm LR2_comp.ASM
>link LR2_comp.OBJ
```

Затем программа была выполнена в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды.

Данные	**Смещение в сегменте DS**
mem1	0000
mem2	0002
mem3	0004
vec1	0006
vec2	000E
matr	0016

Имя	**Значение**	**Шестнадцатеричный код**
EOL	‘$’	24
ind	2	2
n1	500	1F4
n2	-50	FFCE

Адрес команды	Символьный код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	(SP) = 0018 (IP) = 0000 Stack: +0 0000 +2 0000 +4 0000 +6 0000	(SP) = 0016 (IP) = 0001 Stack: +0 119C +2 0000 +4 0000 +6 0000
0001	SUB AX, AX	2BC0	(AX) = 0000 (IP) = 0001	(AX) = 0000 (IP) = 0003
0003	PUSH AX	50	(AX) = 0000 (SP) = 0016 (IP) = 0003 Stack: +0 119C +2 0000 +4 0000 +6 0000	(AX) = 0000 (SP) = 0014 (IP) = 0004 Stack: +0 0000 +2 119C +4 0000 +6 0000
0004	MOV AX, 11 AE	B8AE11	(AX) = 0000 (IP) = 0004	(AX) = 11AE (IP) = 0007
0007	MOV DS,AX	8ED8	(AX) = 11AE (DS) =119C (IP) = 0007	(AX) = 11AE (DS) = 11AE (IP) = 0009
0009	MOV AX, 01F4	B8F401	(AX) = 11AE (IP) = 0009	(AX) = 01F4 (IP) = 000C
000C	MOV CX,AX	8BC8	(AX) = 01F4 (IP) = 000C (CX) = 00B6	(AX) = 01F4 (IP) = 000E (CX) = 01F4
000E	MOV BL,24	B324	(BX) = 0000 (IP) = 000E	(BX) = 0024 (IP) = 0010
0010	MOV BH,CE	B7CE	(BX) = 0024 (IP) = 0010	(BX) = CE24 (IP) = 0012
0012	MOV [0002],FFCE	C7060200CEFF	(IP) = 0012 DS[0002] = 00 DS[0003] = 0	(IP) = 0018 DS[0002] = CE DS[0003] = FF
0018	MOV BX, 0006	BB0600	(BX) = CE24 (IP) = 0018	(BX) = 0006 (IP) = 001B

001B	MOV [0000], AX	A30000	(AX) = 01F4 (IP) = 001B DS[0000] = 00 DS[0001] = 00	(AX) = 01F4 (IP) = 001E DS[0000] = F4 DS[0001] = 01
001E	MOV AL, [BX]	8A07	(AX) = 01F4 (BX) = 0006 (IP) = 001E	(AX) = 0108 (BX) = 0006 (IP) = 0020
0020	MOV AL, [BX+03]	8A4703	(IP)= 0020 (AX)= 0108	(IP) = 0023 (AX) = 0105
0023	MOV CX, [BX+03]	8B4F03	(CX) = 01F4 (BX) = 0006 (IP) = 0023	(CX) = 0105 (BX) = 0006 (IP) = 0026
0026	MOV DI, 0002	BF0200	(DI) = 0000 (IP) = 0026	(DI) = 0002 (IP) = 0029
0029	MOV AL, [DI+ 000E]	8A850E00	(AX) = 0105 (DI) = 0002 (IP) = 0029	(AX) = 011E (DI) = 0002 (IP) = 002D
002D	MOV CX, [DI+ 000E]	8A850E00	(CX) = 0105 (DI) = 0002 (IP) = 002D	(CX) = 281E (DI) = 0002 (IP) = 0031
0031	MOV BX, 0003	BB03000	(IP) = 0031 (BX) = 0006	(IP) = 0034 (BX) = 0003
0034	MOV AL, [BX+DI+0016]	8A811600	(AX) = 011E (IP) = 0034 (DI) = 0002 (BX) = 0003	(AX) = 0107 (IP) = 0038 (DI) = 0002 (BX) = 0003
0038	MOV CX, [BX+DI+0016]	8B891600	(IP) = 0034 (DI) = 0002 (BX) = 0003 (CX) = 281E	(IP) = 003C (DI) = 0002 (BX) = 0003 (CX) = 0203
003C	MOV AX, 11AE	B8AE11	(AX) = 0107 (IP) = 003C	(AX) = 11AE (IP) = 003F
003F	MOV ES, AX	8EC0	(AX) = 11AE (ES) = 119C (IP) = 003F	(AX) = 11AE (ES) = 11AE (IP) = 0041
0041	MOV AX, ES:[BX]	268B07	(AX) = 11AE (ES) = 11AE (BX) = 0003 (IP) = 0041	(AX) = 00FF (ES) = 11AE (BX) = 0003 (IP) = 0044
0044	MOV AX, 0000	B80000	(AX) = 00FF (IP) = 0044	(AX) = 0000 (IP) = 0047
0047	MOV ES, AX	8EC0	(AX) = 0000 (ES) = 11AE (IP) = 0047	(AX) = 0000 (ES) = 0000 (IP) = 0049

0049	PUSH DS	1E	(SP) = 0014 (DS) = 11AE (IP) = 0049 Stack: +0 0000 +2 119C +4 0000 +6 0000	(SP) = 0012 (DS) = 11AE (IP) = 004A Stack: +0 11AE +2 0000 +4 119C +6 0000
004A	POP ES	07	(SP) = 0012 (ES) = 0000 (IP) = 004A Stack: +0 11AE +2 0000 +4 119C +6 0000	(SP) = 0014 (ES) = 11AE (IP) = 004B Stack: +0 0000 +2 119C +4 0000 +6 0000
004B	MOV CX, ES:[BX-01]	268B4FFF	(CX) = 0607 (ES) = 11AE (BX) = 0003 (IP) = 004B	(CX) = FFCE (ES) = 11AE (BX) = 0003 (IP) = 004F
004F	XCHG AX, CX	91	(AX) = 0000 (CX) = FFCE (IP) = 004F	(AX) = FFCE (CX) = 0000 (IP) = 0050
0050	MOV DI, 0002	BF0200	(DI) = 0002 (IP) = 0050	(DI) = 0002 (IP) = 0053
0053	MOV ES:[BX+DI], AX	268901	(ES) = 11AE DS[0005] = 00 DS[0006] = 12 (DI) = 0002 (BX) = 0003 (IP) = 0053	(ES) = 11AE DS[0005] = CE DS[0006] = FF (DI) = 0002 (BX) = 0003 (IP) = 0056
0056	MOV BP, SP	8BEC	(BP) = 0000 (SP) = 0014 (IP) = 0056	(BP) = 0014 (SP) = 0014 (IP) =0058
0058	PUSH [0000]	FF360000	(SP) = 0014 (IP) = 0058 Stack: +0 0000 +2 119C +4 0000 +6 0000	(SP) = 0012 (IP) = 005C Stack: +0 01F4 +2 0000

				+4 119C +6 0000
005C	PUSH [0002]	FF360200	(SP) = 0012 (IP) = 005C Stack: +0 01F4 +2 0000 +4 119C +6 0000	(SP) = 0010 (IP) = 0060 Stack: +0 FFCE +2 01F4 +4 0000 +6 119C
0060	MOV BP, SP	8BEC	(BP) = 0014 (SP) = 0010 (IP) = 0060	(BP) = 0010 (SP) = 0010 (IP) = 0062
0062	MOX DX, [BP+02]	8B5602	(DX) = 0000 (BP) = 0010 (IP) = 0062	(DX) = 01F4 (BP) = 0010 (IP) = 0065
0065	RET FAR	CA0200	(CS) = 11B1 (SP) = 0010 (IP) = 0065 Stack: +0 FFCE +2 01F4 +4 0000 +6 119C	(CS) = 01F4 (SP) = 0014 (IP) = FFCE Stack: +0 0000 +2 119C +4 0000 +6 0000

Выводы.

В ходе работы были получены теоретические практические навыки по использованию режимов адресации основной памяти в языке Ассемблер, были изучены некоторые ошибки, возникающие при неправильной адресации.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

Название файла: LAB2.ASM

```
EOL   EQU   '$'
ind   EQU    2
n1    EQU   500
n2    EQU   -50
AStack     SEGMENT  STACK ; Стек  программы
           DW 12 DUP(?)
AStack     ENDS
DATA       SEGMENT ; Данные программы
;  Директивы описания данных
mem1       DW    0
mem2       DW    0
mem3       DW    0
vec1       DB    8,7,6,5,1,2,3,4
vec2       DB    -30,-40,30,40,-10,-20,10,20
matr       DB    -1,-2,-3,-4,8,7,6,5,-5,-6,-7,-8,4,3,2,1
DATA       ENDS

CODE       SEGMENT ; Код программы
           ASSUME CS:CODE, DS:DATA, SS:AStack
Main       PROC  FAR ; Головная процедура
           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
```

Название файла: LIST.LST

```
#Microsoft  (R)  Macro  Assembler  Version  5.10
10/16/20 14:38:3

Page     1-1


 = 0024             EOL  EQU  '$'
 = 0002             ind  EQU   2
 = 01F4             n1   EQU  500
 =-0032             n2   EQU  -50
```

```
 0000                   AStack    SEGMENT  STACK ;
 0000  000C[                      DW 12 DUP(?)
       ????
             ]

 0018                   AStack    ENDS

 0000                   DATA      SEGMENT ;
                  ;
 0000  0000             mem1      DW    0
 0002  0000             mem2      DW    0
 0004  0000             mem3      DW    0
 0006  08 07 06 05 01 02 vec1         DB    8,7,6,5,1,2,3,4
       03 04
 000E  E2 D8 1E 28 F6 EC vec2         DB    -30,-40,30,40,-
10,-20,10,20
       0A 14
 0016  FF FE FD FC 08 07 matr         DB    -1,-2,-3,-
4,8,7,6,5,-5,-6,-7,-8
                  ,4,3,2,1
       06 05 FB FA F9 F8
       04 03 02 01
 0026                   DATA      ENDS

 0000                   CODE      SEGMENT ;
                              ASSUME CS:CODE, DS:DATA,
SS:AStack

 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
```

```
                ;
                ;
0009  B8 01F4                     mov  ax,n1
000C  8B C8                   mov  cx,ax
000E  B3 24                   mov  bl,EOL
0010  B7 CE                   mov  bh,n2
                ;
0012  C7 06 0002 R FFCE           mov  mem2,n2
0018  BB 0006 R               mov  bx,OFFSET vec1
001B  A3 0000 R               mov  mem1,ax
                ;
001E  8A 07                   mov  al,[bx]
                            ;mov  mem3,[bx]
                ;
0020  8A 47 03                    mov  al,[bx]+3
0023  8B 4F 03                    mov  cx,3[bx]
                ;
```

```
#Microsoft (R) Macro Assembler Version 5.10
10/16/20 14:38:3

Page     1-2


 0026  BF 0002                      mov  di,ind
 0029  8A 85 000E R                 mov  al,vec2[di]
 002D  8B 8D 000E R                 mov  cx,vec2[di]
lab2.ASM(49): warning A4031: Operand types must match
                 ;
 0031  BB 0003                      mov  bx,3
 0034  8A 81 0016 R                 mov  al,matr[bx][di]
 0038  8B 89 0016 R                 mov  cx,matr[bx][di]
lab2.ASM(53): warning A4031: Operand types must match
                           ;mov  ax,matr[bx*4][di]
                 ;
                 ;
                 ;
 003C  B8 ---- R                mov  ax, SEG vec2
 003F  8E C0                    mov  es, ax
 0041  26: 8B 07                mov  ax, es:[bx]
 0044  B8 0000                   mov  ax, 0
                 ;
 0047  8E C0                    mov  es, ax
 0049  1E                       push ds
 004A  07                       pop  es
 004B  26: 8B 4F FF                 mov  cx, es:[bx-1]
 004F  91                       xchg cx,ax
                 ;
 0050  BF 0002                      mov  di,ind
 0053  26: 89 01                mov  es:[bx+di],ax
                 ;
 0056  8B EC                    mov  bp,sp
                           ;mov  ax,matr[bp+bx]
```

```
                                ;mov  ax,matr[bp+di+si]
                    ;
0058  FF 36 0000 R                    push  mem1
005C  FF 36 0002 R                    push  mem2
0060  8B EC                       mov   bp,sp
0062  8B 56 02                        mov   dx,[bp]+2
0065  CB                          ret
0066                  Main        ENDP
0066                  CODE        ENDS
                                END Main
```

#Microsoft (R) Macro Assembler Version 5.10
10/16/20 14:38:3

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0018	PARA	STACK
CODE	0066	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 = 0066			
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  lab2
@VERSION . . . . . .      TEXT  510


     83 Source  Lines
     83 Total   Lines
     19 Symbols

  47812 + 459448 Bytes symbol space free

      2 Warning Errors
      0 Severe  Errors
```