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

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Организация ЭВМ и систем»
Тема: Изучение режимов адресации и формирования исполнительного адреса

Студент гр. 0382 _____ Шангичев В. А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

Изучить работу режимов адресации, исправить ошибки в данной программе, закомментировав ошибочные строки.

Задание.

Лабораторная работа 2 предназначена для изучения режимов адресации, использует готовую программу lr2_comp.asm 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 1 предыдущей лабораторной работы и подписать его у преподавателя. На защите студенты должны уметь объяснить результат выполнения каждой команды с учетом используемого вида адресации. Результаты, полученные с помощью отладчика, не являются объяснением, а только должны подтверждать ваши объяснения.

Порядок выполнения работы.

1. Получить у преподавателя вариант набора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2.dat, приведенного в каталоге Задания и занести свои данные вместо значений, указанных в приведенной ниже программе.
2. Протранслировать программу с созданием файла диагностических сообщений; объяснить обнаруженные ошибки и закомментировать соответствующие операторы в тексте программы.
3. Снова протранслировать программу и скомпоновать загрузочный модуль.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команды.
5. Результаты прогона программы под управлением отладчика должны быть подписаны преподавателем и представлены в отчете.

Выполнение работы.

Описание ошибок, полученных при первоначальной трансляции:

1. `mov mem3,[bx]- error A2053: Improper operand type.` Операции типа память -> память недопустимы. Одним из вариантов устранения ошибки может быть запись памяти в регистр `ax,` с последующей записью в `mem3`.
2. `mov cx,vec2[di] – warning A4031: Operand types must match.` Размер регистра `cx` – 2 байта, а размер элемента массива `vec2` – 1 байт. Одним из вариантов разрешения ошибки будет запись значения в `ch`.
3. `mov cx, matr[bx][di] – warning A4031: Operand types must match.` Ошибка идентична ошибке из пункта 2.
4. `mov ax,matr[bx*4][di] – error A2055: Illegal register value.` Нельзя умножать 2-х байтовые регистры. Можно записать в `bx` нужное значение перед обращением по индексу.
5. `mov ax,matr[bp+bx] – error Multiple base registers.` Для адресации можно использовать только один базовый регистр. Решение аналогично решению в пункте 4.
6. `mov ax,matr[bp+di+si] – error A2047: Multiple index registers.` Нельзя использовать несколько индексных регистров для адресации.

ПРОТОКОЛ

Таблица 1. Результат выполнения программы в пошаговом режиме.

Адрес команды	Символический код команды	Машинный код	Содержимое регистров и ячеек памяти	
			До выполнения	После

0000	PUSH DS	1E	Stack +0 0000 SP = 0018	Stack +0 19F5 SP = 0016
0001	SUB AX, AX	2BC0	AX = 0000	AX = 0000
0003	PUSH AX	50	Stack +0 19F5 +2 0000 SP = 0016	Stack +0 0000 +2 19F5 SP = 0014
0004	MOV AX, 1A07	B8071A	AX = 0000	AX = 1A07
0007	MOV DS, AX	8ED8	DS = 19F5	DS = 1A07
0009	MOV AX,01F4	B8F401	AX = 1A07	AX = 01F4
000C	MOV CX, AX	8BC8	CX = 00B0	CX = 01F4
000E	MOV BL, 24	B324	BX = 0000	BX = 0024
0010	MOV BH, CE	B7CE	BX = 0024	BX = CE24
0012	MOV [0002],FFCE	C7060200CEFF		
0018	MOV BX, 0006	BB0600	BX = CE24	BX = 0006
001B	MOV [0000], AX	A30000		
001E	MOV AL, [BX]	8A07	AX = 01F4	AX = 0126
0020	MOV AL,[BX+03]	8A4703	AX = 0126	AX = 0123
0023	MOV CX,[BX+03]	8B4F03	CX = 01F4	CX = 1F23
0026	MOV DI, 0002	BF0200	DI = 0000	DI = 0002
0029	MOV AL, [000E+DI]	8A850E00	AX = 0123	AX = 01BA
002D	MOV BX, 0003	BB0300	BX = 0006	BX = 0003
0030	MOV AL, [0016+BX+DI]	8A811600	AX = 01BA	AX = 01F9
0034	MOV AX, 1A07	B8071A	AX = 01F9	AX = 1A07
0037	MOV ES, AX	8EC0	ES = 19F5	ES = 1A07
0039	MOV AX, ES: [BX]	268B07	AX = 1A07	AX = 00FF
003C	MOV AX, 0000	B80000	AX = 00FF	AX = 0000
003F	MOV ES, AX	8EC0	ES = 1A07	ES = 0000
0041	PUSH DS	1E	Stack+0 0000 +2 19F5 SP = 0014	Stack+0 1A07 +2 0000 +4 19F5 SP = 0012
0042	POP ES	07	Stack+0 1A07 +2 0000 +4 19F5	Stack+0 0000 +2 19F5 +4 0000

			SP = 0012 ES = 0000	SP = 0014 ES = 1A07
0043	MOV CX,ES:[BX-01]	268B4FFF	CX = 1F23	CX = FFCE
0047	XCHG AX, CX	91	AX = 0000 CX = FFCE	AX = FFCE CX = 0000
0048	MOV DI, 0002	BF0200	DI = 0002	DI = 0002
004B	MOV ES: [BX+DI], AX	268901		
004E	MOV BP, SP	8BEC	BP = 0000	BP = 0014
0050	PUSH [0000]	FF360000	Stack+0 0000 +2 19F5 SP = 0014	Stack+0 01F4 +2 0000 +4 19F5 SP = 0012
0054	PUSH [0002]	FF360200	Stack+0 01F4 +2 0000 +4 19F5 SP = 0012	Stack+0FFCE +2 01F4 +4 0000 +6 19F5 SP = 0010
0058	MOV BP, SP	8BEC	BP = 0014	BP = 0010
005A	MOV DX, [BP+02]	8B5602	DX = 0000	DX = 01F4
005D	RET FAR 0002	CA0200	CS = 1A0A SP = 0010 Stack:+0 FFCE +2 01F4 +4 0000 +6 19F5 IP: 005D	CS = 01F4 SP = 0016 Stack:+0 19F5 IP: FFCE

Выводы.

В ходе данной лабораторной работы были изучены различные способы адресации и исправлены ошибки в программе.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.asm

```
; IntelX86 Processor Addressing Modes Study Program

EOL EQU '$'
ind EQU 2
n1 EQU 500
n2 EQU -50

; Program stack
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

; Program data
DATA SEGMENT

; Data description directives
mem1 DW 0
mem2 DW 0
mem3 DW 0
vec1 DB 38,37,36,35,31,32,33,34
vec2 DB 70,80,-70,-80,50,60,-50,-60
matr DB -2,-1,5,6,-8,-7,3,4,-4,-3,7,8,-6,-5,1,2
DATA ENDS


; Program code
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, SS:AStack


; Head procedure
Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

; CHECKING THE ADDRESSING MODES AT THE OFFSET LEVEL
; Register addressing
    mov ax,n1
    mov cx,ax
    mov bl,EOL
    mov bh,n2

; Direct addressing
    mov mem2,n2
    mov bx,OFFSET vec1
    mov mem1,ax

; Indirect addressing
    mov al,[bx]
    mov mem3,[bx]
```

```
; Based addressing
    mov al,[bx]+3
    mov cx,3[bx]

; Indexed addressing
    mov di,ind
    mov al,vec2[di]
    mov cx,vec2[di]

; Basing and Indexing Addressing
    mov bx,3
    mov al,matr[bx][di]
    mov cx,matr[bx][di]
    mov ax,matr[bx*4][di]

; VERIFICATION OF ADDRESSING MODES TAKING INTO ACCOUNT SEGMENTS
; Segment redefinition
; ------ variant 1
    mov ax, SEG vec2
    mov es, ax
    mov ax, es:[bx]
    mov ax, 0

; ------ variant 2
    mov es, ax
    push ds
    pop es
    mov cx, es:[bx-1]
    xchg cx,ax

; ------ variant 3
    mov di,ind
    mov es:[bx+di],ax

; ------ variant 4
    mov bp,sp
    mov ax,matr[bp+bx]
    mov ax,matr[bp+di+si]

; Using a stack segment
    push mem1
    push mem2
    mov bp,sp
    mov dx,[bp]+2
    ret 2
Main ENDP
CODE ENDS
 END Main
```

Файл mainedit.asm

```
; IntelX86 Processor Addressing Modes Study Program

EOL EQU '$'
ind EQU 2
n1 EQU 500
n2 EQU -50

; Program stack
```

```
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

; Program data
DATA SEGMENT

; Data description directives
mem1 DW 0
mem2 DW 0
mem3 DW 0
vec1 DB 38,37,36,35,31,32,33,34
vec2 DB 70,80,-70,-80,50,60,-50,-60
matr DB -2,-1,5,6,-8,-7,3,4,-4,-3,7,8,-6,-5,1,2
DATA ENDS


; Program code
CODE SEGMENT
 ASSUME CS:CODE, DS:DATA, SS:AStack


; Head procedure
Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

; CHECKING THE ADDRESSING MODES AT THE OFFSET LEVEL
; Register addressing
    mov ax,n1
    mov cx,ax
    mov bl,EOL
    mov bh,n2

; Direct addressing
    mov mem2,n2
    mov bx,OFFSET vec1
    mov mem1,ax

; Indirect addressing
    mov al,[bx]
;    mov mem3,[bx]

; Based addressing
    mov al,[bx]+3
    mov cx,3[bx]

; Indexed addressing
    mov di,ind
    mov al,vec2[di]
;    mov cx,vec2[di]

; Basing and Indexing Addressing
    mov bx,3
    mov al,matr[bx][di]
;    mov cx,matr[bx][di]
;    mov ax,matr[bx*4][di]
```

```
; VERIFICATION OF ADDRESSING MODES TAKING INTO ACCOUNT SEGMENTS
; Segment redefinition
; ------ variant 1
    mov ax, SEG vec2
    mov es, ax
    mov ax, es:[bx]
    mov ax, 0

; ------ variant 2
    mov es, ax
    push ds
    pop es
    mov cx, es:[bx-1]
    xchg cx,ax

; ------ variant 3
    mov di,ind
    mov es:[bx+di],ax

; ------ variant 4
    mov bp,sp
;    mov ax,matr[bp+bx]
;    mov ax,matr[bp+di+si]

; Using a stack segment
    push mem1
    push mem2
    mov bp,sp
    mov dx,[bp]+2
    ret 2
Main ENDP
CODE ENDS
 END Main
```

Файл main.lst

```
                        ; IntelX86 Processor Addressing Modes S
                             tudy Program

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

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

0018                    AStack ENDS

                        ; Program data
0000                    DATA SEGMENT

                        ; Data description directives
0000  0000              mem1 DW 0
0002  0000              mem2 DW 0
0004  0000              mem3 DW 0
0006  26 25 24 23 1F 20 vec1 DB 38,37,36,35,31,32,33,34
      21 22
000E  46 50 BA B0 32 3C vec2 DB 70,80,-70,-80,50,60,-50,-60
      CE C4
0016  FE FF 05 06 F8 F9 matr DB -2,-1,5,6,-8,-7,3,4,-4,-3,7,8,-
                           6,-5,1,2
      03 04 FC FD 07 08
      FA FB 01 02
0026                    DATA ENDS


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


                        ; Head procedure
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

                        ; CHECKING THE ADDRESSING MODES AT THE
                             OFFSET LEVEL
                        ; Register addressing
0009  B8 01F4                       mov ax,n1
000C  8B C8                 mov cx,ax
000E  B3 24                 mov bl,EOL
0010  B7 CE                 mov bh,n2
```

```
      52
      53                ; Direct addressing
      54 0012  C7 06 0002 R FFCE     mov mem2,n2
      55 0018  BB 0006 R             mov bx,OFFSET vec1
      56 001B  A3 0000 R             mov mem1,ax
      57
      58                ; Indirect addressing
      59 001E  8A 07                 mov al,[bx]
      60                   mov mem3,[bx]
main.asm(53): error A2052: Improper operand type
      61
      62                ; Based addressing
      63 0020  8A 47 03                   mov al,[bx]+3
      64 0023  8B 4F 03                   mov cx,3[bx]
      65
      66                ; Indexed addressing
      67 0026  BF 0002                    mov di,ind
      68 0029  8A 85 000E R          mov al,vec2[di]
      69 002D  8B 8D 000E R          mov cx,vec2[di]
main.asm(62): warning A4031: Operand types must match
      70
      71                ; Basing and Indexing Addressing
      72 0031  BB 0003                    mov bx,3
      73 0034  8A 81 0016 R          mov al,matr[bx][di]
      74 0038  8B 89 0016 R          mov cx,matr[bx][di]
main.asm(67): warning A4031: Operand types must match
      75 003C  8B 85 0022 R          mov ax,matr[bx*4][di]
main.asm(68): error A2055: Illegal register value
      76
      77                ; VERIFICATION OF ADDRESSING MODES TAKI
                              NG INTO ACCOUNT SEGMENTS
      78                ; Segment redefinition
      79                ; ------ variant 1
      80 0040  B8 ---- R             mov ax, SEG vec2
      81 0043  8E C0                 mov es, ax
      82 0045  26: 8B 07             mov ax, es:[bx]
      83 0048  B8 0000                    mov ax, 0
      84
      85                ; ------ variant 2
      86 004B  8E C0                 mov es, ax
      87 004D  1E                    push ds
      88 004E  07                    pop es
      89 004F  26: 8B 4F FF          mov cx, es:[bx-1]
      90 0053  91                    xchg cx,ax
      91
      92                ; ------ variant 3
      93 0054  BF 0002                    mov di,ind
      94 0057  26: 89 01             mov es:[bx+di],ax
      95
      96                ; ------ variant 4
      97 005A  8B EC                 mov bp,sp
      98 005C  3E: 8B 86 0016 R           mov ax,matr[bp+bx]
main.asm(91): error A2046: Multiple base registers
      99 0061  3E: 8B 83 0016 R           mov ax,matr[bp+di+si]
main.asm(92): error A2047: Multiple index registers
```

```
100
101                ; Using a stack segment
102 0066  FF 36 0000 R          push mem1
103 006A  FF 36 0002 R          push mem2
104 006E  8B EC                 mov bp,sp
```

```
     105 0070  8B 56 02                  mov dx,[bp]+2
     106 0073  CA 0002                   ret 2
     107 0076                    Main ENDP
main.asm(100): error A2006: Phase error between passes
     108 0076                    CODE ENDS
     109              END Main
```

```
Microsoft (R) Macro Assembler Version 5.10                  10/5/21
18:13:12
                                                             Symbols-1


Segments and Groups:

                N a m e         Length      Align     Combine Class

ASTACK . . . . . .              0018  PARA  STACK
CODE . . . . . .                0076  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 = 0076
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  main
@VERSION . . . . . .            TEXT  510


    102 Source  Lines
    102 Total   Lines
     19 Symbols

  47288 + 459972 Bytes symbol space free

      2 Warning Errors
      5 Severe  Errors


Файл mainedit.lst
```

```
                        ; IntelX86 Processor Addressing Modes S
                             tudy Program

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

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

0018                    AStack ENDS

                        ; Program data
0000                    DATA SEGMENT

                        ; Data description directives
0000  0000              mem1 DW 0
0002  0000              mem2 DW 0
0004  0000              mem3 DW 0
0006  26 25 24 23 1F 20 vec1 DB 38,37,36,35,31,32,33,34
      21 22
000E  46 50 BA B0 32 3C vec2 DB 70,80,-70,-80,50,60,-50,-60
      CE C4
0016  FE FF 05 06 F8 F9 matr DB -2,-1,5,6,-8,-7,3,4,-4,-3,7,8,-
                           6,-5,1,2
      03 04 FC FD 07 08
      FA FB 01 02
0026                    DATA ENDS


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


                        ; Head procedure
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

                        ; CHECKING THE ADDRESSING MODES AT THE
                             OFFSET LEVEL
                        ; Register addressing
0009  B8 01F4                       mov ax,n1
000C  8B C8                 mov cx,ax
000E  B3 24                 mov bl,EOL
0010  B7 CE                 mov bh,n2
```

```
                        ; Direct addressing
0012  C7 06 0002 R FFCE     mov mem2,n2
0018  BB 0006 R             mov bx,OFFSET vec1
001B  A3 0000 R             mov mem1,ax

                        ; Indirect addressing
001E  8A 07                 mov al,[bx]
                        ;    mov mem3,[bx]

                        ; Based addressing
0020  8A 47 03                  mov al,[bx]+3
0023  8B 4F 03                  mov cx,3[bx]

                        ; Indexed addressing
0026  BF 0002                   mov di,ind
0029  8A 85 000E R          mov al,vec2[di]
                        ;    mov cx,vec2[di]

                        ; Basing and Indexing Addressing
002D  BB 0003                   mov bx,3
0030  8A 81 0016 R          mov al,matr[bx][di]
                        ;    mov cx,matr[bx][di]
                        ;    mov ax,matr[bx*4][di]

                        ; VERIFICATION OF ADDRESSING MODES TAKI
                             NG INTO ACCOUNT SEGMENTS
                        ; Segment redefinition
                        ; ------ variant 1
0034  B8 ---- R             mov ax, SEG vec2
0037  8E C0                 mov es, ax
0039  26: 8B 07             mov ax, es:[bx]
003C  B8 0000                    mov ax, 0

                        ; ------ variant 2
003F  8E C0                 mov es, ax
0041  1E                    push ds
0042  07                    pop es
0043  26: 8B 4F FF          mov cx, es:[bx-1]
0047  91                    xchg cx,ax

                        ; ------ variant 3
0048  BF 0002                   mov di,ind
004B  26: 89 01             mov es:[bx+di],ax

                        ; ------ variant 4
004E  8B EC                 mov bp,sp
                        ;    mov ax,matr[bp+bx]
                        ;    mov ax,matr[bp+di+si]

                        ; Using a stack segment
0050  FF 36 0000 R          push mem1
0054  FF 36 0002 R          push mem2
0058  8B EC                 mov bp,sp
```

```
105 005A  8B 56 02                  mov dx,[bp]+2
106 005D  CA 0002                   ret 2
107 0060                    Main ENDP
108 0060                    CODE ENDS
109                 END Main
```

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0018	PARA	STACK
CODE	0060	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 = 0060
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	mainedit	
@VERSION	TEXT	510	

```
    102 Source  Lines
    102 Total   Lines
     19 Symbols

  47252 + 457961 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```