

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.
Вариант 2.

Студентка гр. 0382

Охотникова Г.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Целью данной работы является изучение режимов адресации и формирование исполнительного адреса.

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.

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.

Программа с подставленными в нее данными из варианта 2 была протранслирована, создан файл диагностических сообщений.

Ошибки:

1. `mov mem3,[bx]: error A2052: Improper operand type`

Ошибка заключается в том, что перемещение из памяти в память недопустимо. С помощью одной команды MOV нельзя напрямую переслать операнд из одной области памяти в другую. Поэтому вначале нужно загрузить исходное значение в один из регистров общего назначения, а затем переслать его в нужное место памяти.

2. `mov cx,vec2[di]: warning A4031: Operand types must match`

Ошибка заключается в том, что размер регистра cx равен два байта, а размер элементов vec2 1 байт. Оба операнда должны иметь одинаковую длину.

3. `mov cx,matr[bx][di]: warning A4031: Operand types must match`

Ошибка заключается в том, что размер регистра cx равен два байта, а размер элементов matr 1 байт. Оба операнда должны иметь одинаковую длину.

4. `mov ax,matr[bx*4][di]: error A2055: Illegal register value`

Ошибка заключается в том, что умножение двухбайтовых регистров недопустимо. Следует сначала записать в регистр bx значение и только потом обращаться к нему по индексу.

5. `mov ax,matr[bp+bx]: error A2046: Multiple base registers`

Ошибка заключается в том, что для адресации нельзя использовать более одного базового регистра. Следует сначала записать в регистр bx значение и только потом обращаться к нему по индексу.

6. `mov ax,matr[bp+di+si]: error A2047: Multiple index registers`

Ошибка заключается в том, что для адресации нельзя использовать больше одного индексного регистра.

После того, как ошибки были закомментированы, программа была протранслирована снова и выполнена в пошаговом режиме под управлением отладчика.

Результаты выполнения данной программы представлены в таблице 1.

Таблица 1.

Начальные значения сегментных регистров: CS =1A05, DS =19F5, ES =19F5, SS =1A09.

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			До выполнения	После выполнения
0000	PUSH DS	1E	IP = 0000 DS = 19F5 SP = 0018 STACK +0 0000	IP = 0001 DS = 19F5 SP = 0016 STACK +0 19F5
0001	SUB AX, AX	2BC0	AX = 0000 IP = 0001	AX = 0000 IP = 0003
0003	PUSH AX	50	IP = 0003 AX = 0000 SP = 0016 STACK +0 19F5 +2 0000	IP = 0004 AX = 0000 SP = 0014 STACK +0 0000 +2 19F5
0004	MOV AX, 1A07	B8071A	AX = 0000 IP = 0004	AX = 1A07 IP = 0007
0007	MOV DS, AX	8ED8	DS = 19F5 IP = 0007	DS = 1A07 IP = 0009
0009	MOV AX, 01F4	B8F401	AX = 1A07 IP = 0009	AX = 01F4 IP = 000C
000C	MOV CX, AX	8BC8	IP = 000C CX = 00B0	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	C7060200CE FF	IP = 0012 DS = 1A07 00 00 00 00 00	IP = 0018 DS = 1A07 00 00 CE FF 00
0018	MOV BX, 0006	BB0600	BX = CE24 IP = 0018	BX = 0006 IP = 001B
001B	MOV [0000], AX	A30000	IP = 001B AX = 01F4 DS = 1A07 00 00 CE FF 00	IP = 001E AX = 01F4 DS = 1A07 F4 01 CE FF 00
001E	MOV AL, [BX]	8A07	AX = 01F4 IP = 001E	AX = 0105 IP = 0020
0020	MOV AL, [BX+03]	8A4703	IP = 0020 AX = 0105	IP = 0023 AX = 0108
0023	MOV CX, [BX+03]	8B4F03	CX = 01F4 IP = 0023	CX = 0c08 IP = 0026
0026	MOV DI, 0002	BF0200	IP = 0026 DI = 0000	IP = 0029 DI = 0002
0029	MOV AL, [000E+DI]	8A850E00	IP = 0029 AX = 0108	IP = 002D AX = 0114
002D	MOV BX, 0003	BB0300	IP = 002D BX = 0006	IP = 0030 BX = 0003
0030	MOV AL, [0016+BX+DI]	8A811600	IP = 0030 AX = 0114	IP = 0034 AX = 0103
0034	MOV AX, 1A07	D8071A	IP = 0034 AX = 0103	IP = 0037 AX = 1A07
0037	MOV ES, AX	8ECO	IP = 0037 ES = 19F5	IP = 0039 ES = 1A07

0039	MOV AX, ES:[BX]	268B07	IP = 0039 AX = 1A07	IP = 003C AX = 00FF
003C	MOV AX, 0000	B80000	IP = 003C AX = 00FF	IP = 003F AX = 0000
003F	MOV ES, AX	8EC0	IP = 003F ES = 1A07	IP = 0041 ES = 0000
0041	PUSH DS	1E	IP = 0041 DS = 1A07 SP = 0014 STACK +0 0000 +2 19F5 +4 0000	IP = 0042 DS = 1A07 SP = 0012 STACK +0 1A07 +2 0000 +4 19F5
0042	POP ES	07	IP = 0042 ES = 0000 SP = 0012 STACK +0 1A07 +2 0000 +4 19F5	IP = 0043 ES = 1A07 SP = 0014 STACK +0 0000 +2 19F5 +4 0000
0043	MOV CX, ES:[BX-01]	268B4FFF	IP = 0043 CX = 0C08	IP = 0047 CX = FFCE
0047	XCHG AX, CX	91	IP = 0047 AX = 0000 CX = FFCE	IP = 0048 AX = FFCE CX = 0000
0048	MOV DI, 0002	BF0200	IP = 0048 DI = 0002	IP = 004B DI = 0002
004B	MOV ES:[BX+DI],AX	268901	IP = 004B ES = 1A07 DS = 1A07	IP = 004E ES = 1A07 DS = 1A07

			F4 01 CE FF 00 00 0C	F4 01 CE FF 00 CE FF
004E	MOV BP, SP	8BEC	IP = 004E BP = 0000	IP = 0050 BP = 0014
0050	PUSH [0000]	FF360000	IP = 0050 SP = 0014 STACK +0 0000 +2 19F5 +4 0000	IP = 0054 SP = 0012 STACK +0 01F4 +2 0000 +4 19F5
0054	PUSH [0002]	FF360200	IP = 0054 SP = 0012 STACK +0 01F4 +2 0000 +4 19F5 +6 0000	IP = 0058 SP = 0010 STACK +0 FFCE +2 01F4 +4 0000 +6 19F5
0058	MOV BP, SP	8BEC	IP = 0058 BP = 0014	IP = 005A BP = 0010
005A	MOV DX, [BP+02]	8B5602	IP = 005A DX = 0000	IP = 005D DX = 01F4
005D	RET FAR 0002	CA0200	IP = 005D CS = 1A0A SP = 0010 STACK +0 FFCE +2 01F4 +4 0000 +6 19F5	IP = FFCE CS = 01F4 SP = 0016 STACK +0 19F5 +2 0000 +4 0000 +6 0000

Выводы.

При выполнении данной лабораторной работы были изучены принципы режимов адресации и формирования исполнительного адреса.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: pr1.asm

```
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 5,6,7,8,12,11,10,9
vec2 DB -20,-30,20,30,-40,-50,40,50
matr DB -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-4,8,7,6,5
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]
;   Индексная 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 РЕЖИМОВ 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  2
Main      ENDP
CODE      ENDS
END Main

```

Название файла: pr2.asm

```

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 5,6,7,8,12,11,10,9
vec2 DB -20,-30,20,30,-40,-50,40,50
matr DB -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-4,8,7,6,5
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]

; Индексная 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 РЕЖИМОВ 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 2

Main ENDP
CODE ENDS
        END Main

```

Название файла: pr1.lst

__Microsoft (R) Macro Assembler Version 5.10

10/5/21

16:47:13

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
```

< C...

```
0000 0000          mem1 DW 0
0002 0000          mem2 DW 0
0004 0000          mem3 DW 0
0006 05 06 07 08 0C 0B  vec1 DB 5,6,7,8,12,11,10,9
          0A 09
000E EC E2 14 1E D8 CE  vec2 DB -20,-30,20,30,-40,-50,40,50
          28 32
0016 FB FA F9 F8 04 03  matr  DB  -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-
4,8,7,6,5
          02 01 FF FE FD FC
          08 07 06 05
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

```

__Microsoft (R) Macro Assembler Version 5.10

10/5/21

16:47:13

Page

1-2

```

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

```

pr1.asm(49): error A2052: Improper operand type

```

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

```

```

0026  BF 0002                      mov di,ind
0029  8A 85 000E R                  mov al,vec2[di]
002D  8B 8D 000E R                  mov cx,vec2[di]
pr1.asm(58): 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]
pr1.asm(63): warning A4031: Operand types must match
003C  8B 85 0022 R                  mov ax,matr[bx*4][di]
pr1.asm(64): error A2055: Illegal register value

0040  B8 ---- R                    mov ax, SEG vec2
0043  8E C0                          mov es, ax
0045  26: 8B 07                      mov ax, es:[bx]
0048  B8 0000                      mov ax, 0

004B  8E C0                          mov es, ax
004D  1E                          push ds
004E  07                          pop es
004F  26: 8B 4F FF                  mov cx, es:[bx-1]
0053  91                          xchg cx,ax

0054  BF 0002                      mov di,ind
0057  26: 89 01                  mov es:[bx+di],ax

005A  8B EC                      mov bp,sp
005C  3E: 8B 86 0016 R              mov ax,matr[bp+bx]
pr1.asm(87): error A2046: Multiple base registers
0061  3E: 8B 83 0016 R              mov ax,matr[bp+di+si]
pr1.asm(88): error A2047: Multiple index registers

0066  FF 36 0000 R                push mem1

```



```

006A  FF 36 0002 R          push mem2
006E  8B EC                mov bp,sp
0070  8B 56 02             mov dx,[bp]+2
0073  CA 0002              ret 2

```

__Microsoft (R) Macro Assembler Version 5.10 10/5/21

16:47:13

Page

1-3

```

0076                      Main ENDP
pr1.asm(96): error A2006: Phase error between passes
0076                      CODE ENDS
                      END Main

```

__Microsoft (R) Macro Assembler Version 5.10 10/5/21

16:47:13

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 =
MATR	L BYTE	0016	DATA

0076

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	pr1
@VERSION	TEXT	510

98 Source Lines

98 Total Lines

19 Symbols

47842 + 459418 Bytes symbol space free

2 Warning Errors

5 Severe Errors

Название файла: pr2.lst

Microsoft (R) Macro Assembler Version 5.10

10/5/21

19:09:03

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 05 06 07 08 0C 0B    vec1 DB 5,6,7,8,12,11,10,9
      0A 09
000E EC E2 14 1E D8 CE    vec2 DB -20,-30,20,30,-40,-50,40,50
      28 32
0016 FB FA F9 F8 04 03    matr  DB  -5,-6,-7,-8,4,3,2,1,-1,-2,-3,-
4,8,7,6,5
      02 01 FF FE FD FC
      08 07 06 05

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

__Microsoft (R) Macro Assembler Version 5.10

10/5/21

19:09:03

Page

1-2

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]

0026 BF 0002 mov di,ind

0029 8A 85 000E R mov al,vec2[di]

;mov cx,vec2[di]

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]

; ----- PIP°CTPëP°PSC, 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

```

; ----- PIP°CтРёР°PSC, 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

```

```

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

```

```

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

```

```

0050 FF 36 0000 R        push mem1
0054 FF 36 0002 R        push mem2
0058 8B EC              mov bp, sp
005A 8B 56 02           mov dx, [bp]+2
005D CA 0002            ret 2

```

__Microsoft (R) Macro Assembler Version 5.10

10/5/21

19:09:03

Page

1-3

```

0060                Main ENDP
0060                CODE ENDS
                END Main

```

__Microsoft (R) Macro Assembler Version 5.10

10/5/21

19:09:03

Symbols-1

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	pr2	
@VERSION	TEXT	510	

98 Source Lines

98 Total Lines

19 Symbols

47842 + 459418 Bytes symbol space free

0 Warning Errors

0 Severe Errors