

**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»**  
**ТЕМА: ИЗУЧЕНИЕ РЕЖИМОВ АДРЕСАЦИИ И**  
**ФОРМИРОВАНИЯ ИСПОЛНИТЕЛЬНОГО АДРЕСА**

Студентка гр. 0382

Андрющенко К.С

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

### **Цель работы.**

Изучение работы режимов адресации, используя программу на языке Ассемблера, выполнение которой производится под управлением отладчика в пошаговом режиме.

### **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.

### Вариант №2:

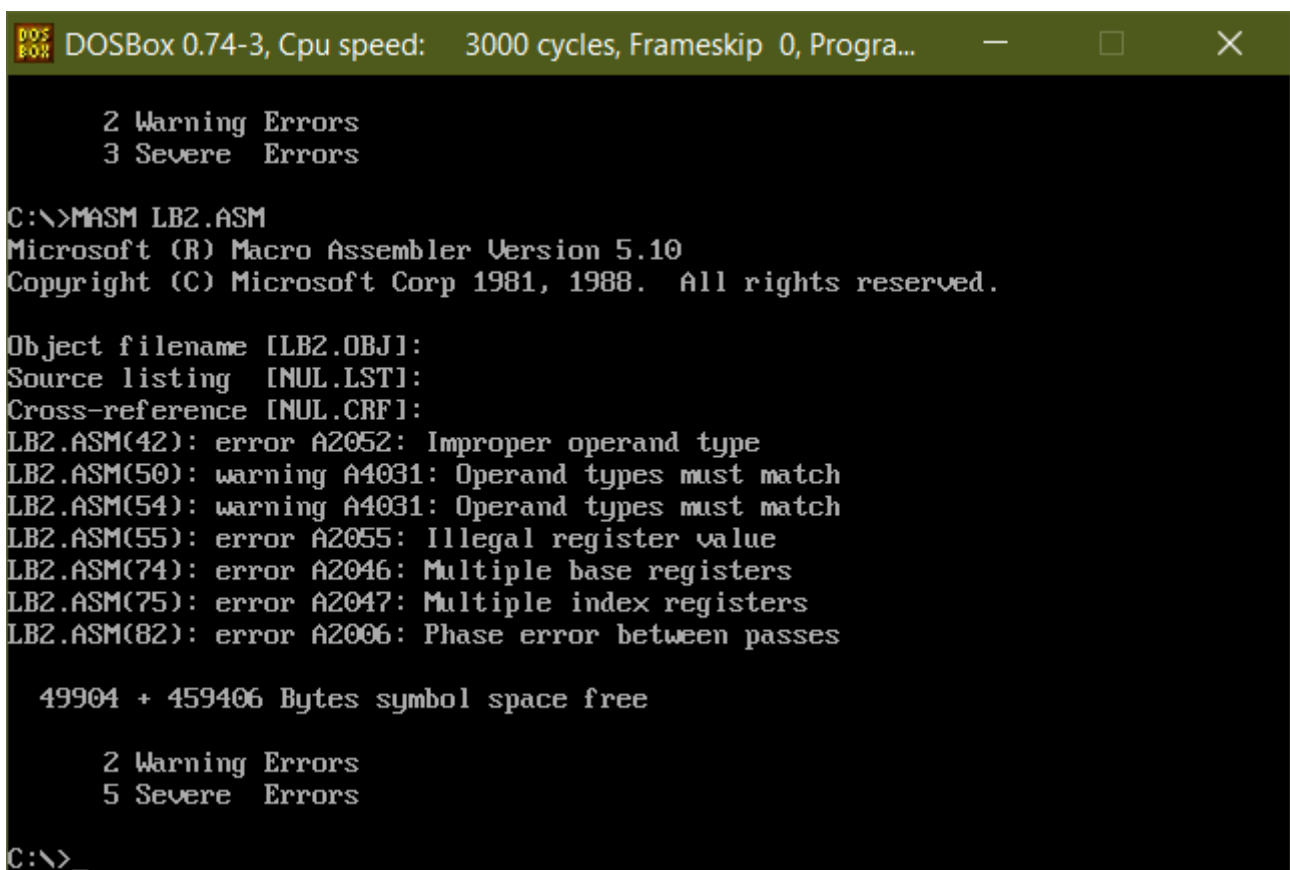
vec1 5,6,7,8,12,11,10,9

vec2 -20,-30,20,30,-40,-50,40,50

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

### Выполнение работы.

При первоначальной трансляции были обнаружены следующие ошибки см. Рисунок 1.



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

2 Warning Errors
3 Severe Errors

C:\>MASM LB2.ASM
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [LB2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
LB2.ASM(42): error A2052: Improper operand type
LB2.ASM(50): warning A4031: Operand types must match
LB2.ASM(54): warning A4031: Operand types must match
LB2.ASM(55): error A2055: Illegal register value
LB2.ASM(74): error A2046: Multiple base registers
LB2.ASM(75): error A2047: Multiple index registers
LB2.ASM(82): error A2006: Phase error between passes

49904 + 459406 Bytes symbol space free

2 Warning Errors
5 Severe Errors

C:\>_
```

Рисунок 1 – Ошибки при трансляции

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Libraries [.LIB]:
C:\>AFDPRO LB2.EXE
AFD-Pro is done
C:\>MASM LB2.ASM
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988. All rights reserved.

Object filename [LB2.OBJ]:
Source listing [NUL.LST]: LB2_ERROR.LST
Cross-reference [NUL.CRF]:
LB2.ASM(50): warning A4031: Operand types must match
LB2.ASM(54): warning A4031: Operand types must match
LB2.ASM(55): error A2055: Illegal register value
LB2.ASM(74): error A2046: Multiple base registers
LB2.ASM(75): error A2047: Multiple index registers

47830 + 459430 Bytes symbol space free

2 Warning Errors
3 Severe Errors
C:\>
```

Описание ошибок, обнаруженных при первоначальной трансляции:

1. `mov mem3,[bx]` – `lb2.asm(42): error A2052: Improper operand type`  
(неправильный тип операнда);  
Одновременное чтение и запись из памяти. Поскольку прямой операции память-память не существует, данная операция выполняется через регистр.
2. `mov cx,vec2[di]` - `lb2.asm(50): warning A4031: Operand types must match`  
(Предупреждение: типы операндов должны совпадать);  
Размер регистра CX – 16 бит, а элемент массива `vec2` имеет размер в байт (8 бит).
3. `mov cx,matr[bx][di]` - `lb2.asm(54): warning A4031: Operand types must match`  
(Предупреждение: типы операндов должны совпадать);  
Размер регистра CX – 16 бит, а элемент матрицы `matr` имеет размер в байт (8 бит).
4. `mov ax,matr[bx*4][di]` - `lb2.asm(55): error A2055: Illegal register value`  
(Недопустимое значение регистра);

Попытка умножить регистр на какое-либо число - недопустимая операция (исключение: определение масштабного индексного операнда для процессора 80386).

5. `mov ax,matr[bp+bx]` - `lb2.asm(74): error A2046: Multiple base registers` (Использование нескольких базовых регистров);

Недопустимое использование более одного базового регистра для адресации.

6. `mov ax,matr[bp+di+si]` - `lb2.asm(75): error A2047: Multiple index registers` (Использование нескольких индексных регистров);

Недопустимое использование более одного индексного регистра.

### **Вывод.**

В результате работы была изучена работа режимов адресации с использованием программы на языке Ассемблера.

## ПРОТОКОЛ

Таблица 1 - Результат выполнения программы в пошаговом режиме

Адрес команды	Символический код команды	16-ричный код команды	Содержимое регистров и ячеек памяти	
			до выполнения	После выполнения
1A0A	PUSH DS	1E	(SP) = 0018 (DS) = 19F5 (CS) = 1A0A (IP) = 0000 (SP) = 0018 Stack: +0 0000	(SP) = 0016 (DS) = 19F5 (CS) = 1A0A (IP) = 0001 (SP) = 0016 Stack: +0 19F5
1A0B	SUB AX, AX	2BC0	(AX) = 0000 (SP) = 0016 (IP) = 0001	(AX) = 0000 (SP) = 0016 (IP) = 0003
1A0D	PUSH AX	50	(SP) = 0016 (AX) = 0000 (IP) = 0003 Stack: +0 19F5	(SP) = 0014 (AX) = 0000 (IP) = 0004 Stack: +0 0000 +2 19F5
1A0E	MOV AX, 1A07	B8071A	(AX) = 0000 (IP) = 0004	(AX) = 1A07 (IP) = 0007
1A11	MOV DS, AX	8ED8	(DS) = 19F5 (IP) = 0007	(DS) = 1A07 (IP) = 0009
1A13	MOV AX, 01F4	B8F401	(AX) = 1A07 (IP) = 0009	(AX) = 01F4 (IP) = 000C
1A16	MOV CX, AX	8BC8	(CX) = 00B0 (IP) = 000C	(CX) = 01F4 (IP) = 000E

Продолжение таблицы 1				
1A18	MOV BL, 24	B324	(BX) = 0000 (IP) = 000E	(BX) = 0024 (IP) = 0010
1A1A	MOV BH, CE	B7CE	(BX) = 0024 (IP) = 0010	(BX) = CE24 (IP) = 0012
1A1C	MOV [0002], FFCE	C7060200CE FF	(IP) = 0012	(IP) = 0018
1A22	MOV BX, 0006	BB0600	(BX) = CE24 (IP) = 0018	(BX) = 0006 (IP) = 001B
1A25	MOV [0000], AX	A30000	(IP) = 001B	(IP) = 001E
1A28	MOV AL, [BX]	8A07	(AX) = 01F4 (IP) = 001E	(AX) = 0101 (IP) = 0020
1A2A	MOV AL, [BX+03]	8A4703	(AX) = 0101 (IP) = 0020	(AX) = 0104 (IP) = 0023
1A2D	MOV CX, [BX+03]	8B4F03	(CX) = 01F4 (IP) = 0023	(CX) = 0804 (IP) = 0026
1A30	MOV DI, 0002	BF0200	(DI) = 0000 (IP) = 0026	(DI) = 0002 (IP) = 0029
1A33	MOV AL, [000E+DI]	8A850E00	(AX) = 0104 (IP) = 0029	(AX) = 010A (IP) = 002D
1A37	MOV BX, 0003	BB0300	(BX) = 0006 (IP) = 002D	(BX) = 0003 (IP) = 0030
1A3A	MOV AL, [0016+BX+DI]	8A811600	(AX) = 010A (IP) = 0030	(AX) = 01FD (IP) = 0034
1A3E	MOV AX, 1A07	B8071A	(AX) = 01FD (IP) = 0034	(AX) = 1A07 (IP) = 0037
1A41	MOV ES, AX	8EC0	(ES) = 19F5 (IP) = 0037	(ES) = 1A07 (IP) = 0039
1A43	MOV AX, ES:[BX]	268B07	(AX) = 1A07	(AX) = 00FF

			(IP) = 0039	(IP) = 003C
1A46	MOV AX, 0000	B80000	(AX) = 00FF (IP) = 003C	(AX) = 0000 (IP) = 003F
1A49	MOV ES, AX	8EC0	(ES) = 1A07 (IP) = 003F	(ES) = 0000 (IP) = 0041
1A4B	PUSH DS	1E	(DS) = 1A07 (SP) = 0014 (IP) = 0041 Stack: +0 0000 +2 19F5	(DS) = 1A07 (SP) = 0012 (IP) = 0042 Stack: +0 1A07 +2 0000 +4 19F5
1A4C	POP ES	07	(ES) = 0000 (SP) = 0012 (IP) = 0042 Stack: +0 1A07 +2 0000 +4 19F5	(ES) = 1A07 (SP) = 0014 (IP) = 0043 Stack: +0 0000 +2 19F5
1A4D	MOV CX, ES:[BX-01]	268B4FFF	(CX) = 0804 (IP) = 0043	(CX) = FFCE (IP) = 0047
1A51	XCHG AX, CX	91	(AX) = 0000 (CX) = FFCE (IP) = 0047	(AX) = FFCE (CX) = 0000 (IP) = 0048
1A52	MOV DI, 0002	BF0200	(DI) = 0002 (IP) = 0048	(DI) = 0002 (IP) = 004B
1A55	MOV ES:[BX+DI], AX	268901	(IP) = 004B DS:[5]00 [6]01	(IP) = 004E DS:[5]CE [6]FF
1A58	MOV BP, SP	8BEC	(BP) = 0000 (IP) = 004E	(BP) = 0014 (IP) = 0050
1A5A	PUSH 01F4	FF360000	(SP) = 0014	(SP) = 0012



			(IP) = 0050 Stack: +0 0000 +2 19F5 +4 0000	(IP) = 0054 Stack: +0 01F4 +2 0000 +4 19F5
1A5E	PUSH FFCE	FF360200	(SP) = 0012 (IP) = 0054 Stack: +0 01F4 +2 0000 +4 19F5 +6 0000	(SP) = 0010 (IP) = 0058 Stack: +0 FFCE +2 01F4 +4 0000 +6 19F5
1A62	MOV BP, SP	8BEC	(BP) = 0014 (IP) = 0058	(BP) = 0010 (IP) = 005A
1A64	MOV DX, [BP+02]	8B5602	(DX) = 0000 (IP) = 005A	(DX) = 01F4 (IP) = 005D
1A67	RET far 0002	CA0200	(CS) = 1A0A (SP) = 0010 (IP) = 005D Stack: +0 FFCE +2 01F4 +4 0000 +6 19F5	(CS) = 01F4 (SP) = 0016 (IP) = Stack: +0 19F5 +2 0000 +4 0000 +6 0000

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММ

#### Файл LB2\_ERROR.ASM

```
; Программа изучения режимов адресации процессора IntelX86
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 ;определяет переменную размером в слово.
mem2 DW 0
mem3 DW 0
vec1 DB 5,6,7,8,12,11,10,9;определяет данные размером в байт
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

```

## Файл LB2\_READY.ASM

```
; Программа изучения режимов адресации процессора IntelX86
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 ;определяет переменную размером в слово.
mem2 DW 0
mem3 DW 0
vec1 DB 5,6,7,8,12,11,10,9 ;определяет данные размером в байт
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

```

# ПРИЛОЖЕНИЕ В

## ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ

### Файл LB2\_ERROR.LST

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:32:23

Page

1-1

```

; Программа изучения режи?
?ов адресации процессора I
ntelX86
= 0024          EOL EQU '$'
= 0002          ind EQU 2
= 01F4          n1 EQU 500
=-0032          n2 EQU -50
; Стек программы
AStack SEGMENT STACK
0000            DW 12 DUP(?)
0000 000C[
        ????
]

0018            AStack ENDS
; Данные программы
0000            DATA SEGMENT
; Директивы описания данн?
?х
0000 0000            mem1 DW 0 ;определяет перемен?
?ую размером в слово.
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 ;определяе?
? данные разме-ром в байт
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
; 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 АДРЕСА?
?ИИ НА УРОВНЕ СМЕЩЕНИЙ
; Регистровая адресация
0009 B8 01F4            mov ax,n1

```

```

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]

```

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:32:23

Page

1-2

```

                ; mov mem3,[bx]
                ; Базированная адресация
                ;
0020 8A 47 03          mov al,[bx]+3
0023 8B 4F 03          mov cx,3[bx]
                ; Индексная адресация
0026 BF 0002          mov di,ind
0029 8A 85 000E R      mov al,vec2[di]
002D 8B 8D 000E R      mov cx,vec2[di]
LB2.ASM(50): warning A4031: Operand types must match
                ; Адресация с базирование?
                ? и индексированием
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]
LB2.ASM(54): warning A4031: Operand types must match
003C 8B 85 0022 R      mov ax,matr[bx*4][di]
LB2.ASM(55): error A2055: Illegal register value
                ; ПРОВЕРКА РЕЖИМОВ АДРЕСА?
                ?ИИ С УЧЕТОМ СЕГМЕНТОВ
                ; Переопределение сегмент
                a
                ; ----- вариант 1
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
                ; ----- вариант 2
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
                ; ----- вариант 3
0054 BF 0002          mov di,ind
0057 26: 89 01        mov es:[bx+di],ax
                ; ----- вариант 4
005A 8B EC            mov bp,sp
005C 3E: 8B 86 0016 R  mov ax,matr[bp+bx]
LB2.ASM(74): error A2046: Multiple base registers
0061 3E: 8B 83 0016 R  mov ax,matr[bp+di+si]
LB2.ASM(75): error A2047: Multiple index registers
                ; Использование сегмента ?

```

```

                                ?teka
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
0076                                Main ENDP
0076                                CODE ENDS
                                END Main

```

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:32:23

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	
0076 MAIN	. . . . .	F PROC	0000	CODE Length =
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	LB2	
@VERSION	. . . . .	TEXT	510	
	84 Source Lines			
	84 Total Lines			
	19 Symbols			

47830 + 459430 Bytes symbol space free

2 Warning Errors  
3 Severe Errors



## Файл LB2\_READY.LST

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:43:43

Page

1-1

```

; Программа изучения режи?
?ов адресации процессора I
ntelX86
= 0024          EOL EQU '$'
= 0002          ind EQU 2
= 01F4          n1 EQU 500
=-0032          n2 EQU -50
; Стек программы
AStack SEGMENT STACK
0000 000C[      DW 12 DUP(?)
      ????)
]

0018          AStack ENDS
; Данные программы
0000          DATA SEGMENT
; Директивы описания данн?
?х
0000 0000          mem1 DW 0 ;определяет перемен?
?ую размером в слово.
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 ;определяе?
? данные разме-ром в байт
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
; 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 АДРЕСА?
?ИИ НА УРОВНЕ СМЕЩЕНИЙ
; Регистровая адресация
0009 B8 01F4          mov ax,n1
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]

```

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:43:43

Page

1-2

```

; mov mem3,[bx]
; Базированная адресация
;
0020 8A 47 03 mov al,[bx]+3
0023 8B 4F 03 mov cx,3[bx]
; Индексная адресация
0026 BF 0002 mov di,ind
0029 8A 85 000E R mov al,vec2[di]
; mov cx,vec2[di]
; Адресация с базирование?
? и индексированием
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]
; ПРОВЕРКА РЕЖИМОВ АДРЕСА?
? ИИ С УЧЕТОМ СЕГМЕНТОВ
; Переопределение сегмент
a
; ----- вариант 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
; ----- вариант 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
; ----- вариант 3
0048 BF 0002 mov di,ind
004B 26: 89 01 mov es:[bx+di],ax
; ----- вариант 4
004E 8B EC mov bp,sp
; mov ax,matr[bp+bx]
; mov ax,matr[bp+di+si]
; Использование сегмента ?
? тека
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
0060 Main ENDP

```

0060

CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10

10/3/21 23:43:43

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	
0060	MAIN	F PROC	0000	CODE Length =
	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	LB2	
	@VERSION	TEXT	510	

84 Source Lines

84 Total Lines

19 Symbols

47830 + 459430 Bytes symbol space free

0 Warning Errors

0 Severe Errors