**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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

## Цель работы.

## Ознакомиться с режимом работы адресации основной памяти в языке Ассемблер.

## 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.

Пример используемой программы приведен ниже.

; 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

mem2 DW 0

mem3 DW 0

vec1 DB 1,2,3,4,8,7,6,5

vec2 DB -10,-20,10,20,-30,-40,30,40

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

## Ход работы.

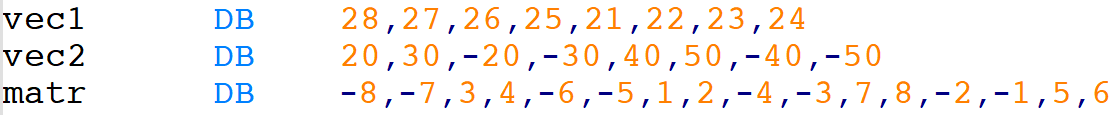
1. Из файла lr2.dat взят вариант значений исходных данных (массивов) vec1, vec2 и matr и занесены вместо значений, указанных в приведённой для образца программе.

Вариант 8

vec 1 - 28,27,26,25,21,22,23,24

vec 2 - 20,30,-20,-30,40,50,-40,-50

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



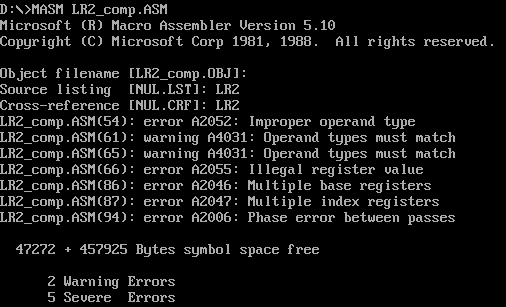
1. Смонтирован каталог с MASM в эмуляторе DOSBox с помощью команды mount <диск в эмуляторе> <реальный путь к каталогу с MASM>. В данной лабораторной работе смонтируем mount каталог C:\lr2\MASM как диск d.



1. Осуществлен переход на только что смонтированный диск, подав команду в эмуляторе d:. Содержимым диска d: в эмуляторе будет содержимое каталога C:\MASM\_EXE реального компьютера, то есть можно непосредственно запускать ассемблер, компоновщик и отладчик для работы с заданной программой.



1. Программа была протранслирована с помощью строки MASM LR2\_comp.ASM c созданием файла диагностических сообщений – файла листинга, LR2.lst.



Пояснение ошибок:

* LR2\_comp.ASM(54): error A2052: Improper operand type



Ошибка: Неправильный тип операнда.

Пояснение: Машинные команды не могут манипулировать одновременно двумя операндами, находящимися в оперативной памяти, то есть в команде только 1 операнд может указывать на ячейку памяти, другой операнд должен быть либо регистром, либо непосредственным значением.

* LR2\_comp.ASM(61): warning A4031: Operand types must match



Ошибка: Типы операндов должны соответствовать друг другу.

Пояснение: Различный размер операндов - размер данного регистра составляет 2 байта, а размер элемента в массиве составляет 1 байт.

* LR2\_comp.ASM(65): warning A4031: Operand types must match



Ошибка: Типы операндов должны соответствовать друг другу.

Пояснение: Различный размер операндов - размер данного регистра составляет 2 байта, а размер элемента в массиве составляет 1 байт.

* LR2\_comp.ASM(66): error A2055: Illegal register value



Ошибка: Запрещенное значение регистра.

Пояснение: В непосредственной адресации с базированием и индексированием для вычисления исполнительного адреса берется сумма базового и индексного регистра, к которым добавляется непосредственно фигурирующее в команде

смещение. Там не применяется умножение.

* LR2\_comp.ASM(86): error A2046: Multiple base registers



Ошибка: Несколько базовых регистров.

Пояснение: В базированной адресации необходимо указывать базовый регистр, затем производить смещение с помощью индексного. В данной случае оба регистра являются базовыми.

* LR2\_comp.ASM(87): error A2047: Multiple index registers



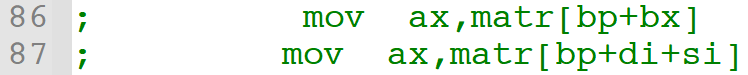
Ошибка: Несколько индексных регистров.

Пояснение: В базированной адресации необходимо указывать базовый регистр, затем производить смещение с помощью индексного. В данном случае один из регистров – базовый, два других – индексные.

Для дальнейшей корректной работы программы соответствующие операторы, из-за которых возникали ошибки, были закомментированы в тексте программы.



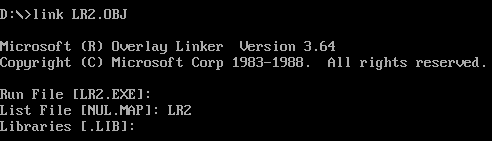




1. Программа была снова протранслирована с учетом закомментированных команд с помощью строки MASM LR2\_comp.ASM.



Скомпонован загрузочный модуль, LR2.EXE, с помощью строки link LR2.OBJ.



1. Выполнение программы было запущено под управлением отладчика с помощью команды afd LR2.EXE.

Стартовые значения регистров представлены в табл.1.

Таблица 1 – стартовые значения регистров.

|  |  |  |  |
| --- | --- | --- | --- |
| CS =11B1 | AX =0000 | SI =0000 | IP =0000 |
| DS =119C | BX =0000 | DI =0000 | HS =119C |
| SS =11AC | CX =00B6 | BP =0000 | FS =119C |
| ES =119C | DX =0000 | SP =0018 |

Выполнена программа в пошаговом режиме с фиксацией используемых регистров и ячеек памяти до и после выполнения каждой команды. Результаты представлены в табл.2

Таблица 2 – отладка LR2\_comp.asm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0000 | PUSH DS | 1E | CS = 119C  ES = 119C  SS = 119C  SP = 0018  DS = 119C  IP = 0000  Stack +0 0000  +2 0000  +4 0000  +6 0000 | CS = 119C  ES = 119C  SS = 119C  SP = 0016  DS = 119C  IP = 0001  Stack +0 119C  +2 0000  +4 0000  +6 0000 |
| 0001 | SUB AX, AX | 2BC0 | IP = 0001  AX = 0000 | IP = 0003  AX = 0000 |
| 0003 | PUSH AX | 50 | SP = 0016  IP = 0003  AX = 0000  Stack +0 119C  +2 0000  +4 0000  +6 0000 | SP = 0014  IP = 0004  AX = 0000  Stack +0 0000  +2 119C  +4 0000  +6 0000 |
| 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 | AX = 11AE  IP = 0009 | AX = 01F4  IP = 000C |
| 000C | MOV CX, AX | 8BC8 | CX = 00B6  IP = 000C | CX = 01F4  IP = 000E |
| 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 = 00 | 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 = 011C  BX = 0006  IP = 0020 |
| 0020 | MOV AL, [BX+03] | 8A4703 | AX = 011C  BX = 0006  IP = 0020 | AX = 0119  BX = 0006  IP = 0023 |
| 0023 | MOV CX, [BX+03] | 8B4F03 | CX = 01F4  BX = 0006  IP = 0023 | CX = 1519  BX = 0006  IP = 0026 |
| 0026 | MOV DI, 0002 | BF0200 | DI = 0000  IP = 0026 | DI = 0002  IP = 0029 |
| 0029 | MOV AL, [DI+000E] | 8A850E00 | AX = 0119  IP = 0029 | AX = 01EC  IP = 002D |
| 002D | MOV CX, [DI+000E] | 8B8D0E00 | CX = 1519  DI = 0002  IP = 002D | CX = E2EC  DI = 0002  IP = 0031 |
| 0031 | MOV BX, 0003 | BB0300 | BX = 0006  IP = 0031 | BX = 0003  IP = 0034 |
| 0034 | MOV AL, [BX+DI+0016] | 8A811600 | AX = 01EC  BX = 0003  DI = 0002  IP = 0034 | AX = 01FB  BX = 0003  DI = 0002  IP = 0038 |
| 0038 | MOV CX, [BX+DI+0016] | 8B891600 | CX = E2EC  BX = 0003  DI = 0002  IP = 0038 | CX = 01FB  BX = 0003  DI = 0002  IP = 003C |
| 003C | MOV AX, 11AE | B8AE11 | AX = 01FB  IP = 003C | AX = 11AE  IP = 003F |
| 003F | MOV ES, AX | 8EC0 | ES = 119C  AX = 11AE  IP = 003F | ES = 11AE  AX = 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 | ES = 11AE  AX = 0000  IP = 0047 | ES = 0000  AX = 0000  IP = 0049 |
| 0049 | PUSH DS | 1E | DS = 11AE  SP = 0014  IP = 0049  Stack +0 0000  +2 119C  +4 0000  +6 0000 | DS = 11AE  SP = 0012  IP = 004A  Stack +0 11AE  +2 0000  +4 119C  +6 0000 |
| 004A | POP ES | 07 | ES = 0000  IP = 004A  Stack +0 11AE  +2 0000  +4 119C  +6 0000 | ES = 11AE  IP = 004B  Stack +0 0000  +2 119C  +4 0000  +6 0000 |
| 004B | CX,ES:[BX-01] | 268B4FFF | CX = 01FB  BX = 0003  ES = 11AE  SP = 0012  IP = 004B | CX = FFCE  BX = 0003  ES = 11AE  SP = 0014  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  BX = 0003  DI = 0002  AX = FFCE  IP = 0053  DS:0005 = 00  DS:0006 = 1C | ES = 11AE  BX = 0003  DI = 0002  AX = FFCE  IP = 0056  DS:0005 = CE  DS:0006 = EF |
| 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 | MOV DX, [BP+02] | 8B5602 | DX = 0000  BP = 0010  IP = 0062 | DX = 01F4  BP = 0010  IP = 0065 |
| 0065 | RET Far | CB | SP = 0010  CS = 11B1  IP = 0065  Stack +0 FFCE  +2 01F4  +4 0000  +6 119C | SP = 0014  CS = 01F4  IP = FFCE  Stack +0 0000  +2 119C  +4 0000  +6 0000 |

## Вывод.

В результате работы были разобраны некоторые базовые концепции языка Ассемблера. Были изучены различные способы адресации в языке ассемблера. Была исправлена, протранслирована и собрана тестовая программа, разобраны ошибки, встречающиеся в ней, произведена пошаговая отладка с занесением результатов в таблицу.

# ПРИЛОЖЕНИЕ.

**LR2\_comp.ASM**

Программа до исправления ошибок/

; 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 28**,**27**,**26**,**25**,**21**,**22**,**23**,**24

vec2 DB 20**,**30**,-**20**,-**30**,**40**,**50**,-**40**,-**50

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

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

**LR2.lst**

Microsoft **(**R**)** Macro Assembler Version 5**.**10 10**/**6**/**22 00**:**52**:**47

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 1C 1B 1A 19 15 16 vec1 DB 28**,**27**,**26**,**25**,**21**,**22**,**23**,**24

17 18

000E 14 1E EC E2 28 32 vec2 DB 20**,**30**,-**20**,-**30**,**40**,**50**,-**40**,-**50

D8 CE

0016 F8 F9 03 04 FA FB matr DB **-**8**,-**7**,**3**,**4**,-**6**,-**5**,**1**,**2**,-**4**,-**3**,**7**,**8**,-**

2**,-**1**,**5**,**6

01 02 FC FD 07 08

FE FF 05 06

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 10/6/22 00:52:47

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]

LR2\_comp.ASM(54): error A2052: Improper operand type

; Базированная адресация

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

0023 8B 4F 03 mov cx,3[bx]

; Индексированная 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]

LR2\_comp.ASM(61): warning A4031: Operand types must match

; Адресация с базированием 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]

LR2\_comp.ASM(65): warning A4031: Operand types must match

003C 8B 85 0022 R mov ax,matr[bx\*4][di]

LR2\_comp.ASM(66): error A2055: Illegal register value

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ 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]

LR2\_comp.ASM(86): error A2046: Multiple base registers

0061 3E: 8B 83 0016 R mov ax,matr[bp+di+si]

LR2\_comp.ASM(87): error A2047: Multiple index registers

; Использование сегмента стека

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 CB ret

0074 Main ENDP

LR2\_comp.ASM(94): error A2006: Phase error between passes

0074 CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10 10/6/22 00:52:47

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

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

CODE . . . . . . . . . . . . . . 0074 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 = 0074

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

47822 + 459438 Bytes symbol space free

2 Warning Errors

5 Severe Errors

## LR2.ASM

Код программы после комментирования ошибок.

; 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 28**,**27**,**26**,**25**,**21**,**22**,**23**,**24

vec2 DB 20**,**30**,-**20**,-**30**,**40**,**50**,-**40**,-**50

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

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

## LR2.lst

Microsoft **(**R**)** Macro Assembler Version 5**.**10 10**/**6**/**22 00**:**54**:**06

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 1C 1B 1A 19 15 16 vec1 DB 28**,**27**,**26**,**25**,**21**,**22**,**23**,**24

17 18

000E 14 1E EC E2 28 32 vec2 DB 20**,**30**,-**20**,-**30**,**40**,**50**,-**40**,-**50

D8 CE

0016 F8 F9 03 04 FA FB matr DB **-**8**,-**7**,**3**,**4**,-**6**,-**5**,**1**,**2**,-**4**,-**3**,**7**,**8**,-**

2**,-**1**,**5**,**6

01 02 FC FD 07 08

FE FF 05 06

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 10**/**6**/**22 00**:**54**:**06

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

0023 8B 4F 03 mov cx**,**3**[**bx**]**

**;** Индексированная адресация

0026 BF 0002 mov di**,**ind

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

**;** mov cx**,**vec2**[**di**]**

**;** Адресация с базированием 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 АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

**;** Переопределение сегмента

**;** **------** 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 стека

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 CB ret

005E Main ENDP

005E CODE ENDS

END Main

Microsoft **(**R**)** Macro Assembler Version 5**.**10 10**/**6**/**22 00**:**54**:**06

Symbols-1

Segments and Groups**:**

N a m **e** **Length** Align Combine Class

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

CODE **.** **.** **.** **.** **.** **.** **.** **.** **.** **.** **.** **.** **.** **.** 005E 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** **=** 005E

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

47822 **+** 459438 Bytes symbol space **free**

0 Warning Errors

0 Severe Errors