**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 основной памяти.**

|  |  |
| --- | --- |
| Студент гр. 3382 | Копасова К. А. |
| Преподаватель | Фирсов М. А. |

Санкт-Петербург

2024

**Цель работы:** Изучить режимы адресации основной памяти на ассемблере Intel X86.

**Задание:**

1. Получить у преподавателя вариант выбора значений исходных данных (массивов) vec1, vec2 и matr из файла lr2+.dat и занести свои данные вместо значений, указанных в приведённой для образца программе. (вариант 6)

2. Протранслировать программу с созданием файла диагностических сообщений и объяснить обнаруженные ошибки (error) и предупреждения (warning). Листинг привести в приложении к отчёту. Закомментировать операторы с ошибками в тексте программы, а операторы с предупреждениями оставить без изменения. Объяснения ошибок и предупреждений должны быть приведены в отчёте по лабораторной работе.

3. Снова протранслировать программу и скомпоновать загрузочный модуль. Учесть, что программа учебная и может выполняться только под отладчиком. В автоматическом режиме она выполняться не должна.

4. Выполнить программу в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения каждой команды. Разобраться в используемых режимах адресации и получаемых результатах. Результаты прогона программы под управлением отладчика должны быть представлены в отчёте по лабораторной работе в табличном виде, аналогичном указанному в лаб.работе №1.

Вариант 6:

і vec1 і 18,17,16,15,11,12,13,14 і

6 і vec2 і 30,40,-30,-40,10,20,-10,-20 і

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

**Теоретический материал**

Режимы адресации основной памяти в ассемблере Intel x86 определяют, как процессор получает доступ к данным, хранящимся в оперативной памяти. В х86 используются разные режимы адресации для указания на данные, находящиеся по определённым адресам в памяти. Разберём основные режимы адресации в архитектуре х86, включая регистровую и косвенную регистрации, а также различные варианты индексации и базирования.

**Основные режимы адресации:**

1. *Непосредственная адресация:*

В этом режиме значения операнда находится прямо в коде инструкции. Операнд передаётся в регистр или используется непосредственно в операции. Пример: mov ax, 5 ; В регистр ах записывается значение 5

Здесь 5 - непосредственный операнд, который напрямую передаётся в регистр ах.

1. *Регистровая адресация:*

В этом режиме операнд находится в регистре процессора. В этом

случае инструкция работает непосредственно с регистрами.

Пример: mov ax, bx ; В регистр ax копируется значение из регистра bx

Операнд находится в регистре, и перемещением данных происходит между регистрами.

1. *Прямая адресация:*

В этом режиме адрес данных указан прямо в инструкции. Процессор получает данные по указанному адресу.

Пример: mov ax, [1234h] ; В регистр загружается значение из ячейки памяти по адресу 1234h

Здесь 1234h - непосредственный адрес в памяти, где хранятся данные.

1. *Регистрово-косвенная адресация:*

Адрес данных содержится в регистре. Инструкция указывает на регистр, который содержит адрес в памяти, где находятся данные.

Пример: mov ax, [bx] ; В регистр ax загружается значение из памяти по адресу, который хранится в регистре bx

Здесь bx содержит адрес в памяти, и значение из этой памяти перемещаетася в ax.

1. *Базовая адресация:*

Этот режим использует базовый регистр, такой как bx, bp, si или di, для указания адреса в памяти. Пример является таким же, как и в пункте 4 (естественно, можем заменить bx на соответствующие ему базовые регистры)

1. *Индексная адресация:*

Адрес данных определяется с помощью индексного регистра, такого как si или di, который указывает на смещение базового адреса.

Пример: mov ax, [si] ; Используется индексный регистр si для указания адреса данных.

Регистр si используется как индекс, который указывает на смещение от базового адреса в сегменте данных.

1. *Базово-индексная адресация:*

В этом режиме используется одновременно базовый и индексный регистры. Базовый регистр задает начальный адрес, а индексный - смещение базового регистра.

Пример: mov ax, [bx+si] ; В регистр ax загружается значение по адресу, который является суммой значений регистров bx и si.

1. *Базово-индексная адресация с масштабированием:*

Этот режим аналогичен базово-индексному, но позволяет умножить индексный регистр на коэффициент (1, 2, 4 или 8), что удобно для работы с массивами, где каждый элемент имеет фиксированный размер (например, 4 байта).

Пример: mov ax, [bx+si\*4] ; Адрес формируется как bx + (si \* 4)

1. *Косвенная адресация с переопределением сегмента:*

Процессор использует сегментные регистры для обращения к памяти. Обычно для данных используется регистр DS, а для стека SS. Однако этот режим позволяет явно указывать сегмент, используя префикс.

Пример: mov ax, es:[bx] ; В регистр ax заружается значение из сегмента es по адресу bx.

Здесь префикс es: указывает на то, что нудно использовать сегмент es, а не сегмент по умолчанию.

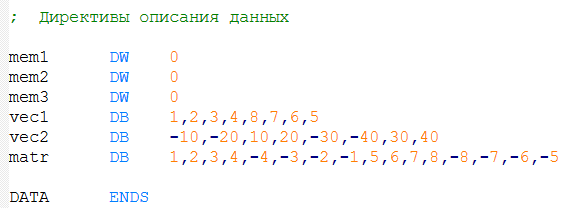
Получается, что каждый режим адресации позволяет эффективно управлять доступом к данным в памяти в зависимости от задач программы:

- Непосредственная и регистровая адресации - самые быстрые, т. к. не требуют обращения к памяти.

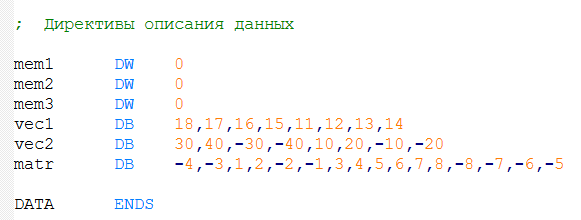
- Прямая, косвенная и индексная адресации - более гибкие, т. к. позволяют работать с данными в памяти, включая сложные структуры, такие как массивы или строки.

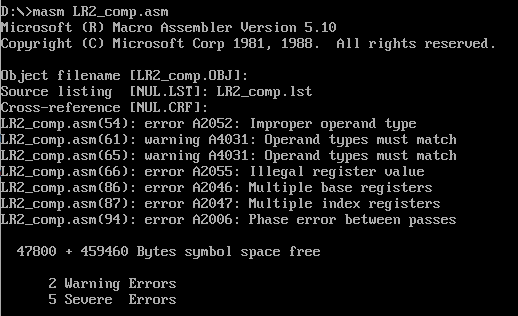
**Выполнение работы**

1. Заносим полученные значения исходных данных (массивов) vec1, vec2 и matr из файла LR2+.DAT вместо значений, приведённых для образца программе.

Данные программы до редактирования:

Данные программы после редактирования:



1. Аналогично, как в 1 лабораторной работе, транслируем программу (с помощью строки masm LR2\_comp.asm):

В результате выполнения программы были созданы объектный файл LR2\_comp.obj и файл листинга LR2\_comp.lst. Были вызваны следующие ошибки:

1. *LR2\_COMP.ASM(54): error A2052: Improper operand type* –

операнд не соответствует ожидаемому типу для данной инструкции. Инструкция MOV требует, чтобы один из операндов был регистром, а другой — либо значением, либо содержимым ячейки памяти, а в данном случае оба операнда указывают на адреса в памяти. Чтобы исправить эту ошибку, надо сперва сохранить данные из bx в регистр, а потом из регистра переместить в mem.

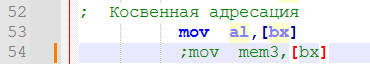
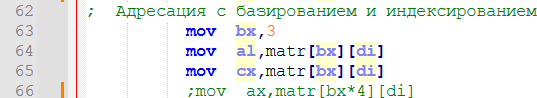
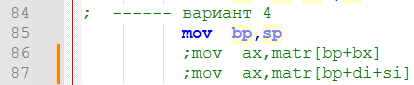
1. *LR2\_COMP.ASM(61): warning A4031: Operand types must match* –предупреждение несоответствия типов данных, так как используется словесный регистр cx, размер которого 16 бит. А Переменная vec2 определена как массив байтов (8-битных значений). Соответственно при попытке загрузить в 16 битный регистр 8 битное значение возникает ошибка несоответствия типов данных.
2. *LR2\_COMP.ASM(65): warning A4031: Operand types must match* – предупреждение. Возникает такая же проблема, как и в предыдущем пункте. Инструкция «mov cx, matr[bx][di]» пытается загрузить байт (8 бит) из массива matr в 16-битный регистр cx, что недопустимо, поскольку операнды имеют разные размеры.
3. *LR2\_COMP.ASM(66): error A2055: Illegal register value* – ошибка некорректного значения регистра. Возникает ошибка из-за неправильного способа адресации памяти. Масштабирование применимо только к 32битным регистрам, а здесь используется базово-индексная адресация. В данном случае необходимо сначала изменить значение регистра, а потом переводить информацию.
4. *LR2\_COMP.ASM(86): error A2046: Multiple base registers* – ошибка

использования нескольких базовых регистров одновременно. Возникает из-за того, что в выражении адресации используется два базовых регистра одновременно. В архитектуре x86 в 16-битном режиме разрешено использовать комбинации одного базового регистра и одного индексного регистра для вычисления адреса. Необходимо один из регистров заменить на индексный.

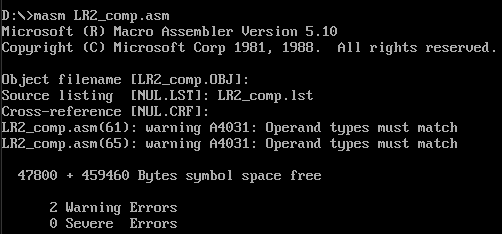
1. *LR2\_COMP.ASM(87): error A2047: Multiple index registers* – ошибка использования нескольких индексных регистров одновременно в строке кода «mov ax,matr[bp+di+si]». В архитектуре x86 использование двух индексных регистров одновременно не поддерживается. Для исправления проблемы нужно вычислить адрес в промежуточном регистре
2. *LR2\_COMP.ASM(94): error A2006: Phase error between passes* – внутренняя ошибка компилятора. Возникает позднее в процессе компиляции, когда фазы многопроходного ассемблера обнаруживают несоответствие или внутренние ошибки, вызванные предыдущими этапами обработки кода. Это означает, что между разными проходами (или фазами) процесса ассемблирования возникла проблема, из-за которой компилятор не смог корректно обработать код. Эта ошибка пропадает, если исправить предыдущие ошибки.

*Листинг с ошибками приведён в Приложении 1.*

Закомментируем операторы с ошибками в тексте программы, а операторы с с предупреждениями оставим без изменений.



1. Снова протранслируем программу с закомментированными инструкциями, которые вызывают ошибки. Получим следующее:

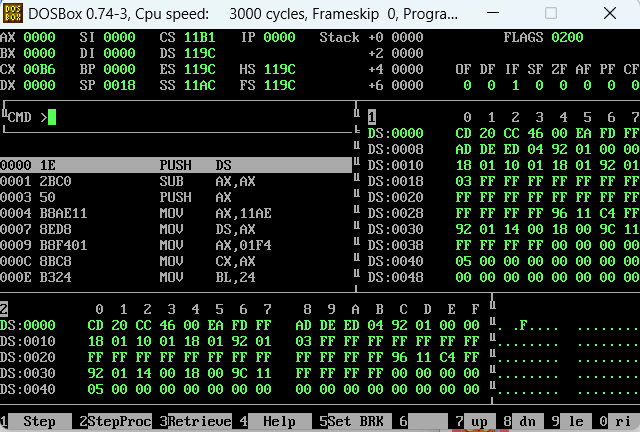


Используем линковщик (link LR2\_COMP.OBJ), чтобы скомпоновать загрузочный модуль:



*Листинг с закомментированными ошибками приведён в Приложении 1.*

1. Выполним программу в пошаговом режиме под управлением отладчика с помощью команды afd LR2\_comp.exe.



Составим таблицу с пошаговыми изменениями в отладчике с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения каждой команды:

Таблица 1 - Пошаговое изменение программы в отладчике.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| До выполнения | После выполнения |
| 0000 | PUSH DS | 1E | (IP) = 0000  (SP)=0018  Stack +0 0000  +2 0000  +4 0000  +6 0000 | (IP) = 0001  (SP)=0016  Stack +0 119C  +2 0000  +4 0000  +6 0000 |
| 0001 | SUB AX, AX | 2BC0 | (IP) = 0001 | (IP) = 0003 |
| 0003 | PUSH AX | 50 | (IP) = 0003  (SP) = 0016  Stack +0 119C  +2 0000  +4 0000  +6 0000 | (IP) = 0004  (SP) = 0014  Stack +0 0000  +2 119C  +4 0000  +6 0000 |
| 0004 | MOV AX, 11AE | B8AE11 | (IP) = 0004  (AX) = 0000 | (IP) = 0007  (AX) = 11AE |
| 0007 | MOV DS, AX | 8ED8 | (IP) = 0007  (DS) = 119C | (IP) = 0009  (DS) = 11AE |
| 0009 | MOV AX, 01F4 | B8F401 | (IP) = 0009  (AX) = 11AE | (IP) = 000C  (AX) = 01F4 |
| 000C | MOV CX, AX | 8BC8 | (IP) = 000C  (CX)=00B6 | (IP) = 000E  (CX)=01F4 |
| 000E | MOV BL, 24 | B324 | (IP) = 000E  (BX) = 0000 | (IP) = 0010  (BX) = 0024 |
| 0010 | MOV BH, CE | B7CE | (IP) = 0010  (BX) = 0024 | (IP) = 0012  (BX) = CE24 |
| 0012 | MOV [0002], FFCE | C7060200CEFF | (IP) = 0012 | (IP) = 0018 |
| 0018 | MOV BX, 0006 | BB0600 | (IP) = 0018  (BX) = CE24 | (IP) = 001B  (BX) = 0006 |
| 001B | MOV [0000], AX | A30000 | (IP) = 001B | (IP) = 001E |
| 001E | MOV AL, [BX] | 8A07 | (IP) = 001E  (AX) = 01F4 | (IP) = 0020  (AX) = 0112 |
| 0020 | MOV AL, [BX+03] | 8A4703 | (IP) = 0020  (AX) = 0112 | (IP) = 0023  (AX) = 010F |
| 0023 | MOV CX, [BX+03] | 8B4F03 | (IP) = 0023  (CX) = 01F4 | (IP) = 0026  (CX) = 0B0F |
| 0026 | MOV DI, 0002 | BF0200 | (IP) = 0026  (DI) = 0000 | (IP) = 0029  (DI) = 0002 |
| 0029 | MOV AL, [DI+000E] | 8A850E00 | (IP) = 0029  (AX) = 010F | (IP) = 002D  (AX) = 01E2 |
| 002D | MOX CX, [DI+000E] | 8B8D0E00 | (IP) = 002D  (CX) = 0B0F | (IP) = 0031  (CX) = D8E2 |
| 0031 | MOV BX, 0003 | BB0300 | (IP) = 0031  (BX) = 0006 | (IP) = 0034  (BX) = 0003 |
| 0034 | MOV AL, [BX+DI+0016] | 8A811600 | (IP) = 0034  (AX) = 01E2 | (IP) = 0038  (AX) = 01FF |
| 0038 | MOV CX, [BX+DI+0016] | 8B891600 | (IP) = 0038  (CX) = D8E2 | (IP)=003C  (CX)=03FF |
| 003C | MOV AX, 11AE | B8AE11 | (IP) = 003C  (AX) = 01FF | (IP) = 003F  (AX) = 11AE |
| 003F | MOV ES, AX | 8ECO | (IP) = 003F  (ES) = 119C | (IP) = 0041  (ES) = 11AE |
| 0041 | MOV AX, ES:[BX] | 268B07 | (IP) = 0041  (AX) = 11AE | (IP) = 0044  (AX) = 00FF |
| 0044 | MOV AX, 0000 | B80000 | (IP) = 0044  (AX) = 00FF | (IP) = 0047  (AX) = 0000 |
| 0047 | MOV ES, AX | 8ECO | (IP) = 0047  (ES) = 11AE | (IP) = 0049  (ES) = 0000 |
| 0049 | PUSH DS | 1E | (IP) = 0049  (SP) = 0014  Stack +0 0000  +2 119C  +4 0000  +6 0000 | (IP) = 004A  (SP) = 0012  Stack +0 11AE  +2 0000  +4 119C  +6 0000 |
| 004A | POP ES | 07 | (IP) = 004A  (SP) = 0012  (ES) = 0000  Stack +0 11AE  +2 0000  +4 119C  +6 0000 | (IP) = 004B  (SP) = 0014  (ES) = 11AE  Stack +0 0000  +2 119C  +4 0000  +6 0000 |
| 004B | MOV CX, ES:[BX-01] | 268B4FFF | (IP) = 004B  (CX) = 03FF | (IP) = 004F  (CX) = FFCE |
| 004F | XCHG AX, CX | 91 | (IP) = 004F  (AX) = 0000  (CX) = FFCE | (IP) = 0050  (AX) = FFCE  (CX) = 0000 |
| 0050 | MOV DI, 0002 | BF0200 | (IP) = 0050  (DI) = 0002 | (IP) = 0053  (DI) = 0002 |
| 0053 | MOV ES:[BX+DI], AX | 268901 | (IP) = 0053  DS:0005 00  DS:0006 15 | (IP) = 0056  DS:0005 CE  DS:0006 FF |
| 0056 | MOV BP, SP | 8BEC | (IP) = 0056  (BP) = 0000 | (IP) = 0058  (BP) = 0014 |
| 0058 | PUSH [0000] | FF360000 | (IP) = 0058  (SP) = 0014  Stack +0 0000  +2 119C  +4 0000  +6 0000 | (IP) = 005C  (SP) = 0012  Stack +0 01F4  +2 0000  +4 119C  +6 0000 |
| 005C | PUSH [0002] | FF360200 | (IP) = 005C  (SP) = 0012  Stack +0 01F4  +2 0000  +4 119C  +6 0000 | (IP) = 0060  (SP) = 0010  Stack +0 FFCE  +2 01F4  +4 0000  +6 119C |
| 0060 | MOV BP, SP | 8BEC | (IP) = 0060  (BP) = 0014 | (IP) = 0062  (BP) = 0010 |
| 0062 | MOV DX, [BP+02] | 8B5602 | (IP) = 0062  (DX) = 0000 | (IP) = 0065  (DX) = 01F4 |
| 0065 | RET FAR | CB | (IP) = 0065  (SP) = 0010  (CS) = 11B1  Stack +0 FFCE  +2 01F4  +4 0000  +6 119C | (IP) = FFCE  (SP) = 0014  (CS) = 01F4  Stack +0 0000  +2 119C  +4 0000  +6 0000 |

**Вывод**

В ходе данной лабораторной работы были изучены режимы адресации основной памяти на ассемблере Intel X86. Удалось рассмотреть файл LR2\_comp.asm, в котором находились различные ошибки и предупреждения, которые были подробно изучены. В процессе лабораторной работы также были закомментированы все ошибки и был проведён пошаговый анализ программы с помощью отладчика afd. Файлы, созданные и использованные в процессе лабораторной работы находятся в Приложении 1.

**ПРИЛОЖЕНИЕ 1**

Файл LR2\_comp.asm (версия с ошибками):

; 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 18,17,16,15,11,12,13,14

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

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

; 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

Файл LR2\_comp.lst (версия с ошибками):

Microsoft (R) Macro Assembler Version 5.10 9/22/24 14:51:55

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 12 11 10 0F 0B 0C vec1 DB 18,17,16,15,11,12,13,14

0D 0E

000E 1E 28 E2 D8 0A 14 vec2 DB 30,40,-30,-40,10,20,-10,-20

F6 EC

0016 FC FD 01 02 FE FF matr DB -4,-3,1,2,-2,-1,3,4,5,6,7,8,-8,-7,-6,-5

03 04 05 06 07 08

F8 F9 FA FB

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 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]

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

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

47800 + 459460 Bytes symbol space free

2 Warning Errors

5 Severe Errors

Файл LR2\_comp.map:

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 00045H 00026H DATA

00050H 000B5H 00066H CODE

Program entry point at 0005:0000

Файл LR2\_comp.asm (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 18,17,16,15,11,12,13,14

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

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

; 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

Файл LR2\_comp.lst (после комментирования ошибок):

Microsoft (R) Macro Assembler Version 5.10 9/22/24 21:23:20

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 12 11 10 0F 0B 0C vec1 DB 18,17,16,15,11,12,13,14

0D 0E

000E 1E 28 E2 D8 0A 14 vec2 DB 30,40,-30,-40,10,20,-10,-20

F6 EC

0016 FC FD 01 02 FE FF matr DB -4,-3,1,2,-2,-1,3,4,5,6,7,8,-8,-7,-6,-5

03 04 05 06 07 08

F8 F9 FA FB

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 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]

;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]

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

;mov ax,matr[bx\*4][di]

; ПРОВЕРКА АДРЕСАЦИИ С УЧЕТОМ СЕГМЕНТОВ

; Переопределение сегмента

; ------ вариант 1

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

; ------ вариант 2

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

; ------ вариант 3

0050 BF 0002 mov di,ind

0053 26: 89 01 mov es:[bx+di],ax

; ------ вариант 4

0056 8B EC mov bp,sp

;mov ax,matr[bp+bx]

;mov ax,matr[bp+di+si]

; Использование сегмента стека

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

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 LR2\_comp

@VERSION . . . . . . . . . . . . TEXT 510

96 Source Lines

96 Total Lines

19 Symbols

47800 + 459460 Bytes symbol space free

2 Warning Errors

0 Severe Errors