**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Организация систем и ЭВМ»**

Тема: Изучение режимов адресации основной памяти

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 |  | Еникеев А.А. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2024

## Цель работы

Изучить режимы адресации основной памяти, найти ошибки и предупреждения в файле с кодом программы, объяснить их.

## Задание

**Вариант 9**

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.
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 9:

і vec1 DB 31,32,33,34,38,37,36,35

9 і vec2 DB 50,60,-50,-60,70,80,-70,-80

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

## Основные теоритические положения

Непосредственный операнд — это фиксированное значение, которое прописано напрямую в коде программы. В отличие от других типов операндов, таких как регистры или адреса памяти, которые могут изменяться во время выполнения программы, непосредственные операнды всегда имеют одно и то же значение.

Пересылка типа память-память невозможна. Также нельзя пересылать данные из одного сегментного регистра в другой и записывать непосредственный операнд в сегментный регистр — для этого нужно использовать промежуточные несегментные регистры.

Запись в квадратных скобочках, например [00504032h] означает, что нужно взять значение из памяти расположенное по адресу 00504032h. Важная деталь, в команде MOV приемник не может иметь такой вид: [00504032h]. Нужно обязательно использовать промежуточный регистр. Нужно также помнить еще об одной важной вещи, что размерность обоих операндов вкоманде MOV должна совпадать (либо байты, либо слова), иначе ассемблер зафиксирует ошибку.

* **Непосредственная адресация**: операнд указывается непосредственно в поле команды, например:

mov ax,312 ; здесь 312 задается непосредственно

* **Регистровая адресация**: операнд находится в одном из регистров. Например, оба операнда в команде

mov ds,bx

задаются с помощью регистрового способа адресации.

* **Косвенная адресация**: адрес операнда находится в одном из регистров— SI, DI, BX, BP. Например, команда

mov ax,[si]

помещает в регистр AX слово из ячейки памяти, смещение которой указано в регистре SI. Второй операнд задан с помощью косвенной адресации.

* **Прямая адресация (адресация по смещению)**: абсолютный адрес операнда можно задавать в виде СЕГМЕНТ:СМЕЩЕНИЕ, где СЕГМЕНТ — адрес сегмента на который указывает какой-либо сегментный регистр (CS, DS, SS или ES), а СМЕЩЕНИЕ — адрес операнда относительно сегментного регистра (относительный адрес). Примеры:

mov ax,ss:0037h

mov bx,es:var

* **Базовая адресация**: адрес операнда формируется сложением содержимого базового регистра (BP или BX) и смещения. Если смещение не задано, то предполагается нулевое значение. Если используется BP, то адрес определенного операнда отсчитывается относительно сегмента, на который указывает регистр SS. Если используется BX, то адрес отсчитывается относительно сегмента, на который указывает регистр DS.

Например, команда:

mov ax,[bx+2]

помещает в регистр AX слово, которое находится в сегменте, указанном в DS, со смещением на два больше, чем число из BX.

* **Индексная адресация**: этот метод адресации подобен предыдущему, только адрес операнда формируется сложением содержимого индексного регистра (SI или DI) и смещения. Если смещение не задано, то предполагается нулевое значение. В индексной адресации можно использовать множитель 1, 2, 4 или 8 чтобы прочитать элемент равный соответственно – байту, слову, двойному или учетверенному слову. Это называется масштабированием индексного регистра. Примеры:

mov ax,[si]

mov ax,[di]

mov ax,12[di]

mov ax,[esi\*4]+2

* **Базовая-индексная адресация**: адрес операнда формируется сложением содержимого базового регистра (BP или BX) и индексного регистра (SI или DI), и смещения, если оно указано. Если используется регистр BP, то адрес отсчитывается относительно сегмента, на который указывает регистр SS. В ином случае этот адрес отсчитывается относительно сегмента, на который указывает регистр DS. Примеры:

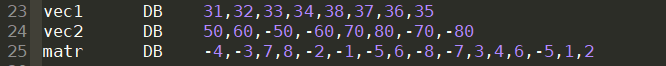
mov ax,[bp][si]

mov ax,12[bp+di]

mov ax,[bx][si+2]

## Выполнение работы

1. Занесем данные соотвествующие варианту в массивы vec1, vec2 и matr (см. рис. 1)

  
Рисунок 1

1. Протранслируем программу (см. рис. 2)

  
Рисунок 2

При трансляции обнаружено 2 предупреждения и 5 ошибок, рассмотрим причины их возникновения:

* **LR2\_comp.ASM(54): error A2052: Improper operand type**: проблема в том, что инструкция MOV позволяет загружать данные из памяти в регистр или сохранять данные из регистра в память, но не позволяет напрямую копировать данные из одной области памяти в другую, как пытаются сделать в этом фрагменте кода:

mov mem3,[bx]

* **LR2\_comp.ASM(61): warning A4031: Operand types must match**: проблема в том, что размерность регистра CX 2 байта, а vec2 – массив байтов, соотвественно 1 байт не получится корректно загрузить, как пытаются сделать в этом фрагменте кода:

mov cx,vec2[di]

* **LR2\_comp.ASM(65): warning A4031: Operand types must match**: проблема описана выше (несовпадают размерности), фрагмент кода:

mov cx,matr[bx][di]

* **LR2\_comp.ASM(66): error A2055: Illegal register value**: здесь проблема в том, что регистр BX 16-битный, поэтому не допускается умножение:

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

* **LR2\_comp.ASM(86): error A2046: Multiple base registers**: в ассемблере x86 разрешено использовать только один базовый регистр для адресации в одном выражении. Здесь используется два базовых регистра:

mov ax,matr[bp+bx]

* **LR2\_comp.ASM(87): error A2047: Multiple index registers:** Нельзя использовать два или более индексных регистра одновременно в одном выражении:

mov ax,matr[bp+di+si]

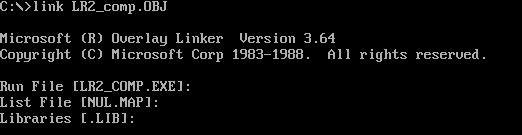
* **LR2\_comp.ASM(94): error A2006: Phase error between passes:** возникает из-за неправильного использования инструкций, в данном случае

из-за ошибки **error A2052**.

Закомментируем в коде все возникшие ошибки.

1. Снова протранслируем программу и скомпoнуем загрузочный модуль (см. рис. 3 и 4).

  
Рисунок 3

  
Рисунок 4

1. Выполним программу LR2\_comp с использованием отладчика в пошаговом режиме. Значения регистров до выполнения программы см. на рисунке 5. Фиксация используемых регистров на каждом шаге приведена в табл. 1.

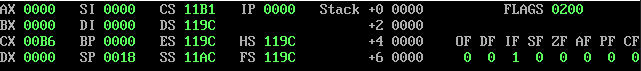
  
Рисунок 5

Табл. 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес команды | Символический код команды | 16-ричный код команды | Содержимое регистров и ячеек памяти | |
| до вып. команды | после вып. команды |
| 0000 | PUSH DS | 1E | (Stack + 0) = 0000  (SP) =0018  (IP) = 0000 | (Stack +0) = 119C  (SP) = 0016  (IP) = 0001 |
| 0001 | SUB AX, AX | 2BC0 | (IP) = 0001 | (IP) = 0003 |
| 0003 | PUSH AX | 50 | (Stack +0) = 119C  (Stack + 2) = 0000  (SP) = 0016  (IP) = 0003 | (Stack +0) = 0000  (Stack +2) = 119C  (SP) = 0014  (IP) = 0004 |
| 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 | [0002] = 0000  (IP) = 0012 | [0002] = CEFF  (IP) = 0018 |
| 0018 | MOV BX,0006 | BB0600 | (BX) = CE24  (IP) = 0018 | (BX) = 0006  (IP) = 001B |
| 001B | MOV [0000],AX | A30000 | [0000] = 0000  (IP) = 001B | [0000] = F401  (IP) = 001E |
| 001E | MOV AL,[BX] | 8A07 | (AX) = 01F4  (IP) = 001E | (AX) = 011F  (IP) = 0020 |
| 0020 | MOV AL,[BX+03] | 8A4703 | (AX) = 011F  (IP) = 0020 | (AX) = 0122  (IP) = 0023 |
| 0023 | MOV CX,[BX+03] | 8B4F03 | (CX) = 01F4  (IP) = 0023 | (CX) = 2622  (IP) = 0026 |
| 0026 | MOV DI,0002 | BF0200 | (DI) = 0000  (IP) = 0026 | (DI) = 0002  (IP) = 0029 |
| 0029 | MOV AL,[DI+000E] | 8A850E00 | (AX) = 0122  (IP) = 0029 | (AX) = 01CE  (IP) = 002D |
| 002D | MOV CX,[DI+000E] | 8B8D0E00 | (CX) = 2622  (IP) = 002D | (CX) = C4CE  (IP) = 0031 |
| 0031 | MOV BX,0003 | BB0300 | (BX) = 0006  (IP) = 0031 | (BX) = 0003  (IP) = 0034 |
| 0034 | MOV AL,[BX+DI+0016] | 8A811600 | (AX) = 01CE  (IP) = 0034 | (AX) = 01FF  (IP) = 0038 |
| 0038 | MOV CX,[BX+DI+0016] | 8B891600 | (CX) = C4CE  (IP) - 0038 | (CX) = FBFF  (IP) = 003C |
| 003C | MOV AX,11AE | B8AE11 | (AX) = 01FF  (IP) = 003C | (AX) = 11AE  (IP) = 003F |
| 003F | MOV ES,AX | 8EC0 | (ES) = 119C  (IP) = 003F | (ES) = 11AE  (IP) = 0041 |
| 0041 | MOV AX,ES:[BX] | 268B07 | (AX) = 11AE  (IP) = 0041 | (AX) = 00FF  (IP) = 0044 |
| 0044 | MOV AX,0000 | B80000 | (AX) = 00FF  (IP) = 0044 | (AX) = 0000  (IP) = 0047 |
| 0047 | MOV ES,AX | 8EC0 | (ES) = 11AE  (IP) = 0047 | (ES) = 0000  (IP) = 0049 |
| 0049 | PUSH DS | 1E | (Stack +0) = 0000  (Stack +2) = 119C  (Stack +4) = 0000  (SP) = 0014  (IP) = 0049 | (Stack +0) = 11AE  (Stack +2) = 0000  (Stack +4) = 119C  (SP) = 0012  (IP) = 004A |
| 004A | POP ES | 07 | (Stack +0) = 11AE  (Stack +2) = 0000  (Stack +4) = 119C  (SP) = 0012  (IP) = 004A  (ES) = 0000 | (Stack +0) = 0000  (Stack +2) = 119C  (Stack +4) = 0000  (SP) = 0014  (IP) = 004B  (ES) = 11AE |
| 004B | MOV CX,ES:[BX-01] | 268B4FFF | (CX) = FBFF  (IP) = 004B | (CX) = FFCE  (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:[BX+DI] = 001F  (IP) = 0053 | ES:[BX+DI] = CEFF  (IP) = 0056 |
| 0056 | MOV BP, SP | 8BEC | (BP) = 0000  (IP) = 0056 | (BP) = 0014  (IP) = 0058 |
| 0058 | PUSH [0000] | FF360000 | (Stack +0) = 0000  (Stack +2) = 119C  (Stack +4) = 0000  (SP) = 0014  (IP) = 0058 | (Stack +0) = 01F4  (Stack +2) = 0000  (Stack +4) = 119C  (SP) = 0012  (IP) = 005C |
| 005C | PUSH [0002] | FF360200 | (Stack +0) = 01F4  (Stack +2) = 0000  (Stack +4) = 119C  (Stack +6) = 0000  (SP) = 0012  (IP) = 005C | (Stack +0) = FFCE  (Stack +2) = 01F4  (Stack +4) = 0000  (Stack +6) = 119C  (SP) = 0010  (IP) = 0060 |
| 0060 | MOV BP,SP | 8BEC | (BP) = 0014  (IP) = 0060 | (BP) = 0010  (IP) = 0062 |
| 0062 | MOV DX,[BP+02] | 8B5602 | (DX) = 0000  (IP) = 0062 | (DX) = 01F4  (IP) = 0065 |
| 0065 | RET Far | CB | (SP) = 0010  (IP) = 0065 | (SP) = 0014  (IP) = FFCE |

# Выводы

Были изучены различные варианты адресации основной памяти программы, пронализированы ошибки, допущенные в исходном коде программы, составлена таблица с содержимым регистров на каждом шаге выполнения программы.

# Приложение A

**Файл LR2\_comp.ASM:**

; 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 31,32,33,34,38,37,36,35

vec2 DB 50,60,-50,-60,70,80,-70,-80

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

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:**

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 16:49:42

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 01 02 03 04 08 07 vec1 DB 1,2,3,4,8,7,6,5

06 05

000E F6 EC 0A 14 E2 D8 vec2 DB -10,-20,10,20,-30,-40,30,40

1E 28

0016 01 02 03 04 FC FD matr DB 1,2,3,4,-4,-3,-2,-1,5,6,7,8,-8,

-7,-6,-5

FE FF 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 СМЕЩЕНИ

Й

; Регистровая адресация

0009 B8 01F4 mov ax,n1

#Microsoft (R) Macro Assembler Version 5.10 9/25/24 16:49:42

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 9/25/24 16:49:42

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

47800 + 459460 Bytes symbol space free

2 Warning Errors

5 Severe Errors