Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №5

по дисциплине: «Архитектура ИС»

**“** Исследование методов адресации и программирования арифметических и

логических операций**”**

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Дрозин А.Ю.

г. Севастополь

2016 г.

1. Цель работы

Изучить основные директивы языка ассемблера, исследовать их воздействие на процесс ассемблирования и формирования листинга программы.

Исследовать особенности функционирования блоков 16-разрядного микропроцессора при выполнении арифметических и логических операций и при использовании различных способов адресации. Приобрести практические навыки программирования на языке ассемблера МП 8086 арифметических и логических операций с применением различных способов адресации.

2.Вариант задания

2.1. Изучить основные директивы ассемблера и их воздействие на процесс ассемблирования и формирования листинга программы. Повторить команды пересылки данных, а также команды арифметических и логических операций (выполняется в процессе домашней подготовки к лабораторной работе).

2.2. Изучить методы адресации, используемые в 16-разрядных процессорах (выполняется во время домашней подготовке к работе).

2.3. Составит программу, состоящую из следующих процедур обработки строк:

2.3.1. Заполнить 100+10i ячеек области памяти, начинающейся с адреса MAS рядом натуральных чисел. Здесь i – последняя цифра номера Вашей зачетной книжки.

2.3.2. Переслать массив слов из области памяти, начиная с адреса MAS1 в область с начальным адресом MAS2.

2.3.3. Найти в заданном массиве число, равное двум последним цифрам Вашей зачетной книжки и определить его индекс.

2.4. Переслать в память с адресом 2020:300 диагональные элементы матрицы размером 8x8. Значения элементов матрицы должны быть определены в сегменте данных программы.

2.5. Произвести отладку разработанных программ в пошаговом режиме и проследить за изменениями содержимого регистров

2.6. Произвести ассемблирование программы и получить объектный и исполняемый модуль программы в Ехе-формате и ее листинг.

2.7. Рассчитать время выполнения программы.

3. Код программы

title lab5.asm

data segment para 'data'

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

DB 9, 10, 11, 12, 13, 14, 15, 16

DB 17, 18, 19, 20, 21, 22, 23, 24

DB 25, 26, 27, 28, 29, 30, 31, 32

DB 33, 34, 35, 36, 37, 38, 39, 40

DB 41, 42, 43, 44, 45, 46, 47, 48

DB 49, 50, 51, 52, 53, 54, 55, 56

DB 57, 58, 59, 60, 61, 62, 63, 64

data ends

stack segment para 'stack'

dw 128 dup(0)

ends

code segment para 'code'

MOV ax, 1000h; MAS adress

MOV ds, ax;

MOV ax, 1h;

MOV si, 0;

MOV bh, 120d;

cBegin:

MOV [si], ax;

INC si;

INC ax;

DEC bh;

jnz cBegin;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

MOV si, 120d;

copyBegin:

MOV ax, 1000h; MAS1 adress

MOV ds, ax;

MOV bx, [si];

cmp bx, 2d;

jz found;

MOV ax, 2000h; MAS1 adress

MOV ds, ax;

MOV [si], bx;

DEC si;

jnz copyBegin

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

LEA dx, array;

MOV ax, 2020H;

MOV ds, ax;

MOV si, 0300H;

MOV al, 8d;

c1:

LEA bx, dx;

INC bx;

MOV [si], bx;

INC si;

MOV ah, 9d;

c2:

INC dx;

DEC ah;

jnz c2;

DEC al;

JNZ c1;

ret;

found:

MOV cx, 1111h;

ret

4. Листинг программы

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

ais5.exe -- emu8086 assembler version: 4.05

[ 01.05.2016 -- 12:58:35 ]

===================================================================================================

[LINE] LOC: MACHINE CODE SOURCE

===================================================================================================

[ 1] : title lab5.asm

[ 2] :

[ 3] : data segment para 'data'

[ 4] 0000: 01 02 03 04 05 06 07 08 array DB 1, 2, 3, 4, 5, 6, 7, 8

[ 5] 0008: 09 0A 0B 0C 0D 0E 0F 10 DB 9, 10, 11, 12, 13, 14, 15, 16

[ 6] 0010: 11 12 13 14 15 16 17 18 DB 17, 18, 19, 20, 21, 22, 23, 24

[ 7] 0018: 19 1A 1B 1C 1D 1E 1F 20 DB 25, 26, 27, 28, 29, 30, 31, 32

[ 8] 0020: 21 22 23 24 25 26 27 28 DB 33, 34, 35, 36, 37, 38, 39, 40

[ 9] 0028: 29 2A 2B 2C 2D 2E 2F 30 DB 41, 42, 43, 44, 45, 46, 47, 48

[ 10] 0030: 31 32 33 34 35 36 37 38 DB 49, 50, 51, 52, 53, 54, 55, 56

[ 11] 0038: 39 3A 3B 3C 3D 3E 3F 40 DB 57, 58, 59, 60, 61, 62, 63, 64

[ 12] :

[ 13] : data ends

[ 14] :

[ 15] : stack segment para 'stack'

[ 16] 0040: 00 00 00 00 00 00 00 00 00 00 00 00 dw 128 dup(0)

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00

[ 17] : ends

[ 18] :

[ 19] : code segment para 'code'

[ 20] :

[ 21] 0140: B8 00 10 MOV ax, 1000h; MAS adress

[ 22] 0143: 8E D8 MOV ds, ax;

[ 23] :

[ 24] 0145: B8 00 20 MOV ax, 2000h; MAS adress

[ 25] 0148: 8E C0 MOV es, ax;

[ 26] :

[ 27] :

[ 28] 014A: B8 01 00 MOV ax, 1h;

[ 29] 014D: BE 00 00 MOV si, 0;

[ 30] 0150: B7 78 MOV bh, 120d;

[ 31] :

[ 32] :

[ 33] 0152: cBegin:

[ 34] 0152: 89 04 MOV [si], ax;

[ 35] :

[ 36] 0154: 46 INC si;

[ 37] 0155: 40 INC ax;

[ 38] :

[ 39] 0156: FE CF DEC bh;

[ 40] 0158: 75 F8 jnz cBegin;

[ 41] :

[ 42] :

[ 43] : ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[ 44] 015A: BE 78 00 MOV si, 120d;

[ 45] :

[ 46] 015D: copyBegin:

[ 47] 015D: B8 00 10 MOV ax, 1000h; MAS1 adress

[ 48] 0160: 8E D8 MOV ds, ax;

[ 49] :

[ 50] 0162: 8B 1C MOV bx, [si];

[ 51] 0164: 83 FB 02 cmp bx, 2d;

[ 52] 0167: 74 29 jz found;

[ 53] :

[ 54] 0169: B8 00 20 MOV ax, 2000h; MAS1 adress

[ 55] 016C: 8E D8 MOV ds, ax;

[ 56] :

[ 57] 016E: 89 1C MOV [si], bx;

[ 58] :

[ 59] 0170: 4E DEC si;

[ 60] 0171: 75 EA jnz copyBegin

[ 61] :

[ 62] :

[ 63] : ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[ 64] 0173: test1:

[ 65] 0173: BA 00 00 LEA dx, array;

[ 66] :

[ 67] :

[ 68] 0176: B8 20 20 MOV ax, 2020H;

[ 69] 0179: 8E D8 MOV ds, ax;

[ 70] 017B: BE 00 03 MOV si, 0300H;

[ 71] 017E: B0 08 MOV al, 8d;

[ 72] :

[ 73] 0180: c1:

[ 74] 0180: 8B DA LEA bx, dx;

[ 75] 0182: 43 INC bx;

[ 76] :

[ 77] 0183: 89 1C MOV [si], bx;

[ 78] 0185: 46 INC si;

[ 79] 0186: B4 09 MOV ah, 9d;

[ 80] :

[ 81] 0188: c2:

[ 82] 0188: 42 INC dx;

[ 83] 0189: FE CC DEC ah;

[ 84] 018B: 75 FB jnz c2;

[ 85] :

[ 86] 018D: FE C8 DEC al;

[ 87] 018F: 75 EF JNZ c1;

[ 88] : ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[ 89] 0191: C3 ret;

[ 90] :

[ 91] :

[ 92] 0192: found:

[ 93] 0192: B9 11 11 MOV cx, 1111h;

[ 94] 0195: C3 ret

[ 95] : ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[ 96] :

===================================================================================================

EXE HEADER - bytes from 0000 to 01FF inclusive.

0000: 4D - exe signature (M)

0001: 5A - exe signature (Z)

0002: 96 - bytes on last page (l.byte)

0003: 01 - bytes on last page (h.byte)

0004: 02 - 512 byte pages in file (l.byte)

0005: 00 - 512 byte pages in file (h.byte)

0006: 00 - relocations (l.byte)

0007: 00 - relocations (h.byte)

0008: 20 - paragraphs in header (l.byte)

0009: 00 - paragraphs in header (h.byte)

000A: 00 - minimum memory (l.byte)

000B: 00 - minimum memory (h.byte)

000C: FF - maximum memory (l.byte)

000D: FF - maximum memory (h.byte)

000E: 04 - SS - stack segment (l.byte)

000F: 00 - SS - stack segment (h.byte)

0010: 00 - SP - stack pointer (l.byte)

0011: 01 - SP - stack pointer (h.byte)

0012: 89 - check sum (l.byte)

0013: BF - check sum (h.byte)

0014: 00 - IP - instruction pointer (l.byte)

0015: 00 - IP - instruction pointer (h.byte)

0016: 14 - CS - code segment (l.byte)

0017: 00 - CS - code segment (h.byte)

0018: 1E - relocation table adress (l.byte)

0019: 00 - relocation table adress (h.byte)

001A: 00 - overlay number (l.byte)

001B: 00 - overlay number (h.byte)

001C: 01 - signature (l.byte)

001D: 00 - signature (h.byte)

001E to 01FF - reserved relocation area (00)

5. Выполнение программы

Данная программа выполняет следующие действия в заданном порядке:

Записать в адрес памяти MAS (1000H) ряд из 120 натуральных чисел (рис.5.1).

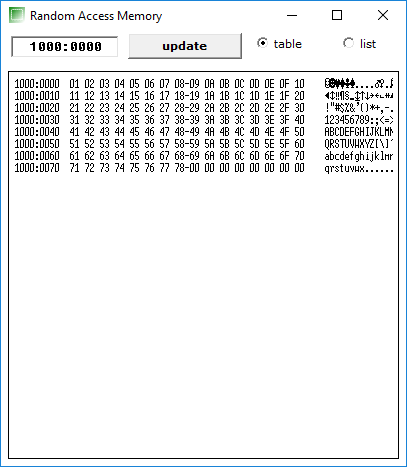


Рисунок 5.1. – Ряд натуральных чисел.

Переместить данные из области памяти MAS1(1000H) в область памяти MAS2 (2000H) в количестве 120 элементов. (рис.5.2)

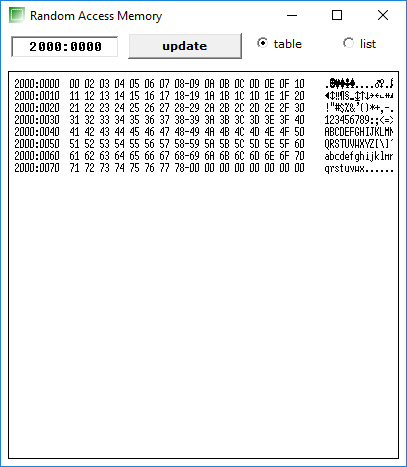


Рисунок 5.2 – Копирование.

При нахождении конкретного числа в ряде чисел в регистр CX будет выведено число 1111H.

Также осуществляется вывод элементов в область памяти, начиная с адреса 2020:0300 главной диагонали из матрицы, заданной в сегменте данных.

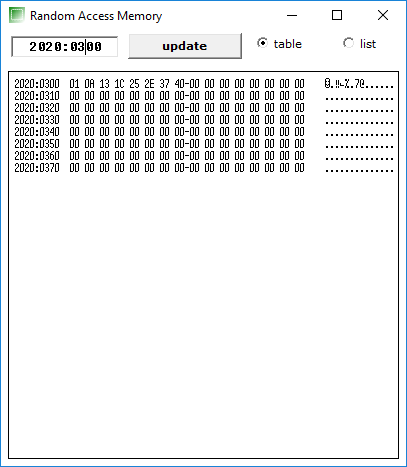


Рисунок 5.3 – Вывод диагональных элементов.

ВЫВОД

В данной лабораторной были изучены команды процессора Intel8086 в эмуляторе emu8086, их взаимодействие с регистрами, а также директивы, используемые при написании программы на Assembler. Был получен листинг программы и exe-файл.