

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №1

Основные конструкции. Написание простейшей программы

Выполнил: Студент: гр. 853501
Астрашаб Владислав Владимирович

Руководитель: ст.преподаватель
Шиманский Валерий Владимирович

Минск 2019

СОДЕРЖАНИЕ

1. Введение
2. Постановка задачи
3. Программная реализация
4. Выводы

Литература

Исходный код

1. Введение

Целью данной работы является изучить следующий материал:

- 1) Регистры процессора 8086.
- 2) Логика работы команд MOV, ADD, SUB, MUL, DIV.
- 3) Логические операции AND, OR, XOR, NOT.
- 4) Команды сдвига SHL и SHR.
- 5) Работа команд CMP и TEST.
- 6) Последовательное выполнение команд. Назначение регистра IP.
- 7) Логика работы следующих команд условного и безусловного переходов: JMP, JE, JNE, JC, JNC.

Назначение флагов CF и ZF. Использование меток.

- 8) Размещение данных в сегменте данных. Размерность данных: DB, DW, DD. Работа с переменными, определенными в сегменте данных.
- 9) Компилирование, линковка, выполнение и отладка ассемблерных программ.

2. Постановка задачи

2.1. Текст задания

Если $a^2 \leq b^3$ то

Если $c * d = a / b$ то

Результат = $a \text{ XOR } b$

Иначе

Результат = найти 2 наибольших среди a, d, c, d и перемножить

Иначе

Результат = $a * b + c/d$

2.2. Условие задания

В каждом из заданий переменные a, b, c, d определяются в сегменте данных и имеют размерность слово. Необходимо выполнить над ними заданные арифметические и логические операции, а результат поместить в регистр AX.

При выполнении умножения считаем, что результат вмещается в слово. При выполнении деления считаем, что оно целочисленное.

3. Программная реализация

3.1. Значения переменных устанавливаются при объявлении сегмента данных. Программа разбита при помощи меток на несколько логических частей, каждая из которых выполняет определенную ветку условия.

3.2. Результат можно видеть в отладчике в регистре AX

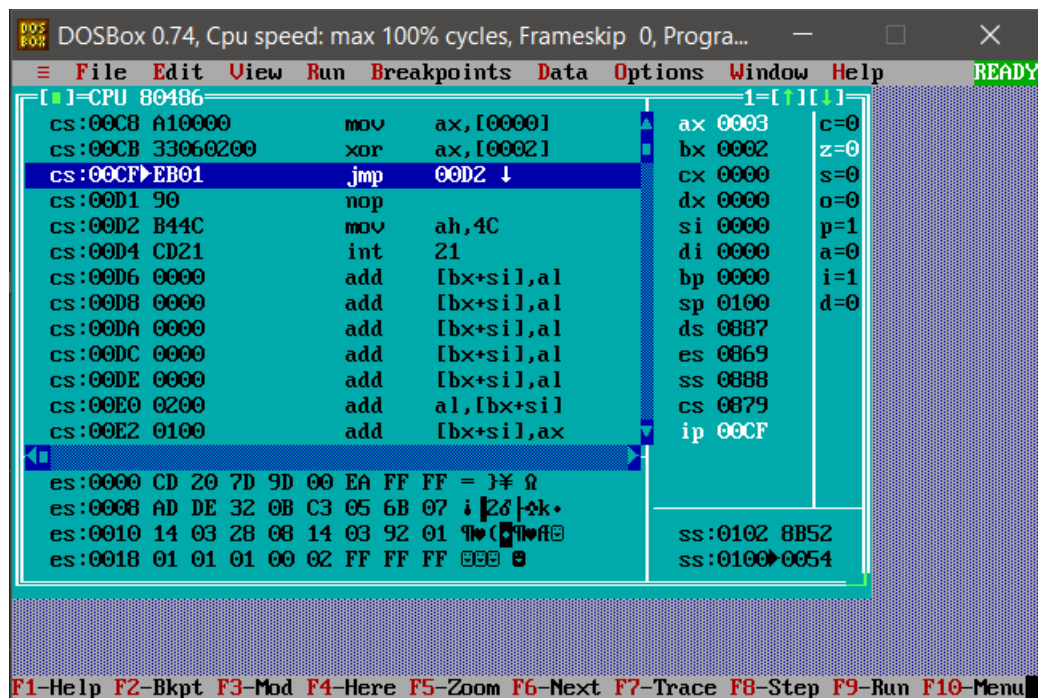
3.3. Примеры:

3.3.1 Отладка ветки №1

Тест для ветки

« Результат = a XOR b »

a = 2 b = 1 c = 1 d = 2



The screenshot shows the DOSBox 0.74 interface with the assembly window open. The assembly code is as follows:

Address	Disassembly	Comment
cs:00C8 A10000	mov ax,[0000]	
cs:00CB 33060200	xor ax,[0002]	
cs:00CF EB01	jmp 00D2 ↓	
cs:00D1 90	nop	
cs:00D2 B44C	mov ah,4C	
cs:00D4 CD21	int 21	
cs:00D6 0000	add [bx+si],al	
cs:00D8 0000	add [bx+si],al	
cs:00DA 0000	add [bx+si],al	
cs:00DC 0000	add [bx+si],al	
cs:00DE 0000	add [bx+si],al	
cs:00E0 0200	add al,[bx+si]	
cs:00E2 0100	add [bx+si],ax	

The registers window on the right shows the following values:

Register	Value
ax	0003
bx	0002
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	0100
ds	0887
es	0869
ss	0888
cs	0879
ip	00CF

The status bar at the bottom shows the following values:

Register	Value
ss:0102	8B52
ss:0100	0054

Ответ: $3_{16} = 3_{10}$

3.3.2 Отладка ветки №2

Тест для ветки

«Результат = найти 2 наибольших среди a,d,c,d и
перемножить»

a = 1 b = 2 c = 3 d = 4

The screenshot shows the DOSBox 0.74 interface with the CPU window open. The assembly code is displayed in a list, and the registers are shown on the right. The code is as follows:

Address	Hex	Assembly
cs:00BE	A10000	mov ax,[0000]
cs:00C1	F7260200	mul word ptr [0002]
cs:00C5	EB0B	jmp 00D2
cs:00C7	90	nop
cs:00C8	A10000	mov ax,[0000]
cs:00CB	33060200	xor ax,[0002]
cs:00CF	EB01	jmp 00D2
cs:00D1	90	nop
cs:00D2	B44C	mov ah,4C
cs:00D4	CD21	int 21
cs:00D6	0000	add [bx+si],al
cs:00D8	0000	add [bx+si],al
cs:00DA	0000	add [bx+si],al

The registers are shown on the right:

Register	Value
ax	000C
bx	0003
cx	0000
dx	0000
si	0000
di	0000
bp	0000
sp	0100
ds	0887
es	0869
ss	0888
cs	0879
ip	00C5

The status bar at the bottom shows the following keyboard shortcuts: F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Ответ: $C_{16} = 12_{10}$

3.3.3 Отладка ветки №3

Тест для ветки

«Результат = $a * b + c/d$ »

$a = 8$ $b = 4$ $c = 5$ $d = 2$

The screenshot shows the DOSBox 0.74 interface with the assembly window open. The assembly window displays the following code:

```
cs:004F 8BD8      mov     bx,ax
cs:0051 33D2      xor     dx,dx
cs:0053 A10400      mov     ax,[0004]
cs:0056 F7360600   div     word ptr [0006]
cs:005A 03C3      add     ax,bx
cs:005C EB74      jmp     00D2 ↓
cs:005E 90        nop
cs:005F A10400      mov     ax,[0004]
cs:0062 F7260600   mul     word ptr [0006]
cs:0066 8BD8      mov     bx,ax
cs:0068 A10000      mov     ax,[0000]
cs:006B 33D2      xor     dx,dx
cs:006D F7360200   div     word ptr [0002]
```

The registers window on the right shows the following values:

ax	0022	c=0
bx	0020	z=0
cx	0000	s=0
dx	0001	o=0
si	0000	p=1
di	0000	a=0
bp	0000	i=1
sp	0100	d=0
ds	0887	
es	0869	
ss	0888	
cs	0879	
ip	005C	

The status bar at the bottom shows the following function key shortcuts:

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Ответ: $22_{16} = 34_{10}$

4. Выводы

На практике я изучил и опробовал, в соответствии с поставленной задачей: регистры процессора 8086, логику команд MOV, ADD, SUB, MUL, DIV, логические операции AND, OR, XOR, NOT, работу команды CMP, использование меток, логику работы команд условного и безусловного переходов JMP, JE, JNE, JC, JNC, размещение данных в сегменте данных, размерность данных: DB, DW, DD.

В процессе выполнения лабораторной работы я освоил DosBox. С помощью DosBox я отслеживал ход выполнения работы программы.

Литература

1. Юров В.И. – «Assembler. Учебник для вузов. 2-ое издание, 2003 год».
2. Юров В.И. – «Assembler. Практикум. 2-ое издание, 2006 год».
3. Калашников О.А. – «Ассемблер - это просто. 2-ое издание, 2011 год».

Исходный код

```
model small
.stack 100h
.data
a dw 8
b dw 4
c dw 5
d dw 2

.code

Swap_a_b proc
    mov ax, a
    mov bx, b
    mov a, bx
    mov b, ax
    ret
Swap_a_b endp

Swap_b_c proc
    mov ax, b
    mov bx, c
    mov b, bx
    mov c, ax
    ret
Swap_b_c endp

Swap_c_d proc
    mov ax, c
    mov bx, d
    mov c, bx
    mov d, ax
    ret
Swap_c_d endp

Start:
    mov ax, @data
    mov ds, ax

; calculate b^3 and move to bx
    mov ax, b
    mul b
    mul b
    mov bx, ax
```

; calculate a^2 and move to ax

mov ax, a

mul ax

; if $a^2 \neq b^3$ jump to Cond2

cmp ax, bx

jnz Cond2

*; else $res = a*b + c/d$*

*; calculate $a*b$ and save to bx*

mov ax, a

mul b

mov bx, ax

; calculate c/d and save to ax

xor dx, dx

mov ax, c

div d

*; calculate $a*b + c/d$ and save to ax, finish the
program*

add ax, bx

jmp Final

Cond2:

*; calculate $c*d$ and save to bx*

mov ax, c

mul d

mov bx, ax

; calculate a/b and save to ax

mov ax, a

xor dx, dx

div b

*; if $c*d == a/b$ jump to Equals*

cmp ax, bx

jz Equals

*; else choose two biggest from a, b, c, d and multiply
; choose two biggest from a, b, c, d (bubble sort until
a and b contain biggest numbers)*

mov ax, c

```
    cmp ax, d
    jc Swap_1
Return_1:
    mov ax, b
    cmp ax, c
    jc Swap_2
Return_2:
    mov ax, a
    cmp ax, b
    jc Swap_3
Return_3:
    mov ax, c
    cmp ax, d
    jc Swap_4
Return_4:
    mov ax, b
    cmp ax, c
    jc Swap_5
Return_5:
    jmp End_sort
```

```
Swap_1:
    call Swap_c_d
    jmp Return_1
```

```
Swap_2:
    call Swap_b_c
    jmp Return_2
```

```
Swap_3:
    call Swap_a_b
    jmp Return_3
```

```
Swap_4:
    call Swap_c_d
    jmp Return_4
```

```
Swap_5:
    call Swap_b_c
    jmp Return_5
```

```
End_sort:
    ; multiply two greater numbers and save to ax, finish
    the program
    mov ax, a
```

```
mul b
```

```
jmp Final
```

```
Equals:
```

```
; calculate a xor b and move to ax, finish the program
```

```
mov ax, a
```

```
xor ax, b
```

```
jmp Final
```

```
Final:
```

```
mov ah, 4Ch
```

```
int 21h
```

```
end Start
```