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

Отчет по лабораторной работе №1

Основные конструкции. Написание простейшей программы

Выполнил: Студент гр. 053502
Герчик Артём Вадимович

Руководитель: ст. преподаватель
Шиманский В.В.

Минск 2021

СОДЕРЖАНИЕ

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.

8) Назначение флагов CF и ZF.

9) Использование меток.

10) Размещение данных в сегменте данных. Размерность данных: DB, DW, DD. Работа с переменными, определенными в сегменте данных.

11) Компилирование, линковка, выполнение и отладка ассемблерных программ.

2. Постановка задачи

2.1.Текст задания

```
Если a * (c + b) * (d ^ 2) = (a – d) * (b + c) то
   Если a > b ^ 2 то
     Результат = c ^ 2 /(d-c)-d^2
   Иначе
     Если a < c + d то
       Результат = d^2 + (b OR c)
     Иначе
       Результат = a + (b AND c)
```

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. Выполнение программы необходимо показать в Отладчике.

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

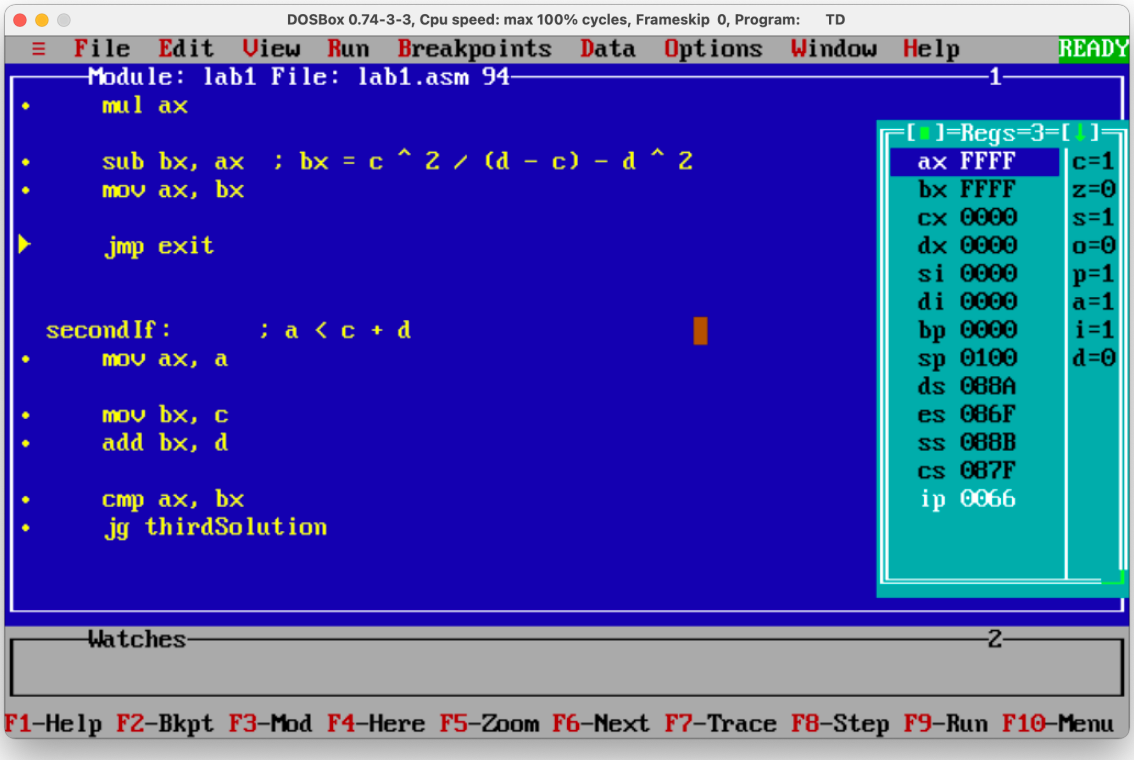
3.3. Примеры:

3.3.1 Отладка ветки №1

Тест для ветки

« Результат = c ^ 2 /(d - c) - d ^ 2 »

a = 1 b = 0 c = 0 d = 1



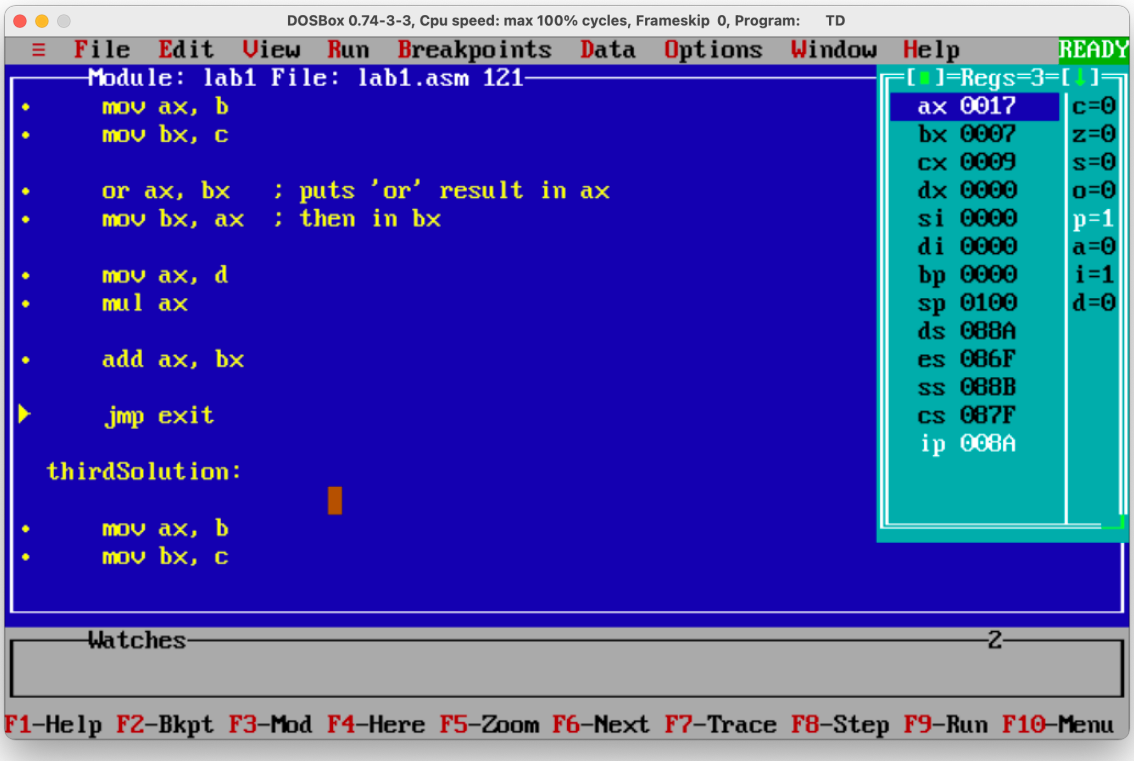
Ответ: $FFFF_{16} = -1_{10}$

3.3.2 Отладка ветки №2

Тест для ветки

« Результат = d ^ 2 + (b OR c) »

a = 5 b = 2 c = 7 d = 4



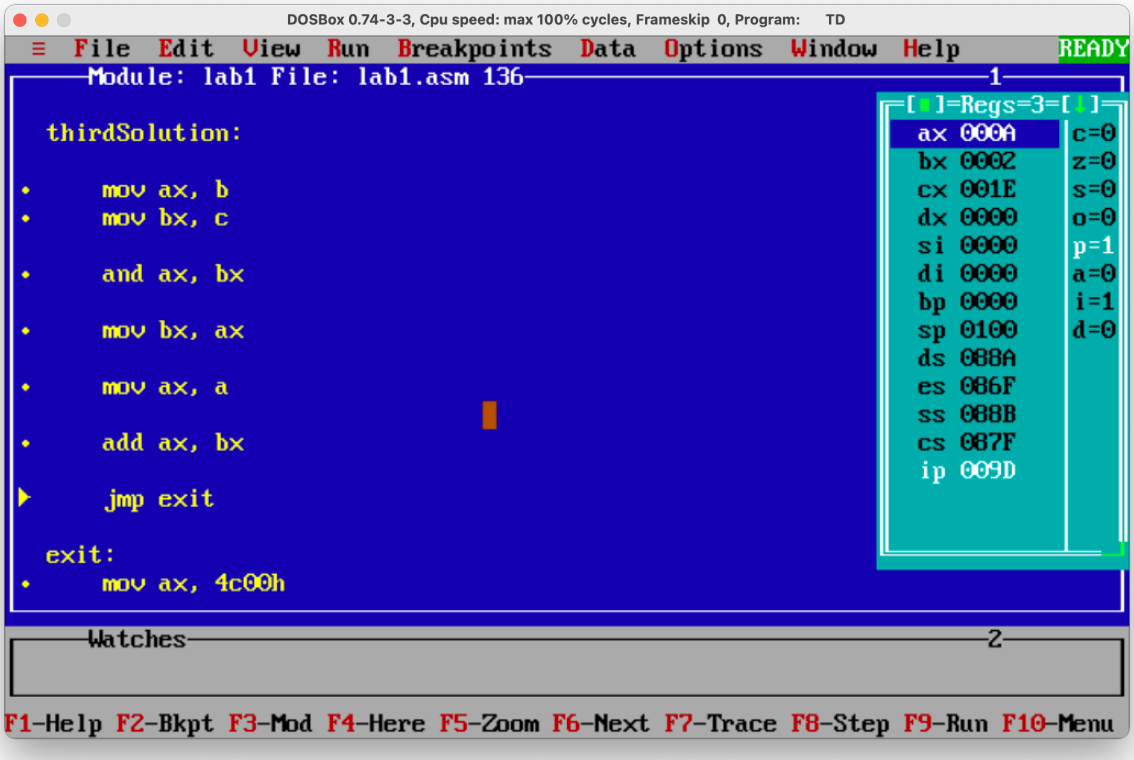
Ответ: $0017_{16} = 23_{10}$

3.3.3 Отладка ветки №3

Тест для ветки

« Результат = a + (b AND c) »

a = 8 b = 3 c = 2 d = 2



Ответ: $000A_{16} = 10_{10}$

4. Выводы

На практике было изучено и опробовано, в соответствии с поставленной задачей: регистры процессора 8086, логика команд MOV, ADD, SUB, MUL, DIV, логические операции AND, OR, XOR, NOT, работа команды CMP, использование меток, логика 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 мной были освоены DosBox, TurboDebuger. С помощью TurboDebuger вы можете отслеживать 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».

Приложение

```
.model small
.stack 256

.data
a dw 1
b dw 0
c dw 0
d dw 1

.code

main:

    mov ax, @data
    mov ds, ax

leftPart:

    mov ax, a

    mov bx, b
    add bx, c

    mul bx    ; ax = ax * bx
    mov bx, ax

    mov ax, d
    mul ax    ; ax = ax ^ 2

    mul bx    ; ax = ax * bx

    mov bx, ax ; bx = a * (c + b) * (d ^ 2) leftPart
    mov ax, 0

rightPart:

    mov ax, a
    sub ax, d

    mov cx, b
    add cx, c

    mul cx     ; ax = ax * cx
    mov cx, ax ; cx = (a – d) * (b + c) rightPart
```

```
    mov ax, 0

compareLeftAndRight:

    cmp cx, bx
    jne secondIf


firstIf:       ; a > b ^ 2

    mov bx, a

    mov ax, b
    mul ax     ; ax = ax ^ 2

    cmp bx, ax ; bx = a, ax = b ^ 2
    jl secondIf


firstSolution:  ; c ^ 2 / (d - c) - d ^ 2

    mov ax, c
    mul ax      ; ax = c ^ 2

    mov bx, d
    sub bx, c   ; bx = d - c

    cmp bx, 0
    je exit     ; if denominator = 0 => exit (exception)

    div bx      ; ax = ax / bx

    mov bx, ax  ; bx = c ^ 2 / (d - c)

    mov ax, d
    mul ax

    sub bx, ax  ; bx = c ^ 2 / (d - c) - d ^ 2
    mov ax, bx

    jmp exit


secondIf:      ; a < c + d
    mov ax, a
```

```
    mov bx, c
    add bx, d

    cmp ax, bx
    jg thirdSolution



secondSolution: ; d ^ 2 + (b OR c)

    mov ax, b
    mov bx, c

    or ax, bx    ; puts 'or' result in ax
    mov bx, ax   ; then in bx

    mov ax, d
    mul ax

    add ax, bx

    jmp exit

thirdSolution:   ; a + (b AND c)

    mov ax, b
    mov bx, c

    and ax, bx

    mov bx, ax

    mov ax, a

    add ax, bx

    jmp exit

exit:
    mov ax, 4c00h
    mov al, 0
    int 21h

end main
```