МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе№3

по дисциплине «Организация ЭВМ и систем»

Тема: Представление и обработка целых чисел. Организация ветвящихся процессов.

Студент гр. 1303 Беззубов Д.В.

Преподаватель Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров вычисляет значения функций.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

a) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

b) значения результирующей функции res = f3(i1,i2,k),

где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1,n2,n3), приведенным в табл.4.

Значения a, b, i, k являются исходными данными, которые должны выбираться студентом самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть всевозможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы, а также различные знаки параметров a и b.

$$f1 = \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a<=b \end{cases}$$

$$f4 = \begin{cases} -(6*i - 4), & \text{при } a>b \\ 3*(i+2), & \text{при } a<=b \end{cases}$$

$$f3 = \begin{cases} |i1 + i2|, & \text{при } k=0 \\ \min(i1,i2), & \text{при } k/=0 \end{cases}$$

Выполнение работы

Были реализованы функции из Каталога Заданий, соответствующие 3 Варианту. Реализованная программа протранслирована с различными тестовыми данными.

Для выполнения данного задания были использованы такие команды общего назначения как:

Команды передачи данных.

1) *mov* – присваивание

Двоичные арифметические команды.

1) *add* - сложение
2) *sub* - вычитание
3) *cmp* – сравнение
4) *neg* – смена знака

Команды побитового сдвига.

1) *sal* - арифметический сдвиг влево

Команды передачи управления.

1) *jmp* – команда безусловного перехода
2) *Int* - вызов программного прерывания
3) *jg(jump greater)* - выполняет короткий переход, если первый операнд больше второго операнда при выполнении операции сравнения с помощью команды cmp.
4) *Jne(jump negative equal)* - выполняет короткий переход, если первый операнд не равен второму операнду при выполнении операции сравнения с помощью команды cmp.
5) *jl(jump less)* - выполняет короткий переход, если первый операнд меньше второго операнда при выполнении операции сравнения с помощью команды cmp.

Для реализации ветвления в программе использовались метки. Метка - это символьное имя, обозначающее ячейку памяти, которая содержит некоторую команду.

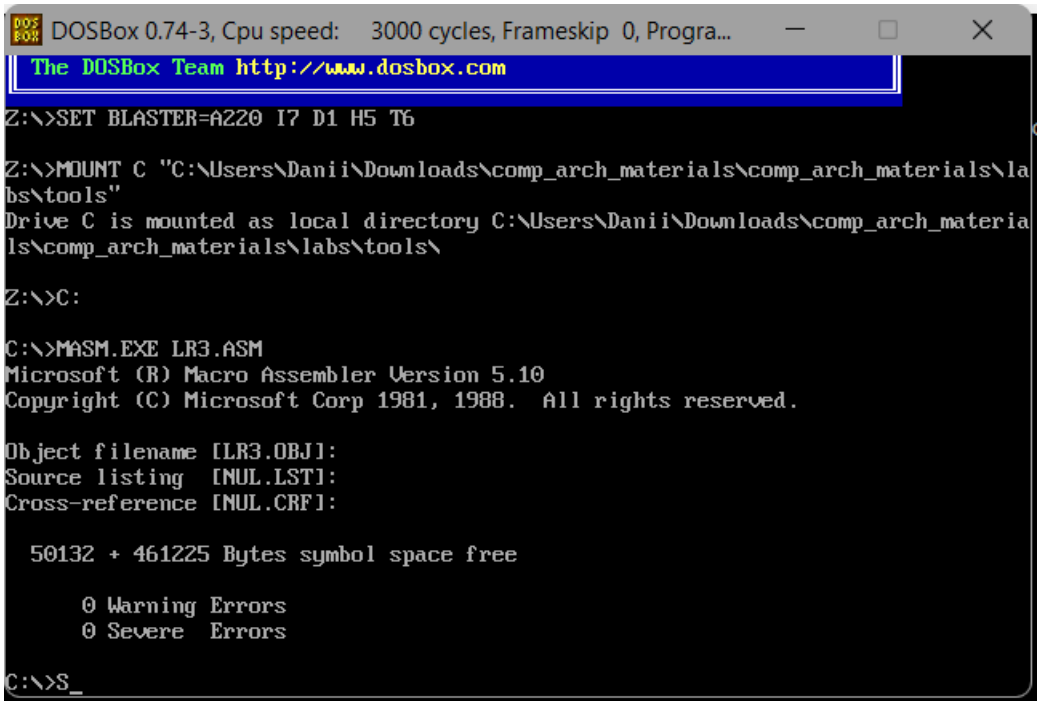

рис.1 -Трансляция программы

Тестирование

Программа выполнена в пошаговом режиме под управлением отладчика с фиксацией значений используемых переменных.

№ теста	Тестируемый случай	Функции для данного случая	Данные	
			входные	выходные
1	a > b k = 0	f1 = 15 – 2*i f2 = -(6*i – 4) f3 = abs(f1 + f2)	a = 1, b = 0 k = 0 i = 1	f1 = 13 = 000D f2 = -2 = FFFE f3 = 11 = 000B
2	a > b k != 0	f1 = 15 – 2*i f2 = -(6*i – 4) f3 = min(f1, f2)	a = 1, b = 0 k = 1 i = 1	f1 = 13 = 000D f2 = -2 = FFFE f3 = -2 = FFFE
3	a <= b k = 0	f1 = 3*i + 4 f2 = 3*(i + 2) f3 = abs(f1 + f2)	a = 1, b = 1 k = 0 i = 1	f1 = 7 = 0007 f2 = 9 = 0009 f3 = 16 = 0010
4	a <= b	f1 = 3*i + 4	a = 1, b = 1	f1 = 7 = 0007

	k != 0	f2 = 3*(i + 2) f3 = min(f1, f2)	k = 1 i = 1	f2 = 9 = 0009 f3 = 7 = 0007

Выводы

В ходе выполнения лабораторной работы были получены навыки разработки программы с заданными целочисленными значениями на языке программирования Ассемблер.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: source.asm

```
ASSUME CS:CODE, SS: AStack, DS: DATA

AStack  SEGMENT STACK
        DW 12 DUP('!')
AStack  ENDS

DATA    SEGMENT

a   DW  0
b   DW  0
i   DW  0
k   DW  0

i1  DW  0
i2  DW  0
res DW  0

DATA    ENDS

CODE    SEGMENT

Main    PROC    FAR

    push DS
    sub AX,AX
    push AX

    mov AX, DATA
    mov DS, AX

    ;Вычисление f1 и f2

    mov AX, a
    mov CX, i
    cmp AX, b
```

```
    jg A_Greater

    sal CX, 1 ; i<<1 = i*2
    add CX, i ; i*2+i = i*3
    add CX, 4 ; i*3 + 4
    mov i1, CX

    mov CX, i
    add CX, 2 ;i+2
    mov AX, CX ;помещаем i+2 в ax
    sal CX, 1 ; (i+2)*2
    add CX, AX ; (i+2)*2 + (i+2)
    mov i2, CX

    jmp FUNCTION_3

A_Greater:

    mov AX, 15
    sal CX, 1
    sub AX, CX
    mov i1, AX

    mov CX, i
    mov AX, 4
    sal CX, 1 ;i*2
    add CX, i ;i*2 + i = 3i
    sal CX, 1 ; 3i*2
    sub AX, CX
    mov i2, AX

FUNCTION_3:

    mov AX, k
    cmp AX, 0
    JNe K_NOT_EQUAL_ZERO

    mov AX, i1
    add AX, i2
```

```
    cmp AX, 0
    jl SUM_LESS_ZERO
    mov res, AX
    jmp QUIT

SUM_LESS_ZERO:

    neg AX
    mov res, AX
    jmp QUIT

K_NOT_EQUAL_ZERO:

    mov AX, i1
    mov CX, i2
    cmp AX, i2
    jl  i1_LESS_i2
    mov res, CX
    jmp QUIT

i1_LESS_i2:

    mov res, AX

QUIT:

    int 20h

Main    ENDP
CODE    ENDS
        END Main
```