**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.**

Студентка гр. 9383 _____ Лысова А.М,

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

**Цель работы.**

Познакомиться с представлением и обработкой целых чисел. Создать программу на языке Ассемблер, реализовывающую ветвящийся процесс.

**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.

Вариант 10.

$$f2 = \begin{cases} -(4*i+3), & \text{при } a>b \\ 6*i-10, & \text{при } a<=b \end{cases}$$

$$f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases}$$

$$f6 = \begin{cases} |i1 - i2|, & \text{при } k<0 \\ \max(7, |i2|), & \text{при } k>=0 \end{cases}$$

**Ход работы.**

В ходе работы была реализована программа на языке Ассемблер, которая по заданным целочисленным параметрам вычисляет значения функций.

Исходные данные заносятся в программу до выполнения, а выходные данные отслеживаются через отладчик.

Также были организованы ветвящиеся процессы, для которых использовались:

- cmp — сравнение с изменением флага ZF (0 — аргументы не равны, 1 — аргументы равны)
- jmp — безусловный переход по заданной метке (без изменения флагов)
- jl — условный переход по заданной метке, в случае если первый аргумент меньше второго после выполнения cmp.
- jg - условный переход по заданной метке, в случае если первый аргумент больше второго после выполнения cmp.

  Также использовался логический сдвиг влево (shl), он позволяет умножать значение на 2.

**Тестирование.**

1. a = 1, b = 2, i = 3, k = 4 => f1 = 8, f2 = -12, f3 = 12
2. a = 1, b = 2, i = 3, k = -3 => f1 = 8, f2 = -12, f3 = 20
3. a = 2, b = 1, i = 3, k = 4 => f1 = -15, f2 = 8, f3 = 8
4. a = 2, b = 1, i = 3, k = -3 => f1 = -15, f2 = 8, f3 = 23

**Выводы.**

Была реализована программа на языке Ассемблер с ветвящимися процессами. Изучены представление и обработка целых чисел.

## ПРИЛОЖЕНИЕ А.

## ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: lb3.asm

```
AStack SEGMENT STACK
        DW 12 DUP(?)
AStack ENDS

DATA SEGMENT
a       DW    1
b       DW    2
i       DW    3
k       DW    4
i1      DW    ?
i2      DW    ?
res     DW    ?
DATA ENDS

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

Main PROC FAR
        mov ax, DATA
        mov ds, ax

f1:
        mov ax, a
        cmp ax, b
        jg f1_1         ;if a>b
                        ;a<=b
        mov ax, i
        shl ax, 1       ;ax = 2*ax
        mov bx, ax      ;bx = 2*ax
        shl ax, 1       ;ax = 4*ax
        add ax, bx      ;ax = 6*ax
        sub ax, 10      ;ax = ax - 10
        mov i1, ax

        jmp f2_1
f1_1:
        mov ax, i
        shl ax, 1
        shl ax, 1
        add ax, 3
        neg ax
        mov i1, ax

f2:
        add ax, 23      ;ax = ax + 23
```

```
        mov i2, ax
        jmp f3

f2_1:
        neg ax          ;ax = -ax
        sub ax, 4       ;ax = ax - 4

        mov i2, ax

f3:
        mov ax, k
        cmp k, 0
        jl f3_1         ;if k < 0

        mov ax, i2
        cmp ax, 0       ;if ax < 0
        jl f_abs        ;then |ax|

        jmp f3_cmp_7

f3_1:
        mov ax, i1      ;ax = i1
        sub ax, i2      ;ax = i1 - i2
        cmp ax, 0       ;if ax < 0
        jl f_abs_1      ;then ax = |ax|

        jmp f3_res
f_abs:
        neg ax          ;ax = -ax

f3_cmp_7:
        cmp ax, 7       ;if ax < 7
        jl f3_7         ;res = 7

f3_res:
        mov res, ax     ;else res = ax

        jmp f_end

f_abs_1:
        neg ax          ;ax = |ax|

        jmp f3_res

f3_7:
        mov res, 7      ;res = 7

f_end:
        mov ah, 4ch
        int 21h
```

```
Main    ENDP
CODE ENDS
        END Main
```

## ПРИЛОЖЕНИЕ Б.
## ЛИСТИНГ ПРОГРАММЫ.

Название файла: lb3.lst

```
#Microsoft (R) Macro Assembler Version 5.10              10/22/20 10:13:4
                                                             Page     1-1


0000                            AStack SEGMENT STACK
0000  000C[                            DW 12 DUP(?)
        ????
                                ]

0018                            AStack ENDS

0000                            DATA SEGMENT
0000  0001                      a       DW    1
0002  0002                      b       DW    2
0004  0003                      i       DW    3
0006  0004                      k       DW    4
0008  0000                      i1      DW    ?
000A  0000                      i2      DW    ?
000C  0000                      res     DW    ?
000E                            DATA ENDS

0000                            CODE SEGMENT
                                        ASSUME CS:CODE, DS:DATA, SS:AStack

0000                            Main PROC FAR
0000  B8 ---- R                        mov ax, DATA
0003  8E D8                            mov ds, ax

0005                            f1:
0005  A1 0000 R                        mov ax, a
0008  3B 06 0002 R                     cmp ax, b
000C  7F 14                            jg f1_1         ;if a>b
                                                       ;a<=b
000E  A1 0004 R                        mov ax, i
0011  D1 E0                            shl ax, 1       ;ax = 2*ax
0013  8B D8                            mov bx, ax      ;bx = 2*ax
0015  D1 E0                            shl ax, 1       ;ax = 4*ax
0017  03 C3                            add ax, bx      ;ax = 6*ax
0019  2D 000A                                  sub ax, 10      ;ax = ax - 10
001C  A3 0008 R                        mov i1, ax

001F  EB 19 90                                 jmp f2_1
0022                            f1_1:
0022  A1 0004 R                        mov ax, i
0025  D1 E0                            shl ax, 1
0027  D1 E0                            shl ax, 1
0029  05 0003                                  add ax, 3
```

```
002C  F7 D8                     neg ax
002E  A3 0008 R                 mov i1, ax

0031                    f2:
0031  05 0017                       add ax, 23      ;ax = ax + 23

0034  A3 000A R                 mov i2, ax
0037  EB 09 90                      jmp f3

003A                    f2_1:
003A  F7 D8                     neg ax          ;ax = -ax
```



```
003C  2D 0004                       sub ax, 4       ;ax = ax - 4

003F  A3 000A R                 mov i2, ax

0042                    f3:
0042  A1 0006 R                 mov ax, k
0045  83 3E 0006 R 00               cmp k, 0
004A  7C 0B                     jl f3_1         ;if k < 0

004C  A1 000A R                 mov ax, i2
004F  3D 0000                       cmp ax, 0       ;if ax < 0
0052  7C 12                     jl f_abs        ;then |ax|

0054  EB 12 90                      jmp f3_cmp_7

0057                    f3_1:
0057  A1 0008 R                 mov ax, i1      ;ax = i1
005A  2B 06 000A R              sub ax, i2      ;ax = i1 - i2
005E  3D 0000                       cmp ax, 0       ;if ax < 0
0061  7C 10                     jl f_abs_1      ;then ax = |ax|

0063  EB 08 90                      jmp f3_res
0066                    f_abs:
0066  F7 D8                     neg ax          ;ax = -ax

0068                    f3_cmp_7:
0068  3D 0007                       cmp ax, 7       ;if ax < 7
006B  7C 0A                     jl f3_7         ;res = 7

006D                    f3_res:
006D  A3 000C R                 mov res, ax     ;else res = ax

0070  EB 0B 90                      jmp f_end

0073                    f_abs_1:
0073  F7 D8                     neg ax          ;ax = |ax|
```

```
0075  EB F6                     jmp f3_res

0077                        f3_7:
0077  C7 06 000C R 0007          mov res, 7      ;res = 7

007D                        f_end:
007D  B4 4C                     mov ah, 4ch
007F  CD 21                     int 21h

0081                        Main   ENDP
0081                        CODE ENDS
                                END Main
#Microsoft (R) Macro Assembler Version 5.10                 10/22/20 10:13:4
                                                             Symbols-1


Segments and Groups:

                N a m e                 Length  Align  Combine Class

ASTACK . . . . . . . . . . . .  0018    PARA  STACK
CODE . . . . . . . . . . . . .  0081    PARA  NONE
DATA . . . . . . . . . . . . .  000E    PARA  NONE

Symbols:

                N a m e                 Type     Value   Attr

A  . . . . . . . . . . . . . .  L WORD       0000    DATA

B  . . . . . . . . . . . . . .  L WORD       0002    DATA

F1 . . . . . . . . . . . . . .  L NEAR       0005    CODE
F1_1 . . . . . . . . . . . . .  L NEAR       0022    CODE
F2 . . . . . . . . . . . . . .  L NEAR       0031    CODE
F2_1 . . . . . . . . . . . . .  L NEAR       003A    CODE
F3 . . . . . . . . . . . . . .  L NEAR       0042    CODE
F3_1 . . . . . . . . . . . . .  L NEAR       0057    CODE
F3_7 . . . . . . . . . . . . .  L NEAR       0077    CODE
F3_CMP_7 . . . . . . . . . . .  L NEAR       0068    CODE
F3_RES . . . . . . . . . . . .  L NEAR       006D    CODE
F_ABS  . . . . . . . . . . . .  L NEAR       0066    CODE
F_ABS_1  . . . . . . . . . . .  L NEAR       0073    CODE
F_END  . . . . . . . . . . . .  L NEAR       007D    CODE

I  . . . . . . . . . . . . . .  L WORD       0004    DATA
I1 . . . . . . . . . . . . . .  L WORD       0008    DATA
I2 . . . . . . . . . . . . . .  L WORD       000A    DATA

K  . . . . . . . . . . . . . .  L WORD       0006    DATA
```

```
MAIN . . . . . .        F PROC        0000    CODE Length = 0081

RES  . . . . . .        L WORD        000C    DATA

@CPU . . . . . .        TEXT  0101h
@FILENAME  . . . . . .          TEXT  test
@VERSION . . . . . .    TEXT  510

#Microsoft (R) Macro Assembler Version 5.10                 10/22/20 10:13:4

                                                            Symbols-2


    100 Source  Lines
    100 Total   Lines
     28 Symbols

  48040 + 459220 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```