

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №3**  
**по дисциплине «Организация ЭВМ и систем»**  
**Тема: Представление и обработка целых чисел. Организация ветвящихся**  
**процессов**

Студент гр. 9383

\_\_\_\_\_

Поплавский И.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Изучение представления и обработки целых чисел на языке Ассемблер.

### **Постановка задачи.**

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров  $a, b, i, k$  вычисляет: а) значения функций  $i1 = f1(a,b,i)$  и  $i2 = f2(a,b,i)$ ; б) 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 16. Функции  $f1 = 3, f2 = 6, f3 = 4$

$$f3 = \begin{cases} / 7 - 4*i, & \text{при } a > b \\ \backslash 8 - 6*i, & \text{при } a \leq b \end{cases}$$

**Рисунок 1 – Функция f3**

$$f6 = \begin{cases} / 2*(i+1) - 4, & \text{при } a > b \\ \backslash 5 - 3*(i+1), & \text{при } a \leq b \end{cases}$$

**Рисунок 2 – Функция f6**

$$f4 = \begin{cases} / \min(|i1 - i2|, 2), & \text{при } k < 0 \\ \backslash \max(-6, -i2), & \text{при } k \geq 0 \end{cases}$$

**Рисунок 3 – Функция f4**

### **Выполнение работы.**

Сначала происходит инициализация дата сегмента, после программа идет по меткам и вычисляет функции, при реализации были использованы следующие функции:

Mov – перемещает значения в регистр

Cmp, jg – сравнение двух регистров, если первые больше второго, то происходит переход к метке указанной в jg

Jmp – перемещение к метке

Shl – побитовый сдвиг, используется для умножения

Sub – отнять у левого регистра правый

Add – добавить левому регистру правый

Neg – переводит число в обратное ему по модулю

### **Вывод.**

В результате выполнения лабораторной работы были изучены различные виды обработки целых чисел на языке Ассемблер

## ПРИЛОЖЕНИЕ А

### РАЗРАБОТАННЫЙ КОД ПРОГРАММЫ

AStack SEGMENT STACK

DW 32 DUP(?)

AStack ENDS

DATA SEGMENT

A DW 5

B DW 1

I DW 3

K DW -5

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:

cmp A, B

jg f1\_1

mov bx, I

mov ax, I

shl bx, 1

shl bx, 1

shl bx, 1

shl ax, 1

sub bx, ax

mov ax, 8

sub ax, bx

mov I1, ax

jmp f2

f1\_1:

mov bx, I

shl bx, 1

shl bx, 1

mov ax, 7

```
sub ax, bx
mov I1, ax
jmp f2
```

```
f2:
    cmp A, B
    jg f2_1
    shr ax, 1
    sub ax, 2
    mov I2, ax
    jmp f3
```

```
f2_1:
    mov ax, I
    shl ax, 1
    mov bx, 2
    sub ax, bx
    mov I2, ax
    jmp f3
```

```
f3:
    cmp K, 0
    jg f3_1
    mov ax, I1
    mov bx, I2
    sub ax, bx
    mov bx, 0
    cmp bx, ax
    jg f3_2_neg
    jmp f3_2_1
```

```
f3_1:
    mov ax, I2
    neg ax
    mov bx, 6
    neg bx
    cmp ax, bx
    jg f3_1_1
    mov RES, bx
    jmp f_end
```

```
f3_1_1:
    mov RES, ax
    jmp f_end
```

```

f3_2_neg:
    neg ax
    jmp f3_2_1

f3_2_1:
    mov bx, 2
    cmp ax, bx
    jg f3_2_1_a
    mov RES, ax
    jmp f_end

f3_2_1_a:
    mov RES, bx
    jmp f_end

f_end:
    mov ah, 4ch
    int 21h

```

```

Main    ENDP
CODE    ENDS
END Main

```

```

Main    ENDP
CODE    ENDS
END Main

```