

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

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

Студент гр. 1383

\_\_\_\_\_

Ковалев П. А.

Преподаватель

\_\_\_\_\_

Ефремов М. А.

Санкт-Петербург  
2022

## Цель работы

Изучение организации ветвлений в программах на языке Ассемблера.

## Задание

Вариант 2. ( $f_1$ ,  $f_3$ ,  $f_2$ )

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

- а) значения функций  $i_1 = f_1(a, b, i)$  и  $i_2 = f_2(a, b, i)$ ;
- б) значения результирующей функции  $res = f_3(i_1, i_2, k)$ ,

где вид функций  $f_1$  и  $f_2$  определяется из табл. 2, а функции  $f_3$  - из табл.3 по цифрам шифра индивидуального задания ( $n_1, n_2, n_3$ ), 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  $b$ .

## Выполнение работы

1. В формуле  $15 - 2i = -2i + 15$  упрощений не требуется;
2. Формулу  $3 * i + 4$  можно представить, как  $i + 2 * i + 4$ ;
3. В формуле  $7 - 4 * i$  упрощений не требуется;
4. Формулу  $8 - 6 * i$  можно представить, как  $-(4 * i + 2 * i) + 8$ .

По результатам преобразований можно заранее подсчитать  $2 * i$ .

После предварительных вычислений программа сравнивает  $a$  и  $b$ . В зависимости от результата сравнения программа вычисляет функции  $f_1$  и  $f_3$ . При  $a > b$  в формулах дублируется  $-2 * i$ , что можно заранее подсчитать. В функции  $f_2$  при любых  $k$  используется только  $-i * 2$ , операцию отрицания можно выполнить перед сравнением  $k$  и 0.

Результаты выполнения функций записываются в ячейки памяти [f1], [f3], [f2]. В таблице (1) представлен протокол тестирования программы.

Таблица 1: Результаты тестирования

№	Исходные данные	Вывод программы	Комментарий
1.	$a = 1, b = 2, i = 5, k = 3$	$29_{16} = 41_{10}$	[f2] = 29 00
2.	$a = 2, b = 1, i = 5, k = 3$	$12_{16} = 18_{10}$	[f2] = 12 00
3.	$a = 1, b = 2, i = 5, k = -1$	$20_{16} = 32_{10}$	[f2] = 20 00
4.	$a = 2, b = 1, i = 5, k = -1$	$17_{16} = 23_{10}$	[f2] = 17 00
5.	$a = 1, b = 1, i = 5, k = -1$	$20_{16} = 32_{10}$	[f2] = 20 00
6.	$a = 1, b = 1, i = 5, k = 3$	$29_{16} = 41_{10}$	[f2] = 29 00
7.	$a = 1, b = 2, i = -5, k = 2$	$31_{16} = 49_{10}$	[f2] = 31 00
8.	$a = 2, b = 1, i = -5, k = 2$	$2_{16} = 2_{10}$	[f2] = 02 00
9.	$a = 1, b = -2, i = 5, k = 3$	$12_{16} = 18_{10}$	[f2] = 12 00
10.	$a = -1, b = -2, i = -5, k = -3$	$19_{16} = 25_{10}$	[f2] = 19 00
11.	$a = 1, b = 2, i = -5, k = -3$	$B_{16} = 11_{10}$	[f2] = FF F5
12.	$a = 1, b = 2, i = 1, k = 3$	$5_{16} = 5_{10}$	[f2] = 05 00

## Вывод

В результате выполнения работы было изучена организация ветвлений на языке Ассемблера и написана эффективная программа, использующая изученный функционал языка.

# Приложение А

## Исходный код программы

Название файла: lr3.asm

DOSSEG

.MODEL small

```
;;          / 15 - 2*i, при a>b
;; f1 = <
;;          \ 3*i + 4, при a<=b
;;
;;          / 7 - 4*i, при a>b
;; f3 = <
;;          \ 8 - 6*i, при a<=b
;;
;;          / max(i1, 10 - i2), при k<0
;; f2 = <
;;          \ |i1 - i2|, при k>=0
```

a EQU 1

b EQU 2

k EQU 3

i EQU 1

.STACK

.DATA

f1 dw 0

f3 dw 0

f2 dw 0

.CODE

mov ax, i ; ax = i

shl ax, 1 ; ax = 2\*i

mov cx, a

cmp cx, b

jle less1

neg ax ; ax = -2\*i

mov dx, ax ; dx = -2\*i

```

    shl dx, 1                ; dx = -4*i
    add ax, 15               ; ax = 15 - 2*i
    add dx, 7                ; dx = 7 - 4*i
    jmp finish1

less1:
    add ax, i                ; ax = 3*i
    mov dx, ax               ; dx = 3*i
    shl dx, 1                ; dx = 6*i
    add ax, 4                ; ax = 3*i + 4
    neg dx                   ; dx = -6*i
    add dx, 8                ; dx = 8 - 6*i

finish1:
    mov [f1], ax
    mov [f3], dx

    neg dx                   ; dx = -i2

    mov cx, k
    cmp cx, 0
    jge greater2

    add dx, 10                ; dx = 10 - i2
    cmp ax, dx
    jg finish2
    mov ax, dx
    jmp finish2

greater2:
    add ax, dx                ; ax = i1 - i2
    jns finish2
    neg ax

finish2:
    mov [f1], ax
END

```