

**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. 1381		Луценко Д. А.
Преподаватель		Ефремов М. А.

Санкт-Петербург

2022

## Цель работы.

Изучить представление и обработку целых чисел. Научиться организации ветвящихся процессов. Применить полученные знания на практике.

## Задание.

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  $res = f3(i1,i2,k)$ ;

Вариант 13:

$$\begin{aligned} f1 = & \begin{cases} / - (4*i+3) , & \text{при } a > b \\ \backslash 6*i - 10 , & \text{при } a \leq b \end{cases} \\ f2 = & \begin{cases} / - (6*i+8) , & \text{при } a > b \\ \backslash 9 - 3*(i-1) , & \text{при } a \leq b \end{cases} \\ f3 = & \begin{cases} / |i1 + i2| , & \text{при } k=0 \\ \backslash \min(i1,i2) , & \text{при } k \neq 0 \end{cases} \end{aligned}$$

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  $b$ .

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций  $f_1$  и  $f_2$  вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций  $f_1$  и  $f_2$  нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

### **Выполнение работы.**

Созданы три сегмента AStack, DATA, CODE - сегмент стека, сегмент кода и сегмент данных соответственно. В сегменте данных объявлены переменные  $a, b, i, k, i_1, i_2, res$ .

В сегменте кода находится процедура Main, в которой вычисляются значения данных в условии функций. Сначала в регистрах записывается значение  $a$ . Далее с помощью функции `cmp` происходит сравнение значений переменных  $a$  и  $b$ . Команда `jle` проверяет условие  $a \leq b$ , при его выполнении производится переход по указанному адресу (к метке `second`). Там происходит расчёт значений  $i_1$  и  $i_2$ . Расчёт делается при помощи `shl` (это логический сдвиг влево, что эквивалентно умножению на 2), так же используются `sub` (вычитания) и `add` (сложение) и ещё `neg` (меняет знак у числа). Если переход не был совершен, то начинается расчёт по метке `first`. В конце каждой ветки использована команда безусловного перехода (`jmp`) к метке `final`. В `final` мы при помощи сравнения (`cmp`) узнаём какой подсчёт результата нам нужен, если  $k = 0$ , то мы перейдём к метке `res_1` и выполним

вычисления при этом если знак будет отрицательный это вызовет метку `abs_1`, и она поменяет знак. Если мы не перешли в `res_1`, то мы переходим в `res_2` и в зависимости какое число меньше дальше вызываем либо `min_i1`, либо `min_i2`, которые записывают в `ax` либо `i1`, либо `i2`. И в конечном счете переходим к конечной метке `ex`, где в `res` записываем значение из `ax`.

### **Тестирование.**

<b>№</b>	<b>a</b>	<b>b</b>	<b>i</b>	<b>k</b>	<b>i1</b>	<b>i2</b>	<b>res</b>
<b>1</b>	<b>4</b>	<b>2</b>	<b>2</b>	<b>0</b>	<b>FFF5</b>	<b>FFEC</b>	<b>001F</b>
<b>2</b>	<b>4</b>	<b>2</b>	<b>-2</b>	<b>1</b>	<b>0005</b>	<b>0004</b>	<b>0004</b>
<b>3</b>	<b>1</b>	<b>2</b>	<b>-2</b>	<b>1</b>	<b>FFE6</b>	<b>12</b>	<b>FFE6</b>

### **Выводы.**

В ходе лабораторной работы были изучены представление и обработка целых чисел, организация ветвящихся процессов на языке Ассемблер.

## ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

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

DATA SEGMENT
    a DW 4
    b DW 2
    i DW 2
    k DW 0
    i1 DW ?
    i2 DW ?
    res DW ?
DATA ENDS

CODE SEGMENT

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

Main PROC FAR
    push ds
    sub ax, ax
    push ax
    mov ax, DATA
    mov ds, ax

    mov ax, a
    cmp ax, b
    jle second ; ( a<=b)

first: ; (a > b)
    mov ax, i ;i
    shl ax, 1 ;2i
    shl ax, 1 ;4i
    add ax, 3 ; (4i + 3)
    neg ax
    mov i1, ax
    mov ax, i
    shl ax, 1
    shl ax, 1 ;4i
    add ax, i ;5i
    add ax, i ;6i
    add ax, 8 ; (6i + 8)
    neg ax
    mov i2, ax
    jmp final
```

```

second:
    mov ax, i
    shl ax, 1 ;2i
    shl ax, 1 ;4i
    shl ax, 1 ;6i
    sub ax, 10
    mov i1, ax
    mov ax, i
    shl ax, 1 ;2i
    add ax, i ;3i
    neg ax
    add ax, 3
    add ax, 9
    mov i2, ax
    jmp final

abs_1:
    neg i1
    mov ax, i1
    jmp ex

final:
    cmp k, 0
    jne res_2 ;k != 0
res_1: ;k = 0
    mov ax, i2
    add i1, ax
    cmp i1, -1
    jle abs_1 ;if(i1 <= -1)
    jmp ex

res_2:
    mov cx, i2
    cmp i1, cx
    jl min_i1

min_i2:
    mov ax, i2
    jmp ex

min_i1:
    mov ax, i1

ex:
    mov res, ax
    ret

Main ENDP
CODE ENDS
END Main

```

## ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ

Название файла: lb3.lst

Microsoft (R) Macro Assembler Version 5.10  
23:49:1

10/30/22

Page 1-1

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

0018          AStack ENDS

0000          DATA SEGMENT
0000 0004          a DW 4
0002 0002          b DW 2
0004 0002          i DW 2
0006 0000          k DW 0
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 1E          push ds
0001 2B C0          sub ax,ax
0003 50          push ax
0004 B8 ---- R      mov ax, DATA
0007 8E D8          mov ds, ax

0009 A1 0000 R      mov ax, a
000C 3B 06 0002 R   cmp ax,b
0010 7E 29          jle second ; ( a<=b)

0012          first: ; (a > b)
0012 A1 0004 R      mov ax, i ;i
0015 D1 E0          shl ax, 1 ;2i
0017 D1 E0          shl ax, 1 ;4i
0019 05 0003        add ax, 3 ; (4i + 3)
001C F7 D8          neg ax
001E A3 0008 R      mov i1, ax
0021 A1 0004 R      mov ax, i
0024 D1 E0          shl ax, 1
```



```

0026 D1 E0                shl ax, 1 ;4i
0028 03 06 0004 R        add ax, i ;5i
002C 03 06 0004 R        add ax, i ;6i
0030 05 0008            add ax, 8 ;(6i + 8)
0033 F7 D8                neg ax
0035 A3 000A R          mov i2, ax
0038 EB 31 90            jmp final

```

```

003B                      second:
003B A1 0004 R          mov ax, i
003E D1 E0                shl ax, 1 ;2i
0040 D1 E0                shl ax, 1 ;4i
0042 D1 E0                shl ax, 1 ;6i

```

\_\_Microsoft (R) Macro Assembler Version 5.10  
23:49:1

10/30/22

Page 1-2

```

0044 2D 000A            sub ax, 10
0047 A3 0008 R          mov i1, ax
004A A1 0004 R          mov ax, i
004D D1 E0                shl ax, 1 ;2i
004F 03 06 0004 R        add ax, i ;3i
0053 F7 D8                neg ax
0055 05 0003            add ax, 3
0058 05 0009            add ax, 9
005B A3 000A R          mov i2, ax
005E EB 0B 90            jmp final

```

```

0061                      abs_1:
0061 F7 1E 0008 R        neg i1
0065 A1 0008 R          mov ax, i1
0068 EB 2C 90            jmp ex
006B                      final:
006B 83 3E 0006 R 00      cmp k,0
0070 75 11                jne res_2 ;k != 0
0072                      res_1: ;k = 0
0072 A1 000A R          mov ax, i2
0075 01 06 0008 R        add i1, ax
0079 83 3E 0008 R FF      cmp i1, -1
007E 7E E1                jle abs_1 ;if(i1 <= -1)
0080 EB 14 90            jmp ex
0083                      res_2:
0083 8B 0E 000A R          mov cx, i2
0087 39 0E 0008 R          cmp i1, cx
008B 7C 06                jl min_i1
008D                      min_i2:
008D A1 000A R          mov ax, i2
0090 EB 04 90            jmp ex
0093                      min_i1:

```

```

0093  A1 0008 R          mov ax, i1
0096                      ex:
0096  A3 000C R          mov res, ax
0099  CB                  ret

```

```

009A                      Main ENDP
009A                      CODE ENDS
                      END Main

```

Microsoft (R) Macro Assembler Version 5.10  
23:49:1

10/30/22

Symbols-1

# Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK . . . . .	0018	PARA	STACK
CODE . . . . .	009A	PARA	NONE
DATA . . . . .	000E	PARA	NONE

## Symbols:

N a m e	Type	Value	Attr
A . . . . .	L WORD	0000	DATA
ABS_1 . . . . .	L NEAR	0061	CODE
B . . . . .	L WORD	0002	DATA
EX . . . . .	L NEAR	0096	CODE
FINAL . . . . .	L NEAR	006B	CODE
FIRST . . . . .	L NEAR	0012	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 = 009A
MIN_I1 . . . . .	L NEAR	0093	CODE
MIN_I2 . . . . .	L NEAR	008D	CODE
RES . . . . .	L WORD	000C	DATA
RES_1 . . . . .	L NEAR	0072	CODE
RES_2 . . . . .	L NEAR	0083	CODE
SECOND . . . . .	L NEAR	003B	CODE

@CPU . . . . .	TEXT	0101h
@FILENAME . . . . .	TEXT	LB3
@VERSION . . . . .	TEXT	510

91 Source Lines

91 Total Lines

25 Symbols

47992 + 461315 Bytes symbol space free

0 Warning Errors

0 Severe Errors