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

# ОТЧЕТ

**по лабораторной работе №3**

**по дисциплине «ОргЭВМиС»**

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

Студент гр. 0382 _____ Афанасьев Н. С.

Преподаватели _____ Ефремов М.А.

Санкт-Петербург
2021

**Цель работы.**

Изучить представление и обработку целых чисел. Научиться организовывать ветвящиеся процессы на языке Ассемблера.

**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),

f1 = 15-2i , при a>b;

3i+4 , при a<=b

f2 = -(6i - 4) , при a>b;

3*(i+2) , при a<=b

f3 = |i1 + i2|, при k=0;

min(i1,i2), при k!=0

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.**

Для функций *f1* и *f2* условия одинаковы, поэтому их вычисление проходит в одном блоке. Сначала командой *cmp* сверяются значения *a* и *b*. С помощью команды *jle* проверяется, что *a <= b*, и в зависимости от результата программа переходит к лейблу *f_case1* или *f_case2*, где высчитываются соответствующие значения *f1* и *f2*. Далее значение *k* сверяется с *0*, команда *jne* проверяет условие *k != 0* и в зависимости от результата переводит программу в лейбл *case1* или *case2*, где высчитывается соответствующее значение *f3*.

Разработанный программный код см. в приложении А.

Файл листинга см. в приложении В.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | i1 | i2 | Результат | Комментарии |
|---|---|---|---|---|---|
| 1. | a = -3<br>b = 7<br>i = -2<br>k = 1 | -2 | 0 | res = -2 | Верно. a < b =><br>i1 = 3*-2+4 = -2; i2 = 3*(-2+2) = 0<br>k != 0 => min(-2, 0) = -2 |
| 2. | a = -3<br>b = 7<br>i = -2<br>k = 0 | -2 | 0 | res = 2 | Верно. a < b =><br>i1 = 3*-2+4 = -2; i2 = 3*(-2+2) = 0<br>k == 0 => \|-2 + 0\| = 2 |
| 3. | a = 0<br>b = -1<br>i = 2<br>k = -3 | 11 | -8 | res = -8 | Верно. a > b =><br>i1 = 15-2*2 = 11; i2 = -(6*2-4) = -8<br>k != 0 => min(11, 8) = -8 |

**Выводы.**

Были изучены представление и обработка целых чисел. Получены знания об организации ветвящихся процессов на языке Ассемблера.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЕ КОДЫ ПРОГРАММ

Название файла: lr3.asm

```
AStack SEGMENT STACK
     DW 2 DUP(?)
AStack ENDS

DATA SEGMENT
     a     DW -3
     b     DW 7
     i     DW -2
     k     DW 1
     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

          ; f1: i1 = 15-2i if a>b else 3i+4
          ; f2: i2 = 4 - 6i if a>b else 3i + 6
          mov ax, i
          ;3i
          mov bx, i
          shl bx,1
          shl bx,1
          sub bx,ax

          mov cx, a
          cmp cx, b
          jle f_case2
          f_case1:
               ;f1
               shl ax,1
               mov i1,15
               sub i1,ax
               ;f2
               shl bx,1
               mov i2,4
               sub i2,bx
               jmp f_final
          f_case2:
               ;f1
               mov i1,4
               add i1,bx
               ;f2
```

```
            mov i2,6
            add i2,bx
        f_final:

        ; f3: res = |i1 + i2| if k == 0 else min(i1,i2)
        mov ax,i1
        mov bx,i2
        mov res, ax

        mov cx,k
        cmp k, 0
        jne case2
        case1:
            add res, bx
            cmp res, 0
            jge final
            neg res
            jmp final
        case2:
            cmp res,bx
            jle final
            mov res,bx
        final:
        ret
    Main ENDP
CODE ENDS
END Main
```

# ПРИЛОЖЕНИЕ В

# ФАЙЛЫ ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ

Название файла: lr2-err.lst

```
0000                    AStack SEGMENT STACK
0000  0002[                      DW 2 DUP(?)
   ????
 ]

0004                    AStack ENDS

0000                    DATA SEGMENT
0000  FFFD                   a     DW -3
0002  0007                   b     DW 7
0004  FFFE                   i     DW -2
0006  0001                   k     DW 1
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

; f1: i1 = 15-2i if a>b else 3i + 4
; f2: i2 = 4 - 6i if a>b else 3i + 6
0009  A1 0004 R                   mov ax, i
;3i
000C  8B 1E 0004 R                mov bx, i
0010  D1 E3                       shl bx,1
0012  D1 E3                       shl bx,1
0014  2B D8                       sub bx,ax

0016  8B 0E 0000 R                mov cx, a
001A  3B 0E 0002 R                cmp cx, b
001E  7E 1B                       jle f_case2
0020                              f_case1:
;f1
0020  D1 E0                            shl ax,1
0022  C7 06 0008 R 000F               mov i1,15
0028  29 06 0008 R                    sub i1,ax
;f2
002C  D1 E3                            shl bx,1
002E  C7 06 000A R 0004               mov i2,4
0034  29 1E 000A R                    sub i2,bx
0038  EB 15 90                         jmp f_final
003B                              f_case2:
;f1
003B  C7 06 0008 R 0004               mov i1,4
0041  01 1E 0008 R                    add i1,bx
```

```
 ;f2
 0045  C7 06 000A R 0006              mov i2,6
 004B  01 1E 000A R                   add i2,bx
 004F                              f_final:

 ; f3: res = |i1 + i2| if k == 0
  else min(i1,i2)
 004F  A1 0008 R                   mov ax,i1
 0052  8B 1E 000A R                mov bx,i2
 0056  A3 000C R                   mov res, ax

 0059  8B 0E 0006 R                mov cx,k
 005D  83 3E 0006 R 00             cmp k, 0
 0062  75 12                       jne case2
 0064                              case1:
 0064  01 1E 000C R                     add res, bx
 0068  83 3E 000C R 00                  cmp res, 0
 006D  7D 11                            jge final
 006F  F7 1E 000C R                     neg res
 0073  EB 0B 90                         jmp final
 0076                              case2:
 0076  39 1E 000C R                     cmp res,bx
 007A  7E 04                            jle final
 007C  89 1E 000C R                     mov res,bx
 0080                              final:
 0080  CB                          ret
 0081                         Main ENDP
 0081                    CODE ENDS
 END Main
```

Segments and Groups:

| N a m e | Length | Align | Combine | Class |
|---|---|---|---|---|
| ASTACK . . . . . | 0004 | 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 |
| CASE1 . . . . . | L NEAR | 0064 | CODE |
| CASE2 . . . . . | L NEAR | 0076 | CODE |
| FINAL . . . . . | L NEAR | 0080 | CODE |
| F_CASE1 . . . . . | L NEAR | 0020 | CODE |
| F_CASE2 . . . . . | L NEAR | 003B | CODE |
| F_FINAL . . . . . | L NEAR | 004F | 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  lr3
@VERSION . . . . . .    TEXT  510


     77 Source  Lines
     77 Total   Lines
     22 Symbols

  48032 + 461275 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```