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

# ОТЧЕТ

**по лабораторной работе №3**
**по дисциплине «Организация ЭВМ и систем»**
**Тема: Представление и обработка целых чисел. Организация ветвящихся процессов**
**Вариант 4**

Студентка гр.1303 _____ Герасименко Я.Д.
Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

**Цель работы.**

Изучить представление и обработку целых чисел на Ассемблере. Научиться организовывать ветвящиеся процессы для выполнения задания.

**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 11*

| | | |
|---|---|---|
| / 15-2*i, при a>b<br>f1 = <<br>\ 3*i+4, при a<=b | / 20-4*i, при a>b<br>f5 = <<br>\ -(6*i-6), при a<=b | / min(\|i1-i2\|,2), при k=0<br>f5 = <<br>\ max(-6,-i2), при k>=0 |

**Выполнение работы.**

1. Были созданы три сегмента: сегмент стека (AStack), сегмент данных (DATA) и сегмент кода (CODE). Метки сегментов были записаны в соответствующие регистры с помощью директивы ASSUME (полное определение сегментов). Исходный код программы см. в приложении А.

2. В сегменте DATA были объявлены переменные A, B, I, K, I1, I2, RES. В этом сегменте будут меняться некоторые переменные во время тестирования.

3. В сегменте CODE была создана процедура Main, в которой написаны инструкции для завершения программы после операции ret. Для выполнения

задания использовались следующие переходы, чтобы избежать обращение к процедурам:

1). JMP – команда безусловного перехода. Выполняет безусловный переход в указанное место. В процедуре Main используется в случае, когда A больше B. Также используется в F3 и F3_1, чтобы перейти к записи результата вычисления функции.

2). JLE – команда, выполняющая короткий переход, если первый операнд меньше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp. В процедуре Main используется в самом начале для перехода к метке ALessB, если A не больше B

3). JGE – команда, выполняющая короткий переход, если первый операнд больше второго операнда или равен ему при выполнении сравнения с помощью команды cmp.

**Тестирование.**

Корректность работы программы была проверена тремя тестами.

1. Результаты работы программы при a=14; b=-5; i=2; k=2 представлены в табл.1.

| i1 | i2 | res | Корректность результата |
|---|---|---|---|
| 000B (11) | 000C(12) | FFF4 (-6) | Верно |

Таблица 1 – Результаты первого теста

2. Результаты работы программы при a=14; b=-5; i=2; k=0 представлены в табл.2.

| i1 | i2 | res | Корректность результата |
|---|---|---|---|
| 000B (11) | 000C(12) | 0001(1) | Верно |

**Выводы.**

В ходе выполнения лабораторной работы было изучена обработка целых чисел, их представление и организация ветвящихся процессов. Для выполнения задания написана программа, которая вычисляет значения функций согласно заданным условиям.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *lb3.asm*

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

DATA      SEGMENT
A        DW 14
B        DW -5
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
          mov CX, I
          cmp AX, B
          jle ALessB

BLessA:

       ;2(i+1)-4
          add CX, 1
          shl CX, 1
          sub CX, 4
          mov I2, CX
```

```
        ;-(4i+3)
        shl CX, 1
        add CX, 7
        neg  CX
        mov I1, CX
           jmp ABSI1

ALessB:

        ; 5 - 3*(i+1)
        add CX,  1
           mov BX, CX

           shl CX, 1
           shl CX, 1
           sub CX, BX
           neg  CX
           add CX, 5
           mov I2, CX

           ;6i - 10
           shl CX, 1
           neg  CX
           sub CX, 6
           mov I1, CX

ABSI1:

           mov CX, I1
           cmp CX, 0
           jge  F3
           neg I1

F3:

           mov CX, K
           cmp CX, 0
           jne ABSI2

F3_1:

           mov CX, I1
           cmp CX, 6
           jle MIN
```

```
mov AX, 6
```

```
jmp
F3RES
ULT
```

```
MIN:
          mov AX, I1
          jmp F3RESULT

ABSI2:
          mov CX, I2
          cmp CX, 0
          jge F3_2
          neg I2

F3_2:
          mov AX, I1
          add AX, I2
          jmp F3RESULT

F3RESULT:
          mov RES, AX
          ret

Main      ENDP
CODE      ENDS
          END Main
```

Название файла: *lb3.lst*

```
Microsoft (R) Macro Assembler Version 5.10                  10/30/22
17:15:2
                                                            Page     1-1


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

   0018                        AStack    ENDS

0000                           DATA      SEGMENT
   0000  000E                  A         DW 14
   0002  FFFB                  B         DW -5
   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
DS:DATA, SS:AStack                    ASSUME CS CODE,

   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  8B 0E 0004 R                     mov CX, I
   0010  3B 06 0002 R                     cmp AX, B
   0014  7E 1A                            jle ALessB

   0016                         BLessA:
                                          ;2(i+1)-4
   0016  83 C1 01                         add CX, 1
   0019  D1 E1                            shl CX, 1
   001B  83 E9 04                         sub CX, 4
   001E  89 0E 000A R                     mov I2, CX

                                          ;-(4i+3)
   0022  D1 E1                            shl CX, 1
   0024  83 C1 07                         add CX, 7
   0027  F7 D9                            neg CX
   0029  89 0E 0008 R                     mov I1, CX
   002D  EB 20 90                         jmp ABSI1

   0030                         ALessB:
                                          ; 5 - 3*(i+1)
   0030  83 C1 01                         add CX, 1
   0033  8B D9                            mov BX, CX

   0035  D1 E1                            shl CX, 1

 Microsoft (R) Macro Assembler Version 5.10                  10/30/22
 17:15:2
                                                              Page     1-2


   0037  D1 E1                            shl CX, 1
   0039  2B CB                            sub CX, BX
   003B  F7 D9                            neg CX
   003D  83 C1 05                         add CX, 5
   0040  89 0E 000A R                     mov I2, CX

                                          ;6i - 10
   0044  D1 E1                            shl CX, 1
   0046  F7 D9                            neg CX
   0048  83 E9 06                         sub CX, 6
   004B  89 0E 0008 R                     mov I1, CX

   004F                         ABSI1:
   004F  8B 0E 0008 R                     mov CX, I1
```

```
0053  83 F9 00                      cmp CX, 0
0056  7D 04                        jge F3
0058  F7 1E 0008 R                 neg I1

005C                          F3:
005C  8B 0E 0006 R                 mov CX, K
0060  83 F9 00                     cmp CX, 0
0063  75 15                        jne ABSI2

0065                          F3_1
0065  8B 0E 0008 R                 mov CX, I1
0069  83 F9 06                     cmp CX, 6
006C  7E 06                        jle MIN
006E  B8 0006                      mov AX, 6
0071  EB 1E 90                     jmp F3RESULT

0074                          MIN:
0074  A1 0008 R                        mov AX, I1
0077  EB 18 90                     jmp F3RESULT

007A                          ABSI2:
007A  8B 0E 000A R                 mov CX, I2
007E  83 F9 00                     cmp CX, 0
0081  7D 04                        jge F3_2
0083  F7 1E 000A R                 neg I2

0087                          F3_2:
0087  A1 0008 R                        mov AX, I1
008A  03 06 000A R                 add AX, I2
008E  EB 01 90                     jmp F3RESULT

0091                          F3RESULT:
0091  A3 000C R                        mov RES, AX
0094  CB                               ret

0095                          Main     ENDP
0095                          CODE     ENDS
                                       END Main
```

Segments and Groups:

| N a m e | Length | Align | Combine Class |
|---|---|---|---|
| ASTACK | 0018 | PARA | STACK |
| CODE | 0095 | PARA | NONE |
| DATA | 000E | PARA | NONE |

Symbols:

| N a m e | Type | Value | Attr | |
|---|---|---|---|---|
| A | L WORD | 0000 | DATA | |
| ABSI1 | L NEAR | 004F | CODE | |
| ABSI2 | L NEAR | 007A | CODE | |
| ALESSB | L NEAR | 0030 | CODE | |
| B | L WORD | 0002 | DATA | |
| BLESSA | L NEAR | 0016 | CODE | |
| F3 | L NEAR | 005C | CODE | |
| F3RESULT | L NEAR | 0091 | CODE | |
| F3_1 | L NEAR | 0065 | CODE | |
| F3_2 | L NEAR | 0087 | 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 = 0095 |
| MIN | L NEAR | 0074 | CODE | |
| RES | L WORD | 000C | DATA | |
| @CPU | TEXT | 0101h | | |
| @FILENAME | TEXT | lb3 | | |
| @VERSION | TEXT | 510 | | |

103 Source Lines
103 Total Lines
25 Symbols

48012 + 459248 Bytes symbol space free

0 Warning Errors
0 Severe Errors