**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.0382 _____ Литягин С.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

**Цель работы.**

Изучить представление и обработку целых чисел и организацию ветвящихся процессов.

**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:**

$$f2 = \begin{cases} -(4*i+3), & \text{при } a>b \\ 6*i-10, & \text{при } a<=b \end{cases} \qquad f6 = \begin{cases} 2*(i+1)-4, & \text{при } a>b \\ 5-3*(i+1), & \text{при } a<=b \end{cases} \qquad f5 = \begin{cases} \min(|i1|, 6), & \text{при } k=0 \\ |i1|+|i2|, & \text{при } k/=0 \end{cases}$$

**Выполнение работы:**

Создано три сегмента: AStack – сегмент стека, DATA – сегмент данных, CODE – сегмент кода. Используя директиву ASSUME, метки сегментов записаны в соответствующие регистры. В сегменте данных объявлены переменные a, b, I, k, i1, i2, res. В сегменте кода создана процедура Main, а также написаны инструкции для успешного завершения программы после операции ret.

Для выполнения работы использовались переходы во избежание использования процедур. Используемые переходы представлены в таблице 1.

Таблица 1 – Используемые переходы.

| Переход | Описание |
|---|---|
| JMP | Безусловый переход к метке. Используется при a > b, чтобы избежать выполнения кода при a<=b; в функции f3, чтобы перейти на запись результата ее вычисления во избежание выполнения кода других условий |
| JLE | Условный переход. Используется в самом начале программы для перехода к метке AlowerB, если a<=b; используется при вычислении результата функции f3, при условии k=0, а именно: если \|i1\| <=6, то переход к метке min |
| JGE | Условный переход. Используется в следующих ситуациях: переход к метке f3, если i1>=0; переход к метке f3_2, если i2>=0 |
| JNE | Условный переход. Используется в функции f3, чтобы при k = 0 избежать выполнения кода при k /= 0 |

Исходный код программы см. в приложении А.

**Тестирование:**

Для проверки работоспособности были проведены тесты, представленные в таблице 2.

Таблица 2 – Тесты

| Номер теста | a | b | i | k |
|---|---|---|---|---|
| 1 | 5 | -1 | 2 | 0 |
| 2 | 2 | 4 | -3 | 0 |
| 3 | 2 | 4 | -3 | 5 |

Результаты тестирования представлены в таблице 3.

Таблица 3 – Результаты тестирования

| Номер теста | i1 | i2 | res | Оценка результата |
|---|---|---|---|---|
| 1 | 000B (11) | 0002 (2) | 0006 (6) | Верно |
| 2 | 001C (28) | 000B (11) | 0006 (6) | Верно |
| 3 | 001C (28) | 000B (11) | 0027 (39) | Верно |

**Выводы.**

В ходе работы были изучены представление и обработка целых чисел и организация ветвящихся процессов, а также разработана программа, производящая вычисления функций, согласно условиям.

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

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

DATA SEGMENT
a DW 2
b DW 4
i DW -3
k DW 5
i1 DW 0
i2 DW 0
res DW 0
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, i
      add AX, 1 ; i+1
      mov CX, a
      cmp CX, b
      jle AlowerB
AbiggerB:
      shl AX, 1 ; 2*i+2
      sub AX, 4 ; 2*(i+1)-4 = 2i - 2
      mov i2, AX
      shl AX, 1 ; 4i-4
      add AX, 7 ; 4i+3
      neg AX ; -(4i+3)
      mov i1, AX
      jmp abs_i1
AlowerB:
      mov BX, AX
      shl AX, 1
      shl AX, 1
      sub AX, BX ; 3*(i+1)
      neg AX ; -3*(i+1)
      add AX, 5 ; 5-3*(i+1) = -3i+2
      mov i2, AX
      shl AX, 1 ; -6i+4
      add AX, 6 ; -6i+10
      neg AX ; 6i-10
```

```
      mov i1, AX
abs_i1:
      mov CX, i1
      cmp CX, 0
      jge f3
      neg i1
f3:
      mov CX, k
      cmp CX, 0
      jne abs_i2
f3_1:
      mov CX, i1
      cmp CX, 6
      jle min
      mov AX, 6
      jmp f3_res
min:
      mov AX, i1
      jmp f3_res
abs_i2:
      mov CX, i2
      cmp CX, 0
      jge f3_2
      neg i2
f3_2:
      mov AX, i1
      add AX, i2
      jmp f3_res
f3_res:
      mov res, AX
      ret
Main ENDP
CODE ENDS
      END Main
```

# ПРИЛОЖЕНИЕ Б

# ЛИСТИНГИ

Название файла: lb2.lst

```
Microsoft (R) Macro Assembler Version 5.10              10/31/21 06:53:5
                                                                     Page
1-1


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

0018                    AStack ENDS

0000                    DATA SEGMENT
0000  0002                    a DW 2
0002  0004                    b DW 4
0004  FFFD                    i DW -3
0006  0005                    k DW 5
0008  0000                    i1 DW 0
000A  0000                    i2 DW 0
000C  0000                    res DW 0
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 0004 R              mov AX, i
000C  05 0001                add AX, 1 ; i+1
000F  8B 0E 0000 R           mov CX, a
0013  3B 0E 0002 R           cmp CX, b
0017  7E 15                  jle AlowerB
0019                    AbiggerB:
0019  D1 E0                  shl AX, 1 ; 2*i+2
001B  2D 0004                sub AX, 4 ; 2*(i+1)-4 = 2i - 2
001E  A3 000A R              mov i2, AX
0021  D1 E0                  shl AX, 1 ; 4i-4
0023  05 0007                add AX, 7 ; 4i+3
0026  F7 D8                  neg AX ; -(4i+3)
0028  A3 0008 R              mov i1, AX
002B  EB 1B 90               jmp abs_i1
002E                    AlowerB:
002E  8B D8                  mov BX, AX
0030  D1 E0                  shl AX, 1
0032  D1 E0                  shl AX, 1
```

```
0034  2B C3              sub AX, BX ; 3*(i+1)
0036  F7 D8              neg AX ; -3*(i+1)
0038  05 0005            add AX, 5 ; 5-3*(i+1) = -3i+2
003B  A3 000A R          mov i2, AX
003E  D1 E0              shl AX, 1 ; -6i+4
0040  05 0006            add AX, 6 ; -6i+10
0043  F7 D8              neg AX ; 6i-10
0045  A3 0008 R          mov i1, AX
0048                abs_i1:
```



```
0048  8B 0E 0008 R       mov CX, i1
004C  83 F9 00           cmp CX, 0
004F  7D 04              jge f3
0051  F7 1E 0008 R       neg i1
0055                f3:
0055  8B 0E 0006 R       mov CX, k
0059  83 F9 00           cmp CX, 0
005C  75 15              jne abs_i2
005E                f3_1:
005E  8B 0E 0008 R       mov CX, i1
0062  83 F9 06           cmp CX, 6
0065  7E 06              jle min
0067  B8 0006            mov AX, 6
006A  EB 1E 90           jmp f3_res
006D                min:
006D  A1 0008 R          mov AX, i1
0070  EB 18 90           jmp f3_res
0073                abs_i2:
0073  8B 0E 000A R       mov CX, i2
0077  83 F9 00           cmp CX, 0
007A  7D 04              jge f3_2
007C  F7 1E 000A R       neg i2
0080                f3_2:
0080  A1 0008 R          mov AX, i1
0083  03 06 000A R       add AX, i2
0087  EB 01 90           jmp f3_res
008A                f3_res:
008A  A3 000C R          mov res, AX
008D  CB                 ret
008E                Main ENDP
008E                CODE ENDS
                         END Main
```



Segments and Groups:

| N a m e | Length | Align | Combine | Class |
|---|---|---|---|---|
| ASTACK . . . . . | 0018 | PARA | STACK | |

```
CODE . . . . . .      008E  PARA  NONE
DATA . . . . . .      000E  PARA  NONE

Symbols:

                N a m e         Type  Value     Attr

A  . . . . . .      L WORD    0000  DATA
ABIGGERB . . . . . .      L NEAR    0019  CODE
ABS_I1 . . . . . .      L NEAR    0048  CODE
ABS_I2 . . . . . .      L NEAR    0073  CODE
ALOWERB  . . . . . .      L NEAR    002E  CODE

B  . . . . . .      L WORD    0002  DATA

F3 . . . . . .      L NEAR    0055  CODE
F3_1 . . . . . .      L NEAR    005E  CODE
F3_2 . . . . . .      L NEAR    0080  CODE
F3_RES . . . . . .      L NEAR    008A  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 =
008E
MIN  . . . . . .      L NEAR    006D  CODE

RES  . . . . . .      L WORD    000C  DATA

@CPU . . . . . .      TEXT  0101h
@FILENAME  . . . . . .      TEXT  lb3
@VERSION . . . . . .      TEXT  510


     83 Source  Lines
     83 Total   Lines
     25 Symbols

  48030 + 461277 Bytes symbol space free

0 Warning Errors
0 Severe  Errors
```