**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 № 9**

Студент гр.1381 ____ Кагарманов Д. И.

Преподаватель ____ Ефремов М. А.

Санкт-Петербург

2022

**Цель работы.**

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

**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 №9

| | | |
|---|---|---|
| f2 = < / -(4*i+3), при a>b<br>\ 6*i - 10, при a<=b | f4 = < / -(6*i-4), при a>b<br>\ 3*(i+2), при a<=b | f7 = < / \|i1\|+\|i2\|, при k<0<br>\ max(6, \|i1\|) при k>=0 |

**Выполнение работы.**

1. Была объявлена успрощенная модель сегмантации типа SMALL. Под стек отведено 256 байт. Исходный код программы см. в приложении А.

2. В сегменте .DATA были объявлены однобайтные переменные a, b, i, k, i_2, i_4, res.

3. В сегменте .CODE адрес сегмента данных помещается в регистр ds, а дальше происходит работа с функциями. Были задействованы следующие регистры: al, bl, cl. Для выполнения задания при реализации функций использовались следующие команды:

1) JMP (JUMP) – команда безусловного перехода, то есть прыжок может

быть как дальним, так и ближним.

2) JGE (Jump if greater or equal) – команда, выполняющая короткий переход, если первый операнд больше или равен второму операнду при выполнении операции сравнения с помощью команды cmp.

3) JLE (Jump if less or equal) – команда, выполняющая короткий переход, если первый операнд меньше или равен второму при выполнении сравнения с помощью команды cmp.

4) JS (Jump if signed) – команда, выполняющая переход, если установлен флаг SF (если результат вычислений – отрицательное число).

5) NEG – команда, изменяющая знак числа.

6) SHL/SHR - команда, осуществляющая сдвиг всех битов операнда влево(вправо).

7) TEST – выполняет логическое И между всеми битами двух операндов. Необходима для проверки знакового бита при делении смещением вправо.

8) JNZ (Jump if not zero) – команда, выполняющая переход, если не установлен флаг ZF (предыдущая команда не равна 0).

Также для минимизации длины кода были упрощены вычисления функций:

При a > b:

f1 = -4i – 3

f2 = -6i + 4 = $\underline{f1 + 2i – 7}$

При a <= b:

f1 = 6i – 10

f2 = 3i + 6 = $\underline{f1/2 + 11}$

**Тестирование.**

Чтобы проверить корректность работы программы, было проведено три

1. Результаты работы программы при a=1; b=2; i=3; k=4 представлены в табл.1.

| i1 | i2 | res | Итог |
|---|---|---|---|
| 08 (8) | 0F(15) | 08 (8) | Верно |

Таблица 1 – Результаты первого теста

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

| i1 | i2 | res | Итог |
|---|---|---|---|
| 01 (1) | 0A(10) | 0B (11) | Верно |

Таблица 2 – Результаты второго теста

3. Результаты работы программы при a=-2; b=4; i=-6; k=5 представлены в табл.3.

| i1 | i2 | res | Итог |
|---|---|---|---|
| D2 (-46) | F4(-12) | 2E (46) | Верно |

Таблица 3 – Результаты третьего теста

**Выводы.**

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

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *LB3.ASM*

```
; a > b:
;     i_1 = -4i - 3
;     i_2 = i_1 +2*i - 7
;
;a <= b:
;     i_1 = 6i - 10
;     i_2 = i_1 / 2  + 11
;
DOSSEG
.MODEL SMALL
.STACK  100H
.DATA
     a db -2
     b db  4
     i db -6
     k db  5
     i_2 db ? ; значение первой функции == i1
     i_4 db ? ; значение второй функции == i2
     res db 0
.CODE
     mov ax, @data
     mov ds, ax
     mov al, a
     cmp al, b
     jle ElsePart
     ; f2
     mov al, i ; i
     shl al, 1 ; 2*i
     shl al, 1 ; 4*i
     add al, 3 ; 4*i + 3
     neg al ;    -(4*i + 3) = -4*i - 3
     mov i_2, al; al = i_2
     ; f4
     sub al, i ; -5*i - 3
     sub al, i ; -6*i - 3
     add al, 7 ; -6*i + 4
     mov i_4, al
     cmp k, 0
     jmp f7

     ElsePart:
     ;f2
          mov al, i ; i
          mov bl, i ; i
          shl al, 1 ; 2*i
          shl al, 1 ; 4*i
          shl bl, 1 ; 2*i
```

```
          add al, bl; 6*i
          sub al, 10; 6*i - 10
          mov i_2, al
     ;f4
          test al, 10000000b ; проверяем знаковый бит
          jnz negative ; если число отрицательное, то при сдвиге
вправо нужно вернуть знаковый бит
          shr al, 1 ; 3*i - 5
          add al, 11 ; 3*i + 6
          mov i_4, al
          jmp f7
          negative:
               shr al, 1
               add al, 10000000b ; возвращаем знаковый бит
отрицательному числу, потерянному при сдвиге
               add al, 11
               mov i_4, al
     f7:
          mov al, i_2
          mov bl, i_4

          getabs2:
          neg al
          js getabs2 ; меняем у числа i_2 знак до тех пор, пока оно не
станет положительным

          getabs4:
          neg bl
          js getabs4

          mov cl, k
          cmp cl, 0
          jge ElsePart_Final
          add res, al
          add res, bl
          jmp Ending

     ElsePart_Final:
          cmp al, 6
          jl min
          mov res, al
          jmp Ending
     min:
          mov res, 6
          jmp Ending
     Ending:
     mov al, i_2
     mov bl, i_4
     mov cl, res
     mov ah, 4ch
     int 21h
END
```

Название файла: *LB3.LST*

```
 Microsoft (R) Macro Assembler Version 5.10
11/8/22 02:24:25

Page     1-1


                        ; a > b:
                        ;    i_1 = -4i - 3
                        ;    i_2 = i_1 +2*i - 7
                        ;
                        ;a <= b:
                        ;    i_1 = 6i - 10
                        ;    i_2 = i_1 / 2  + 11
                        ;
                        DOSSEG
                        .MODEL SMALL
                        .STACK  100H
                        .DATA
 0000  FE                       a db -2
 0001  04                       b db  4
 0002  FA                       i db -6
 0003  05                       k db  5
 0004  00                       i_2 db ? ; значение первой
                        ¹ функции == i1
 0005  00                       i_4 db ? ; значение второй
                        ¹ функции == i2
 0006  00                       res db 0
                        .CODE
 0000  B8 ---- R                mov ax, @data
 0003  8E D8                    mov ds, ax
 0005  A0 0000 R                mov al, a
 0008  3A 06 0001 R                  cmp al, b
 000C  7E 23                    jle ElsePart
                             ; f2
 000E  A0 0002 R                mov al, i ; i
 0011  D0 E0                    shl al, 1 ; 2*i
 0013  D0 E0                    shl al, 1 ; 4*i
 0015  04 03                    add al, 3 ; 4*i + 3
 0017  F6 D8                    neg al ;    -(4*i + 3) = -4*i - 3
 0019  A2 0004 R                mov i_2, al; al = i_2
```

```
                        ; f4
001C  2A 06 0002 R              sub al, i ; -5*i - 3
0020  2A 06 0002 R              sub al, i ; -6*i - 3
0024  04 07                  add al, 7 ; -6*i + 4
0026  A2 0005 R              mov i_4, al
0029  80 3E 0003 R 00             cmp k, 0
002E  EB 2C 90                   jmp f7

0031                         ElsePart:
                        ;f2
0031  A0 0002 R                  mov al, i ; i
0034  8A 1E 0002 R                    mov bl, i ; i
0038  D0 E0                      shl al, 1 ; 2*i
003A  D0 E0                      shl al, 1 ; 4*i
003C  D0 E3                      shl bl, 1 ; 2*i
003E  02 C3                      add al, bl; 6*i
0040  2C 0A                      sub al, 10; 6*i - 10
0042  A2 0004 R                  mov i_2, al
                        ;f4
0045  A8 80                      test al, 10000000b ; прове


 Microsoft (R) Macro Assembler Version 5.10
11/8/22 02:24:25

Page     1-2


                    ряем знаковый бит
0047  75 0A                      jnz negative ; если чисЙ
                    »о отрицательное, то при сЙ
                    ´виге вправо нужно вернутэ
                    □ знаковый бит
0049  D0 E8                      shr al, 1 ; 3*i - 5
004B  04 0B                      add al, 11 ; 3*i + 6
004D  A2 0005 R                  mov i_4, al
0050  EB 0A 90                        jmp f7
0053                             negative:
0053  D0 E8                           shr al, 1
0055  04 80                           add al, 10000000b ; вЙ
                    ¾звращаем знаковый бит отэ
                    □ицательному числу, потерэ
```

```
                □нному при сдвиге
0057  04 0B                          add al, 11
0059  A2 0005 R                      mov i_4, al
005C                          f7:
005C  A0 0004 R                 mov al, i_2
005F  8A 1E 0005 R                    mov bl, i_4

0063                                  getabs2:
0063  F6 D8                           neg al
0065  78 FC                           js getabs2 ; меняем у ч
                исла i_2 знак до тех пор, поЙ
                °а оно не станет положител
                ьным

0067                                  getabs4:
0067  F6 DB                           neg bl
0069  78 FC                           js getabs4

006B  8A 0E 0003 R                    mov cl, k
006F  80 F9 00                        cmp cl, 0
0072  7D 0B                           jge ElsePart_Final
0074  00 06 0006 R                    add res, al
0078  00 1E 0006 R                    add res, bl
007C  EB 13 90                        jmp Ending

007F                          ElsePart_Final:
007F  3C 06                      cmp al, 6
0081  7C 06                      jl min
0083  A2 0006 R                 mov res, al
0086  EB 09 90                       jmp Ending
0089                          min:
0089  C6 06 0006 R 06                mov res, 6
008E  EB 01 90                        jmp Ending
0091                          Ending:
0091  A0 0004 R                 mov al, i_2
0094  8A 1E 0005 R                 mov bl, i_4
0098  8A 0E 0006 R                 mov cl, res
009C  B4 4C                    mov ah, 4ch
009E  CD 21                    int 21h
                END
```

```
11/8/22 02:24:25

Symbols-1


Segments and Groups:

                N a m e         Length     Align    Combine
Class

DGROUP . . . . . .      GROUP
  _DATA  . . . . . .      0007 WORD PUBLIC
   'DATA'
  STACK  . . . . . .      0100 PARA STACK
   'STACK'
_TEXT  . . . . . .      00A0 WORD PUBLIC
   'CODE'

Symbols:

                N a m e         Type  Value     Attr

A  . . . . . .      L BYTE    0000 _DATA

B  . . . . . .      L BYTE    0001 _DATA

ELSEPART . . . . . .      L NEAR    0031 _TEXT
ELSEPART_FINAL . . . . . .      L NEAR    007F _TEXT
ENDING . . . . . .      L NEAR    0091 _TEXT

F7 . . . . . .      L NEAR    005C _TEXT

GETABS2  . . . . . .      L NEAR    0063 _TEXT
GETABS4  . . . . . .      L NEAR    0067 _TEXT

I  . . . . . .      L BYTE    0002 _DATA
I_2  . . . . . .      L BYTE    0004 _DATA
I_4  . . . . . .      L BYTE    0005 _DATA

K  . . . . . .      L BYTE    0003 _DATA

MIN  . . . . . .      L NEAR    0089 _TEXT
```

```
NEGATIVE . . . . . .      L NEAR    0053 _TEXT

RES  . . . . . .          L BYTE    0006 _DATA

@CODE  . . . . . .        TEXT  _TEXT
@CODESIZE  . . . . . .    TEXT  0
@CPU . . . . . .          TEXT  0101h
@DATASIZE  . . . . . .    TEXT  0
@FILENAME  . . . . . .    TEXT  LB3
@VERSION . . . . . .      TEXT  510


 Microsoft (R) Macro Assembler Version 5.10
11/8/22 02:24:25


Symbols-2




     96 Source  Lines
     96 Total   Lines
     32 Symbols

  48022 + 459238 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```