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. 1383 _____ Малых А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Изучить представление и обработку целых чисел, а также организацию ветвящихся процессов на языке Ассемлера.

Задание.

(Вариант №13)

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.

i1 = f2 = -(4*i+3), при a>b; 6*i -10 , при a<=b

i2 = f8 = - (6*i+8), при a>b; 9 -3*(i-1), при a<=b

res = f3 = |i1 + i2|, при k=0; min(i1,i2), при k/=0

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 f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

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

Выполнение работы.

Разработана программа, реализующая вычисление значений функций по заданным параметрам. Проведена трансляция программы с различными значениями параметров. В отладчики проведено тестирование работы программы на соответствие полученным требованиям. Результаты тестирования приведены в таблице №1.

Таблица 1 – Примеры тестовых случаев

№	Входные данные	Выходные данные
1	a = 3 b = 0 i = 4 k = 0	i1 = FFED = -19 i2 = FFE0 = -32 res = 0033 = 33
2	a = -2 b = 0 i = -5 k = 0	i1 = FFD8 = -40 i2 = 001B = 27 res = 000D = 13
3	a = 2 b = 2 i = 1 k = 2	i1 = FFFC = -4 i2 = 0009 = 9 res = FFFC = -4
4	a = -3 b = 2 i = -3 k = 0	i1 = FFE4 = -28 i2 = 0015 = 21 res = 0007 = 7
5	a = 3 b = 0 i = 4 k = -5	I1 = FFED = -19 i2 = FFE0 = -32 res = FFE0 = -32

Исходный код программы см. в Приложении А

Файл диагностических сообщений см. в Приложении Б

Выводы.

Изучены представление и обработка целых чисел, а также организация ветвящихся процессов на языке Ассемлера.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

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

DATA SEGMENT
      a DW 3
      b DW 0
      i DW 4
      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
      AND ax, 0
      push ax
    mov ax, DATA
    mov ds, ax

      ; Calculate i1
      mov ax, a
      cmp ax, b

      jle @F
            ; if a > b: i1 = -(4i + 3)
            mov ax, i
            sal ax, 1
```

```
        sal ax, 1
        add ax, 3
        neg ax
        mov i1, ax
        jmp I2_a_GREATER_b
@@:
        ; if a <= b: i1 = 6i - 10
        mov ax, i
        sal ax, 1
        mov i1, ax ; i1 = 2i
        sal ax, 1
        sub ax, 10
        add i1, ax

; Calculating i2
; if a <= b: i2 = 9 - 3(i - 1) = -3i + 12
mov ax, i
mov i2, ax
sal ax, 1
add ax, i2
neg ax
add ax, 12
mov i2, ax
jmp RES_CALC

I2_a_GREATER_b:
        ; if a > b: i2 = - (6i + 8)
        mov ax, i
        sal ax, 1
        mov i2, ax
        sal ax, 1
        add ax, i2
        add ax, 8
        neg ax
        mov i2, ax

; Calculating res
```

```
      RES_CALC:
            cmp k, 0
            je ABS_CALC
                  ; if k != 0:  res = min(i1, i2)
                  cmp i1, ax ; Compare i1 and i2
                  jg @f
                        ; if i1 < i2
                        mov ax, i1
                  @@:
                        mov res, ax
                        jmp PROG_END
            ABS_CALC:
                  ; if k == 0: res = |i1 + i2|
                  mov res, ax ; res = i2
                  mov ax, i1
                  add res, ax
                  cmp res, 0
                  jge PROG_END
                        ; if res is negative
                        neg res
      PROG_END:
            ret
Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ В

ТЕКСТЫ ФАЙЛОВ ДИАГНОСТИЧЕСКИХ СООБЩЕНИЙ

Название файла: lab3.lst

```
     #Microsoft      (R)      Macro      Assembler      Version      5.10
10/30/22 20:05:1
                                                                      Page
1-1


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

       0018                 AStack ENDS

       0000                 DATA SEGMENT
       0000  0003                        a DW 3
       0002  0000                        b DW 0
       0004  0002                        i DW 2
       0006  FFFB                        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  25 0000                       AND ax, 0
       0004  50                      push ax
       0005  B8 ---- R              mov ax, DATA
       0008  8E D8                       mov ds, ax

                                     ; Calculate i1
       000A  A1 0000 R               mov ax, a
       000D  3B 06 0002 R                  cmp ax, b

       0011  7E 12                         jle @F
                                           ; if a > b: i1 = -(4i + 3)
       0013  A1 0004 R                     mov ax, i
       0016  D1 E0                               sal ax, 1
       0018  D1 E0                               sal ax, 1
       001A  05 0003                             add ax, 3
       001D  F7 D8                               neg ax
       001F  A3 0008 R                     mov i1, ax
       0022  EB 29 90                            jmp I2_a_GREATER_b
       0025                          @@:
                                           ; if a <= b: i1 = 6i - 10
```

```
0025  A1 0004 R               mov ax, i
0028  D1 E0                        sal ax, 1
002A  A3 0008 R               mov i1, ax ; i1 = 2i
002D  D1 E0                        sal ax, 1
002F  2D 000A                      sub ax, 10
0032  01 06 0008 R                 add i1, ax

                          ; Calculating i2
                          ; if a <= b: i2 = 9 - 3(i - 1) = -3i +
                     12
     #Microsoft     (R)     Macro     Assembler     Version     5.10
10/30/22 20:05:1
                                                                Page
1-2


0036  A1 0004 R          mov ax, i
0039  A3 000A R          mov i2, ax
003C  D1 E0                    sal ax, 1
003E  03 06 000A R             add ax, i2
0042  F7 D8                    neg ax
0044  05 000C                  add ax, 12
0047  A3 000A R          mov i2, ax
004A  EB 17 90                 jmp RES_CALC

004D                      I2_a_GREATER_b:
                               ; if a > b: i2 = - (6i + 8)
004D  A1 0004 R                mov ax, i
0050  D1 E0                         sal ax, 1
0052  A3 000A R                mov i2, ax
0055  D1 E0                         sal ax, 1
0057  03 06 000A R                  add ax, i2
005B  05 0008                       add ax, 8
005E  F7 D8                         neg ax
0060  A3 000A R                mov i2, ax

                          ; Calculating res
0063                      RES_CALC:
0063  83 3E 0006 R 00                   cmp k, 0
0068  74 0F                        je ABS_CALC
                                   ; if k != 0:  res = min
                         (i1, i2)
006A  39 06 0008 R                      cmp  i1,  ax  ;  Compare
i1
                          and i2
006E  7F 03                             jg @f
                                        ; if i1 < i2
0070  A1 0008 R                         mov ax, i1
0073                                @@:
0073  A3 000C R                         mov res, ax
0076  EB 16 90                               jmp PROG_END
0079                                ABS_CALC:
                                        ; if k == 0: res = |i1
                         + i2|
0079  A3 000C R                         mov res, ax ; res = i2
007C  A1 0008 R                         mov ax, i1
```

```
 007F  01 06 000C R                        add res, ax
 0083  83 3E 000C R 00                          cmp res, 0
 0088  7D 04                               jge PROG_END
                                           ; if res is neg
                      ative
 008A  F7 1E 000C R                             neg res
 008E                           PROG_END:
 008E  CB                             ret
 008F                      Main ENDP
 008F                      CODE ENDS
                           END Main
```

#Microsoft (R) Macro Assembler Version 5.10 10/30/22 20:05:1
Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	008F	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
ABS_CALC	L NEAR	0079	CODE
B	L WORD	0002	DATA
I	L WORD	0004	DATA
I1	L WORD	0008	DATA
I2	L WORD	000A	DATA
I2_A_GREATER_B	L NEAR	004D	CODE
K	L WORD	0006	DATA
MAIN	F PROC	0000	CODE Length = 008F
PROG_END	L NEAR	008E	CODE
RES	L WORD	000C	DATA
RES_CALC	L NEAR	0063	CODE
@0	L NEAR	0025	CODE
@1	L NEAR	0073	CODE
@CPU	TEXT	0101h	
@FILENAME	TEXT	LAB3	
@VERSION	TEXT	510	

```
   96 Source  Lines
   96 Total   Lines
   24 Symbols

47994 + 461297 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```