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 Тюленев Т. В.

Преподаватель Ефремов М. А.

Санкт-Петербург
2021

Цель работы.

Изучить и применить на практике навыки обработки целых чисел и организации 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.

№ студенческого билета =19

f1 = -(6*i-4), при a > b,

f1 = 3*(i+2), при a <= b

f2 = 20-4*i, при a > b

f2 = -(6*i-6), при a <= b

f3 = |i1|+|i2|, при k < 0

f3 =max(6, |i2|-3), при k >= 0

Выполнение работы.

После определения сегмента стека и сегмента данных и инициализации сегментных регистров в главной функции программы реализуется операция загрузки PSP в стек и инициализация сегментного регистра данных. После этого происходит сравнение переменных a и b. Если a > b, то последующими строками кода осуществляется запись значений в переменные i1 и i2 в соответствии с заданием, после чего осуществляется переход на метку с вычислением модуля первой функции. В противном случае выполняется код с меткой f12, где переменным i1 и i2 задаются значения, соответствующие второму случаю.

Блоки кода с метками reversi1 и reversi2 записывают в переменные i1 и i2 значение их модулей, используя, если нужно, команду neg, меняющую знак числа.

Вычисление результата происходит в блоке кода с меткой f3. Логика ветвления схожа с реализацией функций f1 и f2. В противном случае код выполняется по порядку с переходом в блок exit.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	a = 4 b = 3 i = 2 k = 1	answer = 8	Программа работает корректно
2	a = 3 b = 4 i = 2 k = 1	answer = 12	Программа работает корректно
3	a = 4 b = 3 i = 2 k = -1	answer = 20	Программа работает корректно

4	a = 3 b = 4 i = 2 k = -1	answer = 18	Программа работает корректно

Выводы.

Были изучены и применены на практике навыки обработки целых чисел и организации ветвящихся процессов.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.asm

```
; Program stack
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

; Program data
DATA SEGMENT
a DW 4
b DW 3
i DW 2
k DW 1
i1 DW 0
i2 DW 0
res DW 0

DATA ENDS

; Program code
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 cx, a
```

```
    cmp cx, b
    jle f12 ; a <= b

    ; i1   a > b
    mov ax, i
    mov bx, i

    mov cl, 03
    shl ax, cl

    shl bx, 1
    sub ax, bx

    mov i1, 4
    sub i1, ax

    ; i2   a > b
    mov ax, i

     mov cl, 02
    shl ax, cl

    mov i2, 20
    sub i2, ax

    jmp continuation

    f12:
        ; i1   a <=
b
          mov ax, i
          mov bx, i

          mov cl,
02
```

```
        shl ax, cl

          sub ax,
bx

        mov i1, 6
        add i1, ax

        ; i2   a <=
b
        mov ax, i
        mov bx, i

        mov cl, 03
        shl ax, cl

        shl bx, 1
        sub ax, bx

        mov i2, 6
        sub i2, ax

    continuation:
        cmp i1, 0
        jl
reversi1 ; i1 < 0

          cmp i2, 0
        jl
reversi2 ; i2 < 0

        jmp f3

    reversi1:
        neg i1
```

```
          cmp i2, 0
        jl
reversi2 ; i2 < 0


     reversi2:
          neg i2


    f3:
        cmp k, 0
        jl ff3 ; k <
0


          cmp i1, 6
          jl fff3 ;
i1 < 0


          mov ax,
i1
          mov res,
ax


          jmp exit


    ff3:
          mov ax,
i1
          mov bx,
i2
        mov res, ax
        add res, bx


          jmp exit


     fff3:
          mov res,
6


    exit:
```

```
        ret
Main ENDP
CODE ENDS
 END Main
```

Файл lb3.lst

```
#Microsoft (R) Macro Assembler Version 5.10                 11/8/21
15:45:31
                                                            Page
1-1


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

 0018                       AStack ENDS

                    ; Program data
 0000                       DATA SEGMENT
 0000  0004                 a DW 4
 0002  0003                 b DW 3
 0004  0002                 i DW 2
 0006  0001                 k DW 1
 0008  0000                 i1 DW 0
 000A  0000                 i2 DW 0
 000C  0000                 res DW 0

 000E                       DATA ENDS

                    ; Program code
 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  8B 0E 0000 R             mov cx, a
 000D  3B 0E 0002 R             cmp cx, b
 0011  7E 2D                    jle f12 ; a <= b

                        ; i1   a > b
 0013  A1 0004 R                mov ax, i
 0016  8B 1E 0004 R             mov bx, i

 001A  B1 03                    mov cl, 03
 001C  D3 E0                    shl ax, cl

 001E  D1 E3                    shl bx, 1
 0020  2B C3                    sub ax, bx
```

```
0022  C7 06 0008 R 0004       mov i1, 4
0028  29 06 0008 R            sub i1, ax


                          ; i2   a > b
002C  A1 0004 R               mov ax, i

002F  B1 02                    mov cl, 02
0031  D3 E0                   shl ax, cl
```

```
0033  C7 06 000A R 0014      mov i2, 20
0039  29 06 000A R           sub i2, ax

003D  EB 31 90               jmp continuation

0040                         f12:
                           ; i1   a <= b
0040  A1 0004 R                    mov ax, i
0043  8B 1E 0004 R                 mov bx, i

0047  B1 02                        mov cl, 02
0049  D3 E0                     shl ax, cl

004B  2B C3                        sub ax, bx

004D  C7 06 0008 R 0006         mov i1, 6
0053  01 06 0008 R              add i1, ax

                           ; i2   a <= b
0057  A1 0004 R                 mov ax, i
005A  8B 1E 0004 R              mov bx, i

005E  B1 03                     mov cl, 03
0060  D3 E0                     shl ax, cl

0062  D1 E3                     shl bx, 1
0064  2B C3                     sub ax, bx

0066  C7 06 000A R 0006         mov i2, 6
006C  29 06 000A R              sub i2, ax

0070                         continuation:
0070  83 3E 0008 R 00                cmp i1, 0
0075  7C 0A                     jl reversi1 ; i1 < 0

0077  83 3E 000A R 00                   cmp i2, 0
007C  7C 0E                     jl reversi2 ; i2 < 0

007E  EB 10 90                  jmp f3

0081                         reversi1:
0081  F7 1E 0008 R              neg i1

0085  83 3E 000A R 00                   cmp i2, 0
008A  7C 00                     jl reversi2 ; i2 < 0

008C                          reversi2:
008C  F7 1E 000A R                 neg i2

0090                         f3:
0090  83 3E 0006 R 00                cmp k, 0
0095  7C 10                     jl ff3 ; k < 0
```

```
 0097  83 3E 0008 R 06                    cmp i1, 6
 009C  7C 1A                          jl fff3 ; i1 < 0

 009E  A1 0008 R                      mov ax, i1
 00A1  A3 000C R                      mov res, ax

 00A4  EB 18 90                       jmp exit

 00A7                           ff3:
 00A7  A1 0008 R                      mov ax, i1
 00AA  8B 1E 000A R                   mov bx, i2
 00AE  A3 000C R                   mov res, ax
 00B1  01 1E 000C R                add res, bx

 00B5  EB 07 90                       jmp exit

 00B8                            fff3:
 00B8  C7 06 000C R 0006              mov res, 6

 00BE                           exit:
 00BE  CB                      ret
 00BF                       Main ENDP
 00BF                       CODE ENDS
                        END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	00BF	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr	
A	L WORD	0000	DATA	
B	L WORD	0002	DATA	
CONTINUATION	L NEAR	0070	CODE	
EXIT	L NEAR	00BE	CODE	
F12	L NEAR	0040	CODE	
F3	L NEAR	0090	CODE	
FF3	L NEAR	00A7	CODE	
FFF3	L NEAR	00B8	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 = 00BF
RES	L WORD	000C	DATA	
REVERSI1	L NEAR	0081	CODE	
REVERSI2	L NEAR	008C	CODE	
@CPU	TEXT	0101h		
@FILENAME	TEXT	lb3		
@VERSION	TEXT	510		

```
    129 Source  Lines
    129 Total   Lines
     24 Symbols

  48070 + 461237 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```