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.

Студентка гр. 1383 _____ Чернякова А.Д.
Преподаватель _____ Ефремов М.А.

Санкт-Петербур
г 2022

Цель работы.

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

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.

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.

Входные данные: вариант 24

i1:

$$f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases}$$

i2:

$$f7 = \begin{cases} -(4*i - 5), & \text{при } a>b \\ 10 - 3*i, & \text{при } a<=b \end{cases}$$

res:

$$f5 = \begin{cases} \min(|i1|, 6), & \text{при } k=0 \\ |i1|+|i2|, & \text{при } k/=0 \end{cases}$$

Выполнение работы.

Созданы три сегмента: Astack – сегмент стека, CODE – сегмент кода, DATA – сегмент данных. В сегменте данных объявлены переменные, соответствующие условию задания: a, b, i, k, i1, i2, res. В сегменте кода создана процедура Main, в которой вычисляются значения функций, заданных в условии. Данная процедура также содержит все необходимые инструкции для успешного завершения работы программы.

Для вычисления значений функций f1 и f2 сначала в регистр ax записывается переменная i и далее выполняются необходимые преобразования. Сравнение a и b происходит с помощью команды cmp, и если a<=b, с помощью команды jle происходит условный переход к нужному случаю. После всех преобразований в переменные i1 и i2 записываются данные из регистров ax и bx, которые содержат значения функций f1 и f2 соответственно.

Далее реализуется вычисление значения функции f3. Сначала вычисляется модуль значения переменной i1: с помощью команды jge проверяется условие i1>=0, и если оно верно, происходит переход к заданной метке. Далее происходит сравнение переменной k с нулем с помощью команды cmp, и если k!=0, то с помощью команды jne происходит переход к нужному случаю. В результате всех преобразований, в переменную res записываются данные из регистра ax, который содержит значение функции f3.

Исходный код программы смотреть в приложении А.

Файл листинг смотреть в приложении Б.

Тестирование.

Результаты тестирования представлены в таблице 1.

Таблица 1 - Результаты тестирования

№	Входные данные	Выходные данные	Комментарии
1.	a = 3 b = 2 i = 1 k = 0	i1 = 16 i2 = 1 res = 6	верно
2.	a = 3 b = 2 i = 1 k = 1	i1 = 16 i2 = 1 res = 17	верно
3.	a = -3 b = 2 i = 1 k = 0	i1 = 0 i2 = 7 res = 0	верно
3.	a = -3 b = 2 i = 2 k = 1	i1 = -6 i2 = 4 res = 10	верно

Выводы.

В процессе выполнения работы были изучены представление и обработка целых чисел, организация ветвящихся процессов. Была разработана программа, вычисляющая значения функций по заданным целочисленным параметрам.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

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

DATA SEGMENT
    a DW -3
    b DW 2
    i DW 2
    k DW 1
    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

f12:
    mov ax,i
    shl ax,1
    ;ax=2i add
    ax,i ;ax=3i
    neg ax ;ax=-3i
    mov cx,a
```

```
cmp cx,b
jle f12_case2 ;if a<=b
```

```
;f12_case1:
    sub ax,i ;ax=-4i
    add ax,5
    ;ax=-4i+5
    mov bx,ax
    ;bx=-(4i-5) add
    ax,15 ;ax=-4i+20 jmp
    f12_end

f12_case2:
    mov bx,ax ;bx=-3i
    shl ax,1 ;ax=-6i
    add ax,6
    ;ax=-6i+6
    add bx,10 ;bx=-3i+10

f12_end:
    mov i1,ax
    ;i1=-(6i-6) mov
    i2,bx ;i2=-3i+10


f3:
    cmp i1,0
    jge abs_i1 ;if i1>=0


    ;if i1<0
    neg i1


abs_i1:
    cmp k,0
    jne k_not_0 ;if k!=0

;k=0
    cmp i1,6
    jge min_6 ;if i1>=6
```

```
;min_i1
    mov ax,i1
    ;ax=i1 jmp
    f3_end


min_6:
    mov ax,6
    ;ax=6 jmp
    f3_end


k_not_0:
    cmp i2,0
    jge abs_i2 ;if i2>=0

    ;if i2<0
    neg i2


abs_i2:
    mov ax,i1 ;ax=|i1|
    add ax,i2 ;ax=|i1|+|i2|

f3_end:
    mov res,ax ;res=ax
    ret

Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГИ

Название файла: lab3.lst

```
 Microsoft (R) Macro Assembler Version 5.10                  11/6/22
16:07:17
                                                            Page     1-1


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

 0018                   AStack ENDS

 0000                   DATA SEGMENT
 0000  FFFD                     a DW -3
 0002  0002                     b DW 2
 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

 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                   f12:
 0009  A1 0004 R                mov ax,i
 000C  D1 E0               shl ax,1 ;ax=2i
 000E  03 06 0004 R        add ax,i ;ax=3i
 0012  F7 D8               neg ax ;ax=-3i
```

```
 0014  8B 0E 0000 R           mov cx,a
 0018  3B 0E 0002 R           cmp cx,b
 001C  7E 0F                  jle f12_case2 ;if a<=b

                     ;f12_case1
 001E  2B 06 0004 R           sub ax,i ;ax=-4i
 0022  05 0005                add ax,5 ;ax=-4i+5
 0025  8B D8                  mov bx,ax ;bx=-(4i-5)
 0027  05 000F                add ax,15 ;ax=-4i+20
 002A  EB 0B 90               jmp f12_end

 002D                f12_case2:
 002D  8B D8                  mov bx,ax ;bx=-3i
 002F  D1 E0                  shl ax,1 ;ax=-6i
 0031  05 0006                add ax,6 ;ax=-6i+6
 0034  83 C3 0A               add bx,10 ;bx=-3i+10

 0037                f12_end:
 0037  A3 0008 R                   mov i1,ax ;i1=-(6i-6)
 003A  89 1E 000A R           mov i2,bx ;i2=-3i+10

 003E                f3:
Microsoft (R) Macro Assembler Version 5.10                  11/6/22
16:07:17
                                                             Page     1-2


 003E  83 3E 0008 R 00        cmp i1,0
 0043  7D 04                  jge abs_i1 ;if i1>=0

                              ;if i1<0
 0045  F7 1E 0008 R           neg i1

 0049                abs_i1:
 0049  83 3E 0006 R 00        cmp k,0
 004E  75 13                  jne k_not_0 ;if k!=0

                     ;k=0
 0050  83 3E 0008 R 06        cmp i1,6
 0055  7D 06                  jge min_6 ;if i1>=6

                     ;min_i1
 0057  A1 0008 R                   mov ax,i1 ;ax=i1
 005A  EB 19 90               jmp f3_end
```

```
 005D                 min_6:
 005D  B8 0006                 mov ax,6 ;ax=6
 0060  EB 13 90                jmp f3_end

 0063                 k_not_0:
 0063  83 3E 000A R 00         cmp i2,0
 0068  7D 04                   jge abs_i2 ;if i2>=0

                               ;if i2<0
 006A  F7 1E 000A R            neg i2

 006E                 abs_i2:
 006E  A1 0008 R                    mov ax,i1 ;ax=|i1|
 0071  03 06 000A R            add ax,i2 ;ax=|i1|+|i2|

 0075                 f3_end:
 0075  A3 000C R                     mov res,ax ;res=ax
 0078  CB                       ret

 0079                 Main ENDP
 0079                 CODE ENDS
                      END Main
Microsoft (R) Macro Assembler Version 5.10                  11/6/22
16:07:17
                                                             Symbols-1


Segments and Groups:

                N a m e                 Length     Align     Combine Class

ASTACK . . . . . . . . . . . . .        0018  PARA  STACK
CODE . . . . . . . . . . . . . .        0079  PARA  NONE
DATA . . . . . . . . . . . . . .        000E  PARA  NONE

Symbols:

                N a m e                 Type   Value      Attr

A  . . . . . . . . . . . . . . .        L WORD     0000  DATA
ABS_I1 . . . . . . . . . . . . .        L NEAR     0049  CODE
ABS_I2 . . . . . . . . . . . . .        L NEAR     006E  CODE
```

```
B  . . . . . .     L WORD     0002  DATA

F12  . . . . . .     L NEAR     0009  CODE
F12_CASE2  . . . . . .     L NEAR     002D  CODE
F12_END  . . . . . .     L NEAR     0037  CODE
F3 . . . . . .     L NEAR     003E  CODE
F3_END . . . . . .     L NEAR     0075  CODE

I  . . . . . .     L WORD     0004  DATA
I1 . . . . . .     L WORD     0008  DATA
I2 . . . . . .     L WORD     000A  DATA

K  . . . . . .     L WORD     0006  DATA
K_NOT_0  . . . . . .     L NEAR     0063  CODE

MAIN . . . . . .     F PROC     0000  CODE  Length = 0079
MIN_6  . . . . . .     L NEAR     005D  CODE

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

@CPU . . . . . .     TEXT  0101h
@FILENAME  . . . . . .     TEXT  lab3
@VERSION . . . . . .     TEXT  510


     91 Source  Lines
     91 Total   Lines
     25 Symbols

  48002 + 461305 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```