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 _____ Злобин А. С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Разобрать и научится использовать механизм ветвления в программах на языке Ассемблер. Разработать программу на основе полученных знаний.

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.

Выполнение работы.

Функции выбранные в соответствии с вариантом:

$$f1 = \begin{cases} 15-2*i, & \text{при } a>b \\ 3*i+4, & \text{при } a<=b \end{cases}$$

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

$$f6 = \begin{cases} |i1-i2|, & \text{при } k<0 \\ \max(7, |i2|), & \text{при } k>=0 \end{cases}$$

В процессе выполнения задания была разработана программа, которая состоит из несколько частей:

1. Описание сегментов программы. В их число входят сегмент стека , сегмент данных в котором была выделена память для переменных ***a, b, i, k, i1, i2, res***.

2. Потом идет сегмент кода, в котором прописана сама программа.

3. Прописываются необходимые вещи для нормальной работы любой программы , такие как сохранение адреса начала PSP в стеке, загрузка сегментного регистра данных и т.д.

4. Затем анализируются значения ***a*** и ***b.*** Если ***a>b*** то выполняется блок программы ответственный за функции ***f1*** и ***f2*** для ***a>b***. Иначе выполняется блок для функций ***f1*** и ***f2*** при ***a<=b.*** Все это происходит в блоке начиная с метки ***f1_f2*** до метки ***f1_f2_end***.

5. В блоке с меткой ***f3*** анализируется значение ***k*** и выполняется функция ***f3*** в соответствии со значением ***k***.

6. В блоках ***k_more_0*** и ***k_less_0*** вычисляется значение функции f3 для значений k больше и меньше 0 соответственно

7. ***f3_end*** выполняет выход из программы

Тестирование.

Результаты тестирования представлены в табл. 1. Тестирование проводилось в отладчике **AFDPRO**.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии

1.	a = 5; b = -2; i = 2; k = 0	i1 = 000B (11);
		i2 = FFFD (-3);
		res = 0007 (7);
			Программа работает корректно

2.	a = 5; b = 2; i = -2; k = 1	i1 = 0013 (19);
		i2 = 000D (13);
		res =000D (13);
			Программа работает корректно

3.	a = -5; b = 2; i = -2;
	k = -1
		i1 = FFFE (-2);
		i2 = 0010 (16);
		res = 0012 (18);
			Программа работает корректно

Выводы.

Был изучен механизм ветвления в программах на языке Ассемблер.

Разработана программа, выполняющая поставленную задачу , а именно 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),

Приложение А

Исходный код программы

Название файла: lb3.asm

```
AStack SEGMENT STACK
    DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
    a Dw -5
    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

f1_f2:
    mov ax, i
    shl ax, 1; ax = 2i
    mov cx, i
    add cx, ax ; cx = 3i
    mov bx, ax ; bx = 2i

    mov ax, a
    cmp ax, b
    jg a_more_b

a_less_b:
    mov bx, cx ; bx = cx = 3i
    neg cx ; cx = -3i
    add bx, 4h ; bx = 3i + 4
    add cx, 0Ah ; cx = -3i + 10
    jmp f1_f2_end

a_more_b:
    mov cx, bx ; cx = bx = 2i
    neg bx ; bx = -2i
    shl cx, 1 ; cx = 4i
    neg cx ; cx = -4i
    add bx, 0Fh ; bx = -2i + 15
```

```
        add cx, 5h ; cx = -4i + 5

    f1_f2_end:
        mov i1, bx ; i1 = f1(i)
        mov i2, cx ; i2 = f2(i)

    f3:
        mov ax, k
        cmp ax, 0h
        jge k_more_0
    k_less_0:
        mov ax, i1
        sub ax, i2
        cmp ax, 0h
        jge sub_abs
        neg ax
    sub_abs:
        jmp f3_end
    k_more_0:
        mov ax, i2
        cmp ax, 0h
        jge abs_i2
        neg ax
    abs_i2:
        cmp ax, 7h
        jge f3_end
        mov ax, 7h

    f3_end:
        mov res, ax
        ret
    Main ENDP
    CODE ENDS
    END Main
```

Название файла: list.lst

```
    Microsoft      (R)      Macro      Assembler      Version      5.10
11/25/21 20:10:5
                                                                  Page
1-1


 0000                AStack SEGMENT STACK
 0000  0020[                   DW 32 DUP(?)
        ????
                ]

 0040                AStack ENDS

 0000                DATA SEGMENT
 0000  FFFB                    a Dw -5
```

```
 0002  0002                    b Dw 2
 0004  FFFE                    i Dw -2
 0006  FFFF                    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                f1_f2:
 0009  A1 0004 R                  mov ax, i
 000C  D1 E0                      shl ax, 1; ax = 2i
 000E  8B 0E 0004 R               mov cx, i
 0012  03 C8                      add cx, ax ; cx = 3i
 0014  8B D8                      mov bx, ax ; bx = 2i

 0016  A1 0000 R                  mov ax, a
 0019  3B 06 0002 R               cmp ax, b
 001D  7F 0D                      jg a_more_b

 001F                a_less_b:
 001F  8B D9                      mov bx, cx ; bx = cx = 3i
 0021  F7 D9                      neg cx ; cx = -3i
 0023  83 C3 04                   add bx, 4h ; bx = 3i + 4
 0026  83 C1 0A                   add cx, 0Ah ; cx = -3i + 10
 0029  EB 0F 90                   jmp f1_f2_end

 002C                a_more_b:
 002C  8B CB                      mov cx, bx ; cx = bx = 2i
 002E  F7 DB                      neg bx ; bx = -2i
 0030  D1 E1                      shl cx, 1 ; cx = 4i
 0032  F7 D9                      neg cx ; cx = -4i
 0034  83 C3 0F                   add bx, 0Fh ; bx = -2i + 15
 0037  83 C1 05                   add cx, 5h ; cx = -4i + 5

 003A                f1_f2_end:
```



```
 003A  89 1E 0008 R               mov i1, bx ; i1 = f1(i)
 003E  89 0E 000A R               mov i2, cx ; i2 = f2(i)
```

```
 0042                f3:
 0042  A1 0006 R              mov ax, k
 0045  3D 0000                cmp ax, 0h
 0048  7D 11                  jge k_more_0
 004A                k_less_0:
 004A  A1 0008 R              mov ax, i1
 004D  2B 06 000A R           sub ax, i2
 0051  3D 0000                cmp ax, 0h
 0054  7D 02                  jge sub_abs
 0056  F7 D8                  neg ax
 0058                sub_abs:
 0058  EB 13 90               jmp f3_end
 005B                k_more_0:
 005B  A1 000A R              mov ax, i2
 005E  3D 0000                cmp ax, 0h
 0061  7D 02                  jge abs_i2
 0063  F7 D8                  neg ax
 0065                abs_i2:
 0065  3D 0007                cmp ax, 7h
 0068  7D 03                  jge f3_end
 006A  B8 0007                mov ax, 7h

 006D                f3_end:
 006D  A3 000C R              mov res, ax
 0070  CB                 ret
 0071                Main ENDP
 0071                CODE ENDS
                          END Main
Microsoft (R) Macro Assembler Version 5.10                  11/25/21
20:10:5
                                                          Symbols-1


Segments and Groups:

                N a m e               Length      Align     Combine
Class

ASTACK . . . . . . . . . . . . . .     0040  PARA  STACK
CODE . . . . . . . . . . . . . . .     0071  PARA  NONE
DATA . . . . . . . . . . . . . . .     000E  PARA  NONE

Symbols:

                N a m e               Type   Value      Attr

A  . . . . . . . . . . . . . . . .     L WORD     0000  DATA
ABS_I2 . . . . . . . . . . . . . .     L NEAR     0065  CODE
A_LESS_B . . . . . . . . . . . . .     L NEAR     001F  CODE
A_MORE_B . . . . . . . . . . . . .     L NEAR     002C  CODE

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

```
F1_F2  . . . . . .     L NEAR     0009  CODE
F1_F2_END  . . . . .   L NEAR     003A  CODE
F3 . . . . . .         L NEAR     0042  CODE
F3_END . . . . . .     L NEAR     006D  CODE

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

K  . . . . . .         L WORD     0006  DATA
K_LESS_0 . . . . . .   L NEAR     004A  CODE
K_MORE_0 . . . . . .   L NEAR     005B  CODE

MAIN . . . . . .       F PROC     0000  CODE  Length      =
0071

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

SUB_ABS  . . . . . .   L NEAR     0058  CODE

@CPU . . . . . .       TEXT  0101h
@FILENAME  . . . . .   TEXT  LB3
@VERSION . . . . . .   TEXT  510


     82 Source  Lines
     82 Total   Lines
     26 Symbols

  48032 + 461275 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```