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 _____ Ильин Д. А.

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:

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

$$f8 = \begin{cases} -(6*i+8), & \text{при } a>b \\ 9-3*(i-1), & \text{при } a<=b \end{cases}$$

$$f7 = \begin{cases} |i1| + |i2|, & \text{при } k<0 \\ \max(6, |i1|), & \text{при } k>=0 \end{cases}$$

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 = FFEC (-20);
		res = 000B (11);
			Программа работает корректно

2.	a = 5; b = 2; i = -2; k = 1	i1 = 0013 (19);
		i2 = 0004 (4);
		res =0013 (19);
			Программа работает корректно

3.	a = -5; b = 2; i = -2;
	k = -1
		i1 = FFFE (-2);
		i2 = 0012 (18);
		res = 0014 (20);
			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 bx, i
   shl bx, 1; bx = 2i
   mov cx, i
   add cx, bx ; cx = 3i

   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, 0Ch ; cx = -3i + 12
   jmp f1_f2_end

a_more_b:
   shl cx, 1; cx = 6i
   add cx, 8h ; cx = 6i + 8
   neg bx ; bx = -2i
   neg cx ; cx = -(6i + 8)
   add bx, 0Fh ; bx = -2i + 15

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_1:
    mov ax, i1
    cmp ax, 0h
    jge k_less_0_2
    neg ax
k_less_0_2:
    mov bx, i2
    cmp bx, 0h
    jge k_less_0_3
    neg bx
k_less_0_3:
    add ax, bx
    cmp ax, 0h
    jmp f3_end

k_more_0:
    mov ax, i1
    cmp ax, 0h
    jge abs_i1
    neg ax
abs_i1:
    cmp ax, 6h
    jge f3_end
    mov ax, 6h

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

Название файла: list.lst

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

0040                            AStack ENDS

0000                            DATA SEGMENT
0000  0005                        a Dw 5
0002  0002                        b Dw 2
0004  FFFE                        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                            f1_f2:
0009  8B 1E 0004 R                mov bx, i
000D  D1 E3                       shl bx, 1; bx = 2i
000F  8B 0E 0004 R                mov cx, i
0013  03 CB                       add cx, bx ; cx = 3i

0015  A1 0000 R                   mov ax, a
0018  3B 06 0002 R                cmp ax, b
001C  7F 0D                       jg a_more_b

001E                            a_less_b:
001E  8B D9                       mov bx, cx ; bx = cx = 3i
0020  F7 D9                       neg cx ; cx = -3i
0022  83 C3 04                    add bx, 4h ; bx = 3i + 4
0025  83 C1 0C                          add cx, 0Ch ; cx = -3i + 12
0028  EB 0D 90                          jmp f1_f2_end

002B                            a_more_b:
002B  D1 E1                       shl cx, 1; cx = 6i
002D  83 C1 08                          add cx, 8h ; cx = 6i + 8
0030  F7 DB                       neg bx ; bx = -2i
0032  F7 D9                       neg cx ; cx = -(6i + 8)
0034  83 C3 0F                    add bx, 0Fh ; bx = -2i + 15
```

```
0037                            f1_f2_end:
0037  89 1E 0008 R                 mov i1, bx ; i1 = f1(i)
003B  89 0E 000A R                 mov i2, cx ; i2 = f2(i)
```

```
003F                    f3:
003F  A1 0006 R             mov ax, k
0042  3D 0000               cmp ax, 0h
0045  7D 1D                 jge k_more_0
0047                    k_less_0_1:
0047  A1 0008 R             mov ax, i1
004A  3D 0000               cmp ax, 0h
004D  7D 02                 jge k_less_0_2
004F  F7 D8                 neg ax
0051                    k_less_0_2:
0051  8B 1E 000A R           mov bx, i2
0055  83 FB 00              cmp bx, 0h
0058  7D 02                 jge k_less_0_3
005A  F7 DB                 neg bx
005C                    k_less_0_3:
005C  03 C3                 add ax, bx
005E  3D 0000               cmp ax, 0h
0061  EB 13 90              jmp f3_end

0064                    k_more_0:
0064  A1 0008 R             mov ax, i1
0067  3D 0000               cmp ax, 0h
006A  7D 02                 jge abs_i1
006C  F7 D8                 neg ax
006E                    abs_i1:
006E  3D 0006               cmp ax, 6h
0071  7D 03                 jge f3_end
0073  B8 0006               mov ax, 6h

0076                    f3_end:
0076  A3 000C R             mov res, ax
0079  CB                    ret
007A                    Main ENDP
007A                    CODE ENDS
                            END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0040	PARA	STACK	
CODE	007A	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr	
A	L WORD	0000	DATA	
ABS_I1	L NEAR	006E	CODE	
A_LESS_B	L NEAR	001E	CODE	
A_MORE_B	L NEAR	002B	CODE	
B	L WORD	0002	DATA	
F1_F2	L NEAR	0009	CODE	
F1_F2_END	L NEAR	0037	CODE	
F3	L NEAR	003F	CODE	
F3_END	L NEAR	0076	CODE	
I	L WORD	0004	DATA	
I1	L WORD	0008	DATA	
I2	L WORD	000A	DATA	
K	L WORD	0006	DATA	
K_LESS_0_1	L NEAR	0047	CODE	
K_LESS_0_2	L NEAR	0051	CODE	
K_LESS_0_3	L NEAR	005C	CODE	
K_MORE_0	L NEAR	0064	CODE	
MAIN	F PROC	0000	CODE	Length = 007A
RES	L WORD	000C	DATA	
@CPU	TEXT	0101h		
@FILENAME	TEXT	LB3		
@VERSION	TEXT	510		

87 Source Lines
87 Total Lines
27 Symbols

48008 + 459252 Bytes symbol space free

0 Warning Errors
0 Severe Errors