МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Организация ЭВМ и систем»

Тема: Представление и обработка целых чисел. Организация ветвящихся процессов.

Студент гр. 9383 _____ Хотяков Е.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер. Научиться строить программы с условными переходами.

Текст задания.

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.

Исходные данные.

Вариант 20

Таблица 1 — Исходные данные.

Имя функции	Функция
f1	-(6*i - 4) , при a>b
	3*(i+2) , при a<=b
f2	2*(i+1) — 4 = 2*(i-1) , при a>b
	5- 3*(i+1) , при a<=b
f3	|i1| - |i2|, при k<0
	max(4,|i2|-3), при k>=0

Ход работы.

В ходе работы была разработана программа, которая вычисляет значение функции по заданным целочисленным параметрам.

Исходные данные записываются в сегмент данных, как и выходные. Правильность записи была проверена с помощью отладчика.

Для подсчета значений функции были использованы следующие операнды:

- add – для суммирования
- sub – для вычитания
- shl – для логического сдвига влево, что равнозначно умножению на два

Результаты записывались по заранее заданным адресам переменных i1, i2 и res.

Для реализации условных переходов были использованы следующие операнды:

- cmp – для сравнения двух чисел. При использовании данного операнда его результат записывается с помощью выставления соответствующих флагов.
- jle – условный переход, срабатывающий, если левый аргумент выражения в cmp был меньше или равен второму. Для этого операнд проверяет, что флаги SF и OF не равны или флаг ZF = 1.
- jge – условный переход, срабатывающий, если левый аргумент выражения в cmp был больше или равен второму. Для этого операнд проверяет, что флаги SF и OF равны.
- jg - условный переход, срабатывающий, если левый аргумент выражения в cmp был больше второго. Для этого операнд проверяет, что флаги SF и OF равны и флаг ZF = 0.
- jl - условный переход, срабатывающий, если левый аргумент выражения в cmp был меньше второго. Для этого операнд проверяет, что флаги SF и OF не равны.
- jmp – безусловный переход. Используется, если для перехода к следующему адресу не нужно делать дополнительных проверок.

Исходный код и листинг программы представлены в приложении А.

Примеры работы программы.

Таблица 2 — Примеры работы программы.

№	a	b	i	k	i1	i2	res
1	2	1	2	-1	-8(FFF8)	2	6
2	1	2	2	-1	12	-4(FFFC)	8
3	2	1	4	3	-20(FFEC)	6	4
4	-1	-2	-4	3	28	-10(FFF6)	7

Выводы.

Было изучено представление и обработка целых чисел на языке Ассемблер. Была построена программа с условными переходами, которая считает значения функций с заданными целочисленными параметрами.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

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

DATA SEGMENT

a DW 2
b DW 1
i DW 2
k DW -1

i1 DW ?
i2 DW ?
res DW ?

DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR
     mov ax, DATA
     mov ds, ax
     mov ax, a
     cmp ax, b
     jle f12 ; if a <= b -> f12

f11: ;if a > b
     mov ax, i
     mov cl, 2
     shl ax, cl; 4*i
     add ax, i; 5*i
     add ax, i; 6*i
     sub ax, 4; 6*i-4
     neg ax; -(6*i - 4)
```

```
     mov i1, ax

f21: ; a > b
     mov ax, i
     sub ax, 1; i - 1
     shl ax, 1; 2(i-1)
     mov i2, ax

     jmp f3_cmp


f12: ; a <= b
     mov ax, i
     shl ax, 1; (2*i)
     add ax, i; (3*i)
     add ax, 6; (3*i + 6)
     mov i1, ax

f22:
     mov ax, i
     shl ax, 1;(2*i)
     add ax, i; (3*i)
     neg ax; (-3*i)
     add ax, 2;(-3*i + 2)
     mov i2, ax

     jmp f3_cmp

f3_cmp:
     cmp k, 0
     jge f32; if k >= 0 -> f32

f31: ;if k<0
     mov ax, i1
     cmp ax, 0
     jg f31_abs1 ; if i1 > 0
     neg ax ; |i1|
f31_abs1:
     mov bx, i2
     cmp bx, 0
     jg f31_abs2 ; if i2 > 0
     neg bx ;|i2|
```

```
f31_abs2:
     sub ax, bx; |i1| - |i2|
     mov ax, res
     jmp main_end

f32:
     mov ax, i2
     cmp ax, 0
     jg f32_abs2 ; if i2 > 0
     neg ax ; |i2|
f32_abs2:
     sub ax, 3; i2 - 3
     cmp ax, 4
     jg f32_max_right; if |i2| - 3 > 4
     mov res, 4
     jmp main_end
f32_max_right:
     mov res, ax

main_end:
     mov ah, 4ch
     int 21h

Main      ENDP
CODE      ENDS
          END Main
```

Название файла: lab3.lst

```
#MICROSOFT (R) MACRO ASSEMBLER VERSION 5.10          11/4/20 19:41:29
                                                     PAGE    1-1


0000                            ASTACK SEGMENT STACK
0000  0002[                        DW 2 DUP(?)
        ????
                                ]

0004                            ASTACK ENDS

0000                            DATA SEGMENT

0000  0002                      A DW 2
0002  0001                      B DW 1
0004  0002                      I DW 2
0006  FFFF                      K DW -1

0008  0000                      I1 DW ?
000A  0000                      I2 DW ?
000C  0000                      RES DW ?

000E                            DATA ENDS
                                ; Ð  ÐŸÐŽ Ð¿Ñ  ÐŸÐ³Ñ  Ð°ÐŒÐŒÑ
0000                            CODE SEGMENT
                                ASSUME CS:CODE, DS:DATA, SS:ASTACK

0000                            MAIN PROC FAR
0000  B8 ---- R              MOV AX, DATA
0003  8E D8                      MOV DS, AX
0005  A1 0000 R                  MOV AX, A
0008  3B 06 0002 R               CMP AX, B
000C  7E 25                      JLE F12 ; IF A <= B -> F12
```

```
000E                        F11: ;IF A > B
000E  A1 0004 R                  MOV AX, I
0011  B1 02                      MOV CL, 2
0013  D3 E0                      SHL AX, CL; 4*I
0015  03 06 0004 R               ADD AX, I; 5*I
0019  03 06 0004 R               ADD AX, I; 6*I
001D  2D 0004                    SUB AX, 4; 6*I-4
0020  F7 D8                      NEG AX; -(6*I - 4)
0022  A3 0008 R                  MOV I1, AX

0025                        F21: ; A > B
0025  A1 0004 R                  MOV AX, I
0028  2D 0001                    SUB AX, 1; I - 1
002B  D1 E0                      SHL AX, 1; 2(I-1)
002D  A3 000A R                  MOV I2, AX

0030  EB 24 90                   JMP F3_CMP


0033                        F12: ; A <= B
0033  A1 0004 R                  MOV AX, I
0036  D1 E0                      SHL AX, 1; (2*I)
#MICROSOFT (R) MACRO ASSEMBLER VERSION 5.10          11/4/20 19:41:29
                                    PAGE     1-2


0038  03 06 0004 R               ADD AX, I; (3*I)
003C  05 0006                    ADD AX, 6; (3*I + 6)
003F  A3 0008 R                  MOV I1, AX

0042                        F22:
0042  A1 0004 R                  MOV AX, I
0045  D1 E0                      SHL AX, 1;(2*I)
0047  03 06 0004 R               ADD AX, I; (3*I)
004B  F7 D8                      NEG AX; (-3*I)
004D  05 0002                    ADD AX, 2;(-3*I + 2)
0050  A3 000A R                  MOV I2, AX

0053  EB 01 90                   JMP F3_CMP
```

```
0056                        F3_CMP:
0056  83 3E 0006 R 00          CMP K, 0
005B  7D 1D                   JGE F32; IF K >= 0 -> F32

005D                        F31: ;IF K<0
005D  A1 0008 R               MOV AX, I1
0060  3D 0000                 CMP AX, 0
0063  7F 02                   JG F31_ABS1 ; IF I1 > 0
0065  F7 D8                   NEG AX ; |I1|
0067                        F31_ABS1:
0067  8B 1E 000A R           MOV BX, I2
006B  83 FB 00                CMP BX, 0
006E  7F 02                   JG F31_ABS2 ; IF I2 > 0
0070  F7 DB                   NEG BX ;|I2|
0072                        F31_ABS2:
0072  2B C3                   SUB AX, BX; |I1| - |I2|
0074  A1 000C R               MOV AX, RES
0077  EB 1F 90                    JMP MAIN_END

007A                        F32:
007A  A1 000A R               MOV AX, I2
007D  3D 0000                 CMP AX, 0
0080  7F 02                   JG F32_ABS2 ; IF I2 > 0
0082  F7 D8                   NEG AX ; |I2|
0084                        F32_ABS2:
0084  2D 0003                 SUB AX, 3; I2 - 3
0087  3D 0004                 CMP AX, 4
008A  7F 09                   JG F32_MAX_RIGHT; IF |I2| - 3 > 4
008C  C7 06 000C R 0004       MOV RES, 4
0092  EB 04 90                JMP MAIN_END
0095                        F32_MAX_RIGHT:
0095  A3 000C R               MOV RES, AX

0098                        MAIN_END:
0098  B4 4C                   MOV AH, 4CH
009A  CD 21                   INT 21H

009C                        MAIN    ENDP
009C                        CODE    ENDS
                                END MAIN
```

SEGMENTS AND GROUPS:

N A M E	LENGTH	ALIGN	COMBINE	CLASS
ASTACK	0004	PARA	STACK	
CODE	009C	PARA	NONE	
DATA	000E	PARA	NONE	

SYMBOLS:

N A M E	TYPE	VALUE	ATTR
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F11	L NEAR	000E	CODE
F12	L NEAR	0033	CODE
F21	L NEAR	0025	CODE
F22	L NEAR	0042	CODE
F31	L NEAR	005D	CODE
F31_ABS1	L NEAR	0067	CODE
F31_ABS2	L NEAR	0072	CODE
F32	L NEAR	007A	CODE
F32_ABS2	L NEAR	0084	CODE
F32_MAX_RIGHT	L NEAR	0095	CODE
F3_CMP	L NEAR	0056	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 = 009C

MAIN_END . . . . . L NEAR 0098 CODE

RES . . . . . L WORD 000C DATA

@CPU . . . . . TEXT 0101H
@FILENAME . . . . . TEXT LAB3
@VERSION . . . . . TEXT 510

104 SOURCE LINES
104 TOTAL LINES
28 SYMBOLS

47998 + 457212 BYTES SYMBOL SPACE FREE

0 WARNING ERRORS
0 SEVERE ERRORS