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. 8381 _____ Мельукмянц Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Получить знания о представлении и обработке целых чисел. Изучить понятие ветвящихся процессов и их организацию. Разработать на языке Ассемблера программу, вычисляющую значения функций, в зависимости от заданных параметров.

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 fn1,fn2 определяется из табл.1, а функции fn3- из табл.2 по цифрам шифра индивидуального задания (n1.n2.n3).

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.

Заданные функции:

/ 7 - 4*i , при a>b

f1= <

\ 8 -6*i , при a<=b

/ 20 - 4*i , при a>b

f2 = <

\ -(6*I - 6), при a<=b

/ |i1 + i2|, при k=0

f3 = <

\ min(i1,i2), при k/=0

Ход работы.

1. Определение модели памяти с помощью директивы .model. Модель памяти - small. Описание упрощенных директив сегментации:

- .stack – для указания начала сегмента стека;
- .data – для указания начала сегмента данных;
- .code – для указания начала сегмента кода;

2. Выбор размера стека, и инициализация переменных i, a, b, k;
3. Сравнение переменных a и b с помощью команды cmp. Если a > b переход к метке second, иначе выполняются команды метки first.
4. Далее происходит вычисление значений i1и i2 заданных функций. Значение i1 сохраняется в регистре ax. Значение i2 сохраняется в регистре cx.
5. При k = 0 вычисляется модуль суммы |i1+i2|. Иначе сравнивается i1 и i2 и минимальное значение в ax. В переменную res сохраняется конечное значение.

Минимизация длины кода.

- $a \leq b$

i1 = -6i+8; i2 = -6i+6;

Пусть j = -6i+6, тогда

i1 = j+2; i2 = j;

- $a > b$

i1 = -4i+7; i2 = -4i+20;

Пусть j = -4i+7, тогда

i1 = j; i2 = j+13;

Вывод.

В ходе выполнения данной лабораторной работы были получены сведения о реализации сравнения, меток и перехода по ним, а также изучены организации ветвлений в программах на языке Ассемблера.

Приложение А. Код программы lr3.asm

```
dosseg
.model small
.stack 100h
.data
i  dw -1
a dw -5
b dw -6
k dw 1
res dw ?
.code
mov ax, @data
mov ds, ax

mov ax, a
cmp ax, b
jg second

first: ;if(a<=b)
mov cx, i ;i
add cx, i ;2i
add cx, i ; 3i
neg cx  ; -3i
add cx, 3 ;-3i+3
sub cx, i
sub cx, i
sub cx, i
add cx, 3
;shl cx, 2 ; -(6i-6)

mov ax, cx ; -(6i-6)
add ax, 2 ; -6i+8
jmp final

second: ;if(a>b)
mov ax, i ; i
add ax, i
```

```
add ax, i
add ax, i;shl ax, 2 ; 4i
neg ax    ; -4i
add ax, 7 ; -4i+7

mov cx, ax ; -4i+7
add cx, 13 ; -4i+20

final:
abs_1:
cmp k, 0
jnz abs_0
add ax, cx
neg ax
js abs_1
jmp ex


abs_0:
cmp cx, ax
jl mnr
jmp ex
mnr: mov ax, cx
ex: mov res, ax
mov ah, 4ch
int 21h
end
```

Приложение Б. Тестирование программы.

№	Исходные данные	Ожидаемый результат	Полученный результат
1	$i = 0005_{16} = 5_{10}$ $a = 000F_{16} = 15_{10}$ $b = FFFD_{16} = -3_{10}$ $k = 0000_{16} = 0_{10}$	$i1 = FFF3_{16} = -13_{10}$ $i2 = 0000_{16} = 0_{10}$ $res = 000D_{16} = 13_{10}$	$i1 = FFF3_{16} = -13_{10}$ $i2 = 0000_{16} = 0_{10}$ $res = 000D_{16} = 13_{10}$
2	$i = 0002_{16} = 2_{10}$ $a = 0005_{16} = 5_{10}$ $b = 0000_{16} = 0_{10}$ $k = FFFF_{16} = -1_{10}$	$i1 = FFFF_{16} = -1_{10}$ $i2 = 000C_{16} = 12_{10}$ $res = FFFF_{16} = -1_{10}$	$i1 = FFFF_{16} = -1_{10}$ $i2 = 000C_{16} = 12_{10}$ $res = FFFF_{16} = -1_{10}$
3	$i = FFFE_{16} = -2_{10}$ $a = FFF8_{16} = -8_{10}$ $b = 0007_{16} = 7_{10}$ $k = FFFB_{16} = -5_{10}$	$i1 = 0014_{16} = 20_{10}$ $i2 = 0012_{16} = 18_{10}$ $res = 0012_{16} = 18_{10}$	$i1 = 0014_{16} = 20_{10}$ $i2 = 0012_{16} = 18_{10}$ $res = 0012_{16} = 18_{10}$
4	$i = 0005_{16} = 5_{10}$ $a = 000F_{16} = 15_{10}$ $b = FFFD_{16} = -3_{10}$ $k = 0009_{16} = 9_{10}$	$i1 = FFF3_{16} = -13_{10}$ $i2 = 0000_{16} = 0_{10}$ $res = FFF3_{16} = -13_{10}$	$i1 = FFF3_{16} = -13_{10}$ $i2 = 0000_{16} = 0_{10}$ $res = FFF3_{16} = -13_{10}$
5	$i = 000A_{16} = 10_{10}$ $a = 000A_{16} = 10_{10}$ $b = 000A_{16} = 10_{10}$ $k = 0000_{16} = 0_{10}$	$i1 = FFCC_{16} = -52_{10}$ $i2 = FFCA_{16} = -54_{10}$ $res = 006A_{16} = 106_{10}$	$i1 = FFCC_{16} = -52_{10}$ $i2 = FFCA_{16} = -54_{10}$ $res = 006A_{16} = 106_{10}$
6	$i = FFFF_{16} = -1_{10}$ $a = FFFB_{16} = -5_{10}$ $b = FFFA_{16} = -6_{10}$ $k = 0001_{16} = 1_{10}$	$i1 = 000B_{16} = 11_{10}$ $i2 = 0018_{16} = 24_{10}$ $res = 000B_{16} = 11_{10}$	$i1 = 000B_{16} = 11_{10}$ $i2 = 0018_{16} = 24_{10}$ $res = 000B_{16} = 11_{10}$

Приложение В. Содержимое файла листинга

```
#Microsoft (R) Macro Assembler Version 5.10              10/19/22 17:04:0
                                                        Page     1-1


                                                        dosseg
                                                        .model small
                                                        .stack 100h
                                                        .data
0000  FFFF                                              i  dw -1
0002  FFFB                                              a dw -5
0004  FFFA                                              b dw -6
0006  0001                                              k dw 1
0008  0000                                              res dw ?
                                                        .code
0000  B8 ---- R                                 mov ax, @data
0003  8E D8                                             mov ds, ax

0005  A1 0002 R                                         mov ax, a
0008  3B 06 0004 R                                      cmp ax, b
000C  7F 28                                             jg second

000E                                                    first: ;if(a<=b)
000E  8B 0E 0000 R                                      mov cx, i ;i
0012  03 0E 0000 R                                      add cx, i ;2i
0016  03 0E 0000 R                                      add cx, i ; 3i
001A  F7 D9                                             neg cx  ; -3i
001C  83 C1 03                                          add cx, 3 ;-3i+3
001F  2B 0E 0000 R                                      sub cx, i
0023  2B 0E 0000 R                                      sub cx, i
0027  2B 0E 0000 R                                      sub cx, i
002B  83 C1 03                                          add cx, 3
                                                        ;shl cx, 2 ; -(6i-6)

002E  8B C1                                             mov ax, cx ; -(6i-6)
```

```
0030  05 0002                  add ax, 2 ; -6i+8
0033  EB 1A 90                       jmp final

0036                             second: ;if(a>b)
      0036  A1 0000 R            mov ax, i ; i
      0039  03 06 0000 R           add ax, i
      003D  03 06 0000 R           add ax, i
0041  03 06 0000 R             add ax, i;shl ax, 2 ; 4i
      0045  F7 D8                  neg ax    ; -4i
      0047  05 0007              add ax, 7 ; -4i+7

004A  8B C8                    mov cx, ax ; -4i+7
004C  83 C1 0D                       add cx, 13 ; -4i+20

004F                                  final:
004F                                  abs_1:
      004F  83 3E 0006 R 00        cmp k, 0
      0054  75 09                  jnz abs_0
      0056  03 C1                  add ax, cx
      0058  F7 D8                   neg ax
      005A  78 F3                  js abs_1
005C  EB 0A 90                       jmp ex
```

```
005F                    abs_0:
005F  3B C8               cmp cx, ax
0061  7C 03               jl mnr
0063  EB 03 90            jmp ex
0066  8B C1             mnr: mov ax, cx
0068  A3 0008 R         ex: mov res, ax
006B  B4 4C               mov ah, 4ch
006D  CD 21               int 21h
                        end
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
DGROUP	GROUP			
_DATA	000A	WORD	PUBLIC	'DATA'
STACK	0100	PARA	STACK	'STACK'
_TEXT	006F	WORD	PUBLIC	'CODE'

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0002	_DATA
ABS_0	L NEAR	005F	_TEXT
ABS_1	L NEAR	004F	_TEXT
B	L WORD	0004	_DATA
EX	L NEAR	0068	_TEXT
FINAL	L NEAR	004F	_TEXT
FIRST	L NEAR	000E	_TEXT
I	L WORD	0000	_DATA
K	L WORD	0006	_DATA
MNR	L NEAR	0066	_TEXT
RES	L WORD	0008	_DATA

```
SECOND . . . . . L NEAR    0036        _TEXT

@CODE . . . . . TEXT  _TEXT
@CODESIZE . . . . . TEXT  0
@CPU . . . . . TEXT  0101h
@DATASIZE . . . . . TEXT  0
@FILENAME . . . . . TEXT  main
@VERSION . . . . . TEXT  510


        63 Source  Lines
        63 Total   Lines
        29 Symbols

    48022 + 461285 Bytes symbol space free

        0 Warning Errors
        0 Severe  Errors
```