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.
Вариант №4

Студентка гр. 1381 _____ Туркова Д.Н.

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:

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

$$f2 = \begin{cases} 20 - 4 * i, & \text{при } a > b \\ -(6 * i - 6), & \text{при } a \leq b \end{cases}$$

$$f3 = \begin{cases} \min(|i1 - i2|, 2), & \text{при } k < 0 \\ \max(-6, -i2), & \text{при } k \geq 0 \end{cases}$$

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. Записываем значение, а в регистр AX и с помощью cmp сравниваем ax и b. Выполняем короткий переход, если первый операнд больше второго.
4. Переходим к метке first, если a<=b. Записываем в cx результаты вычисления 1 функции, а в ax второй. Выполняется безусловный переход в final.
5. В метке second аналогично в cx записываются результаты вычисления первой функции, в ax результаты второй.
6. В метке final cmp сравнивает k и 0, далее идёт разделение:

- Метка f_0 для k<0. Sub вычитает из ax cx. Значение cx записывается в ax. Меняем знак ax, если число со знаком, то совершаем переход к getabs_0 и снова выполняем смену знака. Сравниваем ax с 2, оставляем ax наименьшее значение и переходим к ответу.
- В f_1 меняем знак ax и совершаем переход, если если число без знака. Сравниваем ax с -6, оставляем в ax наибольшее.

7. Полученное значение сохраняется в регистре ax. В переменную res сохраняется конечное значение.

Минимизация длины кода.

- $a \leq b$

  i1 = 3i+4; i2 = -(6i-6);

  Пусть j = 3i+4, тогда

  i1 = j; i2 = 2(j-7);

- $a > b$

  i1 = 15-2i; i2 = 20-4i;

  Пусть j = 15-2i, тогда

  i1 = j; i2 = 2(j-5);

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 2
b dw 3
k dw -1
res dw 0
.code
mov ax, @data
mov ds, ax

mov ax, a
cmp ax, b  ;сравниваем равны ли a и b.
jg second  ;выполняет короткий переход, если первый операнд БОЛЬШЕ
второго операнда

first:     ;if(a<=b)
mov cx, i  ;i
add cx, i  ;2i
add cx, i  ;3i
add cx, 4  ;3i+4

mov ax, cx ;3i+4
sub ax, 7  ;3i-3
sal ax, 1  ;6i-6
neg ax     ;-(6i-6)
jmp final

second:    ;if(a>b)
mov cx, 15 ;15
sub cx, i  ;15 - i
sub cx, i  ;15-2i

mov ax, cx ;15-2i
sub ax, 5  ;10-2i
sal ax, 1  ;20-4i

final:
f_0:
cmp k, 0   ; сравниваем равны ли k и 0
jge f_1   ;переход, если больше или равно
sub cx, ax
mov ax, cx
getabs_0:  ;модуль
neg ax
js getabs_0
cmp ax, 2  ;сравниваем
jl answer  ; ax < 2 => переход
mov ax, 2
jmp answer ;безусловный переход


f_1:
getabs_1:  ; отрицательный
```

```
neg ax
jns getabs_1 ; переход, если нет знака

cmp ax, -6
jg answer   ; переходим, если ax > -6
mov ax, -6

answer:
mov res, ax
mov ah, 4ch
int 21h

                                    end
```

Приложение Б. Тестирование программы.

№	Исходные данные	Ожидаемый результат	Полученный результат
1	i = $0001_{16} = 1_{10}$ a = $0002_{16} = 2_{10}$ b = $0003_{16} = 3_{10}$ k = $FFFF_{16} = -1_{10}$	i1 = $0007_{16} = 7_{10}$ i2 = $0000_{16} = 0_{10}$ res = $0002_{16} = 2_{10}$	i1 = $0007_{16} = 7_{10}$ i2 = $0000_{16} = 0_{10}$ res = $0002_{16} = 2_{10}$
2	i = $0001_{16} = 1_{10}$ a = $0002_{16} = 2_{10}$ b = $0003_{16} = 3_{10}$ k = $0000_{16} = 0_{10}$	i1 = $0007_{16} = 7_{10}$ i2 = $0000_{16} = 0_{10}$ res = $0000_{16} = 0_{10}$	i1 = $0007_{16} = 7_{10}$ i2 = $0000_{16} = 0_{10}$ res = $0000_{16} = 0_{10}$
3	i = $0001_{16} = 1_{10}$ a = $0003_{16} = 3_{10}$ b = $0002_{16} = 2_{10}$ k = $0000_{16} = 0_{10}$	i1 = $000D_{16} = 13_{10}$ i2 = $0010_{16} = 16_{10}$ res = $FFFA_{16} = -6_{10}$	i1 = $000D_{16} = 13_{10}$ i2 = $0010_{16} = 16_{10}$ res = $FFFA_{16} = -6_{10}$
4	i = $0005_{16} = 2_{10}$ a = $0005_{16} = 5_{10}$ b = $FFFD_{16} = -3_{10}$ k = $0000_{16} = 0_{10}$	i1 = $000B_{16} = 11_{10}$ i2 = $000C_{16} = 12_{10}$ res = $FFFA_{16} = -6_{10}$	i1 = $000B_{16} = 11_{10}$ i2 = $000C_{16} = 12_{10}$ res = $FFFA_{16} = -6_{10}$
5	i = $FFFE_{16} = 10_{10}$ a = $0001_{16} = 1_{10}$ b = $0002_{16} = 2_{10}$ k = $0000_{16} = 0_{10}$	i1 = $FFFE_{16} = -2_{10}$ i2 = $0012_{16} = 18_{10}$ res = $FFFA_{16} = -6_{10}$	i1 = $FFFE_{16} = -2_{10}$ i2 = $0012_{16} = 18_{10}$ res = $FFFA_{16} = -6_{10}$
6	i = $FFFF_{16} = -1_{10}$ a = $FFFB_{16} = -5_{10}$ b = $FFFB_{16} = -5_{10}$ k = $FFFA_{16} = -6_{10}$	i1 = $0001_{16} = 1_{10}$ i2 = $000C_{16} = 12_{10}$ res = $0002_{16} = 2_{10}$	i1 = $0001_{16} = 1_{10}$ i2 = $000C_{16} = 12_{10}$ res = $0002_{16} = 2_{10}$

Приложение В. Содержимое файла листинга


```
                        dosseg
                        .model small
                        .stack 100h
                        .data
0000  0001              i dw 1
0002  0002              a dw 2
0004  0003              b dw 3
0006  FFFF              k dw -1
0008  0000              res dw 0
                        .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  ;сравниваем равны лЙ
                        , a и b.
000C  7F 1B                   jg second  ;выполняет короткий
                         переход, если первый опер
                        анд БОЛЬШЕ второго операн
                        да

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  83 C1 04                add cx, 4  ;3i+4

001D  8B C1                   mov ax, cx ;3i+4
001F  2D 0007                 sub ax, 7  ;3i-3
0022  D1 E0                   sal ax, 1  ;6i-6
0024  F7 D8                   neg ax     ;-(6i-6)
0026  EB 13 90                jmp final
```

```
0029                    second:    ;if(a>b)
0029  B9 000F                 mov cx, 15 ;15
002C  2B 0E 0000 R            sub cx, i  ;15 - i
0030  2B 0E 0000 R            sub cx, i  ;15-2i

0034  8B C1                   mov ax, cx ;15-2i
0036  2D 0005                 sub ax, 5  ;10-2i
0039  D1 E0                   sal ax, 1  ;20-4i ????????

003B                    final:
003B                    f_0:
003B  83 3E 0006 R 00         cmp k, 0   ; сравниваем равны л
                        и k и 0
0040  7D 13                   jge f_1   ;переход, если большЙ
                        µ или равно
0042  2B C8                   sub cx, ax
0044  8B C1                   mov ax, cx
0046                    getabs_0:  ;модуль
0046  F7 D8                   neg ax
0048  78 FC                   js getabs_0
004A  3D 0002                 cmp ax, 2  ;сравниваем
```

```
004D  7C 12               jl answer  ; ax < 2 => переход
004F  B8 0002             mov ax, 2
0052  EB 0D 90            jmp answer ;безусловный перехо
                        д


0055                    f_1:
0055                    getabs_1:  ; отрицательный
0055  F7 D8                   neg ax
0057  79 FC                   jns getabs_1 ; переход, если нет
                        знака

0059  3D FFFA                 cmp ax, -6
005C  7F 03                   jg answer   ; переходим, если ax >
                         -6
005E  B8 FFFA                 mov ax, -6

0061                    answer:
0061  A3 0008 R         mov res, ax
0064  B4 4C                   mov ah, 4ch
0066  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	0068	WORD	PUBLIC	'CODE'

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0002	_DATA
ANSWER	L NEAR	0061	_TEXT
B	L WORD	0004	_DATA
FINAL	L NEAR	003B	_TEXT
FIRST	L NEAR	000E	_TEXT
F_0	L NEAR	003B	_TEXT
F_1	L NEAR	0055	_TEXT
GETABS_0	L NEAR	0046	_TEXT
GETABS_1	L NEAR	0055	_TEXT
I	L WORD	0000	_DATA
K	L WORD	0006	_DATA
RES	L WORD	0008	_DATA
SECOND	L NEAR	0029	_TEXT

```
@CODE . . . . . .       TEXT  _TEXT
@CODESIZE . . . . . .   TEXT  0
@CPU . . . . . .        TEXT  0101h
@DATASIZE . . . . . .   TEXT  0
@FILENAME . . . . . .   TEXT  lr3
@VERSION . . . . . .    TEXT  510


    67 Source  Lines
    67 Total   Lines
    30 Symbols

 47982 + 459278 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```