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.

Студентка гр. 9383 _____ Пономаренко С. А.

Преподаватель _____ Ефремов М. А.

Санкт-Петербург
2020

Цель работы.

Написать программу на языке Ассемблер, выполняющую функции с обработкой целых чисел, включающую ветвящиеся процессы.

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.

Вариант 15.

/ 7 - 4*i , при a>b

f3 = <

\ 8 -6*i , при a<=b

/ 20 - 4*i , при a>b

f5 = <

\ -(6*I - 6), при a<=b

/ |i1 + i2|, при k=0

f3 = <

\ min(i1,i2), при k/=0

Ход работы.

Исходные данные заносятся в программу до выполнения, выходные проверяются через отладчик. Были использованы команды:

cmp - сравнение двух операндов (флаг ZF устанавливается единицей, если числа равны, нулем - если числа не равны);

jle - выполнение короткого перехода, если первый операнд меньше второго операнда или равен ему при выполнении операции сравнения с помощью команды cmp;

shl - логический сдвиг влево (умножение числа в двоичной с. с. на 2);

neg - умножение на -1;

add - сложение двух операндов;

jmp - выполнение безусловного перехода;

sub - умножение двух операндов;

je - выполнение короткого перехода, если первый операнд равен второму операнду при выполнении операции сравнения с помощью команды cmp.

jl - выполнение короткого переход,а если первый операнд меньше второго операнда при выполнении операции сравнения с помощью команды cmp.

Тестирование.

1. a = 1, b = 2, i = 3, k = 4. i1 = -22, i2 = -24, res = 46.

2. a = 6, b = 6, i = 5, k = 0. i1 = -10, i2 = -12, res = -12.

Вывод.

Написали программу на языке Ассемблер, выполняющую функции с обработкой целых чисел, включающую ветвящиеся процессы.

Приложение А.

Код написанной программы.

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

DATA SEGMENT
a       DW   1       ;i1 = -22 i2 = -24 res = 46
b       DW   2
i       DW   3
k       DW   4
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

f1:
mov ax, a
cmp ax, b
jle f1_2       ; если первое < или = второму
mov ax, i      ; ax = i
shl ax, 1      ; ax = 2i
shl ax, 1      ; ax = 4i
```

```
neg ax                  ; ax = -4i
add ax, 7     ; ax = 7-4i
mov i1, ax    ; i1 = 7-4i
jmp f2_1
;7-4i (a>b)
;8-6i (a<=b)

f1_2:
mov ax, i     ; ax = i
shl ax, 1     ; ax = 2i !!!!!!!!!!!!!!!!!
mov bx, ax    ; bx = 2i
shl ax, 1     ; ax = 4i
add ax, bx    ; ax = 6i
neg ax                  ; ax = -6i
add ax, 8     ; ax = 8-6i
mov i1, ax    ; i1 = 8-6i

f2:
sub ax, 2     ; ax = 6-6i
mov i2, ax    ; i2 = 6-6i
jmp f3
;20-4i (a>b)
;-(6i-6) (a<=b)

f2_1: ; a > b
add ax, 13    ; ax = 20-4i
mov i2, ax    ; i2 = 20-4i
jmp f3

f3:
```

```
mov ax, k    ; ax = k
cmp ax, 0    ; if ax = 0
je f3_1
mov ax, i1   ; ax = i1
cmp ax, i2   ; if ax < i2
jl f3_min1
mov ax, i2   ; ax = i2
mov res, ax ; res = i2
jmp f_end
;|i1 + i2| (k=0)
;min(i1,i2) (k!= 0)

f3_1:
mov ax, i1   ; ax = i1
add ax, i2    ; ax = i1+i2
cmp ax, 0    ; if ax < 0
jl f3_abs
mov res, ax ; res = i1+i2
jmp f_end

f3_min1:
mov ax, i1   ; ax = i1
mov res, ax ; res = i1
jmp f_end

f3_abs:
neg ax                  ; ax = -(i1+i2)
mov res, ax ; res = -(i1+i2)
jmp f_end
```

```
f_end:
mov ah, 4ch
int 21h

Main  ENDP
CODE        ENDS
END Main
```