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. 9382 _____ Бочаров Г.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить арифметические команды ассемблера, разработать программу, вычисляющую необходимые переменные и углубить свои знания, в процессе написания программы.

Задание:

2 Вариант - 1.3.2

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.

/15-2*i , при a>b

f1 = <

\ 3*i+4 , при a<=b

/ / 7 - 4*i , при a>b

f3 = <

\ 8 -6*i , при a<=b

/ max(i1,10-i2), при k<0

f2 = <

\ |i1 - i2| , при k>=0

Ход работы

В ходе выполнения работы были использованы следующие команды арифметических операций:

Add – выполняет арифметическое сложение приемника и источника и помещает сумму в приемник.

Sub – вычитает источник из приемника и помещает разность в приемник.

Cmp - сравнивает приемник и источник и устанавливает флаги.

Neg - выполняет над числом, содержащимся в приемнике, операцию дополнения до двух.

В ходе выполнения работы были использованы следующие сдвиговые команды:

Sal – выполняет арифметический сдвиг влево.

В ходе выполнения работы были использованы следующие команды передачи ууправления:

Jg - переход, если больше (ZF = 0 и SF = OF).

Jz - переход, если 0 (ZF = 1).

Jl - переход, если меньше (SF <> OF).

Jmp - передает управление в другую точку программы.

Тестирование.

Вводные данные	Результат
a = 3 b = 5 i = 6 k = 1	i1 = 22 i2 = -28 res = 50

a = 2 b = 1 i = 1 k = 1	i1 = 13 i2 = 3 res = 10
a = 1 b = 1 i = 1 k = -1	i1 = 7 i2 = 2 res = 8
a = 2 b = 1 i = 1 k = -1	i1 = 13 i2 = 3 res = 13

Выводы.

В результате выполнения лабораторной работы был разработан код для подсчитывания определенных выражений, оптимизировано умножение. Были улучшены навыки письма в ассемблере.

Приложение.

Текст файла 1.ASM

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

DATA SEGMENT
a DW 1
b DW 1
i DW 1
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

mov ax, DATA
mov ds, ax


f1:
    mov ax, a
    cmp ax, b
    jle f1_1 ; a <= b
    jmp f1_0 ; a > b

f1_0:
    mov ax, i
    add ax, ax; i*2
    mov bx, 15; 15
    xchg bx, ax
    sub ax, bx

    mov i1, ax
    jmp f2

f1_1:

    mov ax, i
    mov bx, ax
    shl ax, 1; i*3
    add ax, bx
```

```
    mov bx, 4; 4
    add ax, bx

    mov i1, ax

    jmp f2

f2:
    mov ax, a
    cmp ax, b
    jle f2_1 ; a <= b
    jmp f2_0 ; a > b

f2_0:
    mov ax, i
    shl ax, 1; i*4
    shl ax, 1; i*4
    mov bx, 7; 7
    xchg ax, bx
    sub ax, bx

    mov i2, ax

    jmp f3
f2_1:
    mov ax, i
    mov bx, ax
    shl ax, 1; i*6
    add ax, bx
    shl ax, 1; i*6
    mov bx, 8; 8
    xchg ax, bx
    sub ax, bx

    mov i2, ax

    jmp f3
f3:
    mov ax, k;
    cmp k, 0
    jge f3_0 ; k >= 0
    jmp f3_1 ; k < 0
f3_1:

    mov ax, i2
    mov bx, 10; 10
    xchg ax, bx
    sub ax, bx

    cmp i1, ax;
    jae res1 ; >=
```

```
    mov res, ax

    jmp endLL
    res1:
    mov ax, i1
    mov res, ax
    jmp endLL
f3_0:
    mov ax, i1;
    mov bx, i2;
    sub ax, bx

    cmp ax, 0
    jl f3_0_abs

    mov res, ax

    jmp endLL

f3_0_abs:

    neg ax

    mov res, ax
    jmp endLL

endLL:

    mov bx, res
    mov ah, 4ch
    int 21h

Main ENDP
CODE ENDS
    END Main
```