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. 9382 _____ Докукин В.М.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить арифметические команды Ассемблера, организацию ветвлящихся процессов на языке Ассемблера; разработать программу, вычисляющую необходимые переменные и углубить свои знания в процессе написания программы.

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 7, шифр 1.4.3

f1 = 15-2*i, a > b

f1 = 3*i+4, a <= b

f2 = -(6*i+8), a > b

f2 = 9-3*(i-1), a <= b

f3 = |i1|+|i2|, k < 0

f3 = max(6, |i1|), k >= 0

Ход работы:

Согласно требованиям задания, f1, f2, f3 вычисляются непосредственно в функции Main; выбор нужного способа вычисления осуществляется при помощи меток AgreaterB, AlessequalB, KlessZero, KgreaterequalZero.

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 регистр флагов в соответствующее положение.

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 управления:

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

jge - переход, если больше или равно (SF == OF).

js - переход, если установлен флаг знака (SF == 1).

jmp - передает управление в другую точку программы.

Тестирование.

Вводные данные	Результат
a = 1 b = 1 i = 1 k = 1	i1 = 7 (0007) i2 = 9 (0009) res = 7 (0007)
a = 5 b = 4 i = 3	i1 = 9 (0009) i2 = -26 (FFE6) res = 9 (0005)

k = 2	
a = -5 b = -4 i = -3 k = -2	i1 = -5 (FFFB) i2 = 10 (000A) res = 10 (000A)

Выводы.

В результате выполнения лабораторной работы была разработана программа для вычисления значений выражений, указанных в варианте задания. Кроме того, были изучены особенности умножения на языке Ассемблера, а также улучшены навыки написания программ.

Приложение.

Имя файла: BRANCH.ASM

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

DATA SEGMENT
a   DW 5
b   DW 4
i   DW 3
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
 push DS
 sub  AX,AX
 push AX
 mov AX, DATA
 mov DS, AX
 mov CX, DS:a
 mov DX, DS:b
 cmp CX, DX
 jg AgreaterB
 jmp AlessequalB
```

```
AgreaterB:
 mov AX, DS:i  ; AX = i
 sal AX, 1     ; AX = 2i
 mov BX, 15    ; BX = 15
 sub BX, AX    ; BX = 15 - 2i
 mov DS:i1, BX ; i1 = 15 - 2i
 add AX, DS:i  ; AX = 3i
 sal AX, 1     ; AX = 6i
 mov BX, -8    ; BX = -8
 sub BX, AX    ; BX = -6i - 8
 mov DS:i2, BX ; i2 = -8 - 6i
 jmp cont

AlessequalB:
 mov AX, DS:i  ; AX = i
 sal AX, 1     ; AX = 2i
 add AX, DS:i  ; AX = 3i
 mov BX, 4     ; BX = 4
 add BX, AX    ; BX = 3i + 4
 mov DS:i1, BX ; i1 = 3i + 4
 mov BX, 12    ; BX = 12
 sub BX, AX    ; BX = 12 - 3i
 mov DS:i2, BX ; i2 = 12 - 3i

cont:
 mov AX, DS:i1 ; AX = i
absI1:
 neg AX
 js absI1      ; while SF is set (AX < 0)
 ; AX = |i1|
 mov BX, DS:k
 cmp BX, 0
```

```
 jge KgreaterequalZero
 jmp KlessZero

KgreaterequalZero:
 mov BX, 6
 cmp AX, BX
 jg I1greater6
 jmp endmain

KlessZero:
 mov BX, DS:i2
absI2:
 neg BX
 js absI2      ; while SF is set (BX < 0)
 ; BX = |i2|
 add BX, AX    ; BX = |i1| + |i2|
 jmp endmain

I1greater6:
 mov BX, AX

endmain:
 mov DS:res, BX ; res = |i1| + |i2| OR res = max(6, |i1|)
 ret
Main ENDP
CODE ENDS
 END Main
```