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 _____ Савельев И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

На практике изучить организация ветвящихся процессов в языке assembler. Научиться представлять и обрабатывать целые числа.

Задание.

Вариант 18

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a,b,i,k вычисляет:a) значения функций 1 = 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.

Ход выполнения.

В начале программы были объявлены сегмент стека, сегмент данных, сегмент кода. В сегменте данных были объявлены следующие переменные размером 2 байта: a, b, i, k, i1, i2, res. В сегменте кода были реализованы три условные функции с помощью меток и ветвления через команды cmp, jle, jl je. Для арифметических операций применялись команды sub, add, shl, shr.

Вывод.

В результате выполнения лабораторной работы была изучена организация ветвящихся процессов на языке assembler. На практике применены арифметические операции для работы с числами.

Приложение А. Исходный код программы.

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

DATA SEGMENT
     a     DW    1h
     b     DW    -1h
     i     DW    1h
     k     DW  -1h
     i1    DW  0
     i2    DW  0
     res   DW  0
DATA ENDS

CODE      SEGMENT
ASSUME  CS:CODE, DS:DATA

Main PROC  FAR
      mov  ax, DATA
      mov  ds, ax

f1:
     mov ax, a        ; ax = a
     cmp ax, b       ; с р а в н и в а е м  а  и  b
     jle f1_jle       ; a <= b

                      ; a > b
     mov ax, i        ; ax = i
     shl ax, 1        ; i * 4
     shl ax, 1
     mov bx, 7h       ; bx = 7
     sub bx, ax       ; bx - ax = 7 - i * 4
     mov i1, bx       ; i1 = bx - ax = 7 - i * 4
     jmp f2

f1_jle:               ; a <= b
     mov ax, i        ; ax = i
     shl ax, 1        ; i * 4
     shl ax, 1
     mov bx, i        ; bx = i
     shl bx, 1        ; i * 2
     add ax, bx       ; ax + bx = (4 * i) + (2 * i) = i * 6
     mov bx, 8h
     sub bx, ax       ; 8 - i * 6
     mov i1, bx
     jmp f2

f2:
     mov ax, a        ; ax = a
```

```
    cmp ax, b       ; с р а в н и в а е м  a  и  b
    jle f2_jle      ; a <= b

                    ; a > b

    mov ax, i       ; ax = i
    shl ax, 1       ; i * 4
    shl ax, 1
    mov bx, i       ; bx = i
    shl bx, 1       ; i * 2
    add ax, bx      ; ax + bx = i * 6
    mov bx, 8h      ; bx = 8
    add ax, bx      ; i * 6 + 8
    neg ax          ; * (-1)
    mov i2, ax
    jmp f3          ; п е р е х о д и м  к  f3_i2

f2_jle:
    mov ax, i       ; ax = i
    mov bx, 1h
    sub ax, bx      ; ax = i - 1
    mov bx, ax      ; bx = i - 1
    shl ax, 1       ; i * 4
    shl ax, 1
    shl bx, 1       ; i * 2
    add ax, bx      ; ax + bx = i * 6
    shr ax, 1       ; (i * 6) / 2 = i * 3
    mov bx, 9h      ; bx = 9
    sub bx, ax      ; bx - ax
    mov i2, bx
    jmp f3          ; к f3_i2

f3:
   mov bx, k        ; bx = k
   cmp bx, 0        ; с р а в н и в а е м  k  с  0
   jl  f3_jl_up     ; k < 0  в е р х н я я  в е т к а

                    ; k >= 0  н и ж н я я  в е т к а
    mov ax, i2      ; ax = i2
    cmp ax, 0       ; с р а в н и в а е м  i2  и  0
    jl f3_i2_c      ; е с л и  i2 < 0

                    ; i2 >= 0
    jmp f3_down

f3_i2_c:
    neg ax          ; м е н я е м  з н а к  i2
    jmp f3_down

f3_down:
    cmp ax, 7
```

```
    jl f3_down_i2    ; 7 >= i2

                         ; 7 < i2
    mov res, ax
    jmp end_f

f3_down_i2:
    mov res, 7h
    jmp end_f


f3_jl_up:
    mov ax, i1       ; ax = i1
    mov bx, i2       ; bx = i2
    sub ax, bx       ; i1 - i2
    cmp ax, 0        ; с р а в н и в а е м  i1 - i2  и  0
    jl f3_jl_up_c    ; i1 < 0

    mov res, ax     ; i1 >= 0
    jmp end_f

f3_jl_up_c:
    neg ax           ; м е н я е м  з н а к  i1
    mov res, ax     ; i1 > 0
    jmp end_f


end_f:
    mov ah, 4ch
    int 21h


Main      ENDP
CODE      ENDS
END Main                  ;ENDS CODE
```