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. 1303 _____ Попандопуло А. Г.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Разработка программы на языке Ассемблера, поведение которой определяется заданными целочисленными значениями параметров и представляет собой вычисление значений некоторых функций.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет: а) 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 20.

Выполнение работы

В соответствии с выбранным вариантом были реализованы заданные функции. Программа протранслирована с различными текстовыми данными.

В ходе выполнения лабораторной работы были использованы команды:

Для передачи данных:

- mov – присваивание

Арифметические команды:

1) add - сложение
2) sub - вычитание
3) cmp - сравнение
4) neg – смена знака

Сдвига:

- sal – сдвиг влево

Передача управления:

1) jmp – безусловный переход
2) jg – короткий переход, при условии что первый операнд больше второго при сравнении командой cmp.
3) jl – короткий переход, при условии что первый операнд меньше второго при сравнении командой cmp.
4) jge – короткий переход, при условии что первый операнд больше или равен второму при сравнении командой cmp.

В работе наблюдалась явна необходимость реализации ветвления, которое на языке Ассемблера возможно представить с помощью меток – символьных имен, содержащих определенные команды. Переходы между метками обеспечиваются с помощью описанных ранее команд передачи управления.

Трансляция и линковка программы:

```
C:\>masm lb3.asm
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.

Object filename [lb3.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  50132 + 461225 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link lb3.obj

Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserved.

Run File [LB3.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Приложение А.

Исходный код программы.

```
;task option №20
;f1&f2:
;        / -(6*i - 4) , при a>b
;f4 = <
;        \ 3*(i+2) , при a<=b

;        / 2*(i+1) -4 , при a>b
;f6 = <
 ;       \ 5 - 3*(i+1), при a<=b

;f3:
;        / |i1| - |i2|, при k<0
;f8 = <
;        \ max(4,|i2|-3), при k>=0

ASSUME CS:CODE, SS: AStack, DS: DATA

AStack  SEGMENT STACK
        DW 12 DUP('!')
AStack  ENDS

DATA    SEGMENT
a   DW  0
b   DW  0
i   DW  0
k   DW  0

i1  DW  0
```

```
i2  DW  0
res DW  0

DATA    ENDS
CODE    SEGMENT
Main    PROC    FAR
    push DS
    sub AX,AX
    push AX

    mov AX, DATA
    mov DS, AX

; f1&f2:
    mov ax, a
    mov cx, i
    cmp ax, b
    jg a_greater

;   a<=b

    sal cx, 1   ; i << 1 = i*2
    add cx, i   ; i*2 + i = i*3
    add cx, 6  ;i*3+6 = 3*(i+2)
    mov i1, cx

    neg cx              ; -i*3-6
    add cx, 8  ; 2 - i*3 = 5 - 3(i + 1)
    mov i2,cx
    jmp F3
```

```
a_greater:
    mov ax, 2
    neg ax
    sal cx, 1
    sub ax, cx; -2 + 2*i = 2*(i + 1) - 4
    mov i1, ax

    mov ax, 4
    add cx, i    ; i*2 + i = 3*i
    sal cx, 1    ; 3i*2
    neg cx               ; -6*i
    sub ax, cx ; 4 - 6*i = -(6i-4)
    mov i2, ax


F3:
    mov ax,i1
    mov bx,i2
    cmp ax, 0
    jge cmp_k

pos_i2: ;               i2 = |i2|
    neg ax
    mov res, ax

cmp_k:
        mov cx, k
        cmp cx, 0
        jl neg_k ; k < 0
```

```
pos_k: ; k >= 0
        mov bx,3
        neg bx
        sub ax,bx
        mov res, ax
        cmp ax, bx
        jg final
        mov res, 4
        jmp final

    neg_k: ; k < 0
         cmp bx, 0
         jge case3
         neg bx

    case3:
         sub res, bx

    final:
        ret

Main    ENDP
CODE    ENDS
        END Main
```