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: "Изучение режимов адресации и формирования исполнительного адреса"

Студент гр. 9383 _____ Рыбников Р.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Ознакомиться с представлением целых чисел, а так же с их обработкой.

Реализовать программу на языке Ассемблер.

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 10.

$$f3 = \begin{cases} 7 - 4 * i, & \text{при } a > b \\ 8 - 6 * i, & \text{при } a <= b \end{cases}$$

$$f7 = \begin{cases} -(4*i - 5, & \text{при } a>b \\ 10 - 3*i, & \text{при } a<=b \end{cases}$$

$$f5 = \begin{cases} \min(|i1|, 6), & \text{при } k = 0 \\ |i1| + |i2|, & \text{при } k != 0 \end{cases}$$

Ход работы.

Для решения задачи была реализована программа на низкоуровневом языке программирования Ассемблер, которая вычисляет значение функций по заданным целочисленным параметрам.

Исходные данные заносятся в программу до выполнения, а выходные данные отслеживаются через отладчик.

Для организации ветвления были использованы следующие инструкции:

- cmp — сравнение с изменением флага ZF (0 — аргументы не равны, 1 — аргументы равны)
- jmp — безусловный переход по заданной метке (без изменения флагов)
- jl — условный переход по заданной метке, в случае если первый аргумент меньше второго после выполнения cmp.
- je - условный переход по заданной метке, в случае если флаг ZF равен 1.
- shl - логический сдвиг влево, для реализации умножения на 2.

Тестирование.

1. a = 1, b = 2, i = 3, k = 4 => f3 = -10, f7 = 1, f5 = 11
2. a = 2, b = 1, i = 3, k = 4 => f3 = -5, f7 = -7, f5 = 12
3. a = 2, b = 1, i = 3, k = 0 => f3 = -5, f7 = -7 , f5 = 5
4. a = 1, b = 2, i = 3, k =0 => f3= -10, f7 = 1, f5 = 6

Выводы.

Прошло ознакомление с представлением целых чисел и их обработкой. Была реализована программа для решения поставленной задачи на языке Ассемблер.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД.

Название файла: main.asm

```
        ; 3.7.5 => tab 2 = f3 and f7 and f5(tab3)
Astack SEGMENT STACK
  DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
a         DW 1
b         DW 2
i         DW 3
k         DW 0
i1        DW ?
i2        DW ?
result    DW ?
DATA ENDS

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

Main PROC FAR
  mov ax, DATA
  mov ds, ax


Func3_i1_1:
  mov ax, a     ; ax = 1
  cmp ax, b     ; b = 2 => a <= b
  jg Func3_i1_2    ; переходим к функции "7 - 4*i" если a>b

  ; 8 - 6*i выполняем
  mov ax, i
  shl ax, 1     ; ax = 2*ax
  mov bx, ax    ; bx = ax          ; при этих данных -10 тут
  shl ax, 1     ; ax = 4*ax
  add ax, bx    ; ax = 2*ax + 4*ax = 6*ax
  neg ax        ; ax = -6*ax
```

```
  add ax, 8     ; ax = 8 - 6*ax
  mov i1, ax

  jmp Func7_i2_1


; 7 - 4i
Func3_i1_2:
  mov ax, i
  shl ax, 1      ; ax = 2ax
  shl ax, 1      ; ax = 4ax
  neg ax         ; ax = -4ax
  add ax, 7
  mov i1, ax
  jmp Func7_i2_2


; if a<=b: 10-3*i   (нужно сделать из 8 - 6*i)
Func7_i2_1:       ; взаимосвязь с Func3_i1_1
  mov cx, i
  mov dx, cx      ; dx = cx
  shl cx, 1       ; cx = 2cx          ; тут будет значение 1 при этом i

  add cx, dx      ; cx = cx + 2cx = 3cx
  neg cx          ; cx = -3cx
  add cx, 10      ; 10 - 3cx
  mov i2, cx
  ;готово
  jmp f5_sum



; if a>b: -(4*i - 5) = 5 - 4i   (нужно сделать из 7 - 4*i)
; тут сделать только сложение
Func7_i2_2:
  mov ax, i     ; ax = i
  shl ax, 1     ; ax = 2i
  shl ax, 1     ; ax = 4i
  sub ax, 5     ; ax = 4i - 5
  neg ax        ; 5 - 4i
  mov i2, ax
;  jmp f5_sum


f5_sum:
```

```
  mov ax, k
  cmp k, 0
  je f5_min      ; jump if equal
  jmp f5_res_SumAbs_1

  ;jl f5_res_SumAbs     ; если не равно то прыгаем на f5_res_SumAbs

f5_min:
  mov bx, i1
  mov cx, 6
  cmp bx, 0 ; if i1<0 -> neg i1
  jl f5_neg
  jmp f5_cmp_main



f5_neg:
  mov bx, i1
  neg bx  ; bx = -i1
  jmp f5_cmp_main

f5_cmp_main:
  mov cx, 6
  cmp bx, cx    ; bx v cx
  jl f5_res_i1
  jmp f5_res_6


f5_res_SumAbs_1:
  mov bx, i1 ; тут все верно
  cmp bx, 0
  jl f5_neg_sum
  jmp f5_res_SumAbs_2

f5_res_SumAbs_2:
  mov cx, i2
  cmp cx, 0
  jl f5_neg_sum_2
  jmp f5_res_SumAbs_continue

f5_neg_sum:
  neg bx
  jmp f5_res_SumAbs_2
                    ; эти две ф-ии для работы в случе k != 0
f5_neg_sum_2:
```

```
  neg cx
  jmp f5_res_SumAbs_continue

f5_res_SumAbs_continue:
  mov ax, bx
  add ax, cx
  mov result, ax
  jmp f_end


f5_res_6:
  mov result, 6
  jmp f_end


f5_res_i1:
  mov result, bx
  jmp f_end


f_end:
  mov ah, 4ch
  int 21h


Main  ENDP
CODE  ENDS
      END Main
```

ПРИЛОЖЕНИЕ Б
ЛИСТИНГ ПРОГРАММЫ

Название файла: list.lst

```
Microsoft (R) Macro Assembler Version 5.10              10/26/20 05:39:3
                                                        Page     1-1


                        ; 3.7.5 => tab 2 = f3 and f7 and f5(tab3)
0000                            Astack SEGMENT STACK
0000  0020[                      DW 32 DUP(?)
    ????
                    ]

0040                            AStack ENDS

0000                            DATA SEGMENT
0000  0001                      a       DW 1
0002  0002                      b       DW 2
0004  0003                      i       DW 3
0006  0000                      k       DW 0
0008  0000                      i1      DW ?
000A  0000                      i2      DW ?
000C  0000                      result    DW ?
000E                            DATA ENDS

0000                            CODE SEGMENT
                              ASSUME CS:CODE, DS:DATA, SS:AStack

0000                            Main PROC FAR
0000  B8 ---- R                  mov ax, DATA
0003  8E D8                      mov ds, ax


0005                            Func3_i1_1:
0005  A1 0000 R                  mov ax, a     ; ax = 1
0008  3B 06 0002 R               cmp ax, b     ; b = 2 => a <= b
000C  7F 16                      jg Func3_i1_2    ; переходим к фу
                            нкции "7 - 4*i" если a>b
```

```
                        ; 8 - 6*i выполняем
000E  A1 0004 R             mov ax, i
0011  D1 E0                 shl ax, 1     ; ax = 2*ax
0013  8B D8                 mov bx, ax    ; bx = ax          ; при э
                    тих данных -10 тут
0015  D1 E0                 shl ax, 1     ; ax = 4*ax
0017  03 C3                 add ax, bx    ; ax = 2*ax + 4*ax = 6*ax
0019  F7 D8                 neg ax        ; ax = -6*ax
001B  05 0008                     add ax, 8     ; ax = 8 - 6*ax
001E  A3 0008 R             mov i1, ax

0021  EB 13 90                    jmp Func7_i2_1


                    ; 7 - 4i
0024                      Func3_i1_2:
0024  A1 0004 R             mov ax, i
0027  D1 E0                 shl ax, 1      ; ax = 2ax
0029  D1 E0                 shl ax, 1      ; ax = 4ax
002B  F7 D8                 neg ax         ; ax = -4ax
002D  05 0007                     add ax, 7
0030  A3 0008 R             mov i1, ax


Microsoft (R) Macro Assembler Version 5.10          10/26/20 05:39:3
                                    Page     1-2


0033  EB 17 90                    jmp Func7_i2_2


                    ; if a<=b: 10-3*i   (нужно сделать
                    из 8 - 6*i)
0036                      Func7_i2_1:       ; взаимосвязь с F
                    unc3_i1_1
0036  8B 0E 0004 R                mov cx, i
003A  8B D1                 mov dx, cx      ; dx = cx
003C  D1 E1                 shl cx, 1       ; cx = 2cx         ; тут
                    будет значение 1 при этом i

003E  03 CA                 add cx, dx      ; cx = cx + 2cx = 3cx
0040  F7 D9                 neg cx          ; cx = -3cx
0042  83 C1 0A                    add cx, 10      ; 10 - 3cx
```

```
0045  89 0E 000A R              mov i2, cx
                        ;готово
0049  EB 10 90                  jmp f5_sum



                        ; if a>b: -(4*i - 5) = 5 - 4i   (нужно с?
                        ?елать из 7 - 4*i)
                        ; тут сделать только сложе
                        ние
004C                            Func7_i2_2:
004C  A1 0004 R                  mov ax, i     ; ax = i
004F  D1 E0                      shl ax, 1     ; ax = 2i
0051  D1 E0                      shl ax, 1     ; ax = 4i
0053  2D 0005                           sub ax, 5     ; ax = 4i - 5
0056  F7 D8                      neg ax        ; 5 - 4i
0058  A3 000A R                  mov i2, ax
                        ;  jmp f5_sum


005B                            f5_sum:
005B  A1 0006 R                  mov ax, k
005E  83 3E 0006 R 00                   cmp k, 0
0063  74 03                      je f5_min      ; jump if equal
0065  EB 23 90                          jmp f5_res_SumAbs_1

                         ;jl f5_res_SumAbs     ; если не рав?
                        ?о то прыгаем на f5_res_SumAbs

0068                            f5_min:
0068  8B 1E 0008 R                      mov bx, i1
006C  B9 0006                           mov cx, 6
006F  83 FB 00                          cmp bx, 0 ; if i1<0 -> neg i1
0072  7C 03                      jl f5_neg
0074  EB 0A 90                          jmp f5_cmp_main



0077                            f5_neg:
```

```
0077  8B 1E 0008 R              mov bx, i1
007B  F7 DB                   neg bx  ; bx = -i1
007D  EB 01 90                jmp f5_cmp_main

0080                          f5_cmp_main:
0080  B9 0006                 mov cx, 6
0083  3B D9                   cmp bx, cx    ; bx v cx
0085  7C 37                   jl f5_res_i1
0087  EB 2C 90                jmp f5_res_6


008A                          f5_res_SumAbs_1:
008A  8B 1E 0008 R            mov bx, i1 ; тут все верно
008E  83 FB 00                cmp bx, 0
0091  7C 0F                   jl f5_neg_sum
0093  EB 01 90                jmp f5_res_SumAbs_2

0096                          f5_res_SumAbs_2:
0096  8B 0E 000A R            mov cx, i2
009A  83 F9 00                cmp cx, 0
009D  7C 07                   jl f5_neg_sum_2
009F  EB 0A 90                jmp f5_res_SumAbs_continue

00A2                          f5_neg_sum:
00A2  F7 DB                   neg bx
00A4  EB F0                   jmp f5_res_SumAbs_2
                                    ; эти две ф-и
                        и для работы в случе k != 0
00A6                          f5_neg_sum_2:
00A6  F7 D9                   neg cx
00A8  EB 01 90                jmp f5_res_SumAbs_continue

00AB                          f5_res_SumAbs_continue:
00AB  8B C3                   mov ax, bx
00AD  03 C1                   add ax, cx
00AF  A3 000C R               mov result, ax
00B2  EB 11 90                jmp f_end


00B5                          f5_res_6:
00B5  C7 06 000C R 0006       mov result, 6
00BB  EB 08 90                jmp f_end
```

```
00BE                    f5_res_i1:
00BE  89 1E 000C R              mov result, bx
00C2  EB 01 90                  jmp f_end


00C5                    f_end:
00C5  B4 4C               mov ah, 4ch
00C7  CD 21               int 21h
```

```
00C9                    Main  ENDP
00C9                    CODE  ENDS
                      END Main
```


Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0040	PARA	STACK	
CODE	00C9	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F5_CMP_MAIN	L NEAR	0080	CODE
F5_MIN	L NEAR	0068	CODE
F5_NEG	L NEAR	0077	CODE
F5_NEG_SUM	L NEAR	00A2	CODE
F5_NEG_SUM_2	L NEAR	00A6	CODE
F5_RES_6	L NEAR	00B5	CODE
F5_RES_I1	L NEAR	00BE	CODE
F5_RES_SUMABS_1	L NEAR	008A	CODE
F5_RES_SUMABS_2	L NEAR	0096	CODE
F5_RES_SUMABS_CONTINUE	L NEAR	00AB	CODE
F5_SUM	L NEAR	005B	CODE
FUNC3_I1_1	L NEAR	0005	CODE
FUNC3_I1_2	L NEAR	0024	CODE
FUNC7_I2_1	L NEAR	0036	CODE

```
FUNC7_I2_2 . . . . . .        L NEAR     004C  CODE
F_END  . . . . . .    L NEAR     00C5  CODE

I  . . . . . .        L WORD     0004  DATA
I1 . . . . . .        L WORD     0008  DATA
I2 . . . . . .        L WORD     000A  DATA

K  . . . . . .        L WORD     0006  DATA

MAIN . . . . . .      F PROC     0000  CODE        Length = 00C9

RESULT . . . . . .    L WORD     000C  DATA

@CPU . . . . . .      TEXT  0101h
@FILENAME  . . . . . .        TEXT  main
@VERSION . . . . . .          TEXT  510
```

Microsoft (R) Macro Assembler Version 5.10 10/26/20 05:39:3

Symbols-2

153 Source Lines
153 Total Lines
32 Symbols

48040 + 457170 Bytes symbol space free

0 Warning Errors
0 Severe Errors