МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

отчет

по лабораторной работе №3

по дисциплине «Организация ЭВМ и систем»

Тема: Представление и обработка целых чисел. Организация ветвящихся процессов

Студент гр. 9382 _____ Михайлов Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться представлять и обрабатывать целые числа, а также организовывать ветвящиеся процесса

Основные теоретические положения.

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)

Б) значения результирующей функции res = f3(i1, i2, k), где вид функций f1 и f2 определяется из табл.2, а функция f3 – из табл.3 по цифрам шифра индивидуального задания (n1, n2, n3) приведенным в таблице 4.

Значения a, b, i, k являются исходными данными, которые должны выбираться студентов 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 26

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

f8=< / - (6*i+8) , при a>b
\ 6*i -10 , при a<=b

f1=< /min(i1,i2), при k=0
\ max(i1,i2), при k/=0

Ход работы:

В сегменте данных объявлены переменные a, b, i, k, i1, i2, res. Функции и ветвления реализованы через метки. Возвращаемые значения записываются в сегменте данных под соответствующей переменной или записываются в регистры. Для реализации ветвления использовалась команда CMP, она сравнивает два числа. В зависимости от результата сравнения, выполняется переход на ту или иную метку.

Исходный код программы.

```
STACKSG SEGMENT  PARA STACK 'Stack'
         DW       32 DUP(?)
STACKSG       ENDS

DATASG  SEGMENT  PARA 'Data'
    a       DW -1h
    b       DW 1h
    i       DW 1h
    k        DW  -1h
    i1       DW  1h
    i2       DW  1h
    res      DW  1h
DATASG ENDS

CODE      SEGMENT
ASSUME  DS:DATASG, CS:CODE

Main    PROC  FAR
       mov  ax, DATASG
       mov  ds, ax

f1:
    mov ax, a           ; переменная a в ax
    cmp       ax, b         ; сравниваем переменные a и b соответственно
    jle f1_jle          ; a <= b

                        ; если попали сюда, то a > b
    mov ax, i           ; переменная i в ax
    add ax, 1           ; добавим к i 1 -> (i+1)
    shl ax, 1           ; домножаем ax на 2 -> 2(i+1)
    mov bx, 4           ; положили в bx 4
    sub ax, bx          ; ax - bx =  2(i+1) - 4
    mov i1, ax          ; записываем в i1 результат 2(i+1) - 4
    jmp f2              ; переходим к f2

f1_jle:                 ; a <= b
    mov ax, i           ; переменная i в ax
    add ax, 1        ; положили в ax (i+1)
    mov bx, ax              ; положили в bx ax
    shl ax, 1           ; домножим ax на 2 -> 2(i+1)
    add ax, bx              ; прибавили к ax bx -> 3(i+1)
    mov bx, 5           ; положили 5 в bx
```

```
    sub bx, ax          ; bx - ax = 5 - 3(i+1)
    mov i1, bx          ; записываем в i1 результат 5 - 3(i+1)
    jmp f2

f2:
    mov ax, a           ; переменная a в ax
    cmp ax, b           ; сравниваем переменные a и b
    jle f2_jle          ; a <= b

                        ; если попали сюда, то a > b

    mov ax, i           ; переменная i в ax
    mov bx, ax          ; кладем в bx i
    shl ax, 1           ; 2i
    add ax, bx          ; 3i
    shl ax, 1           ; 6i
    mov bx, 8           ; кладем 8 в bx
    add ax, bx          ; 6i+8
    mov bx, 0           ; bx = 0
    sub bx, ax          ; делаем ax отрицательным
    mov i2, bx          ; записываем в i2 результат -(6 * i + 8)
    jmp f3              ; переходим к f3


f2_jle:
    mov ax, i           ; переменная i в ax
    sub ax, 1           ; отняли от i 1
    mov bx, ax          ; положили в bx ax
    shl ax, 1           ; 2(i-1)
    add ax, bx          ; 3(i-1)
    mov bx, 9           ; bx = 9
    sub bx, ax          ; 9 -3*(i-1)
    mov i2, bx          ; записываем в i2 результат 9 -3*(i-1)
    jmp f3

f3:
    mov ax, k           ; кладем в ax переменную k
    cmp ax, 0           ; сравним k с 0
    je f3_je            ; k = 0

                        ; часть, где k /= 0
    mov ax, i1          ; кладем в ax переменную i1
    mov bx, i2          ; кладем в bx переменную i2
    cmp ax, bx          ; сравним i1 с i2
    jle f3_jle          ; если i1 <= i2
    jmp f3_jl_result_jge


f3_jle:
    mov ax, i2          ; кладем в ax переменную i2
    jmp f3_jl_result_jge


f3_je:
    mov ax, i1          ; кладем в ax переменную i1
    mov bx, i2          ; кладем в ax переменную i1
    cmp ax, bx          ; сравним i1 с i2
    jge f3_jge          ; если i1 >= i2
```

```
    jmp f3_jl_result_jge


f3_jge:
    mov ax, i2          ; кладем в ax переменную i2
    jmp f3_jl_result_jge

f3_jl_result_jge:
    mov res, ax         ; максимум или минимум в зависимости от условия
    jmp end_f           ; завершаем программу


end_f:
    mov ah, 4ch         ; и наконец завершим программу
    int 21h


Main      ENDP
CODE      ENDS
END Main
```

Листинг программы.

```
#Microsoft (R) Macro Assembler Version 5.10          11/4/20 15:20:26
                                                     Page     1-1


 0000                    STACKSG SEGMENT  PARA STACK 'Stack'
 0000  0020[                     DW      32 DUP(?)
       ????
                        ]

 0040                    STACKSG     ENDS

 0000                    DATASG  SEGMENT  PARA 'Data'

 0000  FFFF                a       DW -1h
 0002  0001                b       DW 1h
 0004  0001                i       DW 1h
 0006  FFFF                k      DW  -1h
 0008  0001                i1     DW  1h
 000A  0001                i2     DW  1h
```

```
000C  0001                    res     DW  1h
000E                          DATASG      ENDS


0000                          CODE      SEGMENT

                              ASSUME  DS:DATASG, CS:CODE

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

0005                          f1:
0005  A1 0000 R                 mov ax, a         ; переменная a
                               в ax
0008  3B 06 0002 R              cmp ax, b          ; сравниваем

                               енно
000C  7E 13                     jle f1_jle        ; a <= b

                                                  ; если попали
                               сюда, то a > b
000E  A1 0004 R                 mov ax, i         ; переменная i
                               в ax
0011  05 0001                             add ax, 1         ; добавим к i 1
                               -> (i+1)
0014  D1 E0                     shl ax, 1         ; домножаем ax
                               на 2 -> 2(i+1)
0016  BB 0004                             mov bx, 4         ; положили в bx
                                4
0019  2B C3                     sub ax, bx        ; ax - bx =  2(i+1) - 4
001B  A3 0008 R                 mov i1, ax        ; записываем

001E  EB 19 90                            jmp f2            ; переходим к
                               f2
```

```
0021                    f1_jle:         ; a <= b
0021  A1 0004 R             mov ax, i      ; переменная i
                            в ax
```

```
0024  05 0001                    add ax, 1          ; положили в ax
                                (i+1)
0027  8B D8                      mov bx, ax         ; положили в bx
                                ax
0029  D1 E0                      shl ax, 1          ; домножим ax н
                                a 2 -> 2(i+1)
002B  03 C3                      add ax, bx         ; прибавили к
                                ax bx -> 3(i+1)
002D  BB 0005                    mov bx, 5          ; положили 5 в
                                bx
0030  2B D8                      sub bx, ax         ; bx - ax = 5 - 3(i+1)
0032  89 1E 0008 R               mov i1, bx         ; записываем

0036  EB 01 90                   jmp f2

0039                             f2:
0039  A1 0000 R                  mov ax, a          ; переменная a
                                в ax
003C  3B 06 0002 R               cmp ax, b          ; сравниваем

0040  7E 1C                      jle f2_jle         ; a <= b

                                                    ; если попали
                                сюда, то a > b

0042  A1 0004 R                  mov ax, i          ; переменная i
                                в ax
0045  8B D8                      mov bx, ax         ; кладем в bx i
0047  D1 E0                      shl ax, 1          ; 2i
0049  03 C3                      add ax, bx         ; 3i
004B  D1 E0                      shl ax, 1          ; 6i
```

```
004D  BB 0008                   mov bx, 8         ; кладем 8 в bx
0050  03 C3                    add ax, bx        ; 6i+8
0052  BB 0000                   mov bx, 0         ; bx = 0
0055  2B D8                    sub bx, ax        ; делаем ax от
                               рицательным
0057  89 1E 000A R             mov i2, bx        ; записываем

005B  EB 19 90                  jmp f3            ; переходим к
                               f3


005E                           f2_jle:
005E  A1 0004 R                mov ax, i         ; переменная i
                               в ax
0061  2D 0001                   sub ax, 1         ; отняли от i 1
0064  8B D8                    mov bx, ax        ; положили в bx
                               ax
0066  D1 E0                    shl ax, 1         ; 2(i-1)
0068  03 C3                    add ax, bx        ; 3(i-1)
006A  BB 0009                   mov bx, 9         ; bx = 9
006D  2B D8                    sub bx, ax        ; 9 -3*(i-1)
006F  89 1E 000A R             mov i2, bx        ; записываем
```

```
0073  EB 01 90                    jmp f3

0076                        f3:
0076  A1 0006 R                 mov ax, k       ; кладем в ax п

0079  3D 0000                       cmp ax, 0       ; сравним k с 0
007C  74 14                     je f3_je        ; k = 0

                                        ; часть, где k /
                            = 0
007E  A1 0008 R                 mov ax, i1      ; кладем в ax п

0081  8B 1E 000A R              mov bx, i2      ; кладем в bx п

0085  3B C3                     cmp ax, bx      ; сравним i1 с i2

0087  7E 03                     jle f3_jle      ; если i1 <= i2
0089  EB 1B 90                      jmp f3_jl_result_jge


008C                        f3_jle:
008C  A1 000A R                 mov ax, i2      ; кладем в ax п

008F  EB 15 90                      jmp f3_jl_result_jge


0092                        f3_je:
0092  A1 0008 R                 mov ax, i1      ; кладем в ax п

0095  8B 1E 000A R              mov bx, i2      ; кладем в ax п
```

```
0099  3B C3                 cmp ax, bx        ; сравним i1 с i2

009B  7D 03                 jge f3_jge        ; если i1 >= i2
009D  EB 07 90                    jmp f3_jl_result_jge


00A0                        f3_jge:
00A0  A1 000A R             mov ax, i2        ; кладем в ax п

00A3  EB 01 90                    jmp f3_jl_result_jge

00A6                        f3_jl_result_jge:
00A6  A3 000C R             mov res, ax       ; максимум ил

                            условия
00A9  EB 01 90                    jmp end_f         ; завершаем п


00AC                        end_f:
00AC  B4 4C                 mov ah, 4ch       ; и наконец за
                            вершим программу
00AE  CD 21                 int 21h
```

```
00B0                    Main    ENDP
00B0                    CODE    ENDS
                        END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
CODE	00B0	PARA	NONE	
DATASG	000E	PARA	NONE	'DATA'
STACKSG	0040	PARA	STACK	'STACK'

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATASG
B	L WORD	0002	DATASG
END_F	L NEAR	00AC	CODE
F1	L NEAR	0005	CODE
F1_JLE	L NEAR	0021	CODE
F2	L NEAR	0039	CODE
F2_JLE	L NEAR	005E	CODE
F3	L NEAR	0076	CODE
F3_JE	L NEAR	0092	CODE
F3_JGE	L NEAR	00A0	CODE
F3_JLE	L NEAR	008C	CODE
F3_JL_RESULT_JGE	L NEAR	00A6	CODE
I	L WORD	0004	DATASG
I1	L WORD	0008	DATASG
I2	L WORD	000A	DATASG

```
K . . . . . .          L WORD      0006   DATASG

MAIN . . . . . .       F PROC      0000   CODE Length = 00B0

RES . . . . . .        L WORD      000C   DATASG

@CPU . . . . . .       TEXT  0101h
@FILENAME . . . . . .          TEXT  VBN
@VERSION . . . . . .   TEXT  510


   120 Source  Lines
   120 Total   Lines
    28 Symbols

  48012 + 453103 Bytes symbol space free

     0 Warning Errors
     0 Severe  Errors
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
CODE	00B4	PARA	NONE	
DATASG	000E	PARA	NONE	'DATA'
STACKSG	0040	PARA	STACK	'STACK'

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATASG
B	L WORD	0002	DATASG
END_F	L NEAR	00B0	CODE
F1	L NEAR	0005	CODE
F1_JLE	L NEAR	001F	CODE
F2	L NEAR	003B	CODE
F2_JLE	L NEAR	0060	CODE
F3	L NEAR	007A	CODE
F3_JE	L NEAR	0096	CODE
F3_JGE	L NEAR	00A4	CODE
F3_JLE	L NEAR	0090	CODE
F3_JL_RESULT_JGE	L NEAR	00AA	CODE
I	L WORD	0004	DATASG
I1	L WORD	0008	DATASG
I2	L WORD	000A	DATASG

```
K . . . . . .          L WORD      0006   DATASG

MAIN . . . . . .       F PROC      0000   CODE Length = 00B4

RES . . . . . .        L WORD      000C   DATASG

@CPU . . . . . .       TEXT  0101h
@FILENAME . . . . . .          TEXT  ZXC
@VERSION . . . . . .   TEXT  510


   122 Source  Lines
   122 Total   Lines
    28 Symbols

  48000 + 426521 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```

Выводы

В ходе выполения данной лабораторной работы была изучена организация ветвления, а так же операция сравнения, реализация меток и перехода по данным меткам на языке Ассемблера. В ходе разработки программы была применена минимизация кода. Результаты вычислений контролировались в отладчике