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 _____ Иерусалимов Н.

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 16

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

$$f6 = \begin{cases} 2*(i+1)-4, & \text{при } a>b \\ 5 - 3*(i+1), & \text{при } a<=b \end{cases}$$

$$f4 = \begin{cases} \min(|i1-i2|,2), & \text{при } k<0 \\ \max(-6, -i2), & \text{при } k>=0 \end{cases}$$

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.

Тестирование.

№	Входные данные	Выходные данные	Верность Вых.Д
1	**a=1, b=1, i=1, k=1**	**i1=2, i2=-1, res=1**	**+**
2	**a=1, b=1, i=1, k=-1**	**i1=2, i2=-1, res=2**	**+**
3	**a=2, b=1, i=1, k=1**	**i1=3, i2=0, res=0**	**+**
4	**a=2, b=1, i=1, k=-1**	**i1=3, i2=0, res=2**	**+**
5	**a=-1, b=1, i=1, k=1**	**i1=2, i2=-1, res=1**	**+**
6	**a=-1, b=1, i=1, k=-1**	**i1=2, i2=-1, res=2**	**+**

Выводы.

Были изучены режимы адресации, определены ошибки в программе, и было дано объяснение ошибкам.

Исходный код программы.

Приложение А

Название файла:main1.asm

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

DATASG  SEGMENT  PARA 'Data'                    ;SEG DATA
     a          DW      2h
     b          DW      4h
     i          DW      1h
     k      DW  3h
     i1     DW  ?
     i2     DW  ?
     res    DW  ?
DATASG      ENDS                                        ;ENDS DATA

CODE      SEGMENT                                   ;SEG CODE
ASSUME  DS:DATASG, CS:CODE

Main    PROC  FAR
      mov  ax, DATASG
      mov  ds, ax

f1:
    mov ax, a ;записали в ax a
    cmp ax, b   ;сравнили a с b
    jle f1_jle      ;a<=b

    ;a>b(jg)
    mov ax, i   ;в ax засунули i
    shl ax, 1   ;умножили ax на 2
    mov i2, ax  ;записали в i2 ax
    mov bx, -2  ; расскрыли скобки из второй функции получили -2 записали ее
в bx
    sub i2, bx  ; отняли от i2 bx и записали ответ в i2
    ;end f2

    shl ax, 1   ; умножили ax на 2
    mov bx, 7   ; раскрыли скобки получили 7 записали ее
    sub bx, ax  ;отнял от 7 ax
    mov i1, bx  ;записали ответ в i1
    jmp f3

f1_jle: ;a <= b
    mov ax, i
    mov bx, 2
    mov i2, ax
    shl ax, 1
    add i2, ax
    neg i2
    add i2, bx
```

```
    ;end f2

    mov bx, i
    shl ax, 1
    shl bx, 1
    add ax, bx
    mov bx, 8
    sub bx, ax
    mov i1, bx
    jmp f3

f3:
     mov ax, k          ; кладем в ax переменную k
     cmp ax, 0          ; сравним k с 0
     jge f3_jge         ; k >= 0

                        ; если оказались здесь, то k < 0 (jl)
     mov ax, i1         ; кладем в ax переменную i1
     sub ax, i2         ; ax = i1 - i2

     cmp ax, 0          ; сравним i1 - i2 с нулем
     jl f3_ABS          ; если i1 - i2 < 0, то стоит взять модуль
     jmp f3_jl_result   ; переход в f3_jl_result

f3_ABS:
     neg ax             ; взяли модуль i1 - i2

f3_jl_result:
     cmp ax, 2h         ; сравним |i1 - i2| с 2
     jge f3_jl_result_jge; если |i1 - i2| >= 2, переместимся в
f3_jl_result_jge
     mov res, ax        ; |i1 - i2| < 2 => res = |i1 - i2|
     jmp end_f          ; завершаем программу

f3_jl_result_jge:
     mov res, 2h        ; |i1 - i2| >= 2 => res = 2
     jmp end_f          ; завершаем программу

f3_jge:                 ; k >= 0
     mov ax, i2         ; кладем в ax переменную i2
     neg ax             ; ax = -ax
     cmp ax, -6h        ; сравниваем ax, -6
     jle f3_jge_jle     ; если -i2 <= -6, переместимся в f3_jge_jle
     mov res, ax        ; -i2 > -6 => res = -i2
     jmp end_f          ; завершаем программу

f3_jge_jle:
     mov res, -6h       ; -i2 <= -6 => res = -6

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


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

Листинг программы

Приложение В

Название файла:main.lst

```
__Microsoft (R) Macro Assembler Version 5.10                 12/2/20 18:10:06
                                                             Page     1-1


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

0040                    STACKSG     ENDS

0000                    DATASG  SEGMENT  PARA 'Data'
                                ;SEG DATA
0000  0002                   a          DW      2h
0002  0004                   b          DW      4h
0004  0001                   i          DW      1h
0006  0003                   k      DW  3h
0008  0000                   i1     DW  ?
000A  0000                   i2     DW  ?
000C  0000                   res    DW  ?
000E                    DATASG      ENDS
                                            ;ENDS DATA

0000                    CODE      SEGMENT
                                        ;SEG CODE
                        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 ;записали в ax a
0008  3B 06 0002 R                cmp ax, b   ;сравнили a с b
000C  7E 1D                       jle f1_jle      ;a<=b

                            ;a>b(jg)
000E  A1 0004 R             mov ax, i   ;в ax засунули i
0011  D1 E0                       shl ax, 1   ;умножили ax на 2
0013  A3 000A R             mov i2, ax  ;записали в i2 ax
0016  BB FFFE                     mov bx, -2  ; расскрыли скобки из второй фу
                        нкции получили -2 записали ее в bx
0019  29 1E 000A R                sub i2, bx  ; отняли от i2 bx и записали от
                        вет в i2
                            ;end f2

001D  D1 E0                       shl ax, 1   ; умножили ax на 2
001F  BB 0007                     mov bx, 7   ; раскрыли скобки получили 7 за
                        писали ее
0022  2B D8                       sub bx, ax  ;отнял от 7 ax
0024  89 1E 0008 R                mov i1, bx  ;записали ответ в i1
0028  EB 2E 90                    jmp f3
```

```
002B                    f1_jle: ;a <= b
002B  A1 0004 R             mov ax, i
002E  BB 0002                     mov bx, 2
0031  A3 000A R             mov i2, ax
0034  D1 E0                       shl ax, 1
```


```
0036  01 06 000A R                add i2, ax
003A  F7 1E 000A R                neg i2
003E  01 1E 000A R                add i2, bx
                              ;end f2

0042  8B 1E 0004 R                mov bx, i
0046  D1 E0                       shl ax, 1
0048  D1 E3                       shl bx, 1
004A  03 C3                       add ax, bx
004C  BB 0008                     mov bx, 8
004F  2B D8                       sub bx, ax
0051  89 1E 0008 R                mov i1, bx
0055  EB 01 90                    jmp f3

0058                    f3:
0058  A1 0006 R              mov ax, k          ; кладем в ax переменну
                        ю k
005B  3D 0000                      cmp ax, 0          ; сравним k с 0
005E  7D 25                        jge f3_jge         ; k >= 0

                                                ; если оказались здесь,
                         то k < 0 (jl)
0060  A1 0008 R              mov ax, i1         ; кладем в ax переменну
                        ю i1
0063  2B 06 000A R                 sub ax, i2         ; ax = i1 - i2

0067  3D 0000                      cmp ax, 0          ; сравним i1 - i2 с нул
                        ем
006A  7C 03                        jl f3_ABS          ; если i1 - i2 < 0, то
                        стоит взять модуль
006C  EB 03 90                     jmp f3_jl_result   ; переход в f3_jl_resul
                        t

006F                    f3_ABS:
006F  F7 D8                        neg ax             ; взяли модуль i1 - i2

0071                    f3_jl_result:
0071  3D 0002                      cmp ax, 2h         ; сравним |i1 - i2| с 2
0074  7D 06                        jge f3_jl_result_jge; если |i1 - i2| >= 2,
                         переместимся в f3_jl_result_jge
0076  A3 000C R              mov res, ax        ; |i1 - i2| < 2 => res
                        = |i1 - i2|
0079  EB 20 90                     jmp end_f          ; завершаем программу

007C                    f3_jl_result_jge:
007C  C7 06 000C R 0002            mov res, 2h        ; |i1 - i2| >= 2 => res
                         = 2
0082  EB 17 90                     jmp end_f          ; завершаем программу
```

```
 0085                   f3_jge:                 ; k >= 0
 0085  A1 000A R              mov ax, i2        ; кладем в ax переменну
                        ю i2
 0088  F7 D8                        neg ax            ; ax = -ax
 008A  3D FFFA                      cmp ax, -6h       ; сравниваем ax, -6
__Microsoft (R) Macro Assembler Version 5.10                 12/2/20 18:10:06
                                                            Page     1-3


 008D  7E 06                        jle f3_jge_jle    ; если -i2 <= -6, перем
                        естимся в f3_jge_jle
 008F  A3 000C R              mov res, ax       ; -i2 > -6 => res = -i2
 0092  EB 07 90                     jmp end_f         ; завершаем программу

 0095                   f3_jge_jle:
 0095  C7 06 000C R FFFA            mov res, -6h      ; -i2 <= -6 => res = -6

 009B                   end_f:
 009B  B4 4C                        mov ah, 4ch       ; и наконец завершим пр
                        ограмму
 009D  CD 21                        int 21h


 009F                   Main      ENDP
 009F                   CODE      ENDS
                        END Main                     ;ENDS C
                        ODE
__Microsoft (R) Macro Assembler Version 5.10                 12/2/20 18:10:06
                                                            Symbols-1


Segments and Groups:

                N a m e             Length       Align      Combine Class

CODE . . . . . . . . . . . . . .    009F  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      009B  CODE

F1 . . . . . . . . . . . . . . .    L NEAR      0005  CODE
F1_JLE . . . . . . . . . . . . .    L NEAR      002B  CODE
F3 . . . . . . . . . . . . . . .    L NEAR      0058  CODE
F3_ABS . . . . . . . . . . . . .    L NEAR      006F  CODE
F3_JGE . . . . . . . . . . . . .    L NEAR      0085  CODE
F3_JGE_JLE . . . . . . . . . . .    L NEAR      0095  CODE
F3_JL_RESULT . . . . . . . . . .    L NEAR      0071  CODE
```

```
F3_JL_RESULT_JGE . . . . . .    L NEAR      007C  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 = 009F

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

@CPU . . . . . .    TEXT  0101h
@FILENAME  . . . . . .    TEXT  MAIN1
@VERSION . . . . . .    TEXT  510


    104 Source  Lines
    104 Total   Lines
     27 Symbols

  47978 + 457232 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors
```