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 _____ Русинов 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 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.

Исходный код программы.

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

DATASG  SEGMENT  PARA 'Data'            ;SEG DATA
    a       DW 3h
    b       DW 2h
    i       DW 3h
    k        DW  1h
    i1       DW  1h
    i2       DW  1h
    res      DW  1h
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           ; переменная a в ax
    cmp    ax, b         ; сравниваем переменные a и b соответственно
    jle f1_jle          ; a <= b

                        ; если попали сюда, то a > b (ja)
    mov ax, i           ; переменная i в ax
    mov bx, 4h          ; 4 в bx
    mul bx;             ; ax = ax * bx = i * 4
    mov bx, 7h          ; кладем в bx 7
    sub bx, ax          ; bx - ax = 7 - i * 4
    mov i1, bx          ; записываем в i1 результат 7 - i * 4
    jmp f2              ; переходим к f2

f1_jle:                 ; a <= b
    mov ax, i           ; переменная i в ax
    mov bx, 6h          ; 6 в bx
    mul bx;             ; ax = ax * bx = i * 6
    mov bx, 8h          ; кладем в bx 8
```

```
    sub bx, ax          ; bx - ax = 8 - i * 6
    mov i1, bx          ; записываем в i1 результат 8 - i * 6
    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 (ja)
    mov ax, 1h          ; кладем в ax 1
    add ax, i           ; ax = 1 + i
    mov bx, 2h          ; кладем в bx 2
    mul bx              ; ax = ax * bx = (1 + i) * 2
    sub ax, 4h          ; ax = ax - 4
    mov i2, ax          ; записываем в i2 результат (1 + i) * 2 - 4
    jmp f3              ; переходим к f3

f2_jle:
    mov ax, 1h          ; кладем в ax 1
    add ax, i           ; ax = 1 + i
    mov bx, 3h          ; кладем в bx 3
    mul bx              ; ax = ax * bx = (1 + i) * 3
    mov bx, 5h          ; кладем в bx 5
    sub bx, ax          ; bx - ax = 5 - (1 + i) * 3
    mov i2, bx          ; записываем в i2 результат 5 - (1 + i) * 3
    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
    je f3_jl_result_jge; если |i1 - i2| >= 2, переместимся в f3_jl_re-
sult_jge
    mov res, 2h         ; |i1 - i2| < 2 => res = 2
    jmp end_f           ; завершаем программу

f3_jl_result_jge:
    mov res, ax         ; |i1 - i2| >= 2 => res = |i1 - i2|
    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_jae_jle      ; если -i2 <= -6, переместимся в f3_jae_jle
    mov res, ax         ; -i2 > -6 => res = -i2
    jmp end_f           ; завершаем программу

f3_jae_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.

Microsoft (R) Macro Assembler Version 5.10 10/13/20 03:38:2

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  0003                a       DW 3h
0002  0002                b       DW 2h
0004  0003                i       DW 3h
0006  0001                k      DW  1h
0008  0001                i1     DW  1h
000A  0001                i2     DW  1h
000C  0001                res    DW  1h
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         ; –њ–µ—А–µ–Ј–∞–њ–Є—Б–Є–Є a
                        –≤ ax
0008  3B 06 0002 R        cmp ax, b                ; —Б—А–∞–≤–љ–µ–љ–Є–µ a –Є b —Б–Њ–Њ—В–≤–µ—В—Б—В–≤–µ–љ–љ–Њ
–Ј–љ–∞–µ–љ–Є—П
                        —В–µ–Ї—Г—Й–Є–µ –Ј–љ–∞—З–µ–љ–Є—П
000C  7E 14               jle f1_jle        ; a <= b

                                  ; –µ—Б–ї–Є –љ–µ—А–∞–≤–µ–љ—Б—В–≤–Њ
                        —Б—О–і–∞, —В–Њ a > b (ja)
000E  A1 0004 R           mov ax, i         ; –њ–µ—А–µ–Ј–∞–њ–Є—Б–Є i
                        –≤ ax
0011  BB 0004                   mov bx, 4h        ; 4 –≤ bx
0014  F7 E3               mul bx;           ; ax = ax * bx = i * 4
0016  BB 0007                   mov bx, 7h        ; –Ї–Њ–љ—Б—В–∞–љ—В–∞ –≤ bx 7
0019  2B D8               sub bx, ax        ; bx - ax = 7 - i * 4
001B  89 1E 0008 R        mov i1, bx        ; –Ј–∞–њ–Є—Б—М —А–µ–Ј—Г–ї—М—В–∞—В–∞ –≤
i1 —А–µ–Ј—Г–ї—М—В–∞—В 7 - i * 4
001F  EB 15 90                  jmp f2            ; –њ–µ—А–µ—Е–Њ–і –Ї
f2

0022                    f1_jle:           ; a <= b
0022  A1 0004 R           mov ax, i         ; –њ–µ—А–µ–Ј–∞–њ–Є—Б–Є i
                        –≤ ax
0025  BB 0006                   mov bx, 6h        ; 6 –≤ bx
```

```
0028  F7 E3                 mul bx;          ; ax = ax * bx = i * 6
002A  BB 0007                     mov bx, 7h       ; –Ї–ї–∞–і–Ь –≤ bx 8
Microsoft (R) Macro Assembler Version 5.10                10/13/20 03:38:2
                                        Page     1-2


002D  2B D8                   sub bx, ax       ; bx - ax = 8 - i * 6
002F  89 1E 0008 R            mov i1, bx       ; –J–∞–њ–Є—Б—Б—Л–≤–∞–µ—В –
                              ≤ i1 —А–µ–Ј—Г–ї—М—В–∞—В 8 - i * 6
0033  EB 01 90                    jmp f2

0036                          f2:
0036  A1 0000 R               mov ax, a        ; –њ–µ—А–µ–Д–∞–µ—В –∞–і—А–µ—Б –
                              –≤ ax
0039  3B 06 0002 R            cmp ax, b        ; —Б—А–∞–≤–љ–Є–≤–∞–µ—В –
                              –њ–µ—А–µ–Д–∞–µ—В—Л—Л–µ a –Є b
003D  7E 15                   jle f2_jle       ; a <= b

                                    ; –µ—Б–ї–Є –љ–µ –≤—Л–њ–Њ–ї–љ–µ–Є–µ
                              —Б—О–і–∞, —В–Њ a > b (ja)
003F  B8 0001                      mov ax, 1h       ; –Ї–ї–∞–і–µ—Л–≤–∞–µ—В –≤ ax 1
0042  03 06 0004 R            add ax, i        ; ax = 1 + i
0046  BB 0002                      mov bx, 2h       ; –Ї–ї–∞–і–µ—Л–≤–∞–µ—В –≤ bx 2
0049  F7 E3                   mul bx           ; ax = ax * bx = (1 + i)
                               * 2
004B  2D 0004                      sub ax, 4h       ; ax = ax - 4
004E  A3 000A R                mov i2, ax       ; –J–∞–њ–Є—Б—Б—Л–≤–∞–µ—В –
                              ≤ i2 —А–µ–Ј—Г–ї—М—В–∞—В (1 + i) * 2 - 4
0051  EB 19 90                    jmp f3           ; –њ–µ—А–µ–Е–Њ–і–Є–Є—Б–Ь –Ї
                              f3

0054                          f2_jle:
0054  B8 0001                      mov ax, 1h       ; –Ї–ї–∞–і–µ—Л–≤–∞–µ—В –≤ ax 1
0057  03 06 0004 R             add ax, i        ; ax = 1 + i
005B  BB 0003                      mov bx, 3h       ; –Ї–ї–∞–і–µ—Л–≤–∞–µ—В –≤ bx 3
```

```
005E  F7 E3                   mul bx            ; ax = ax * bx = (1 + i)
                             * 3
0060  BB 0005                       mov bx, 5h        ; –Ї–ї–∞–і–µ–Ъ –≤ bx 5
0063  2B D8                   sub bx, ax        ; bx - ax = 5 - (1 + i)
                             * 3
0065  89 1E 000A R            mov i2, bx        ; –J–∞–нь–Є—Б—Л–≤–∞–µ–Ъ –
                             ≤ i2 —А–µ–J—Г–ї—М—В–∞—В 5 - (1 + i) * 3
0069  EB 01 90                      jmp f3

006C                          f3:
006C  A1 0006 R               mov ax, k         ; –Ї–ї–∞–і–µ–Ъ –≤ ax –нь–
                             µ—А–µ–Ъ–µ–ль–ъ—Г—О k
006F  3D 0000                       cmp ax, 0         ; —Б—А–∞–≤–љ–Є–Ъ k —Б 0
0072  7D 25                   jge f3_jge        ; k >= 0

                                    ; –µ—Б–ї–Є –Нь–Ї–∞–J–∞–ї–
                             Є—Б—М –J–і–µ—Б—М, —В–Нь k < 0 (jl)
0074  A1 0008 R               mov ax, i1        ; –Ї–ї–∞–і–µ–Ъ –≤ ax –нь–
                             µ—А–µ–Ъ–µ–ль–ъ—Г—О i1
0077  2B 06 000A R            sub ax, i2        ; ax = i1 - i2

007B  3D 0000                       cmp ax, 0         ; —Б—А–∞–≤–љ–Є–Ъ i1 - i2
                             —Б –ль—Г–ї–µ–Ъ
007E  7C 03                   jl f3_ABS         ; –µ—Б–ї–Є i1 - i2 < 0,
                             —В–Нь —Б—В–Нь–Є—В –≤–J—П—В—М –Ъ–Нь–і—Г–ї—
M
```



```
0080  EB 03 90                      jmp f3_jl_result  ; –нь–µ—А–µ—Е–Нь–і –≤ f3_j
                             l_result

0083                          f3_ABS:
0083  F7 D8                   neg ax            ; –≤–J—П–Є –Ъ–Нь–і—Г–ї—
```

```
                         M i1 - i2

0085                     f3_jl_result:
0085  3D 0002                      cmp ax, 2h        ; —Б—А–∞–≤–љ–Є–Љ |i1 - i
                         2| —Б 2
0088  74 09                 je f3_jl_result_jge; –µ—Б–ї–Є |i1 - i2| >=
                         2, –њ–µ—А–µ–є–і–µ–Љ –Ї f3_jl_result_jge
008A  C7 06 000C R 0002     mov res, 2h       ; |i1 - i2| < 2 => res =
                          2
0090  EB 1D 90                     jmp end_f         ; –ј–∞–≤–µ—А—И–∞–µ–Љ –њ—
                         А–Њ–≥—А–∞–Љ–Љ—Г

0093                     f3_jl_result_jge:
0093  A3 000C R             mov res, ax       ; |i1 - i2| >= 2 => res
                         = |i1 - i2|
0096  EB 17 90                     jmp end_f         ; –ј–∞–≤–µ—А—И–∞–µ–Љ –њ—
                         А–Њ–≥—А–∞–Љ–Љ—Г

0099                     f3_jge:              ; k >= 0
0099  A1 000A R             mov ax, i2        ; –Ї–ї–∞–і–µ–Љ –≤ ax –њ–
                         µ—А–µ–Љ–µ–љ–љ—Г—О i2
009C  F7 D8                 neg ax            ; ax = -ax
009E  3D FFFA                      cmp ax, -6h       ; —Б—А–∞–≤–љ–Є–≤–∞–µ–Љ a
                         x, -6
00A1  7E 06                 jle f3_jae_jle    ; –µ—Б–ї–Є -i2 <= -6, –њ
                         –µ—А–µ–є–і–µ–Љ –Ї f3_jae_jle
00A3  A3 000C R             mov res, ax       ; -i2 > -6 => res = -i2
00A6  EB 07 90                     jmp end_f         ; –ј–∞–≤–µ—А—И–∞–µ–Љ –њ—
                         А–Њ–≥—А–∞–Љ–Љ—Г

00A9                     f3_jae_jle:
00A9  C7 06 000C R FFFA     mov res, -6h      ; -i2 <= -6 => res = -6

00AF                     end_f:
00AF  B4 4C                 mov ah, 4ch       ; –Є –љ–∞–Ї–Њ–љ–µ—Ж –≤–∞
```

```
                                     –≤–µ—А—И–Є–Љ –њ—А–Њ–≥—А–∞–Љ–Љ—Г
00B1  CD 21                           int 21h


00B3                                 Main      ENDP
00B3                                 CODE      ENDS
                                     END Main                       ;ENDS C
                                     ODE
□Microsoft (R) Macro Assembler Version 5.10              10/13/20 03:38:2
                                               Symbols-1
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
CODE	00B3	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	00AF	CODE
F1	L NEAR	0005	CODE
F1_JLE	L NEAR	0022	CODE
F2	L NEAR	0036	CODE
F2_JLE	L NEAR	0054	CODE
F3	L NEAR	006C	CODE

```
F3_ABS . . . . . .      L NEAR      0083   CODE
F3_JAE_JLE . . . . . .  L NEAR      00A9   CODE
F3_JGE . . . . . .      L NEAR      0099   CODE
F3_JL_RESULT . . . . . .        L NEAR      0085   CODE
F3_JL_RESULT_JGE . . . . . .    L NEAR      0093   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 = 00B3

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

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

Microsoft (R) Macro Assembler Version 5.10              10/13/20 03:38:2

                                        Symbols-2

  113 Source  Lines
  113 Total   Lines
   29 Symbols

 47962 + 453153 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```

Тестирование.

№	Входные данные	Выходные данные	Правильный результат
1	a=1, b=1, i=1, k=1	i1=2, i2=-1, res=1	i1=2, i2=-1, res=1
2	a=1, b=1, i=1, k=-1	i1=2, i2=-1, res=2	i1=2, i2=-1, res=2
3	a=2, b=1, i=1, k=1	i1=3, i2=0, res=0	i1=3, i2=0, res=0
4	a=2, b=1, i=1, k=-1	i1=3, i2=0, res=2	i1=3, i2=0, res=2
5	a=-1, b=1, i=1, k=1	i1=2, i2=-1, res=1	i1=2, i2=-1 res=1
6	a=-1, b=1, i=1, k=-1	i1=2, i2=-1, res=2	i1=2, i2=-1, res=2

Выводы.

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