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.

Студентка гр. 9383 _____ Сергиенкова А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Познакомиться с представлением и обработкой целых чисел. Написать на языке Ассемблер программу, в которой нужно реализовать ветвящиеся процессы.

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 19.

$$f4 = \begin{cases} -(6*i-4), & \text{при } a>b \\ 3*(i+2), & \text{при } a<=b \end{cases}$$

$$f5 = \begin{cases} 20 - 4*i, & \text{при } a>b \\ -(6*I - 6), & \text{при } a<=b \end{cases}$$

$$f7 = \begin{cases} |i1| + |i2|, & \text{при } k<0 \\ \max(6, |i1|), & \text{при } k>=0 \end{cases}$$

Ход работы.

Первая функция вычисляется следующим образом:
С помощью ветвящегося процесса cmp, который производит сравнение с изменением флага ZF (0 — аргументы не равны, 1 — аргументы равны). Если a<=b, то происходит переход на метку jg, которая является условным переходом по заданной метке, в случае если первый аргумент больше второго после выполнения cmp. В i1 запишется 15. Иначе в i1 запишется -14 и выполнится безусловный переход jmp.

Вторая функция зависит от сравнения в первой, если в первой функции a<=b, то выполнится функция f2_1. В i2 запишется -12. Иначе, выполнится функция f2 и в i2 запишется 8.

Третья функция вычисляется следующим образом:
Мы сравниваем k и 0. Если k<0, то совершается условный переход jl, по заданной метке. Дальше идёт проверка модулей, после чего в res кладётся их сумма. Если же k<=0, то сравниваем модуль i1 с 6 и устанавливаем максимум из них, результат записываем в res.

Тестирование.

1. a = 1, b = 2, i = 3, k = 4 => f1 = 15, f2 = -12, f3 = 15
2. a = 2, b = 1, i = 3, k = 4 => f1 = -14, f2 = 8, f3 = 14
3. a = 2, b = 1, i = 3, k = -2 => f1 = -15, f2 = 8, f3 = 22

Выводы.

Была реализована программа на языке Ассемблер с ветвящимися процессами. Изучены представление и обработка целых чисел.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: lb3.asm

```
AStack SEGMENT STACK
	DW 32 DUP(?)
AStack ENDS

DATA SEGMENT
a	DW	1
b	DW	2
i	DW	3
k	DW	4
i1	DW	?
i2	DW	?
res	DW	?
DATA ENDS

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

Main PROC FAR
	mov ax, DATA
	mov ds, ax


f1:
  mov ax, a
  cmp ax, b ; a<=b
  jg f1_1		; if a>b

  mov ax, i
  mov bx, ax
  shl ax, 1
  add ax, bx	;3*ax
  add ax, 6	;3ax + 6?
  mov i1, ax

jmp f2_1

f1_1:
  mov ax, i
  shl ax, 1
  mov bx, ax	;bx = 2*ax
  shl ax, 1	;ax = 4*ax
  add ax, bx	;ax = 6*ax
  sub ax, 4	;ax = 6*ax - 4
  neg ax
  mov i1, ax
```

```
f2:
  mov ax, i
  shl ax, 1        ;ax = 2*ax
  shl ax, 1        ;ax = 4*ax
  neg ax
  add ax, 20
  mov i2, ax
  jmp f3

f2_1:
  mov ax, i
  shl ax, 1        ; ax = 2*ax
  mov bx, ax
  shl ax, 1
  add ax, bx       ; ax = 6*ax
  sub ax, 6
  neg ax
  mov i2, ax

f3:
mov ax, k
cmp k, 0
jl f3_1

  mov ax, i1
  cmp ax, 0        ;if ax < 0
  jl f_abs
  jmp f3_cmp_6


f3_1:
  mov bx, i1
  cmp bx, 0        ;if i1 < 0
  jl f_abs_1       ;then i1 = |i1|
  jmp f3_2


f_abs:
  neg ax           ;ax = -ax
  jmp f3_res


f_abs_1:
  neg bx           ;i1 = |i1|
  jmp f3_2

f3_2:
  mov cx, i2
  cmp cx, 0
  jl f_abs_2
  jmp f_sum
```

```
f_abs_2:
  neg cx          ;i2 = |i2|
  jmp f_sum

f_sum:
  mov ax, bx    ;ax = i1
  add ax, cx    ;ax = i1 + i2
  jmp f3_res

f3_cmp_6:
  cmp ax, 6     ;if ax < 6
  jl f3_6         ;res = 6


f3_res:
  mov res, ax   ;else res = ax
  jmp f_end


f3_6:
  mov res, 6    ;res = 6

f_end:
  mov ah, 4ch
  int 21h

Main   ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ Б.
ЛИСТИНГ ПРОГРАММЫ.

Название файла: lb3.lst

```
Microsoft (R) Macro Assembler Version 5.10              10/26/20 22:26:2
                                                        Page     1-1


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  0004                      k       DW    4
0008  0000                      i1      DW    ?
000A  0000                      i2      DW    ?
000C  0000                      res     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                            f1:

0005  A1 0000 R                   mov ax, a
0008  3B 06 0002 R                cmp ax, b       ; a<=b
000C  7F 12                       jg f1_1         ; if a>b

000E  A1 0004 R                   mov ax, i
0011  8B D8                       mov bx, ax
0013  D1 E0                       shl ax, 1
0015  03 C3                       add ax, bx ; 3*ax
0017  05 0006                             add ax, 6 ; 3ax + 6?
001A  A3 0008 R                   mov i1, ax

001D  EB 26 90                          jmp f2_1

0020                            f1_1:
0020  A1 0004 R                   mov ax, i
0023  D1 E0                       shl ax, 1
```

```
0025  8B D8                  mov bx, ax   ;bx = 2*ax
0027  D1 E0                  shl ax, 1       ; ax = 4*ax
0029  03 C3                  add ax, bx    ; ax = 6*ax
002B  2D 0004                        sub ax, 4      ;ax = 6*ax - 4
002E  F7 D8                  neg ax
0030  A3 0008 R              mov i1, ax

0033                         f2:
0033  A1 0004 R              mov ax, i
0036  D1 E0                  shl ax, 1       ;ax = 2*ax
0038  D1 E0                  shl ax, 1       ;ax = 4*ax
```



```
003A  F7 D8                  neg ax
003C  05 0014                        add ax, 20
003F  A3 000A R              mov i2, ax
0042  EB 14 90                       jmp f3 ;8

0045                         f2_1:
0045  A1 0004 R              mov ax, i
0048  D1 E0                  shl ax, 1       ; ax = 2*ax
004A  8B D8                  mov bx, ax
004C  D1 E0                  shl ax, 1
004E  03 C3                  add ax, bx ; ax = 6*ax
0050  2D 0006                        sub ax, 6
0053  F7 D8                  neg ax
0055  A3 000A R              mov i2, ax ;-6

0058                         f3:
0058  A1 0006 R                    mov ax, k
005B  83 3E 0006 R 00                     cmp k, 0
0060  7C 0B                        jl f3_1

0062  A1 0008 R              mov ax, i1
0065  3D 0000                        cmp ax, 0     ;if ax < 0
0068  7C 0F                  jl f_abs
006A  EB 2F 90                       jmp f3_cmp_6


006D                         f3_1:
006D  8B 1E 0008 R           mov bx, i1
0071  83 FB 00                       cmp bx, 0  ;if i1 < 0
0074  7C 08                  jl f_abs_1     ;then i1 = |i1|
0076  EB 0B 90                       jmp f3_2


0079                         f_abs:
0079  F7 D8                  neg ax                 ;ax = -ax
007B  EB 23 90                       jmp f3_res
```

```
007E                    f_abs_1:
007E  F7 DB               neg bx                ;i1 = |i1|
0080  EB 01 90                  jmp f3_2

0083                    f3_2:
0083  8B 0E 000A R       mov cx, i2
0087  83 F9 00                  cmp cx, 0
008A  7C 03               jl f_abs_2
008C  EB 06 90                  jmp f_sum

008F                    f_abs_2:
008F  F7 D9               neg cx ;i2 = |i2|
0091  EB 01 90                  jmp f_sum

0094                    f_sum:
0094  8B C3               mov ax, bx    ;ax = i1
```

```
0096  03 C1                add ax, cx     ;ax = i1 + i2
0098  EB 06 90                    jmp f3_res

009B                       f3_cmp_6:
009B  3D 0006                      cmp ax, 6      ;if ax < 6
009E  7C 06                jl f3_6                ;res = 6


00A0                       f3_res:
00A0  A3 000C R            mov res, ax   ;else res = ax
00A3  EB 07 90                     jmp f_end


00A6                       f3_6:
00A6  C7 06 000C R 0006    mov res, 6     ;res = 6

00AC                       f_end:
00AC  B4 4C                mov ah, 4ch
00AE  CD 21                int 21h

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


Segments and Groups:

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

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
B	L WORD	0002	DATA
F1	L NEAR	0005	CODE
F1_1	L NEAR	0020	CODE
F2	L NEAR	0033	CODE
F2_1	L NEAR	0045	CODE
F3	L NEAR	0058	CODE
F3_1	L NEAR	006D	CODE

```
F3_2 . . . . . .          L NEAR      0083   CODE
F3_6 . . . . . .          L NEAR      00A6   CODE
F3_CMP_6 . . . . . .      L NEAR      009B   CODE
F3_RES . . . . . .        L NEAR      00A0   CODE
F_ABS . . . . . .         L NEAR      0079   CODE
F_ABS_1 . . . . . .       L NEAR      007E   CODE
F_ABS_2 . . . . . .       L NEAR      008F   CODE
F_END . . . . . .         L NEAR      00AC   CODE
F_SUM . . . . . .         L NEAR      0094   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 = 00B0

RES . . . . . .           L WORD      000C   DATA

@CPU . . . . . .          TEXT  0101h
@FILENAME . . . . . .           TEXT  lb3
@VERSION . . . . . .      TEXT  510


Microsoft (R) Macro Assembler Version 5.10          10/26/20 22:26:2

                                Symbols-2




  128 Source  Lines
  128 Total   Lines
   31 Symbols

 48042 + 457168 Bytes symbol space free

    0 Warning Errors
    0 Severe  Errors
```