

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

Субботин М.О.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться представлять и обрабатывать целые числа, а также организовывать ветвящиеся процессы.

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 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 20: 4.6.8

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

$$\backslash 3*(i+2), \text{ при } a \leq b$$

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

$$\backslash 5 - 3*(i+1), \text{ при } a \leq b$$

$$f8 = \begin{cases} /|i1|-|i2|, & \text{при } k < 0 \\ \end{cases}$$

$$\backslash \max(4, |i2|-3), \text{ при } k \geq 0$$

Ход выполнения:

В сегменте данных заданы метки для переменных a, b, k, l, i1, i2, res. Функции и их ветвления были реализованы с помощью меток в фрагменте кода. Эти “функции” кладут возвращаемые значения на стек. Для реализации ветвления функций и самих функций использовалась операция CMP, которая сравнивала два числа. В зависимости от сравнения двух чисел с помощью переходов мы переходили на метки, соответствующие конкретному ветвлению в функции. К примеру, если мы сравниваем два числа a и b и хотим, чтобы в случае $a > b$ выполнялась функция по метке f, то следует выполнить:

```
cmp a,b
```

```
    jg f1
```

Если же у нас существует ветвление и при $a \leq b$, то код, выполняемый в таком случае можно просто поместить ниже. Собственно с помощью таких ветвлений и переходов по коду и были реализованы функции с ветвлениями.

Исходный код программы:

```
STACKSG SEGMENT PARA STACK 'Stack'
```

```
    DW    32 DUP(?)
```

```
STACKSG ENDS
```

```
DATASG SEGMENT PARA 'Data' ;SEG DATA
```

```
VARA    DW  1h
```

```
VARB    DW  1h
```

```
VARIDW  1h
```

```
VARK    DW  1h
```

```
VARI1   DW  1h
```

```

VARI2    DW    1h
VARRES   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,VARA      ;переменная а в ах

```

```

    mov si,VARB      ;переменная b в si

```

```

    mov bx,VARI      ;переменная i в bx

```

```

    shl bx,1         ;умножаем переменную i в bx на 2

```

```

    cmp ax,si        ;сравниваем переменные а и b соответственно

```

```

    jg  f1_1         ;если a>b переходим к метке f1_1

```

```

f1_2:                ;если a<=b ,то считаем 3*(i+2)

```

```

    mov ax,VARI      ;ax = i

```

```

    add ax,bx        ;ax = ax + bx = i + 2*i = 3i

```

add ax,6 ;ax +=6, ax = 3i+6

mov VAR1,ax

jmp f2

f1_1: ; если a>b, то считаем $-(6*i-4) = -6*i + 4$

shl bx, 1 ;bx *= 2 , bx = 4*i

shl bx, 1 ;bx *= 2, bx = 8*i

mov ax,VAR1 ;ax = i

shl ax,1 ;ax = 2*i

sub ax,bx ;ax = ax - bx = 2*i - 8*i = -6*i

add ax,4 ;ax = -6*i + 4

mov VAR1,ax

f2:

mov ax,VARA ;переменная a в ax

cmp ax,VARB ;сравниваем переменные a и b соответственно

jg f2_1 ;если a>b переходим к метке f2_1

f2_2: ; если a<=b, то считаем $5 - 3*(i+1) = 2 - 3*i$

mov ax,VAR1 ;ax = i

shl ax, 1 ;ax *= 2 = 2*i

shl ax, 1 ;ax *= 2 = 4*i

mov bx,VAR1 ;bx = i

sub bx,ax ;bx = bx - ax = i - 4*i = -3*i

add bx,2 ;bx+=2

mov VARI2,bx

jmp f3

f2_1: ; если a>b, то считаем $2*(i+1)-4 = 2*i - 2$

mov bx,VARI ;bx = i

shl bx,1 ;bx *= 2

sub bx,2 ;bx = bx - 2 = 2*i - 2

mov VARI2,bx

f3:

mov ax,VARI2 ;кладем в ax i2

cmp ax,0 ;сравниваем i2 с 0

jl f3_I2_NEG ;если i2 < 0

f3_K_POS:

mov bx,VARK ;кладем в bx k

cmp bx, 0 ;сравниваем k с 0

jl f3_K_NEG

sub ax,3 ;ax = ax-3 = |i2|-3

cmp 4,ax ;сравниваем 4 с |i2|-3

jl f3_ax ;если 4 < |i2|-3

mov VARRES,4

```
jmp f3_end
```

```
f3_ax:
```

```
mov VARRES,ax
```

```
jmp f3_end
```

```
f3_K_NEG:
```

```
mov bx, VARI1    ;кладем в bx i1
```

```
cmp bx, 0        ;сравниваем i1 с 0
```

```
jl f3_I1_NEG     ;если i1<0
```

```
f3_K_NEG_COUNT:
```

```
sub bx,ax
```

```
mov VARRES,bx
```

```
jmp f3_end
```

```
f3_I1_NEG:
```

```
neg bx
```

```
jmp f3_K_NEG_COUNT
```

```
f3_I2_NEG:
```

```
neg ax
```

```
jmp f3_K_POS
```

f3_end:

mov ah, 4ch ;завершаем программу

int 21h

Main ENDP

CODE ENDS

END Main ;ENDS CODE

Листинг программы:

□ Microsoft (R) Macro Assembler Version 5.10

10/9/20 19:22:55

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 0001      VARA      DW   1h
0002 0001      VARB      DW   1h
0004 0001      VARIDW    1h
0006 0001      VARK      DW   1h
0008 0001      VARI1     DW   1h
000A 0001      VARI2     DW   1h
000C 0001      VARRES    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 EB 1A 90            jmp f1

0008           f1_end:

0008 A1 0008 R           mov ax, VARI1
000B 8F 06 0008 R        pop VARI1

000F EB 41 90            jmp f2

```



```

003E          fl_1:                                ; shl bx, 1
          a>b, 1                                  ;bx *=
          2 , bx = 4*i

003E D1 E3          shl bx, 1                      ;bx *= 2, bx = 8*i

0042 A1 0004 R      mov ax,VAR1                    ;ax = i

0045 D1 E0          shl ax,1                      ;ax = 2*i

0047 2B C3          sub ax,bx                      ;ax = ax - bx = 2*i
          - 8*i = -6*i

0049 05 0004        add ax,4                      ;ax = -6*i + 4

004C 50             push ax                        ;ret ax

004D EB 01 90        jmp fl_c

0050          fl_c:

0050 EB B6          jmp fl_end


0052          f2: ;dw f2(VARA,VARB,VARI)

0052 A1 0000 R      mov ax,VARA                    ;mov ax,VARA
a
          mov ax,VARA
          cmp ax,VARB
0055 3B 06 0002 R      cmp ax,VARB
          ;mov ax,VARA

```

	mov ax,VAR1	;ax = 1
0059 7F 14	jg f2_1	;ax > b
	mov ax,VAR2	
005B	f2_2:	;ax = 2
	mov ax,VAR3	
005E D1 E0	shl ax, 1	;ax *= 2
	mov ax,VAR4	
0060 D1 E0	shl ax, 1	;ax *= 2 = 4*i
0062 8B 1E 0004 R	mov bx,VAR1	;bx = i
0066 2B D8	sub bx,ax	;bx = bx - ax = i - 1
□Microsoft (R) Macro Assembler Version 5.10 10/9/20 19:22:55		
Page 1-3		
	mov ax,VAR5	
0068 83 C3 02	add bx,2	;bx+=2
006B 53	push bx	;ret bx
006C EB 0E 90	jmp f2_c	

```

006F          f2_1:                                ; 2* i - 2
          a > b, 2* (i+1) - 4 = 2* i - 2

006F 8B 1E 0004 R          mov bx, VARI1          ; bx = i
0073 D1 E3          shl bx, 1          ; bx *= 2
0075 83 EB 02          sub bx, 2          ; bx = b
          x - 2 = 2* i - 2

0078 53          push bx          ; ret bx
0079 EB 01 90          jmp f2_c

007C          f2_c:
007C EB 94          jmp f2_end

007E          f3: ; dw f3(VARI1, VARI2, VARK)
007E A1 0006 R          mov ax, VARK          ; ax = *si
          (k)
0081 3D 0000          cmp ax, 0          ; cmp k, 0
0084 7C 21          jl f3_1_1          ; k < 0

0086          f3_2_1:                                ; k >= 0
          k >= 0
0086 A1 000A R          mov ax, VARI2          ; ax = i2

```

0089 3D 0000 cmp ax,0 ; $\neg B \neg A$

$i2 \geq 0$

008C 7C 0E jl f3_2_c1 ; $i2 < 0$

008E f3_2_2:

008E 2D 0003 sub ax,3 ; $ax = |i2| - 3$

0091 3D 0004 cmp ax,4 ; $\neg B \neg A$

$|i2| - 3 \leq 4$

0094 7F 2F jg f3_1_c ; $|i2| - 3 > 4$

0096 B8 0004 mov ax,4 ; $\neg B$

$|i2| - 3 \leq 4, \neg B ax = 4$

0099 EB 2A 90 jmp f3_1_c ; $\neg A$

$\neg B \neg A$

$ax \geq \neg B \neg B$

009C f3_2_c1: ; $i2 < 0$

009C F7 D8 neg ax ; $i2 \geq 0$
































jmp f3_2_2

□Microsoft (R) Macro Assembler Version 5.10

10/9/20 19:22:55

Page 1-4

[illegible]

00A7	f3_1_1:	;   
	  k < 0,                          	

```
00AA 3D 0000      cmp ax,0      ; 00000000
```

[illegible]

00AD 7C 0E j1 f3_1_c1 ;||=B||

[illegible]

00AF	f3_1_2:	; $\llcorner \neg A \llcorner$
	$\llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$, $\llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$ ax $\llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$ $\neg B$ i1	
	A $\neg M$ $\llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$ $\neg B \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$ $\neg B$ bx	
00AF 8B 1E 000A R	mov bx,VARI2	; $\llcorner \llcorner$ bx
$\llcorner \llcorner \neg A \llcorner \llcorner \llcorner \llcorner$		
	$\llcorner \llcorner \llcorner \llcorner \llcorner \neg \Pi$ i2	
00B3 83 FB 00	cmp bx,0	
$\neg \neg A \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner \llcorner$		
	$\llcorner \llcorner$ bx $\neg B$ 0	
00B6 7C 09	jl f3_1_c2	; $\llcorner \neg B \llcorner \llcorner \llcorner \llcorner$ bx $\llcorner \neg B \neg A$
	$\llcorner \neg \llcorner \llcorner \neg B \llcorner \llcorner \neg M \llcorner \llcorner$	
00B8	f3_1:	
00B8 2B C3	sub ax,bx	; ax = a
	x-bx = i1 - i2	
00BA EB 09 90	jmp f3_1_c	
00BD	f3_1_c1:	; $\llcorner \neg B \llcorner \llcorner \llcorner \llcorner$ ax(i1) $\llcorner \neg B \neg A$
	$\llcorner \neg \llcorner \llcorner \neg B \llcorner \llcorner \neg M \llcorner \llcorner$	
00BD F7 D8	neg ax	; $\llcorner \llcorner \llcorner \neg \Pi \llcorner \llcorner \llcorner \llcorner \llcorner$
	$\llcorner \llcorner \neg \Gamma$ i1	
00BF EB EE	jmp f3_1_2	
00C1	f3_1_c2:	; $\llcorner \neg B \llcorner$

```

    mov     bx,i2)  mov     bx,A  mov     bx,B  mov     bx,M  mov     bx,
00C1  F7 DB                neg     bx                ;mov     bx,
                                mov     bx,i2
00C3  EB F3                jmp     f3_1

```

```

00C5                f3_1_c:

```

```

00C5  50                push     ax                ;mov     ax,
                                mov     bx,ax
00C6  EB 01 90          jmp     f3_c

```

```

00C9                f3_c:
00C9  E9 0019 R          jmp     f3_end

```

```

00CC                Main     ENDP

```

Microsoft (R) Macro Assembler Version 5.10 10/9/20 19:22:55

Page 1-5

```

00CC                CODE     ENDS

```

```

                                END Main                ;ENDS C

```

```

                                ODE

```

Microsoft (R) Macro Assembler Version 5.10 10/9/20 19:22:55

Symbols-1

Segments and Groups:

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

Symbols:

N a m e	Type	Value	Attr
F1	L NEAR	0021	CODE
F1_1	L NEAR	003E	CODE
F1_2	L NEAR	0032	CODE
F1_C	L NEAR	0050	CODE
F1_END	L NEAR	0008	CODE
F2	L NEAR	0052	CODE
F2_1	L NEAR	006F	CODE
F2_2	L NEAR	005B	CODE
F2_C	L NEAR	007C	CODE

F2_END	L NEAR	0012	CODE
F3	L NEAR	007E	CODE
F3_1	L NEAR	00B8	CODE
F3_1_1	L NEAR	00A7	CODE
F3_1_2	L NEAR	00AF	CODE
F3_1_C	L NEAR	00C5	CODE
F3_1_C1	L NEAR	00BD	CODE
F3_1_C2	L NEAR	00C1	CODE
F3_2_1	L NEAR	0086	CODE
F3_2_2	L NEAR	008E	CODE
F3_2_C	L NEAR	00A0	CODE
F3_2_C1	L NEAR	009C	CODE
F3_C	L NEAR	00C9	CODE
F3_END	L NEAR	0019	CODE

MAIN	F PROC	0000	CODE	Length = 00CC
------------	--------	------	------	---------------

VARA	L WORD	0000	DATASG
VARB	L WORD	0002	DATASG
VARI	L WORD	0004	DATASG
VARI1	L WORD	0008	DATASG
VARI2	L WORD	000A	DATASG
VARK	L WORD	0006	DATASG
VARRES	L WORD	000C	DATASG

@CPU TEXT 0101h

@FILENAME TEXT lab3

@VERSION TEXT 510

□Microsoft (R) Macro Assembler Version 5.10

10/9/20 19:22:55

Symbols-2

159 Source Lines

159 Total Lines

41 Symbols

48040 + 453075 Bytes symbol space free

0 Warning Errors

0 Severe Errors

Тестирование.

№	Входные данные	Выходные данные	Правильный результат
1	a=2 b=1 i=5 k=-26	i1=-26 i2=8 res=16	i1=-26 i2=8 res=16
2	a=2 b=1 i=3 k=2	i1=-14 i2=4 res=4	i1=-14 i2=4 res=4
3	a=2 b=1 i=5 k=2	i1=-26 i2=8 res=5	i1=-26 i2=8 res=5
4	a=1 b=2 i=5 k=2	i1=21 i2=-13 res=10	i1=21 i2=-13 res=10
5	a=-1 b=-2 i=5 k=2	i1=-26 i2=8 res=5	i1=-26 i2=8 res=5
6	a=-2 b=-1 i=5 k=2	i1=21 i2=-13 res=10	i1=21 i2=-13 res=10

Обработка результатов тестирования.

Были рассмотрены различные варианты входных данных и проверены все возможные пути работы алгоритма, на всех тестах программа отработала корректно и выдала правильные результаты.

Выводы.

Я научился представлять и обрабатывать целые числа, а также организовывать ветвящиеся процессы.

Ответы на вопросы.