

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

 jmp f1

f1_end:

 mov ax, VARI1

 pop VARI1

 jmp f2

f2_end:

 pop VARI2

 jmp f3

f3_end:

 pop VARRES

```

mov ah, 4ch          ;завершаем программу
int 21h

```

f1:

```

mov ax,VARA          ;переменная a в ax
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 a 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
push ax              ;ret ax
jmp f1_c

```

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,VARI      ;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
push ax
;ret ax
jmp fl_c

```

fl_c:

```

jmp fl_end

```

f2:

```

mov ax,VARA      ;переменная а в ax
cmp ax,VARB      ;сравниваем переменные а и b соответственно
jg  f2_1         ;если a>b переходим к метке f2_1

```

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

```

mov ax,VARI      ;ax = i
shl ax, 1        ;ax *= 2 = 2*i
shl ax, 1        ;ax *= 2 = 4*i
mov bx,VARI      ;bx = i
sub bx,ax        ;bx = bx - ax = i - 4*i = -3*i
add bx,2         ;bx+=2
push bx
;ret bx

```

jmp f2_c

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

push bx ;ret bx

jmp f2_c

f2_c:

jmp f2_end

f3:

mov ax,VARK ;ax = *si (k)

cmp ax,0 ;cmp k,0

jl f3_1_1 ;если $k < 0$

f3_2_1: ;если $k \geq 0$

mov ax,VARI2 ;ax = i2

cmp ax,0 ;сравниваем i2 с 0

jl f3_2_c1 ;если $i2 < 0$

f3_2_2:

```

sub ax,3      ;ах = |i2|-3

cmp ax,4      ;сравниваем |i2|-3 с 4

jg f3_1_c     ;если |i2|-3 > 4

mov ax,4      ;если |i2|-3 <=4, то ах = 4

jmp f3_1_c    ;переключаемся на метку, в которой отправляем ах в стек

```

```

f3_2_c1:      ;если i2 < 0

neg ax        ;меняем знак у i2

jmp f3_2_2

```

```

f3_2_c:                          ;else

push VARI2                        ;ret i1

jmp f3_c

```

```

f3_1_1:      ;если k < 0, то считаем |i1|-|i2|

mov ax,VARI1 ;в ах переменная i1

cmp ax,0     ;проверяем ах положительна ли

jl f3_1_c1   ;если отрицательна, то умножаем на -1

```


f3_1_2: ;продолжаем, в ax лежит |i1| теперь делаем
то же с bx

mov bx,VARI2 ;в bx переменная i2

cmp bx,0 ;сравниваем bx с 0

jl f3_1_c2 ;если bx отрицательна

f3_1:

sub ax,bx ;ax = ax-bx = |i1|-|i2|

jmp f3_1_c

f3_1_c1: ;если ax(i1) отрицательна

neg ax ;меняем знак у i1

jmp f3_1_2

f3_1_c2: ;если bx(i2) отрицательна

neg bx ;меняем знак у i2

jmp f3_1

f3_1_c:

push ax ;записываем в стек ax

jmp f3_c

```
f3_c:
    jmp f3_end
```

```
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

0012                  f2_end:

0012 8F 06 000A R      pop VARI2




























0016 EB 66 90           jmp f3






















0019                  f3_end:

0019 8F 06 000C R      pop VARRES

```

001D B4 4C mov ah, 4ch

;                           






























001F CD 21 int 21h

0021 fl: ;dw fl(VARA,VARB,VARI)




























0021 A1 0000 R mov ax,VARA ;              






















a

 ax






















0024 8B 36 0002 R mov si,VARB
;                             


 si

0028 8B 1E 0004 R mov bx,VARI
;                           

002C D1 E3 shl bx,1 ;    

bx  2

[illegible]

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	;ax ← VARa
			ax ← ax
0055	3B 06 0002 R	cmp ax,VARb	;ax ← VARb - ax
			ax ← ax - VARb
			ax ← ax - VARb
0059	7F 14	jg f2_1	;ax ← VARb - ax a > b ax ← ax - VARb
			ax ← ax - VARb
005B		f2_2:	;ax ← VARb

$a \leq b$, $\neg B \rightarrow \neg B \rightarrow 3 \rightarrow \neg B \rightarrow 5 - 3 * (i + 1) = 2 - 3 * i$

005B A1 0004 R mov ax,VARI ;ax = i

005E D1 E0 shl ax, 1 ;ax *=

$2 = 2 * i$

0060 D1 E0 shl ax, 1 ;ax *= 2 = 4 * i

0062 8B 1E 0004 R mov bx,VARI ;bx = i

0066 2B D8 sub bx,ax ;bx = bx - ax = i -

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

Page 1-3

$4 * i = -3 * i$

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: ; $\neg B \rightarrow \neg B$

$a > b$, $\neg B \rightarrow \neg B \rightarrow 3 \rightarrow \neg B \rightarrow 2 * (i + 1) - 4 = 2 * i - 2$

006F 8B 1E 0004 R mov bx,VARI ;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,VARX)

007E A1 0006 R mov ax,VARX ;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: ; $k \geq 0$

0086 A1 000A R mov ax,VAR2 ;ax = i2

0089 3D 0000 cmp ax,0 ; $i2 \geq 0$
 $i2 < 0$

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

008E f3_2_2:

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


```

00A0          f3_2_c:                                ;else
00A0 FF 36 000A R          push VARI2                ;ret i1
00A4 EB 23 90          jmp f3_c

```

00B3 83 FB 00	cmp bx,0	
;B=A		
	bx B 0	
00B6 7C 09	jl f3_1_c2	;B bx B=A
	B B M	
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:	;B ax(i1) B=A
	B B M	
00BD F7 D8	neg ax	;B B B B B B B B
	B B B i1	
00BF EB EE	jmp f3_1_2	
00C1	f3_1_c2:	;B B
	B bx(i2) B=A B B B B B B B	
00C1 F7 DB	neg bx	;B B B B B B B B
	B B B i2	
00C3 EB F3	jmp f3_1	

00C5 f3_1_c:

00C5 50 push ax ;

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	AlignCombine 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

```

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.