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

Студент гр. 1304 Новицкий М.Д.

Преподаватель Кирьянчиков В. А.

Санкт-Петербург

2023

Вариант 20.

Цель работы:

Изучить механизм ветвления в языке программирования ассемблер.

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.

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

Вариант 20.

Функции:

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

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

$$f8 = \begin{cases} |i1| - |i2|, & \text{при } k<0 \\ \max(4,|i2|-3), & \text{при } k>=0 \end{cases}$$

Должно быть |i2| - |i1|, при k<0

Аналитическое преобразование выражений:

Более минимальный код можно получить, если перед ветвлением вычислить и сохранить значения 2i, (-3i).

Для соблюдения пункта 4 , можно сократить код, если заметить, что выражение 3*i+6 можно получить из выражения 3*(i+2). Таким же образом можно получить из выражения 5-3*(i+1) выражение -3*i+2.

5-3*(i+1) = 5 – 3*i -3 = -3*i+2.

first_path: Для соблюдения пункта 4 , можно сократить если преобразовать выражение -(6*i - 4) в выражение -6*i+4,и таким же образом привести выражение 2*(i+1)-4 к выражению 2i-2.

Остальные части выражений оставим исходными.

Программа состоит из сегмента данных(DATA SEGMENT), сегмента стека(ASTACK) и сегмента кода(CODE).

Под сегмент стека отведено 24 байта.

В сегменте стека объявлены все необходимые 2-ух байтовые слова.

```
AStack SEGMENT STACK
 DW 12 DUP(?)
AStack ENDS

DATA
SEGMENT
a DW -5
b DW 5

i DW -1
i1 DW 0
i2 DW 0
k DW 4
res DW 0

DATA ENDS
```

После обьявления главной процедуры Main происходит сохранение адреса начала PSP в стек и сохранение адреса той команды (ip = 0000), которая будет выполнена после команды ret, то есть, int 20h. Далее в регистр ds загружается адрес начала сегмента данных.

```
push ds sub
ax, ax push
ax mov ax,
DATA
mov ds, ax
```

Далее идет основная часть программы .Для соблюдения пункта 4 в регистры были скопированы регистры 2i, (-3i). И происходит сравнение a с b

```
mov AX,i
shl AX,1
mov DX,AX ;DX = 2*i
add AX,i ;AX = 3*i
neg AX ;AX = -3*i
mov CX,a
cmp CX,b
jg first_path ;a > b
```

Затем если a<=b , то для этого случае вычисляется i1 и i2 и далее идет переход к метке f1.

```
;i2 =5-3*(i+1) = -3i+2
add AX,2 ;AX = -3*i +2
```

```
mov i2,AX
;i1 = 3*(i+2)
neg AX
add AX,4
mov i1,AX
jmp f3
```

Иначе для a>b происходит переход к метке first_path.

```
first_path:

;i1 = -(6*i - 4) = -6*i+4

shl AX,1
add AX,4
mov i1,AX

;i2= 2*(i+1)-4 = 2i-2

sub DX,2
mov i2,DX
```

Дальше идет метка f3. В общем случае значение i1 записывается в AX, потом записывается значение i2 в BX. Затем в метке module_1 вычисляется модуль BX, то есть модуль i2. Затем k сравнивается с 0.

```
f3:
	mov AX,i1
	mov BX,i2
module_1:
	neg BX
	js module_1
	mov CX,k
	cmp CX,0
	jl f3_first_path ; K<0
```

Если k>=0 , то прыжка не происходит и выполняется следующий код (Сравнивается значение 4 и |i2|-3, если второе значение больше первого то выполняется прыжок на метку f3_res, а если первое больше, то тогда в результат записывается цифра 4 и происходит прыжок на метку stop).

```
;res=max(4,|i2|-3)

sub BX,3
cmp BX,4
jg f3_res ;|i2|-3 > 4
```

```
        mov res,4
        jmp stop
```

Если |i2|-3 > 4, то осуществляется переход к f3_res. В этой метке BX (наш конечный результат) записывается в переменную res.

```
f3_res:
        mov res,BX
        jmp stop
```

Если k<0 , то осуществляется переход к метке f3_first_path.

```
f3_first_path: ;res = |i2|-|i1|
        neg AX
        js f3_first_path
        sub BX,AX
        mov res,BX
```

В этой метке мы находим модуль AX после чего вычитаем из него модуль BX, потом записываем AX в res.

```
stop:

    ret
    Main ENDP
    CODE ENDS
     END Main
```

Метка stop выполняется в любом случае и сигнализирует о завершении программы.

Тестирование.

Результаты тестирования представлены в табл. 1. Тестирование проводилось в отладчике AFD.

№ п/п	Входные данные	Выходные данные	Комментарии
1.	a = 10 b = -12 i = 5 k = 5	i1= $E6FF_{16}$ =$(-12)_{10}$; i2 = 0800_{16} = 8_{10}; res = 0500_{16} = 5_{10};	Проверяется случай a>b,k>0 и |i2|-3> 4
2.	a = 3 b = 5 i = 1 k = 5	i1 = 0900_{16} = $(9)_{10}$; i2 = $FFFF_{16}$ = $(-1)_{10}$; res = 0400_{16} = $(4)_{10}$;	Проверяется случай a<b,k>0 и |i2|-3< 4

3.	a = 2 b = -1 i = -4 k = -1	i1 = $1C00_{16} = 28_{10}$; i2 = $F6FF_{16} = (-10)_{10}$; res = $1200_{16}=(-18)_{10}$;	Проверяется случай a>b,k<0
4.	a = 3 b = 5 i = 5 k = 5	i1= $1500_{16} =(21)_{10}$; i2 = $F3FF_{16} = (-13)_{10}$; res = $0A00_{16} = 10_{10}$;	Проверяется случай a<b,k>0 |i2|-3> 4
5.	a = 2 b = 0 i = 1 k =5	i1 = $FEFF_{16} = (-2)_{10}$; i2 = $0000_{16} = 0_{10}$; res = $0400_{16} = (4)_{10}$;	Проверяется случай a>b,k>0 и |i2|-3< 4
6.	a = 2 b = 5 i = -4 k =-1	i1 = $FAFF_{16} =(-6)_{10}$; i2 = $0E00_{16} = (14)_{10}$; res = $F8FF_{16} = (8)_{10}$;	Проверяется случай a<b,k<0

Вывод:

Был изучен механизм ветвления, а также основы работы с метками и безусловными и условными переходами в ассемблере.

Приложение

Код файла lb3.asm:

```
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS

DATA SEGMENT

a DW 3
b DW 5
i DW 1
k DW 5
i1 DW 0
i2 DW 0
res DW 0

DATA ENDS

CODE SEGMENT
```

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

Main PROC FAR
    push DS
    sub AX,AX
    push AX
    mov AX,DATA
    mov DS,AX

    mov AX,i
    shl AX,1
    mov DX,AX ;DX = 2*i
    add AX,i ;AX = 3*i
     neg AX ;AX = -3*i
    mov CX,a
    cmp CX,b
    jg first_path ;a > b
     ;i2 =5-3*(i+1) = -3i+2
    add AX,2 ;AX = -3*i +2
    mov i2,AX
    ;i1 = 3*(i+2)
    neg AX
    add AX,8
    mov i1,AX
    jmp f3



first_path:

   ;i1 = -(6*i - 4) = -6*i+4

    shl AX,1
    add AX,4
    mov i1,AX

   ;i2= 2*(i+1)-4 = 2i-2

    sub DX,2
    mov i2,DX

f3:
     mov AX,i1
     mov BX,i2
module_1:
     neg BX
     js module_1
     mov CX,k
     cmp CX,0
     jl f3_first_path ; K<0

   ;res=max(4,|i2|-3)
```

```
    sub BX,3
    cmp BX,4
    jg f3_res ;|i2|-3 > 4

    mov res,4
    jmp stop

f3_res:
    mov res,BX
    jmp stop

f3_first_path: ;res = |i2|-|i1|
     neg AX
     js f3_first_path
     sub BX,AX
     mov res,BX

stop:

    ret
Main ENDP
CODE ENDS
    END Main
```

```
0000                    AStack SEGMENT STACK
0000  000C[                      DW 12 DUP(?)
       ????
                ]

0018                    AStack ENDS

0000                    DATA SEGMENT

0000  0003                   a DW 3
0002  0005                   b DW 5
0004  0001                   i DW 1
0006  0005                   k DW 5
0008  0000                   i1 DW 0
000A  0000                   i2 DW 0
000C  0000                   res DW 0

000E                    DATA ENDS

0000                    CODE SEGMENT
                            ASSUME CS:CODE, DS:DATA, SS:AStack

0000                    Main PROC FAR
0000  1E                    push DS
0001  2B C0                      sub AX,AX
0003  50                    push AX
0004  B8 ---- R             mov AX,DATA
0007  8E D8                      mov DS,AX

0009  A1 0004 R             mov AX,i
000C  D1 E0                      shl AX,1
000E  8B D0                      mov DX,AX ;DX = 2*i
0010  03 06 0004 R               add AX,i ;AX = 3*i
0014  F7 D8                       neg AX ;AX = -3*i
0016  8B 0E 0000 R               mov CX,a
001A  3B 0E 0002 R               cmp CX,b
001E  7F 11                      jg first_path ;a > b
                           ;i2 =5-3*(i+1) = -3i+2
0020  05 0002                    add AX,2 ;AX = -3*i +2
0023  A3 000A R             mov i2,AX
                            ;i1 = 3*(i+2)
0026  F7 D8                       neg AX
0028  05 0004                    add AX,4
002B  A3 0008 R             mov i1,AX
002E  EB 11 90                    jmp f3
```

```
0031                first_path:

                       ;i1 = -(6*i - 4) = -6*i+4

0031  D1 E0                   shl AX,1
0033  05 0004                 add AX,4
```

```
0036  A3 0008 R          mov i1,AX

                        ;i2= 2*(i+1)-4 = 2i-2

0039  8B C2                   mov AX,DX
003B  2D 0002                 sub AX,2
003E  A3 000A R          mov i2,AX

0041                  f3:
0041  A1 0008 R            mov AX,i1
0044  8B 1E 000A R             mov BX,i2
0048                  module_1:
0048  F7 DB                    neg BX
004A  78 FC                    js module_1
004C  8B 0E 0006 R             mov CX,k
0050  83 F9 00                cmp CX,0
0053  7C 18                    jl f3_first_path ; K<0

                        ;res=max(4,|i2|-3)

0055  83 EB 03                 sub BX,3
0058  BA 0004                 mov DX,4
005B  3B DA                   cmp BX,DX
005D  7F 07                   jg f3_res ;|i2|-3 > 4

005F  89 16 000C R            mov res,DX
0063  EB 0E 90                jmp stop

0066                  f3_res:
0066  89 1E 000C R            mov res,BX
006A  EB 07 90                jmp stop

006D                  f3_first_path: ;res = |i1|-|i2|
006D  F7 D8                    neg AX
006F  78 FC                    js f3_first_path
0071  2B D8                    sub BX,AX

0073                  stop:

0073  CB                  ret
0074                  Main ENDP
0074                  CODE ENDS
                          END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0074	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr	
A	L WORD	0000	DATA	
B	L WORD	0002	DATA	
F3	L NEAR	0041	CODE	
F3_FIRST_PATH	L NEAR	006D	CODE	
F3_RES	L NEAR	0066	CODE	
FIRST_PATH	L NEAR	0031	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 = 0074
MODULE_1	L NEAR	0048	CODE	
RES	L WORD	000C	DATA	
STOP	L NEAR	0073	CODE	
@CPU	TEXT	0101h		
@FILENAME	TEXT	lb3		
@VERSION	TEXT	510		

94 Source Lines
94 Total Lines
22 Symbols

48070 + 461237 Bytes symbol space free

0 Warning Errors
0 Severe Errors

Файл карты памяти LB3.MAP

Start	Stop	Length	Name	Class
00000H	00017H	00018H	ASTACK	
00020H	0002DH	0000EH	DATA	
00030H	000A3H	00074H	CODE	

Program entry point at 0003:0000