

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.

Студентка гр. 9383

Чебесова
И.Д.

Преподаватель

Ефремов
М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер.
Написать программу используя условные переходы на языке Ассемблер.

Текст задания.

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 .

Исходные данные.

Вариант 22 (4,8,3)

Таблица 1 — Исходные данные.

Имя функции	Функция
f 1	$-(6*i-4)$, при $a>b$
	$3*(i+2)$, при $a\leq b$
f 2	$-(6i+8)$, при $a>b$
	$9-3*(i-1)$, при $a\leq b$
f 3	$ i1+i2 $, при $k=0$
	$\min(i1,i2)$, при $k\neq 0$

Ход работы.

В ходе работы была разработана программа, которая вычисляет значение трех функций по заданным целочисленным параметрам (выбор варианта функции зависит от заданных данных).

Исходные данные записываются в сегмент данных, как и выходные. Правильность выходных данных была проверена с помощью отладчика.

Для подсчета значений функции были использованы следующие операнды:

- add – для суммирования
- sub – для вычитания
- shl – для логического сдвига влево, что равнозначно умножению на два

Результаты записывались по заранее заданным адресам переменных i1, i2 и res.

Для реализации условных переходов были использованы следующие операнды:

- cmp – для сравнения двух чисел. При использовании данного операнда его результат записывается с помощью выставления соответствующих флагов.
- jle – выполняет короткий переход, если первый операнд МЕНЬШЕ второго операнда или РАВЕН ему при выполнении операции сравнения с помощью команды CMP.
- jge – выполняет короткий переход, если первый операнд БОЛЬШЕ второго операнда или РАВЕН ему при выполнении операции сравнения с помощью команды CMP.
- je – выполняет короткий переход, если первый операнд РАВЕН второму операнду при выполнении операции сравнения с помощью команды CMP
- jmp – безусловный переход. Используется, если для перехода к следующему адресу не нужно делать дополнительных проверок.

Исходный код и листинг программы представлены в приложении А.

Примеры работы программы.

Таблица 2 — Примеры работы программы.

№	a	b	i	k	i1	i2	res
1	0	1	2	2	12	6	6
2	1	0	2	2	-8	-20	-20
3	0	1	2	0	12	6	18
4	1	0	2	0	-8	-20	28

Выводы.

Было изучено представление и обработка целых чисел на языке Ассемблер. Была написана программа с использованием условных переходов на языке Ассемблер.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.asm

```
DATA SEGMENT
DATA SEGMENT
a      DW      1
b      DW      0
i      DW      2
k      DW      0
i1     DW      ?
i2     DW      ?
res    DW      ?
DATA ENDS

AStack SEGMENT STACK
      DW 16 DUP(?)
AStack ENDS

CODE SEGMENT
      ASSUME CS:CODE, SS:AStack, DS:DATA
Main PROC FAR
      mov ax, DATA
      mov ds, ax

f1:
      mov ax, a
      cmp ax, b ;Сравнение а и b
      jle f1_second ;a <= b

      mov ax, i ;
      shl ax, 1 ; ;2i
      add ax, i ; ;3i
      shl ax, 1 ; ;6i
      ;shl ax, 1 ; ;2i
      ;mov bx, ax ; ;2i
      ;shl ax, 1 ; ;4i
      ;add ax, bx ; ;4i+2i=6i
      sub ax, 4 ; ;6i-4
      neg ax ; ;-(6i-4)
      mov i1, ax ;

      jmp f2

f1_second: ;jump сюда если а <= b
      mov ax, i ;
      add ax, 2 ;
      mov bx, ax ; ;bx=i+2
      shl ax, 1 ; ;2*(i+2)
      add ax, bx ; ;3*(i+2)
      mov i1, ax ; ;Помещаем в i1 значение из ax

f2_second: ;если а <= b
      mov ax, i;
```

```

    sub ax, 1;
    mov bx, ax ;    ;bx=i-1
    shl ax, 1 ;    ;2*(i-1)
    add ax, bx ;    ;3*(i-1)
    neg ax ;        ; -3*(i-1)
    add ax, 9 ;     ;9-3*(i-1)
    mov i2, ax ;    ;Помещаем в i2 значение из ax

    jmp f3

f2:                ;jump сюда если a > b
    mov ax, i;
    shl ax, 1 ;    ;2i
    add ax, i ;    ;3i
    shl ax, 1 ;    ;6i
    add ax, 8 ;    ;6i+8
    neg ax ;        ; -(6i+8)
    mov i2, ax ;    ;Помещаем в i2 значение из ax

f3:
    mov ax, k
    cmp ax, 0
    je f3_first ;k = 0

    mov ax, i1 ;
    mov bx, i2 ;
    cmp ax, bx ;
    jge min_i2 ;    ;i1 >= i2
    mov res, ax ;

    jmp f_end

min_i2:
    mov res, bx ;
    jmp f_end

f3_first:
    mov ax, i1 ;
    add ax, i2 ;
    cmp ax, 0 ;
    jle abs_neg ;    ;i1+i2 <= 0
    mov res, ax ;

    jmp f_end

abs_neg:
    neg ax ;
    mov res, ax ;

f_end:
    mov ah, 4ch
    int 21h
Main ENDP
CODE ENDS
    END Main

```

Название файла: lb3.lst

Microsoft (R) Macro Assembler Version 5.10

11/4/20 13:08:13

Page 1-1

```

1 0000                                DATA SEGMENT
2 0000 0001                          a      DW      1
3 0002 0002                          b      DW      2
4 0004 0004                          i      DW      4
5 0006 0000                          k      DW      0
6 0008 0000                          i1     DW      ?
7 000A 0000                          i2     DW      ?
8 000C 0000                          res     DW      ?
9 000E                                DATA ENDS
10
11 0000                                AStack SEGMENT STACK
12 0000 0010[                          DW 16 DUP(?)
13      ????
14      ]
15
16 0020                                AStack ENDS
17
18 0000                                CODE SEGMENT
19                                ASSUME CS:CODE, SS:AStack, DS:DATA
20 0000                                Main PROC FAR
21 0000 B8 ---- R                      mov ax, DATA
22 0003 8E D8                          mov ds, ax
23
24 0005                                f1:
25 0005 A1 0000 R                      mov ax, a
26 0008 3B 06 0002 R                  cmp ax, b ;Сравнение а и б
27 000C 7E 16                          jle f1_second ;а <= b
28
29 000E A1 0004 R                      mov ax, i ;
30 0011 D1 E0                          shl ax, 1 ; ;2i
31 0013 03 06 0004 R                  add ax, i ; ;3i
32 0017 D1 E0                          shl ax, 1 ; ;6i
33                                ;shl ax, 1 ; ;2i
34                                ;mov bx, ax ; ;2i
35                                ;shl ax, 1 ; ;4i
36                                ;add ax, bx ; ;4i+2i=6i
37 0019 2D 0004                      sub ax, 4 ; ;6i-4
38 001C F7 D8                          neg ax ; ;-(6i-4)
39 001E A3 0008 R                      mov i1, ax ;
40
41 0021 EB 27 90                      jmp f2
42
43 0024                                f1_second: ;jump сюда если а <= b
44 0024 A1 0004 R                      mov ax, i ;
45 0027 05 0002                      add ax, 2 ;
46 002A 8B D8                          mov bx, ax ; ;bx=i+2
47 002C D1 E0                          shl ax, 1 ; ;2*(i+2)
48 002E 03 C3                          add ax, bx ; ;3*(i+2)
49 0030 A3 0008 R                      mov i1, ax ; ;Помещаем в i1 значе
50                                ние из ax

```

```
51 0033
52 0033 A1 0004 R
53 0036 2D 0001
```

```
f2_second:           ;если a <= b
                    mov ax, i;
                    sub ax, 1;
```



```

54 0039 8B D8          mov bx, ax ;      ;bx=i-1
55 003B D1 E0          shl ax, 1 ;      ;2*(i-1)
56 003D 03 C3          add ax, bx ;      ;3*(i-1)
57 003F F7 D8          neg ax ;      ; -3*(i-1)
58 0041 05 0009        add ax, 9 ;      ;9-3*(i-1)
59 0044 A3 000A R      mov i2, ax ;      ;Помещаем в i2 значе
                             ние из ax
60
61 0047 EB 14 90        jmp f3
62
63 004A                f2:                ;jump сюда если a >
                             b
64 004A A1 0004 R      mov ax, i;
65 004D D1 E0          shl ax, 1 ;      ;2i
66 004F 03 06 0004 R   add ax, i ;      ;3i
67 0053 D1 E0          shl ax, 1 ;      ;6i
68 0055 05 0008        add ax, 8 ;      ;6i+8
69 0058 F7 D8          neg ax ;      ; -(6i+8)
70 005A A3 000A R      mov i2, ax ;Помещаем в i2 значение
                             из ax
71
72
73 005D                f3:
74 005D A1 0006 R      mov ax, k
75 0060 3D 0000        cmp ax, 0
76 0063 74 18          je f3_second ;k = 0
77
78 0065 A1 0008 R      mov ax, i1 ;
79 0068 8B 1E 000A R   mov bx, i2 ;
80 006C 3B C3          cmp ax, bx ;
81 006E 7D 06          jge min_i2 ;      ;i1 >=i2
82 0070 A3 000C R      mov res, ax ;
83
84 0073 EB 1F 90        jmp f_end
85
86 0076                min_i2:
87 0076 89 1E 000C R   mov res, bx ;
88 007A EB 18 90        jmp f_end
89
90
91 007D                f3_second:
92 007D A1 0008 R      mov ax, i1 ;
93 0080 03 06 000A R   add ax, i2 ;
94 0084 3D 0000        cmp ax, 0 ;
95 0087 7E 06          jle abs_neg ;      ;i1+i2 <= 0
96 0089 A3 000C R      mov res, ax ;
97
98 008C EB 06 90        jmp f_end
99
100 008F                abs_neg:
101 008F F7 D8          neg ax ;
102 0091 A3 000C R      mov res, ax ;
103
104

```

```

105
106 0094          f_end:
107 0094  B4 4C          mov ah, 4ch
108 0096  CD 21          int 21h
109 0098          Main  ENDP
110 0098          CODE  ENDS
111          END Main

```

Microsoft (R) Macro Assembler Version 5.10

11/4/20 13:08:13

Symbols-1

Segments and Groups:

N a m e	Length	Align	Combine Class
ASTACK	0020	PARA	STACK
CODE	0098	PARA	NONE
DATA	000E	PARA	NONE

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
ABS_NEG	L NEAR	008F	CODE
B	L WORD	0002	DATA
F1	L NEAR	0005	CODE
F1_SECOND	L NEAR	0024	CODE
F2	L NEAR	004A	CODE
F2_SECOND	L NEAR	0033	CODE
F3	L NEAR	005D	CODE
F3_SECOND	L NEAR	007D	CODE
F_END	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
MIN_I2	L NEAR	0076	CODE
RES	L WORD	000C	DATA
@CPU	TEXT	0101h	
@FILENAME	TEXT	1b3	
@VERSION	TEXT	510	

108 Source Lines

108 Total Lines

25 Symbols

47538 + 461769 Bytes symbol space free

0 Warning Errors

0 Severe Errors