

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. 9383

Гладких А.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить представление и обработку целых чисел на языке Ассемблер.
Научиться строить программы с условными переходами.

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 2

Таблица 1 — Исходные данные.

Имя функции	Функция
f1	$15-2*i$, при $a>b$
	$3*i+4$, при $a\leq b$
f2	$7 - 4*i$, при $a>b$
	$8 -6*i$, при $a\leq b$
f3	$\max(i1,10-i2)$, при $k<0$
	$ i1 - i2 $, при $k\geq 0$

Ход работы.

В ходе работы была разработана программа, которая вычисляет значение функции по заданным целочисленным параметрам.

Исходные данные записываются в сегмент данных, как и выходные. Правильность записи 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` – условный переход, срабатывающий, если левый аргумент выражения в `cmp` был меньше или равен второму. Для этого операнд проверяет, что флаги `SF` и `OF` не равны или флаг `ZF` = 1.
- `jge` – условный переход, срабатывающий, если левый аргумент выражения в `cmp` был больше или равен второму. Для этого операнд проверяет, что флаги `SF` и `OF` равны.
- `jg` - условный переход, срабатывающий, если левый аргумент выражения в `cmp` был больше второго. Для этого операнд проверяет, что флаги `SF` и `OF` равны и флаг `ZF` = 0.
- `jl` - условный переход, срабатывающий, если левый аргумент выражения в `cmp` был меньше второго. Для этого операнд проверяет, что флаги `SF` и `OF` не равны.
- `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	3	0	2	4	11	-1	12
2	5	-2	2	-2	11	-1	11
3	12	24	2	-2	10	-4	14
4	10	25	-1	2	1	14	13

Выводы.

Было изучено представление и обработка целых чисел на языке Ассемблер. Была построена программа с условными переходами, которая считает значения функций с заданными целочисленными параметрами.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.asm

```
; Стек программы
AStack SEGMENT STACK
    DW 2 DUP(?)
AStack ENDS

;Данные программы
DATA SEGMENT
a DW 3
b DW 0
i DW 2
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

    mov ax, a
    cmp ax, b
    jle f1_down ;jump if a <= b
                    ;ZF = 1 || SF != OF

f1_up:    ;a > b
    mov ax, i
    shl ax, 1
    neg ax
    mov bx, ax
    add ax, 15
    mov i1, ax
```

```

        jmp f2_up

f1_down:    ;a <= b
        mov ax, i
        shl ax, 1
        add ax, i
        mov bx, ax
        add ax, 4
        mov i1, ax

        jmp f2_down

f2_up:      ;a > b
        shl bx, 1
        mov ax, bx
        add ax, 7
        mov i2, ax

        jmp f3_if

f2_down:    ;a <= b
        shl bx, 1
        neg bx
        mov ax, 8
        add ax, bx
        mov i2, ax

        jmp f3_if

f3_if:
        cmp k, 0
        jge f3_down ;jump if k >= 0
                        ;SF = OF

f3_up:      ;k < 0
        mov ax, 10
        sub ax, i2
        cmp i1, ax
        jg f3_up_i1 ;jump if i1 > 10 - i2
                        ;ZF = 0 && SF = OF

```

```

f3_up_i2:
    mov res, ax
    jmp main_end

f3_up_i1:
    mov ax, i1
    mov res, ax

    jmp main_end

f3_down:  ;k >= 0
    mov ax, i1
    sub ax, i2
    cmp ax, 0
    jl f3_down_abs    ;jump if ax < 0
                        ;SF != OF

    mov res, ax

    jmp main_end

f3_down_abs:
    neg ax
    mov res, ax

    jmp main_end

main_end:
    mov ah, 4ch
    int 21h

Main     ENDP
CODE     ENDS
        END Main

```

Название файла: lab3.lst

#MICROSOFT (R) MACRO ASSEMBLER VERSION 5.10

10/26/20 19:31:4

PAGE 1-1

```

                                ; <= 600T
0000                                ASTACK SEGMENT STACK
0000 0002[                                DW 2 DUP(?)
                                ???
                                ]

0004                                ASTACK ENDS

                                ; 600T
0000                                DATA SEGMENT
0000 0003                                A DW 3
0002 0000                                B DW 0
0004 0002                                I DW 2
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 A1 0000 R                                MOV AX, A
0008 3B 06 0002 R                                CMP AX, B
000C 7E 12                                JLE F1_DOWN ;JUMP IF A <= B
                                ;ZF = 1 || SF !
                                = OF

000E                                F1_UP: ;A > B
000E A1 0004 R                                MOV AX, I
```


0011 D1 E0	SHL AX, 1	
0013 F7 D8	NEG AX	
0015 8B D8	MOV BX, AX	
0017 05 000F	ADD AX, 15	
001A A3 0008 R	MOV I1, AX	
001D EB 15 90	JMP F2_UP	
0020	F1_DOWN: ;A <= B	
0020 A1 0004 R	MOV AX, I	
0023 D1 E0	SHL AX, 1	
0025 03 06 0004 R	ADD AX, I	
0029 8B D8	MOV BX, AX	
002B 05 0004	ADD AX, 4	
002E A3 0008 R	MOV I1, AX	
0031 EB 0E 90	JMP F2_DOWN	
0034	F2_UP: ;A > B	
0034 D1 E3	SHL BX, 1	
#MICROSOFT (R) MACRO ASSEMBLER VERSION 5.10		10/26/20 19:31:4
	PAGE 1-2	
0036 8B C3	MOV AX, BX	
0038 05 0007	ADD AX, 7	
003B A3 000A R	MOV I2, AX	
003E EB 10 90	JMP F3_IF	
0041	F2_DOWN: ;A <= B	
0041 D1 E3	SHL BX, 1	
0043 F7 DB	NEG BX	
0045 B8 0008	MOV AX, 8	
0048 03 C3	ADD AX, BX	
004A A3 000A R	MOV I2, AX	
004D EB 01 90	JMP F3_IF	
0050	F3_IF:	

0050 83 3E 0006 R 00	CMP K, 0	
0055 7D 1C	JGE F3_DOWN ;JUMP IF K >= 0	
		;SF = OF
0057	F3_UP: ;K < 0	
0057 B8 000A	MOV AX, 10	
005A 2B 06 000A R	SUB AX, I2	
005E 39 06 0008 R	CMP I1, AX	
0062 7F 06	JG F3_UP_I1 ;JUMP IF I1 > 10 - I2	
		;ZF = 0 && SF =
		OF
0064	F3_UP_I2:	
0064 A3 000C R	MOV RES, AX	
0067 EB 24 90	JMP MAIN_END	
006A	F3_UP_I1:	
006A A1 0008 R	MOV AX, I1	
006D A3 000C R	MOV RES, AX	
0070 EB 1B 90	JMP MAIN_END	
0073	F3_DOWN: ;K >= 0	
0073 A1 0008 R	MOV AX, I1	
0076 2B 06 000A R	SUB AX, I2	
007A 3D 0000	CMP AX, 0	
007D 7C 06	JL F3_DOWN_ABS ;JUMP IF AX < 0	
		;SF !=
		OF
007F A3 000C R	MOV RES, AX	
0082 EB 09 90	JMP MAIN_END	
0085	F3_DOWN_ABS:	
0085 F7 D8	NEG AX	
0087 A3 000C R	MOV RES, AX	

008A EB 01 90 JMP MAIN_END

008D MAIN_END:

008D B4 4C MOV AH, 4CH

008F CD 21 INT 21H

0091 MAIN ENDP

0091 CODE ENDS

END MAIN

SEGMENTS AND GROUPS:

N A M E	LENGTH	ALIGN	COMBINE	CLASS
ASTACK	0004	PARA	STACK	
CODE	0091	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_DOWN	L NEAR	0020	CODE
F1_UP	L NEAR	000E	CODE
F2_DOWN	L NEAR	0041	CODE
F2_UP	L NEAR	0034	CODE
F3_DOWN	L NEAR	0073	CODE
F3_DOWN_ABS	L NEAR	0085	CODE

F3_IF	L NEAR	0050	CODE	
F3_UP	L NEAR	0057	CODE	
F3_UP_I1	L NEAR	006A	CODE	
F3_UP_I2	L NEAR	0064	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 = 0091
MAIN_END	L NEAR	008D	CODE	
RES	L WORD	000C	DATA	
@CPU	TEXT	0101H		
@FILENAME	TEXT	LAB3		
@VERSION	TEXT	510		
111 SOURCE LINES				
111 TOTAL LINES				
27 SYMBOLS				
48056 + 459204 BYTES SYMBOL SPACE FREE				
0 WARNING ERRORS				
0 SEVERE ERRORS				