

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»

Студентка гр. 1381

Васильева О. М.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2022

Задание (Шифр 1.4.3.)

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 .

Выполнение работы.

1. Функции в соответствии с указанным шифром:

$$f1 = \begin{cases} / 15-2*i, & \text{при } a>b \\ \backslash 3*i+4, & \text{при } a\leq b \end{cases}$$

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

$$f3 = \begin{cases} / |i1 + i2|, & \text{при } k=0 \\ \backslash \min(i1,i2), & \text{при } k\neq 0 \end{cases}$$

2. В начале программы описана модель памяти с помощью директивы `.model`. Модель памяти - `.small`.

3. Описание директив сегментации:

`.stack` - начало указания сегмента стека;

`.code` — начало указания сегмента кода;

`.data` — начало указания сегмента данных.

4. Также в начале мы инициализируем переменные `i`, `k`, `a`, `b` и выбираем размер стека.

5. С помощью команды `cmp` мы сравниваем значения `a` и `b`, если `a > b`, то переходим к метке `f12_else`, иначе — `f12_then`.

6. Затем вычисляются значения `i1` и `i2`, которые записываются в регистры `ax` и `cx` соответственно.

7. Финальная функция по заданию должна сравнивать `k` с 0. При `k=0` вычисляется модуль суммы `i1` и `i2`, иначе эти два значения сравниваются. Из минимальное записывается в `ax`.

8. Результат сохраняется в переменной `res`.

Минимизация длины кода.

- $a \leq b$

$$i1 = 3i + 4; \quad i2 = 3 * (i + 2) = 3i + 6$$

Пусть $g = 3i + 4$, тогда

$$i2 = g + 2; \quad g = i1.$$

- $a > b$

$$i1 = 15 - 2i; \quad i2 = -(6i - 4)$$

Пусть $g = 15 - 2i$, тогда

$$i2 = 3 * g - 11; \quad g = i1.$$

Вывод.

В ходе лабораторной работы мы познакомились с организацией ветвления на языке Ассемблер и написали программу в соответствие с индивидуальным заданием.

ПРИЛОЖЕНИЕ А. Код программ.

Имя файла: lr3.asm

```
dosseg
.model small
.stack 100h
.data
i dw 1
a dw -5
b dw 8
k dw 4
res dw ?
.code
mov ax, @data
mov ds, ax
mov ax, a
cmp ax, b ;сравниваем а и b.
jg f12_else ;выполняет короткий переход, если а больше b

f12_that: ;if(a<=b)
mov cx, cx ;3i+4
add cx, 2 ;3*(i+2)

mov ax, i ;i
add ax, i ;2i
add ax, i ;3i
add ax, 4 ;3i+4
jmp final

f12_else: ;if(a>b)
mov cx, cx ;15-2i
sub cx, i ;15-3i
sal cx, 1 ;30-6i
sub cx, 26 ;-(6i-4) = -6i+4
```

```
mov ax, 15 ;15
sub ax, i ;15-i
sub ax, i ;15-2i
```

```
final:
abs_that:
cmp k,0
jnz abs_else
add ax, cx
neg ax
js abs_that
jmp results
```

```
abs_else:
cmp cx, ax
jl min
jmp results
min: mov ax, cx
```

```
results:
mov res, ax
mov ah, 4ch
int 21h
end
```

ПРИЛОЖЕНИЕ В. Листинг успешной трансляции программы.

Текст сообщения (lr3.lst).

#Microsoft (R) Macro Assembler Version 5.10
11/5/22 19:18:41

	Page
1-1	
	dosseg
	.mod
el small	
	.sta
ck 100h	
	.dat
a	
0000 0001	
	i dw 1
0002 FFFB	
	a dw -5
0004 0008	
	b dw 8
0006 0004	
	k dw 4
0008 0000	
	res
dw ?	
	.cod
e	
0000 B8 ---- R	mov
ax, @data	
0003 8E D8	
	mov ds,
ax	
0005 A1 0002 R	mov
ax, a	
0008 3B 06 0004 R	cmp
ax, b ;сравниваем а и b.	
000C 7F 16	
	jg
f12_else ;выполняет коротки	

переход, если а больше b	й
000E	
f12_that: ; if(a<=b)	
000E 8B C9	
cx ;3i+4	mov cx,
0010 83 C1 02	
2 ;3*(i+2)	add cx,
0013 A1 0000 R	mov
ax, i ;i	
0016 03 06 0000 R	add
ax, i ;2i	
001A 03 06 0000 R	add
ax, i ;3i	
001E 05 0004	
4 ;3i+4	add ax,
0021 EB 17 90	
final	jmp
0024	
f12_else: ; if(a>b)	
0024 8B C9	
cx ;15-2i	mov cx,
0026 2B 0E 0000 R	sub
cx, i ;15-3i	
002A D1 E1	
1 ;30-6i	sal cx,
002C 83 E9 1A	
26 ;-(6i-4) = -6i+4	sub cx,

002F	B8 000F	
15	;15	mov ax,
0032	2B 06 0000 R	sub
ax, i	;15-i	
0036	2B 06 0000 R	sub
ax, i	;15-2i	
003A		final:
003A		
abs_that:		
003A	83 3E 0006 R 00	cmp
k,0		
003F	75 09	jnz
abs_else		
0041	03 C1	add ax,
cx		
0043	F7 D8	neg ax
0045	78 F3	js
abs_that		
0047	EB 0A 90	jmp
results		
004A		
abs_else:		
004A	3B C8	cmp cx,
ax		
004C	7C 03	j1 min
004E	EB 03 90	jmp

results

0051 8B C1

min:

mov ax, cx

0053

results:

#Microsoft (R) Macro Assembler Version 5.10
11/5/22 19:18:41

Page

1-2

0053 A3 0008 R
res, ax

mov

0056 B4 4C

mov ah,

4ch

0058 CD 21

int 21h

end

#Microsoft (R) Macro Assembler Version 5.10
11/5/22 19:18:41

Symbols-1

Segments and Groups:

	N a m e	Length Align Combine
Class		
DGROUP	GROUP
_DATA	000A WORD PUBLIC 'DATA'
STACK	0100

	PARA STACK 'STACK'
_TEXT	005A WORD PUBLIC 'CODE'

Symbols:

N a m e	Type Value Attr
A	L WORD 0002 _DATA
ABS_ELSE	L NEAR 004A _TEXT
ABS_THAT	L NEAR 003A _TEXT
B	L WORD 0004 _DATA
F12_ELSE	L NEAR 0024 _TEXT
F12_THAT	L NEAR 000E _TEXT
FINAL	L NEAR 003A _TEXT
I	L WORD 0000 _DATA
K	L WORD 0006 _DATA

MIN	L NEAR 0051 _TEXT
RES	L WORD 0008 _DATA
RESULTS	L NEAR 0053 _TEXT
@CODE _TEXT	TEXT
@CODESIZE	TEXT 0
@CPU 0101h	TEXT
@DATASIZE	TEXT 0
@FILENAME lr3	TEXT
@VERSION 510	TEXT
57 Source Lines	
57 Total Lines	
29 Symbols	
48004 + 459256 Bytes symbol space free	
0 Warning Errors	
0 Severe Errors	