## 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. 9383 Орлов Д.С.

Преподаватель Ефремов М.А.

Санкт-Петербург 2020

**Цель работы.**

Изучить представление целых чисел, научиться их обрабатывать.

Познакомиться с организацией ветвящихся процессов на Ассемблере.

**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);

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. из методички.

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 14:

/ 7 - 4\*i , при a>b

f1 = <

\ 8 -6\*i , при a<=b

/ -(6\*i - 4) , при a>b

f2 = <

\ 3\*(i+2) , при a<=b

/ max(i1,10-i2), при k<0

f3 = <

\ |i1 - i2| , при k>=0

# Ход работы.

В ходе работы была написана программа на языке Ассемблер, которая по заданным целочисленным параметрам вычисляет значения некоторых функций. Процесс выполнения программы ветвящийся и использует следующие команды Ассемблера:

* cmp – сравнение аргументов и установка флага ZF в соответствующее результату сравнения значение. 0, если аргументы равны и 1, если аргументы не равны.
* jle – условный переход по заданной метке при условии, что в предыдущем сравнении с использованием cmp первый аргумент меньше или равен второму.
* shl – побитовый сдвиг влево. Для целых чисел применение сдвига на 1 эквивалентно умножению значения на 2.
* add – арифметическое действие сложения целых чисел
* neg – арифметическое действие взятия противоположного целого числа
* sub – арифметическое действие вычитания целых чисел
* jmp – безусловный переход по заданной метке. Передача управления.

Исходные данные заносятся в программу до выполнения, а результат работы отслеживается через отладчик.

**Тестирование.**

|  |  |
| --- | --- |
| Входные данные (a, b, i, k) | Результат вычислений (i1, i2, res) |
| 1 1 1 1 | 0002=2 0009=9 0007 = 7 |
| 1 -1 4 -2 | FFF7 = -9 FFEC = -20 001E = 30 |
| -2 1 2 1 | FFFC = -4 000C = 16 0010 = 16 |
| -2 1 1 0 | 0002 = 2 0009 = 9 0007 = 7 |

**Выводы.**

Изучено представление целых чисел и разработана программа, выполняющая некоторые арифметические действия над целыми числами. Программа содержит ветвящиеся процессы.

**ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

**Текст файла lab3.asm**

*; / 7 - 4\*i , при a>b (f1)*

*; f1 = <*

*; \ 8 - 6\*i , при a<=b (f2)*

*;*

*; / -(6\*i - 4) , при a>b (f11)*

*; f2 = <*

*; \ 3\*(i+2) , при a<=b (f22)*

*;*

*; / max(i1,10-i2), при k<0 (f3)*

*; f3 = <*

*; \ |i1 - i2| , при k>=0 (absi)*

*AStack SEGMENT STACK*

*DW 32 DUP(?)*

*AStack ENDS*

*DATA SEGMENT*

*A DW -2*

*B DW 1*

*I DW 1*

*K DW 0*

*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*

*f1 :*

*mov ax, A*

*cmp ax, B ; сравнивает A и B*

*jle f11*

*mov ax, 1 ; ax = 1*

*sub ax, I ; ax = 1 - i*

*cmp ax, 0 ;если ax<0*

*jle f1n*

*shl ax, 1 ; ax = 2 - 2i*

*shl ax, 1 ; ax = 4 - 4i*

*add ax, 3 ; ax = 7 - 4i*

*mov I1,ax*

*jmp f2*

*f1n : ;производит умножение, если ax<0*

*neg ax*

*shl ax, 1 ; ax = 2\*(1 - i)*

*shl ax, 1 ; ax = 2\*(2 - 2i)*

*neg ax*

*add ax, 3 ; ax = 7 - 4i*

*mov I1,ax*

*jmp f2*

*f2 : ; A > B, так как мы переходим от f1*

*add ax, I ; ax = 7 - 4i + i = 7 - 3i*

*sub ax, 5 ; ax = 7 - 3i - 5 = 2 - 3i*

*cmp ax, 0 ;если ax<0*

*jle f2n*

*shl ax, 1 ; ax = 2\*(2 - 3i) = 4 - 6i = -(6i - 4)*

*jmp f3*

*f2n : ;производит умножение, если ax<0*

*neg ax*

*shl ax, 1 ; ax = 2\*(2 - 3i) = 4 - 6i = -(6i - 4)*

*neg ax*

*jmp f3*

*f11 :*

*mov ax, 1 ; ax = 1*

*sub ax, I ; ax = 1 - i*

*cmp ax, 0 ;если ax<0*

*jle f11n*

*shl ax, 1 ;ax = 2\*(1 - i) = 2 - 2i*

*shl ax, 1 ;ax = 2\*(2 - 2i) = 4 - 4i*

*add ax, I ;ax = 4 - 3i*

*shl ax, 1 ;ax = 8 - 6i*

*mov I1, ax*

*jmp f22*

*f11n : ;производит умножение, если ax<0*

*neg ax*

*shl ax, 1 ;ax = 2\*(1 - i) = 2 - 2i*

*shl ax, 1 ;ax = 2\*(2 - 2i) = 4 - 4i*

*sub ax,I*

*shl ax, 1 ;ax = 8 - 6i*

*neg ax*

*mov I1, ax*

*jmp f22*

*f22 :*

*cmp bx, ax*

*jle f22n*

*neg ax ;ax = -(8 - 6i) = 6i - 8*

*shr ax, 1 ;ax = (6i - 8) / 2 = 3i - 4*

*add ax, 10 ; ax = 3i - 4 + 10 = 3i + 6 = 3\*(i + 2)*

*jmp f3*

*f22n :*

*shr ax, 1 ;ax = (6i - 8) / 2 = 3i – 4*

*neg ax*

*add ax, 10 ; ax = 3i - 4 + 10 = 3i + 6 = 3\*(i + 2)*

*jmp f3*

*f3 :*

*cmp bx, K ;max(i1, 10 - i2)*

*jle absi*

*neg ax ; ax = -i2*

*add ax, 10 ; ax = 10 - i2*

*cmp I1, ax ; i1 <= 10 - i2 -> ax --- max*

*jle fin*

*mov ax, I1*

*jmp fin*

*absi :*

*neg ax ;ax = -i2*

*add ax, I1 ;ax = i1 - i2*

*cmp bx, ax ; i1 - i2 < 0 -> ax = -ax*

*jle fin*

*neg ax ;ax = -ax*

*jmp fin*

*fin :*

*mov RES, ax*

*mov ah, 4ch*

*int 21h*

*Main ENDP*

*CODE ENDS*

*END Main*

**Текст файла lab3.lst**

Microsoft (R) Macro Assembler Version 5.10 12/17/20 02:52:2

Page 1-1

; / 7 - 4\*i , при a>b (f1)

; f1 = <

; \ 8 - 6\*i , при a<=b (f2)

;

; / -(6\*i - 4) , при a>b (f11)

; f2 = <

; \ 3\*(i+2) , при a<=b (f22)

;

; / max(i1,10-i2), при k<0 (f3)

; f3 = <

; \ |i1 - i2| , при k>=0 (absi)

0000 AStack SEGMENT STACK

0000 0020[ DW 32 DUP(?)

????

]

0040 AStack ENDS

0000 DATA SEGMENT

0000 FFFE A DW -2

0002 0001 B DW 1

0004 0001 I DW 1

0006 0000 K DW 0

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 f1 :

0005 A1 0000 R mov ax, A

0008 3B 06 0002 R cmp ax, B ; сравнивает A и B

000C 7E 44 jle f11

000E B8 0001 mov ax, 1 ; ax = 1

0011 2B 06 0004 R sub ax, I ; ax = 1 - i

0015 3D 0000 cmp ax, 0 ;если ax<0

0018 7E 0D jle f1n

001A D1 E0 shl ax, 1 ; ax = 2 - 2i

001C D1 E0 shl ax, 1 ; ax = 4 - 4i

001E 05 0003 add ax, 3 ; ax = 7 - 4i

0021 A3 0008 R mov I1,ax

0024 EB 12 90 jmp f2

0027 f1n : ;производит э

множение, eсли ax<0

Microsoft (R) Macro Assembler Version 5.10 12/17/20 02:52:2

Page 1-2

0027 F7 D8 neg ax

0029 D1 E0 shl ax, 1 ; ax = 2\*(1 - i)

002B D1 E0 shl ax, 1 ; ax = 2\*(2 - 2i)

002D F7 D8 neg ax

002F 05 0003 add ax, 3 ; ax = 7 - 4i

0032 A3 0008 R mov I1,ax

0035 EB 01 90 jmp f2

0038 f2 : ; A > B, так как мы пЍ

µреходим от f1

0038 03 06 0004 R add ax, I ; ax = 7 - 4i + i = 7 - 3i

003C 2D 0005 sub ax, 5 ; ax = 7 - 3i - 5 = 2 -

3i

003F 3D 0000 cmp ax, 0 ;если ax<0

0042 7E 05 jle f2n

0044 D1 E0 shl ax, 1 ; ax = 2\*(2 - 3i) = 4 -

6i = -(6i - 4)

0046 EB 52 90 jmp f3

0049 f2n : ;производит э

множение, если ax<0

0049 F7 D8 neg ax

004B D1 E0 shl ax, 1 ; ax = 2\*(2 - 3i) = 4 -

6i = -(6i - 4)

004D F7 D8 neg ax

004F EB 49 90 jmp f3

0052 f11 :

0052 B8 0001 mov ax, 1 ; ax = 1

0055 2B 06 0004 R sub ax, I ; ax = 1 - i

0059 3D 0000 cmp ax, 0 ;если ax<0

005C 7E 10 jle f11n

005E D1 E0 shl ax, 1 ;ax = 2\*(1 - i) = 2 - 2i

0060 D1 E0 shl ax, 1 ;ax = 2\*(2 - 2i) = 4 -

4i

0062 03 06 0004 R add ax, I ;ax = 4 - 3i

0066 D1 E0 shl ax, 1 ;ax = 8 - 6i

0068 A3 0008 R mov I1, ax

006B EB 15 90 jmp f22

006E f11n : ;производит э

множение, если ax<0

006E F7 D8 neg ax

0070 D1 E0 shl ax, 1 ;ax = 2\*(1 - i) = 2 - 2

i

0072 D1 E0 shl ax, 1 ;ax = 2\*(2 - 2i) = 4 -

4i

0074 2B 06 0004 R sub ax,I

0078 D1 E0 shl ax, 1 ;ax = 8 - 6i

007A F7 D8 neg ax

007C A3 0008 R mov I1, ax

Microsoft (R) Macro Assembler Version 5.10 12/17/20 02:52:2

Page 1-3

007F EB 01 90 jmp f22

0082 f22 :

0082 3B D8 cmp bx, ax

0084 7E 0A jle f22n

0086 F7 D8 neg ax ;ax = -(8 - 6i) = 6i - 8

0088 D1 E8 shr ax, 1

;ax = (6i - 8) / 2 = 3i

- 4

008A 05 000A add ax, 10 ; ax = 3i - 4 + 10 = 3i + 6

= 3\*(i + 2)

008D EB 0B 90 jmp f3

0090 f22n :

0090 D1 E8 shr ax, 1 ;ax = (6i - 8) / 2 = 3i

- 4

0092 F7 D8 neg ax

0094 05 000A add ax, 10 ; ax = 3i - 4 + 10 = 3i + 6

= 3\*(i + 2)

0097 EB 01 90 jmp f3

009A f3 :

009A 3B 1E 0006 R cmp bx, K ;max(i1, 10 - i2)

009E 7E 11 jle absi

00A0 F7 D8 neg ax ; ax = -i2

00A2 05 000A add ax, 10 ; ax = 10 - i2

00A5 39 06 0008 R cmp I1, ax ; i1 <= 10 - i2 -> ax --- m

ax

00A9 7E 15 jle fin

00AB A1 0008 R mov ax, I1

00AE EB 10 90 jmp fin

00B1 absi :

00B1 F7 D8 neg ax ;ax = -i2

00B3 03 06 0008 R add ax, I1 ;ax = i1 - i2

00B7 3B D8 cmp bx, ax ; i1 - i2 < 0 -> ax = -

ax

00B9 7E 05 jle fin

00BB F7 D8 neg ax ;ax = -ax

00BD EB 01 90 jmp fin

00C0 fin :

00C0 A3 000C R mov RES, ax

00C3 B4 4C mov ah, 4ch

00C5 CD 21 int 21h

00C7 Main ENDP

00C7 CODE ENDS

END Main

Microsoft (R) Macro Assembler Version 5.10 12/17/20 02:52:2

Symbols-1

Segments and Groups:

N a m e Length Align Combine

Class

ASTACK . . . . . . . . . . . . . 0040 PARA STACK

CODE . . . . . . . . . . . . . . 00C7 PARA NONE

DATA . . . . . . . . . . . . . . 000E PARA NONE

Symbols:

N a m e Type Value Attr

A . . . . . . . . . . . . . . . L WORD 0000 DATA

ABSI . . . . . . . . . . . . . . L NEAR 00B1 CODE

B . . . . . . . . . . . . . . . L WORD 0002 DATA

F1 . . . . . . . . . . . . . . . L NEAR 0005 CODE

F11 . . . . . . . . . . . . . . L NEAR 0052 CODE

F11N . . . . . . . . . . . . . . L NEAR 006E CODE

F1N . . . . . . . . . . . . . . L NEAR 0027 CODE

F2 . . . . . . . . . . . . . . . L NEAR 0038 CODE

F22 . . . . . . . . . . . . . . L NEAR 0082 CODE

F22N . . . . . . . . . . . . . . L NEAR 0090 CODE

F2N . . . . . . . . . . . . . . L NEAR 0049 CODE

F3 . . . . . . . . . . . . . . . L NEAR 009A CODE

FIN . . . . . . . . . . . . . . L NEAR 00C0 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 = 00C7

RES . . . . . . . . . . . . . . L WORD 000C DATA

@CPU . . . . . . . . . . . . . . TEXT 0101h

@FILENAME . . . . . . . . . . . TEXT lb3

@VERSION . . . . . . . . . . . . TEXT 510

139 Source Lines

139 Total Lines

27 Symbols

48032 + 459228 Bytes symbol space free

0 Warning Errors

0 Severe Errors