**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. 0383 |  | Сабанов П.А. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**Цель работы.**

Написать программу, реализующую ветвление в зависимости от условий и работу с числами.

**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);

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.

**Ход работы.**

Были написаны необходимые функции f1, f2 и f3. Помимо них были реализованы функции abs и max. abs вычисляет абсолютное значение ax и кладёт его в ax, max вычисляет максимальное значение из ax и dx и кладёт его в ax.

Пусть a = 1, b = 2, c = 3, k = 4.

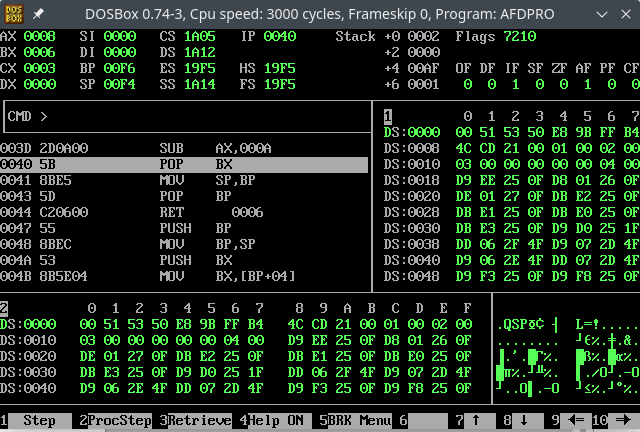
Произведём вычисления:

1) a < b => f1 = 6\*i - 10 = 8;

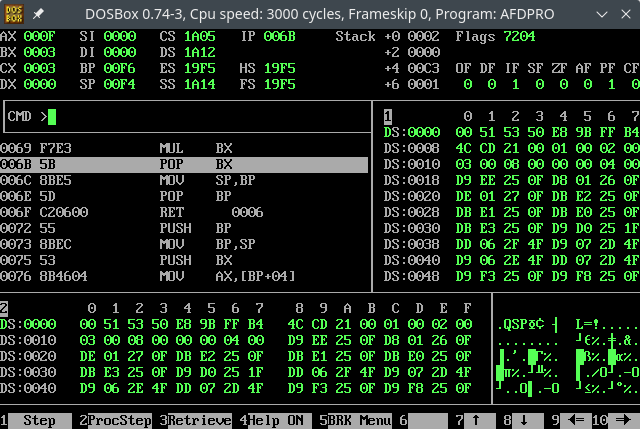
2) a < b => f2 = 3\*(i + 2) = 15;

3) k > 0 => f3 = max(6, |8|) = 8.

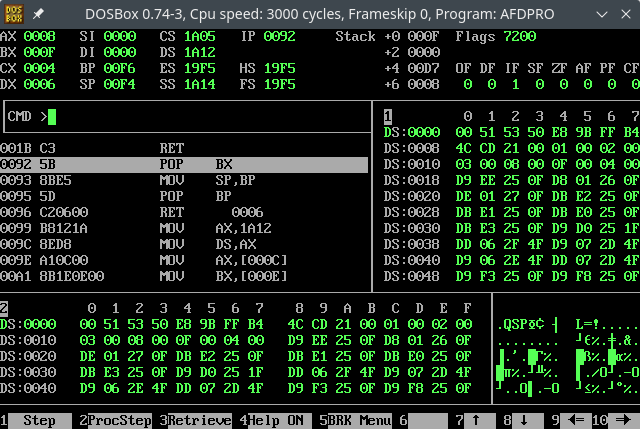
Запустим программу с помощью отладчика afdpro:



Как видно, после работы первой функции в регистре ax находится значение 8.



После работы второй функции в регистре ax находится значение 15.



После работы третьей функции в переменной ax находится значение 8.

**Выводы.**

Была написана программа, реализующая ветвление в зависимости от условий и работу с числами.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

DOSSEG

.MODEL SMALL

.STACK 100h

.DATA

a DW 1

b DW 2

i DW 3

i1 DW 0

i2 DW 0

k DW 4

.CODE

abs PROC NEAR

@abs:

neg ax

js @abs

ret

abs ENDP

max PROC NEAR

cmp ax, dx

jnl max\_exit

mov ax, dx

max\_exit:

ret

max ENDP

;f1(int a, int b, int i) {

; if (a > b)

; return -(4\*i + 3);

; return 6\*i - 10;

;}

;return in ax

f1 PROC NEAR

push bp

mov bp, sp

push bx

mov bx, [bp+4]

mov ax, [bp+8]

cmp bx, [bp+6]

jng f1\_2

mov bx, 4

mul bx

add ax, 3

neg ax

jmp f1\_exit

f1\_2:

mov bx, 6

mul bx

sub ax, 10

f1\_exit:

pop bx

mov sp, bp

pop bp

ret 6

f1 ENDP

;f2(int a, int b, int i) {

; if (a > b)

; return -(6\*i - 4);

; return 3\*(i+2);

;}

;return in ax

f2 PROC NEAR

push bp

mov bp, sp

push bx

mov bx, [bp+4]

mov ax, [bp+8]

cmp bx, [bp+6]

jng f2\_2

mov bx, 6

mul bx

sub ax, 4

neg ax

jmp f2\_exit

f2\_2:

add ax, 2

mov bx, 3

mul bx

f2\_exit:

pop bx

mov sp, bp

pop bp

ret 6

f2 ENDP

;f3(int i1, int i2, int k) {

; if (k < 0)

; return |i1| + |i2|;

; return max(6, |i1|);

;}

;return in ax

f3 PROC NEAR

push bp

mov bp, sp

push bx

mov ax, [bp+4]

call abs

cmp word ptr [bp+8], 0

jnl f3\_2

mov bx, ax

mov ax, [bp+6]

call abs

add ax, bx

f3\_2:

mov dx, 6

call max

f3\_exit:

pop bx

mov sp, bp

pop bp

ret 6

f3 ENDP

Main PROC FAR

mov ax, @data

mov ds, ax

mov ax, a

mov bx, b

mov cx, i

push cx

push bx

push ax

call f1

mov i1, ax

mov ax, a

mov bx, b

mov cx, i

push cx

push bx

push ax

call f2

mov i2, ax

mov ax, i1

mov bx, i2

mov cx, k

push cx

push bx

push ax

call f3

mov ah, 4ch

int 21h

Main ENDP

END Main

**ПРИЛОЖЕНИЕ Б**

**Листинг компиляции программы**

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Page 1-1

1 DOSSEG

2 .MODEL SMALL

3 .STACK 100h

4

5 .DATA

6

7 0000 0001 a DW 1

8 0002 0002 b DW 2

9 0004 0003 i DW 3

10 0006 0000 i1 DW 0

11 0008 0000 i2 DW 0

12 000A 0004 k DW 4

13

14 .CODE

15

16 0000 abs PROC NEAR

17

18 0000 @abs:

19 0000 F7 D8 neg ax

20 0002 78 FC js @abs

21

22 0004 C3 ret

23 0005 abs ENDP

24

25 0005 max PROC NEAR

26

27 0005 3B C2 cmp ax, dx

28 0007 7D 02 jnl max\_exit

29

30 0009 8B C2 mov ax, dx

31

32 000B max\_exit:

33

34 000B C3 ret

35 000C max ENDP

36

37 ;f1(int a, int b, int i) {

38 ; if (a > b)

39 ; return -(4\*i + 3);

40 ; return 6\*i - 10;

41 ;}

42 ;return in ax

43 000C f1 PROC NEAR

44

45 000C 55 push bp

46 000D 8B EC mov bp, sp

47 000F 53 push bx

48

49 0010 8B 5E 04 mov bx, [bp+4]

50 0013 8B 46 08 mov ax, [bp+8]

51 0016 3B 5E 06 cmp bx, [bp+6]

52 0019 7E 0D jng f1\_2

53

54 001B BB 0004 mov bx, 4

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Page 1-2

55 001E F7 E3 mul bx

56 0020 05 0003 add ax, 3

57 0023 F7 D8 neg ax

58 0025 EB 09 90 jmp f1\_exit

59

60 0028 f1\_2:

61 0028 BB 0006 mov bx, 6

62 002B F7 E3 mul bx

63 002D 2D 000A sub ax, 10

64

65 0030 f1\_exit:

66 0030 5B pop bx

67 0031 8B E5 mov sp, bp

68 0033 5D pop bp

69 0034 C2 0006 ret 6

70

71 0037 f1 ENDP

72

73

74 ;f2(int a, int b, int i) {

75 ; if (a > b)

76 ; return -(6\*i - 4);

77 ; return 3\*(i+2);

78 ;}

79 ;return in ax

80 0037 f2 PROC NEAR

81

82 0037 55 push bp

83 0038 8B EC mov bp, sp

84 003A 53 push bx

85

86 003B 8B 5E 04 mov bx, [bp+4]

87 003E 8B 46 08 mov ax, [bp+8]

88 0041 3B 5E 06 cmp bx, [bp+6]

89 0044 7E 0D jng f2\_2

90

91 0046 BB 0006 mov bx, 6

92 0049 F7 E3 mul bx

93 004B 2D 0004 sub ax, 4

94 004E F7 D8 neg ax

95 0050 EB 09 90 jmp f2\_exit

96

97 0053 f2\_2:

98

99 0053 05 0002 add ax, 2

100 0056 BB 0003 mov bx, 3

101 0059 F7 E3 mul bx

102

103 005B f2\_exit:

104 005B 5B pop bx

105 005C 8B E5 mov sp, bp

106 005E 5D pop bp

107 005F C2 0006 ret 6

108

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Page 1-3

109 0062 f2 ENDP

110

111

112 ;f3(int i1, int i2, int k) {

113 ; if (k < 0)

114 ; return |i1| + |i2|;

115 ; return max(6, |i1|);

116 ;}

117 ;return in ax

118 0062 f3 PROC NEAR

119

120 0062 55 push bp

121 0063 8B EC mov bp, sp

122 0065 53 push bx

123

124 0066 8B 46 04 mov ax, [bp+4]

125 0069 E8 0000 R call abs

126

127 006C 83 7E 08 00 cmp word ptr [bp+8], 0

128 0070 7D 0A jnl f3\_2

129

130 0072 8B D8 mov bx, ax

131 0074 8B 46 06 mov ax, [bp+6]

132 0077 E8 0000 R call abs

133 007A 03 C3 add ax, bx

134

135 007C f3\_2:

136 007C BA 0006 mov dx, 6

137 007F E8 0005 R call max

138

139 0082 f3\_exit:

140 0082 5B pop bx

141 0083 8B E5 mov sp, bp

142 0085 5D pop bp

143 0086 C2 0006 ret 6

144

145 0089 f3 ENDP

146

147 0089 Main PROC FAR

148 0089 B8 ---- R mov ax, @data

149 008C 8E D8 mov ds, ax

150

151 008E A1 0000 R mov ax, a

152 0091 8B 1E 0002 R mov bx, b

153 0095 8B 0E 0004 R mov cx, i

154 0099 51 push cx

155 009A 53 push bx

156 009B 50 push ax

157 009C E8 000C R call f1

158 009F A3 0006 R mov i1, ax

159

160 00A2 A1 0000 R mov ax, a

161 00A5 8B 1E 0002 R mov bx, b

162 00A9 8B 0E 0004 R mov cx, i

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Page 1-4

163 00AD 51 push cx

164 00AE 53 push bx

165 00AF 50 push ax

166 00B0 E8 0037 R call f2

167 00B3 A3 0008 R mov i2, ax

168

169 00B6 A1 0006 R mov ax, i1

170 00B9 8B 1E 0008 R mov bx, i2

171 00BD 8B 0E 000A R mov cx, k

172 00C1 51 push cx

173 00C2 53 push bx

174 00C3 50 push ax

175 00C4 E8 0062 R call f3

176

177 00C7 B4 4C mov ah, 4ch

178 00C9 CD 21 int 21h

179 00CB Main ENDP

180 END Main

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

DGROUP . . . . . . . . . . . . . GROUP

\_DATA . . . . . . . . . . . . 000C WORD PUBLIC 'DATA'

STACK . . . . . . . . . . . . 0100 PARA STACK 'STACK'

\_TEXT . . . . . . . . . . . . . 00CB WORD PUBLIC 'CODE'

Symbols:

N a m e Type Value Attr

A . . . . . . . . . . . . . . . L WORD 0000 \_DATA

ABS . . . . . . . . . . . . . . N PROC 0000 \_TEXT Length = 0005

B . . . . . . . . . . . . . . . L WORD 0002 \_DATA

F1 . . . . . . . . . . . . . . . N PROC 000C \_TEXT Length = 002B

F1\_2 . . . . . . . . . . . . . . L NEAR 0028 \_TEXT

F1\_EXIT . . . . . . . . . . . . L NEAR 0030 \_TEXT

F2 . . . . . . . . . . . . . . . N PROC 0037 \_TEXT Length = 002B

F2\_2 . . . . . . . . . . . . . . L NEAR 0053 \_TEXT

F2\_EXIT . . . . . . . . . . . . L NEAR 005B \_TEXT

F3 . . . . . . . . . . . . . . . N PROC 0062 \_TEXT Length = 0027

F3\_2 . . . . . . . . . . . . . . L NEAR 007C \_TEXT

F3\_EXIT . . . . . . . . . . . . L NEAR 0082 \_TEXT

I . . . . . . . . . . . . . . . L WORD 0004 \_DATA

I1 . . . . . . . . . . . . . . . L WORD 0006 \_DATA

I2 . . . . . . . . . . . . . . . L WORD 0008 \_DATA

K . . . . . . . . . . . . . . . L WORD 000A \_DATA

MAIN . . . . . . . . . . . . . . F PROC 0089 \_TEXT Length = 0042

MAX . . . . . . . . . . . . . . N PROC 0005 \_TEXT Length = 0007

MAX\_EXIT . . . . . . . . . . . . L NEAR 000B \_TEXT

@ABS . . . . . . . . . . . . . . L NEAR 0000 \_TEXT

@CODE . . . . . . . . . . . . . TEXT \_TEXT

@CODESIZE . . . . . . . . . . . TEXT 0

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

@DATASIZE . . . . . . . . . . . TEXT 0

@FILENAME . . . . . . . . . . . TEXT lab3

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

#Microsoft (R) Macro Assembler Version 5.10 11/11/21 06:37:2

Symbols-2

180 Source Lines

180 Total Lines

37 Symbols

47270 + 455893 Bytes symbol space free

0 Warning Errors

0 Severe Errors

**ПРИЛОЖЕНИЕ В**

**Карта памяти программы**

Start Stop Length Name Class

00000H 000DAH 000DBH \_TEXT CODE

000DCH 000E7H 0000CH \_DATA DATA

000F0H 001EFH 00100H STACK STACK

Origin Group

000D:0 DGROUP

Program entry point at 0000:0099