# 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. 1304 Басыров В.А.

Преподаватель Кирьянчиков В. А.

Санкт-Петербург

2022

**Вариант 4.**

**Цель работы:**

Изучить механизм ветвления в языке программирования ассемблер.

# 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.

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.

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;

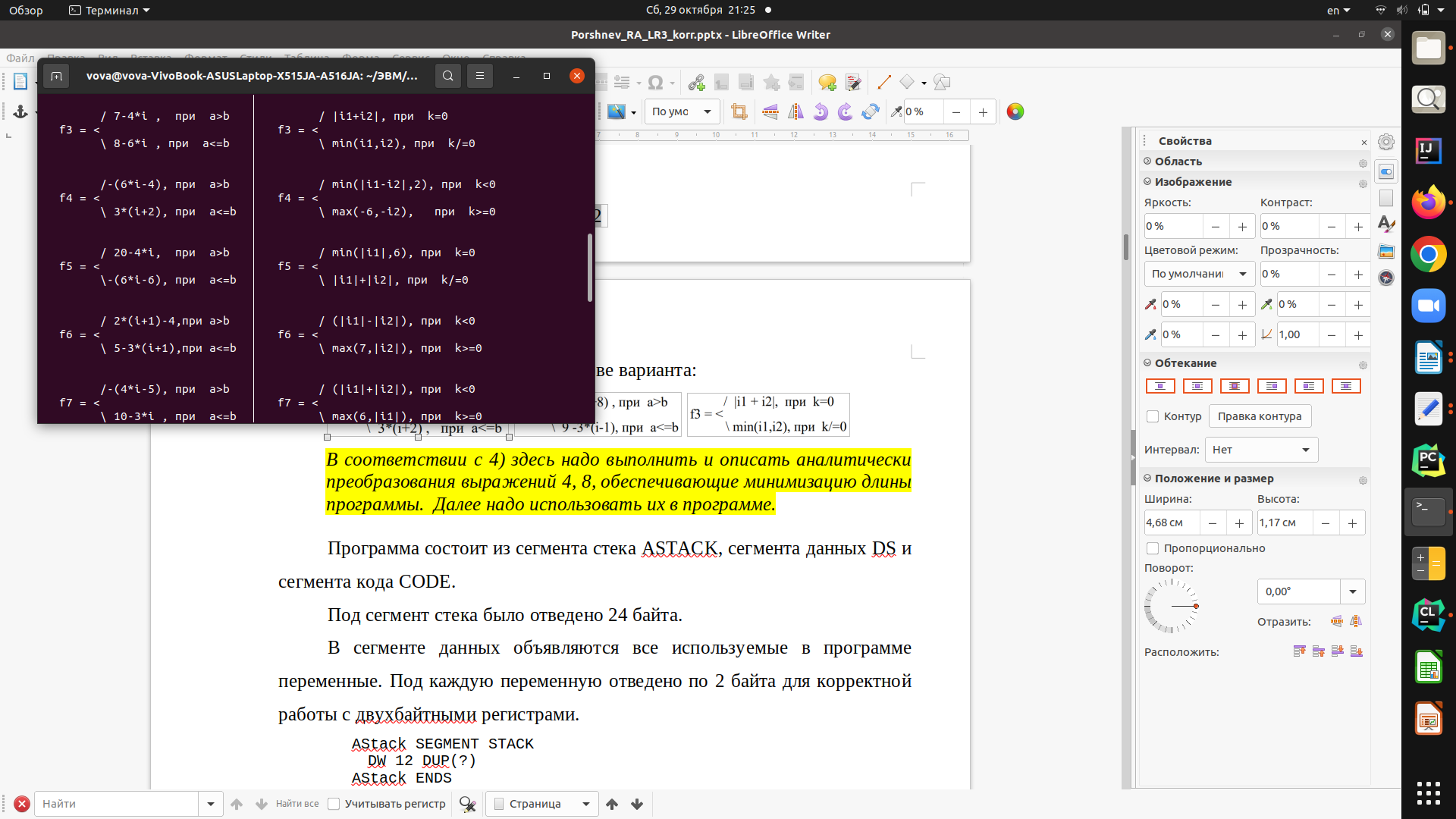
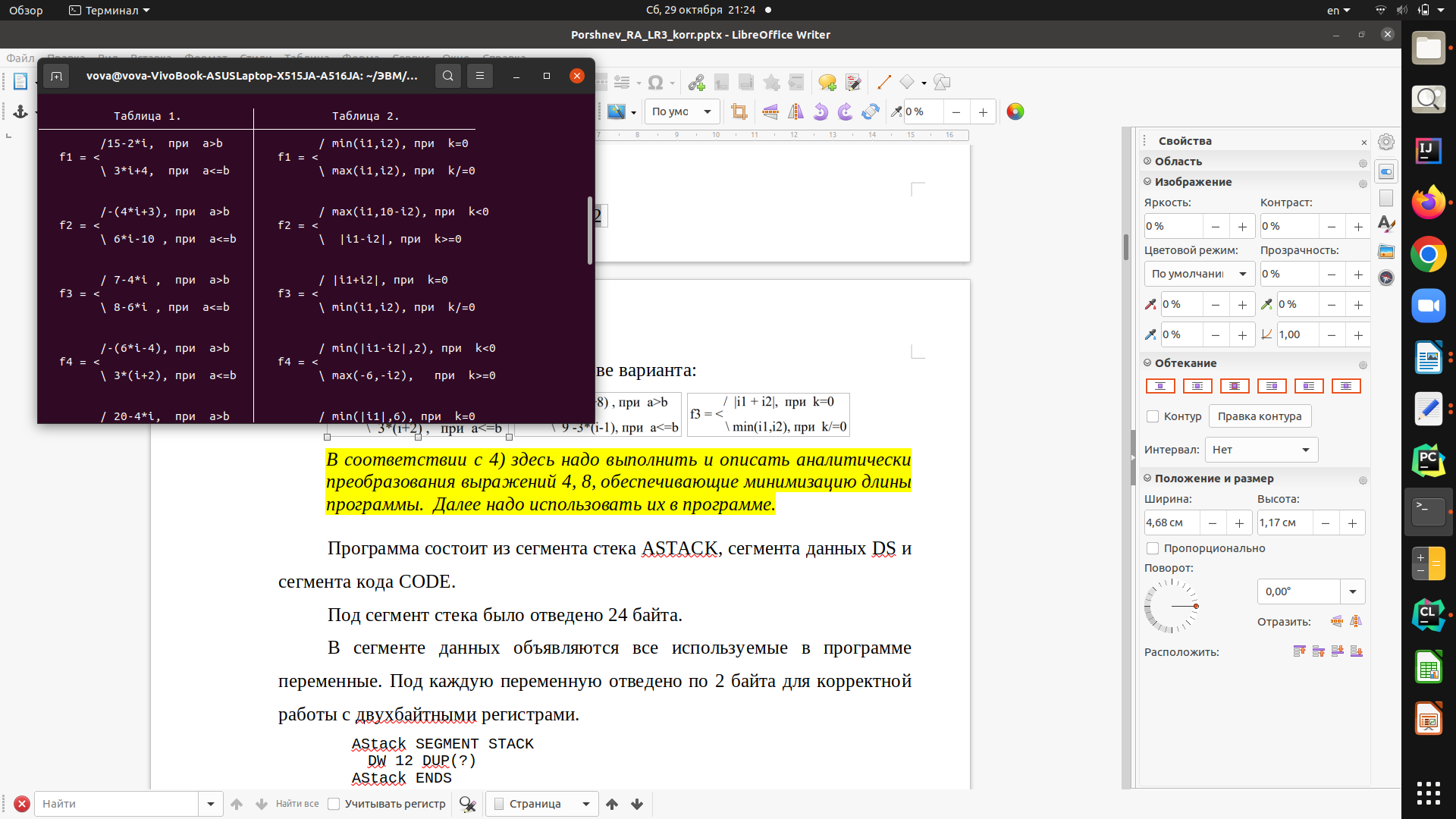
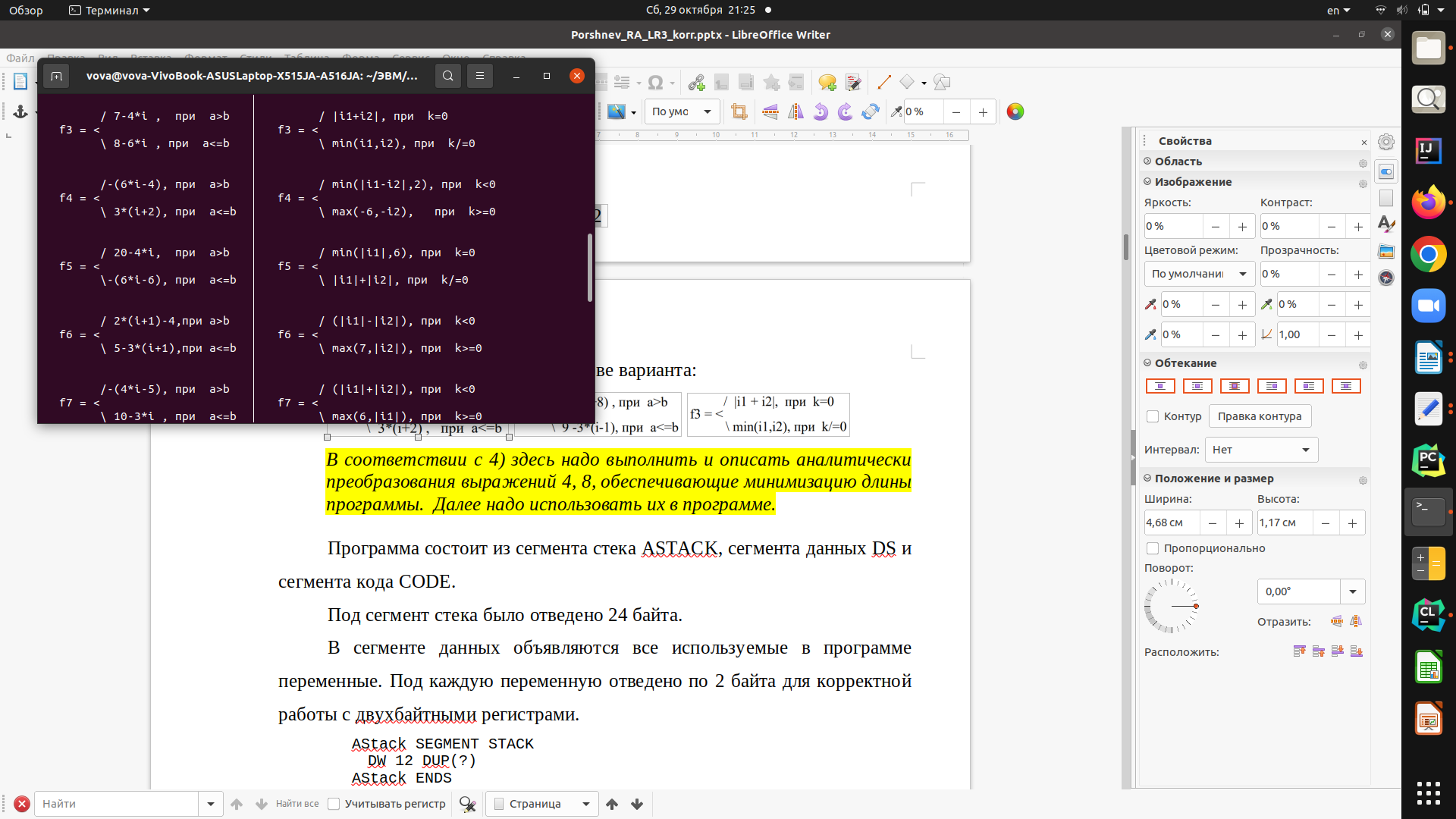
2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

**Вариант 4.**

Функции:



Аналитическое преобразование выражений:

Для того, чтобы выражения удовлетворяли пунктам 2 и 4 раздела «Замечания», были выполнены следующие преобразования:

Для соблюдения пункта 2 перед ветвлением в регистры были записаны значения:- i,- 2\*i,-4\*i.

f1: Для соблюдения пункта 4 , можно сократить код,если заметить, что выражение 3\*i+4 можно получить из выражения 6-6\*i.

3\*i+4=-2\*i-(6-6\*i)-i+10.

F5: Для соблюдения пункта 4, перепишем часть выражения: - (6\*i-6) удобно записать в виде:

Для соблюдения пункта 2, перепишем это выражение в виде.

6-2\*i-4\*i.

Остальные части выражений оставим исходными.

# Программа состоит из сегмента данных(DATA SEGMENT), сегмента стека(ASTACK) и сегмента кода(CODE).

# Под сегмент стека отведено 24 байта.

# В сегменте стека объявлены все необходимые 2-ух байтовые слова.

# AStack SEGMENT STACK

# DW 12 DUP(?)

# AStack ENDS

# DATA SEGMENT

# a DW -5

# b DW 5

# i DW -1

# i1 DW 0

# i2 DW 0

# k DW 4

# res DW 0

# DATA ENDS

# После обьявления главной процедуры Main происходит сохранение адреса начала PSP в стек и сохранение адреса той команды (ip = 0000), которая будет выполнена после команды ret, то есть, int 20h. Далее в регистр ds загружается адрес начала сегмента данных.

# push ds

# sub ax, ax

# push ax

# mov ax, DATA

# mov ds, ax

# 

# Далее идет основная часть программы.Для соблюдения пункта 4 в регистры были скопированы регистры-i,-2i,-4i. И происходит сравнение a с b

# mov cx,i;

# neg cx;-i

# mov bx,cx;

# sal bx,1;-2\*i

# mov dx,bx;

# sal dx,1;-4\*i

# mov ax,a;

# cmp ax,b;сравнить ax и b

# jg T1;сделать переход ,если a>b

# Затем если a<=b , то для этого случае вычисляется i1 и i2 и далее идет переход к метке f1.

add dx,bx;-6\*i

add dx,6;6-6\*i

sub bx,dx;4\*i-6

add bx,cx;3\*i-6

add bx,10;3\*i+4

jmp F1

# Иначе для a>b происходит переход к метке T1.

T1: ;if

add bx,15;15-2\*i

add dx,20;20-4\*i

# Дальше идет метка F1. Так как в обоих случаях результат выражения i1 записывался в bx, а i2 в dx, то начинается метка с перемещением в память этих значений.Затем проверяется значение k.

mov i1,bx

mov i2,dx

mov ax,k

cmp ax,0

jl T3;k<0

# Если k>=0 , то прыжка не происходит и выполняется следующий код.

mov bx,i2;i2

neg bx;-i2

cmp bx,-6

jle T2;bx<=-6

mov res,bx

jmp F2

T2:

mov res,-6

jmp F2

# В этой метке есть вложенная метка, так как нам надо найти максимум среди 2 переменных.Команда jle сравнивает значение i2 с -6 и переходит к метке T2 в этом случае. В конечном итоге в res записывается результат и в обоих случаях осуществляется переход к финальной метке F2, о которой будет сказано ниже.

# Если k<0 , то осуществляется переход к метке T3.

T3:

mov ax,i1;i1

mov bx,i2;i2

sub bx,ax;i2-i1

jns F3;i2-i1>0

;else

neg bx

F3:

# Стоит отметить, что во-первых в этой метке есть также вложенное условие, в котором проверяется является ли выражение положительным или нет. Если выражение является положительным, то происходит сразу переход к следующей метке, иначе выполняется команда смены знака , а только затем переход к новой метке.

F3:

cmp bx,2

jg T5

;else

mov res,bx

jmp F2

T5:

mov res,2

# В метке F3 происходит сравнения получившегося модуля с числом 2, и запись минимального в результат и переход к итоговой метке F2.

F2:

ret

Main ENDP

CODE ENDS

END Main

# Метка F2 выполняется в любом случае и сигнализирует о завершении программы.

# Тестирование.

Результаты тестирования представлены в табл. 1. Тестирование проводилось в отладчике AFD.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Входные данные | Выходные данные | Комментарии |
|  | a = 10  b = -12  i = 10  k = 5 | i1 = FBFF16 = (-5)10;  i2 = ECFF16 = (-20)10;  res = 140016 = 2010; | Проверяется случай a>b,k>0 и - i2> - 6 |
|  | a = -15  b = -10  i = -5  k = 1 | i1 = F5FF16 = (-11)10;  i2 = 240016 = 3610;  res = FAFF16 = (-6)10; | Проверяется случай a<b,k>0 и - i2< - 6 |
|  | a = 5  b = 1  i = 7  k = -5 | i1 = 010016 = 110;  i2 = F8FF16 = (-8)10;  res = 216 = 210; | Проверяется случай a>b,k<0 и |i1- i2|>2 |
|  | a = -3  b = 0  i = 0  k = -1 | i1 = 040016 = 410;  i2 = 060016 = 610;  res = 020016 = 210; | Проверяется случай a<b,k<0 и |i1- i2|>2. |
|  | a = 1  b = -2  i = -2  k =5 | i1 = 130016 = 1910;  i2 = 1C0016 = 2810;  res = FAFF16 = (-6)10; | Проверяется случай a>b,k>0 и - i2< - 6 |
|  | a = -5  b = -7  i =3  k = -2 | i1 = 090016 = 910;  i2 = 080016 = (8)10;  res = 010016 = (1)10; | Проверяется случай a>b,k<0 и |i1- i2|<2 |
|  | a = 5  b = 5  i =0  k = -10 | i1 = 040016 = 410;  i2 = 060016 = (6)10;  res = 020016 = (2)10; | Проверяются граничные случаи k=0,a=b. |

**Вывод:**

Был изучен механизм ветвления, а также основы работы с метками и безусловными и условными переходами в ассемблере.

**Приложение**

Код файла Third.asm:

AStack SEGMENT STACK

DW 12 DUP(?)

AStack ENDS

DATA SEGMENT

a DW -15

b DW -10

i DW -5

i1 DW 0

i2 DW 0

k DW 1

res DW 0

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA, SS:AStack

Main PROC FAR

push DS

sub AX,AX

push AX

mov AX,DATA

mov DS,AX

mov cx,i;

neg cx;-i

mov bx,cx;

sal bx,1;-2\*i

mov dx,bx;

sal dx,1;-4\*i

mov ax,a;

cmp ax,b;сравнить ax и b

jg T1;сделать переход ,если a>b ; 1) заменить на jа T1

;else a≤b

add dx,bx;-6\*i

add dx,6;6-6\*i

sub bx,dx;4\*i-6

add bx,cx;3\*i-6

add bx,10;3\*i+4

jmp F1

T1: ;if a>b

add bx,15;15-2\*i

add dx,20;20-4\*i

F1:

mov i1,bx

mov i2,dx

mov ax,k

cmp ax,0

jl T3;k<0 2) заменить на jl T1

;else

neg dx;-i2

cmp dx,-6

jle T2;dx<=-6 3) заменить на jbe T2

mov res,dx

jmp F2

T2:

mov res,-6

jmp F2

T3:

mov ax,i1;i1

mov bx,i2;i2

sub bx,ax;i2-i1

jns F3;i2-i1>0

;else

neg bx

F3:

cmp bx,2

jg T5 4) заменить на jа T5

;else

mov res,bx

jmp F2

T5:

mov res,2

F2:

ret

Main ENDP

CODE ENDS

END Main

**Вопросы для защиты.**

*Внести указанные изменения (ошибки) 1-4 в код, рассматривая их по отдельности. Указать, почему и как они проявятся. Показать тесты, которые эти ошибки обнаружат и их результаты. Если приведенные в отчете тесты ошибку не находят, то внести дополнительный тест.*

#Microsoft (R) Macro Assembler Version 5.10 10/29/22 23:24:5

Page 1-1

1 0000 AStack SEGMENT STACK

2 0000 000C[ DW 12 DUP(?)

3 ????

4 ]

5

6 0018 AStack ENDS

7

8

9 0000 DATA SEGMENT

10

11

12 0000 FFFB a DW -5

13 0002 0005 b DW 5

14 0004 FFFF i DW -1

15 0006 0000 i1 DW 0

16 0008 0000 i2 DW 0

17 000A 0004 k DW 4

18 000C 0000 res DW 0

19

20 000E DATA ENDS

21 0000 CODE SEGMENT

22 ASSUME CS:CODE, DS:DATA, SS:A

Stack

23

24 0000 Main PROC FAR

25 0000 1E push DS

26 0001 2B C0 sub AX,AX

27 0003 50 push AX

28 0004 B8 ---- R mov AX,DATA

29 0007 8E D8 mov DS,AX

30 0009 A1 0000 R mov ax,a;

31 000C 8B 0E 0004 R mov cx,i;

32 0010 3B 06 0002 R cmp ax,b;сравнить ax

и b

33 0014 7F 1A jg T1;сделать пере

ход ,если a<=b

34 ;else

35 0016 8B C1 mov ax,cx;i записать

в ax

36 0018 D1 E1 sal cx,1;2\*i

37 001A 03 C1 add ax,cx;3\*i

38 001C 05 0004 add ax,4;3\*i+4

39 001F A3 0006 R mov i1,ax;записать р

езультат в i1

40 0022 8B C1 mov ax,cx;2\*i

41 0024 D1 E0 sal ax,1;4\*i

42 0026 03 C8 add cx,ax;6\*i

43 0028 B8 0006 mov ax,6; 6

44 002B 2B C1 sub ax,cx;6-6\*i(-(6\*i-6))

45 002D EB 12 90 jmp F1

46 0030 T1: ;if

47 0030 D1 E1 sal cx,1;2\*i

48 0032 B8 000F mov ax,15;15

49 0035 2B C1 sub ax,cx;15-2\*i

#Microsoft (R) Macro Assembler Version 5.10 10/29/22 23:24:5

Page 1-2

50 0037 A3 0006 R mov i1,ax;

51 003A D1 E1 sal cx,1;4\*i

52 003C B8 0014 mov ax,20;20

53 003F 2B C1 sub ax,cx;20-4\*i

54 0041 F1:

55 0041 A3 0008 R mov i2,ax;

56 0044 A1 000A R mov ax,k

57 0047 3D 0000 cmp ax,0

58 004A 7C 1B jl T3;k<0

59 ;else

60 004C 8B 1E 0008 R mov bx,i2;i2

61 0050 F7 DB neg bx;-i2

62 0052 83 FB FA cmp bx,-6

63 0055 7E 07 jle T2;bx<=-6

64 0057 89 1E 000C R mov res,bx

65 005B EB 29 90 jmp F2

66 005E T2:

67 005E C7 06 000C R FFFA mov res,-6

68 0064 EB 20 90 jmp F2

69 0067 T3:

70 0067 A1 0006 R mov ax,i1;i1

71 006A 8B 1E 0008 R mov bx,i2;i2

72 006E 2B D8 sub bx,ax;i2-i1

73 0070 79 02 jns F3;i2-i1>0

74 ;else

75 0072 F7 DB neg bx

76 0074 F3:

77 0074 83 FB 02 cmp bx,2

78 0077 7F 07 jg T5

79 ;else

80 0079 89 1E 000C R mov res,bx

81 007D EB 07 90 jmp F2

82 0080 T5:

83 0080 C7 06 000C R 0002 mov res,2

84 0086 F2:

85 0086 CB ret

86 0087 Main ENDP

87 0087 CODE ENDS

88 END Main

#Microsoft (R) Macro Assembler Version 5.10 10/29/22 23:24:5

Symbols-1

Segments and Groups:

N a m e Length Align Combine Class

ASTACK . . . . . . . . . . . . . 0018 PARA STACK

CODE . . . . . . . . . . . . . . 0087 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 . . . . . . . . . . . . . . . L NEAR 0041 CODE

F2 . . . . . . . . . . . . . . . L NEAR 0086 CODE

F3 . . . . . . . . . . . . . . . L NEAR 0074 CODE

I . . . . . . . . . . . . . . . L WORD 0004 DATA

I1 . . . . . . . . . . . . . . . L WORD 0006 DATA

I2 . . . . . . . . . . . . . . . L WORD 0008 DATA

K . . . . . . . . . . . . . . . L WORD 000A DATA

MAIN . . . . . . . . . . . . . . F PROC 0000 CODE Length = 0087

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

T1 . . . . . . . . . . . . . . . L NEAR 0030 CODE

T2 . . . . . . . . . . . . . . . L NEAR 005E CODE

T3 . . . . . . . . . . . . . . . L NEAR 0067 CODE

T5 . . . . . . . . . . . . . . . L NEAR 0080 CODE

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

@FILENAME . . . . . . . . . . . TEXT LR2\_COMP

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

85 Source Lines

85 Total Lines

23 Symbols

47428 + 461879 Bytes symbol space free

0 Warning Errors

0 Severe Errors

Файл карты памяти Lr3.MAP

Start Stop Length Name Class

00000H 00017H 00018H ASTACK

00020H 0002DH 0000EH DATA

00030H 000B6H 00087H CODE

Program entry point at 0003:0000