|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**ЛАБОРАТОРНАЯ РАБОТА №4**

**«Использование команд условного перехода»**

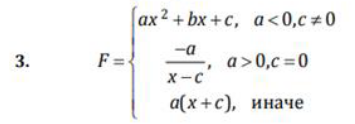
**ДИСЦИПЛИНА: «Системное программирование»**



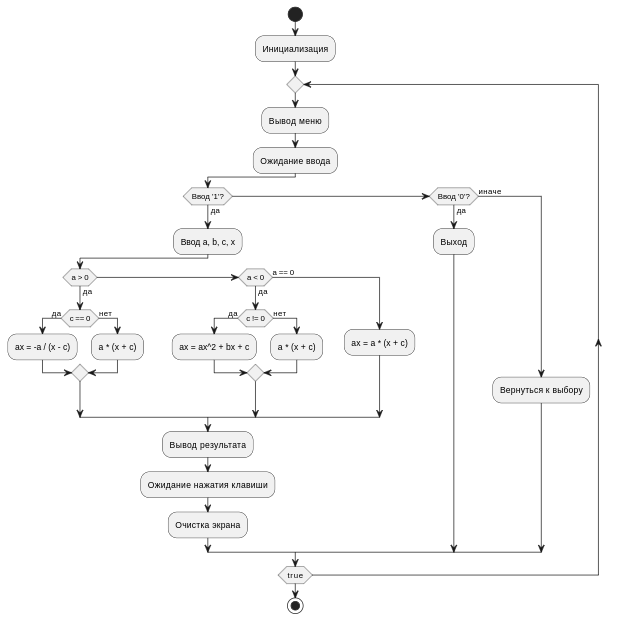
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Амеличева К. А. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** научиться выполнять операции с помощью команд условного перехода и безусловного перехода.

**Вариант**

**Блок схема программы:**

****

**Листинг файла макросов:**

1 *; Макрос ввода 10-чного числа в регистр АХ*

2 mReadAX macro buffer, sizee

3 local input, startOfConvert, endOfConvert

4 push bx *; Данные в стек*

5 push cx

6 push dx

7 input:

8 mov [buffer], sizee *; Задаём размер буфера*

9 mov dx, offset [buffer] *; Поместить в регистр dx строку по адресу buffer*

10 mov ah, 0Ah *; Чтение строки из консоли*

11 int 21h *; Прерывание DOS*

12

13 mov ah, 02h *; Вывод символа на экран*

14 mov dl, 0Dh *; Перевод каретки на новую строку*

15 int 21h *; Прерывание DOS*

16

17 mov ah, 02h *; Вывод символа на экран*

18 mov dl, 0Ah *; Чтение строки из консоли*

19 int 21h *; Прерывание DOS*

20

21 xor ah, ah *; Очистка регистра ah*

22 cmp al, [buffer][1] *; Проверка на пустую строку*

23 jz input *; Переход, если строка пустая*

24

25 xor cx, cx *; Очистка регистра cx*

26 mov cl, [buffer][1] *; инициализация переменной-счётчика*

27

28 xor ax, ax *; Очистка регистра ax*

29 xor bx, bx *; Очистка регистра bx*

30 xor dx, dx *; Очистка регистра dx*

31

32 mov bx, offset [buffer][2] *; Поместить начало строки в регистр bx*

33 cmp [buffer][2], '-' *; Проверка на знак числа*

34 jne startOfConvert *; Переход, если число неотрицательное*

35 inc bx *; Инкремент регистра bx*

36 dec cl *; Декремент регистра-счетчика cl*

37 startOfConvert:

38 mov dx, 10 *; Поместить в регистр ax число 10*

39 mul dx *; Умножение на 10 перед сложением с младшим разрядом*

40 cmp ax, 8000h *; Проверка числа на выход за границы*

41 jae input *; Переход, если число выходит за границы*

42 mov dl, [bx] *; Поместить в регистр dl следующий символ*

43 sub dl, '0' *; Перевод его в числовой формат*

44 add ax, dx *; Прибавляем его к конечному результату*

45 cmp ax, 8000h *; Проверка числа на выход за границы*

46 jae input *; Переход, если число выходит за границы*

47 inc bx *; Переход к следующему символу*

48 loop startOfConvert *; Цикл*

49 cmp [buffer][2], '-' *; Проверка на знак числа*

50 jne endOfConvert *; Переход, если число неотрицательное*

51 neg ax *; Инвертирование числа*

52 endOfConvert:

53 pop dx *; Данные из стека*

54 pop cx

55 pop bx

56 endm

57

58 *; Макрос вывода 10-чного числа из регистра AX*

59 mWriteAX macro

60 local convert, write

61 push ax *; Данные в стек*

62 push bx

63 push cx

64 push dx

65 push di

66

67 mov cx, 10 *; cx - основание системы счисления*

68 xor di, di *; di - количество цифр в числе*

69 or ax, ax *; Проверка числа на ноль*

70 jns convert *; Переход, если число положительное*

71 push ax *; Регистр ax в стек*

72 mov dx, '-' *; Поместить в регистр dx символ '-'*

73 mov ah, 02h *; Вывод символа на экран*

74 int 21h *; Прерывание DOS*

75 pop ax *; Регистр ax из стека*

76 neg ax *; Инвертирование отрицательного числа*

77 convert:

78 xor dx, dx *; Очистка регистра dx*

79 div cx *; После деления dl = остатку от деления ax на cx*

80 add dl, '0' *; Перевод в символьный формат*

81 inc di *; Увеличение количества цифр в числе на 1*

82 push dx *; Регистр dx в стек*

83 or ax, ax *; Проверка числа на ноль*

84 jnz convert *; Переход, если число не равно нулю*

85 write:

86 pop dx *; dl = очередной символ*

87 mov ah, 02h *; Вывод символа на экран*

88 int 21h *; Прерывание DOS*

89 dec di *; Повторение, пока di != 0*

90 jnz write

91

92 pop di *; Данные из стека*

93 pop dx

94 pop cx

95 pop bx

96 pop ax

97 endm

98

99 *; Макрос вывода строки*

100 mWriteStr macro string

101 push ax

102 push dx

103

104 mov ah, 09h

105 mov dx, offset string

106 int 21h

107

108 pop dx

109 pop ax

110 endm

111

112 mClear macro *;Макрос очистки экрана*

113 push ax

114 push bx

115 push cx

116 push dx

117

118 mov ax,0600h *; Подготавливает код для очистки экрана (функция 0).*

119 mov bh, 4Ch *; Устанавливает цвет фона и шрифта*

120 mov cx, 0000 *; Указывает количество строк для очистки (все).*

121 mov dx, 184FH *; Указывает адрес экрана (184FH — адрес видеопамяти).*

122 int 10h *; Вызывает прерывание BIOS для выполнения очистки экрана.*

123

124 pop dx

125 pop cx

126 pop bx

127 pop ax

128 ENDM

129

130 mSetCursor MACRO row, col *;Макрос установки курсора*

131 push ax

132 push bx

133 push cx

134 push dx

135

136 mov ah, 02 *; Установка курсора*

137 mov dh, row *; номер строки в DH*

138 mov dl, col *; номер столбца в DL*

139 mov bh, 0 *; Указывает страницу экрана (0).*

140 int 10h

141

142 pop dx

143 pop cx

144 pop bx

145 pop ax

146 ENDM

**Листинг программы:**

1 include macroses.asm

2 .model small

3 .stack 100h

4 .data

5 buffer db 5, 0, 5 dup(0)

6 mes\_a db 'Enter the number a: ', '$'

7 mes\_b db 'Enter the number b: ', '$'

8 mes\_c db 'Enter the number c: ', '$'

9 mes\_x db 'Enter the number x: ', '$'

10 a dw 0

11 b dw 0

12 c dw 0

13 x dw 0

14 menu1 db '1 - Task', '$',13,10

15 menu2 db '0 - Exit','$',13,10

16 select db 'Select> $'

17 result db 'Result> $'

18 task db 'Your Task:$'

19 .code

20 start:

21 mov ax,@data

22 mov ds,ax

23 mClear

24 select\_loop:

25 mSetCursor 10, 34

26 mWriteStr menu1

27 mSetCursor 11, 34

28 mWriteStr menu2

29 mSetCursor 12, 34

30 mWriteStr select

31

32 mov ah,01h

33 int 21h

34

35 cmp al,'1'

36 je c1

37

38 cmp al,'0'

39 je exit

40

41 jmp select\_loop

42 exit:

43 mov ah, 7h *; Подготавливает функцию для ожидания нажатия клавиши.*

44 int 21h *; Вызывает прерывание DOS для ожидания нажатия клавиши*

45

46 mov ax,4c00h

47 int 21h

48 c1:

49 mClear

50 mSetCursor 10, 34

51 mWriteStr task

52 *; Ввод переменной a*

53 xor ax, ax

54 mSetCursor 11, 27

55 mWriteStr mes\_a

56 mReadAX buffer, 5

57 mov a, ax

58 *; Ввод переменной b*

59 xor ax, ax

60 mSetCursor 12, 27

61 mWriteStr mes\_b

62 mReadAX buffer, 5

63 mov b, ax

64 *; Ввод переменной c*

65 xor ax, ax

66 mSetCursor 13, 27

67 mWriteStr mes\_c

68 mReadAX buffer, 5

69 mov c, ax

70 *; Ввод переменной x*

71 xor ax, ax

72 mSetCursor 14, 27

73 mWriteStr mes\_x

74 mReadAX buffer, 5

75 mov x, ax

76

77 xor ax,ax

78

79 mov ax,a

80 cmp ax,0

81 jg case\_pozitive\_a

82 jl case\_negative\_a

83 jmp else\_case

84 case\_pozitive\_a: *; -a / (x-c)*

85

86 xor ax, ax

87 mov ax, c

88 cmp ax, 0

89 jne else\_case

90

91 xor ax,ax

92 xor bx,bx

93 xor cx,cx

94 xor dx,dx

95

96 mov ax, a

97 neg ax

98

99 mov bx, x

100 sub bx, c

101

102 cwd

103 idiv bx

104

105 jmp otvet

106 case\_negative\_a: *; ax^2 + bx + c*

107 xor cx, cx

108 mov cx, c

109 jcxz else\_case

110

111 xor ax,ax

112 xor bx,bx

113 xor cx, cx

114

115 *; Вычисляем x^2*

116 mov ax, x *; AX = x*

117 imul ax *; AX = x \* x (x^2)*

118 mov bx, ax *; BX = x^2*

119

120 *; Вычисляем ax^2*

121 mov ax, a *; AX = a*

122 imul bx *; AX = a \* x^2*

123 mov cx, ax *; CX = a \* x^2*

124

125 *; Вычисляем bx*

126 mov ax, b *; AX = b*

127 imul x *; AX = b \* x*

128 add ax, cx *; CX += b \* x*

129

130 *; Добавляем c*

131 add ax, c *; CX += c*

132

133 jmp otvet

134 else\_case: *; a \* (x + c)*

135 xor ax,ax

136

137 mov ax, x

138 add ax, c

139 imul a

140

141 jmp otvet

142

143 otvet:

144 mSetCursor 16, 27

145 mWriteStr result

146 mWriteAX

147

148 mov ah, 7h *; Подготавливает функцию для ожидания нажатия клавиши.*

149 int 21h *; Вызывает прерывание DOS для ожидания нажатия клавиши*

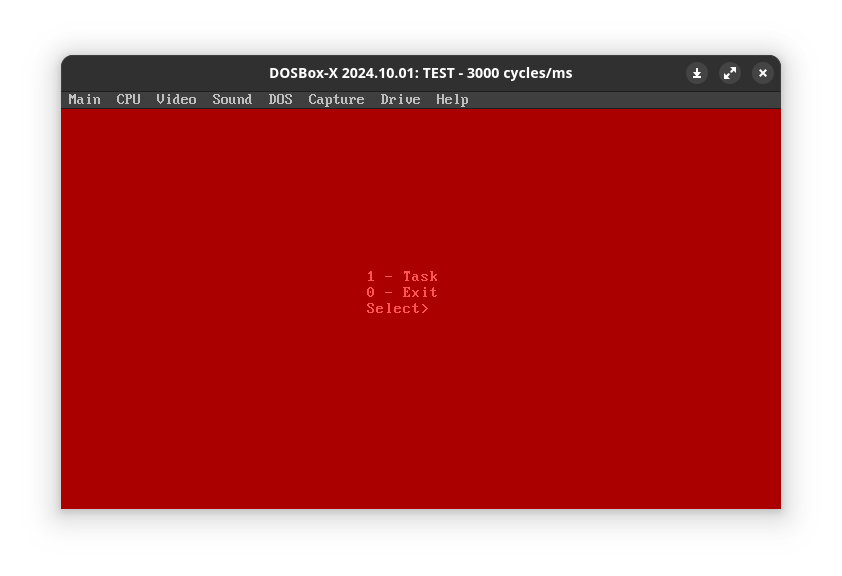
150

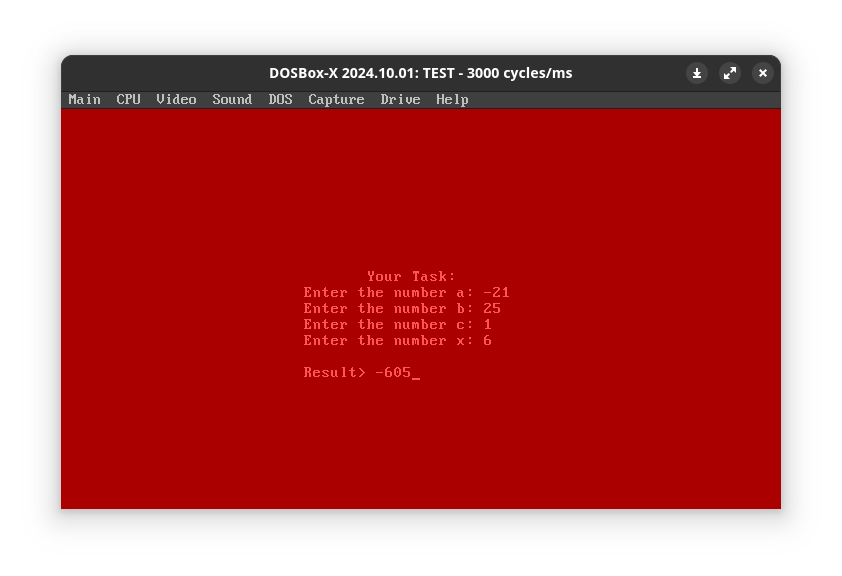
151 mClear

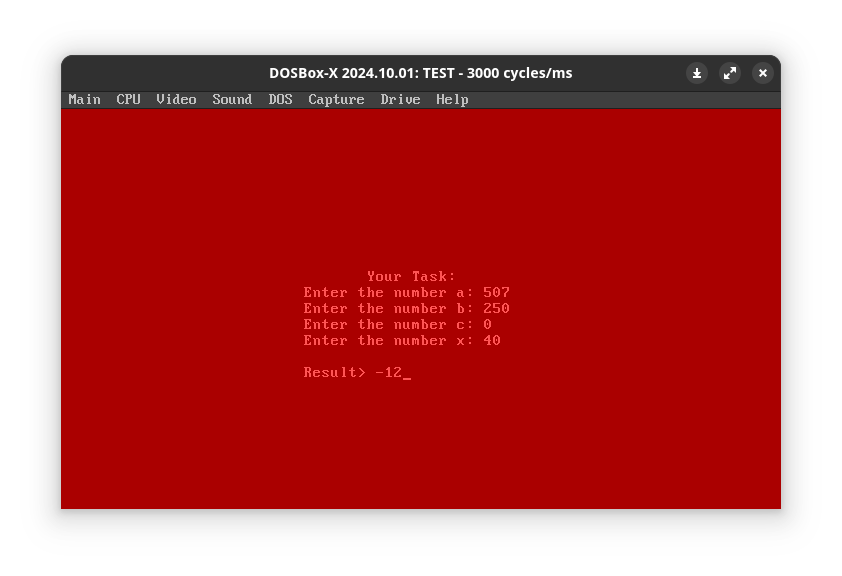
152 jmp select\_loop

153 end start

**Результат выполнения программы:**

****

****



**Вывод:** в ходе работы были получены навыки по использовать команд условного и безусловного перехода.