Лабораторная работа №8. Команды безусловного и условного переходов

Дисциплина Архитектура ЭВМ

Осокин Георгий Иванович. НММбд-02-22

Содержание

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навы- ков написания программ с использованием переходов. Знакомство с назначе- нием и структурой файла листинга.

# 2 Выполнение лабораторной работы

Создадим каталог для работы и файл, в который скопируем листинг 8.1

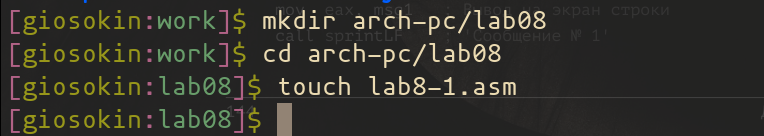


Рис. 1: Создание файла и директории

### 2.0.1 Листинг 8.1.

%include 'in\_out.asm' ; подключение внешнего файла  
   
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3 DB 'Сообщение № 3',0  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
 jmp \_label3  
\_label1:  
 mov eax, msg1 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 1'  
  
 jmp \_end ;Прыжок к выходу  
  
\_label2:  
 mov eax, msg2 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 2'  
  
  
 jmp \_label1 ; прыжок к выводу первого сообщения  
\_label3:  
 mov eax, msg3 ; Вывод на экран строки  
 call sprintLF ; 'Сообщение № 3'  
  
 jmp \_label2  
\_end:  
 call quit ; вызов подпрограммы завершения

## 2.1 Введем листинг 8.1 и исполним код.

Что бы проделывать трансляцию линковку и запуск за одну комманду, я создал отдельный run.sh файл, которым в дальнейшем буду пользоватьcя

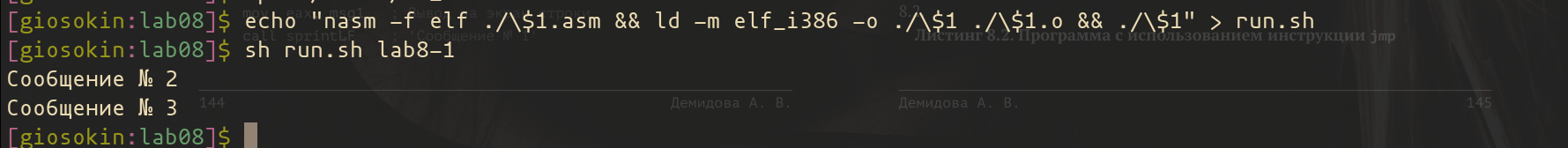


Рис. 2: Компиляция и запуск 8.1

## 2.2 Изменим листинг 8.1

Изменим несколько комманд так, что бы выводилось в начале “Сообщение 2”, потом “Сообщение 1”

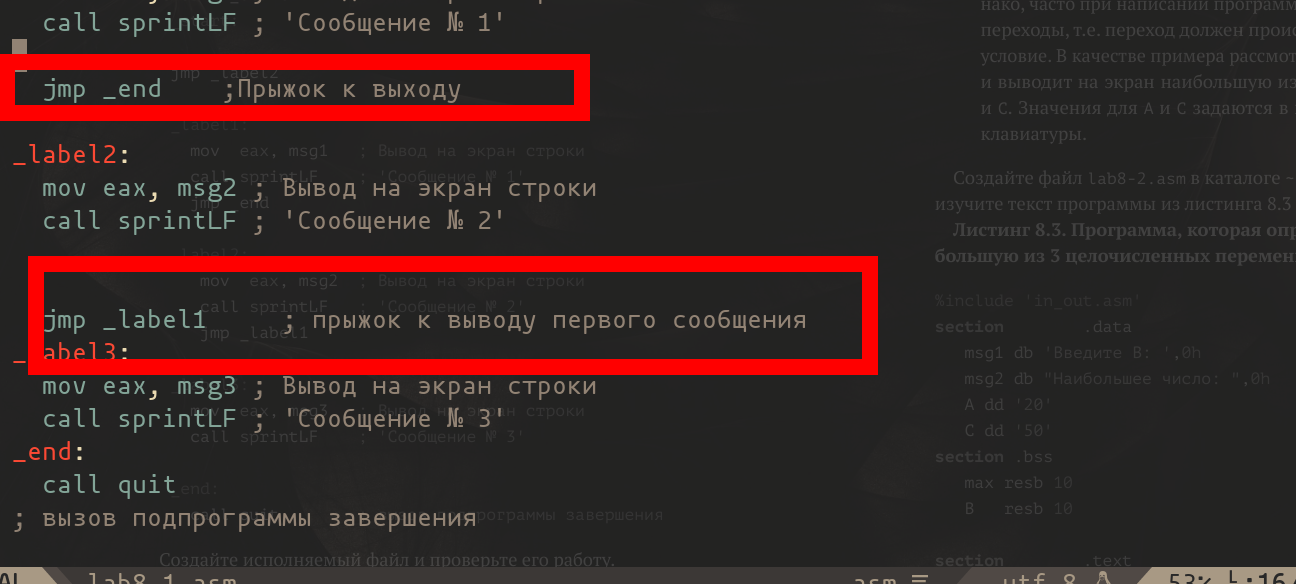


Рис. 3: Измененные строки листинга

Исполним и посмотрим на результат

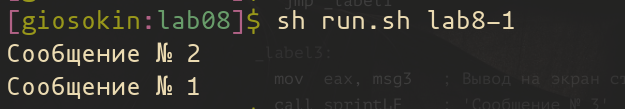


Рис. 4: Вывод измененного листинга

Изменим его так, что бы исполняемый код выводил 3, 2, 1

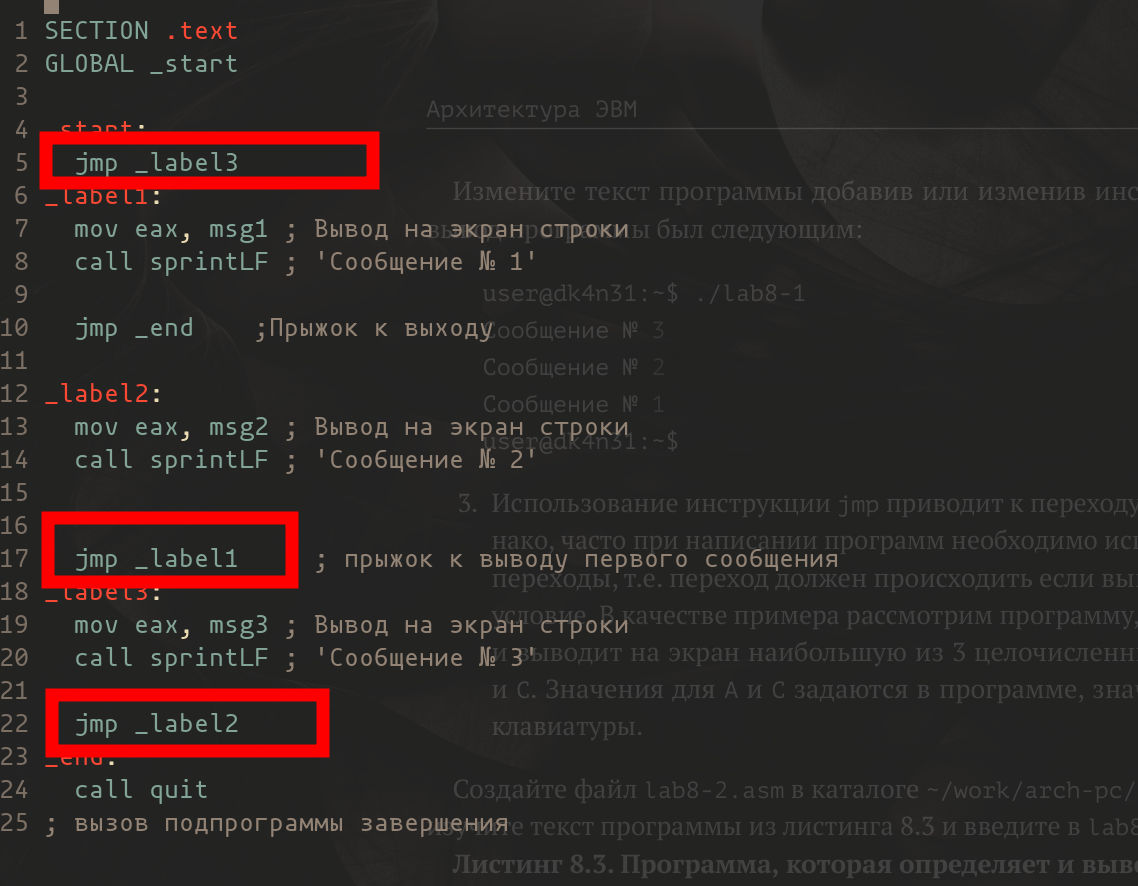


Рис. 5: Измененный листинг 8.2

Исполним код

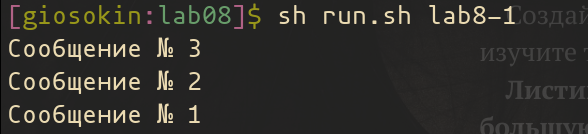


Рис. 6: Вывод листинга 8.2

## 2.3 Программа по выводу наибольшего числа

Скопируем листинг 8.3 в файл lab8-2.asm

%include 'in\_out.asm'  
section .data  
  
 msg1 db 'Введите B: ',0h  
 msg2 db "Наибольшее число: ",0h  
 A dd '20'  
 C dd '50'  
  
section .bss  
 max resb 10  
 B resb 10  
  
section .text  
  
  
global \_start  
\_start:  
  
; ---------- Вывод сообщения 'Введите B: '  
 mov eax,msg1  
 call sprint  
  
; ---------- Ввод 'B'  
 mov ecx,B  
 mov edx,10  
 call sread  
  
; ---------- Преобразование 'B' из символа в число  
 mov eax,B  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [B],eax ; запись преобразованного числа в 'B'  
  
; ---------- Записываем 'A' в переменную 'max'  
 mov ecx,[A] ; 'ecx = A'  
  
 mov [max],ecx ; 'max = A'  
; ---------- Сравниваем 'A' и 'С' (как символы)  
  
 cmp ecx,[C] ; Сравниваем 'A' и 'С'  
 jg check\_B ; если 'A>C', то переход на метку 'check\_B',  
 mov ecx,[C] ; иначе 'ecx = C'  
 mov [max],ecx ; 'max = C'  
  
; ---------- Преобразование 'max(A,C)' из символа в число  
check\_B:  
 mov eax,max  
 call atoi ; Вызов подпрограммы перевода символа в число  
 mov [max],eax ; запись преобразованного числа в `max`  
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)  
 mov ecx,[max]  
 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'  
 jg fin ; если 'max(A,C)>B', то переход на 'fin',  
 mov ecx,[B] ; иначе 'ecx = B'  
mov [max],ecx  
; ---------- Вывод результата  
fin:  
 mov eax, msg2  
 call sprint ; Вывод сообщения 'Наибольшее число: '  
 mov eax,[max]  
 call iprintLF ; Вывод 'max(A,B,C)'  
 call quit ; Выход

Проверим код из листинга 8.3 на нескольких примерах

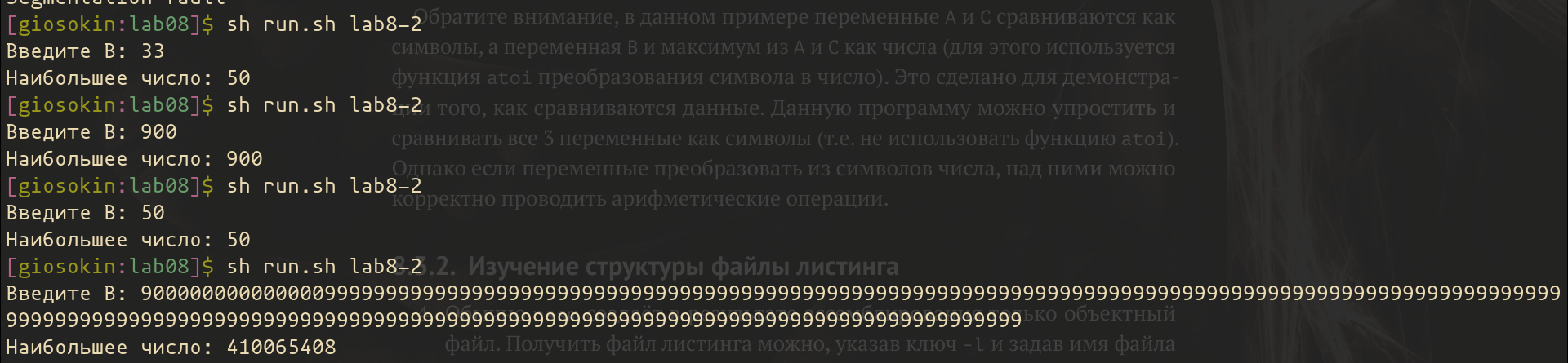


Рис. 7: Ввод различных значений в исполняемую программу

## 2.4 Изучение структуры листинга файла

Сгенерируем листинг для lab8-2.asm и откроем его в текстовом редакторе

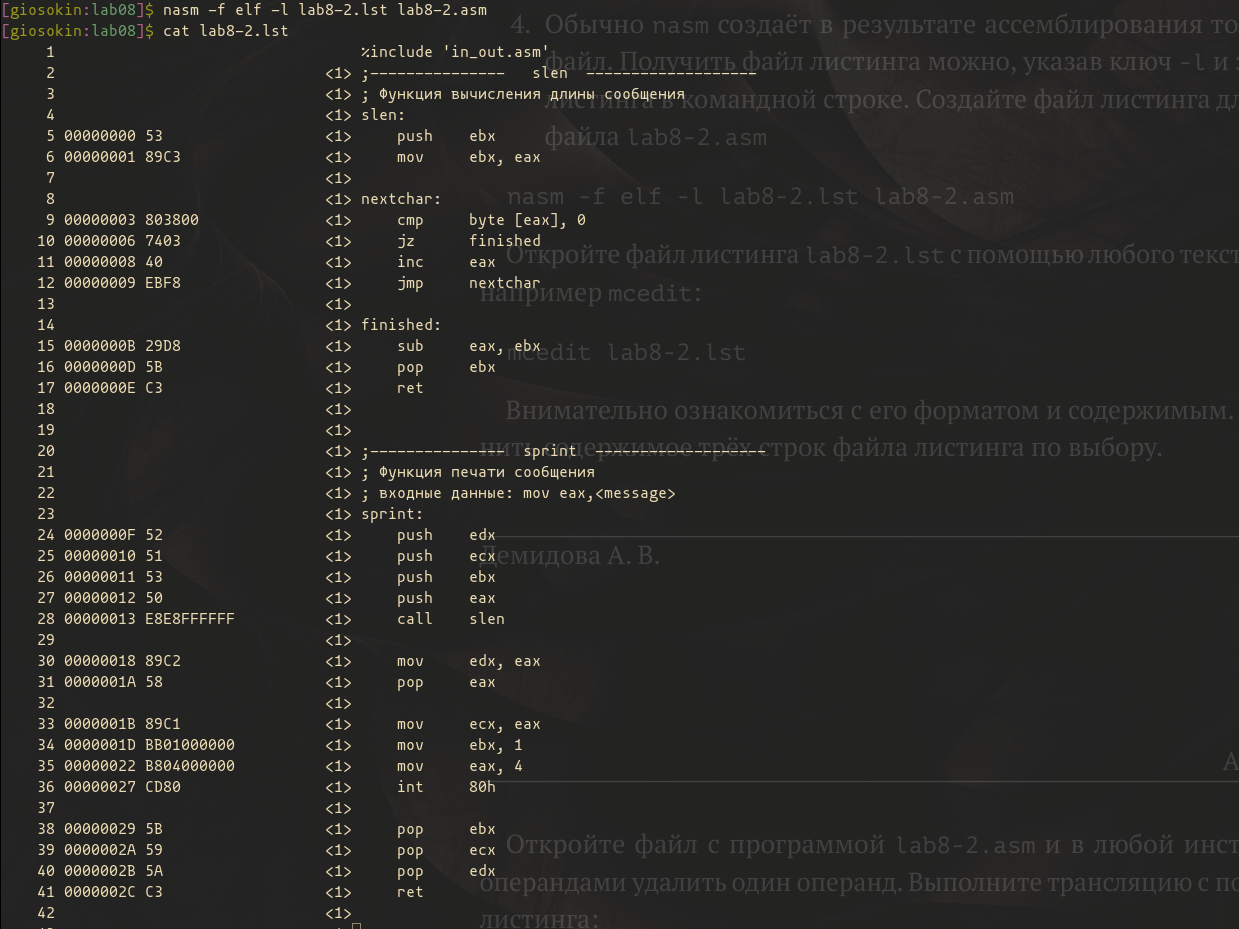


Рис. 8: Генерация и вывод на экран листинга

**Весь листинг:**

1 %include 'in\_out.asm'  
 2 <1> ;--------------- slen -------------------  
 3 <1> ; Функция вычисления длины сообщения  
 4 <1> slen:   
 5 00000000 53 <1> push ebx   
 6 00000001 89C3 <1> mov ebx, eax   
 7 <1>   
 8 <1> nextchar:   
 9 00000003 803800 <1> cmp byte [eax], 0   
 10 00000006 7403 <1> jz finished   
 11 00000008 40 <1> inc eax   
 12 00000009 EBF8 <1> jmp nextchar   
 13 <1>   
 14 <1> finished:  
 15 0000000B 29D8 <1> sub eax, ebx  
 16 0000000D 5B <1> pop ebx   
 17 0000000E C3 <1> ret   
 18 <1>   
 19 <1>   
 20 <1> ;--------------- sprint -------------------  
 21 <1> ; Функция печати сообщения  
 22 <1> ; входные данные: mov eax,<message>  
 23 <1> sprint:  
 24 0000000F 52 <1> push edx  
 25 00000010 51 <1> push ecx  
 26 00000011 53 <1> push ebx  
 27 00000012 50 <1> push eax  
 28 00000013 E8E8FFFFFF <1> call slen  
 29 <1>   
 30 00000018 89C2 <1> mov edx, eax  
 31 0000001A 58 <1> pop eax  
 32 <1>   
 33 0000001B 89C1 <1> mov ecx, eax  
 34 0000001D BB01000000 <1> mov ebx, 1  
 35 00000022 B804000000 <1> mov eax, 4  
 36 00000027 CD80 <1> int 80h  
 37 <1>   
 38 00000029 5B <1> pop ebx  
 39 0000002A 59 <1> pop ecx  
 40 0000002B 5A <1> pop edx  
 41 0000002C C3 <1> ret  
 42 <1>   
 43 <1>   
 44 <1> ;---------------- sprintLF ----------------  
 45 <1> ; Функция печати сообщения с переводом строки  
 46 <1> ; входные данные: mov eax,<message>  
 47 <1> sprintLF:  
 48 0000002D E8DDFFFFFF <1> call sprint  
 49 <1>   
 50 00000032 50 <1> push eax  
 51 00000033 B80A000000 <1> mov eax, 0AH  
 52 00000038 50 <1> push eax  
 53 00000039 89E0 <1> mov eax, esp  
 54 0000003B E8CFFFFFFF <1> call sprint  
 55 00000040 58 <1> pop eax  
 56 00000041 58 <1> pop eax  
 57 00000042 C3 <1> ret  
 58 <1>   
 59 <1> ;--------------- sread ----------------------  
 60 <1> ; Функция считывания сообщения  
 61 <1> ; входные данные: mov eax,<buffer>, mov ebx,<N>  
 62 <1> sread:  
 63 00000043 53 <1> push ebx  
 64 00000044 50 <1> push eax  
 65 <1>   
 66 00000045 BB00000000 <1> mov ebx, 0  
 67 0000004A B803000000 <1> mov eax, 3  
 68 0000004F CD80 <1> int 80h  
 69 <1>   
 70 00000051 5B <1> pop ebx  
 71 00000052 59 <1> pop ecx  
 72 00000053 C3 <1> ret  
 73 <1>   
 74 <1> ;------------- iprint ---------------------  
 75 <1> ; Функция вывода на экран чисел в формате ASCII  
 76 <1> ; входные данные: mov eax,<int>  
 77 <1> iprint:  
 78 00000054 50 <1> push eax   
 79 00000055 51 <1> push ecx   
 80 00000056 52 <1> push edx   
 81 00000057 56 <1> push esi   
 82 00000058 B900000000 <1> mov ecx, 0   
 83 <1>   
 84 <1> divideLoop:  
 85 0000005D 41 <1> inc ecx   
 86 0000005E BA00000000 <1> mov edx, 0   
 87 00000063 BE0A000000 <1> mov esi, 10   
 88 00000068 F7FE <1> idiv esi   
 89 0000006A 83C230 <1> add edx, 48   
 90 0000006D 52 <1> push edx   
 91 0000006E 83F800 <1> cmp eax, 0   
 92 00000071 75EA <1> jnz divideLoop   
 93 <1>   
 94 <1> printLoop:  
 95 00000073 49 <1> dec ecx   
 96 00000074 89E0 <1> mov eax, esp   
 97 00000076 E894FFFFFF <1> call sprint   
 98 0000007B 58 <1> pop eax   
 99 0000007C 83F900 <1> cmp ecx, 0   
 100 0000007F 75F2 <1> jnz printLoop   
 101 <1>   
 102 00000081 5E <1> pop esi   
 103 00000082 5A <1> pop edx   
 104 00000083 59 <1> pop ecx   
 105 00000084 58 <1> pop eax   
 106 00000085 C3 <1> ret  
 107 <1>   
 108 <1>   
 109 <1> ;--------------- iprintLF --------------------  
 110 <1> ; Функция вывода на экран чисел в формате ASCII  
 111 <1> ; входные данные: mov eax,<int>  
 112 <1> iprintLF:  
 113 00000086 E8C9FFFFFF <1> call iprint   
 114 <1>   
 115 0000008B 50 <1> push eax   
 116 0000008C B80A000000 <1> mov eax, 0Ah   
 117 00000091 50 <1> push eax   
 118 00000092 89E0 <1> mov eax, esp   
 119 00000094 E876FFFFFF <1> call sprint   
 120 00000099 58 <1> pop eax   
 121 0000009A 58 <1> pop eax   
 122 0000009B C3 <1> ret  
 123 <1>   
 124 <1> ;----------------- atoi ---------------------  
 125 <1> ; Функция преобразования ascii-код символа в целое число  
 126 <1> ; входные данные: mov eax,<int>  
 127 <1> atoi:  
 128 0000009C 53 <1> push ebx   
 129 0000009D 51 <1> push ecx   
 130 0000009E 52 <1> push edx   
 131 0000009F 56 <1> push esi   
 132 000000A0 89C6 <1> mov esi, eax   
 133 000000A2 B800000000 <1> mov eax, 0   
 134 000000A7 B900000000 <1> mov ecx, 0   
 135 <1>   
 136 <1> .multiplyLoop:  
 137 000000AC 31DB <1> xor ebx, ebx   
 138 000000AE 8A1C0E <1> mov bl, [esi+ecx]  
 139 000000B1 80FB30 <1> cmp bl, 48   
 140 000000B4 7C14 <1> jl .finished   
 141 000000B6 80FB39 <1> cmp bl, 57   
 142 000000B9 7F0F <1> jg .finished   
 143 <1>   
 144 000000BB 80EB30 <1> sub bl, 48   
 145 000000BE 01D8 <1> add eax, ebx  
 146 000000C0 BB0A000000 <1> mov ebx, 10   
 147 000000C5 F7E3 <1> mul ebx   
 148 000000C7 41 <1> inc ecx   
 149 000000C8 EBE2 <1> jmp .multiplyLoop   
 150 <1>   
 151 <1> .finished:  
 152 000000CA 83F900 <1> cmp ecx, 0   
 153 000000CD 7407 <1> je .restore   
 154 000000CF BB0A000000 <1> mov ebx, 10   
 155 000000D4 F7F3 <1> div ebx   
 156 <1>   
 157 <1> .restore:  
 158 000000D6 5E <1> pop esi   
 159 000000D7 5A <1> pop edx   
 160 000000D8 59 <1> pop ecx   
 161 000000D9 5B <1> pop ebx   
 162 000000DA C3 <1> ret  
 163 <1>   
 164 <1>   
 165 <1> ;------------- quit ---------------------  
 166 <1> ; Функция завершения программы  
 167 <1> quit:  
 168 000000DB BB00000000 <1> mov ebx, 0   
 169 000000E0 B801000000 <1> mov eax, 1   
 170 000000E5 CD80 <1> int 80h  
 171 000000E7 C3 <1> ret  
 2 section .data  
 3   
 4 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h  
 4 00000009 B8D182D0B520423A20-  
 4 00000012 00   
 5 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h  
 5 0000001C BED0BBD18CD188D0B5-  
 5 00000025 D0B520D187D0B8D181-  
 5 0000002E D0BBD0BE3A2000   
 6 00000035 32300000 A dd '20'  
 7 00000039 35300000 C dd '50'  
 8   
 9 section .bss  
 10 00000000 <res Ah> max resb 10  
 11 0000000A <res Ah> B resb 10  
 12   
 13 section .text  
 14   
 15   
 16 global \_start  
 17 \_start:  
 18   
 19 ; ---------- Вывод сообщения 'Введите B: '  
 20 000000E8 B8[00000000] mov eax,msg1  
 21 000000ED E81DFFFFFF call sprint  
 22   
 23 ; ---------- Ввод 'B'  
 24 000000F2 B9[0A000000] mov ecx,B  
 25 000000F7 BA0A000000 mov edx,10  
 26 000000FC E842FFFFFF call sread  
 27   
 28 ; ---------- Преобразование 'B' из символа в число  
 29 mov eax, [B]  
 30 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число  
 31 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'  
 32   
 33 ; ---------- Записываем 'A' в переменную 'max'  
 34 0000010B 8B0D[35000000] mov ecx,[A] ; 'ecx = A'  
 35   
 36 00000111 890D[00000000] mov [max],ecx ; 'max = A'  
 37 ; ---------- Сравниваем 'A' и 'С' (как символы)  
 38   
 39 00000117 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'С'  
 40 0000011D 7F0C jg check\_B ; если 'A>C', то переход на метку 'check\_B',  
 41 0000011F 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'  
 42 00000125 890D[00000000] mov [max],ecx ; 'max = C'  
 43   
 44 ; ---------- Преобразование 'max(A,C)' из символа в число  
 45 check\_B:  
 46 0000012B B8[00000000] mov eax,max  
 47 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода символа в число  
 48 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в `max`  
 49 ; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)  
 50 0000013A 8B0D[00000000] mov ecx,[max]  
 51 00000140 3B0D[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'  
 52 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin',  
 53 00000148 8B0D[0A000000] mov ecx,[B] ; иначе 'ecx = B'  
 54 0000014E 890D[00000000] mov [max],ecx  
 55 ; ---------- Вывод результата  
 56 fin:  
 57 00000154 B8[13000000] mov eax, msg2  
 58 00000159 E8B1FEFFFF call sprint ; Вывод сообщения 'Наибольшее число: '  
 59 0000015E A1[00000000] mov eax,[max]  
 60 00000163 E81EFFFFFF call iprintLF ; Вывод 'max(A,B,C)'  
 61 00000168 E86EFFFFFF call quit  
 62 ; Выход

### 2.4.1 Опишем некоторые строки листинга

Как видим, в листинге также и весь код из in\_out.asm

1. **Строка 21 (с начала)** - это комментарий, по этому в листинге нет никакого дополнительной информации, кроме самой строки и ее номера
2. **Строка 61** - Вызов подпрограммы quit в этой строчке находится ее адресс в виртуальной памяти и код комманды call quit
3. **Строка 10 (с конца)** - Резервирование буффера объемом в 10 байт, в строке так же находится аддрес этой комманды в виртуальной памяти. Как видим, на следующей строке адресс не ...00, а ...0A, а A это как раз 10 в шестнадцатиричной системе. То есть эти 10 байтов мы и зарезервировали

### 2.4.2 Создадим ошибку в коде и посмотрим на листинг

Уберем один операнд из комманды mov и посмотрим, что с станет с листингом.

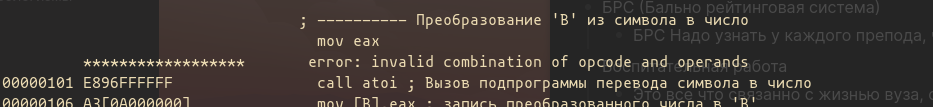


Рис. 9: Ошибка в листинге

На удивление, листинг сгенирировался и мы можем видеть сообщение об ошибке после испорченной инструкции

# 3 Задания для самостоятельной работы

## 3.1 Программа по нахождению наибольшего из трех чисел

Код программы, которая принимает на ввод 3 числа

%include "in\_out.asm"  
  
section .data  
 msg1 db "Введите a: ",0h  
 msg2 db "Введите b: ",0h  
 msg3 db "Введите c: ",0h  
 ans db "Наименьшее число: ",0h  
section .bss  
 num1 resb 20  
 num2 resb 20  
 num3 resb 20  
 min resb 20  
section .text  
  
GLOBAL \_start  
  
\_start:  
  
;;; --- READ NUM1 ---  
  
 mov eax, msg1  
 call sprintLF  
  
 mov ecx, num1  
 mov edx, 20  
 call sread  
  
 mov eax, num1  
  
 call atoi  
  
 mov [min], eax ; min = num1  
  
 ;; push eax  
 ;; mov eax, [min]  
 ;; call iprintLF  
 ;; pop eax  
   
   
  
.comparing2:   
;;; --- READ NUM2 ---  
  
  
 mov eax, msg2  
 call sprintLF  
  
 mov ecx, num2  
 mov edx, 20  
 call sread  
  
 mov eax, num2  
  
 call atoi  
  
  
 cmp [min], eax ; cmp num1, num2  
 jl .comparing3 ; if num1<num2 jpm to .comparing  
  
  
   
 mov [min], eax ; min = eax = num2  
  
  
;;; --- READ NUM3 ---  
  
  
.comparing3:   
  
 ;; push eax  
 ;; mov eax, [min]  
 ;; call iprintLF  
 ;; pop eax  
   
 mov eax, msg3  
 call sprintLF  
  
 mov ecx, num3  
 mov edx, 20  
 call sread  
  
 ;; push eax  
 ;; mov eax, [min]  
 ;; call iprintLF  
 ;; pop eax  
   
  
  
   
 mov eax, num3  
  
 call atoi  
  
  
 cmp [min], eax ; cmp num2, num3  
 jl .final ; if num2<num3 jpm to .comparing  
 mov [min], eax ; min = eax = num3  
  
  
  
   
.final:   
 mov eax, ans  
 call sprintLF  
  
 mov eax, [min]  
 call iprintLF  
  
  
call quit ;Выход

Проверим функцию на данных значениях

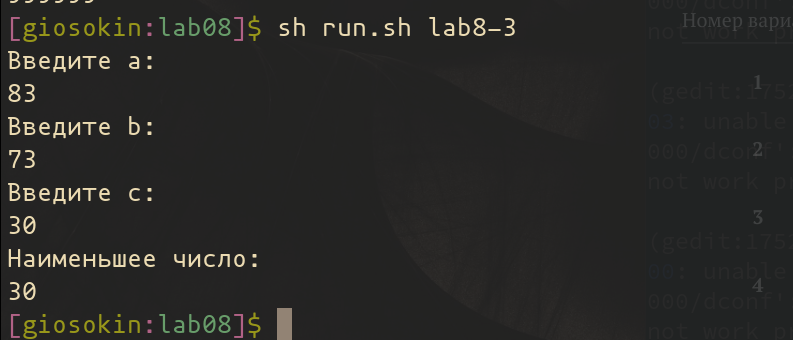


Рис. 10: Проверка первой функции на данных значениях

## 3.2 Программа по вычислению функции

Напишем программу, которая вычисляет значение функции:

Проверим функцию на данных значениях

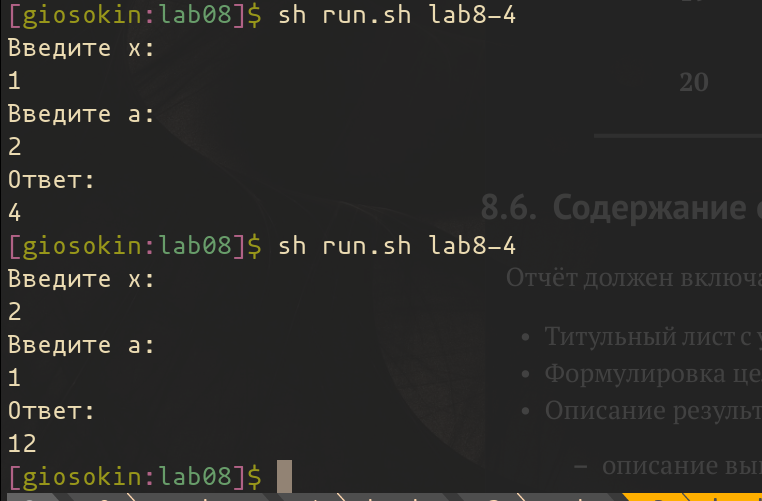


Рис. 11: Проверка второй программы на данных значениях

# 4 Выводы

Мы изучили условные переходы в языке Ассемблера NASM и научились писать программы с их использованием. Также мы ознакомились со структурой файлов листинга и написали программу по нахождению наибольшего из трех чисел и вычислению функции.