Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Студент: Верниковская Екатерина Андреевна

Содержание

# Цель работы

Освоить арифмитические инструкции языка ассемблера NASM.

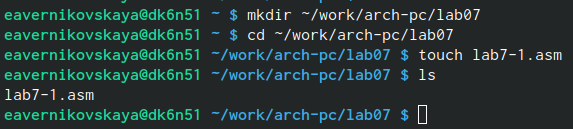
# Задание

1. Создать каталог для программ лабораторной работы №7 и в нём создать файл «lab7-1.asm».
2. Ввести в файл «lab7-1.asm» определённый текст программы с использованием инструкции jmp. Создать исполняемый файл и запустить его.
3. Изменить текст программы. Снова создать исполняемый файл и запустить его.
4. Опять изменить текст программы, чтобы строки выводились в нужном порядке, создать исполняемый файл и запустить его.
5. Создать файл «lab7-2.asm» и ввести в него определённый текст. Создать исполняемый файл и запустить его.
6. Создать файл листинга для программы из файла «lab7-2.asm». Открыть его с помощью любого текстового редактора. Подробно объяснитьсодержимое трёх строк файла истинга по выбору.
7. Открыть файл «lab7-2.asm» и в любой интсрукции двумя операндами удалить один операнд. Ответить на поставленные вопрсы.
8. Напишите программу, которая находит из 3 целочисленных переменных наименьшее. Значения переменных брять из нужной таблицы в соответствии с вариантом, полученным при выполнении лабораторной работы №6 (У меня 17 вариант).
9. Написать программу, которая для введённых значений x и a вычисляет значение заданной функции F(x). Вид функции и значения брать из определённой таблицы, в соответствии с полученным вариантом (В нашем случае это 17 вариант).

# Выполнение лабораторной работы

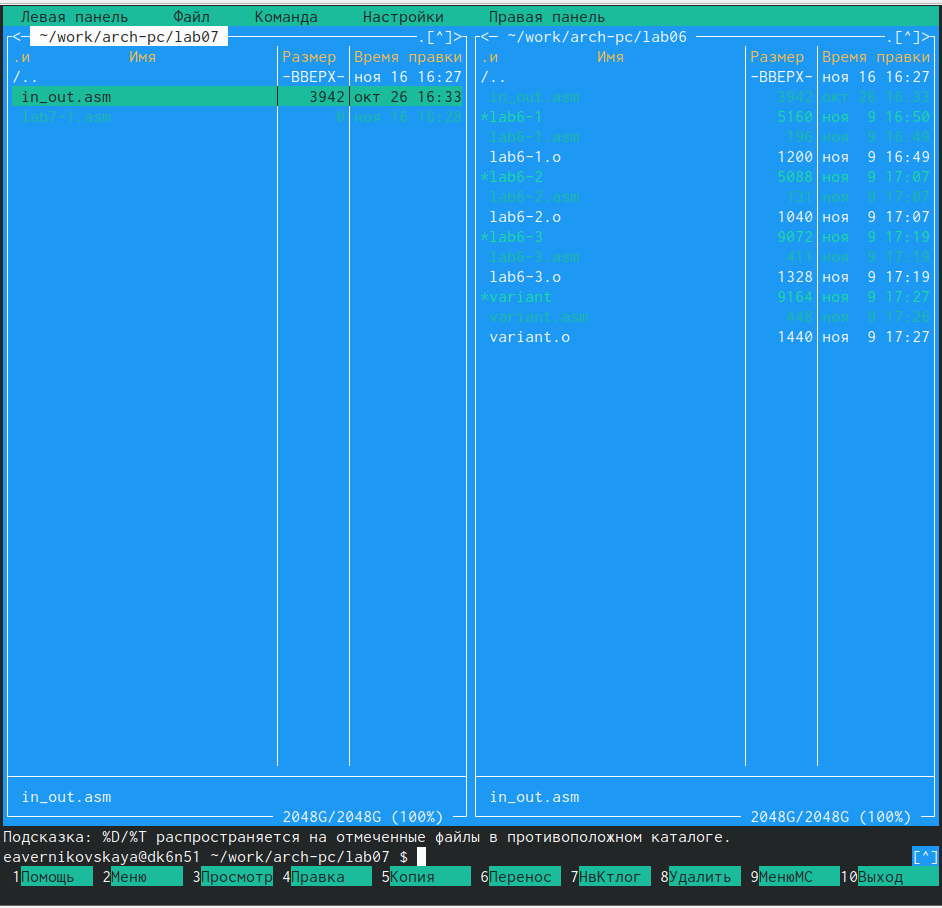
## Реализация переходв в NASM

В созданном каталоге «~/work/arch-pc/lab07» создаём файл «lab7-1.asm» (рис. [-@fig:001])



Создание первого файла

Копируем из каталога «~/work/arch-pc/lab06» файл «in\_out.asm» (рис. [-@fig:002])

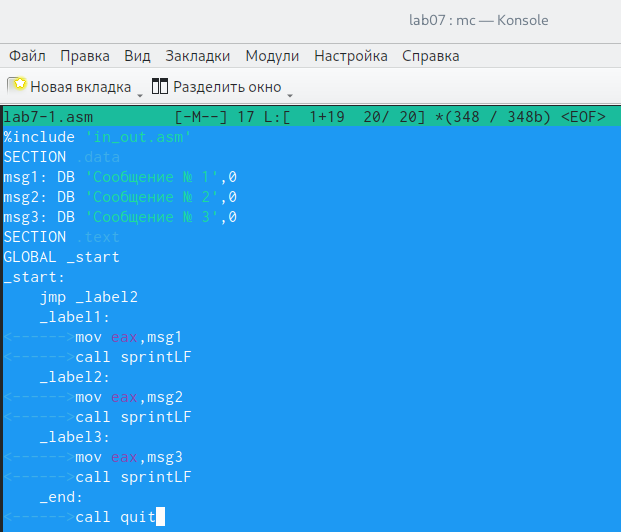


Копирование файла «in\_out.asm»

Вводим нужный текст программы с использованием инструкции jmp (рис. [-@fig:003])

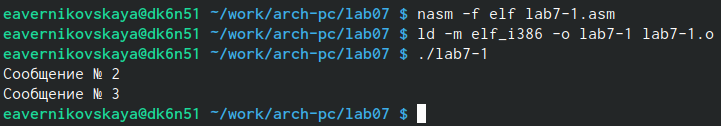
Текст программы:

%include 'in\_out.asm'   
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text   
GLOBAL \_start   
\_start:  
 jmp \_label2  
 \_label1:  
 mov eax,msg1  
 call sprintLF   
 \_label2:  
 mov eax,msg2  
 call sprintLF  
 \_label3:  
 mov eax,msg3  
 call sprintLF  
 \_end:  
 call quit



Ввод текста программы с инструкцией jmp

Создаём исполняемый файл и запускаем его (рис. [-@fig:004])

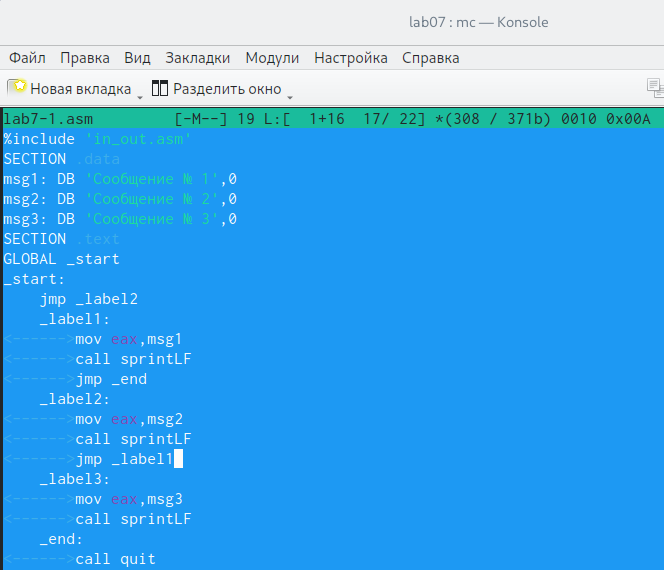


Создание исполняемого файла и его запуск

Изменяем текст программы так, чтобы она выводила сначала ‘Сообщение № 2’ а потом ‘Сообщение № 1’ и завершала работу (рис. [-@fig:005])

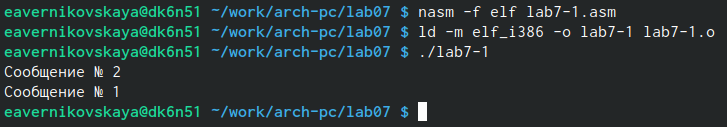
Изменённый текст программы:

%include 'in\_out.asm'   
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text   
GLOBAL \_start   
\_start:  
 jmp \_label2  
 \_label1:  
 mov eax,msg1  
 call sprintLF  
 jmp \_end  
 \_label2:  
 mov eax,msg2  
 call sprintLF  
 jmp \_label1  
 \_label3:  
 mov eax,msg3  
 call sprintLF  
 \_end:  
 call quit



Изменение программы

Снова создаём исполняемый файл и запускаем его (рис. [-@fig:006])

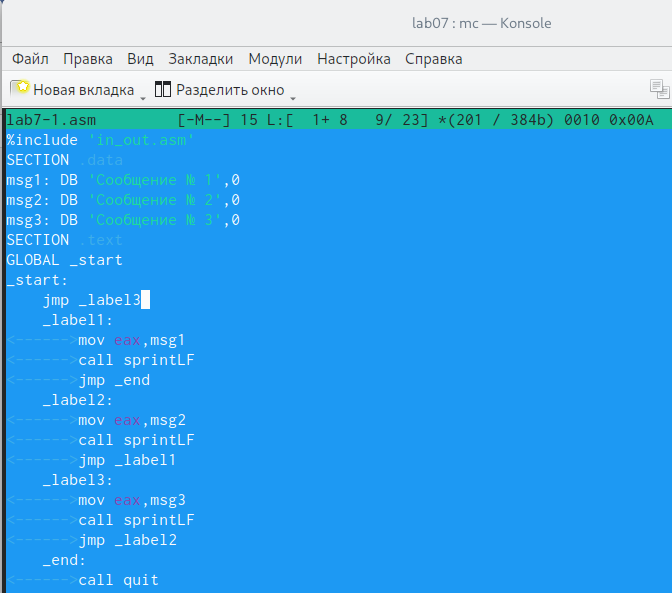


Исполняемый файл + запуск

Изменяем текст программы так, чтобы она выводила сообщения в обратном порядке, т.е. сначала 3, потом 2, потом 1 и завершала работу (рис. [-@fig:007])

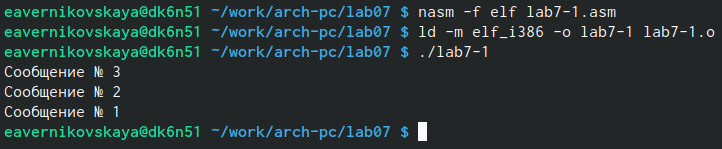
Изменённый текст программы:

%include 'in\_out.asm'   
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  
 call sprintLF  
 jmp \_end  
 \_label2:  
 mov eax,msg2  
 call sprintLF  
 jmp \_label1  
 \_label3:  
 mov eax,msg3  
 call sprintLF  
 jmp \_label2  
 \_end:  
 call quit



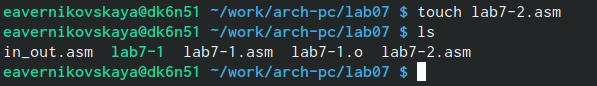
Изменение программы

Опять создаём исполняемый файл и запускаем его (рис. [-@fig:008])



Исполняемый файл + запуск

Создаём файл «lab7-2.asm» (рис. [-@fig:009])

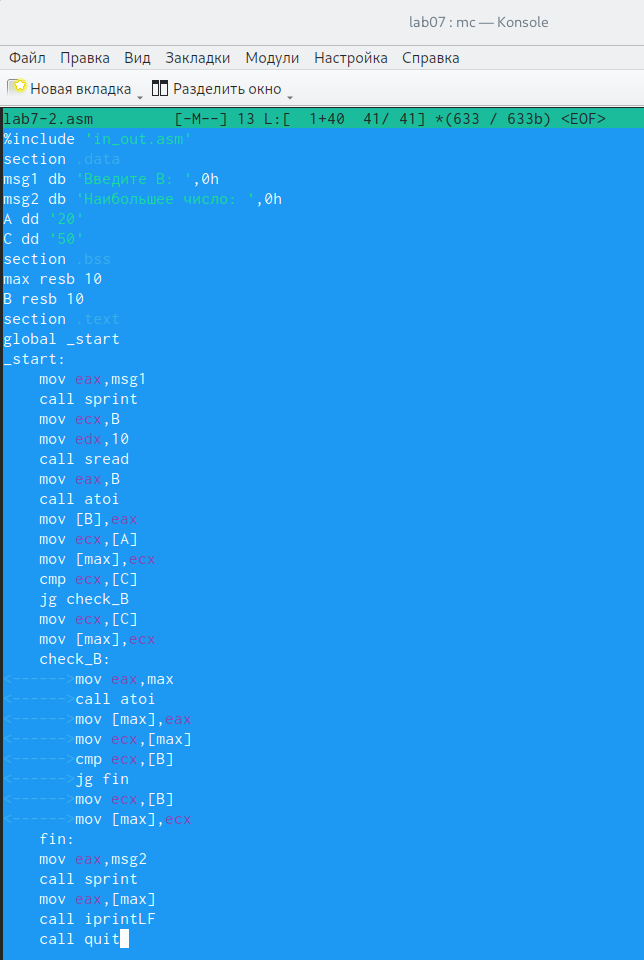


Создание файла «lab7-2.asm»

Вводим текст программы, которая определяет и выводит на экран наибольшее число из 3 целочисленных переменных A, B, C (рис. [-@fig:010])

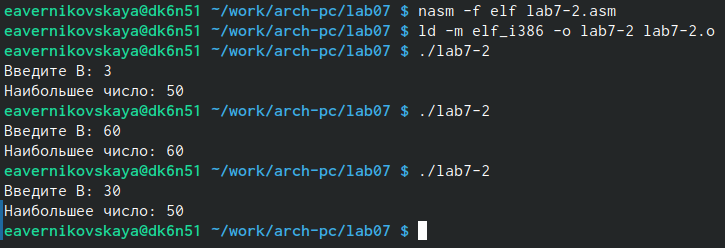
Текст программы:

%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:  
 mov eax,msg1  
 call sprint  
 mov ecx,B  
 mov edx,10  
 call sread  
 mov eax,B  
 call atoi  
 mov [B],eax  
 mov ecx,[A]  
 mov [max],ecx  
 cmp ecx,[C]  
 jg check\_B  
 mov ecx,[C]  
 mov [max],ecx  
 check\_B:  
 mov eax,max  
 call atoi  
 mov [max],eax  
 mov ecx,[max]  
 cmp ecx[B]  
 jg fin  
 mov ecx,[B]  
 mov [max],ecx  
 fin:  
 mov eax,msg2  
 call sprint  
 mov eax,[max]  
 call iprintLF  
 call quit



Ввод текста программы

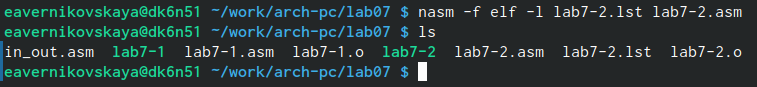
Создаём исполняемый файл и запускаем его (рис. [-@fig:011])



Создание исполняемого файла и его запуск

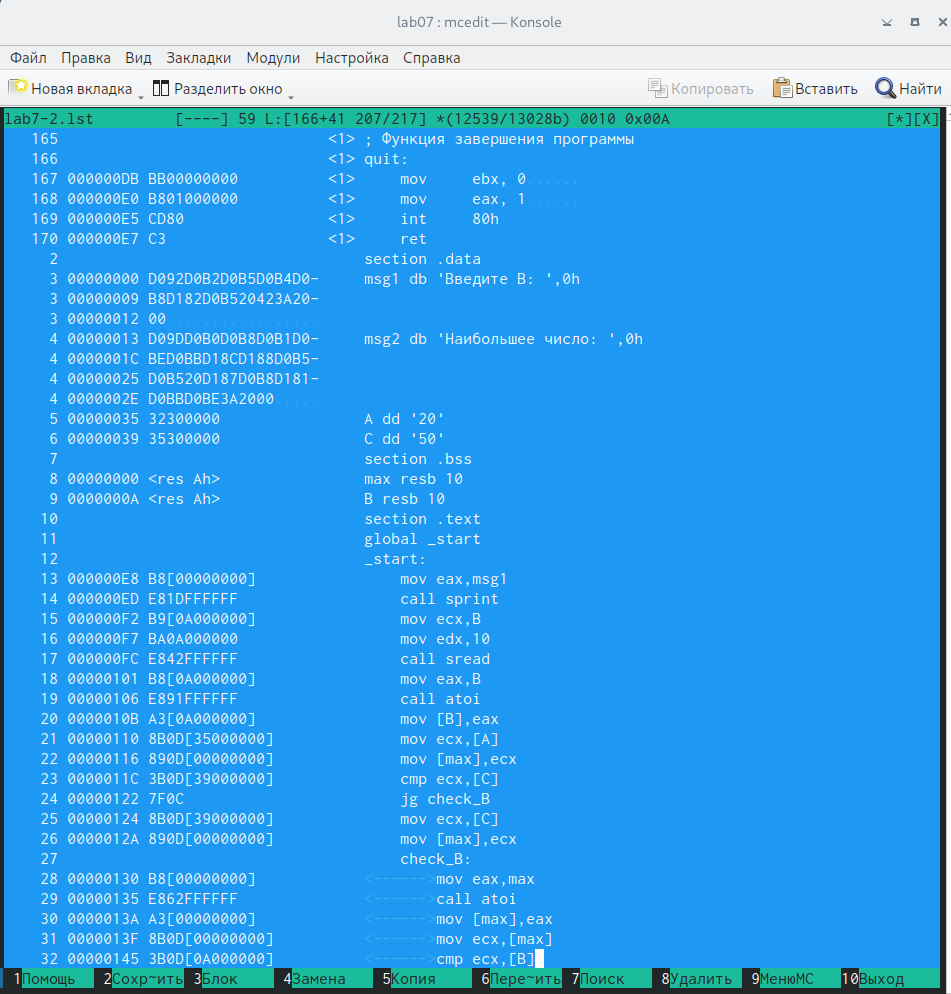
## Изучение структуры файла листинга

Создаем файл лситинга для программы из файла «lab7-2.asm» (рис. [-@fig:012])



Создание листинга

Открываем файл листинга с помощью текстового редактора (рис. [-@fig:013])

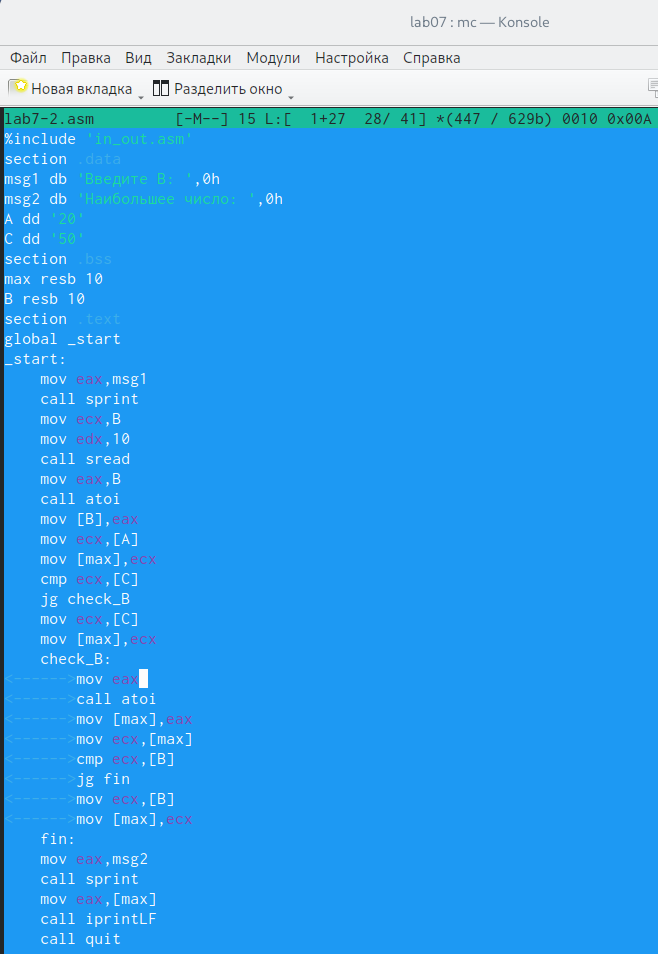


Открытый файл листинга

Опишем строки под номерами 13, 14 и 15.

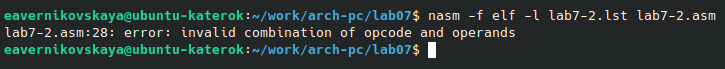
1. 13 (номер строки); 000000E8 (адрес, начинается по смещению 000000E8 в сешменте кода); B8[00000000] (машинный код); mov eax,msg1 (исходный текст программы, в котором мы перемещаем адрес метки msg1 в регистр eax. msg1 - адрес строки, которую мы хотим вывести)
2. 14 (номер строки); 000000ED (адрес, начинается по смещению 000000ED в сегменте кода); E81DFFFFFF (машинный код); call sprint (исходный текст программы, в котором мы вызываем подпрограмму печати сообщения)
3. 15 (номер строки); 000000F2 (адрес, начинается по смещению 000000F2 в сегменте кода); B9[0A000000] (машинный код); mov ecx,B (исходный текст программы, в котором мы записываем адрес введённой переменной в ecx)

Снова открываем файл «lab7-2.asm» и в любой строке с двумя операндами удаляем один операнд. В 28 строке удаляем операнд ‘max’ (рис. [-@fig:014])



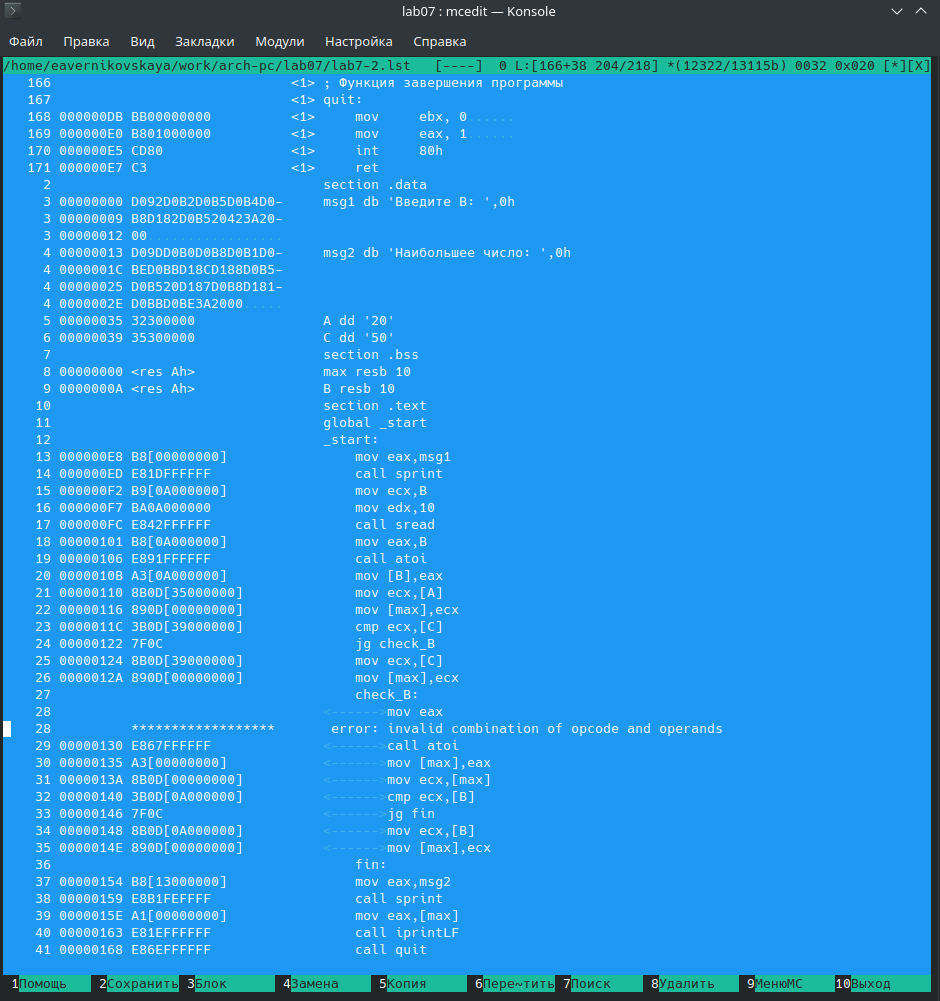
Удаление одного операнда

Выполняем трансляцию с получением файла листинга (рис. [-@fig:015])



Получение файла листинга

В ходе трансляции система выдаёт ошибку и создаёт файлы «lab7-2» и «lab7-2.lst». Заходим в листинг для изучения того, что добавилось в него после возникновения ошибки (рис. [-@fig:016])

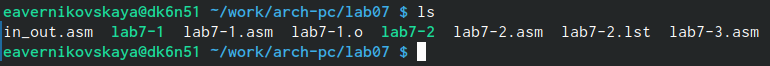


Файл листинга

В 28 строке выводится сообщение об ошибке

## Задание для самостоятельной работы

Создаём файл «lab7-3.asm» (рис. [-@fig:017])

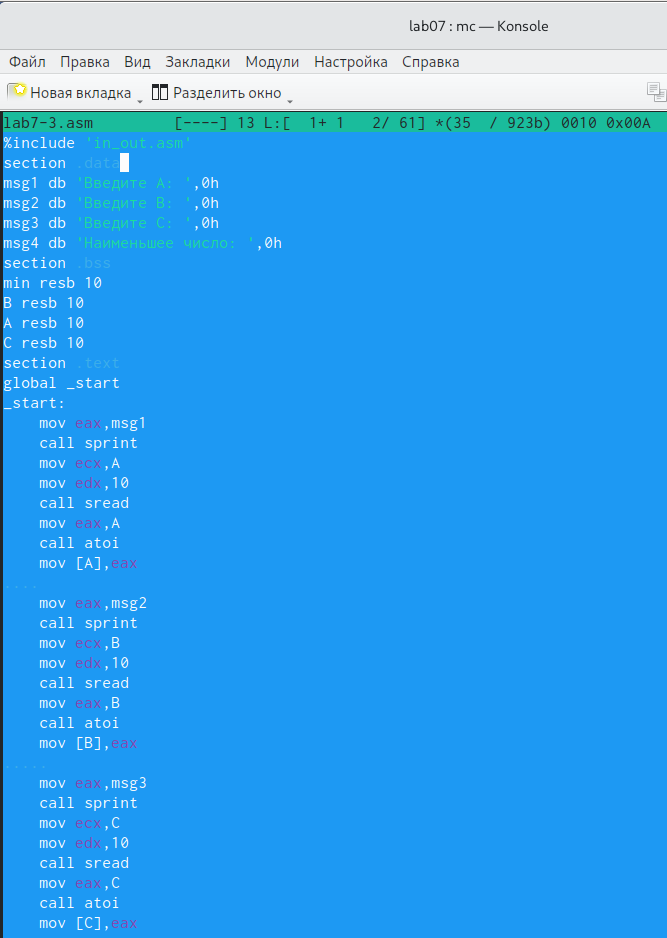


Создание файла «lab7-3.asm»

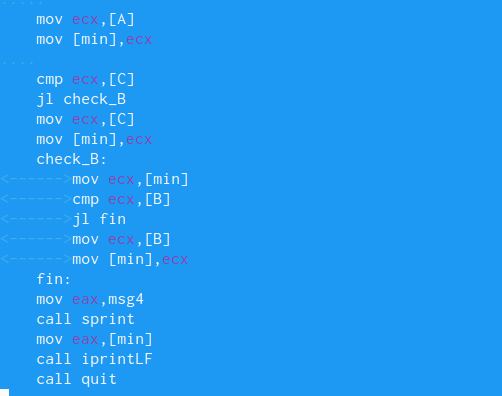
Вводим текст программы, которая находит наименьшее число из 3 целочисленных введённых переменных A, B и C (рис. [-@fig:018]) и (рис. [-@fig:019])

Текст программы:

%include 'in\_out.asm'  
section .data  
msg1 db 'Введите A: ',0h  
msg2 db 'Введите B: ',0h  
msg3 db 'Введите C: ',0h  
msg4 db 'Наименьшее число: ',0h  
section .bss  
min resb 10  
B resb 10  
A resb 10  
C resb 10  
section .text  
global \_start  
\_start:  
 mov eax,msg1  
 call sprint  
 mov ecx,A  
 mov edx,10  
 call sread  
 mov eax,A  
 call atoi  
 mov [A],eax  
   
 mov eax,msg2  
 call sprint  
 mov ecx,B  
 mov edx,10  
 call sread  
 mov eax,B  
 call atoi  
 mov [B],eax  
   
 mov eax,msg3  
 call sprint  
 mov ecx,C  
 mov edx,10  
 call sread  
 mov eax,C  
 call atoi  
 mov [C],eax  
   
 mov ecx,[A]  
 mov [min],ecx  
   
 cmp ecx,[C]  
 jl check\_B  
 mov ecx,[C]  
 mov [min],ecx  
 check\_B:  
 mov ecx,[min]  
 cmp ecx,[B]  
 jl fin  
 mov ecx,[B]  
 mov [min],ecx  
 fin:  
 mov eax,msg4  
 call sprint  
 mov eax,[min]  
 call iprintLF  
 call quit



Ввод программы 1



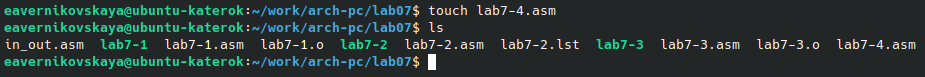
Ввод программы 2

Cоздаём исполняемый файл и запускаем его. Проверяем работу программы для введённых значений (A = 26; B = 12; C = 68) из нужной таблицы (рис. [-@fig:020])



Исполняемый файл + запуск + проверка

Создаём файл «lab7-4.asm» (рис. [-@fig:021])

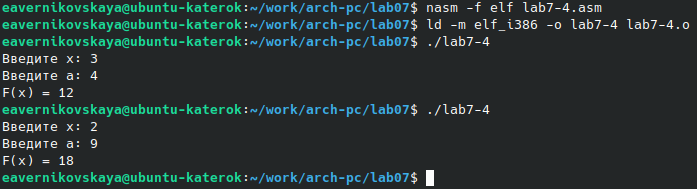


Создание файла «lab7-4.asm»

Вводим текст программы. Программа вычисляет значение функции (F(x): a+8, если a<8 и a\*x, если a>=8) и выводит результат вычислений (рис. [-@fig:022])

%include 'in\_out.asm'  
SECTION .data  
msg1: DB 'Введите x: ',0h  
msg2: DB 'Введите a: ',0h  
otv: DB 'F(x) = ',0h  
SECTION .bss  
x: RESB 80  
a: RESB 80  
res: RESB 80  
SECTION .text  
GLOBAL \_start  
\_start:  
 mov eax,msg1  
 call sprint  
 mov ecx,x  
 mov edx,80  
 call sread  
 mov eax,x  
 call atoi  
 mov [x],eax  
  
 mov eax,msg2  
 call sprint  
 mov ecx,a  
 mov edx,80  
 call sread  
 mov eax,a  
 call atoi  
 mov [a],eax  
  
 mov edx,8  
 cmp edx,[a]  
 jg check\_1  
 mov eax,[a]  
 mov ebx,[x]  
 mul ebx  
 mov [res],eax  
 jmp fin  
 check\_1:  
   
 mov eax,[a]  
 add eax,8  
 mov [res],eax  
 jmp fin  
  
 fin:  
 mov eax,otv  
 call sprint  
 mov eax,[res]  
 call iprintLF  
 call quit

Cоздаём исполняемый файл и запускаем его. Проверяем работу программы для введённых значений (x1=3, a1=4 и x2=2, a2=9) из нужной таблицы (рис. [-@fig:023])



Исполняемый файл + запуск + проверка

# Выводы

В ходе выполнения лабораторной работы мы изучили команды условного и безусловного переходов, приобрели навыки написания программ с использованием переходов. Также мы познакомились с назначением и структурой файла листинга.