Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Седохин Даниил Алексеевич

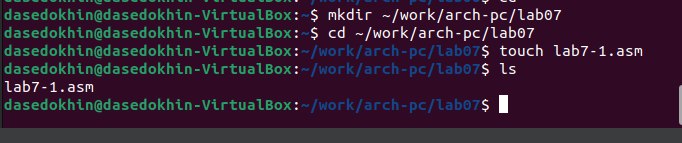
Содержание

# 1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Ознакомиться с назначением и структурой файла листинга.

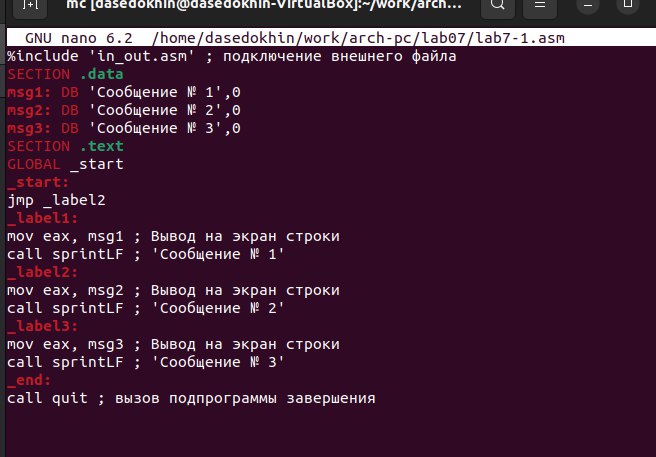
# 2 Выполнение лабораторной работы

1. Создадим каталог для лабораторной работы № 7, перейдём в него и создадим файл lab7-1.asm (рис. ??).



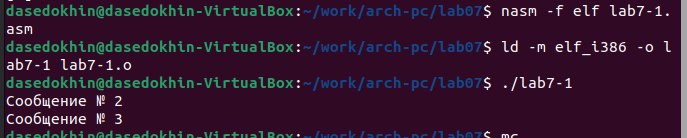
Создание каталога для лабораторной работы №7 и файла lab7-1.asm

1. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введём в файл lab7-1.asm текст программы из листинга 7.1. (рис. ??).



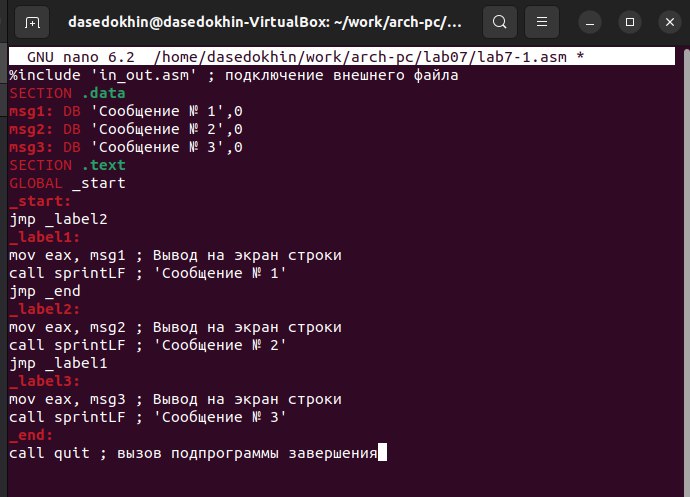
Ввод в файл lab7-1.asm текст программы из листинга 7.1

Создадим исполняемый файл и запустим его. (рис. ??).



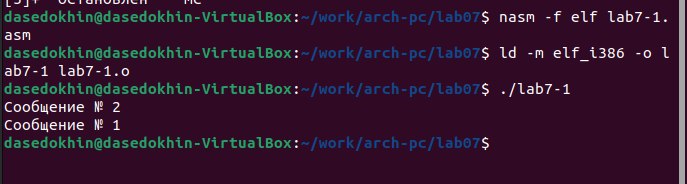
Создание и проверка исполняемого файла

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию jmp с меткой \_label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой \_end (т.е. переход к инструкции call quit). Изменим текст программы в соответствии с листингом 7.2 (рис. ??).



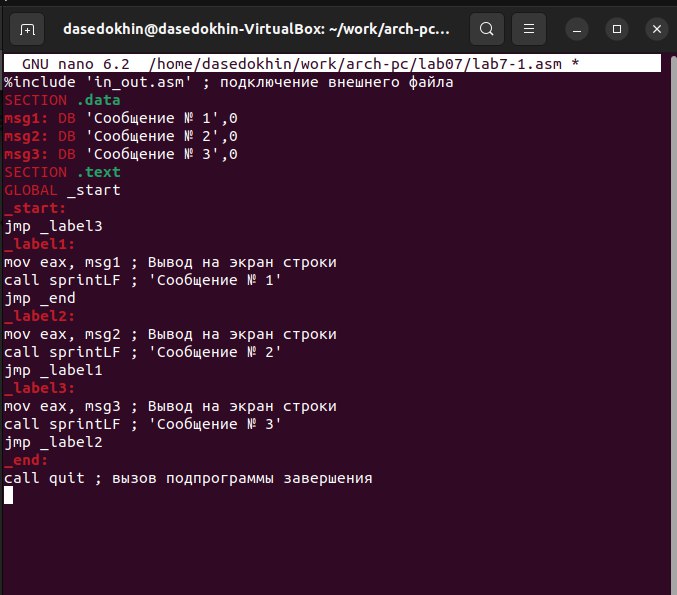
Изменение текста программы в соответствии с листингом 7.2

Создадим исполняемый файл и проверим его работу. (рис. ??).



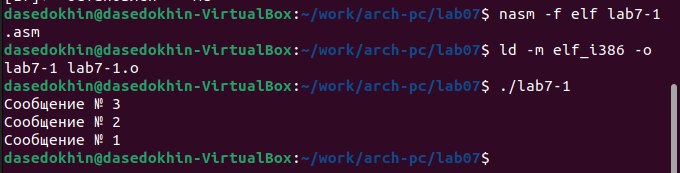
Создание и проверка исполняемого файла

Изменим текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: (рис. ??).  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1



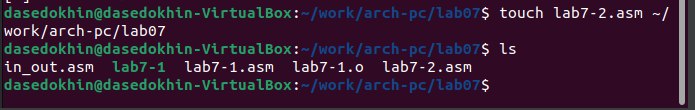
Редактирование текста программы

Создадим исполняемый файл и проверим его. (рис. ??).



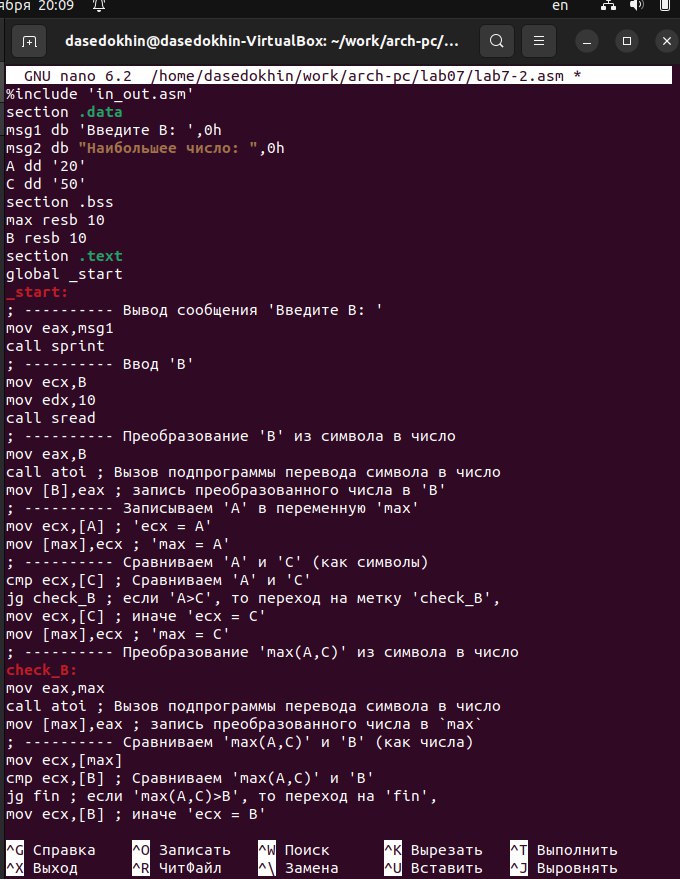
Создание и проверка исполняемого файла

1. Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры. Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. ??).



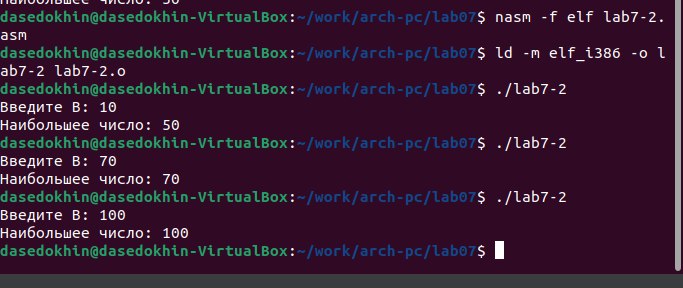
Создание файла lab7-2.asm

Введём текст программы из листинга 7.3 в lab7-2.asm. (рис. ??).



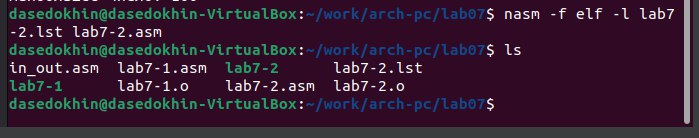
Ввод текста программы из листинга 7.3 в lab7-2.asm

Создадим исполняемый файл и проверим его работу для разных значений B. (рис. ??).



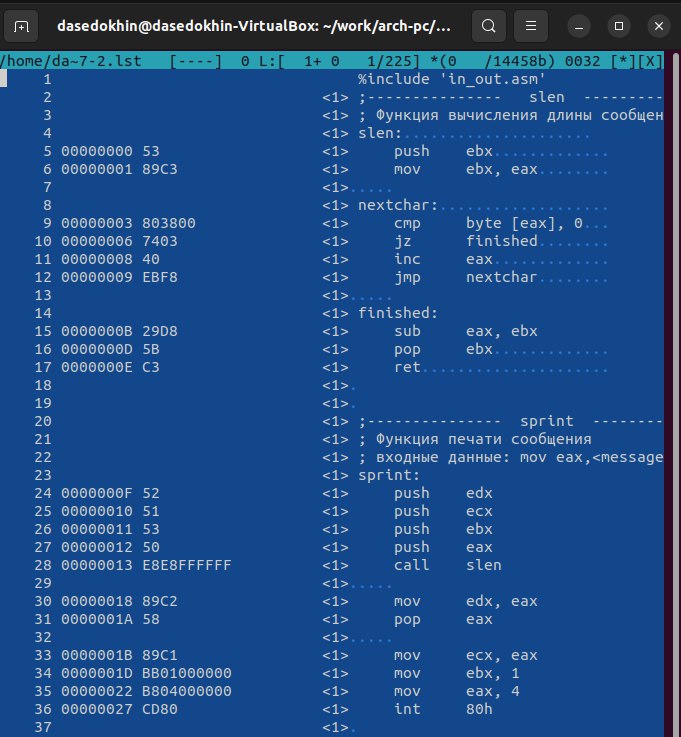
Создание и проверка исполняемого файла

1. Создадим файл листинга для программы из файла lab7-2.asm. (рис. ??).



Создание и проверка исполняемого файла

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора mcedit. (рис. ??).



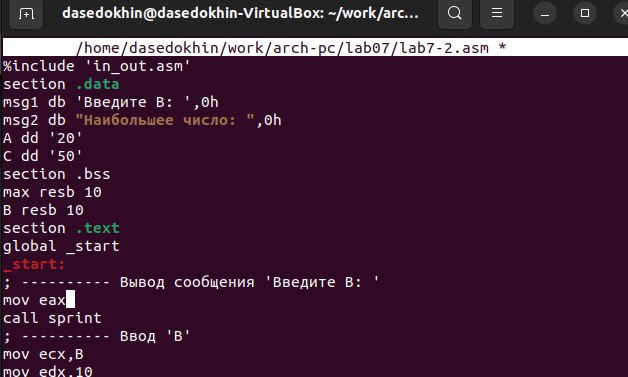
Просмотра файла листинга lab7-2.lst

Три строки выбранных мною следующие: (рис. ??): - 45 - строчка, 00000159 адрес, B8[13000000] - машинный код, mov eax, msg2 - исходный тест программы. В котором записывается адрес msg2 в EAX. - 46 - строчка, 0000015E адрес, E8ACFEFFFF - машинный код, call sprint - исходный тест программы. В котором вызывается функция sprint. Она в свою очередь выводит собщение содержащееся в переменной msg2 - 47 - строчка, 00000163 адрес, A1[00000000] - машинный код, mov eax,[max] - исходный тест программы. В котором записывается адрес переменной [max] в EAX.

Выбранные строки

Выбранные строки

Откроем файл с программой lab7-2.asm и уберём операнд из 14 строки. (рис. ??).

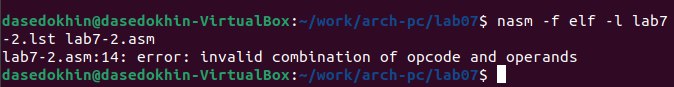


Изменённая строка

Создадим файл листинга в котором на 14 строке появилась ошибка в связи с нашими изменениями операнды. (рис. ?? ??).

Изменения в созданном файле листинга

Изменения в созданном файле листинга



Создание файла листинга

# 3 Задание для самостоятельной работы

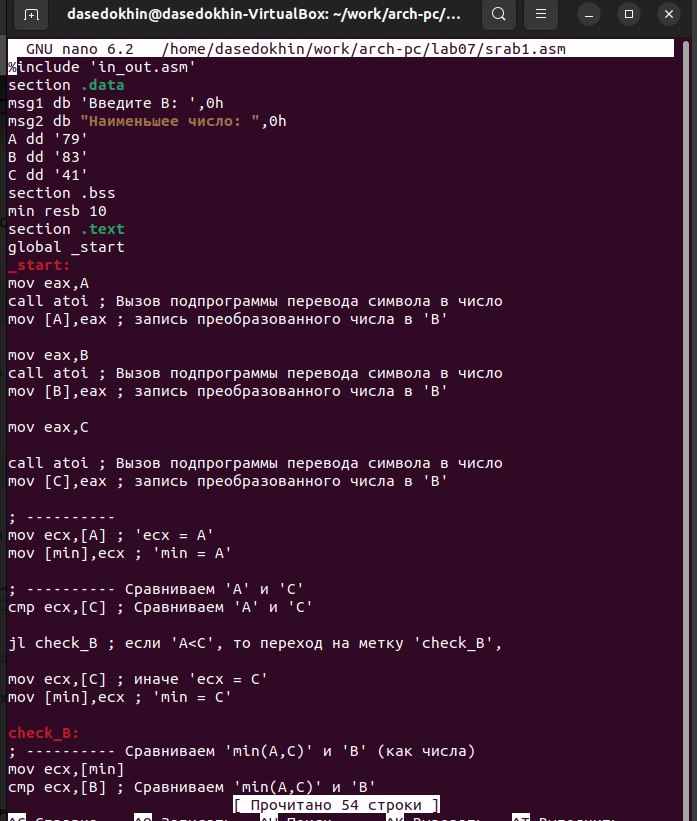
Напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и с. Значения переменных будут равны (79,83,41) в соответствии с 6-м вариантом.

Для начала создадим файл srab1.asm в каталоге ~/work/arch-pc/lab07. (рис. ??).

Создание файла srab1.asm

Создание файла srab1.asm

За основу программы возьмём текст из листинга 7.3. Внесём необходимые изменения в текст файла. (рис. ??).

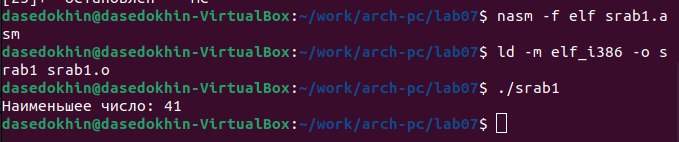


Редактирование текста файла srab1.asm

***Изменённый текст файла srab1.asm***

%include 'in\_out.asm'  
section .data  
msg2 db "Наименьшее число: ",0h  
A dd '79'  
B dd '83'  
C dd '41'  
section .bss  
min resb 10  
section .text  
global \_start  
\_start:  
  
  
mov eax,A  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [A],eax ; запись преобразованного числа в 'B'  
  
mov eax,B  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [B],eax ; запись преобразованного числа в 'B'  
  
mov eax,C  
call atoi ; Вызов подпрограммы перевода символа в число  
mov [C],eax ; запись преобразованного числа в 'B'  
  
; ----------   
mov ecx,[A] ; 'ecx = A'  
mov [min],ecx ; 'min = A'  
  
; ---------- Сравниваем 'A' и 'С'  
cmp ecx,[C] ; Сравниваем 'A' и 'С'  
  
jl check\_B ; если 'A<C', то переход на метку 'check\_B',  
  
mov ecx,[C] ; иначе 'ecx = C'  
mov [min],ecx ; 'min = C'  
  
check\_B:  
; ---------- Сравниваем 'min(A,C)' и 'B' (как числа)  
mov ecx,[min]  
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'  
  
jl fin  
   
mov ecx,[B] ;иначе 'ecx = B'  
mov [min],ecx  
; ---------- Вывод результата  
fin:  
mov eax, msg2  
call sprint ; Вывод сообщения 'Наибольшее число: '  
mov eax,[min]  
call iprintLF ; Вывод 'min(A,B,C)'  
call quit ; Выход

Создадим исполняемый файл и проверим его работу. (рис. ??).



Создание и проверка исполняемого файла

# 4 Выводы

Я изучил команды условного и безусловного переходов. Приобрёл навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.