Отчет по выполнению лабораторной работы №7

Дисциплина: Архитектура компьютера

Ефремова Полина Александровна

Содержание

# 1 Цель работы

Цеоль данной работы - изучить команды условного и безусловного переходов. Кроме этого, - приобрести навыки написания программ с использованием переходов, а также познакомиться с назначением и структурой файла листинга.

# 2 Задание

1. Реализовать переходы в NASM
2. Изучить структуру файлов листинга
3. Выполнить 2 задания для самостоятельной работы: написать программы для решения функции и для нахождения минимального значения среди чисел.

# 3 Теоретическое введение

Выделяют 2 типа переходов/команд передачи управления в ассемблере для реализации ветвления:

*• условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.*

*• безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.*

## 3.1 Команды безусловного перехода

Безусловный переход выполняется инструкцией **jmp (от англ. jump – прыжок)**, которая включает в себя адрес перехода, куда следует передать управление.

## 3.2 Команды условного перехода

Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

#### 3.2.0.1 Регистр флагов

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

#### 3.2.0.2 Описание инструкции cmp

Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

Команда cmp, так же как и команда вычитания, выполняет вычитание - , но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

Команда условного перехода имеет вид:

**j label**

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

#### 3.2.0.3 Файл листинга и его структура

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Cтруктура листинга:

• *номер строки* — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы); • *адрес* — это смещение машинного кода от начала текущего сегмента; • машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра); • *исходный текст программы* — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

# 4 Выполнение лабораторной работы

1. Создаю файл lab7-1.asm (рис. 1).

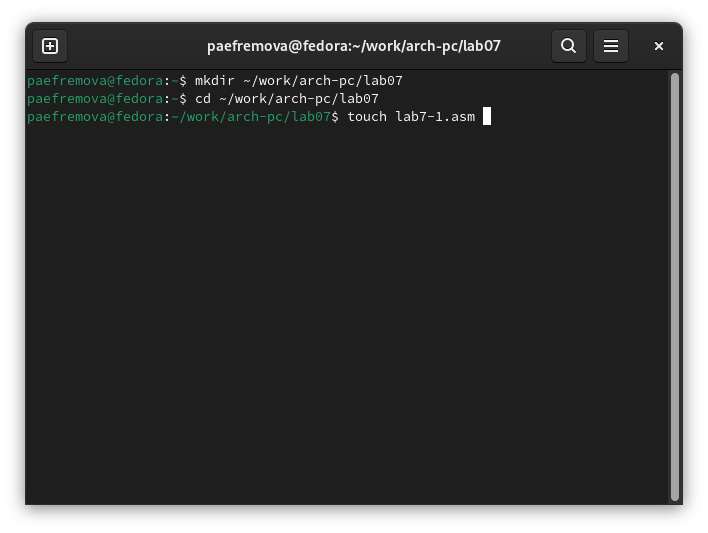


Рис. 1: Создание файла 1

1. Ввожу в файл lab7-1.asm программу (рис. 2).

Листинг 7.1. Программа с использованием инструкции jmp

%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

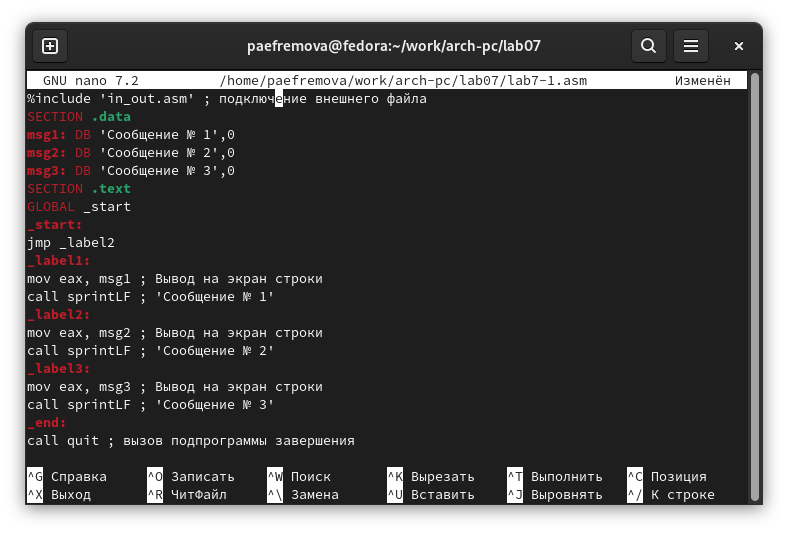


Рис. 2: Ввод программы 1

1. Запускаю файл lab7-1.asm (рис. 3).

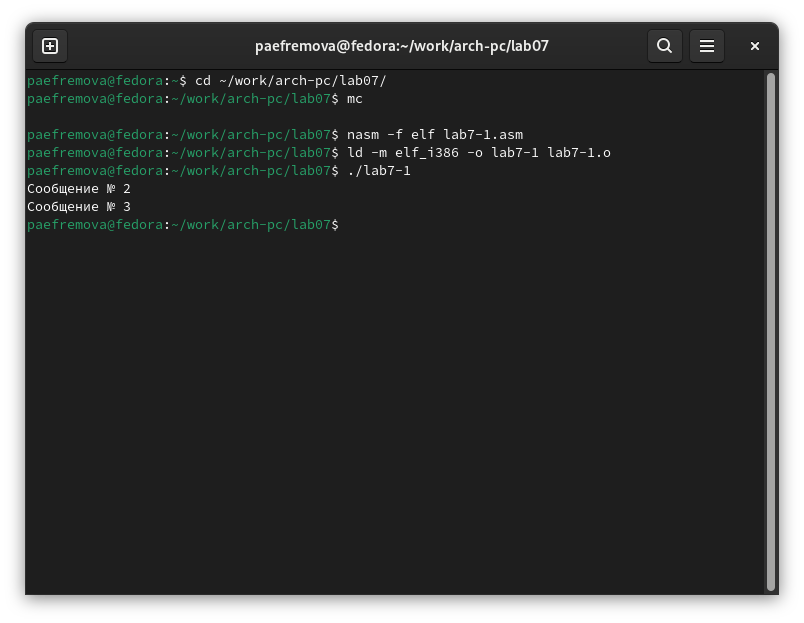


Рис. 3: Запуск файла 1

1. Создаю копию файла и вношу в него изменения (рис. 4).

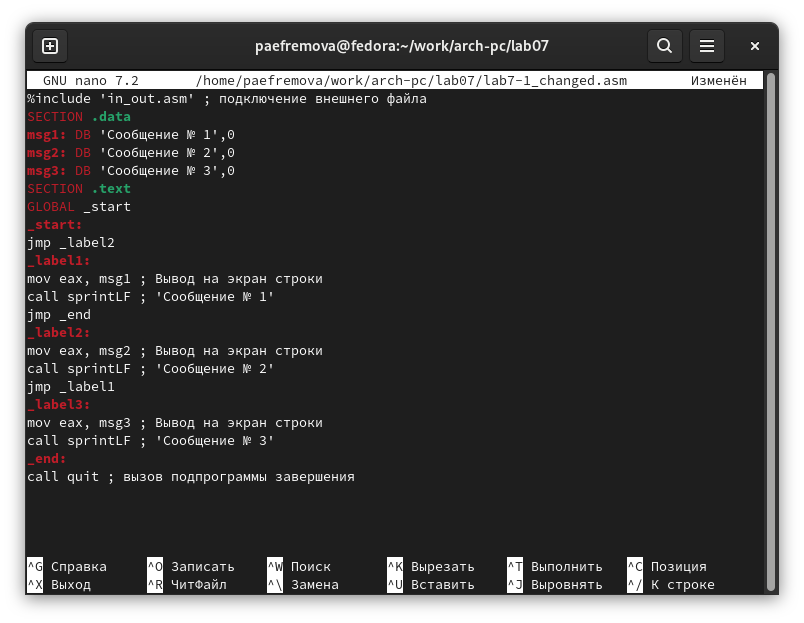


Рис. 4: Изменение файла 1

Листинг 7.2. Программа с использованием инструкции jmp

%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

1. Запуск измененного файла (рис. 5).

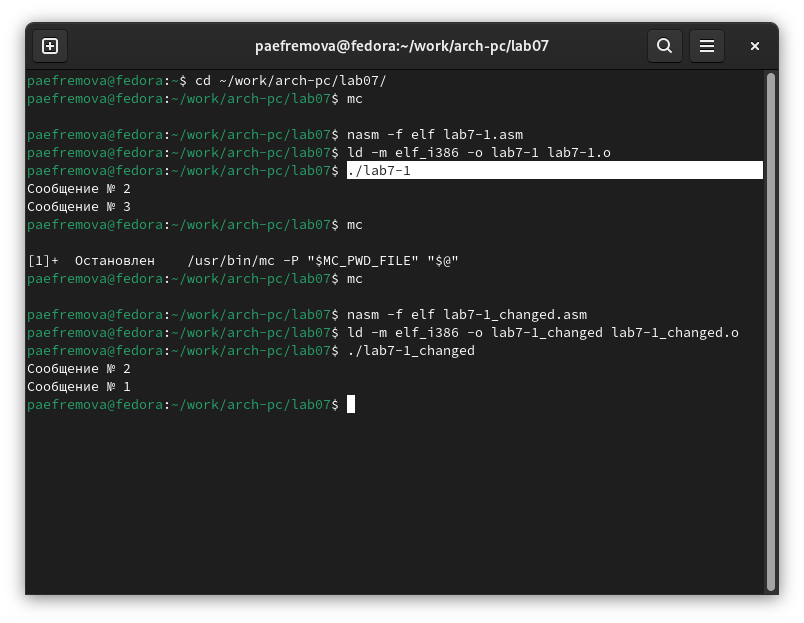


Рис. 5: Запуск измененного файла

1. В этом же файле переделываю команды так, чтобы сообщения выводились иным образом (рис. 6).

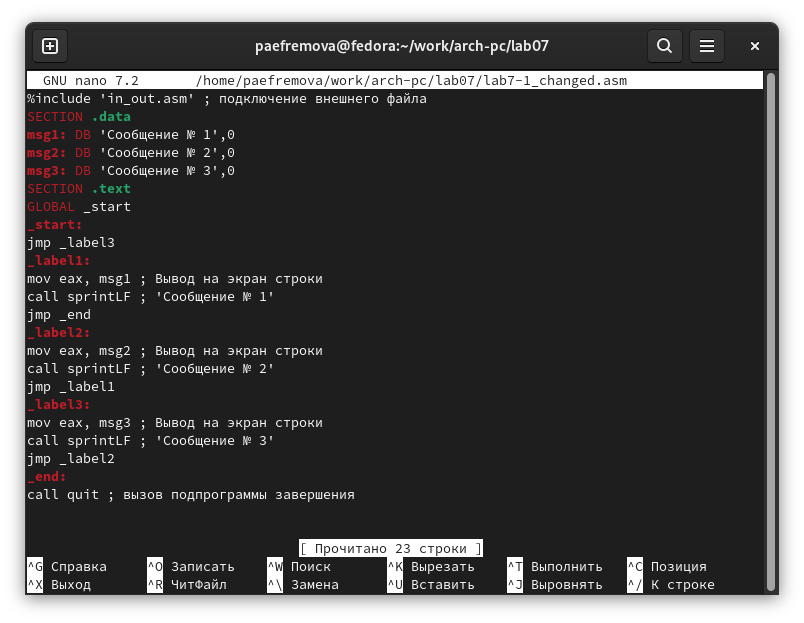


Рис. 6: Изменяю файл еще раз

Измененный мной листинг:

%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

1. Запускаю измененный файл (рис. 7).

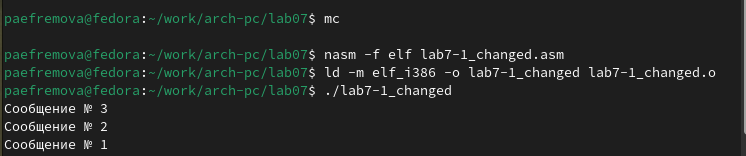


Рис. 7: Запуск

1. Создаю файл lab7-2.asm (рис. 8).

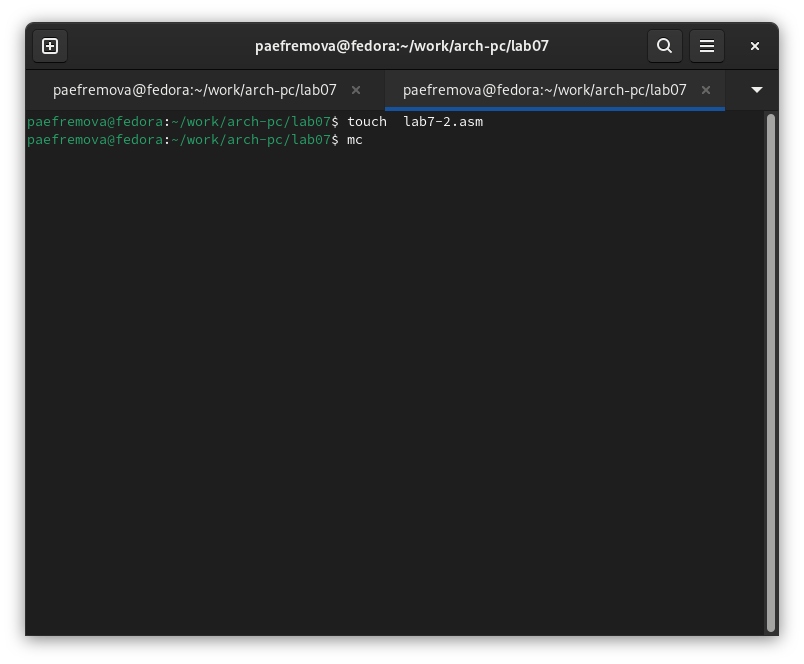


Рис. 8: Создание файла 2

1. Ввожу в файл программу из листинга 7.3. (рис. 9).

Листинг 7.3. Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C.

%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 ; Вызов подпрограммы перевода символа в число  
mov [B],eax ; запись преобразованного числа в 'B'  
  
mov ecx,[A] ; 'ecx = A'  
mov [max],ecx ; 'max = A'  
  
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'  
  
check\_B:  
mov eax,max  
call atoi  
mov [max],eax   
; ---------- Сравниваем 'max(A,C)' и 'B' (как числа)  
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

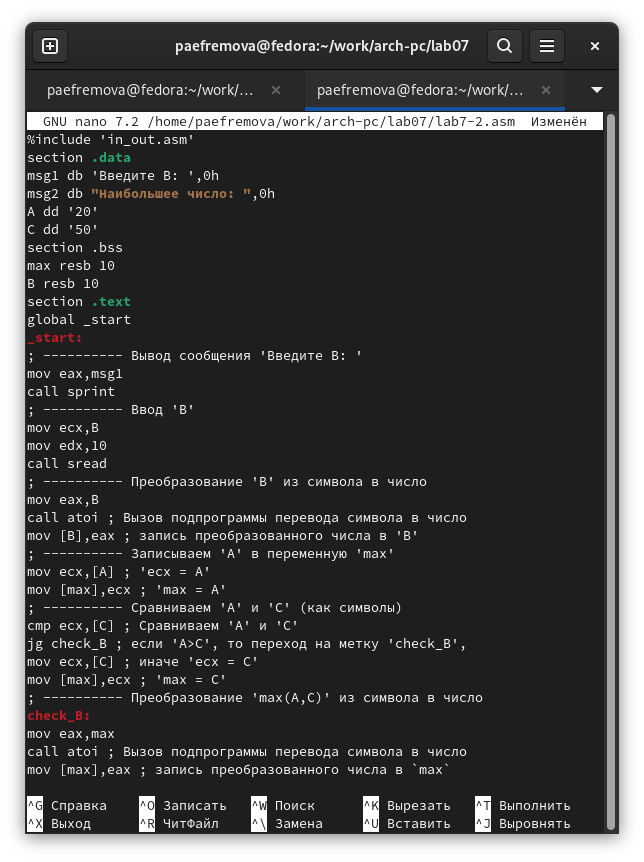


Рис. 9: Ввод программы в файл 2

1. Запуск программы из листинга 7.3. (рис. 10).

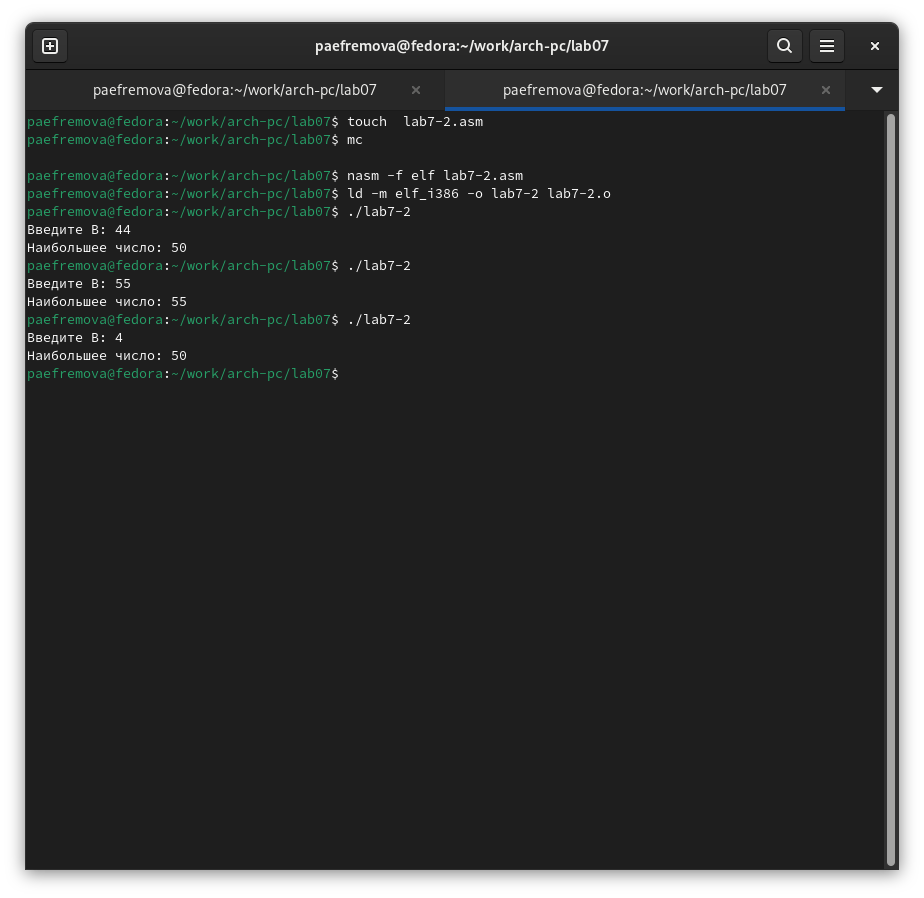


Рис. 10: Запуск программы 2

1. Создаю файл листинга программы 2. (рис. 11).

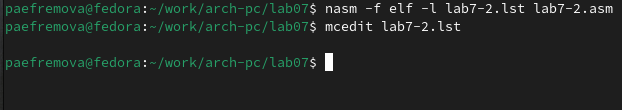


Рис. 11: Создание файла листинга

1. Открываю созданный файл (рис. 12).

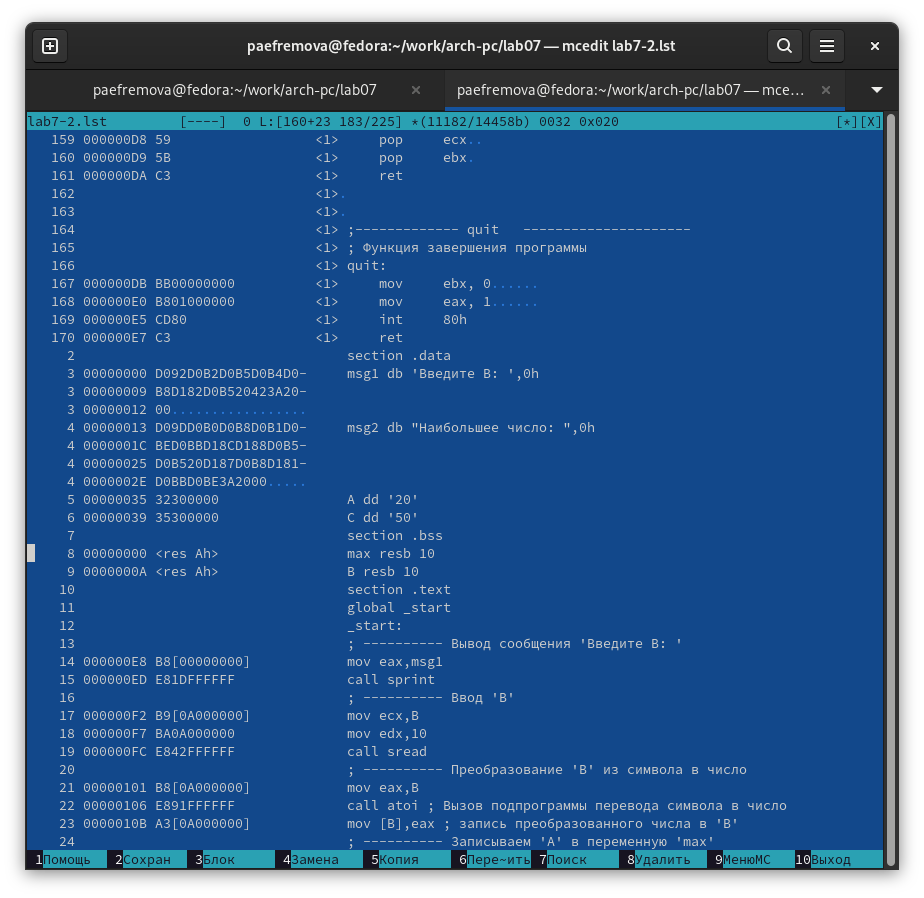


Рис. 12: Запуск программы 2

Изучение листинга, объяснение трех строк из данного файла:

Описание строки 1: вижу, что программа передает код из файла in\_out.asm. Т.е. в данном файле показан тот код, который мы включили в asm файл.

Описание строки 4: 4 - номер строки, 00000000 - смещение машинного кода относительно начала текущего сегмента, 53 - машинный код, push ebx - текст программы.

Описание строки 49: 49 - номер строки, 00000167 - смещение машинного кода относительно начала текущего сегмента, E86FFFFFFF - машинный код, call quit - текст программы, ; Выход - комментарий.

1. Открываю файл с программой lab7-2.asm и удаляю 1 операнд со сторки (на скриншоте есть курсив, там и удалила) (рис. 13).

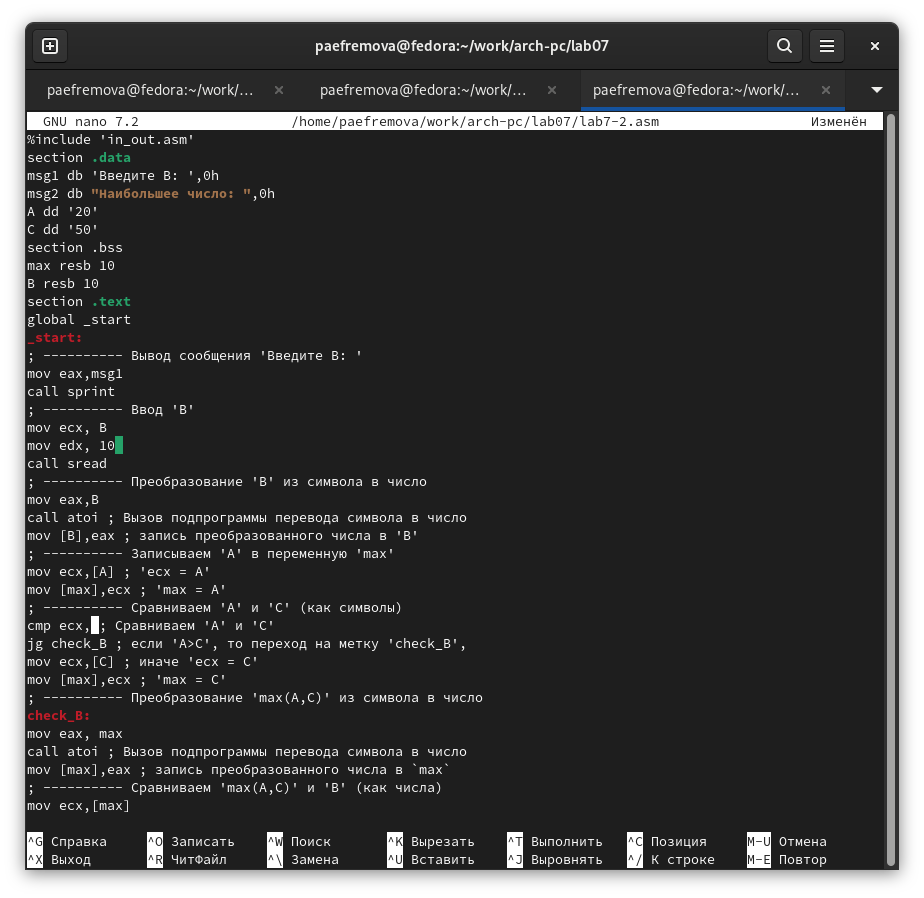


Рис. 13: Удаление операнда

1. Консоль выдает ошибку (рис. 14).

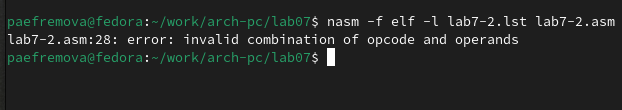


Рис. 14: В консоли ошибка

1. В листинге также появляется ошибка, при этом другие файлы не создаются. (рис. 15).



Рис. 15: В листинге тоже ошибка

# 5 Выполнение заданий для самостоятельной работы

1. Создаю файл lab7-3.asm для работы. (рис. 16).

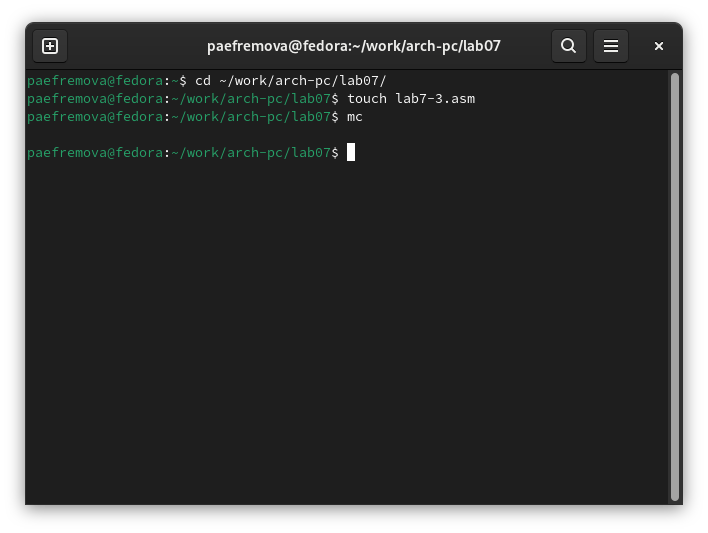


Рис. 16: Создание нового файла

Ввожу в файл программу (рис. 17).

Программа для самостоятельной работы 1:

%include 'in\_out.asm'  
section .data  
msg2 db 'Введите A: ', 0h  
msg db 'Введите B: ', 0h  
msg3 db 'Введите C: ', 0h  
msg1 db 'Наименьшее число: ',0h  
  
section .bss  
min resb 10  
A resb 10  
B resb 10  
C resb 10  
  
section .text  
global \_start  
\_start:  
  
mov eax, msg2  
call sprint  
  
mov ecx, A  
mov edx, 10  
call sread  
  
mov eax, msg  
call sprint  
  
mov ecx, B  
mov edx, 10  
call sread  
  
mov eax, msg3  
call sprint  
  
mov ecx, C  
mov edx, 10  
call sread  
  
  
  
mov eax,A  
call atoi   
mov [A],eax   
  
mov eax,B  
call atoi   
mov [B],eax   
  
mov eax,C  
call atoi   
mov [C],eax   
  
mov ecx,[A] ; ecx = A  
mov [min],ecx ; min = A  
  
cmp ecx,[C] ; A ? C  
jb check\_B ; if A < C |-> check\_B  
mov ecx,[C] ; if A > C |-> ecx = C  
mov [min],ecx ; min = C  
  
check\_B:  
mov ecx,[min] ; ecx = min(A/C)  
cmp ecx,[B] ; A/C ? B  
jb fin ; if A/C < B |-> fin  
mov ecx,[B] ; if A/C > B |-> ecx = B  
mov [min],ecx ; min = B  
  
fin:  
mov eax, msg1 ; eax = msg1  
call sprint ; вывод  
mov eax,[min] ; eax = min  
call iprintLF ; вывод  
call quit

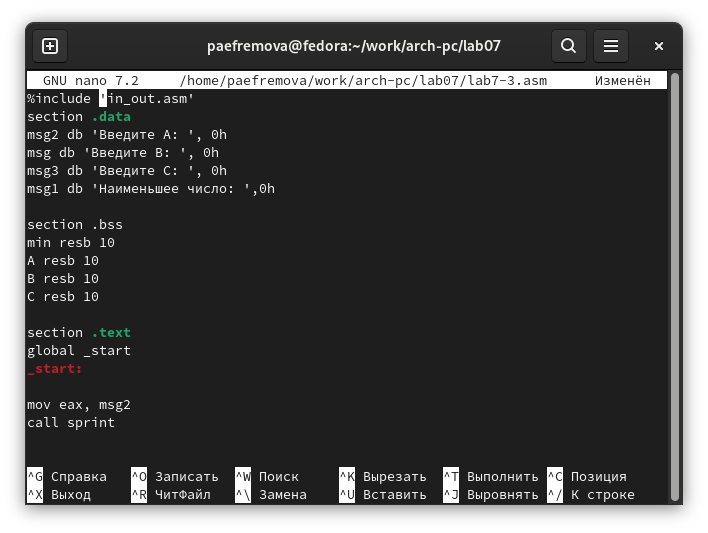


Рис. 17: Ввод программы

Запуск программы (рис. 18).

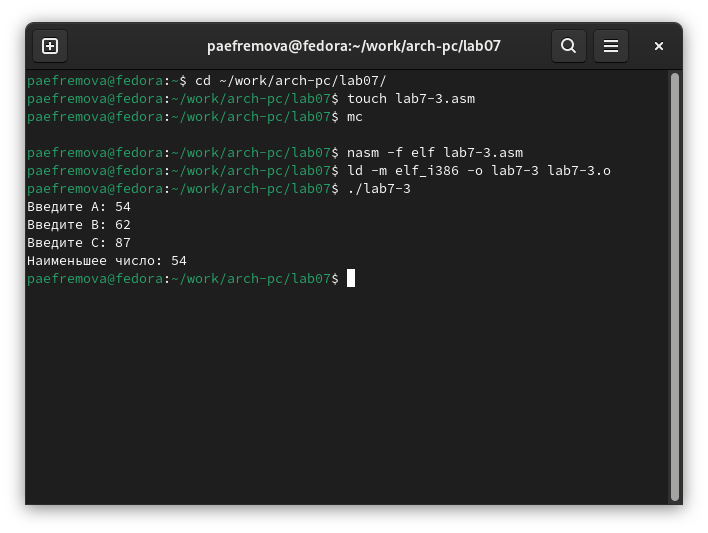


Рис. 18: Запуск программы

2.Создаю файл для второго самостоятельного задания и ввожу туда программу (рис. 19).

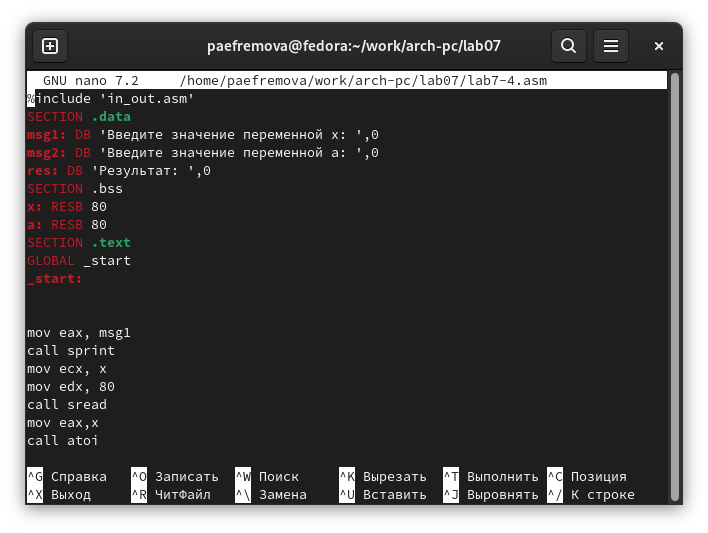


Рис. 19: Создание файла и ввод программы

Программа для задания 2:

%include 'in\_out.asm'  
SECTION .data  
msg1: DB 'Введите значение переменной x: ',0  
msg2: DB 'Введите значение переменной a: ',0  
res: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80  
a: 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 edi,eax ; edi = x   
  
mov eax, msg2  
call sprint  
mov ecx, a  
mov edx, 80  
call sread  
mov eax,a  
call atoi  
mov esi,eax ; esi = a  
  
  
cmp edi,esi  
jg var2 ; a < x |-> var2  
  
mov eax,x  
call atoi  
  
mov edi,15 ; edi = 15  
jmp fin  
  
var2:  
mov eax,x  
call atoi ; x  
sub eax, esi; x - a  
mov ebx, 2 ; ebx = 2  
mul eax ; (x-a)\*2  
mov edi, eax ; edi = aex  
  
  
fin:  
mov eax,res ; eax = res  
call sprint ; строка  
mov eax,edi ; eax = edi  
call iprintLF  
call quit

Запуск программы (рис. 20).

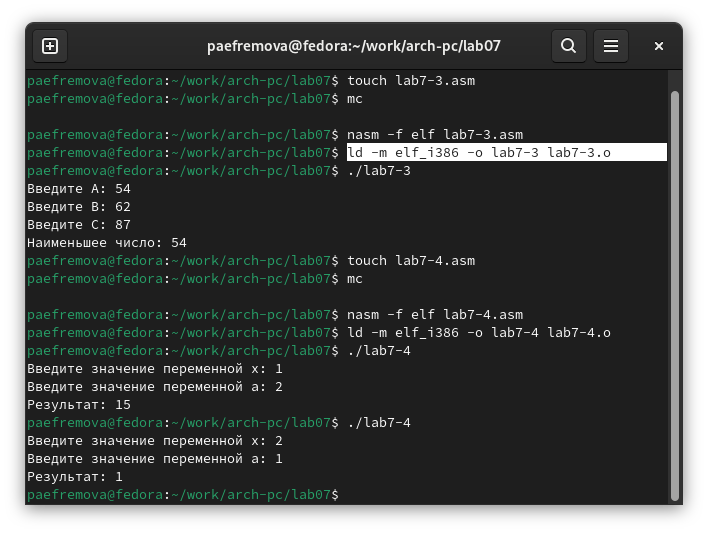


Рис. 20: Запуск программы

# 6 Выводы

При выполнении лабораторной работы я изучила команды условных и безусловных переходво, а также приобрела навыки написания программ с использованием перходов, познакомилась с назначением и структурой файлов листинга.

# Список литературы

1.[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/Лабораторная%20работа%20№7.%20Команды%20безусловного%20и%20условного%20переходов%20в%20Nasm.%20Программирование%20ветвлений..pdf)

2.[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf)