Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, а также знакомство с назначением и структурой файла листинга.

# 2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Задание для самостоятельной работы

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

* условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
* безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

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

jmp <адрес\_перехода>

Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра 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, отра- жающий текущее состояние процессора.

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:

cmp <операнд\_1>, <операнд\_2>

Единственным результатом команды сравнения является формирование флагов.

Команда условного перехода имеет вид

j<мнемоника перехода> label

Мнемоника перехода связана со значением анализируемых флагов или со способом фор- мирования этих флагов.

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

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Итак, структура листинга:

* номер строки — это номер строки файла листинга;
* адрес — это смещение машинного кода от начала текущего сегмента;
* машинный код представляет собой ассемблированную исходную строку в виде шестна- дцатеричной последовательности.;
* исходный текст программы — это просто строка исходной программы вместе с ком- ментариями.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

Создаю каталог lab07 для программ лабораторной работы №7, перехожу в него и со- здаю файл lab7-1.asm, проверяю, что файл создан (рис. [1](#fig:001))

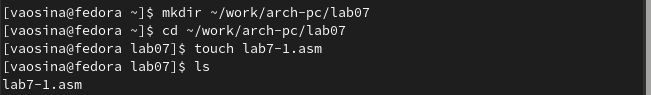


Figure 1: Создание каталога lab07 и файла lab7-1.asm в нем

Перед работой с программами копирую файл in\_out.asm в каталог и проверяю, что файл находится в нужном каталоге (рис. [2](#fig:002)) (рис. [3](#fig:003))

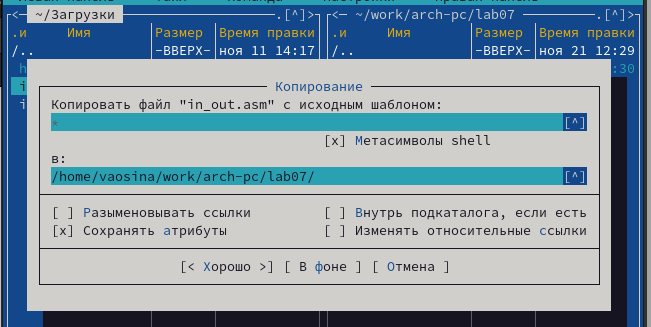


Figure 2: Копирование файла in\_out.asm



Figure 3: Проверка, что файл находится в нужном каталоге

Ввожу в файл lab7-1.asm текст программы (рис. [4](#fig:004)).

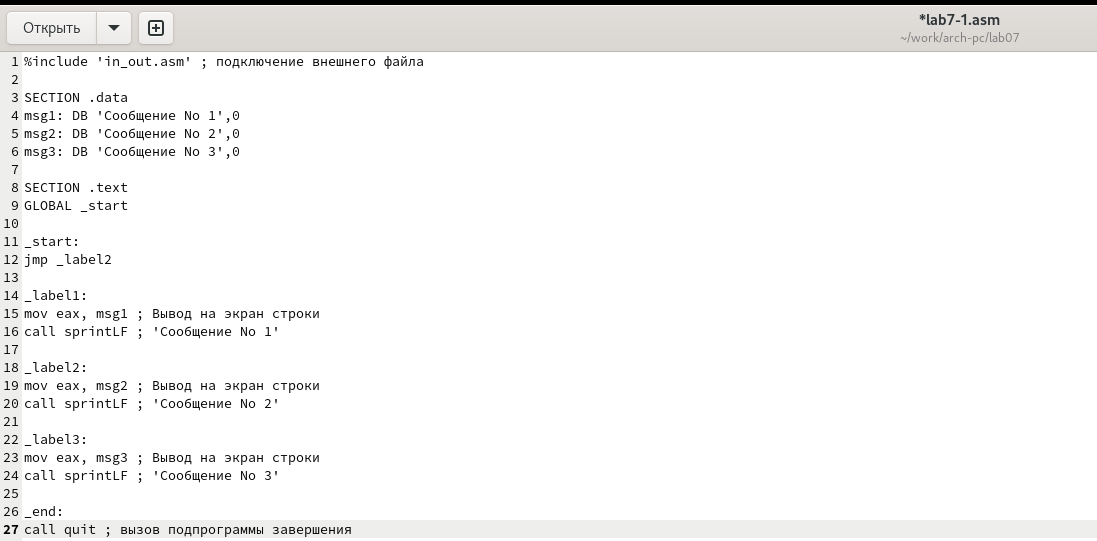


Figure 4: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его (рис. [5](#fig:005)).

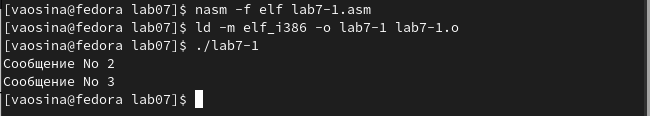


Figure 5: Создание исполняемого файла и его запуск

Изменяю текст программы таким образом, чтобы она выводила сначала ‘Сообщение No 2’, потом ‘Сообщение No 1’ и завершала работу. (рис. [6](#fig:006)).

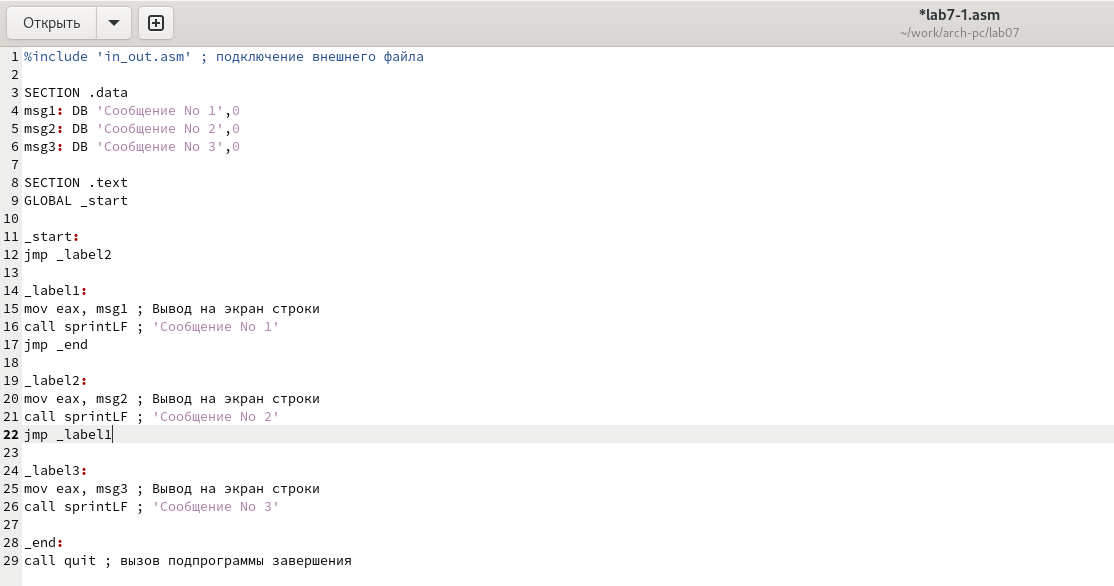


Figure 6: Изменение текста программы

Создаю исполняемый файл и запускаю его. (рис. [7](#fig:007)) Действительно, сначала выводится ‘Сообщение No 2’, потом ‘Сообщение No 1’

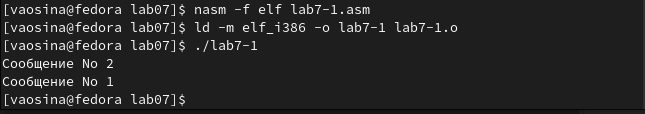


Figure 7: Создание исполняемого файла и его запуск

Теперь изменяю текст программы таким образом, чтобы она выводила сначала ‘Сообщение No 3’, потом ‘Сообщение No 2’, за ним ‘Сообщение No 1’ и завершала работу. (рис. [8](#fig:008)).

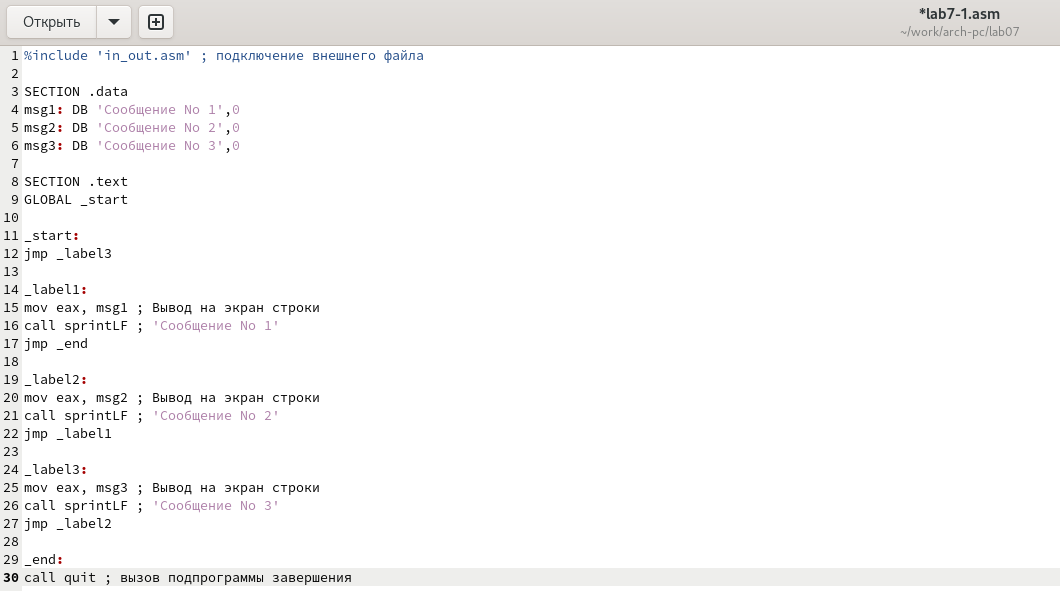


Figure 8: Изменение текста программы

Создаю исполняемый файл и запускаю его. (рис. [9](#fig:009)). Действительно, сначала выводится ‘Сообщение No 3’, потом ‘Сообщение No 2’, потом ‘Сообщение No 1’

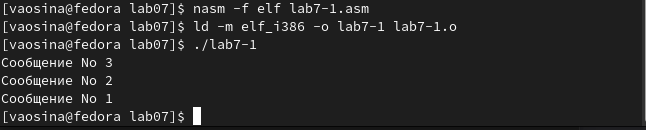


Figure 9: Создание исполняемого файла и его запуск

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. [10](#fig:010))

Figure 10: Создание файла lab7-2.asm

Figure 10: Создание файла lab7-2.asm

Ввожу в файл текст программы

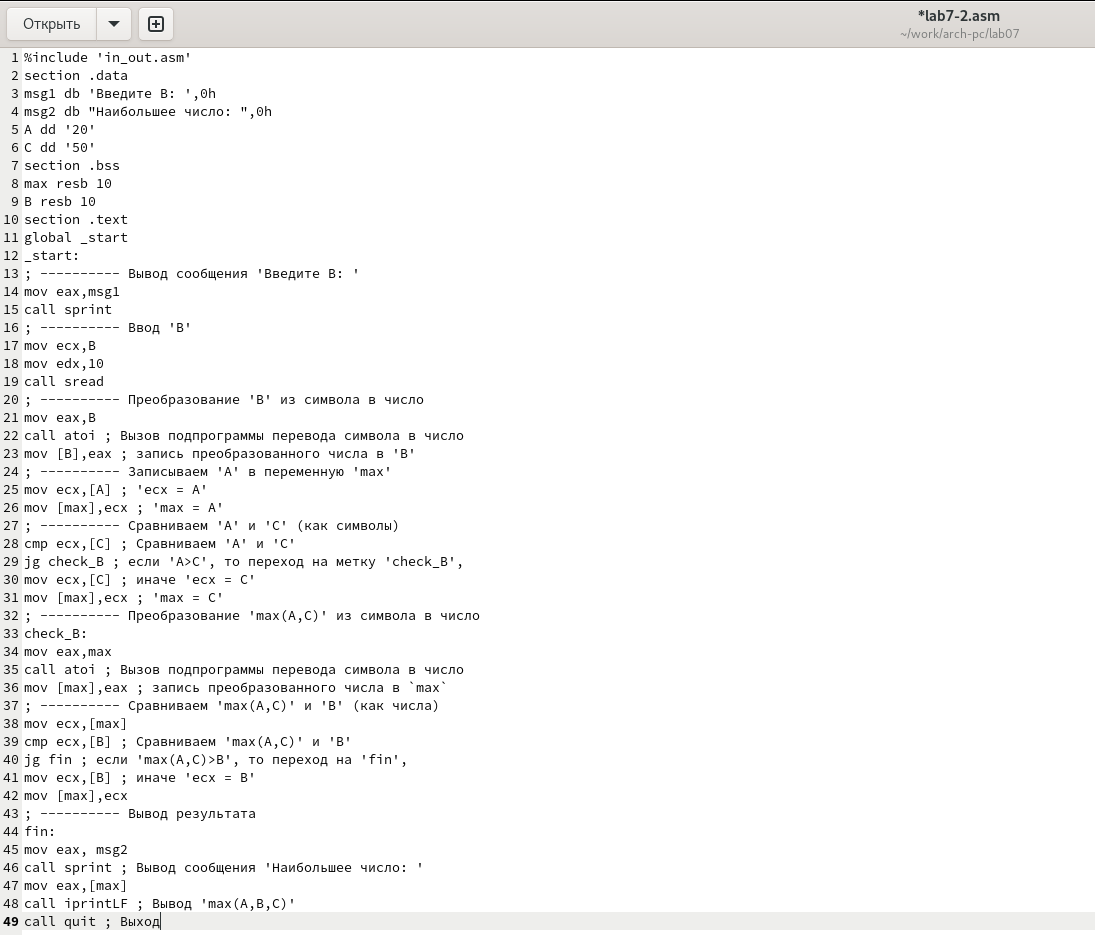


Figure 11: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его.(рис. [12](#fig:012)). Проверяю его работу для разных значений B.

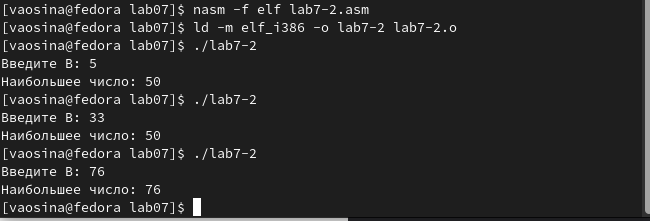


Figure 12: Создание исполняемого файла и его запуск

## 4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm.(рис. [13](#fig:013)).

Figure 13: Создание файла листинга

Figure 13: Создание файла листинга

Открываю файл листинга lab7-2.lst (рис. [14](#fig:014)).

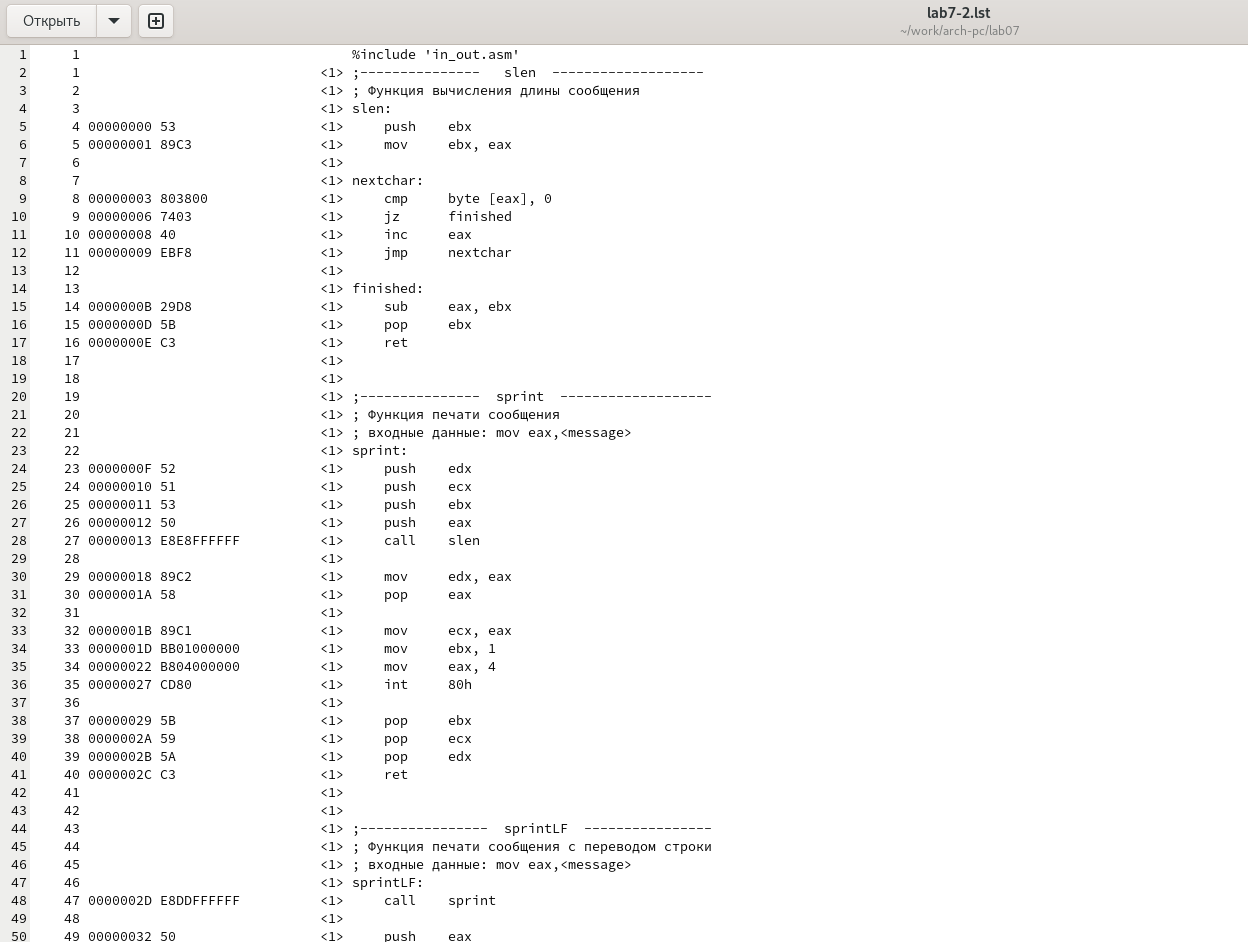


Figure 14: Открытие файла листинга

Привожу три строки из файла листинга для их описания (рис. [15](#fig:015))

Figure 15: Строки из файла листинга

Figure 15: Строки из файла листинга

1. 87 - номер строки; 00000068 - адрес строки; F7FE - машинный код; idiv esi - исходный текст программы, где idiv является инструкцией для знакового деления.
2. 88 - номер строки; 0000006A - адрес строки; 83C230 - машинный код; add edx, 48 - исходный текст программы, где add является инструкцией для сложения.
3. 89 - номер строки; 0000006D - адрес строки; 52 - машинный код; push edx - исходный текст программы, где push является инструкцией для помещения операнда в стек.

Открываю файл с программой lab7-2.asm и в инструкции mov с двумя операндами удаляю один операнд. (рис. [16](#fig:016)) и (рис. [17](#fig:017))

Figure 16: Инструкция mov с двумя операндами

Figure 16: Инструкция mov с двумя операндами

Figure 17: Инструкция mov с одним операндом

Figure 17: Инструкция mov с одним операндом

Выполняю трансляцию с получением файла листинга.(рис. [18](#fig:018)) Выдается ошибка из-за нарушения работы кода, т. к. инструкция mov может работать только при наличии двух операндов

Figure 18: Трансляция файла

Figure 18: Трансляция файла

В файле листинга это выглядит следующим образом (рис. [19](#fig:019))

Figure 19: Файл листинга

Figure 19: Файл листинга

## 4.3 Выполнение задания для самостоятельной работы

### 4.3.1 Задание 1

Необходимо написать программу нахождения наименьшей из 3 целочисленных переменных a, b и c В соответствии с вариантом (7), полученным при выполнении предыдущей лабораторной работы, значения переменных таковы: a = 45, b = 67, c = 15

Создаю файл var7-1.asm и открываю его (рис. [20](#fig:020)).

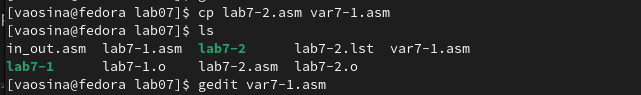


Figure 20: Создание файла var7-1.asm

Ввожу текст программы (рис. [21](#fig:021)) (рис. [22](#fig:022))

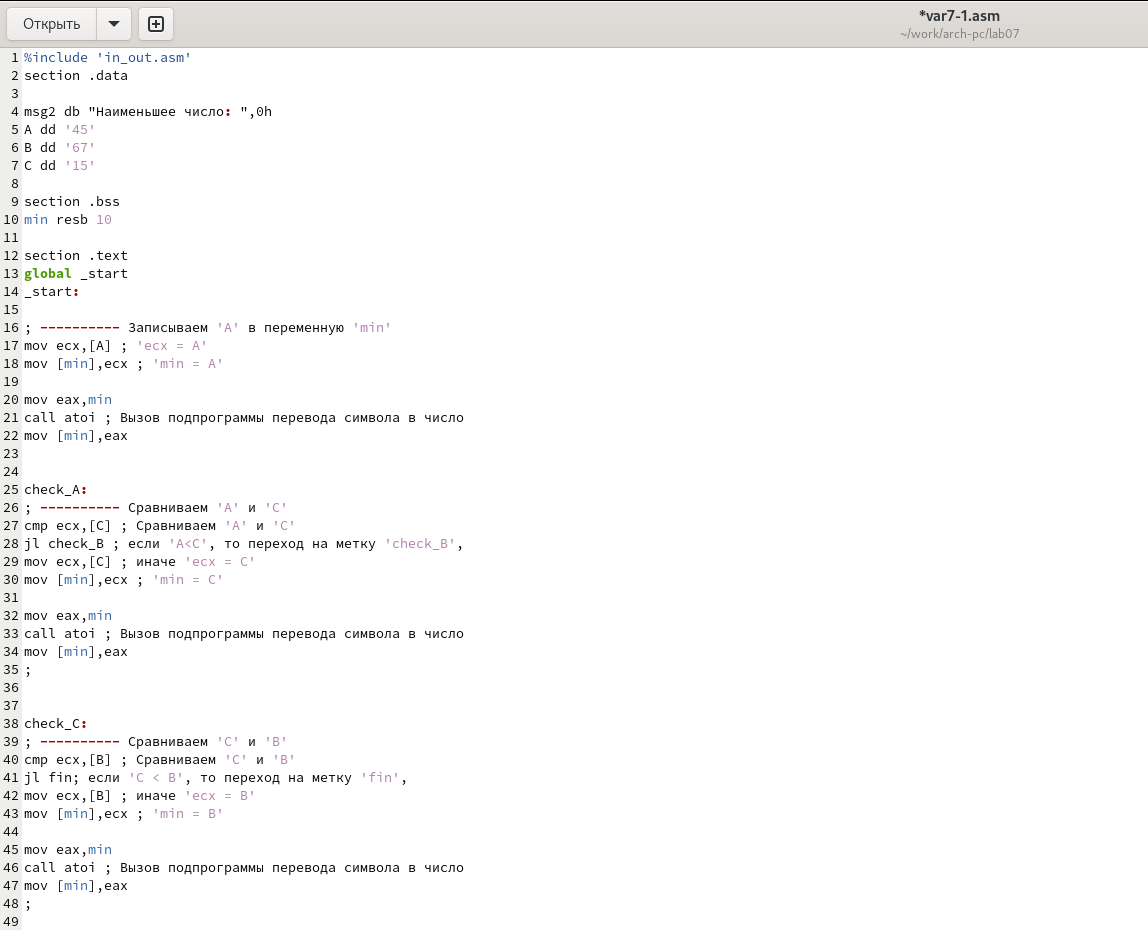


Figure 21: Ввод текста программы

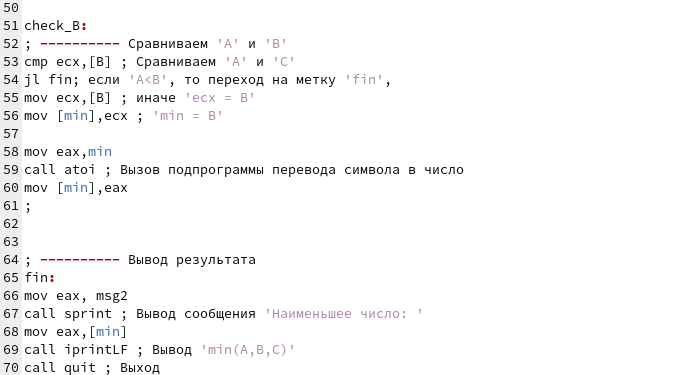


Figure 22: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. [23](#fig:023)). Выводится значение 15. Программа работает корректно.

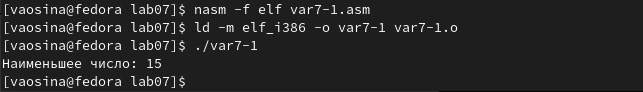


Figure 23: Создание исполняемого файла и его запуск

#### 4.3.1.1 текст программы

```%include ‘in\_out.asm’

section .data

msg2 db “Наименьшее число:”,0h

A dd ‘45’

B dd ‘67’

C dd ‘15’

section .bss

min resb 10

section .text

global \_start

\_start:

mov ecx,[A] ; ‘ecx = A’

mov [min],ecx ; ‘min = A’

mov eax,min

call atoi

mov [min],eax

check\_A:

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’

mov eax,min

call atoi

mov [min],eax ;

check\_C:

cmp ecx,[B]

jl fin; если ‘C < B’, то переход на метку ‘fin’,

mov ecx,[B]

mov [min],ecx

mov eax,min

call atoi

mov [min],eax ;

check\_B:

cmp ecx,[B]

jl fin; если ‘A < B’, то переход на метку ‘fin’,

mov ecx,[B]

mov [min],ecx

mov eax,min

call atoi

mov [min],eax ;

fin:

mov eax, msg2

call sprint ; Вывод сообщения ‘Наименьшее число:’

mov eax,[min]

call iprintLF ; Вывод ‘min(A,B,C)’

call quit ; Выход

### Задание 2  
  
Необходимо написать программу, которая для введенных с клавиатуры значений x и a вычисляет  
значение заданной функции f(x) и выводит результат вычислений.  
Функция: 6а, если х=а; а + х, если х != а  
  
Создаю файл var7-2.asm (рис. @fig:024).  
  
![Создание файла var7-2.asm](image/24.png){#fig:024 width=70%}  
  
Ввожу в файл текст программы (рис. @fig:025).  
  
![Ввод текста программы](image/25.png){#fig:025 width=70%}  
  
Создаю исполняемый файл и запускаю его. (рис. @fig:026)  
Проверяю работу программы для значений х = 1 и а = 1.  
Результат корректный (х = а => 6а = 6\*1 = 6).  
  
![Проверка работа программы для значений х = 1, а = 1] (image/26.png){#fig:026 width=70%}  
  
Теперь проверяю работу программы для значений х = 2 и а = 1. (рис. @fig:027).  
Результат корректный. (х != а => a + x = 2+1 = 3)  
  
![Проверка работа программы для значений х = 2, а = 1](image/27.png){#fig:027 width=70%}  
  
  
#### текст программы   
  
```%include 'in\_out.asm'  
  
section .data  
  
msg1 db 'Введите x: ',0h  
  
msg2 db 'Введите a: ',0h  
  
msg3 db 'Результат: ',0h  
  
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 [x], eax  
  
  
mov eax, msg2  
  
call sprint  
  
  
mov ecx,a  
  
mov edx,80  
  
call sread  
  
  
mov eax,a  
  
call atoi  
  
mov [a], eax  
  
  
cmp eax,[x]  
  
je \_fx  
  
add eax,[x]  
  
jmp \_fin  
  
  
\_fx:  
  
mov edx,6  
  
mul edx  
  
  
\_fin:  
  
mov ecx, eax  
  
mov eax, msg3  
  
call sprint ; Вывод сообщения 'Результат: '  
  
mov eax,ecx  
  
call iprintLF   
  
call quit

# 5 Выводы

При выполнении данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, а также зпознакомилась с назначением и структурой файла листинга.

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089087/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9..pdf)