

Лабораторная работа №7

Архитектура компьютера

Мурашов Иван Вячеславович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация переходов в NASM	7
3.2	Изучение структуры файла листинга	17
3.3	Выполнение заданий для самостоятельной работы	19
4	Выводы	27

Список иллюстраций

3.1	Создание каталога и файла в нём	7
3.2	Редактирование файла	8
3.3	Трансляция, компоновка и запуск файлов	9
3.4	Редактирование файла	10
3.5	Трансляция, компоновка и запуск файлов	11
3.6	Редактирование файла	12
3.7	Трансляция, компоновка и запуск файлов	13
3.8	Создание файла	13
3.9	Редактирование файла	14
3.10	Трансляция, компоновка и запуск файлов	16
3.11	Создание файла	17
3.12	Просмотр файла	17
3.13	Редактирование файла	18
3.14	Трансляция с получением файла листинга	19
3.15	Создание файла	19
3.16	Редактирование файла	20
3.17	Трансляция, компоновка и запуск файлов	22
3.18	Создание файла	22
3.19	Редактирование файла	23
3.20	Трансляция, компоновка и запуск файлов	24

Список таблиц

1 Цель работы

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

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm (рис. [3.1]).

```
[ivmurashov@fedora ~]$ mkdir ~/work/arch-pc/lab07  
[ivmurashov@fedora ~]$ cd ~/work/arch-pc/lab07  
[ivmurashov@fedora lab07]$ touch lab7-1.asm
```

Рис. 3.1: Создание каталога и файла в нём

Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. [3.2]).

```
lab7-1.asm (~/work/arch-pc/lab07)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[+] [📄] [📁] | [↶] [↷] | [✂] [📄] [📁] | [🔍] [🔧]
[📄] Л06_Мурашов_отчет.md x [📄] *Л07_Мурашов_отчет.md x [📄] lab7-1.asm x
%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 _label2
_label1:
    mov eax, msg1 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 1'
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 2'
_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintf ; 'Сообщение № 3'
_end:
    call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Редактирование файла

Листинг 1. Программа с использованием инструкции jmp

```
%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 _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
```

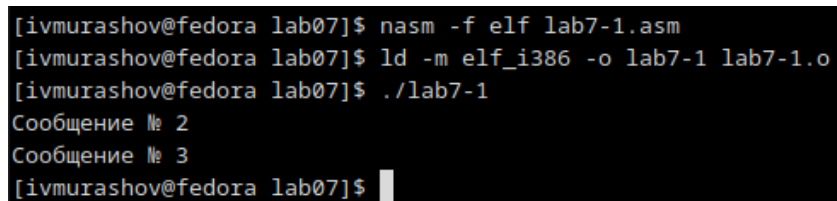


```

call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю его (рис. [3.3]).



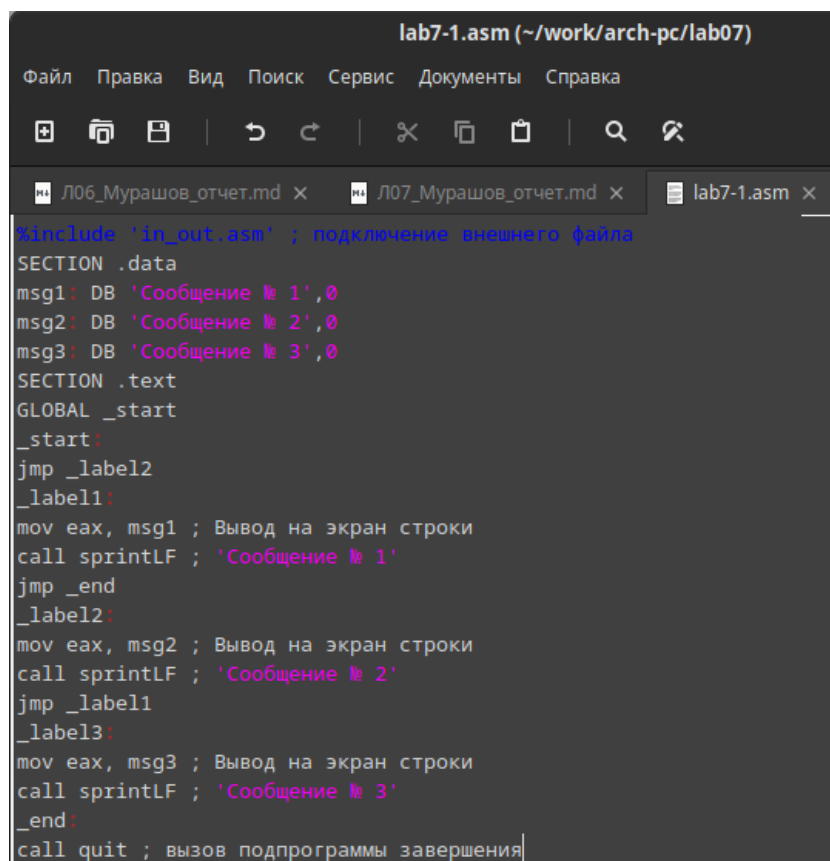
```

[ivmurashov@fedora lab07]$ nasm -f elf lab7-1.asm
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ivmurashov@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[ivmurashov@fedora lab07]$

```

Рис. 3.3: Трансляция, компоновка и запуск файлов

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения №2 добавляю инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения №1) и после вывода сообщения №1 добавляю инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Изменяю текст программы в соответствии с листингом 7.2 (рис. [3.4]).



```
lab7-1.asm (~/work/arch-pc/lab07)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Иконки]
Л06_Мурашов_отчет.md x  Л07_Мурашов_отчет.md x  lab7-1.asm x
%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 _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Редактирование файла

Листинг 2. Программа с использованием инструкции jmp

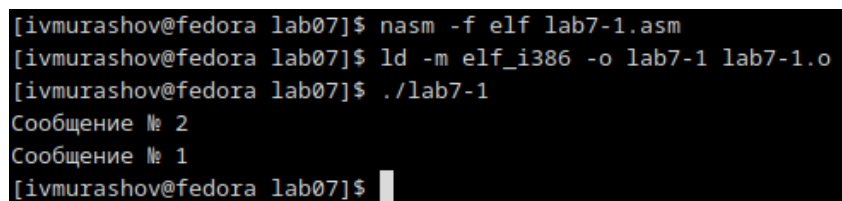
```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
```

```

mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю его (рис. [3.5]).



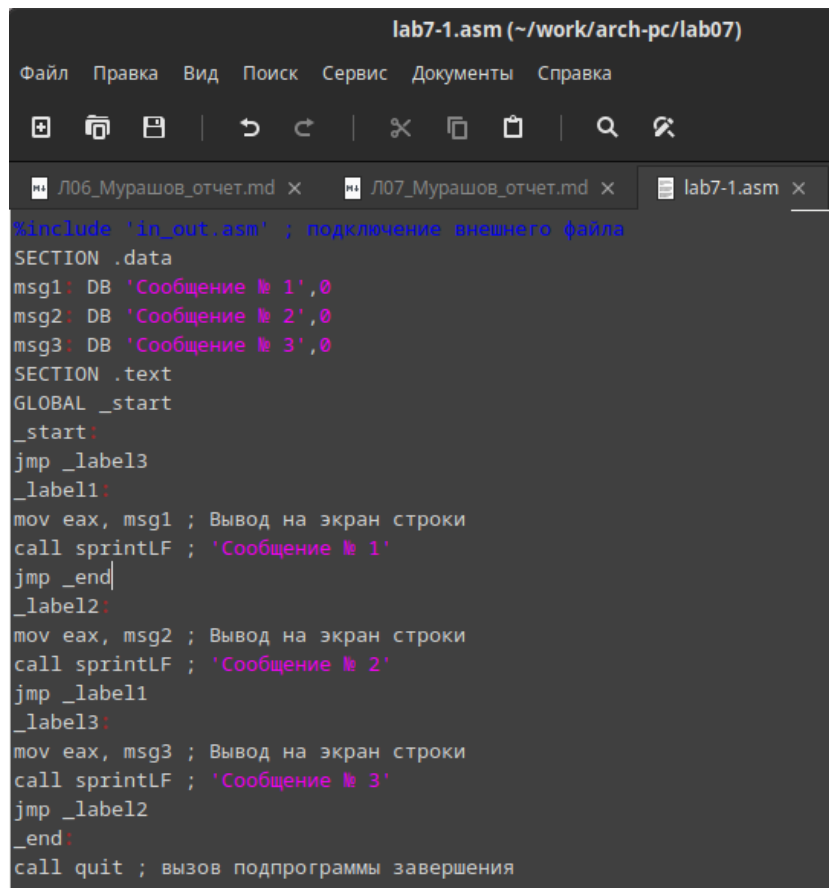
```

[ivmurashov@fedora lab07]$ nasm -f elf lab7-1.asm
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ivmurashov@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[ivmurashov@fedora lab07]$

```

Рис. 3.5: Трансляция, компоновка и запуск файлов

Изменяю текст программы так, чтобы программа сначала выводила ‘Сообщение № 3’, затем ‘Сообщение № 2’, а затем ‘Сообщение № 1’ (рис. [3.6]).



```
lab7-1.asm (~/work/arch-pc/lab07)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Icons]
Л06_Мурашов_отчет.md x  Л07_Мурашов_отчет.md x  lab7-1.asm x
%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 sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.6: Редактирование файла

Листинг 3. Программа с использованием инструкции jmp

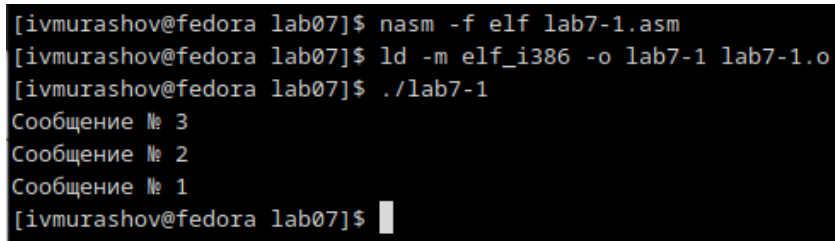
```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
```

```

mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю его (рис. [3.7]).



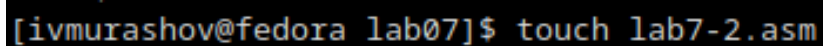
```

[ivmurashov@fedora lab07]$ nasm -f elf lab7-1.asm
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[ivmurashov@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[ivmurashov@fedora lab07]$

```

Рис. 3.7: Трансляция, компоновка и запуск файлов

Создаю файл lab7-2.asm (рис. [3.8]).



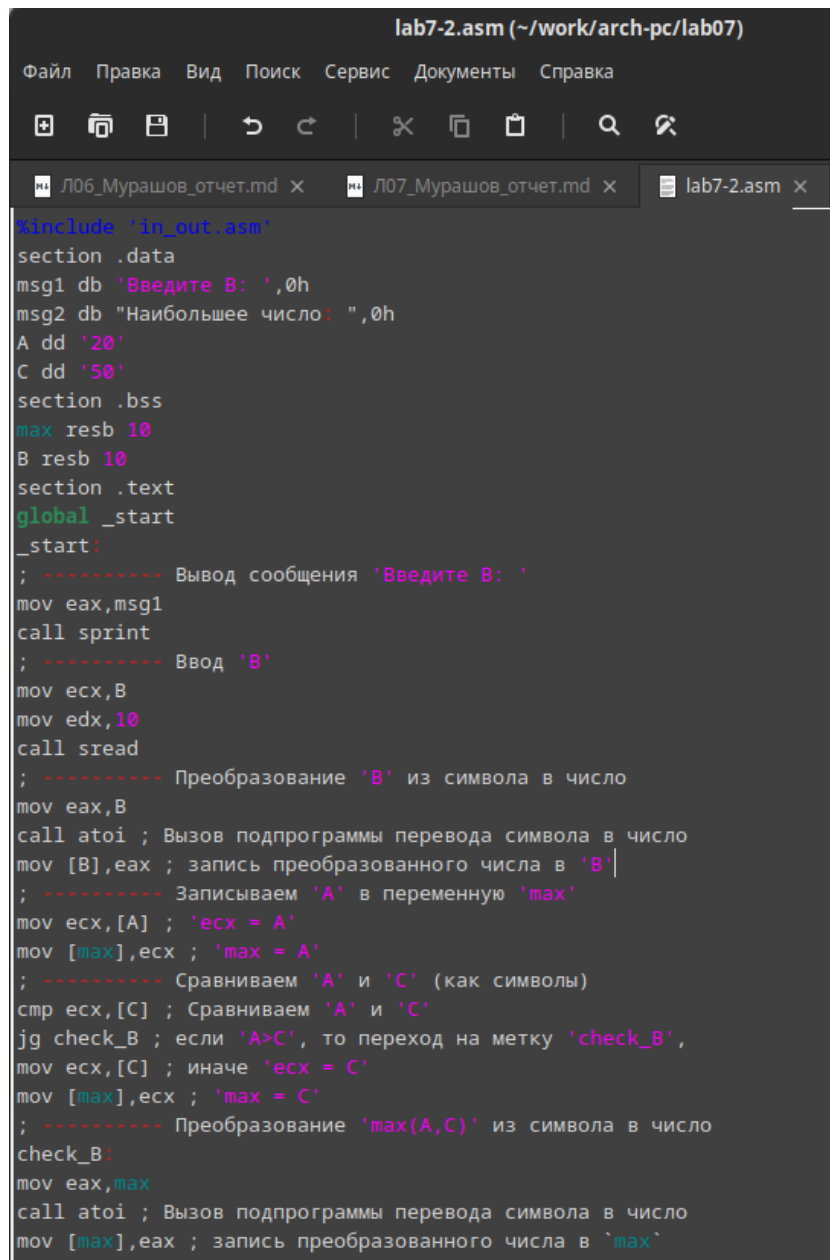
```

[ivmurashov@fedora lab07]$ touch lab7-2.asm

```

Рис. 3.8: Создание файла

Изучаю текст программы из листинга 7.3 и ввожу в lab7-2.asm (рис. [3.9]).



```
lab7-2.asm (~/work/arch-pc/lab07)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Иконки]
Л06_Мурашов_отчет.md x  Л07_Мурашов_отчет.md x  lab7-2.asm x
%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 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
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'
```

Рис. 3.9: Редактирование файла

Листинг 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:
; ----- 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 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
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

```

Создаю исполняемый файл и запускаю его и проверяю его работу для разных значений B.(рис. [3.10]).

```

[ivmurashov@fedora lab07]$ nasm -f elf lab7-2.asm
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[ivmurashov@fedora lab07]$ ./lab7-2
Введите B: 2
Наибольшее число: 50
[ivmurashov@fedora lab07]$ ./lab7-2
Введите B: 20
Наибольшее число: 50
[ivmurashov@fedora lab07]$ ./lab7-2
Введите B: 120
Наибольшее число: 120

```

Рис. 3.10: Трансляция, компоновка и запуск файлов

3.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm о, указав ключ -l и задав имя файла листинга в командной строке (рис. [3.11]).

```
[ivmurashov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 3.11: Создание файла

Открываю файл листинга lab7-2.lst (рис. [3.12]).

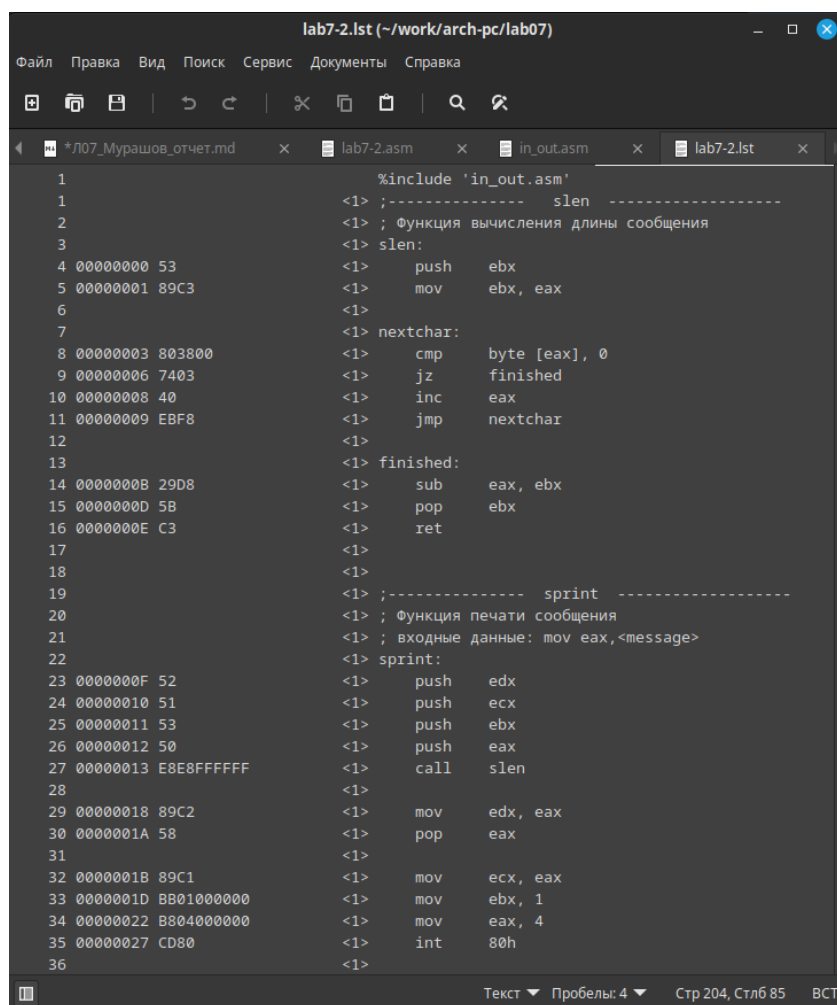


Рис. 3.12: Просмотр файла

Объясняю содержимое первой выбранной строки:

5 00000035 32300000

A dd '20'

5 - номер строки; 00000035 - смещение машинного кода от начала текущего сегмента; инструкция A dd '20' ассемблируется в 32300000 (в шестнадцатеричном представлении); 32300000 - инструкция на машинном языке, определяющая переменную A размером в 4 байта); A dd '20' - исходный текст программы.

Объясняю содержимое второй выбранной строки:

21 00000101 B8[0A000000]

mov eax, B

21 - номер строки; 00000101 - смещение машинного кода от начала текущего сегмента; инструкция mov eax,B ассемблируется в B8[0A000000] (в шестнадцатеричном представлении); B8[0A000000] - инструкция на машинном языке, записывающая значение переменной B в регистр eax); mov eax,B - исходный текст программы.

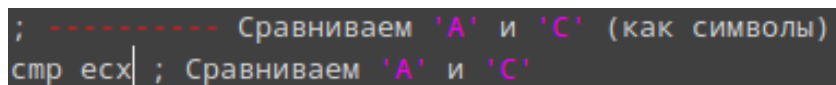
Объясняю содержимое третьей выбранной строки:

29 00000122 7F0C

jg check_B

29 - номер строки; 00000122 - смещение машинного кода от начала текущего сегмента; инструкция jg check_B ассемблируется в 7F0C (в шестнадцатеричном представлении); 7F0C - инструкция на машинном языке, осуществляющая переход на метку 'check_B', если A>C); jg check_B - исходный текст программы.

Открываю файл с программой lab7-2.asm и в разделе сравнения 'A' и 'C' как символов удаляю операнд [C] (рис. [3.13]).



```
; ----- Сравниваем 'A' и 'C' (как символы)
cmp eax, [C]
```

Рис. 3.13: Редактирование файла

Выполняю трансляцию с получением файла листинга (рис. [3.14]).

```
[ivmurashov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm  
lab7-2.asm:28: error: invalid combination of opcode and operands
```

Рис. 3.14: Трансляция с получением файла листинга

На выходе я не получаю никаких файлов, так как инструкция `cmp` подразумевает сравнение двух операндов.

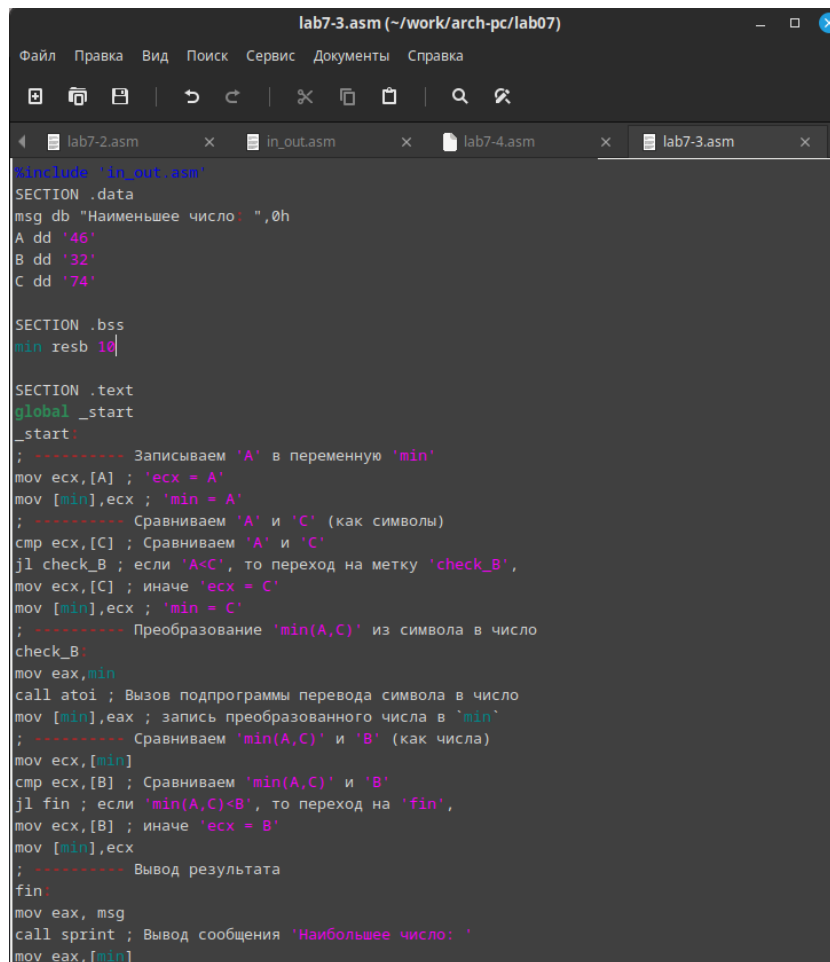
3.3 Выполнение заданий для самостоятельной работы

1. Создаю файл `lab7-3.asm` в каталоге `lab07` (рис. [3.15]).

```
[ivmurashov@fedora lab07]$ touch lab7-3.asm
```

Рис. 3.15: Создание файла

Открываю файл и пишу программу нахождения наименьшей из 3 целочисленных переменных `a`, `b` и `c`. В соответствии с таблицей 7.5 присваиваю переменным значения, указанные в 19 варианте (рис. [3.16]).



```
lab7-3.asm (~/work/arch-pc/lab07)
Файл Правка Вид Поиск Сервис Документы Справка
lab7-2.asm x in_out.asm x lab7-4.asm x lab7-3.asm x

%include 'in_out.asm'
SECTION .data
msg db "Наименьшее число: ",0h
A dd '46'
B dd '32'
C dd '74'

SECTION .bss
min resb 10

SECTION .text
global _start
_start:
; ----- Записываем 'A' в переменную 'min'
mov ecx,A ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,C ; Сравниваем 'A' и 'C'
jnl check_B ; если 'A<C', то переход на метку 'check_B',
mov ecx,C ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,B ; Сравниваем 'min(A,C)' и 'B'
jnl fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,B ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax,msg
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[min]
```

Рис. 3.16: Редактирование файла

Листинг 4. Программа нахождения наименьшей из 3 целочисленных переменных a, b и c

```
%include 'in_out.asm'

SECTION .data
msg db "Наименьшее число: ",0h
A dd '46'
B dd '32'
C dd '74'

SECTION .bss
```

```
min resb 10
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
; ----- Записываем 'A' в переменную 'min'
```

```
mov ecx,[A] ; 'ecx = A'
```

```
mov [min],ecx ; 'min = A'
```

```
; ----- Сравниваем 'A' и 'C' (как символы)
```

```
cmp ecx,[C] ; Сравниваем 'A' и 'C'
```

```
jl check_B ; если 'A<C', то переход на метку 'check_B',
```

```
mov ecx,[C] ; иначе 'ecx = C'
```

```
mov [min],ecx ; 'min = C'
```

```
; ----- Преобразование 'min(A,C)' из символа в число
```

```
check_B:
```

```
mov eax,min
```

```
call atoi ; Вызов подпрограммы перевода символа в число
```

```
mov [min],eax ; запись преобразованного числа в 'min'
```

```
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
```

```
mov ecx,[min]
```

```
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
```

```
jl fin ; если 'min(A,C)<B', то переход на 'fin',
```

```
mov ecx,[B] ; иначе 'ecx = B'
```

```
mov [min],ecx
```

```
; ----- Вывод результата
```

```
fin:
```

```
mov eax, msg
```

```
call sprint ; Вывод сообщения 'Наибольшее число: '
```

```
mov eax,[min]
```

```
call iprintLF ; Вывод 'min(A,B,C)'  
call quit ; Выход
```

Создаю исполняемый файл и запускаю его (рис. [3.17]).

```
[ivmurashov@fedora lab07]$ nasm -f elf lab7-3.asm  
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o  
[ivmurashov@fedora lab07]$ ./lab7-3  
Наименьшее число: 74  
[ivmurashov@fedora lab07]$
```

Рис. 3.17: Трансляция, компоновка и запуск файлов

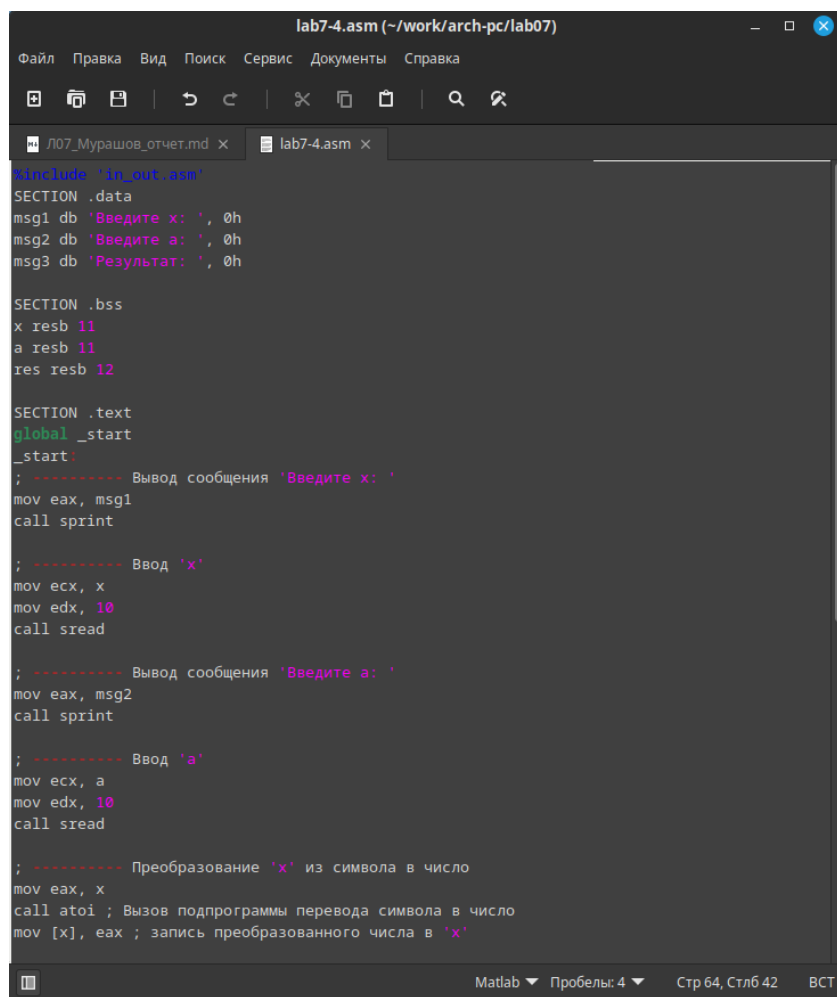
Программа работает корректно. 74 - действительно, наибольшее из данных чисел.

2. Создаю файл lab7-4.asm в каталоге lab07 (рис. [3.18]).

```
[ivmurashov@fedora lab07]$ touch lab7-4.asm
```

Рис. 3.18: Создание файла

Открываю файл и пишу программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. В соответствии с таблицей 7.6 (19 вариант), $f(x) = \{a + x, x > a; x, x \leq a$ (рис. [3.19]).



```
lab7-4.asm (~/.work/arch-pc/lab07)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите x: ', 0h
msg2 db 'Введите a: ', 0h
msg3 db 'Результат: ', 0h

SECTION .bss
x resb 11
a resb 11
res resb 12

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax, msg1
call sprint

; ----- Ввод 'x'
mov ecx, x
mov edx, 10
call sread

; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
call sprint

; ----- Ввод 'a'
mov ecx, a
mov edx, 10
call sread

; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x], eax ; запись преобразованного числа в 'x'
```

Рис. 3.19: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю его работу для значений x и a из таблицы 7.6 (рис. [3.20]).

```

[ivmurashov@fedora lab07]$ nasm -f elf lab7-4.asm
[ivmurashov@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[ivmurashov@fedora lab07]$ ./lab7-4
Введите x: 4
Введите a: 5
Результат: 4
[ivmurashov@fedora lab07]$ ./lab7-4
Введите x: 3
Введите a: 2
Результат: 5
[ivmurashov@fedora lab07]$

```

Рис. 3.20: Трансляция, компоновка и запуск файлов

Листинг 5. Программа для вычисления значения заданной функции $f(x)$

```

#include 'in_out.asm'

SECTION .data
msg1 db 'Введите x: ', 0h
msg2 db 'Введите a: ', 0h
msg3 db 'Результат: ', 0h

SECTION .bss
x resb 11
a resb 11
res resb 12

SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите x: '
mov eax, msg1
call sprint

; ----- Ввод 'x'

```



```

mov ecx, x
mov edx, 10
call sread

; ----- Вывод сообщения 'Введите a: '
mov eax, msg2
call sprint

; ----- Ввод 'a'
mov ecx, a
mov edx, 10
call sread

; ----- Преобразование 'x' из символа в число
mov eax, x
call atoi ; Вызов подпрограммы перевода символа в число
mov [x], eax ; запись преобразованного числа в 'x'

; ----- Преобразование 'a' из символа в число
mov eax, a
call atoi ; Вызов подпрограммы перевода символа в число
mov [a], eax ; запись преобразованного числа в 'a'

; ----- Сравниваем 'x' и 'a' (как числа)
mov eax, [a]
mov ecx, [x]
cmp eax, ecx ; Сравниваем 'a' и 'x'
jl add_xa ; если 'a<x', то переход на метку 'add_xa'
mov eax, ecx ; иначе 'eax = x'

```

```

mov [res], eax ; 'res = x'
jmp _res

; ----- Записываем 'a+x' в переменную 'res'
add_xa:
add eax, ecx ; 'eax = eax + ecx = a + x'
mov [res], eax ; 'res = a + x'
;jmp _res

; ----- Вывод результата
_res:
mov eax, msg3
call sprint ; Вывод сообщения 'Результат: '
mov eax, [res]
call iprintLF ; Вывод
call quit ; Вызов подпрограммы завершения

```

4 Выводы

В ходе выполнения данной лабораторной работы я изучил команды условного и безусловного переходов, приобрёл навыки написания программ с использованием переходов и познакомился с назначением и структурой файла листинга.