

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Хамзина Виктория Валентиновна

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Реализация переходов в NASM	5
2.2	Изучение структуры файла листинга	10
3	Задание для самостоятельной работы	12
4	Выводы	19

Список иллюстраций

2.1	Создание каталога и файла	5
2.2	Редактирование файла	6
2.3	Запуск исполняемого файла	6
2.4	Редактирование файла	7
2.5	Запуск исполняемого файла	7
2.6	Редактирование файла	8
2.7	Запуск исполняемого файла	8
2.8	Создание файла	8
2.9	Редактирование файла	9
2.10	Запуск исполняемого файла	9
2.11	Создание файла листинга и его открытие	10
2.12	Файл листинга	10
2.13	Редактирование файла	11
2.14	Ошибка в файле листинга	11
3.1	Создание файла	12
3.2	Редактирование файла	13
3.3	Запуск исполняемого файла	15
3.4	Создание файла	15
3.5	Редактирование файла	16
3.6	Запуск исполняемого файла	18

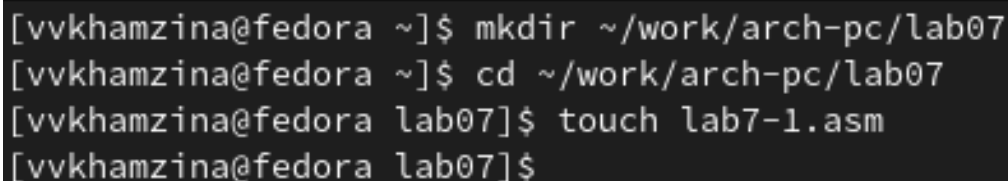
1 Цель работы

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с их использованием. Познакомиться с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

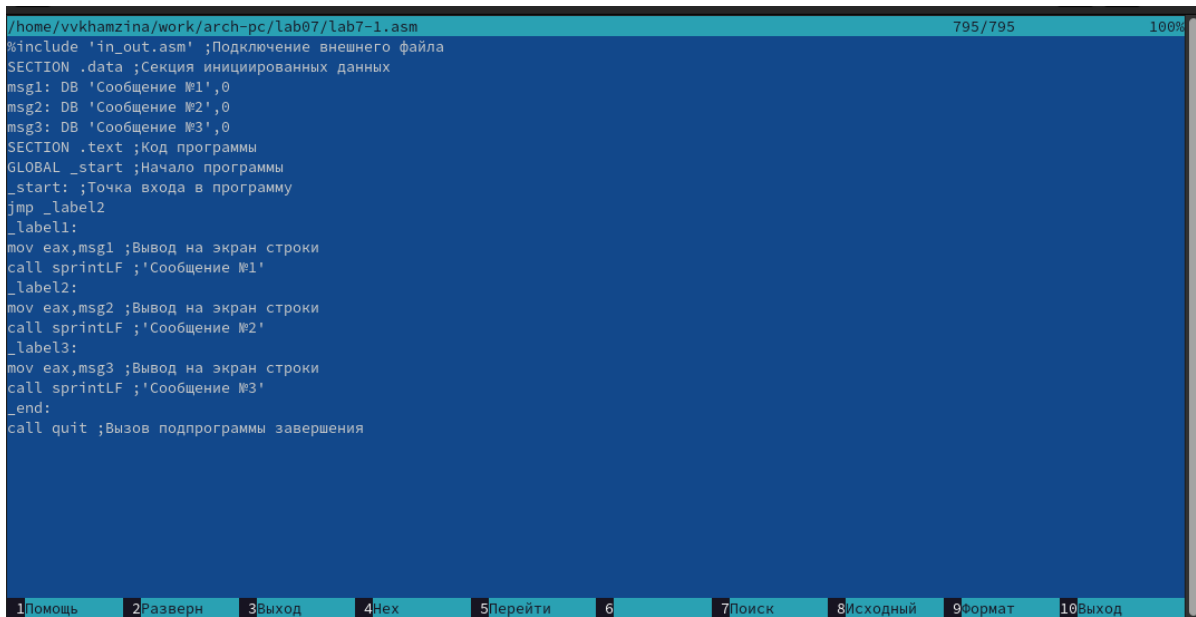
Создала каталог для данной лабораторной работы с помощью команды `mkdir ~/work/arch-pc/lab07`, перешла в него и создала файл `lab7-1.asm` (рис. 2.1).



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

Рис. 2.1: Создание каталога и файла

Ввела в файл текст программы, в которой используется инструкция `jmp` (рис. 2.2).

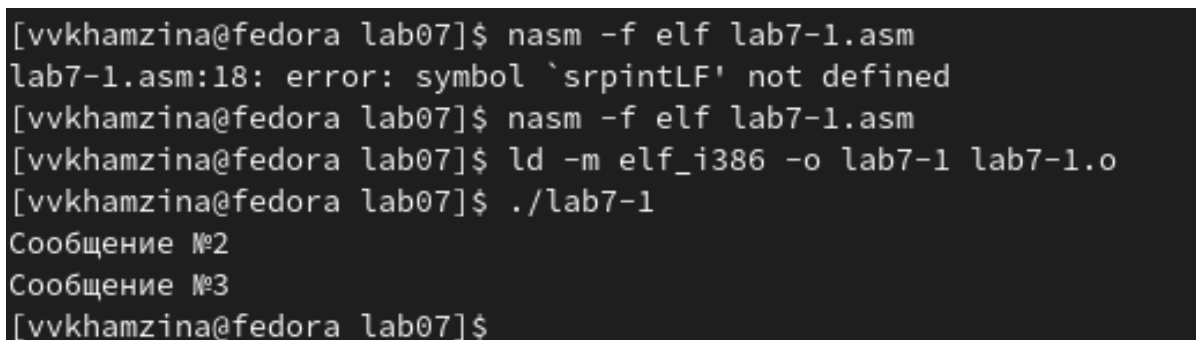


```
/home/vvkhamzina/work/arch-pc/lab07/lab7-1.asm 795/795 100%
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициализированных данных
msg1: DB 'Сообщение №1',0
msg2: DB 'Сообщение №2',0
msg3: DB 'Сообщение №3',0
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
jmp _label2
_label1:
mov eax,msg1 ;Вывод на экран строки
call sprintfLF ;'Сообщение №1'
_label2:
mov eax,msg2 ;Вывод на экран строки
call sprintfLF ;'Сообщение №2'
_label3:
mov eax,msg3 ;Вывод на экран строки
call sprintfLF ;'Сообщение №3'
_end:
call quit ;Вызов подпрограммы завершения

1Помощь 2Разверн 3Выход 4Hex 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рис. 2.2: Редактирование файла

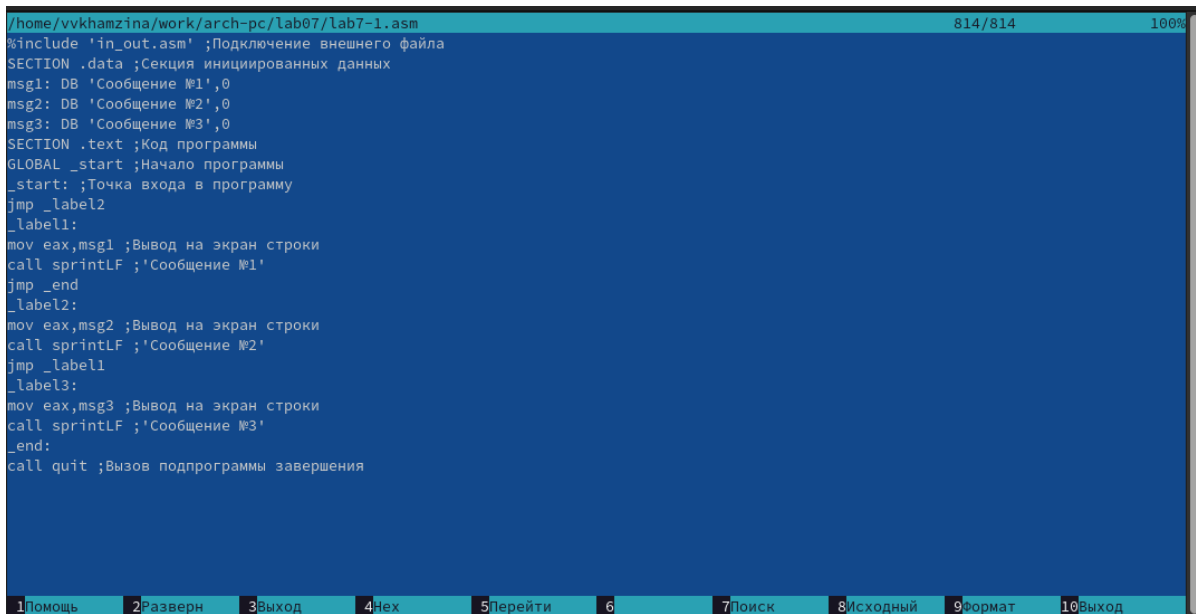
Создала исполняемый файл lab7-1 и проверила его работу (рис. 2.3).



```
[vvkhamzina@fedora lab07]$ nasm -f elf lab7-1.asm
lab7-1.asm:18: error: symbol `srpintLF' not defined
[vvkhamzina@fedora lab07]$ nasm -f elf lab7-1.asm
[vvkhamzina@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[vvkhamzina@fedora lab07]$ ./lab7-1
Сообщение №2
Сообщение №3
[vvkhamzina@fedora lab07]$
```

Рис. 2.3: Запуск исполняемого файла

Изменила текст программы таким образом, чтобы она сначала выводила ‘Сообщение №2’, а потом ‘Сообщение №1’ и завершала работу (рис. 2.4).

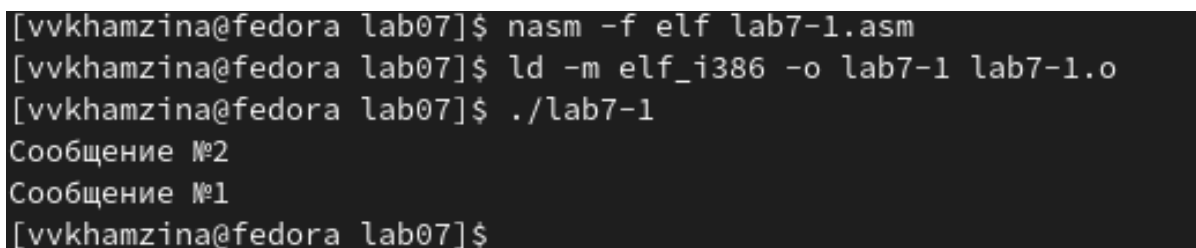


```
/home/vvkhamzina/work/arch-pc/lab07/lab7-1.asm 814/814 100%
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция иницированных данных
msg1: DB 'Сообщение №1',0
msg2: DB 'Сообщение №2',0
msg3: DB 'Сообщение №3',0
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
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
```

1Помощь 2Разверн 3Выход 4Hex 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

Рис. 2.4: Редактирование файла

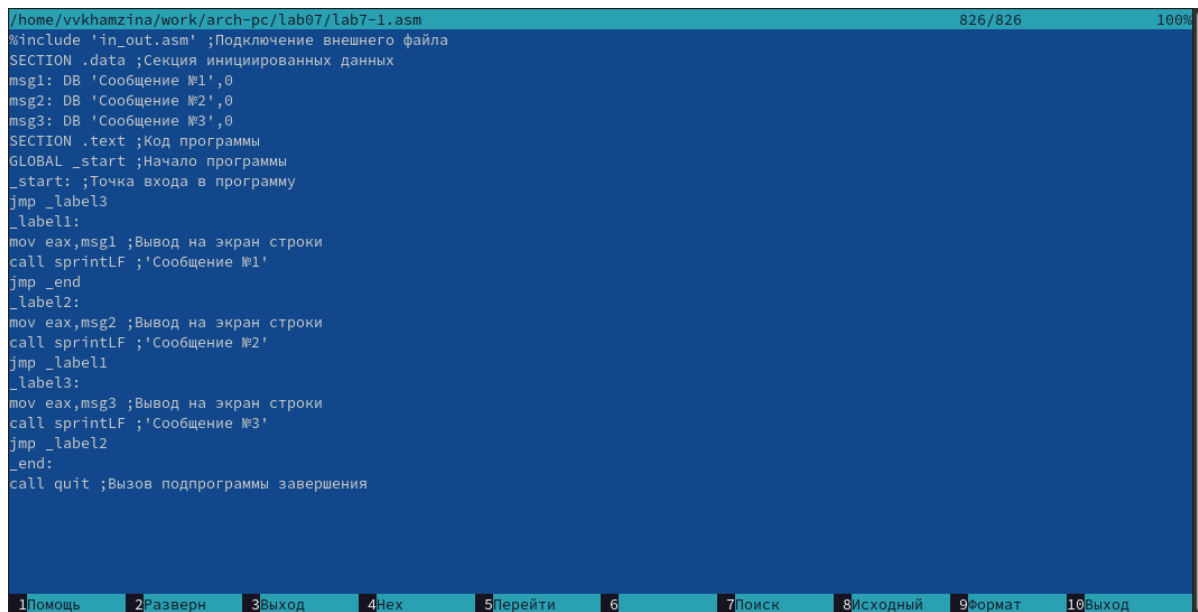
Создала исполняемый файл после редактирования текста программы и проверила его работу (рис. 2.5).



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

Рис. 2.5: Запуск исполняемого файла

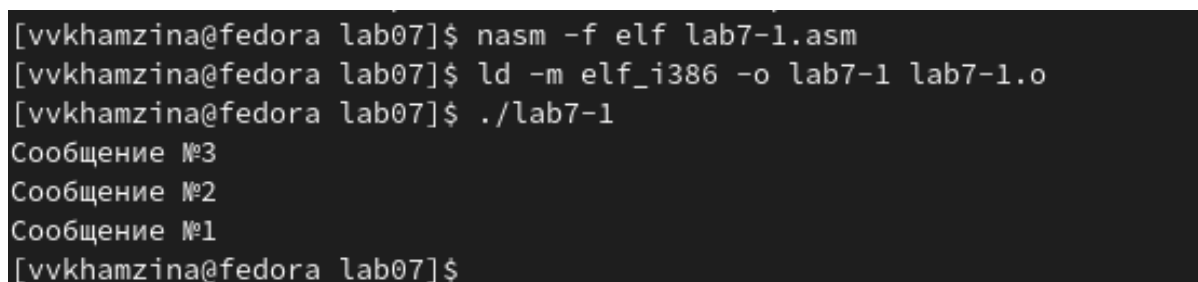
Изменила текст программы в файле lab7-1.asm так, чтобы она выводила сообщения в обратном порядке (рис. 2.6).



```
/home/vvkhamzina/work/arch-pc/lab07/lab7-1.asm 826/826 100%
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициированных данных
msg1: DB 'Сообщение №1',0
msg2: DB 'Сообщение №2',0
msg3: DB 'Сообщение №3',0
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
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. 2.6: Редактирование файла

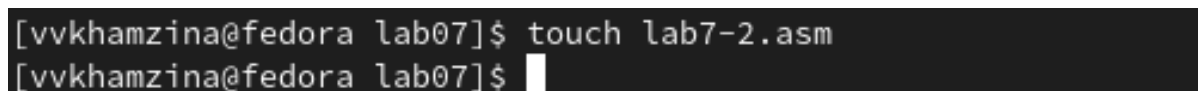
Создала исполняемый файл после изменения текста программы и запустила его (рис. 2.7).



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

Рис. 2.7: Запуск исполняемого файла

Создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 2.8).

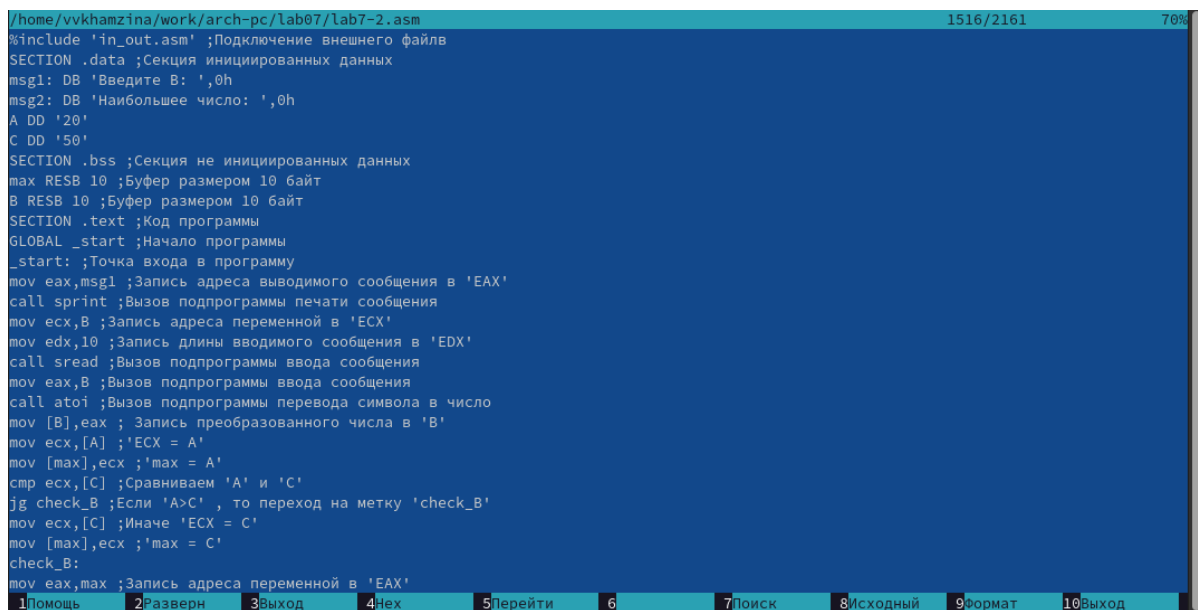


```
[vvkhamzina@fedora lab07]$ touch lab7-2.asm
[vvkhamzina@fedora lab07]$
```

Рис. 2.8: Создание файла

Ввела в файл lab7-2.asm текст программы, которая определяет и выводит на

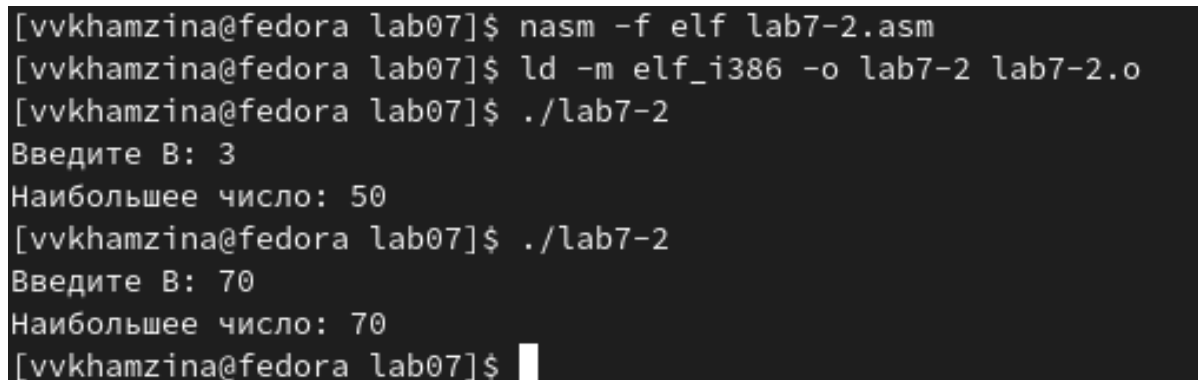
экран наибольшую из 3 целочисленных переменных: A, B и C (рис. 2.9).



```
//home/vvkhazina/work/arch-pc/lab07/lab7-2.asm
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициализированных данных
msg1: DB 'Введите B: ',0h
msg2: DB 'Наибольшее число: ',0h
A DD '20'
C DD '50'
SECTION .bss ;Секция не инициализированных данных
max RESB 10 ;Буфер размером 10 байт
B RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
mov eax,msg1 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
mov ecx,B ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
call sread ;Вызов подпрограммы ввода сообщения
mov eax,B ;Вызов подпрограммы ввода сообщения
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 'C'
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 ;Запись адреса переменной в 'EAX'
```

Рис. 2.9: Редактирование файла

Создала исполняемый файл lab7-2 и проверила его работу для разных значений B (рис. 2.10).



```
[vvkhamzina@fedora lab07]$ nasm -f elf lab7-2.asm
[vvkhamzina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[vvkhamzina@fedora lab07]$ ./lab7-2
Введите B: 3
Наибольшее число: 50
[vvkhamzina@fedora lab07]$ ./lab7-2
Введите B: 70
Наибольшее число: 70
[vvkhamzina@fedora lab07]$
```

Рис. 2.10: Запуск исполняемого файла

2.2 Изучение структуры файла листинга

Создала файл листинга для программы из файла lab7-2.asm с помощью команды `nasm -f elf -l lab7-2.lst lab7-2.asm` и открыла его с помощью текстового редактора `mcedit` (рис. 2.11).

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

Рис. 2.11: Создание файла листинга и его открытие

Рассмотрим строки 11, 14 и 23 и объясним их содержимое. (рис. 2.12).

В строке 11 содержится номер строки [11], адрес [00000009], машинный код [EBF8] и содержимое строки кода [`jmp nextchar`]. В строке 14 содержится ее номер, адрес [0000000B], машинный код [29D8], содержимое - [`sub eax,ebx`]. В 23 строке адрес [0000000F], машинный код [52] и содержимое [`push edx`].

```
lab7-2.lst [----] 0 L: [ 1+ 0 1/217] *(0 /14554b) 0032 0x020 [*] [X]
1      %include 'in_out.asm' ;Подключение внешнего файла
2      <1> ;----- slen -----
3      <1> ; Функция вычисления длины сообщения
4      <1> slen:-----
5      <1>      push     ebx
6      <1>      mov      ebx, eax
7      <1>      nextchar:-----
8      <1>      cmp      byte [eax], 0
9      <1>      jz       finished
10     <1>      inc      eax
11     <1>      jmp      nextchar
12     <1>      finished:-----
13     <1>      sub      eax, ebx
14     <1>      pop      ebx
15     <1>      ret
16     <1>      sprint:-----
17     <1>      ; Функция печати сообщения
18     <1>      ; входные данные: mov eax,<message>
19     <1>      push     edx
20     <1>      push     ecx
21     <1>      push     ebx
22     <1>      push     eax
23     <1>      call    slen
24     <1>      pop      eax
25     <1>      pop      ebx
26     <1>      pop      ecx
27     <1>      pop      edx
```

Рис. 2.12: Файл листинга

Удалила один из операндов инструкции `mov` (рис. 2.13).

```

SECTION .bss ;Секция не инициированных данн
max RESB 10 ;Буфер размером 10 байт
B RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
mov eax, ;Запись адреса выводимого сообщени
call sprint ;Вызов подпрограммы печати сооб
mov ecx,B ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщени
call sread ;Вызов подпрограммы ввода сообще
mov eax,B ;Вызов подпрограммы ввода сообщен
call atoi ;Вызов подпрограммы перевода симе
mov [B],eax ; Запись преобразованного числа
mov ecx,[A] ; 'ECX = A'
mov [max],ecx ; 'max = A'
cmp ecx,[C] ;Сравниваем 'A' и 'C'
jg check B ;Если 'A>C' - то переход на мет

```

Рис. 2.13: Редактирование файла

Выполнила трансляцию с получением файла листинга командой `nasm -f elf -l lab7-2.lst lab7-2.asm`. Описание намеренно созданной ошибки появилось в файле листинга (рис. 2.14).

```

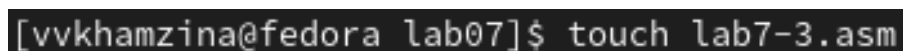
10          SECTION .text ;Код программы
11          GLOBAL _start ;Начало программы
12          _start: ;Точка входа в программу
13          mov eax, ;Запись адреса выводимого сообщения в 'EAX'
14          ***** error: invalid combination of opcode and operands
15          call sprint ;Вызов подпрограммы печати сообщения
16          mov ecx,B ;Запись адреса переменной в 'ECX'
17          mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
18          call sread ;Вызов подпрограммы ввода сообщения
19          mov eax,B ;Вызов подпрограммы ввода сообщения
20          call atoi ;Вызов подпрограммы перевода символа в число
          mov [B],eax ; Запись преобразованного числа в 'B'

```

Рис. 2.14: Ошибка в файле листинга

3 Задание для самостоятельной работы

Создала файл lab7-3.asm (рис. 3.1).

A terminal window with a dark background. The prompt is [vvkhamzina@fedora lab07]\$ and the command touch lab7-3.asm is entered.

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

Рис. 3.1: Создание файла

Написала в созданном файле текст программы для нахождения наименьшей из 3 целочисленных переменных А, В и С. Взяла значения переменных варианта 18, так как этот вариант мне достался в ходе выполнения предыдущей лабораторной работы (рис. 3.2).

```
/home/vvkhazmina/work/arch-pc/lab07/lab7-3.asm 1744/2246 77%
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициированных данных
msg1: DB 'Значения A, B и C: 83, 73, 30',0h
msg2: DB 'Наименьшее число: ',0h
A DD '83'
B DD '73'
C DD '30'
SECTION .bss ;Секция не инициированных данных
min RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
; ----- Вывод сообщения 'Значения A, B и C: 83, 73, 30'
mov eax,msg1 ;Запись адреса выводимого сообщения в 'EAX'
call sprintf ;Вызов подпрограммы печати сообщения
; ----- 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'
; ----- Сравниваем 'min(A,C)' и 'B'(как числа)
check_B:
mov ecx,[min] ;'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 ;'min = B'
; ----- Вывод результата
fin:
; ----- Вывод сообщения 'Наименьшее число: '
mov eax,msg2 ;Запись адреса выводимого сообщения в 'EAX'
call sprintf ;Вызов подпрограммы печати сообщения
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рис. 3.2: Редактирование файла

```
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициированных данных
msg1: DB 'Значения A, B и C: 83, 73, 30',0h
msg2: DB 'Наименьшее число: ',0h
A DD '83'
B DD '73'
C DD '30'
SECTION .bss ;Секция не инициированных данных
min RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
```

```

; ----- Вывод сообщения 'Значения A, B и C: 83, 73, 30'
mov eax,msg1 ;Запись адреса выводимого сообщения в 'EAX'
call sprintLF ;Вызов подпрограммы печати сообщения
; ----- 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'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
check_B:
mov ecx,[min] ;'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 ;'min = B'
; ----- Вывод результата
fin:
; ----- Вывод сообщения 'Наименьшее число: '
mov eax,msg2 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
; ----- Преобразование 'min' из символа в число
mov eax,min ;Запись адреса переменной в 'EAX'
call atoi ;Вызов подпрограммы перевода символа в число
mov [min],eax ;Запись преобразованного числа в 'min'
mov eax,[min] ;Запись адреса переменной в 'EAX'
call iprintLF ;Вывод 'min(A,B,C)'

```

`call quit ;Вызов подпрограммы завершения`

Создала исполняемый файл и проверила его работу (рис. 3.3).

```
[vvkhamzina@fedora lab07]$ nasm -f elf lab7-3.asm
[vvkhamzina@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[vvkhamzina@fedora lab07]$ ./lab7-3
Значения A, B и C: 83, 73, 30
Наименьшее число: 30
```

Рис. 3.3: Запуск исполняемого файла

Создала файл lab7-4.asm (рис. 3.4).

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

Рис. 3.4: Создание файла

Написала программу вычисления значения заданной функции для введенных с клавиатуры `x` и `a` (рис. 3.5).

```
/home/vvkhazina/work/arch-pc/lab07/lab7-4.asm 2607/3223 80
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициированных данных
msg1: DB 'Введите значение x: ',0h
msg2: DB 'Введите значение a: ',0h
result: DB 'Результат: ',0h
SECTION .bss ;Секция не инициированных данных
x RESB 10 ;Буфер размером 10 байт
a RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
_start: ;Точка входа в программу
; ----- Вывод сообщения 'Введите значение x: '
mov eax,msg1 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
; ----- Ввод 'x'
mov ecx,x ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
call sread ;Вызов подпрограммы ввода сообщения
; ----- Вывод сообщения 'Введите значение a: '
mov eax,msg2 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
; ----- Ввод 'a'
mov ecx,a ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
call sread ;Вызов подпрограммы ввода сообщения
; ----- Преобразование 'x' из символа в число
mov eax,x ;Запись введенного сообщения в 'EAX'
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 ;Запись введенного сообщения в 'EAX'
call atoi ;Вызов подпрограммы преобразования символа в число
mov [a],eax ;Запись преобразованного числа в 'a'
; ----- Сравниваем 'A' с 1
mov ecx,[a] ;'ECX = a'
cmp ecx,1 ;Сравниваем 'a' и 1
je another_function ;Если 'a=1', то переход на метку 'another_function'
mov ecx,[a] ;Иначе 'ecx = a'
mul ecx ;'EAX = EAX * EAX'
mov edi,eax ;Запись результата вычисления в 'edi'
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рис. 3.5: Редактирование файла

```
%include 'in_out.asm' ;Подключение внешнего файла
SECTION .data ;Секция инициированных данных
msg1: DB 'Введите значение x: ',0h
msg2: DB 'Введите значение a: ',0h
result: DB 'Результат: ',0h
SECTION .bss ;Секция не инициированных данных
x RESB 10 ;Буфер размером 10 байт
a RESB 10 ;Буфер размером 10 байт
SECTION .text ;Код программы
GLOBAL _start ;Начало программы
```



```

_start: ;Точка входа в программу
; ----- Вывод сообщения 'Введите значение x: '
mov eax,msg1 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
; ----- Ввод 'x'
mov ecx,x ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
call sread ;Вызов подпрограммы ввода сообщения
; ----- Вывод сообщения 'Введите значение x: '
mov eax,msg2 ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
; ----- Ввод 'a'
mov ecx,a ;Запись адреса переменной в 'ECX'
mov edx,10 ;Запись длины вводимого сообщения в 'EDX'
call sread ;Вызов подпрограммы ввода сообщения
; ----- Преобразование 'x' из символа в число
mov eax,x ;Запись адреса переменной в 'EAX'
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 ;Запись адреса переменной в 'EAX'
call atoi ;Вызов подпрограммы перевода символа в число
mov [a],eax ;Запись преобразованного числа в 'a'
; ----- Сравниваем 'A' с 1
mov ecx, [a] ;'ECX = a'
cmp ecx,1 ;Сравниваем 'a' и 1
je another_function ;Если 'a=1', то переход на метку 'another_function'
mov ecx,[a] ;Иначе 'ECX = a'
mul ecx ; 'EAX = ECX * ECX'

```

```

mov edi,eax ;Запись результата вычисления в 'edi'
another_function:
mov eax,[x] ;'EAX = x'
add eax,10 ;'EAX = EAX + 10'
mov edi,eax ;Запись результата в 'edi'
; ----- Вывод результата
fin:
mov eax,result ;Запись адреса выводимого сообщения в 'EAX'
call sprint ;Вызов подпрограммы печати сообщения
mov eax,edi ;Запись результата вычислений в 'EAX'
call iprintLF ;Вывод результата вычислений
call quit ;Вызов подпрограммы завершения

```

Создала исполняемый файл и запустила его для разных значений x и a (рис. 3.6).

```

[vvkhamzina@fedora lab07]$ nasm -f elf lab7-4.asm
[vvkhamzina@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[vvkhamzina@fedora lab07]$ ./lab7-4
Введите значение x: 2
Введите значение a: 1
Результат: 12
[vvkhamzina@fedora lab07]$ ./lab7-4
Введите значение x: 1
Введите значение a: 2
Результат: 4
[vvkhamzina@fedora lab07]$

```

Рис. 3.6: Запуск исполняемого файла

4 Выводы

В ходе данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с их использованием, а также познакомилась с назначением и структурой файла листинга.