

Лабораторная работа №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Солдатов Алексей

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры листинга	16
4.3	Выполнение заданий для самостоятельной работы	19
4.3.1	Задание 1	19
4.3.2	Задание 2	22
5	Выводы	25
	Список литературы	26

Список иллюстраций

4.1	Подготовка к работе	8
4.2	Ввод команд	9
4.3	Перенос файла	10
4.4	Создание и запуск файла	10
4.5	Изменение текста программы	11
4.6	Создание файла	12
4.7	Изменение текста программы	12
4.8	Создание и проверка работы файла	13
4.9	Создание файла	14
4.10	Ввод команд	14
4.11	Создание файла	16
4.12	Создание листинга файла	16
4.13	Ознакомление	17
4.14	Анализ первой строки	17
4.15	Анализ второй строки	18
4.16	Анализ третьей строки	18
4.17	Попытка транслирования файла листинга	18
4.18	Ошибка в листинге	18
4.19	Создание файла	19
4.20	Ввод текста программы	19
4.21	Проверка работы программы	22
4.22	Создание файла	22
4.23	Ввод текста программы	22
4.24	Проверка работы программы	24

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

1 Цель работы

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

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры листинга
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

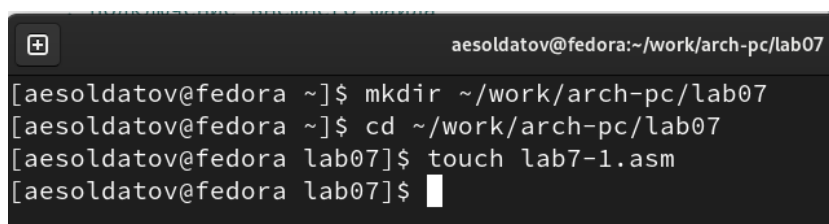
Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

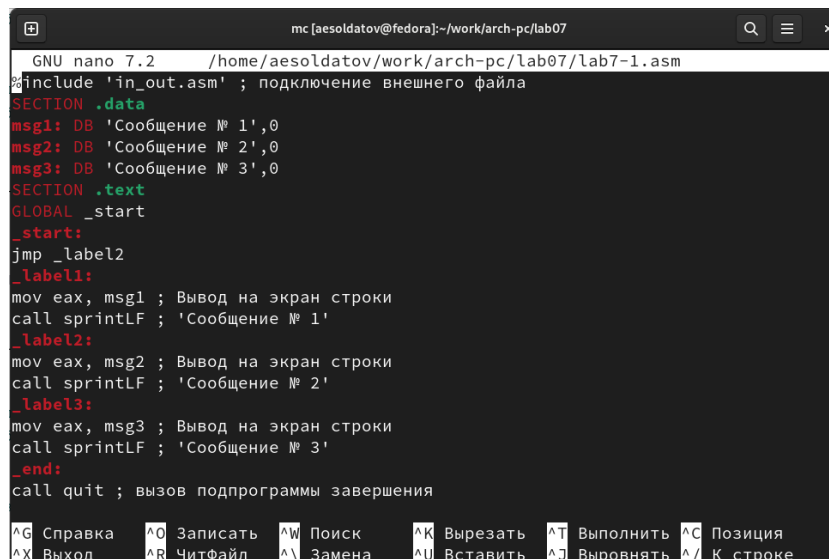
Создал каталог для программ лабораторной работы №7, перешел в него и создал файл “lab7-1.asm” (рис. 4.1).



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

Рис. 4.1: Подготовка к работе

Ввел в файл “lab7-1.asm” текст программы из листинга 7.1. со страницы в ТУИС (рис. 4.2).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/lab7-1.asm
#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

^G Справка      ^O Записать     ^W Поиск       ^K Вырезать    ^T Выполнить   ^C Позиция
^X Выход        ^R ЧитФайл     ^\ Замена      ^U Вставить    ^J Выводить    ^_ К строке
```

Рис. 4.2: Ввод команд

```
%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 sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Перенес файл “in_out.asm” из прошлой папки с лабораторной работой в нынешнюю (рис. 4.3).

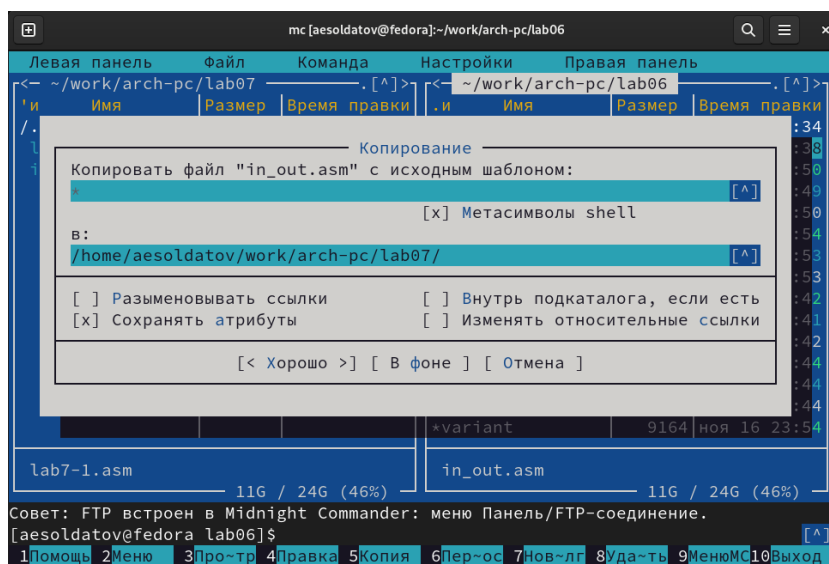


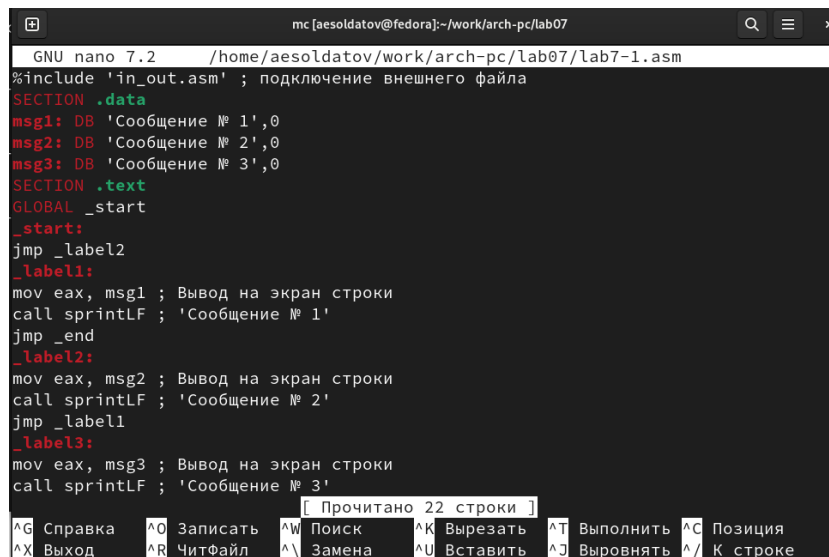
Рис. 4.3: Перенос файла

Создал исполняемый файл и запустил его. Он совпадает с примером (рис. 4.4).

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

Рис. 4.4: Создание и запуск файла

Далее изменил текст программы в соответствии с листингом 7.2 (рис. 4.5).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/lab7-1.asm
#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'
[ Прочитано 22 строки ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.5: Изменение текста программы

```
%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

```

Создал исполняемый файл и запустил его, программа работает правильно (рис. 4.6).

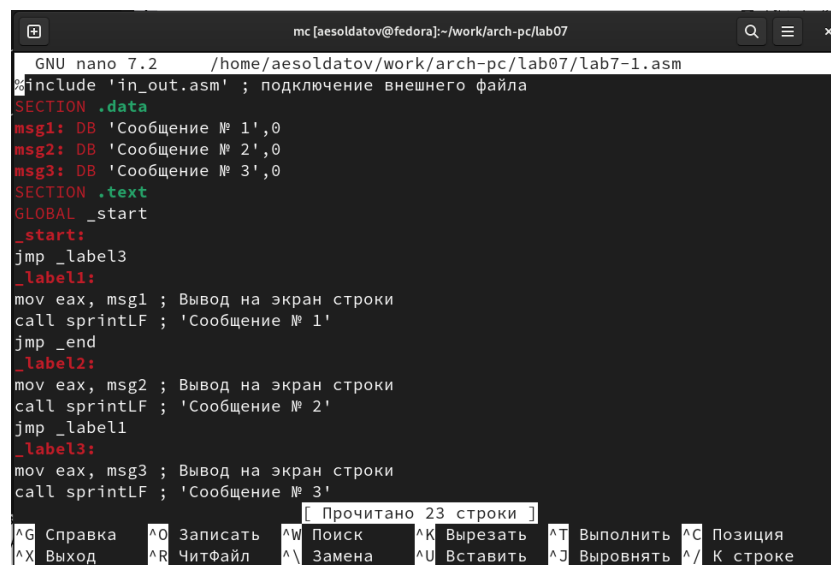
```

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

```

Рис. 4.6: Создание файла

Изменил текст программы, чтобы вывод программы соответствовал примеру (рис. 4.7).



```

GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/lab7-1.asm
%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'

```

[Прочитано 23 строки]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 4.7: Изменение текста программы

```

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data

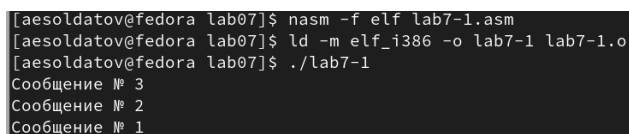
```

```

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

```

Создал исполняемый файл и запустил его, вывод соответствует примеру (рис. 4.8).



```

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

```

Рис. 4.8: Создание и проверка работы файла

Создал файл “lab7-2.asm” в каталоге “~/work/arch-pc/lab07” (рис. 4.9).

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

Рис. 4.9: Создание файла

Внимательно изучил текст программы из листинга 7.3 и ввел его в файл (рис. 4.10).

```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/lab7-2.asm
%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

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.10: Ввод команд

```
%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

```

Создал исполняемый файл и проверил его работу для разных значений (рис. 4.11).

```

[aesoldatov@fedora lab07]$ nasm -f elf lab7-2.asm
[aesoldatov@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[aesoldatov@fedora lab07]$ ./lab7-2
Введите B: 35
Наибольшее число: 50
[aesoldatov@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[aesoldatov@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 50
[aesoldatov@fedora lab07]$

```

Рис. 4.11: Создание файла

4.2 Изучение структуры листинга

Создал файл листинга для файла “lab7-2.asm” с помощью ключа “-l” и открыл его (рис. 4.12).

```

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

```

Рис. 4.12: Создание листинга файла

Внимательно изучил его формат и содержимое (рис. 4.13).


```

lab7-2.lst  [----]  0 L: 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*][X]
1                                     %include 'in_out.asm'
2                                     <1> ;----- slen -----
3                                     <1> ; Функция вычисления длины сообщения
4                                     <1> slen:.....
5 00000000 53                         <1> push    ebx.....
6 00000001 89C3                       <1> mov     ebx, eax.....
7                                     <1> .....
8 00000003 803800                     <1> nextchar:.....
9 00000006 7403                       <1> cmp     byte [eax], 0...
10 00000008 40                       <1> jz      finished.....
11 00000009 EBF8                     <1> inc     eax.....
12                                     <1> jmp     nextchar.....
13                                     <1> .....
14 0000000B 29D8                     <1> finished:
15 0000000D 5B                       <1> sub     eax, ebx
16 0000000E C3                       <1> pop     ebx.....
17                                     <1> ret.....
18                                     <1> .....
19                                     <1> ;----- sprint -----
20                                     <1> ; Функция печати сообщения
21                                     <1> ; входные данные: mov eax,<message>

```

Рис. 4.13: Ознакомление

Строки в файле листинга имеют следующую структуру. Первое число обозначает номер строки файла листинга (может не соответствовать номеру строки в файле с исходным кодом). Далее идет адрес (указывает на смещение машинного кода от начала текущего сегмента). Потом идет сам машинный код, он представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. А в конце пишется исходный текст программы, которая является исходной строкой программы вместе с комментариями.

Первая строка, которую я буду описывать имеет номер 38. Далее идет ее адрес (0000013A), который дает нам информацию о смещении машинного кода. Следом идет сам машинный код который является ассемблированием инструкции “mov ecx, max” в шестнадцатеричный код. В конце мы видим как выглядит строка в самой программе (рис. 4.14).

```

38 0000013A 8B0D[00000000]      mov ecx, [max]

```

Рис. 4.14: Анализ первой строки

Вторая строка, которую я буду описывать имеет номер 29. Далее идет ее адрес (0000011D), который дает нам информацию о смещении машинного кода. Следом

идет сам машинный код который является ассемблированием инструкции “jg check_B” в шестнадцатеричный код. В конце мы видим как выглядит строка в самой программе (рис. 4.15).

```
29 0000011D 7F0C          jg check_B ; если 'A>C', то переход на м
```

Рис. 4.15: Анализ второй строки

Третья строка, которую я буду описывать имеет номер 35. Далее идет ее адрес (00000130), который дает нам информацию о смещении машинного кода. Следом идет сам машинный код который является ассемблированием инструкции “call atoi” в шестнадцатеричный код. В конце мы видим как выглядит строка в самой программе (рис. 4.16).

```
35 00000130 E867FFFFFF    call atoi ; Вызов подпрограммы перевода
```

Рис. 4.16: Анализ третьей строки

Открыл файл с программой “lab7-2.asm” и в инструкции с двумя операндами удалил один операнд, выполнил трансляцию листинга и получил ошибку с указанием номера строки (рис. 4.17).

```
[aesoldatov@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:23: error: invalid combination of opcode and operands
[aesoldatov@fedora lab07]$
```

Рис. 4.17: Попытка транслирования файла листинга

После этого на месте ошибки в файле листинга появился комментарий с типом ошибки (рис. 4.18).

```
23          mov [B] ; запись преобразованного числа в 'B'
23          *****
                error: invalid combination of opcode and operands
```

Рис. 4.18: Ошибка в листинге

4.3 Выполнение заданий для самостоятельной работы

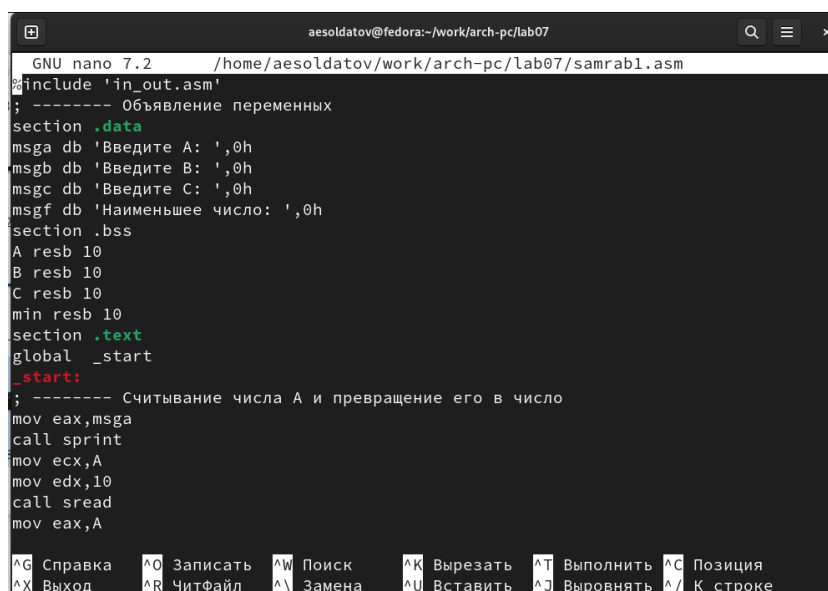
4.3.1 Задание 1

Создал файл “samrab1.asm” (рис. 4.19).

```
[aesoldatov@fedora lab07]$ touch samrab1.asm  
[aesoldatov@fedora lab07]$
```

Рис. 4.19: Создание файла

Написал программу нахождения наименьшей из трех целочисленных переменных (рис. 4.20).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/samrab1.asm  
%include 'in_out.asm'  
; ----- Объявление переменных  
section .data  
msga db 'Введите A: ',0h  
msgb db 'Введите B: ',0h  
msgc db 'Введите C: ',0h  
msgf db 'Наименьшее число: ',0h  
section .bss  
A resb 10  
B resb 10  
C resb 10  
min resb 10  
section .text  
global _start  
_start:  
; ----- Считывание числа A и превращение его в число  
mov eax,msga  
call sprint  
mov ecx,A  
mov edx,10  
call sread  
mov eax,A  
  
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция  
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^_ К строке
```

Рис. 4.20: Ввод текста программы

```
%include 'in_out.asm'  
; ----- Объявление переменных  
section .data  
msga db 'Введите A: ',0h  
msgb db 'Введите B: ',0h  
msgc db 'Введите C: ',0h
```

```

msgf db 'Наименьшее число: ',0h
section .bss
A resb 10
B resb 10
C resb 10
min resb 10
section .text
global _start
_start:
; ----- Считывание числа A и превращение его в число
mov eax,msga
call sprint
mov ecx,A
mov edx,10
call sread
mov eax,A
call atoi
mov [A],eax
; ----- Считывание числа B и превращение его в число
mov eax,msgb
call sprint
mov ecx,B
mov edx,10
call sread
mov eax,B
call atoi
mov [B],eax
; ----- Считывание числа C и превращение его в число
mov eax,msgc

```

```

call sprint
mov ecx,C
mov edx,10
call sread
mov eax,C
call atoi
mov [C],eax
; ----- Нахождение максимума
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx

check_B:
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx

fin:
mov eax, msgf
call sprint
mov eax,[min]
call iprintLF
call quit

```

Создал исполняемый файл и ввел значения переменных из таблицы 7.5 в соответствии с вариантом из прошлой работы (рис. 4.21).

```
[aesoldatov@fedora lab07]$ nasm -f elf samrab1.asm
[aesoldatov@fedora lab07]$ ld -m elf_i386 -o samrab1 samrab1.o
[aesoldatov@fedora lab07]$ ./samrab1
Введите A: 41
Введите B: 62
Введите C: 35
Наименьшее число: 35
[aesoldatov@fedora lab07]$
```

Рис. 4.21: Проверка работы программы

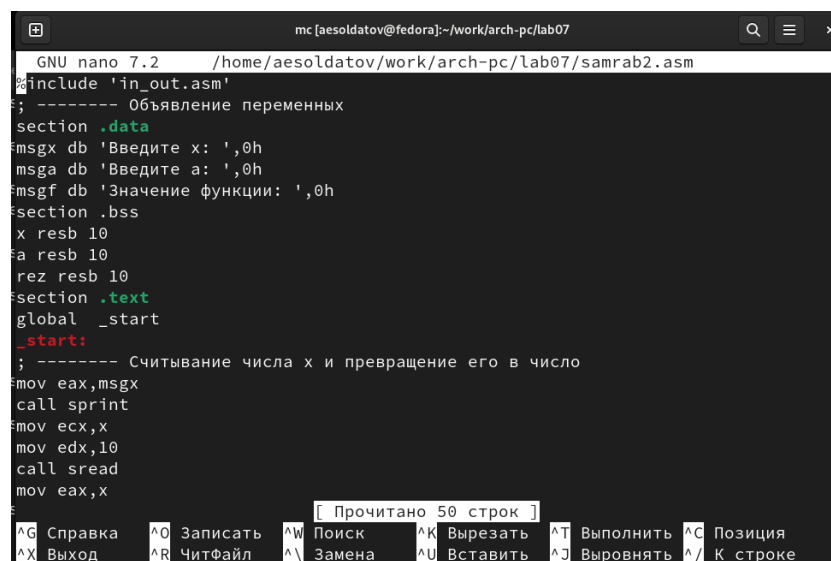
4.3.2 Задание 2

Создал файл “samrab2.asm” (рис. 4.22).

```
[aesoldatov@fedora lab07]$ touch samrab2.asm
```

Рис. 4.22: Создание файла

Написал программу для вычисления значения функции (вид функции взял из таблицы 7.6 в соответствии с вариантом из прошлой работы) (рис. 4.23).



```
GNU nano 7.2 /home/aesoldatov/work/arch-pc/lab07/samrab2.asm
%include 'in_out.asm'
; ----- Объявление переменных
section .data
msgx db 'Введите x: ',0h
msga db 'Введите a: ',0h
msgf db 'Значение функции: ',0h
section .bss
x resb 10
a resb 10
rez resb 10
section .text
global _start
_start:
; ----- Считывание числа x и превращение его в число
mov eax,msgx
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
```

Рис. 4.23: Ввод текста программы

```
%include 'in_out.asm'
; ----- Объявление переменных
```

```

section .data
msgx db 'Введите x: ',0h
msga db 'Введите a: ',0h
msgf db 'Значение функции: ',0h

section .bss
x resb 10
a resb 10
rez resb 10

section .text
global _start
_start:
; ----- Считывание числа x и превращение его в число
mov eax,msgx
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
call atoi
mov [x],eax
; ----- Считывание числа a и превращение его в число
mov eax,msga
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a
call atoi
mov [a],eax

```

```

; ----- Расчет значения функции

mov eax,[x]
cmp eax,2
ja _else
mov eax,[a]
mov ebx,3
mul ebx
mov [rez],eax

_else:
add eax,-2
mov [rez],eax

fin:
mov eax, msgf
call sprint
mov eax,[rez]
call iprintLF
call quit

```

Создал исполняемый файл и ввел значения переменных из таблицы 7.6 в соответствии с вариантом из прошлой работы (рис. 4.24).

```

[aesoldatov@fedora lab07]$ nasm -f elf samrab2.asm
[aesoldatov@fedora lab07]$ ld -m elf_i386 -o samrab2 samrab2.o
[aesoldatov@fedora lab07]$ ./samrab2
Введите x: 3
Введите a: 0
Значение функции: 1
[aesoldatov@fedora lab07]$ ./samrab2
Введите x: 1
Введите a: 2
Значение функции: 4
[aesoldatov@fedora lab07]$

```

Рис. 4.24: Проверка работы программы

5 Выводы

Изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Познакомился с назначением и структурой файла листинга.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.