

Отчет по выполнению лабораторной работы №7

Дисциплина: Архитектура компьютера

Ефремова Полина Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Команды безусловного перехода	7
3.2	Команды условного перехода	7
4	Выполнение лабораторной работы	10
5	Выполнение заданий для самостоятельной работы	25
6	Выводы	33
	Список литературы	34

Список иллюстраций

4.1	Создание файла 1	10
4.2	Ввод программы 1	12
4.3	Запуск файла 1	12
4.4	Изменение файла 1	13
4.5	Запуск измененного файла	14
4.6	Изменяю файл еще раз	15
4.7	Запуск	16
4.8	Создание файла 2	17
4.9	Ввод программы в файл 2	20
4.10	Запуск программы 2	21
4.11	Создание файла листинга	21
4.12	Запуск программы 2	22
4.13	Удаление операнда	23
4.14	В консоли ошибка	23
4.15	В листинге тоже ошибка	24
5.1	Создание нового файла	25
5.2	Ввод программы	28
5.3	Запуск программы	29
5.4	Создание файла и ввод программы	29
5.5	Запуск программы	32

Список таблиц

1 Цель работы

Цель 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.

Структура 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. 4.1).

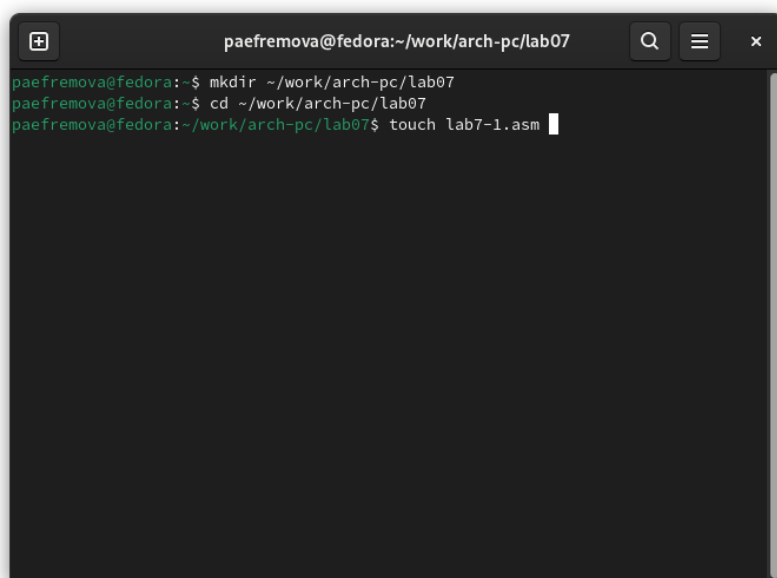


Рис. 4.1: Создание файла 1

2. Ввожу в файл lab7-1.asm программу (рис. 4.2).

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 sprintf
```

```
_label2:
```

```
mov eax, msg2
```

```
call sprintf
```

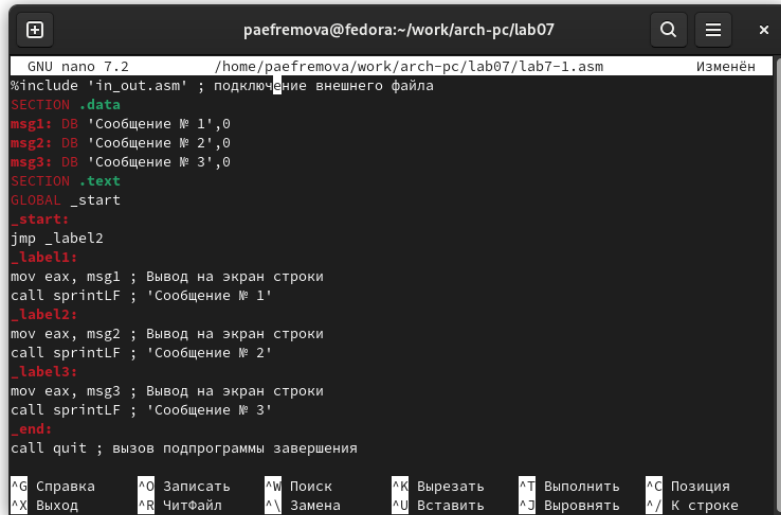
```
label3:
```

```
mov eax, msg3
```

```
call sprintf
```

```
_end:
```

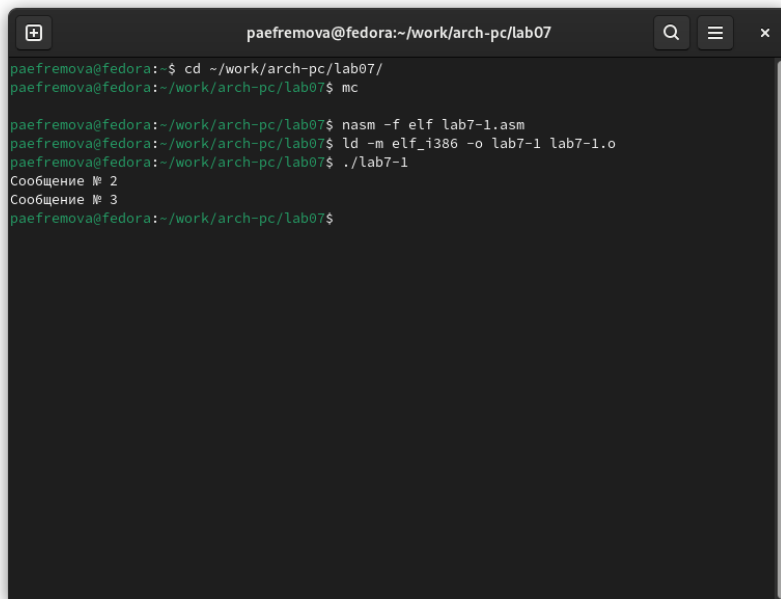
```
call quit
```



```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-1.asm Изменён
#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. 4.2: Ввод программы 1

3. Запускаю файл lab7-1.asm (рис. 4.3).

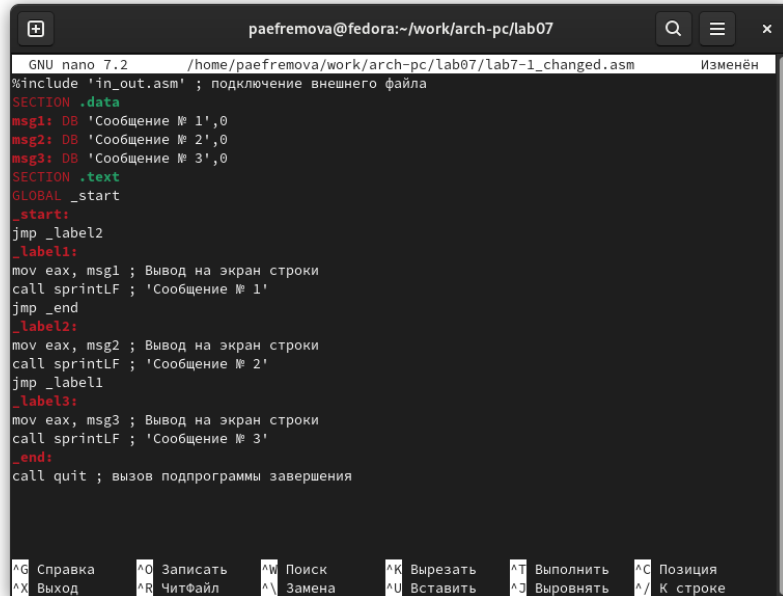


```
paefremova@fedora:~$ cd ~/work/arch-pc/lab07/
paefremova@fedora:~/work/arch-pc/lab07$ mc

paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.3: Запуск файла 1

4. Создаю копию файла и вношу в него изменения (рис. 4.4).



```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-1_changed.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'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: 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
```

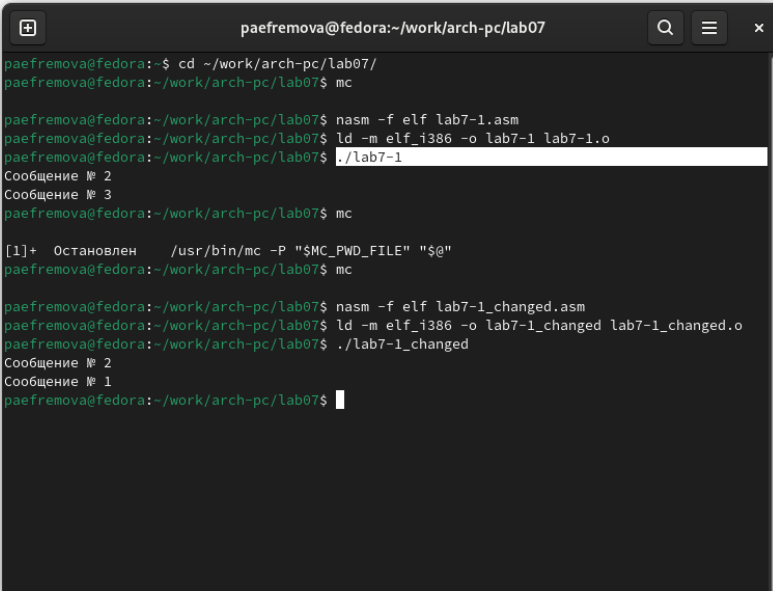
```
call sprintf
jmp _end
```

```
_label2:
mov eax, msg2
call sprintf
jmp _label1
```

```
_label3:
mov eax, msg3
call sprintf
```

```
_end:
call quit
```

5. Запуск измененного файла (рис. 4.5).



```
paefremova@fedora:~/work/arch-pc/lab07
paefremova@fedora:~/work/arch-pc/lab07$ cd ~/work/arch-pc/lab07/
paefremova@fedora:~/work/arch-pc/lab07$ mc

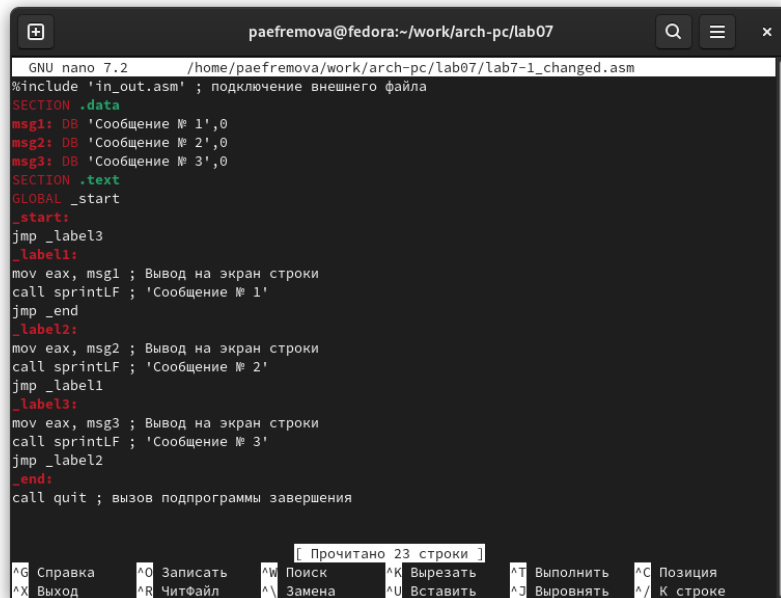
paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
paefremova@fedora:~/work/arch-pc/lab07$ mc

[1]+  Остановлен /usr/bin/mc -P "$MC_PWD_FILE" "$@"
paefremova@fedora:~/work/arch-pc/lab07$ mc

paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1_changed.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1_changed lab7-1_changed.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-1_changed
Сообщение № 2
Сообщение № 1
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.5: Запуск измененного файла

6. В этом же файле переделываю команды так, чтобы сообщения выводились иным образом (рис. 4.6).



```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-1_changed.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'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

Прочитано 23 строки

⌘ Справка ⌘ Записать ⌘ Поиск ⌘ Вырезать ⌘ Выполнить ⌘ Позиция
⌘ Выход ⌘ ЧитФайл ⌘ Замена ⌘ Вставить ⌘ Выводить ⌘ К строке

Рис. 4.6: 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 sprintf
jmp _end

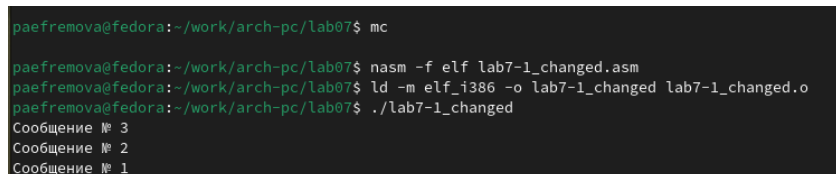
_label2:
mov eax, msg2
call sprintf
jmp _label1

_label3:
mov eax, msg3
call sprintf
jmp _label2

_end:
call quit

```

7. Запускаю измененный файл (рис. 4.7).



```

paefremova@fedora:~/work/arch-pc/lab07$ mc
paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1_changed.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1_changed lab7-1_changed.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-1_changed
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 4.7: Запуск

8. Создаю файл lab7-2.asm (рис. 4.8).

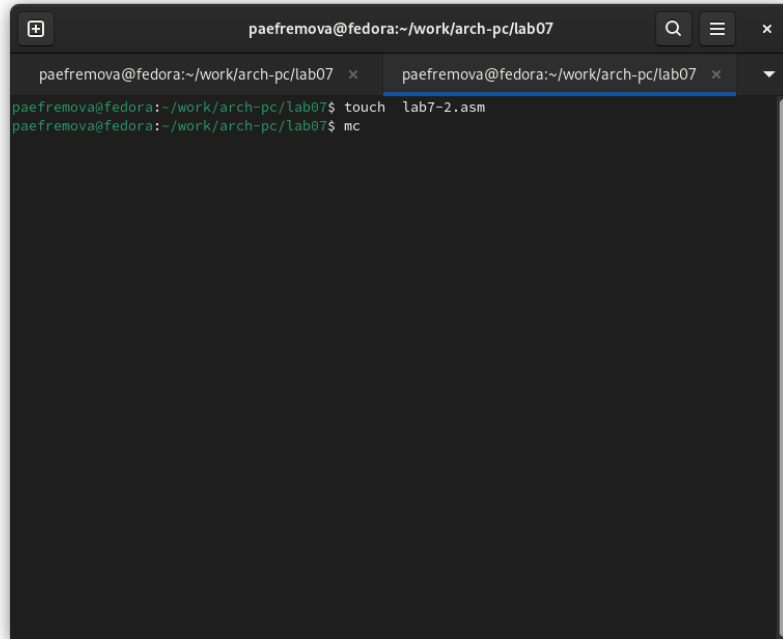


Рис. 4.8: Создание файла 2

9. Ввожу в файл программу из листинга 7.3. (рис. 4.9).

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 '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
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
```

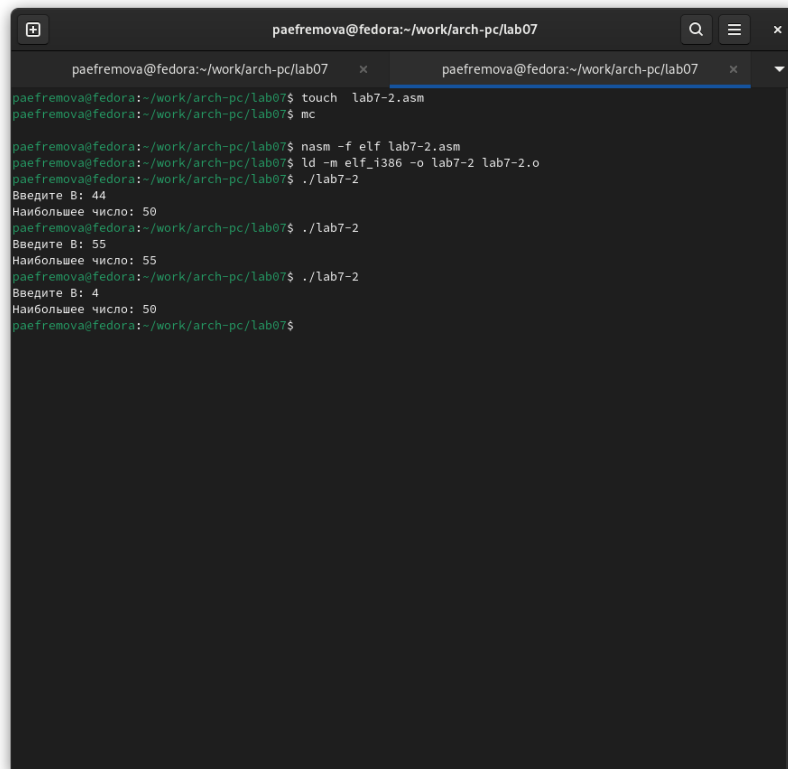
```
fin:
mov eax, msg2
call sprint
mov eax, [max]
call iprintLF
call quit
```

```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-2.asm Изменён
#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'

^G Справка      ^O Записать    ^W Поиск       ^K Вырезать    ^T Выполнить
^X Выход        ^R ЧитФайл    ^N Замена     ^U Вставить    ^J Выровнять
```

Рис. 4.9: Ввод программы в файл 2

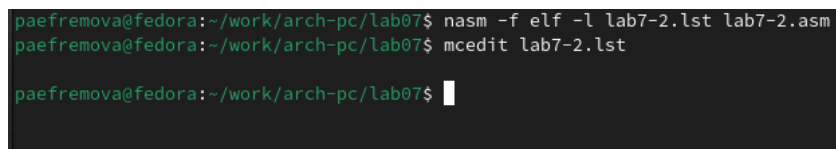
10. Запуск программы из листинга 7.3. (рис. 4.10).



```
paefremova@fedora:~/work/arch-pc/lab07
paefremova@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
paefremova@fedora:~/work/arch-pc/lab07$ mc
paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 44
Наибольшее число: 50
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 55
Наибольшее число: 55
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 4
Наибольшее число: 50
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.10: Запуск программы 2

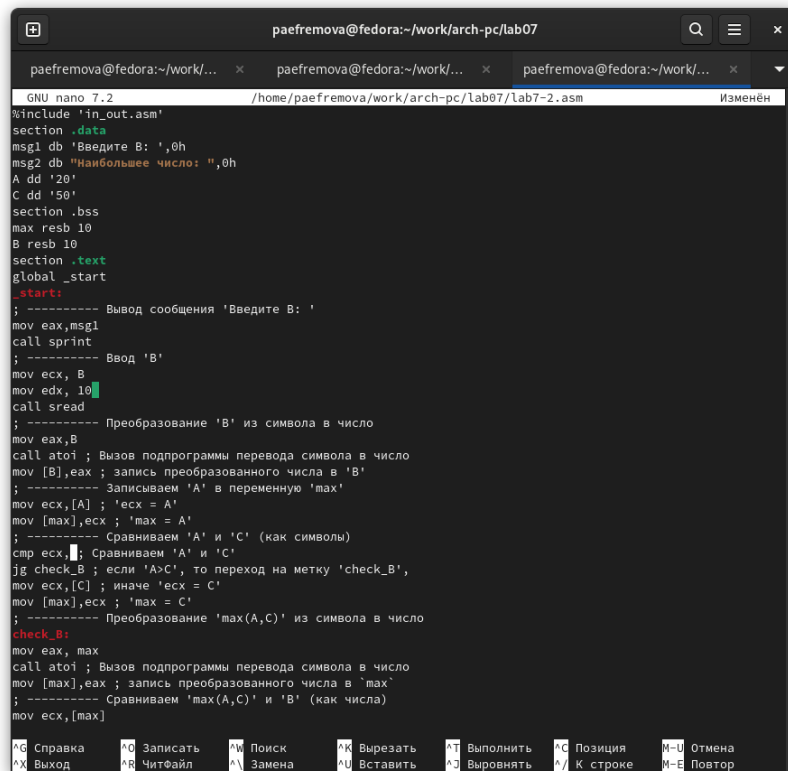
11. Создаю файл листинга программы 2. (рис. 4.11).



```
paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
paefremova@fedora:~/work/arch-pc/lab07$ mcedit lab7-2.lst
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.11: Создание файла листинга

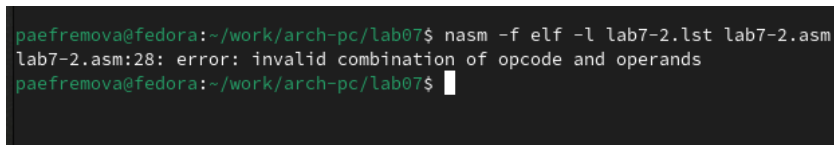
12. Открываю созданный файл (рис. 4.12).



```
GNU nano 7.2 /home/paefremova/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
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]
```

Рис. 4.13: Удаление операнда

14. Консоль выдает ошибку (рис. 4.14).



```
paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 4.14: В консоли ошибка

15. В листинге также появляется ошибка, при этом другие файлы не создаются. (рис. 4.15).

14	000000F8	RR[00000000]	MOV EAX,MAX
15	000000FD	E81DFFFFFF	call sprint
16			; ----- Вывод 'R'.
17	000000F2	RR[0A000000]	MOV ECX, B
18	000000F7	BA0A000000	MOV EDI, 10
19	000000FC	E842FFFFFF	call sprintf
20			; ----- Преобразование 'R' из символа в число
21	00000101	RR[0A000000]	MOV EAX, B
22	00000106	E891FFFFFF	call atoi ; Вывод подпрограммы преобразования символа в число
23	0000010B	A3[0A000000]	MOV [B],EAX ; Записываем преобразованного числа в 'B'.
24			; ----- Записываем 'A' в переменную 'max'.
25	00000110	8B00[35000000]	MOV ECX, [A] ; ECX = A.
26	00000116	8B00[00000000]	MOV [max],ECX ; 'max' = A.
27			; ----- Сравниваем 'A' и 'C' (как символы)
28			CMP ECX, ; Сравниваем 'A' и 'C'.
28		*****	error: invalid combination of opcode and operands
29	0000011C	7F0C	jg check_B ; Если 'A'>'C', то переход на метку 'check_B'.
30	0000011E	8B00[39000000]	MOV ECX, [C] ; Иначе ECX = C.
31	00000124	8B00[00000000]	MOV [max],ECX ; 'max' = C.

Рис. 4.15: В листинге тоже ошибка

5 Выполнение заданий для самостоятельной работы

1. Создаю файл lab7-3.asm для работы. (рис. 5.1).

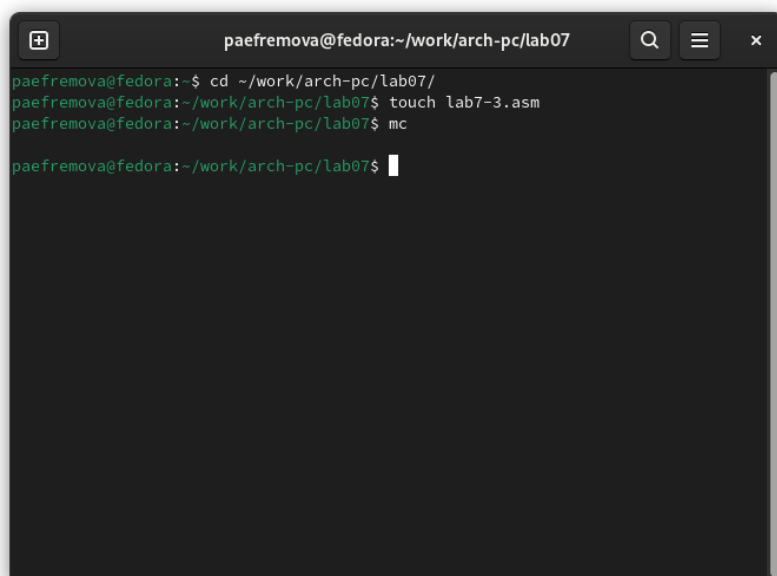


Рис. 5.1: Создание нового файла

Ввожу в файл программу (рис. 5.2).

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

```

```

GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-3.asm
#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

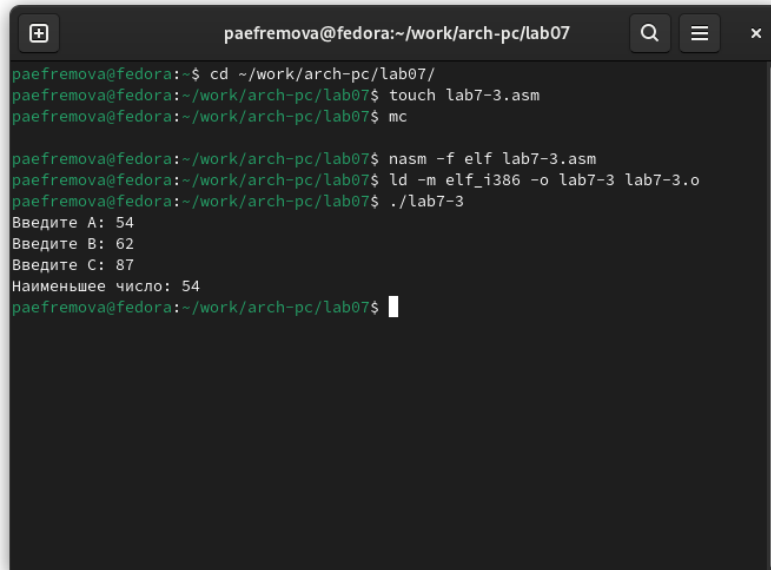
section .text
global _start
_start:

mov eax, msg2
call sprint

```

Рис. 5.2: Ввод программы

Запуск программы (рис. 5.3).

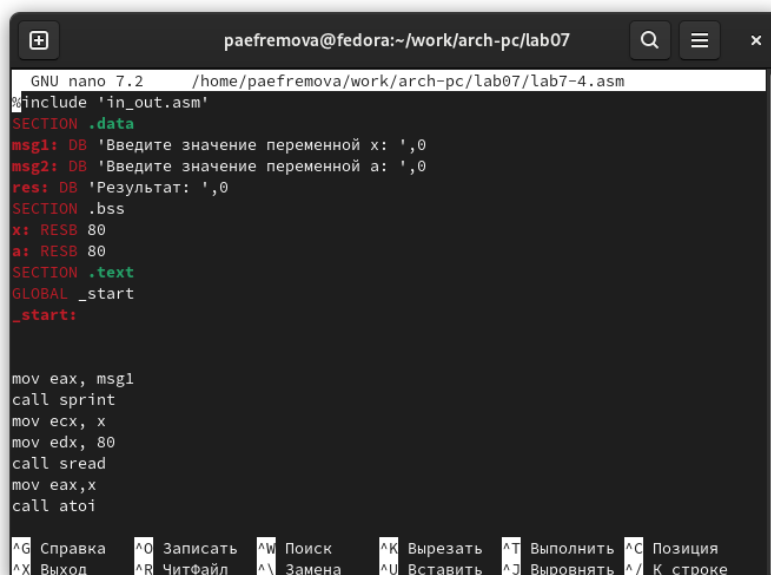


```
paefremova@fedora:~/work/arch-pc/lab07
paefremova@fedora:~/work/arch-pc/lab07$ cd ~/work/arch-pc/lab07/
paefremova@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
paefremova@fedora:~/work/arch-pc/lab07$ mc

paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 54
Введите B: 62
Введите C: 87
Наименьшее число: 54
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 5.3: Запуск программы

2.Создаю файл для второго самостоятельного задания и ввожу туда программу (рис. 5.4).



```
GNU nano 7.2 /home/paefremova/work/arch-pc/lab07/lab7-4.asm
#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

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

Рис. 5.4: 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

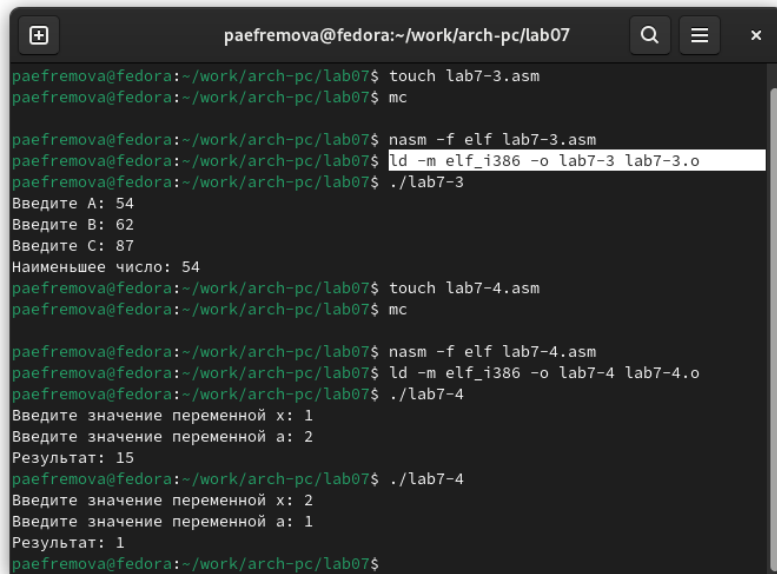
mov edi,15 ; edi = 15
jmp fin

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. 5.5).



```
paefremova@fedora:~/work/arch-pc/lab07
paefremova@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
paefremova@fedora:~/work/arch-pc/lab07$ mc

paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 54
Введите B: 62
Введите C: 87
Наименьшее число: 54
paefremova@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
paefremova@fedora:~/work/arch-pc/lab07$ mc

paefremova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
paefremova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 1
Введите значение переменной a: 2
Результат: 15
paefremova@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 1
Результат: 1
paefremova@fedora:~/work/arch-pc/lab07$
```

Рис. 5.5: Запуск программы

6 Выводы

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

Список литературы

1.Архитектура ЭВМ

2.Архитектура ЭВМ