

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютеров

Воронов Александр Валерьевич

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 файла листинга	13
4.3	Задания для самостоятельной работы	15
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Создание каталога и файла для программы	8
4.2	Сохранение программы	9
4.3	Запуск программы	9
4.4	Изменение программы	10
4.5	Запуск измененной программы	10
4.6	Изменение программы	11
4.7	Проверка изменений	11
4.8	Сохранение новой программы	12
4.9	Проверка программы из листинга	12
4.10	Проверка файла листинга	13
4.11	Удаление операнда из программы	14
4.12	Просмотр ошибки в файле листинга	14
4.13	Первая программа самостоятельной работы	15
4.14	Проверка работы первой программы	16
4.15	Вторая программа самостоятельной работы	16
4.16	Проверка работы второй программы	17

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной 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 программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. -fig. 4.1).

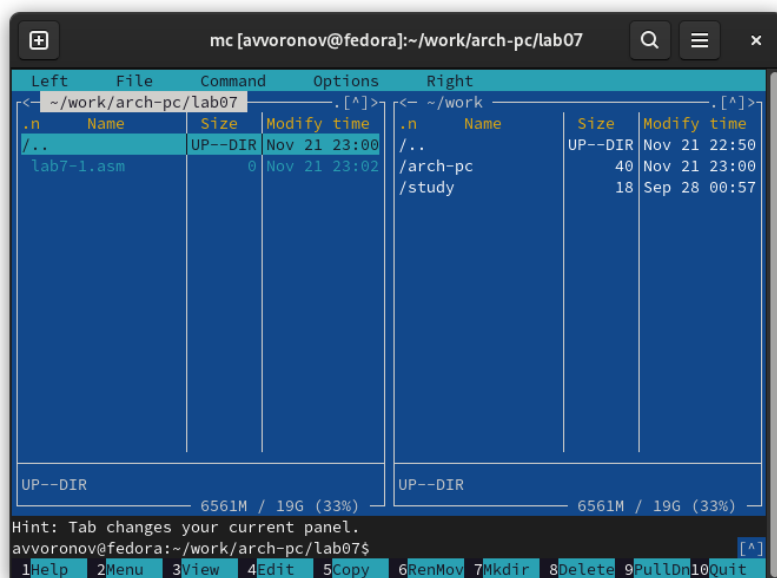
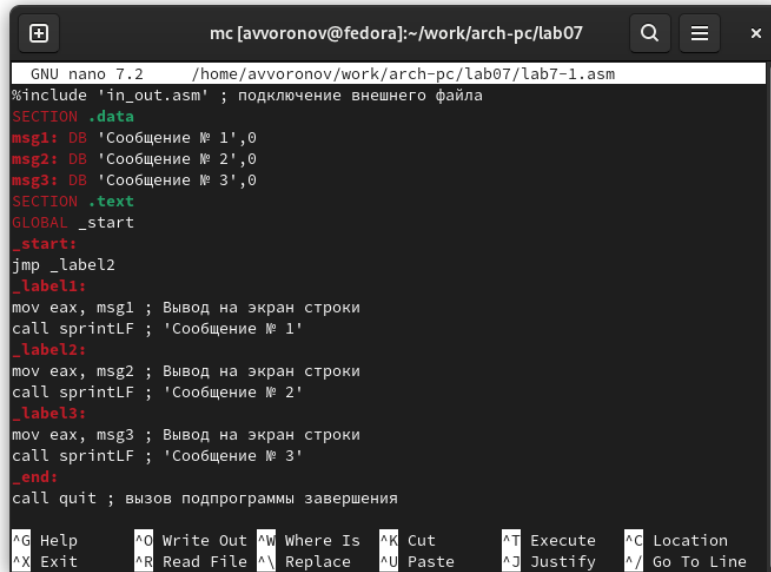


Рис. 4.1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. -fig. 4.2).

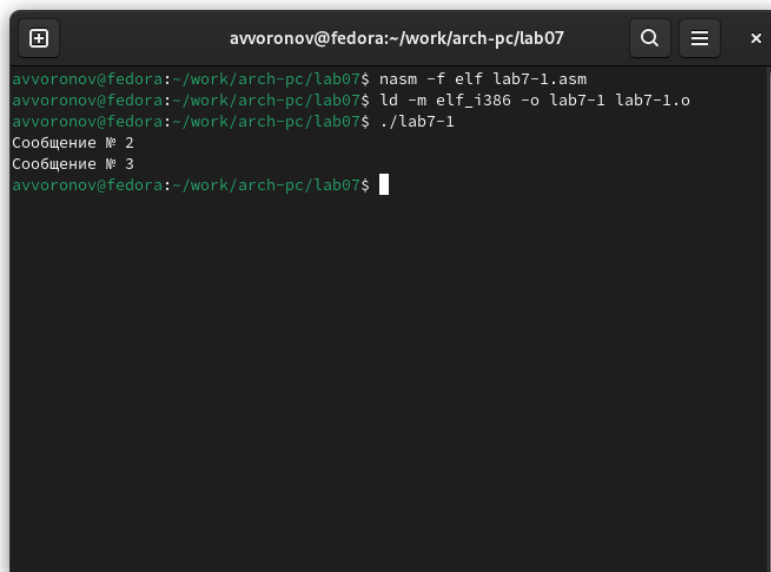


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
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 Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.2: Сохранение программы

При запуске программы я убедился в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. -fig. 4.3).

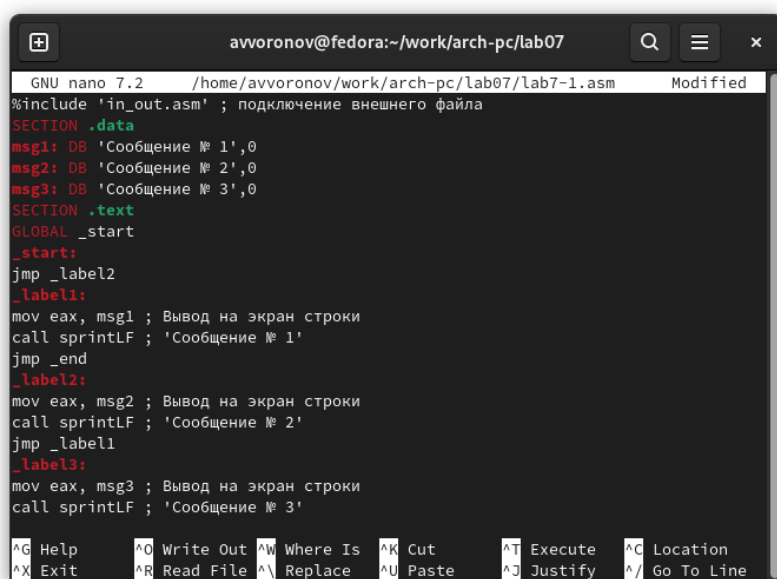


```
avvoronov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
avvoronov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
avvoronov@fedora:~/work/arch-pc/lab07$
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения

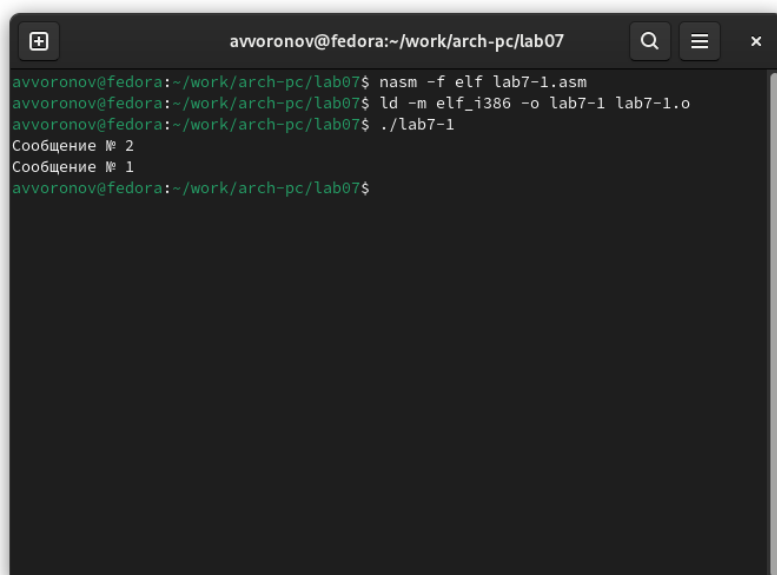
функций (рис. -fig. 4.4).



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

Рис. 4.4: Изменение программы

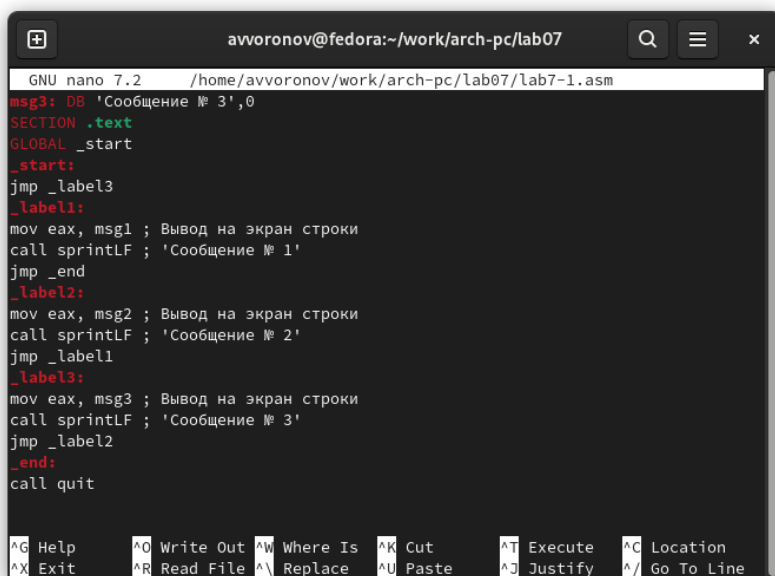
Запускаю программу и проверяю, что примененные изменения верны (рис. -fig. 4.5).



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

Рис. 4.5: Запуск измененной программы

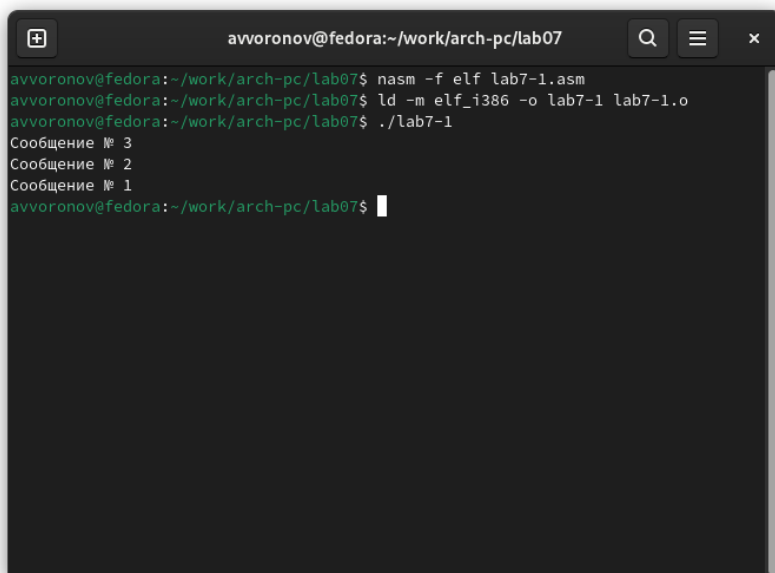
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. -fig. 4.6).



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

Рис. 4.6: Изменение программы

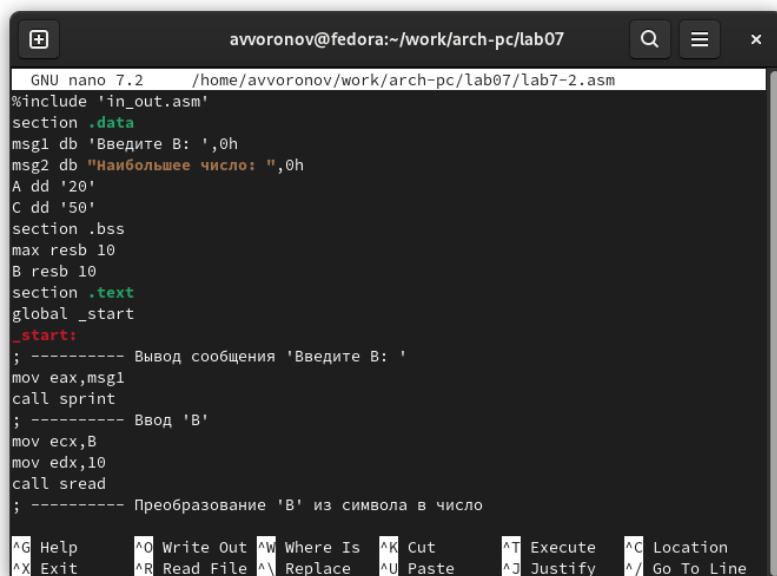
Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. -fig. 4.7).



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

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. -fig. 4.8).

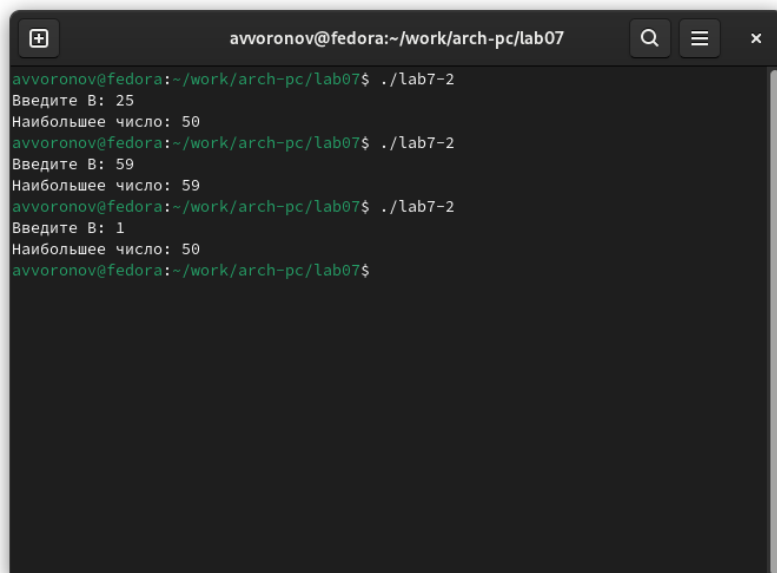


```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. -fig. 4.9).



```
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 59
Наибольшее число: 59
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
avvoronov@fedora:~/work/arch-pc/lab07$
```

Рис. 4.9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. -fig. 4.10).

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

Рис. 4.10: Проверка файла листинга

Объясняю три строки из файла листинга: 23 00000106 E891FFFFFF call atoi - Вызов подпрограммы перевода символа в число; 23 - номер строки, 00000106 - адрес, E891FFFFFF - машинный код; 41 0000014B 7F0C jg fin - переход на label 'fin', если 'max(A,C)>B'; 41 - номер строки, 0000014B - адрес, 7F0C - машинный код; 50 0000016D E869FFFFFF call quit - Выход из программы; 50 - номер строки; 0000016D - адрес; E869FFFFFF - машинный код.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. -fig. 4.11).

```

mc [aworonov@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab07/lab7-2.asm Modified
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
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line

```

Рис. 4.11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. -fig. 4.12).

```

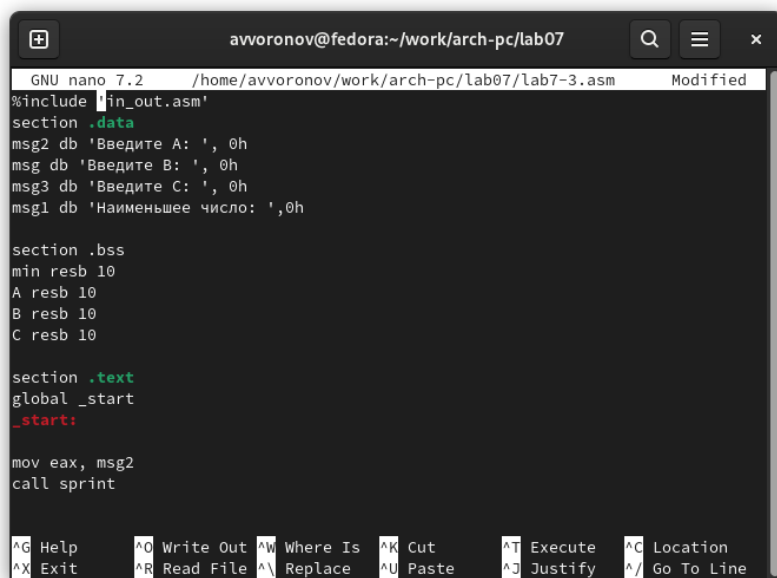
aworonov@fedora:~/work/arch-pc/lab07 — mcedit lab7-2.lst
lab7-2.lst  [----]  0 L:[205+ 1 206/226] *(12915/14545b) 0032 0x020[*][X]
30 00000124 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]    mov [max],ecx ; 'max = C'
32                                ; ----- Преобразование 'max(A,C)' и
33                                check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF        call atoi ; Вызов подпрограммы перевода
36 0000013A A3[00000000]      mov [max],eax ; запись преобразованного
37                                ; ----- Сравниваем 'max(A,C)' и 'B'
38 0000013F 8B0D[00000000]    mov ecx,[max]
39                                cmp ecx ; Сравниваем 'max(A,C)' и 'B'
39                                *****
39                                error: invalid combination of opcode and
40 00000145 7F0C              jg fin ; если 'max(A,C)>B', то переход на
41 00000147 8B0D[0A000000]    mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 890D[00000000]    mov [max],ecx
43                                ; ----- Вывод результата
44                                fin:
45 00000153 B8[13000000]      mov eax, msg2
46 00000158 E8B2FFFFFF        call sprint ; Вывод сообщения 'Наибольшее
47 0000015D A1[00000000]      mov eax,[max]
48 00000162 E81FFFFFFF        call iprintLF ; Вывод 'max(A,B,C)'
49 00000167 E86FFFFFFF        call quit ; Выход
1Help  2Save  3Mark  4Replac 5Copy  6Move  7Search 8Delete 9PullDn10Quit

```

Рис. 4.12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. -fig. 4.13).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab07/lab7-3.asm Modified
%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

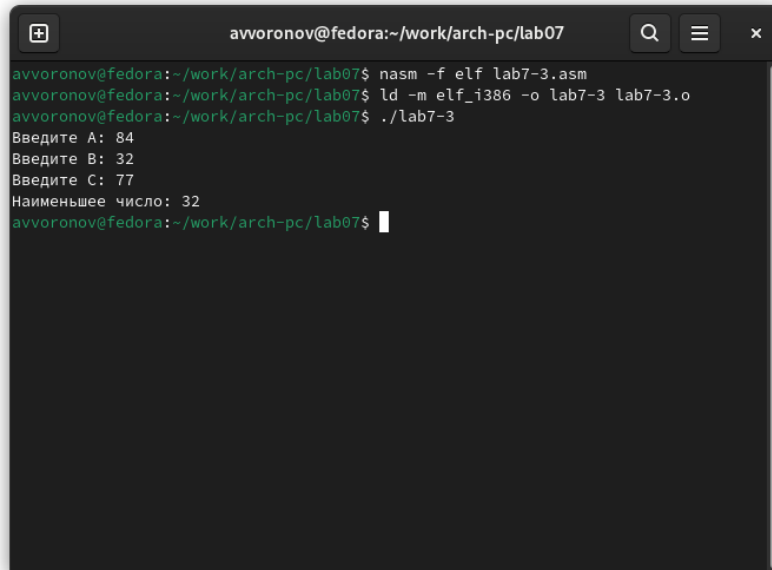
section .bss
min resb 10
A resb 10
B resb 10
C resb 10

section .text
global _start
_start:

mov eax, msg2
call sprint
```

Рис. 4.13: Первая программа самостоятельной работы

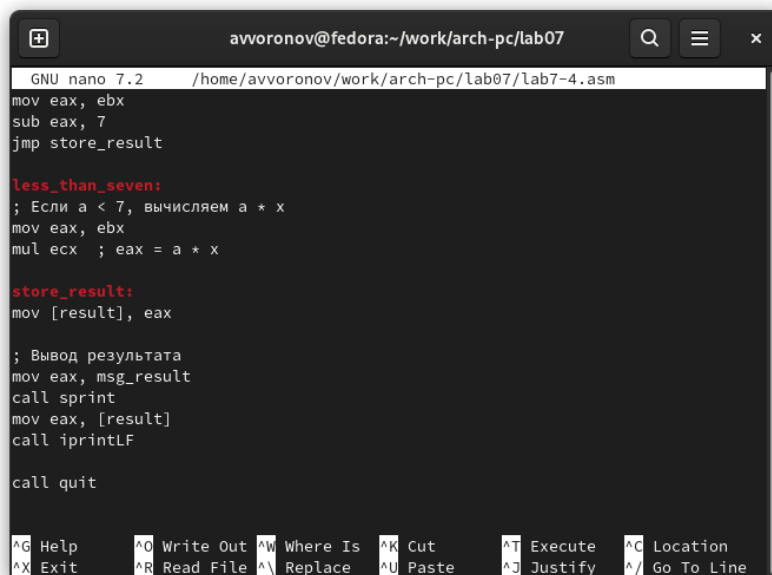
Проверяю корректность написания первой программы (рис. -fig. 4.14).



```
avvoronov@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
avvoronov@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 84
Введите B: 32
Введите C: 77
Наименьшее число: 32
avvoronov@fedora:~/work/arch-pc/lab07$
```

Рис. 4.14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных a и x (рис. -fig. 4.15).



```
GNU nano 7.2 /home/avvoronov/work/arch-pc/lab07/lab7-4.asm
mov eax, ebx
sub eax, 7
jmp store_result

less_than_seven:
; Если a < 7, вычисляем a * x
mov eax, ebx
mul ecx ; eax = a * x

store_result:
mov [result], eax

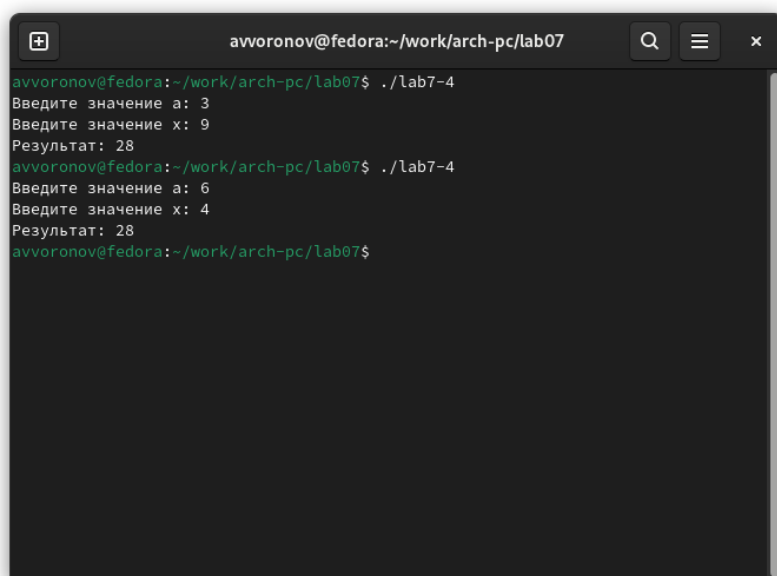
; Вывод результата
mov eax, msg_result
call sprint
mov eax, [result]
call iprintLF

call quit

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^N Replace   ^U Paste     ^D Justify   ^_ Go To Line
```

Рис. 4.15: Вторая программа самостоятельной работы

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х (рис. -fig. 4.16).



```
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение а: 3
Введите значение х: 9
Результат: 28
avvoronov@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение а: 6
Введите значение х: 4
Результат: 28
avvoronov@fedora:~/work/arch-pc/lab07$
```

Рис. 4.16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Программирование на языке ассемблера NASM Столяров А. В.