

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Мазурский Александр Дмитриевич

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	15
4.3	Задания для самостоятельной работы	18
5	Выводы	25
	Список литературы	26

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	11
4.6	Изменение программы	12
4.7	Проверка изменений	13
4.8	Сохранение новой программы	14
4.9	Проверка программы из листинга	15
4.10	Проверка файла листинга	16
4.11	Удаление операнда из программы	17
4.12	Просмотр ошибки в файле листинга	17
4.13	Первая программа самостоятельной работы	18
4.14	Проверка работы первой программы	21
4.15	Вторая программа самостоятельной работы	22
4.16	Проверка работы второй программы	24

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

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

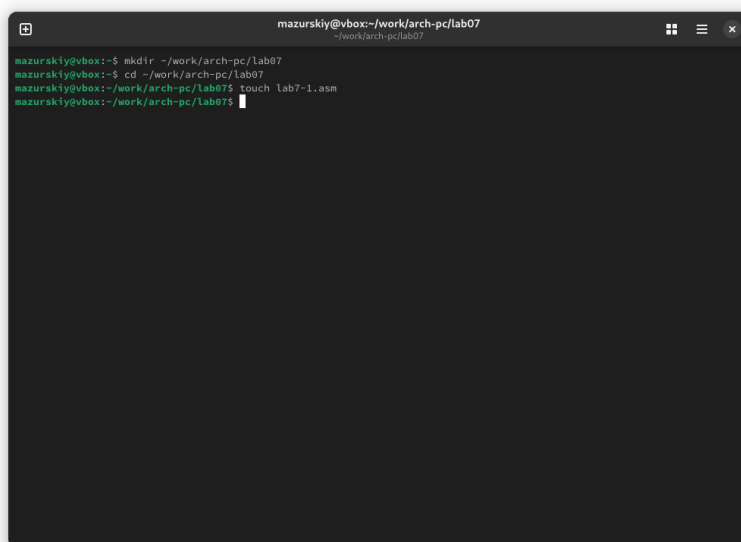
3 Теоретическое введение

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

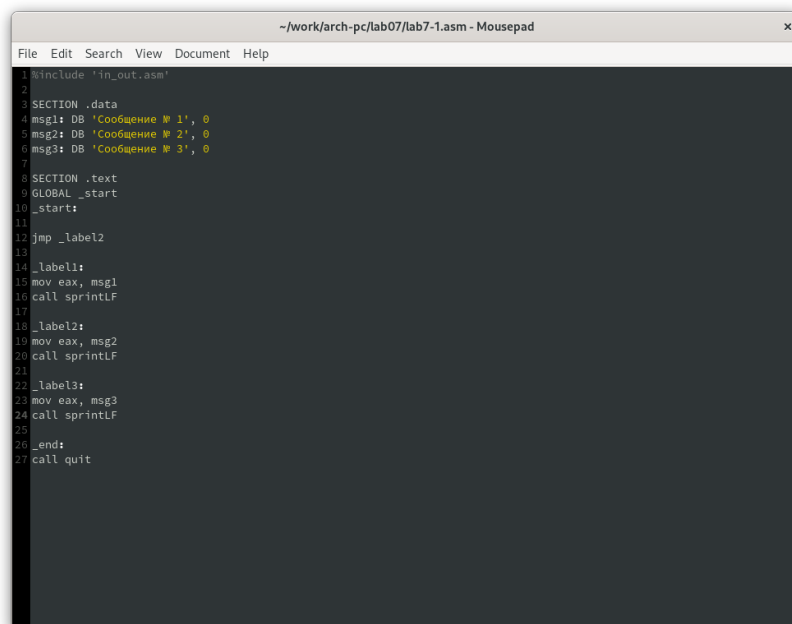
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).



```
mazurskiy@vbox:~/work/arch-pc/lab07
mazurskiy@vbox:~$ mkdir ~/work/arch-pc/lab07
mazurskiy@vbox:~$ cd ~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание каталога и файла для программы

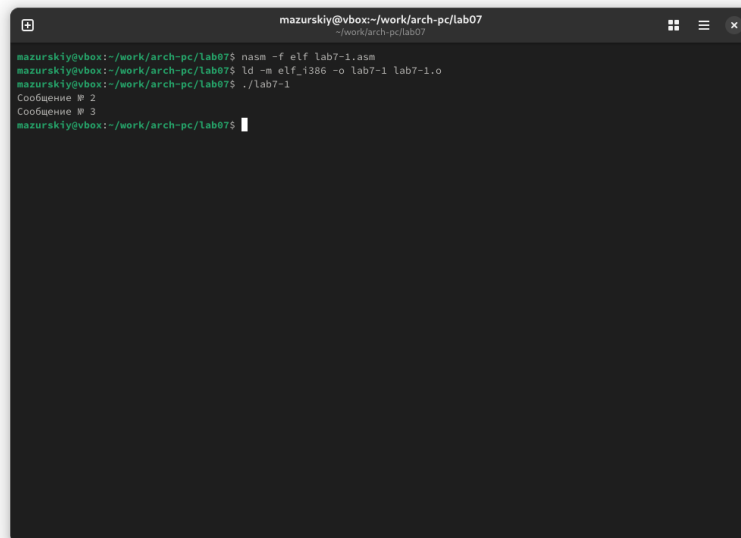
Копирую код из листинга в файл будущей программы. (рис. 4.2).



```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17
18 _label2:
19 mov eax, msg2
20 call sprintf
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25
26 _end:
27 call quit
```

Рис. 4.2: Сохранение программы

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

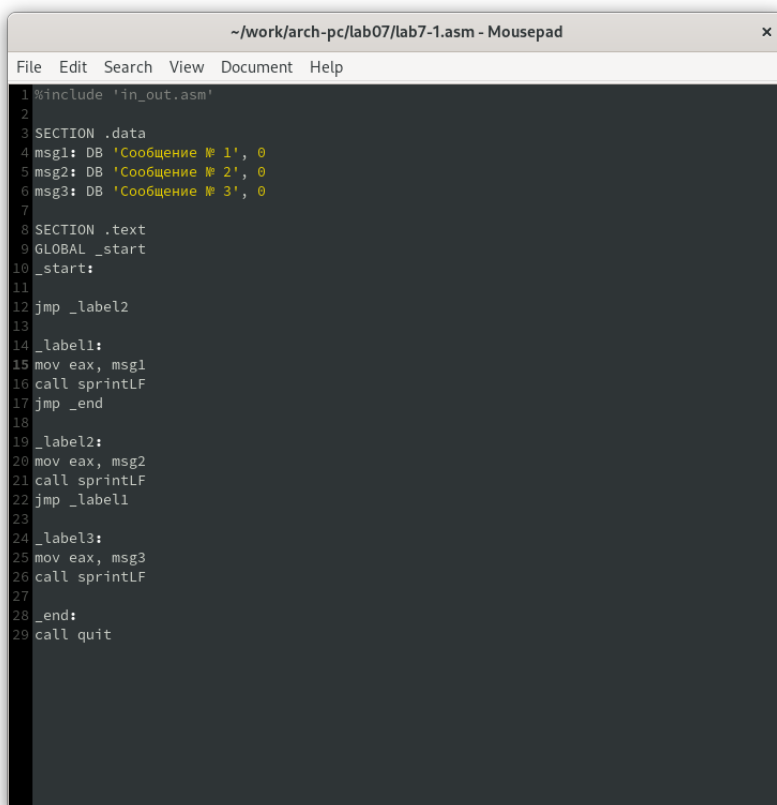


```
mazurskiy@vbox:~/work/arch-pc/lab07
~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
mazurskiy@vbox:~/work/arch-pc/lab07$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

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения

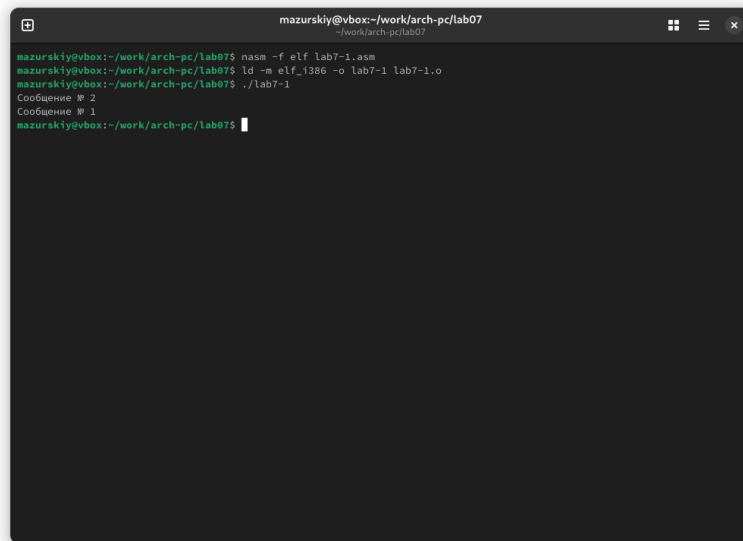
функций (рис. 4.4).



```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintf
27
28 _end:
29 call quit
```

Рис. 4.4: Изменение программы

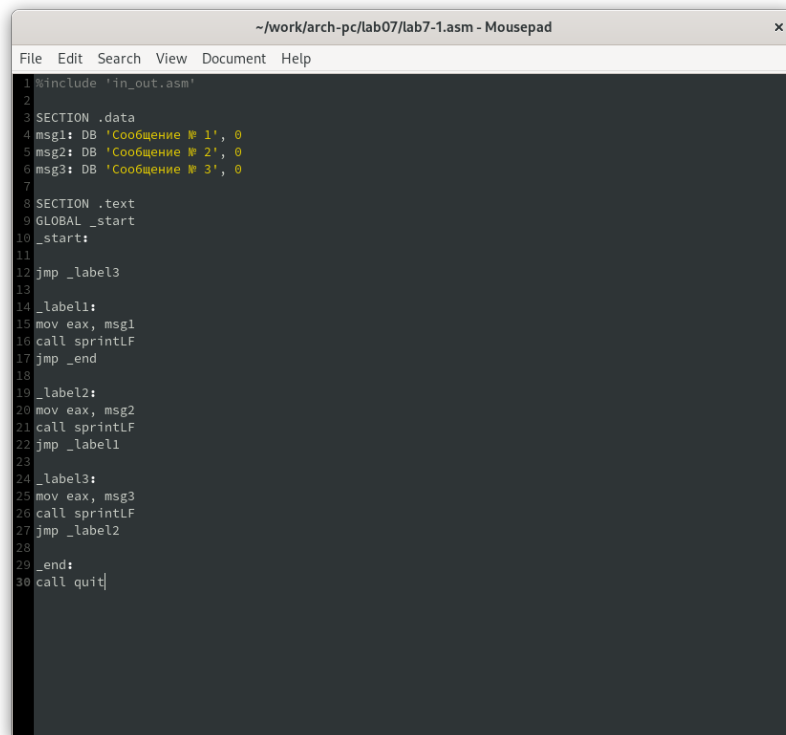
Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).

A terminal window with a dark background and light green text. The window title is "mazurskiy@vbox:~/work/arch-pc/lab07". The terminal shows the following commands and output:

```
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение # 2
Сообщение # 1
mazurskiy@vbox:~/work/arch-pc/lab07$
```

Рис. 4.5: Запуск измененной программы

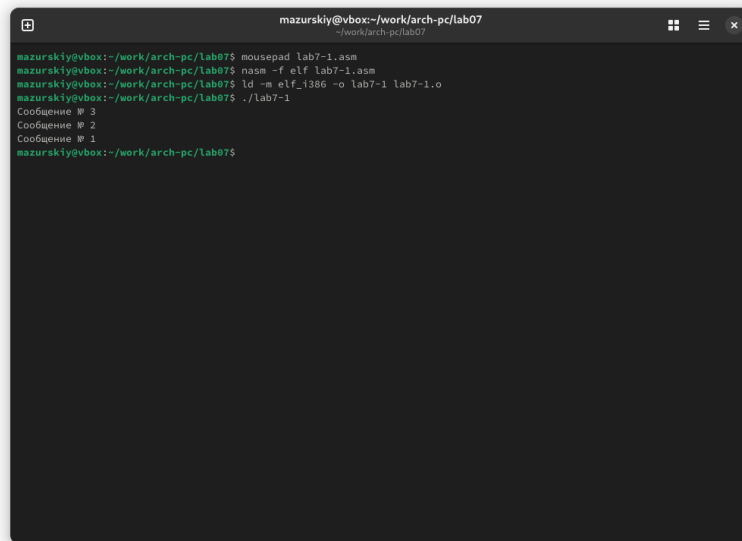
Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 4.6).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1', 0
5 msg2: DB 'Сообщение № 2', 0
6 msg3: DB 'Сообщение № 3', 0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15 mov eax, msg1
16 call sprintf
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintf
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintf
27 jmp _label2
28
29 _end:
30 call quit
```

Рис. 4.6: Изменение программы

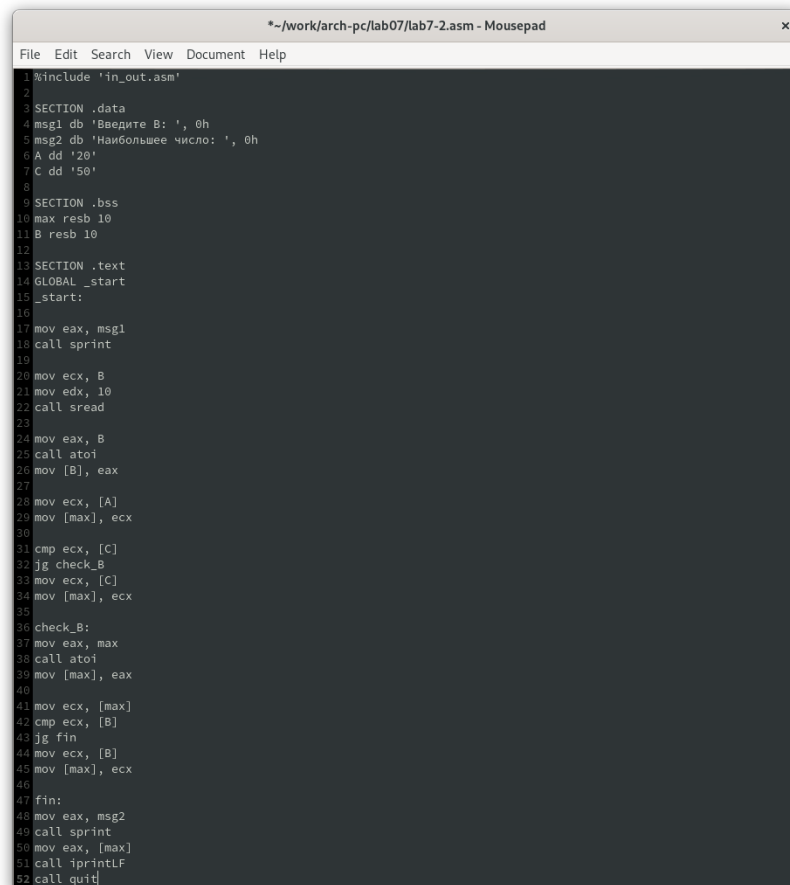
Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 4.7).



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

Рис. 4.7: Проверка изменений

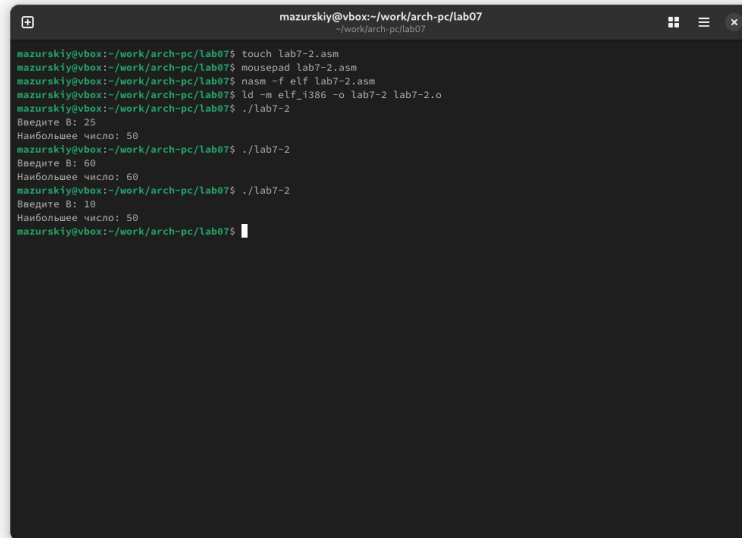
Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 4.8).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наибольшее число: ', 0h
6 A dd '120'
7 C dd '50'
8
9 SECTION .bss
10 max resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [max], ecx
35
36 check_B:
37 mov eax, max
38 call atoi
39 mov [max], eax
40
41 mov ecx, [max]
42 cmp ecx, [B]
43 jg fin
44 mov ecx, [B]
45 mov [max], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [max]
51 call tprintf
52 call quit
```

Рис. 4.8: Сохранение новой программы

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



```
mazurskiy@vbox: ~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ touch lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ mousepad lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 25
Наибольшее число: 50
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
mazurskiy@vbox:~/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. 4.10).

```

~\work\arch-pc\lab07\lab7-2.list - Mousepad
File Edit Search View Document Help
1 1 %include "in_out.asm"
2 1 <|> ;----- slen -----
3 2 <|> ; функция вычисления длины сообщения
4 3 <|> slen:
5 4 00000000 53 <|> push ebx
6 5 00000001 89C3 <|> mov ebx, eax
7 6 <|>
8 7 <|> nextchar:
9 8 00000003 803800 <|> cmp byte [eax], 0
10 9 00000006 7403 <|> jz finished
11 10 00000009 40 <|> inc eax
12 11 00000009 EBF8 <|> jmp nextchar
13 12 <|>
14 13 <|> finished:
15 14 00000008 29D8 <|> sub eax, ebx
16 15 00000000 5B <|> pop ebx
17 16 0000000E C3 <|> ret
18 17 <|>
19 18 <|>
20 19 <|> ;----- sprintf -----
21 20 <|> ; функция печати сообщения
22 21 <|> ; входные данные: mov eax, <message>
23 22 <|> sprintf:
24 23 0000000F 52 <|> push edx
25 24 00000010 51 <|> push ecx
26 25 00000011 53 <|> push ebx
27 26 00000012 50 <|> push eax
28 27 00000013 18E8FFFFFF <|> call slen
29 28 <|>
30 29 00000018 89C2 <|> mov edx, eax
31 30 0000001A 58 <|> pop eax
32 31 <|>
33 32 0000001B 89C1 <|> mov ecx, eax
34 33 0000001D 8B01000000 <|> mov ebx, 1
35 34 00000022 8B04000000 <|> mov eax, 4
36 35 00000027 CD80 <|> int 80h
37 36 <|>
38 37 00000029 5B <|> pop ebx
39 38 0000002A 59 <|> pop ecx
40 39 0000002B 5A <|> pop edx
41 40 0000002C C3 <|> ret
42 41 <|>
43 42 <|>
44 43 <|> ;----- sprintf -----
45 44 <|> ; функция печати сообщения с переводом строки
46 45 <|> ; входные данные: mov eax, <message>
47 46 <|> sprintf:
48 47 0000002D E800FFFFFF <|> call sprintf
49 48 <|>
50 49 00000027 5B <|> push ebx

```

Рис. 4.10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

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


```

File Edit Search View Document Help
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [max], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [max], ecx
35
36 check_B:
37 mov eax, |
38 call atoi
39 mov [max], eax
40
41 mov ecx, [max]
42 cmp ecx, [B]
43 jg fin
44 mov ecx, [B]
45 mov [max], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [max]
51 call iprintf
52 call quit
53
54
55
56
57

```

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. 4.12).

```

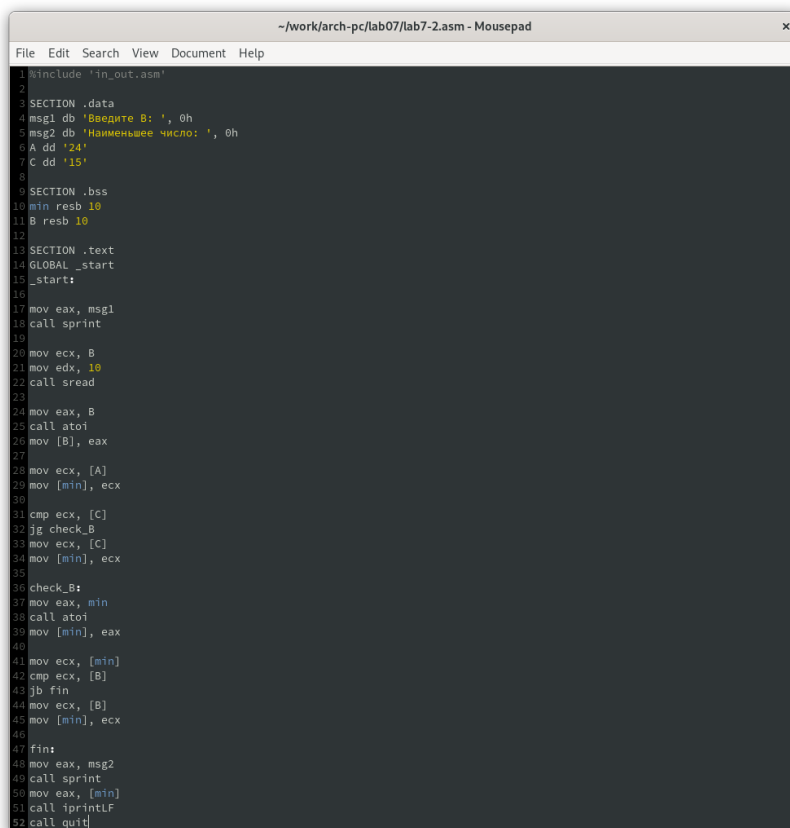
File Edit Search View Document Help
192 18 000000D E810FFFFFF call sprint
193 19
194 20 000000F2 B90A000000 mov ecx, B
195 21 000000F7 B90A000000 mov edx, 10
196 22 000000FC E842FFFFFF call sread
197 23
198 24 00000101 B80A000000 mov eax, B
199 25 00000106 E891FFFFFF call atoi
200 26 0000010B A30A000000 mov [B], eax
201 27
202 28 00000110 8B0036000000 mov ecx, [A]
203 29 00000116 8900000000 mov [max], ecx
204 30
205 31 0000011C 3B003A000000 cmp ecx, [C]
206 32 00000122 7F0C jg check_B
207 33 00000124 8B003A000000 mov ecx, [C]
208 34 0000012A 8900000000 mov [max], ecx
209 35
210 36
211 37
212 37
213 37 ***** error: invalid combination of opcode and operands
214 38 00000120 E807FFFFFF call atoi
215 39 00000135 A300000000 mov [max], eax
216 40
217 41 0000013A 8B00000000 mov ecx, [max]
218 42 00000140 3B000A000000 cmp ecx, [B]
219 43 00000146 7F0C jg fin
220 44 00000148 8B000A000000 mov ecx, [B]
221 45 0000014E 8900000000 mov [max], ecx
222 46
223 47
224 48 00000154 B814000000 fin:
225 49 00000159 E8B1FFFFFF mov eax, msg2
226 50 0000015E A100000000 call sprint
227 51 00000163 E81EFFFFFF mov eax, [max]
228 52 00000168 E86EFFFFFF call iprintf
229 53
230 54
231 55

```

Рис. 4.12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должен был получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.13).



```
~/work/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '24'
7 C dd '15'
8
9 SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [min], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [min], ecx
35
36 check_B:
37 mov eax, min
38 call atoi
39 mov [min], eax
40
41 mov ecx, [min]
42 cmp ecx, [B]
43 jb fin
44 mov ecx, [B]
45 mov [min], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [min]
51 call iprintLF
52 call quit
```

Рис. 4.13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
```

```
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'
```

```
SECTION .bss
```

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

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

```
cmp ecx, [C]
jg check_B
mov ecx, [C]
```

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

```
check_B:
```

```
mov eax, min
```

```
call atoi
```

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

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

```
cmp ecx, [B]
```

```
jb fin
```

```
mov ecx, [B]
```

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

```
fin:
```

```
mov eax, msg2
```

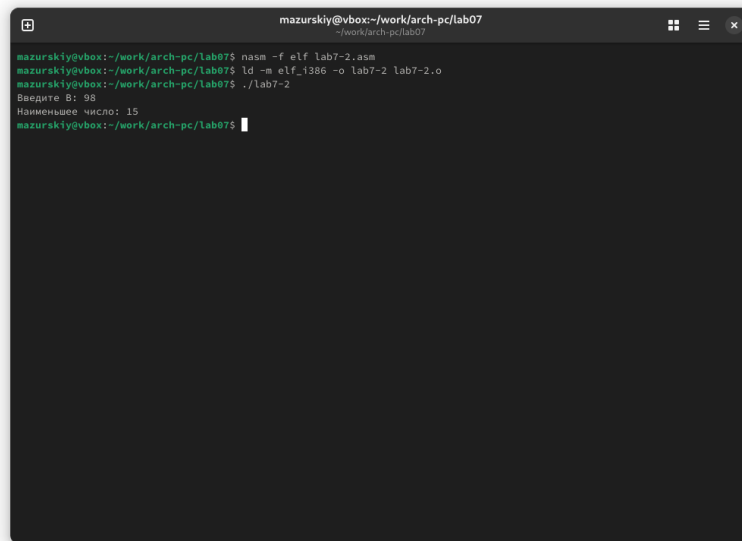
```
call sprint
```

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

```
call iprintLF
```

```
call quit
```

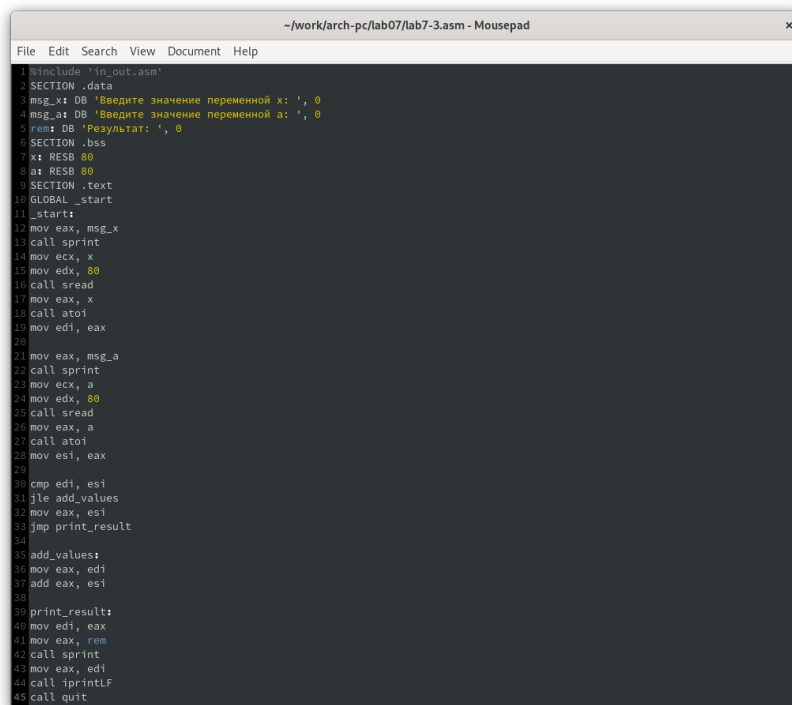
Проверяю корректность написания первой программы (рис. 4.14).



```
mazurskiy@vbox:~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 98
Наименьшее число: 15
mazurskiy@vbox:~/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. 4.15).



```
~/work/arch-pc/lab07/lab7-3.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20
21 mov eax, msg_a
22 call sprint
23 mov ecx, a
24 mov edx, 80
25 call sread
26 mov eax, a
27 call atoi
28 mov esi, eax
29
30 cmp edi, esi
31 jle add_values
32 mov eax, esi
33 jmp print_result
34
35 add_values:
36 mov eax, edi
37 add eax, esi
38
39 print_result:
40 mov edi, eax
41 mov eax, res
42 call sprint
43 mov eax, edi
44 call iprintf
45 call quit
```

Рис. 4.15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: 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, msg_x
call sprint
```

```

mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

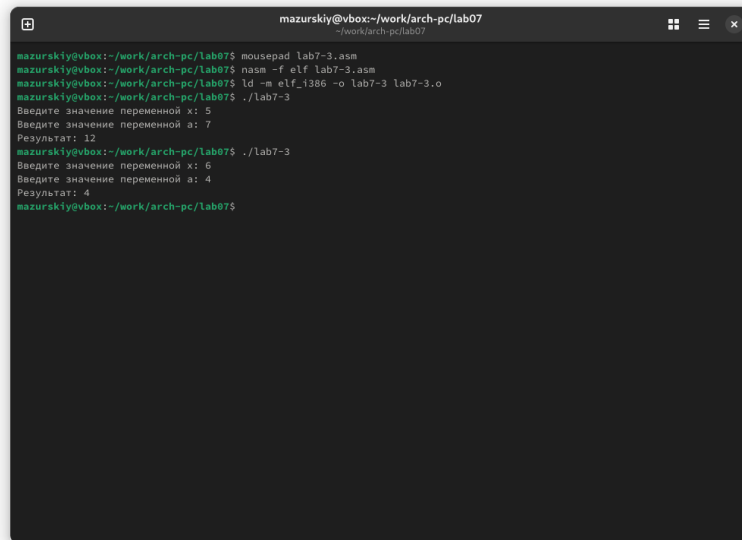
add_values:
mov eax, edi
add eax, esi

print_result:
mov edi, eax
mov eax, res
call sprint

```

```
mov eax, edi
call iprintLF
call quit
```

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



```
mazurskiy@vbox:~/work/arch-pc/lab07
mazurskiy@vbox:~/work/arch-pc/lab07$ mousepad lab7-3.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mazurskiy@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
mazurskiy@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
mazurskiy@vbox:~/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. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.