

Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Киселева Елизавета Александровна

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	14
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создание каталога и файла для программы	8
4.2	Написание 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	Создание нового файла	11
4.9	Создание новой программы	12
4.10	Проверка программы из листинга	12
4.11	Создание файла листинга	13
4.12	Файл листинга	13
4.13	Удаление операнда из программы	14
4.14	Просмотр ошибки в файле листинга	14
4.15	Создание файла	15
4.16	Первая программа самостоятельной работы	15
4.17	Запуск первой программы	17
4.18	Вторая программа самостоятельной работы	18
4.19	Проверка работы второй программы	20

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

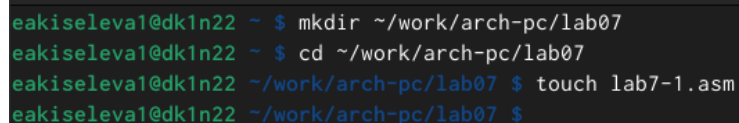
3 Теоретическое введение

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

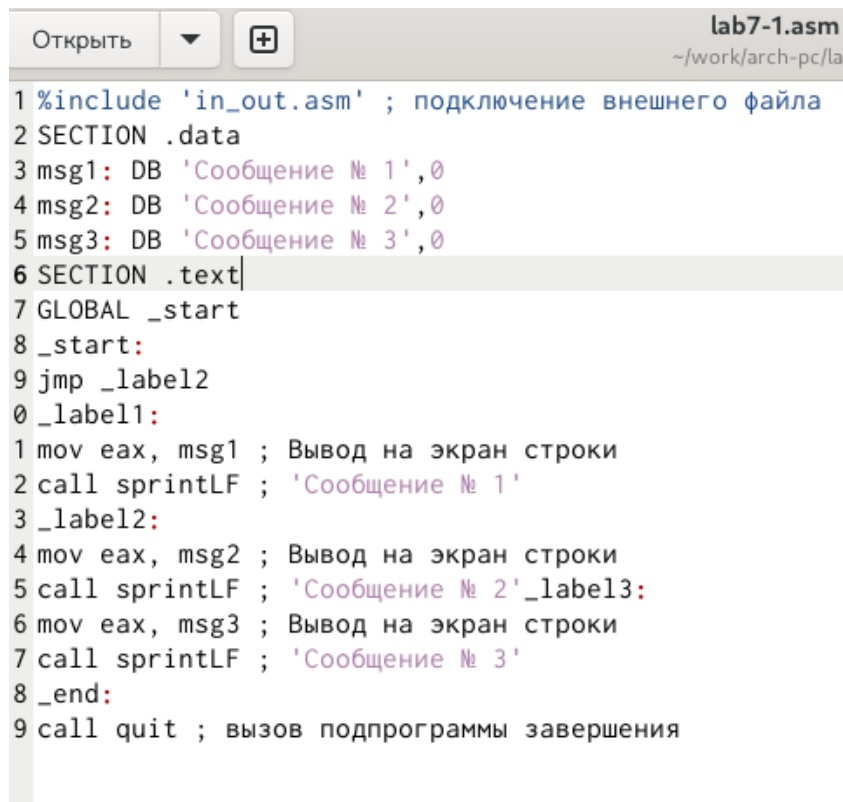
Создаю каталог для программ лабораторной работы №7 и файл lab7-1.asm, открываю его для редактирования (рис. 4.1).



```
eakiseleva1@dk1n22 ~ $ mkdir ~/work/arch-pc/lab07
eakiseleva1@dk1n22 ~ $ cd ~/work/arch-pc/lab07
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-1.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 4.1: Создание каталога и файла для программы

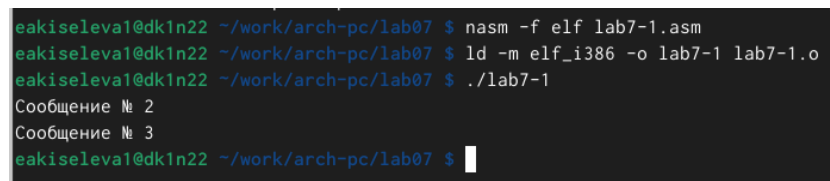
Копирую код из листинга в файл будущей программы (рис. 4.2).



```
Открыть  lab7-1.asm
~/work/arch-pc/la
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
0 _label1:
1 mov eax, msg1 ; Вывод на экран строки
2 call sprintf ; 'Сообщение № 1'
3 _label2:
4 mov eax, msg2 ; Вывод на экран строки
5 call sprintf ; 'Сообщение № 2' _label3:
6 mov eax, msg3 ; Вывод на экран строки
7 call sprintf ; 'Сообщение № 3'
8 _end:
9 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Написание программы

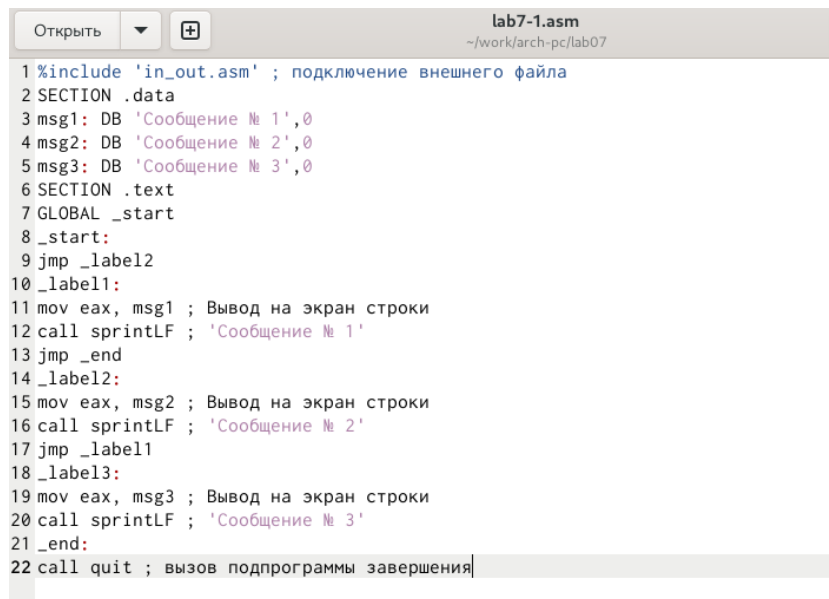
При запуске программы я убедилась в том, что безусловный переход действительно изменяет порядок выполнения инструкций (рис. 4.3).



```
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 4.3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4.4).

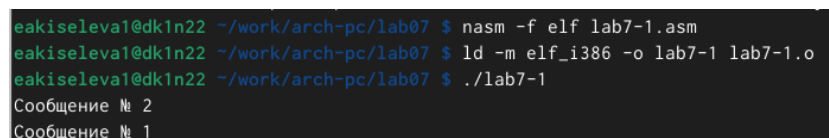


```
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение программы

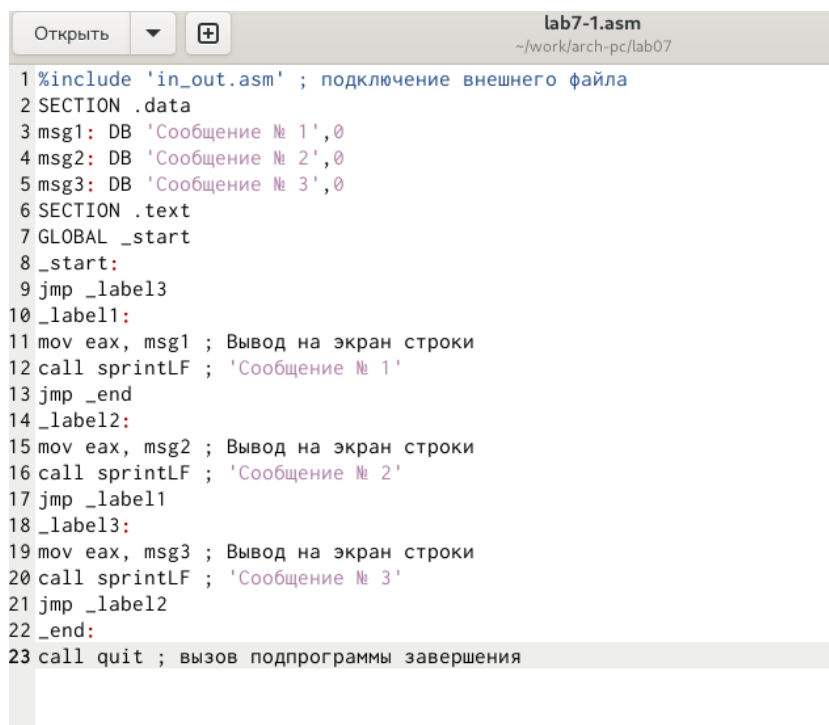
Запускаю программу и проверяю, что примененные изменения верны (рис. 4.5).



```
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 4.5: Запуск измененной программы

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

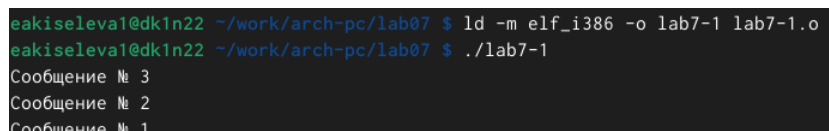


```
lab7-1.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Изменение программы

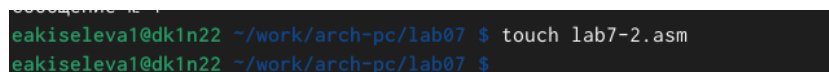
Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 4.7).



```
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Проверка изменений

Создаю новый рабочий файл lab7-2.asm (рис. 4.8).



```
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ touch lab7-2.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 4.8: Создание нового файла

Вставляю в созданный файл код из следующего листинга (рис. 4.9).

lab7-1.asm	lab7-2.asm
<pre> 1 %include 'in_out.asm' 2 section .data 3 msg1 db 'Введите B: ',0h 4 msg2 db "Наибольшее число: ",0h 5 A dd '20' 6 C dd '50' 7 section .bss 8 max resb 10 9 B resb 10 10 section .text 11 global _start 12 _start: 13 ; ----- Вывод сообщения 'Введите B: ' 14 mov eax,msg1 15 call sprint 16 ; ----- Ввод 'B' 17 mov ecx,B 18 mov edx,10 19 call sread 20 ; ----- Преобразование 'B' из символа в число 21 mov eax,B 22 call atoi ; Вызов подпрограммы перевода символа в число 23 mov [B],eax ; запись преобразованного числа в 'B' 24 ; ----- Записываем 'A' в переменную 'max' 25 mov ecx,[A] ; 'ecx = A' 26 mov [max],ecx ; 'max = A' 27 ; ----- Сравниваем 'A' и 'C' (как символы) 28 cmp ecx,[C] ; Сравниваем 'A' и 'C' 29 jg check_B ; если 'A>C', то переход на метку 'check_B', 30 mov ecx,[C] ; иначе 'ecx = C' 31 mov [max],ecx ; 'max = C' 32 ; ----- Преобразование 'max(A,C)' из символа в число 33 check_B: 34 mov eax,max 35 call atoi ; Вызов подпрограммы перевода символа в число 36 mov [max],eax ; запись преобразованного числа в 'max' 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа) 38 mov ecx,[max] 39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B' 40 jg fin ; если 'max(A,C)>B', то переход на 'fin', 41 mov ecx,[B] ; иначе 'ecx = B' 42 mov [max],ecx 43 ; ----- Вывод результата 44 fin: 45 mov eax, msg2 46 call sprint ; Вывод сообщения 'Наибольшее число: ' 47 mov ecx,[max] </pre>	

Рис. 4.9: Создание новой программы

Программа выводит значение переменной с максимальным значением, проверяя работу программы с разными входными данными (рис. 4.10).

```

eakiseleva1@dk1n22 ~ $ cd ~/work/arch-pc/lab07
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 12
Наибольшее число: 50
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 80
Наибольшее число: 80
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $

```

Рис. 4.10: 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 (рис. 4.11).

```
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $ gedit lab7-2.lst
eakiseleva1@dk1n22 ~/work/arch-pc/lab07 $
```

Рис. 4.11: Создание файла листинга

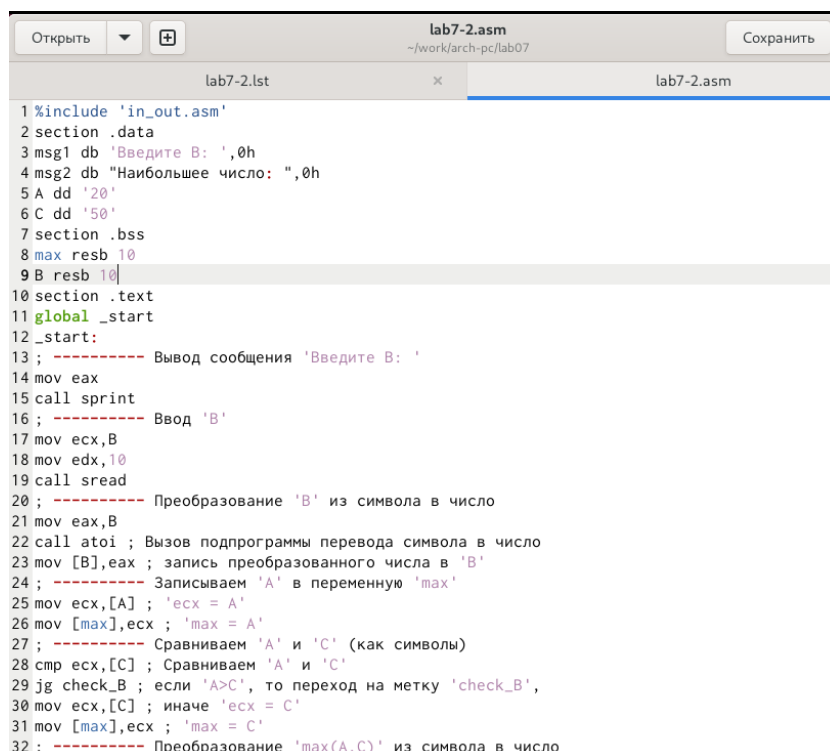
Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями (рис. 4.12).

lab7-2.lst			Открыть	+	Сохранить
~/work/arch-pc/lab07					
1	1	%include 'in_out.asm'			
2	1	<1> ;----- slen -----			
3	2	<1> ; Функция вычисления длины сообщения			
4	3	<1> slen:			
5	4 00000000 53	<1> push ebx			
6	5 00000001 89C3	<1> mov ebx, eax			
7	6	<1>			
8	7	<1> nextchar:			
9	8 00000003 803800	<1> cmp byte [eax], 0			
10	9 00000006 7403	<1> jz finished			
11	10 00000008 40	<1> inc eax			
12	11 00000009 EBF8	<1> jmp nextchar			
13	12	<1>			
14	13	<1> finished:			
15	14 0000000B 29D8	<1> sub eax, ebx			
16	15 0000000D 5B	<1> pop ebx			
17	16 0000000E C3	<1> ret			
18	17	<1>			
19	18	<1>			
20	19	<1> ;----- sprint -----			
21	20	<1> ; Функция печати сообщения			
22	21	<1> ; входные данные: mov eax, <message>			
23	22	<1> sprint:			
24	23 0000000F 52	<1> push edx			
25	24 00000010 51	<1> push ecx			
26	25 00000011 53	<1> push ebx			
27	26 00000012 50	<1> push eax			
28	27 00000013 E8E8FFFFFF	<1> call slen			
29	28	<1>			
30	29 00000018 89C2	<1> mov edx, eax			
31	30 0000001A 58	<1> pop eax			
32	31	<1>			
33	32 0000001B 89C1	<1> mov ecx, eax			
34	33 0000001D BB01000000	<1> mov ebx, 1			

Рис. 4.12: Файл листинга

Удаляю один операнд из случайной инструкции, чтобы проверить поведение

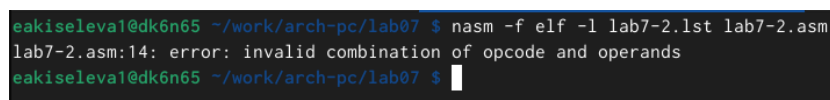
файла листинга в дальнейшем (рис. 4.13).



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
```

Рис. 4.13: 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. 4.14).



```
eakiseleva@dk6n65 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
eakiseleva@dk6n65 ~/work/arch-pc/lab07 $
```

Рис. 4.14: Просмотр ошибки в файле листинга

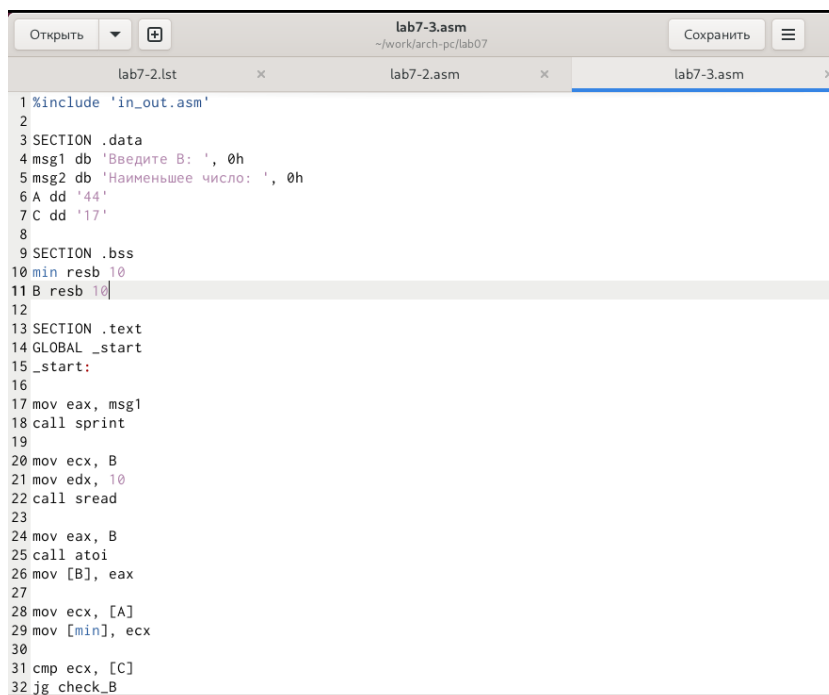
4.3 Задания для самостоятельной работы

Создаю файл lab7-3.asm для написания первого задания самостоятельной работы (рис. 4.15).

```
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ nasm -f elf -i lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ touch lab7-3.asm
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ gedit lab7-3.asm
```

Рис. 4.15: Создание файла

Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 4.16).



```
lab7-3.asm
~/work/arch-pc/lab07
Сохранить

lab7-2.lst x lab7-2.asm x lab7-3.asm x

1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите B: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '44'
7 C dd '17'
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
```

Рис. 4.16: 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 '44'
```

```
C dd '17'
```

```
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
```



```

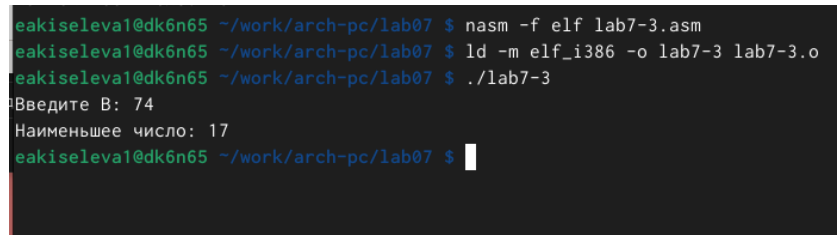
call atoi
mov [min], eax

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 работы первой программы (рис. 4.17).



```

eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $ ./lab7-3
Введите B: 74
Наименьшее число: 17
eakiseleva1@dk6n65 ~/work/arch-pc/lab07 $

```

Рис. 4.17: Запуск первой программы

Создаю файл lab7-4.asm и пишу программу, которая будет вычислять значение заданной функции согласно моему 16 варианту для введенных с клавиатуры переменных а и х (рис. 4.18).

```

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, 4
31 jl add_values
32 mov eax, edi
33 imul eax, esi
34 jmp print_result
35
36 add_values:
37 mov eax, edi
38 add eax, 4
39
40 print_result:
41 mov edi, eax
42 mov eax, res
43 call sprint
44 mov eax, edi
45 call iprintLF

```

Рис. 4.18: 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, 4
jl add_values
mov eax, edi
imul eax, esi
jmp print_result

```

```

add_values:
mov eax, edi
add eax, 4

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. 4.19).

```

eakiseleva1@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
eakiseleva1@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
eakiseleva1@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной х: 1
Введите значение переменной а: 1
Результат: 5
eakiseleva1@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-4
Введите значение переменной х: 7
Введите значение переменной а: 1
Результат: 7
eakiseleva1@dk3n55 ~/work/arch-pc/lab07 $

```

Рис. 4.19: 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