

# **Отчет по лабораторной работе №7**

**дисциплина: Архитектура компьютера**

Белоусова Елизавета Валентиновна

# Содержание

## **Список иллюстраций**

# Список таблиц

## 1 Цель работы

Цель данной лабораторной 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 файлы листинга
3. Выполнение заданий для самостоятельной 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.

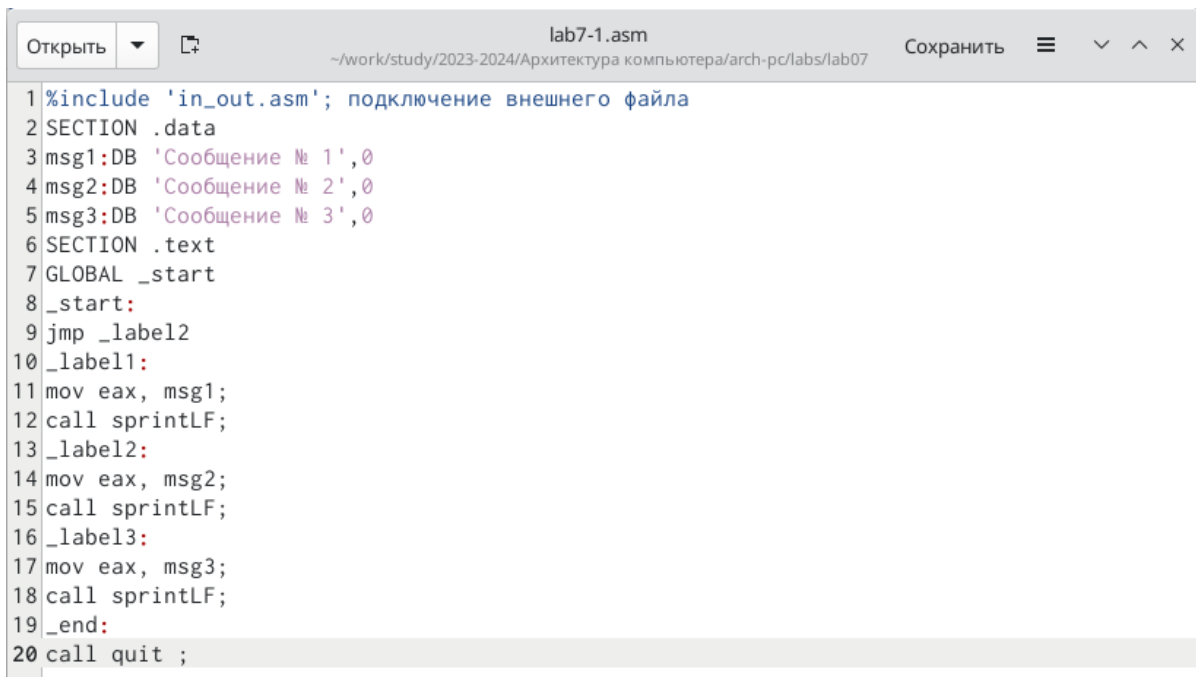
## 4 Выполнение лабораторной работы

Перехожу в каталог, созданный для файлов с программами для лабораторной работы №7 и, перейдя в него, создаю файл lab7-1.asm с помощью утилиты touch (рис. 1).

```
evbelousova@dk8n69 ~ $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab07
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-1.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 1: Создание файла

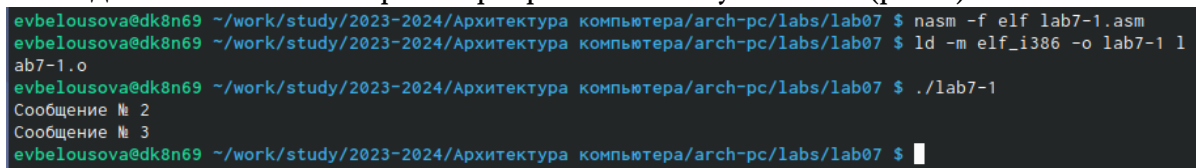
Открываю файл lab7-1.asm и ввожу в него текст программы из листинга 7.1, перед этим скопировав файл in\_out.asm в каталог lab07 (рис. 2)



```
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;
12 call sprintfLF;
13 _label2:
14 mov eax, msg2;
15 call sprintfLF;
16 _label3:
17 mov eax, msg3;
18 call sprintfLF;
19 _end:
20 call quit ;
```

Рис. 2: Редактирование файла


Создаю исполняемый файл программы и запускаю его (рис 3).



```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 3: Запуск исполняемого файла

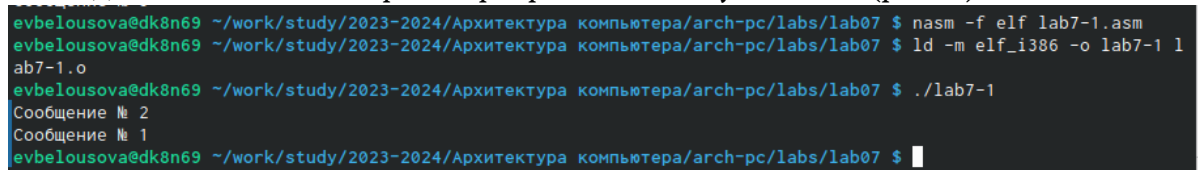
Открываю файл lab7-1.asm, изменяя текст программы так, чтобы выводила сначала “Сообщение № 2”, потом “Сообщение № 1” и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляю инструкцию jmp с меткой \_label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавляю инструкцию jmp с меткой \_end (т.е. переход к инструкции call quit).(рис. 4).



```
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;
12 call sprintfLF;
13 jmp _end
14 _label2:
15 mov eax, msg2;
16 call sprintfLF;
17 jmp _label1
18 _label3:
19 mov eax, msg3;
20 call sprintfLF;
21 _end:
22 call quit ;
```

Рис. 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 5).



```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 l
ab7-1.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 5: Запуск исполняемого файла

Снова открываю файл и редактирую текст программы таким образом, чтобы сначала выводилось “Сообщение № 3”, затем “Сообщение № 2”, “Сообщение № 1” и программа завершала работу (рис. 6).

```
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;
12 call sprintf;
13 jmp _end
14 _label2:
15 mov eax, msg2;
16 call sprintf;
17 jmp _label1
18 _label3:
19 mov eax, msg3;
20 call sprintf;
21 jmp _label2
22 _end:
23 call quit ;
```

Рис. 6: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 7).

```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

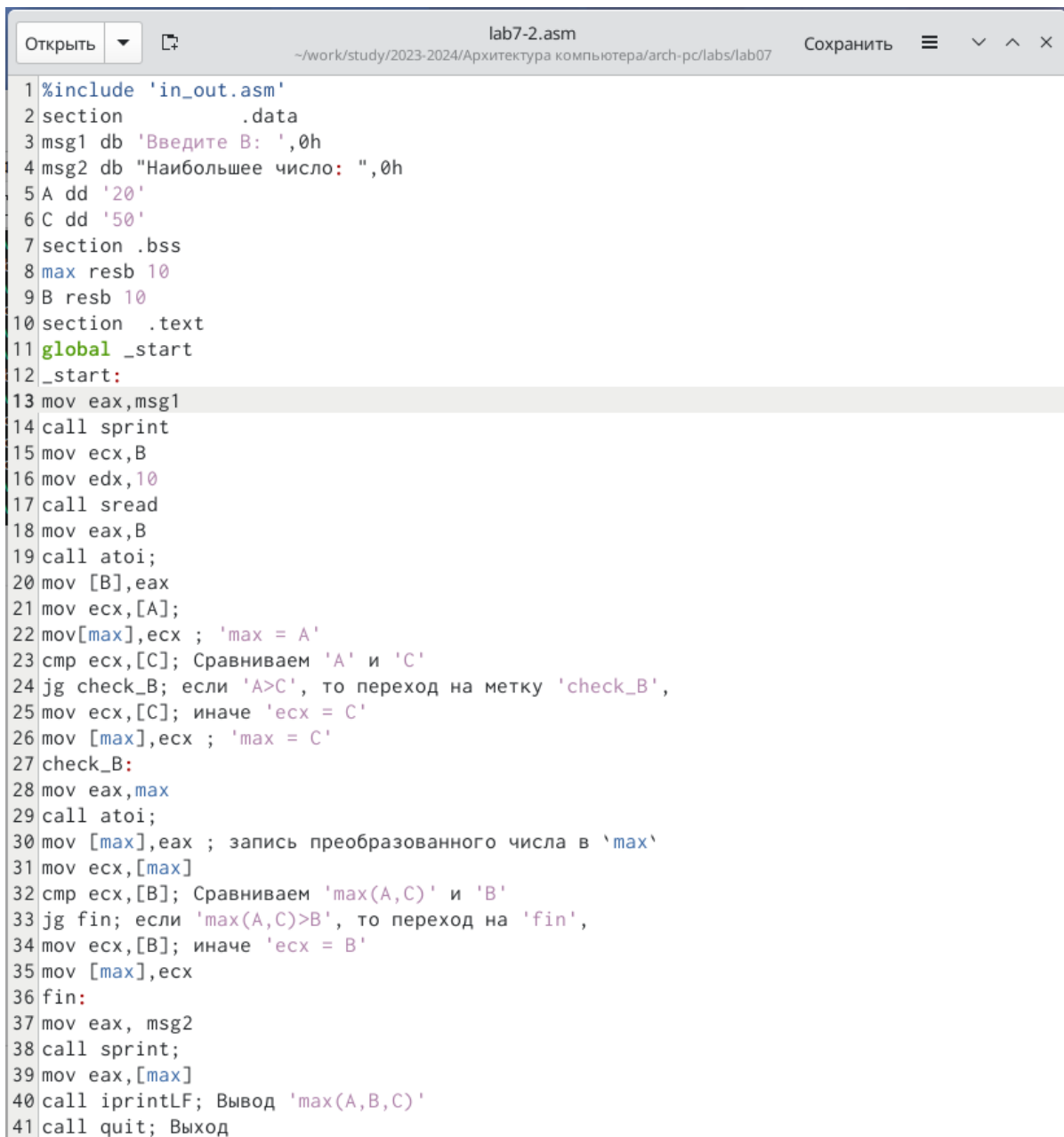
Рис. 7: Запуск исполняемого файла

Создаю новый файл lab7-2.asm с помощью утилиты touch (рис. 8).

```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-2.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 8: Создание файла

Ввожу в файл текст другой программы, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С (рис. 9).



```
lab7-2.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07
Открыть Сохранить

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 mov eax,msg1
14 call sprint
15 mov ecx,B
16 mov edx,10
17 call sread
18 mov eax,B
19 call atoi;
20 mov [B],eax
21 mov ecx,[A];
22 mov[max],ecx ; 'max = A'
23 cmp ecx,[C]; Сравниваем 'A' и 'C'
24 jg check_B; если 'A>C', то переход на метку 'check_B',
25 mov ecx,[C]; иначе 'ecx = C'
26 mov [max],ecx ; 'max = C'
27 check_B:
28 mov eax,max
29 call atoi;
30 mov [max],eax ; запись преобразованного числа в 'max'
31 mov ecx,[max]
32 cmp ecx,[B]; Сравниваем 'max(A,C)' и 'B'
33 jg fin; если 'max(A,C)>B', то переход на 'fin',
34 mov ecx,[B]; иначе 'ecx = B'
35 mov [max],ecx
36 fin:
37 mov eax, msg2
38 call sprint;
39 mov eax,[max]
40 call iprintLF; Вывод 'max(A,B,C)'
41 call quit; Выход
```

Рис. 9: Редактирование файла

Создаю и запускаю исполняемый файл lab7-2 и проверяю корректность работы программы при разных значениях B (рис. 10).



```

evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-2.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-2 l
ab7-2.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-2
Введите B: 10
Наибольшее число: 50
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-2.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-2 l
ab7-2.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-2
Введите B: 55
Наибольшее число: 55
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ █

```

Рис. 10: Запуск исполняемого файла

Создаю файл листинга программы из файла lab7-2.asm командой `nasm -f elf -l lab7-2.lst lab7-2.asm` (рис. 11).

```

evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf -l lab7-2.lst
lab7-2.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ █

```

Рис. 11: Создание файла

Открываю файл листинга lab7-2.lst с помощью текстового редактора gedit (рис. 12). Внимательно ознакамливаюсь с форматом и содержимым файла. Рассмотрим некоторые строки листинга и попробуем объяснить их. Я возьму строки 14-16. 14-16 - это номера строк файла листинга, которые могут не соответствовать номеру строки в файле с исходным текстом программы. Далее “0000000B 29D8”, “0000000D 5B” и “0000000E C3” в каждой из строк 14-16 соответственно обозначают адрес и машинный код, а все, что стоит в этих строках дальше после <1> содержит исходный текст программы, т.е. исходную программу вместе с комментариями.

```

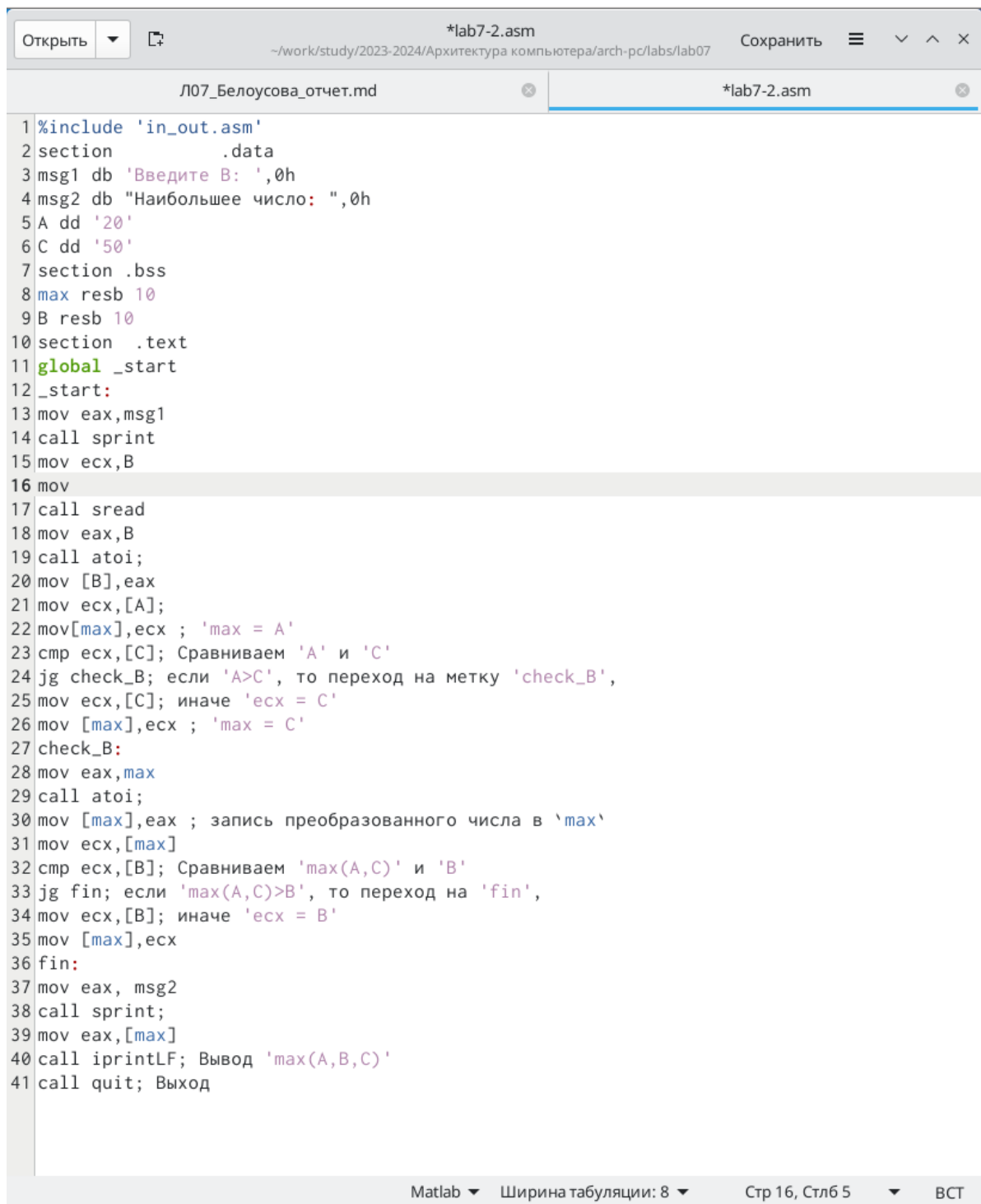
Открыть  lab7-2.lst  Сохранить
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07

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 E8E8FFFF <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
35    34 00000022 B804000000 <1>      mov       eax, 4
36    35 00000027 CD80     <1>      int       80h
37    36      <1>
38    37 00000029 5B      <1>      pop       ebx
39    38 0000002A 59      <1>      pop       ecx
40    39 0000002B 5A      <1>      pop       edx
41    40 0000002C C3      <1>      ret
42    41      <1>
43    42      <1>
44    43      <1> ;----- sprintLF
-----
Текст  Ширина табуляции: 8  Стр 1, Стлб 1  ВСТ

```

Рис. 12: Файл листинга

Открываю файл с программой lab7-2.asm и в одной из инструкций с двумя операндами удаляю один операнд (рис. 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 mov eax,msg1
14 call sprint
15 mov ecx,B
16 mov
17 call sread
18 mov eax,B
19 call atoi;
20 mov [B],eax
21 mov ecx,[A];
22 mov[max],ecx ; 'max = A'
23 cmp ecx,[C]; Сравниваем 'A' и 'C'
24 jg check_B; если 'A>C', то переход на метку 'check_B',
25 mov ecx,[C]; иначе 'ecx = C'
26 mov [max],ecx ; 'max = C'
27 check_B:
28 mov eax,max
29 call atoi;
30 mov [max],eax ; запись преобразованного числа в 'max'
31 mov ecx,[max]
32 cmp ecx,[B]; Сравниваем 'max(A,C)' и 'B'
33 jg fin; если 'max(A,C)>B', то переход на 'fin',
34 mov ecx,[B]; иначе 'ecx = B'
35 mov [max],ecx
36 fin:
37 mov eax, msg2
38 call sprint;
39 mov eax,[max]
40 call iprintLF; Вывод 'max(A,B,C)'
41 call quit; Выход
```

Рис. 13: Редактирование файла

Выполняю трансляцию с получением файла листинга и вижу, что система выдает ошибку (рис. 14)

```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf -l lab7-2.lst
lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

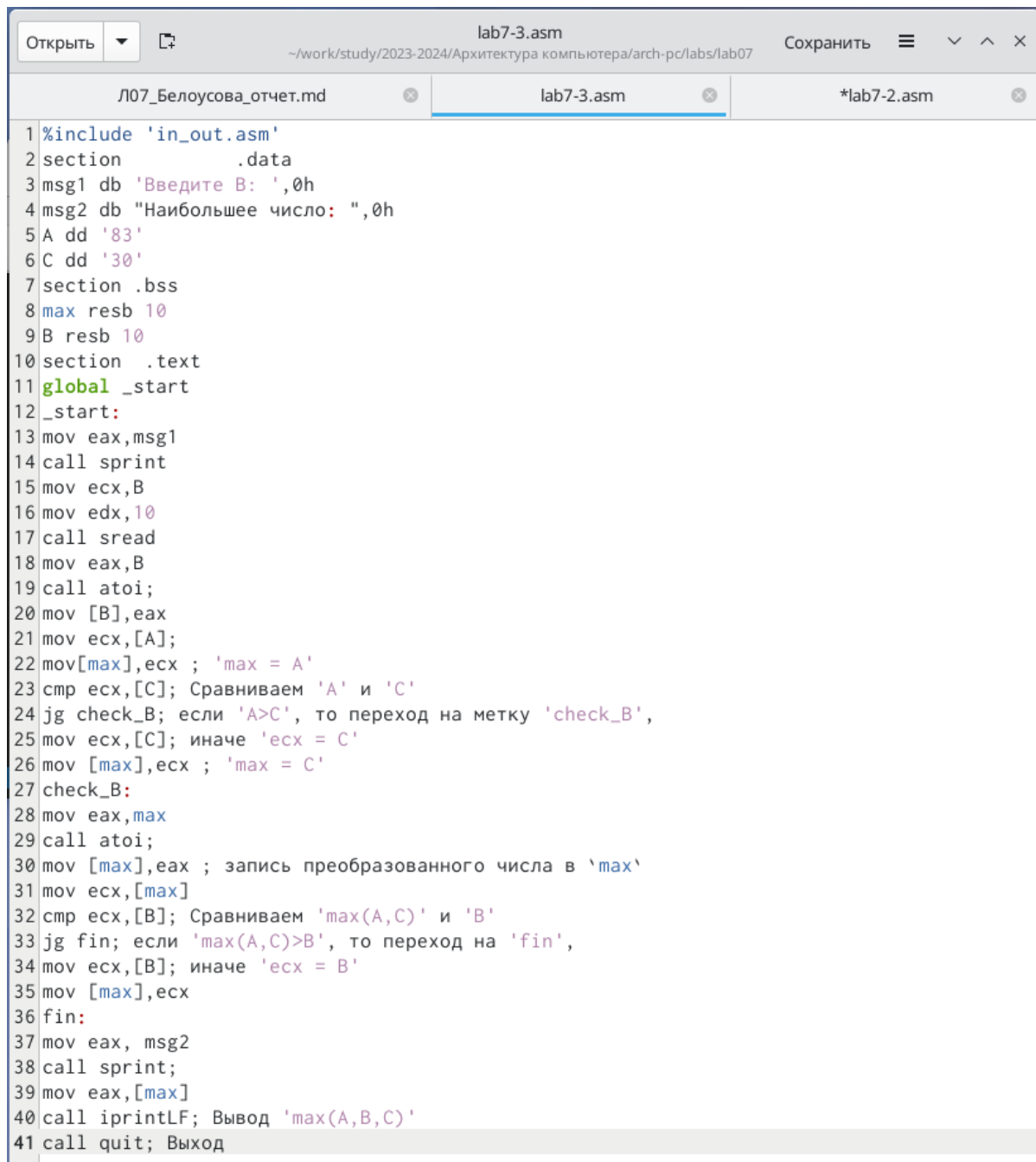
Рис. 14: Запуск исполняемого файла

Для выполнения заданий для самостоятельной работы создаю файл lab07-3.asm с помощью утилиты touch (рис. 15)

```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-3.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 15: Создание файла

Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных беру из таблицы 7.5 в соответствии со своим вариантом (18) (рис. 16)



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '83'
6 C dd '30'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 mov eax,msg1
14 call sprint
15 mov ecx,B
16 mov edx,10
17 call sread
18 mov eax,B
19 call atoi;
20 mov [B],eax
21 mov ecx,[A];
22 mov[max],ecx ; 'max = A'
23 cmp ecx,[C]; Сравниваем 'A' и 'C'
24 jg check_B; если 'A>C', то переход на метку 'check_B',
25 mov ecx,[C]; иначе 'ecx = C'
26 mov [max],ecx ; 'max = C'
27 check_B:
28 mov eax,max
29 call atoi;
30 mov [max],eax ; запись преобразованного числа в 'max'
31 mov ecx,[max]
32 cmp ecx,[B]; Сравниваем 'max(A,C)' и 'B'
33 jg fin; если 'max(A,C)>B', то переход на 'fin',
34 mov ecx,[B]; иначе 'ecx = B'
35 mov [max],ecx
36 fin:
37 mov eax, msg2
38 call sprint;
39 mov eax,[max]
40 call iprintLF; Вывод 'max(A,B,C)'
41 call quit; Выход
```

Рис. 16: Написание программы

Создаю исполняемый файл и проверяю работу программы. (рис. 17) Программа работает корректно.

```
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-3.asm
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-3
Введите В: 73
Наибольшее число: 83
evbelousova@dk8n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 17: Запуск исполняемого файла

## 5 Выводы

При выполнении данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.

## 6 Список литературы

Архитектура ЭВМ