

Отчет по лабораторной работе №7

дисциплина: Архитектура компьютера

Маньковская Дарья Станиславовна

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

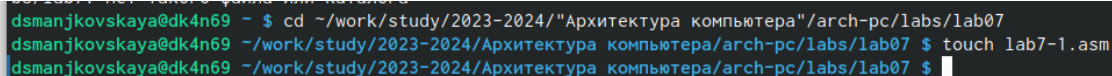
1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной работы

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

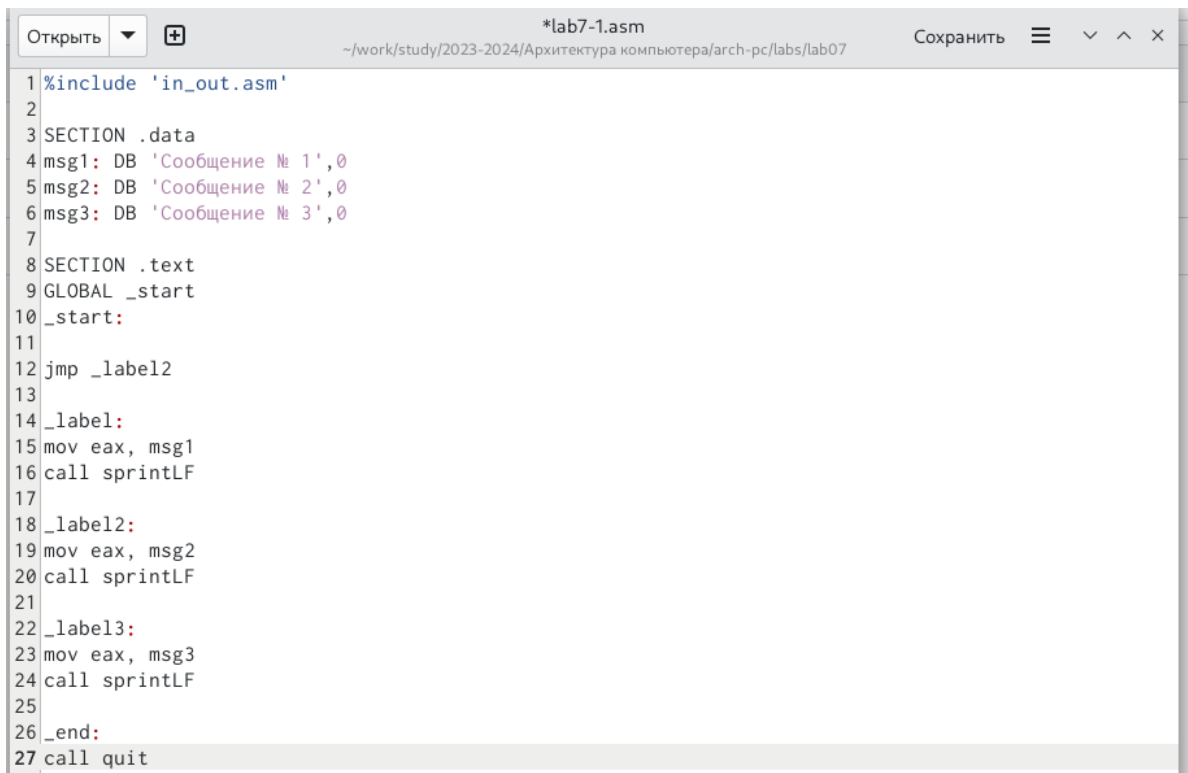
Перехожу в каталог, созданный для файлов с программами для лабораторной работы №7 и, перейдя в него, создаю файл lab7-1.asm с помощью утилиты touch (рис. 1).



```
dsmanjkovskaya@dk4n69 ~ $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab07
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-1.asm
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 1: Создание файла

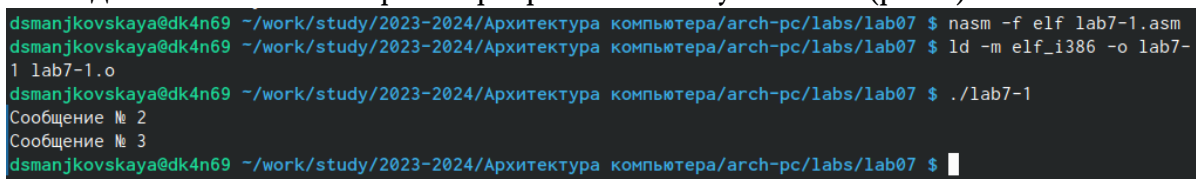
Открываю файл lab7-1.asm и ввожу в него текст программы из листинга 7.1, перед этим скопировав файл in_out.asm в каталог lab07 (рис. 2)



```
1 %include 'in_out.asm'
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 sprintLF
17
18 _label2:
19 mov eax, msg2
20 call sprintLF
21
22 _label3:
23 mov eax, msg3
24 call sprintLF
25
26 _end:
27 call quit
```

Рис. 2: Редактирование файла

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



```
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
dsmanjkovskaya@dk4n69 ~/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'
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 sprintLF
17 jmp _end
18
19 _label2:
20 mov eax, msg2
21 call sprintLF
22 jmp _label1
23
24 _label3:
25 mov eax, msg3
26 call sprintLF
27
28 _end:
29 call quit
```

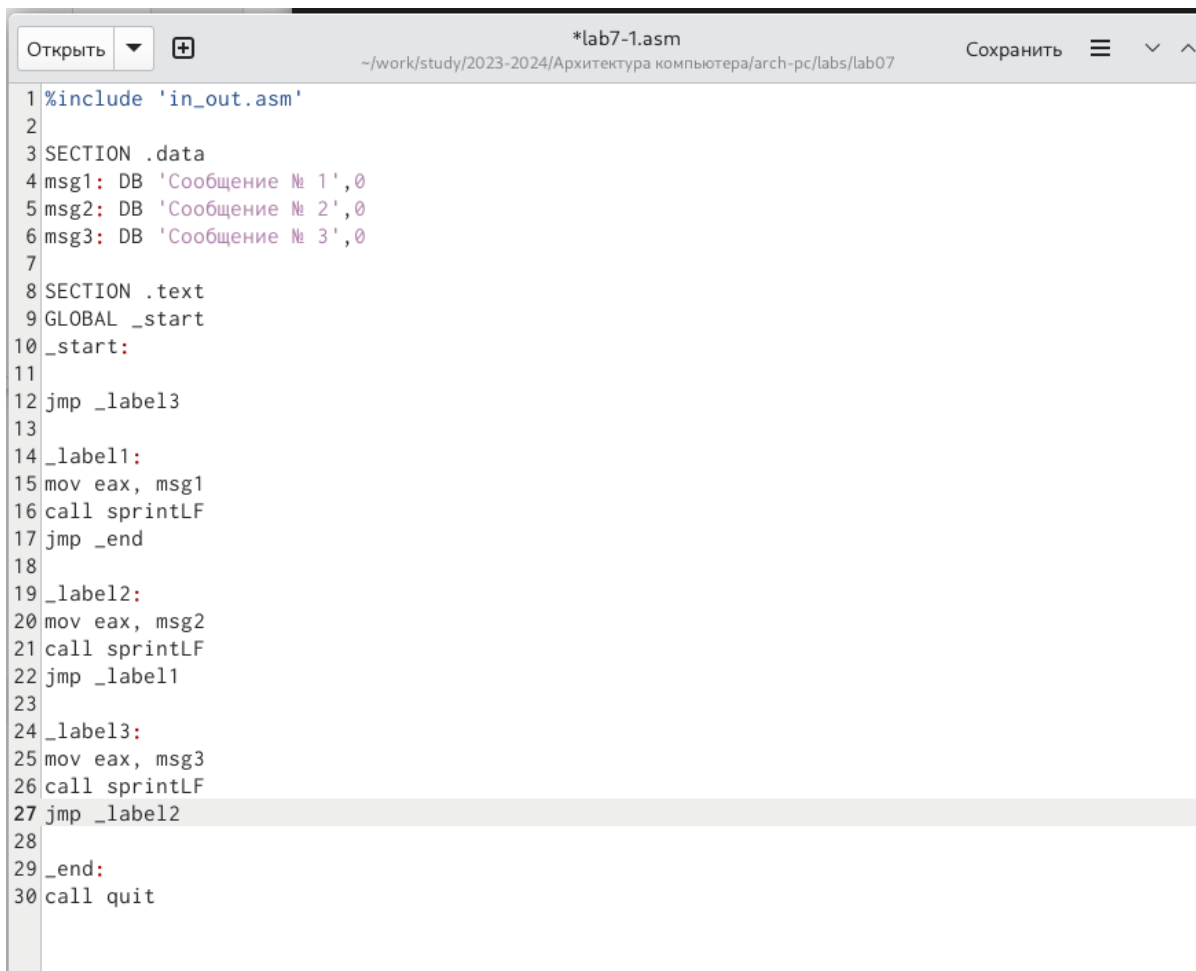
Рис. 4: Редактирование файла

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

```
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 5: Запуск исполняемого файла

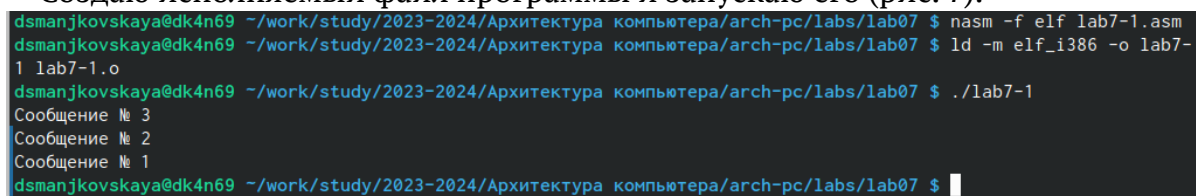
Снова открываю файл и редактирую текст программы таким образом, чтобы сначала выводилось “Сообщение № 3”, затем “Сообщение № 2”, “Сообщение № 1” и программа завершала работу (рис. 6).



```
Открыть + *lab7-1.asm ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 Сохранить ≡ ▾ ^
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. 6: Редактирование файла

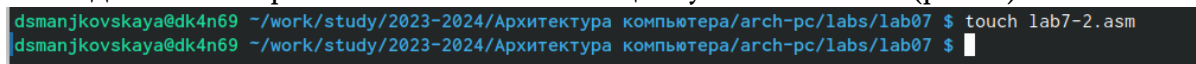
Создаю исполняемый файл программы и запускаю его (рис. 7).



```
dsmanjovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-1.asm
dsmanjovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
dsmanjovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dsmanjovskaya@dk4n69 ~/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).



```
dsmanjovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-2.asm
dsmanjovskaya@dk4n69 ~/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).

```

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 eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход

```

Рис. 9: Редактирование файла

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

```

dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-2.asm
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-2
Введите B: 55
Наибольшее число: 55
dsmanjkovskaya@dk4n69 ~/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).

```

dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
dsmanjkovskaya@dk4n69 ~/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 ознакомливаюсь 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.

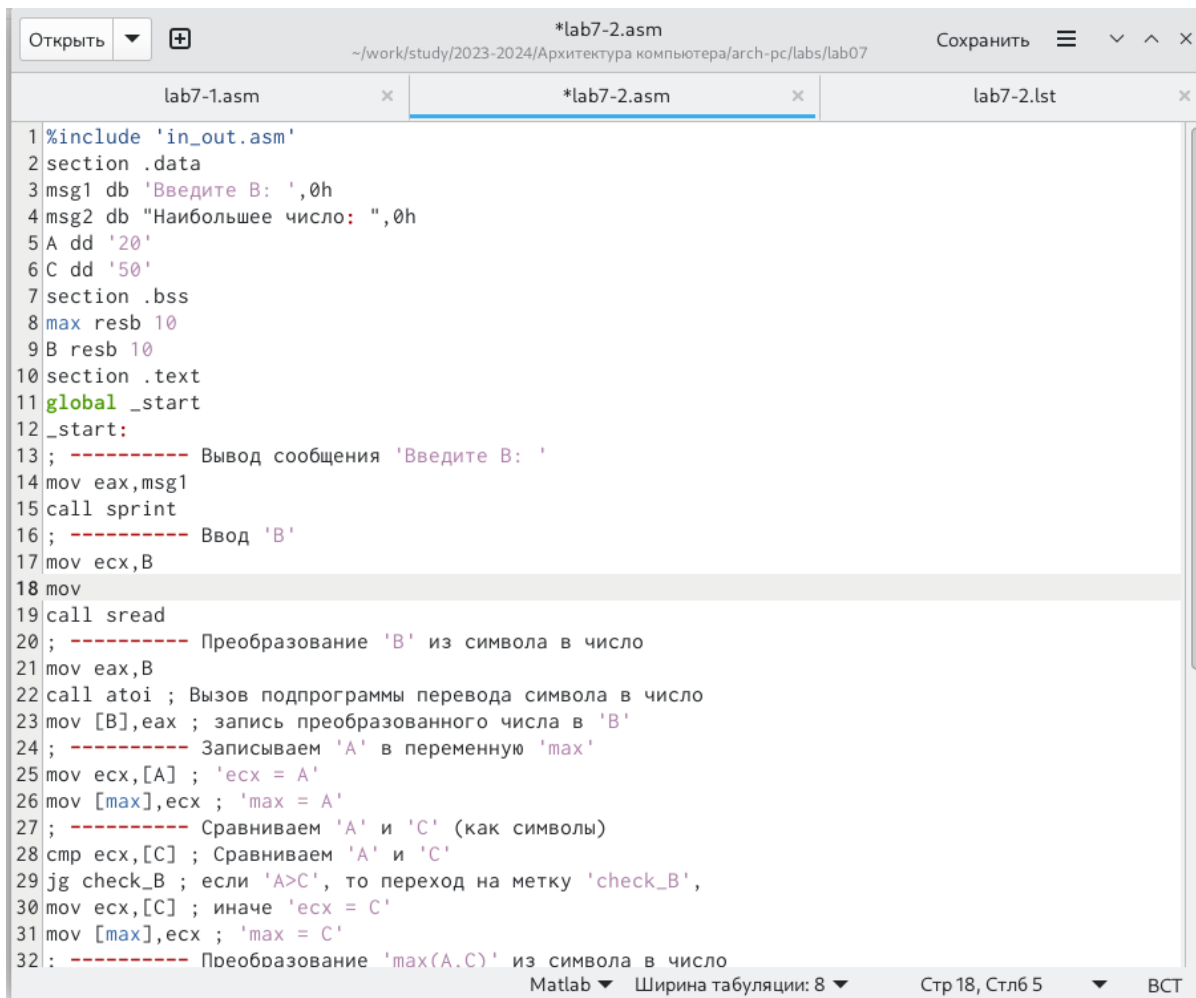
```

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>

```

Рис. 12: Файл листинга

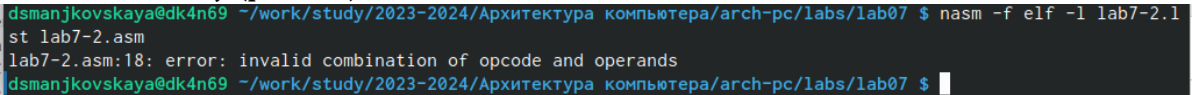
Открываю файл с программой lab7-2.asm и в одной из инструкций с двумя операндами удаляю один операнд (рис. 13)



```
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 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov
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. 13: Редактирование файла

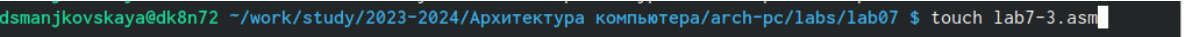
Выполняю трансляцию с получением файла листинга и вижу, что система выдает ошибку (рис. 14)



```
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf -l lab7-2.1
st lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
dsmanjkovskaya@dk4n69 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $
```

Рис. 14: Запуск исполняемого файла

Для выполнения задания 1 самостоятельной работы создаю файл lab07-3.asm с помощью утилиты touch (рис. 15)



```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-3.asm
```

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 (№19) (рис. 16)

Рис. 16: Написание программы

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

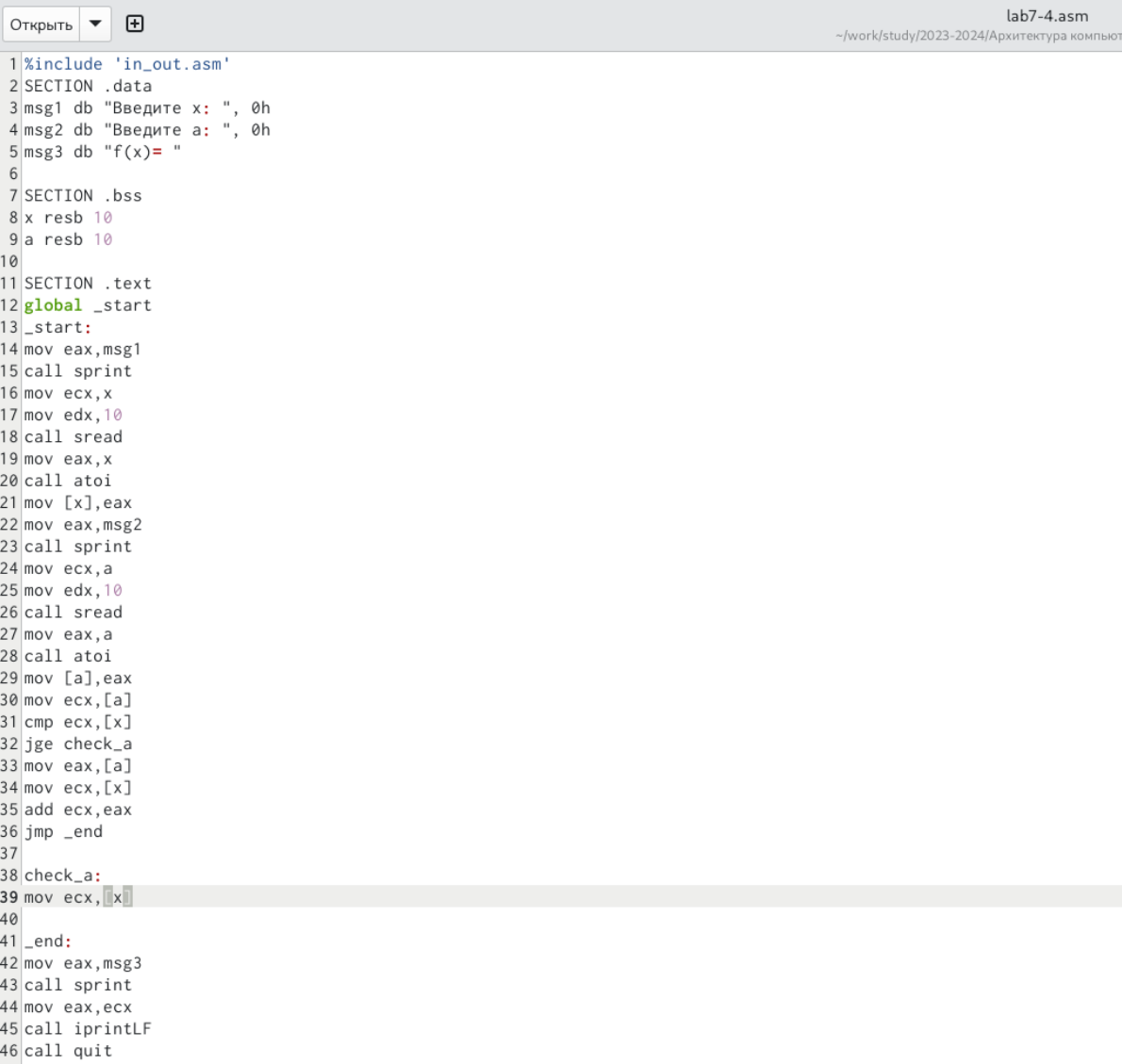
Рис. 17: Запуск исполняемого файла

Для выполнения задания 2 самостоятельной работы создаю файл lab07-4.asm с помощью утилиты touch (рис. 18)

```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ touch lab7-4.asm
```

Рис. 18: Создание файла

Пишу программу для расчёта значения функции для введённых с клавиатуры а и х (рис. 19).



```
lab7-4.asm
~/work/study/2023-2024/Архитектура компьют

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db "Введите x: ", 0h
4 msg2 db "Введите a: ", 0h
5 msg3 db "f(x)= "
6
7 SECTION .bss
8 x resb 10
9 a resb 10
10
11 SECTION .text
12 global _start
13 _start:
14 mov eax,msg1
15 call sprint
16 mov ecx,x
17 mov edx,10
18 call sread
19 mov eax,x
20 call atoi
21 mov [x],eax
22 mov eax,msg2
23 call sprint
24 mov ecx,a
25 mov edx,10
26 call sread
27 mov eax,a
28 call atoi
29 mov [a],eax
30 mov ecx,[a]
31 cmp ecx,[x]
32 jge check_a
33 mov eax,[a]
34 mov ecx,[x]
35 add ecx,eax
36 jmp _end
37
38 check_a:
39 mov ecx,[x]
40
41 _end:
42 mov eax,msg3
43 call sprint
44 mov eax,ecx
45 call iprintLF
46 call quit
```

Рис. 19: Написание программы

Создаю исполняемый файл и проверяю работу программы (рис. 20). Програм-

ма работает корректно. (Мой вариант № 19)

```
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ nasm -f elf lab7-4.asm
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-4
Введите x: 4
Введите a: 5
f(x)= 4
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
dsmanjkovskaya@dk8n72 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07 $ ./lab7-4
Введите x: 3
Введите a: 2
f(x)= 5
```

Рис. 20: 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