

Отчёт по лабораторной работе №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений.**

Майоров Дмитрий Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Изучение структуры файлы листинга	13
5	Задание для самостоятельной работы (Вариант 20)	19
6	Выводы	24
	Список литературы	25

Список иллюстраций

3.1	Создаем каталог лабораторной работы №7 и файл в нем	7
3.2	Открываем файл и заполняем его в соответствии с листингом	8
3.3	Создаем исполнительный файл и запускаем его	8
3.4	Снова открываем файл и редактируем его в соответствии с листингом	9
3.5	Создаем исполнительный файл и запускаем его	9
3.6	Снова открываем файл и редактируем его, чтобы произошел нуж- ный вывод	10
3.7	Создаем исполнительный файл и запускаем его. Проверяем, пра- вильный ли вывод	10
3.8	Создаем новый файл	10
3.9	Открываем файл и заполняем его в соответствии с листингом	11
3.10	Создаем исполнительный файл и запускаем его. Вводим разные значения В	12
4.1	Создаем файл листинга для программы lab7-2.asm	13
4.2	Открываем файл листинга и изучаем его	14
4.3	Открываем файл и удаляем один операнд	16
4.4	Транслируем файл	17
4.5	Открываем файл с ошибкой и изучаем его	18
5.1	Создаем новый файл	19
5.2	Открываем файл и пишем программу	20
5.3	Транслируем файл и смотрим на его работу	21
5.4	Создаем новый файл	21
5.5	Открываем файл и пишем программу	22
5.6	Транслируем файл и проверяем его работу при x=1 и a=2	23
5.7	Транслируем файл и проверяем его работу при x=2 и a=1	23

Список таблиц

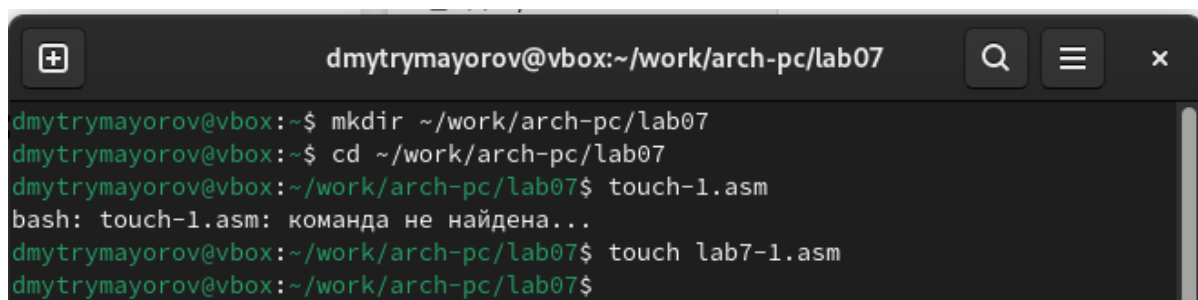
1 Цель работы

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

2 Задание

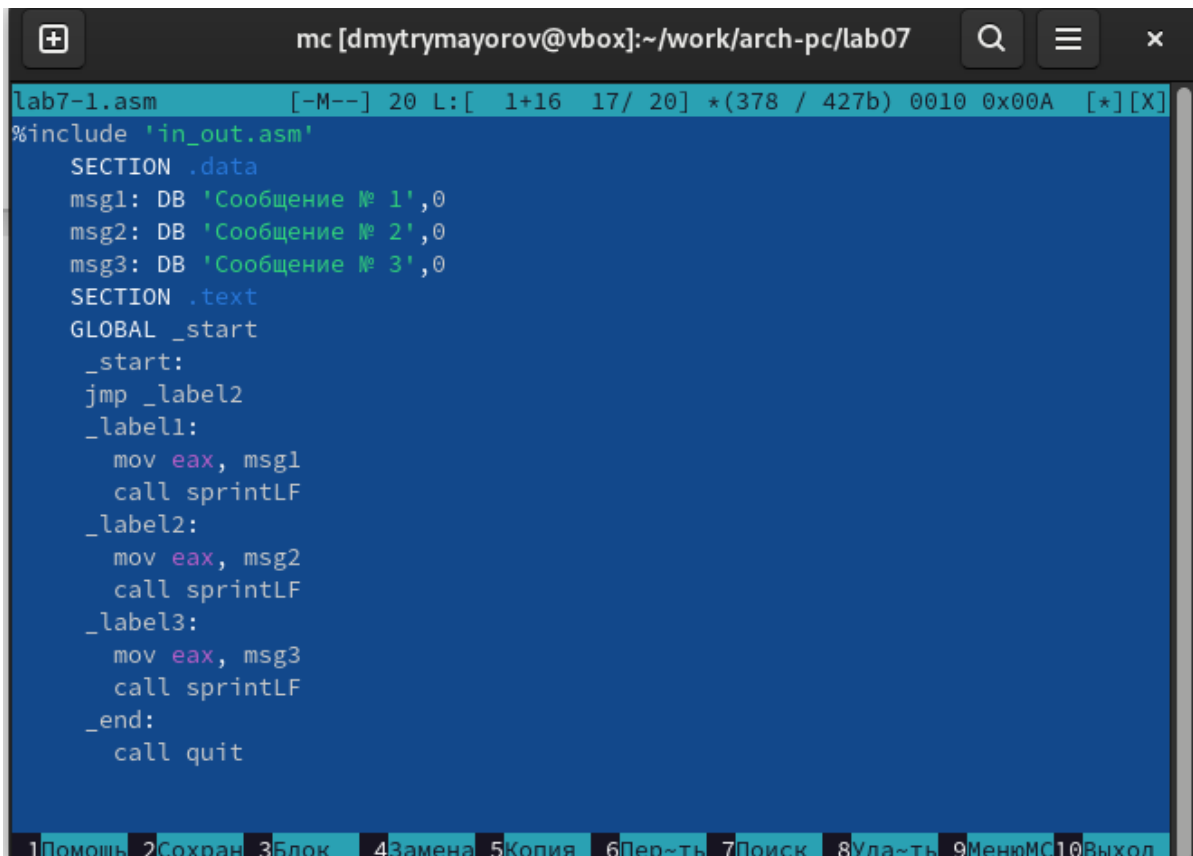
Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

3 Выполнение лабораторной работы



```
dmytrymayorov@vbox:~/work/arch-pc/lab07
dmytrymayorov@vbox:~$ mkdir ~/work/arch-pc/lab07
dmytrymayorov@vbox:~$ cd ~/work/arch-pc/lab07
dmytrymayorov@vbox:~/work/arch-pc/lab07$ touch-1.asm
bash: touch-1.asm: команда не найдена...
dmytrymayorov@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

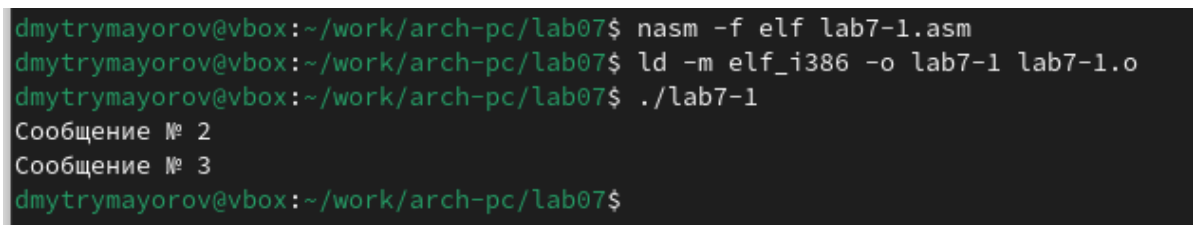
Рис. 3.1: Создаем каталог лабораторной работы №7 и файл в нем



```
lab7-1.asm [-M--] 20 L: [ 1+16 17/ 20] *(378 / 427b) 0010 0x00A [*][X]
#include 'in_out.asm'
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
call sprintLF
_label2:
mov eax, msg2
call sprintLF
_label3:
mov eax, msg3
call sprintLF
_end:
call quit
```

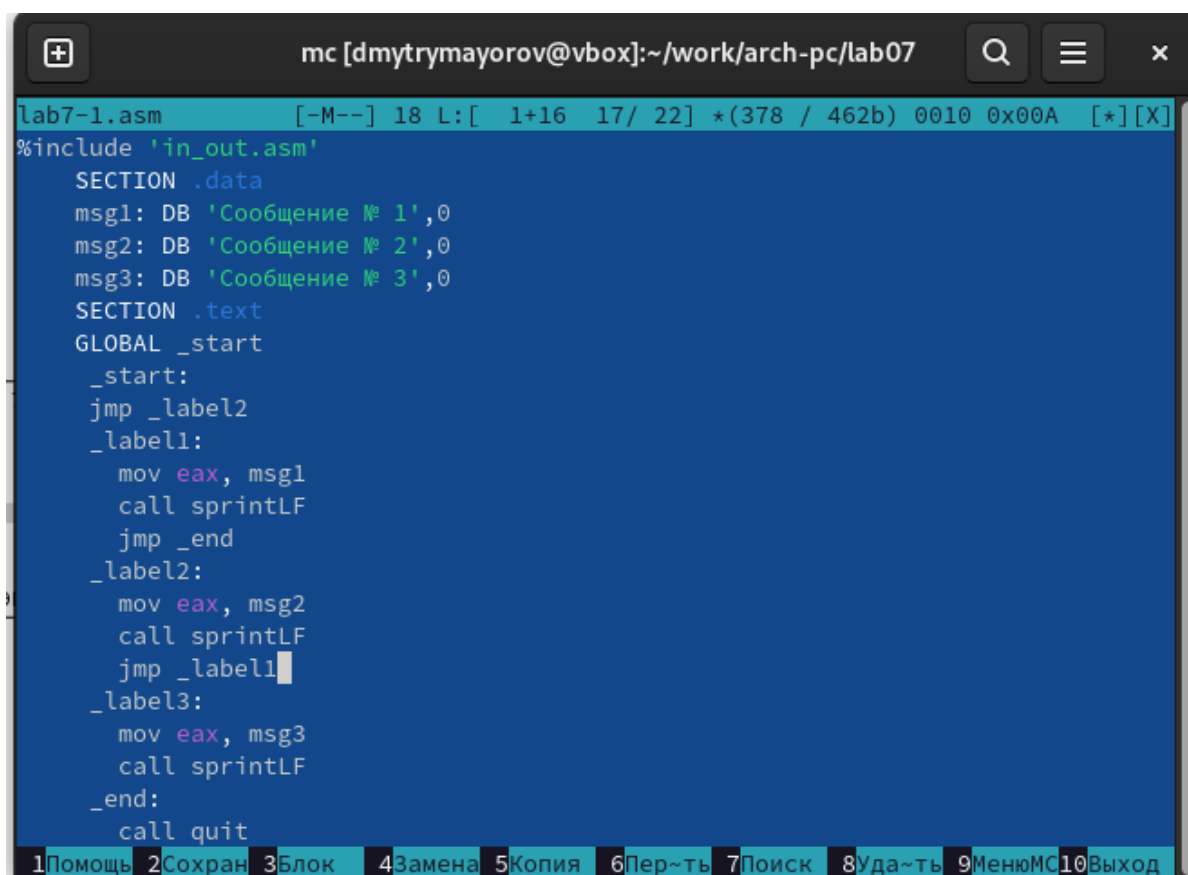
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход

Рис. 3.2: Открываем файл и заполняем его в соответствии с листингом



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

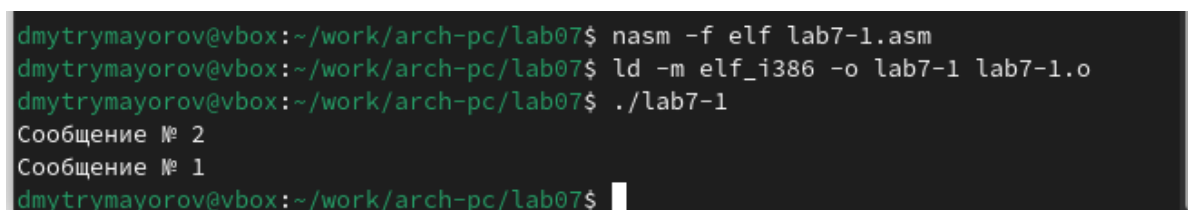
Рис. 3.3: Создаем исполняемый файл и запускаем его



```
lab7-1.asm [-M--] 18 L:[ 1+16 17/ 22] *(378 / 462b) 0010 0x00A [*][X]
#include 'in_out.asm'
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
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

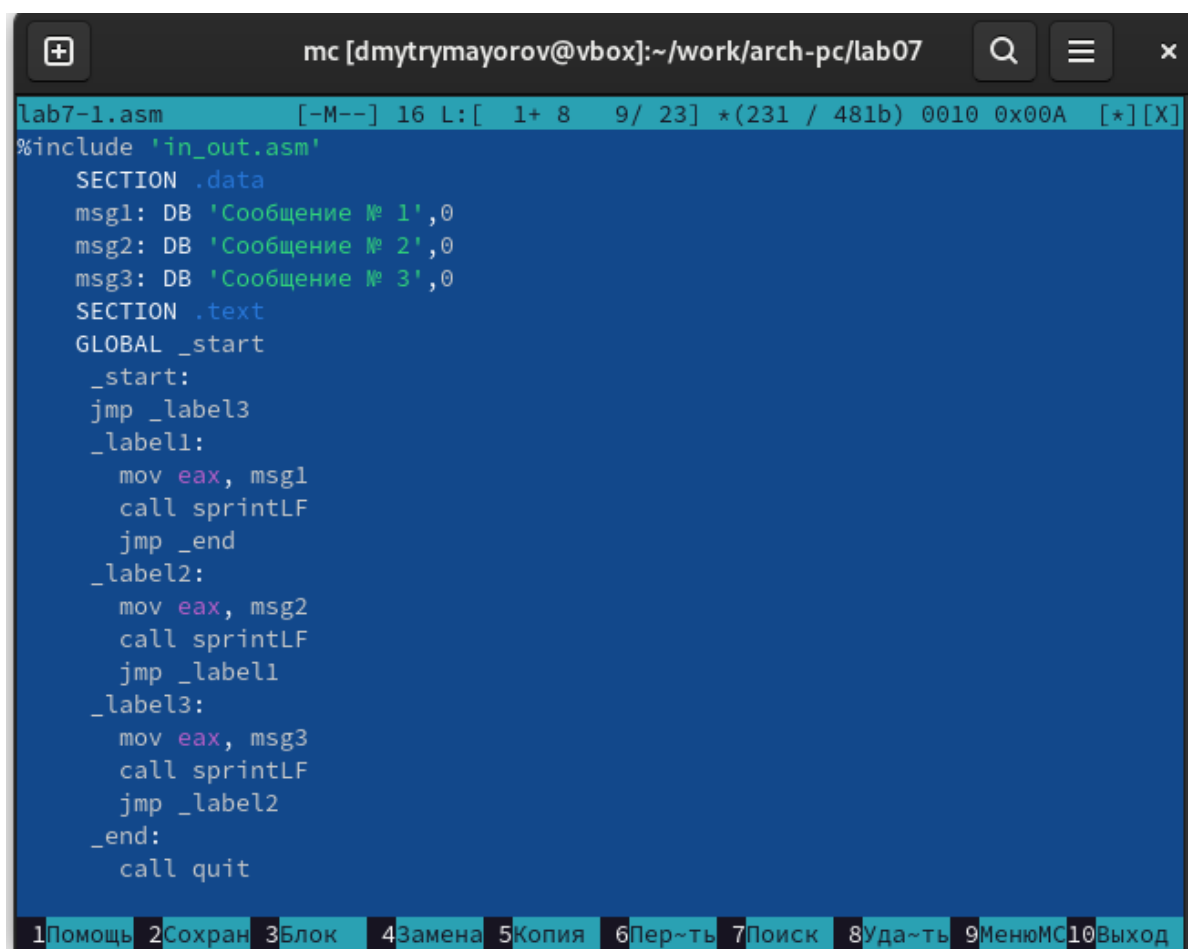
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 3.4: Снова открываем файл и редактируем его в соответствии с листингом



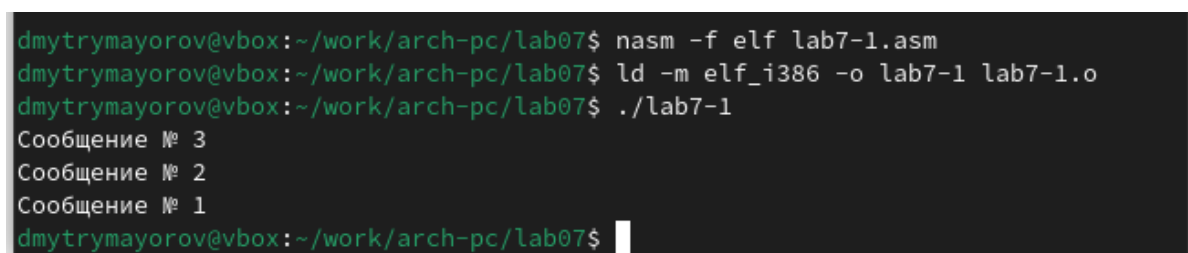
```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 3.5: Создаем исполняемый файл и запускаем его



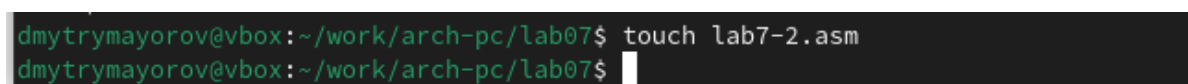
```
lab7-1.asm [-M--] 16 L: [ 1+ 8 9/ 23] *(231 / 481b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1
call sprintf
jmp _end
_label2:
mov eax, msg2
call sprintf
jmp _label1
_label3:
mov eax, msg3
call sprintf
jmp _label2
_end:
call quit
```

Рис. 3.6: Снова открываем файл и редактируем его, чтобы произошел нужный вывод



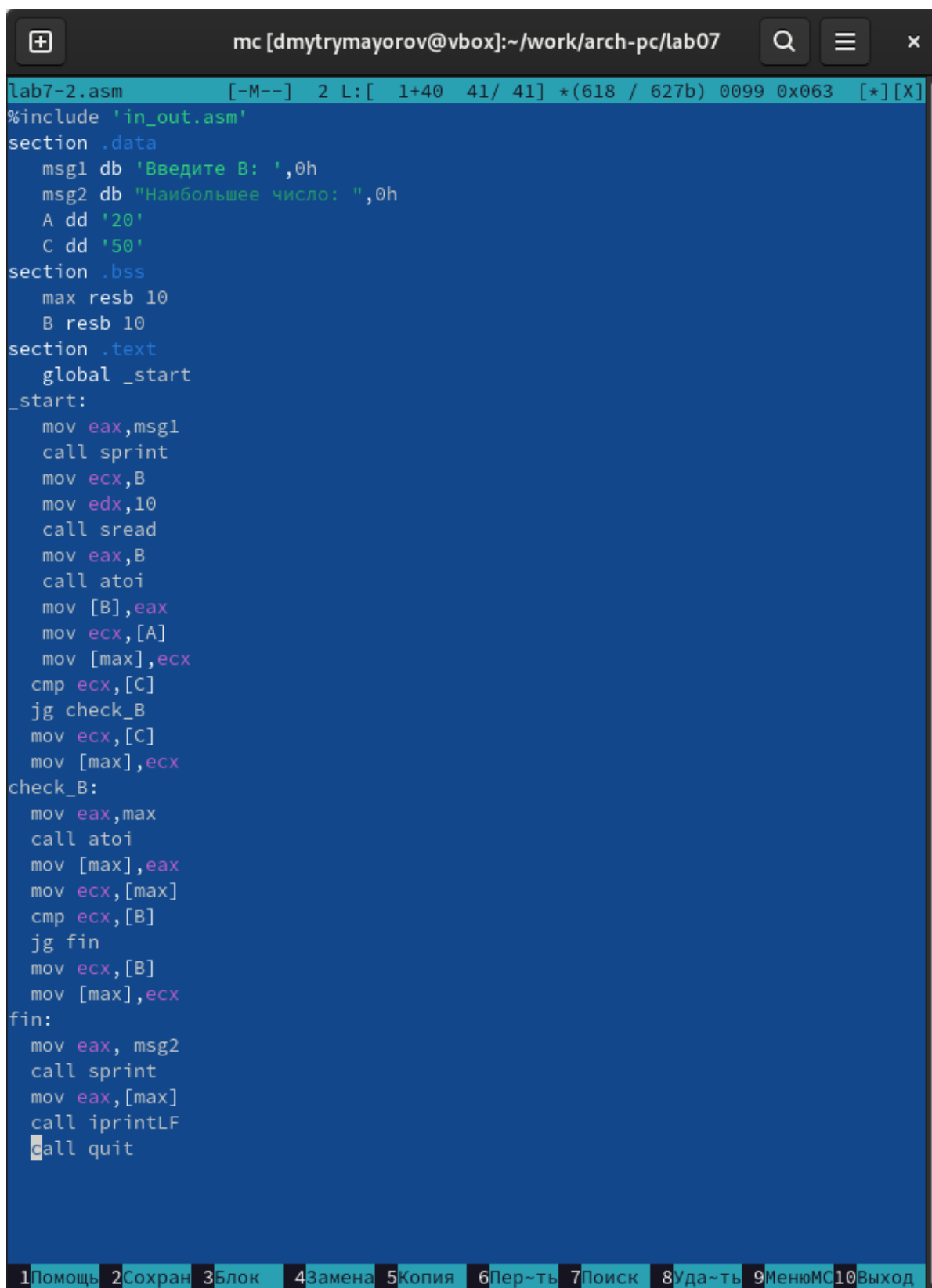
```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 3.7: Создаем исполняемый файл и запускаем его. Проверяем, правильный ли вывод



```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ touch lab7-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 3.8: Создаем новый файл



The screenshot shows a text editor window titled "mc [dmytrymayorov@vbox]:~/work/arch-pc/lab07". The editor displays the assembly file "lab7-2.asm". The code includes a header file "in_out.asm" and defines three sections: .data, .bss, and .text. The .data section contains two strings, "Введите В: ", and two constants, 20 and 50. The .bss section reserves space for a variable "max" and a buffer "B". The .text section contains the main logic: it prints the prompt, reads a number from the user, converts it to an integer, and compares it to the constant 20. If the user's input is greater, it prints the maximum value; otherwise, it prints the constant 20. The program ends by printing a newline and quitting.

```
lab7-2.asm [-M--] 2 L: [ 1+40 41/ 41] *(618 / 627b) 0099 0x063 [*] [X]
#include 'in_out.asm'
section .data
    msg1 db 'Введите В: ',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:
    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 [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx
check_B:
    mov eax,max
    call atoi
    mov [max],eax
    mov ecx,[max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

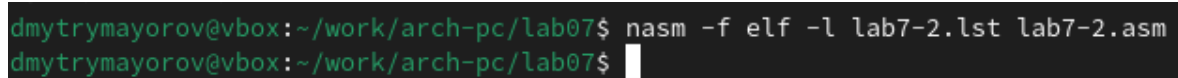
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

Рис. 3.9: Открываем файл и заполняем его в соответствии с листингом

```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 1
Наибольшее число: 50
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 5
Наибольшее число: 50
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 15
Наибольшее число: 50
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 3.10: Создаем исполняемый файл и запускаем его. Вводим разные значения В

4 Изучение структуры файлы листинга

A terminal window with a dark background and green text. The prompt is 'dmytrymayorov@vbox:~/work/arch-pc/lab07\$'. The command entered is 'nasm -f elf -l lab7-2.lst lab7-2.asm'. The prompt is repeated on the next line with a cursor.

```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 4.1: Создаем файл листинга для программы lab7-2.asm

```

dmytrymayorov@vbox:~/work/arch-pc/lab07
lab7-2.lst [----] 0 L: [ 1+ 0 1/217] *(0 /13022b) 0032 0x020 [*] [X]
1 %include 'in_out.asm'
2 <1> ;----- slen -----
3 <1> ; Функция вычисления длины сообщения
4 <1> slen:.....
5 00000000 53 <1> push ebx.....
6 00000001 89C3 <1> mov ebx, eax.....
7 <1>.....
8 <1> nextchar:.....
9 00000003 803800 <1> cmp byte [eax], 0...
10 00000006 7403 <1> jz finished.....
11 00000008 40 <1> inc eax.....
12 00000009 EBF8 <1> jmp nextchar.....
13 <1>.....
14 <1> finished:
15 0000000B 29D8 <1> sub eax, ebx
16 0000000D 5B <1> pop ebx.....
17 0000000E C3 <1> ret.....
18 <1>.....
19 <1> ;----- sprint -----
20 <1> ; Функция печати сообщения
21 <1> ; входные данные: mov eax,<message>
22 <1> sprint:
23 0000000F 52 <1> push edx
24 00000010 51 <1> push ecx
25 00000011 53 <1> push ebx
26 00000012 50 <1> push eax
27 00000013 E8E8FFFFFF <1> call slen
28 <1>.....
29 00000018 89C2 <1> mov edx, eax
30 0000001A 58 <1> pop eax
31 <1>.....
32 0000001B 89C1 <1> mov ecx, eax
33 0000001D BB01000000 <1> mov ebx, 1
34 00000022 B804000000 <1> mov eax, 4
35 00000027 CD80 <1> int 80h
36 <1>.....
37 00000029 5B <1> pop ebx
38 0000002A 59 <1> pop ecx
39 0000002B 5A <1> pop edx
40 0000002C C3 <1> ret
41 <1>.....
42 <1>.....
43 <1> ;----- sprintLF -----
44 <1> ; Функция печати сообщения с переводом с
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

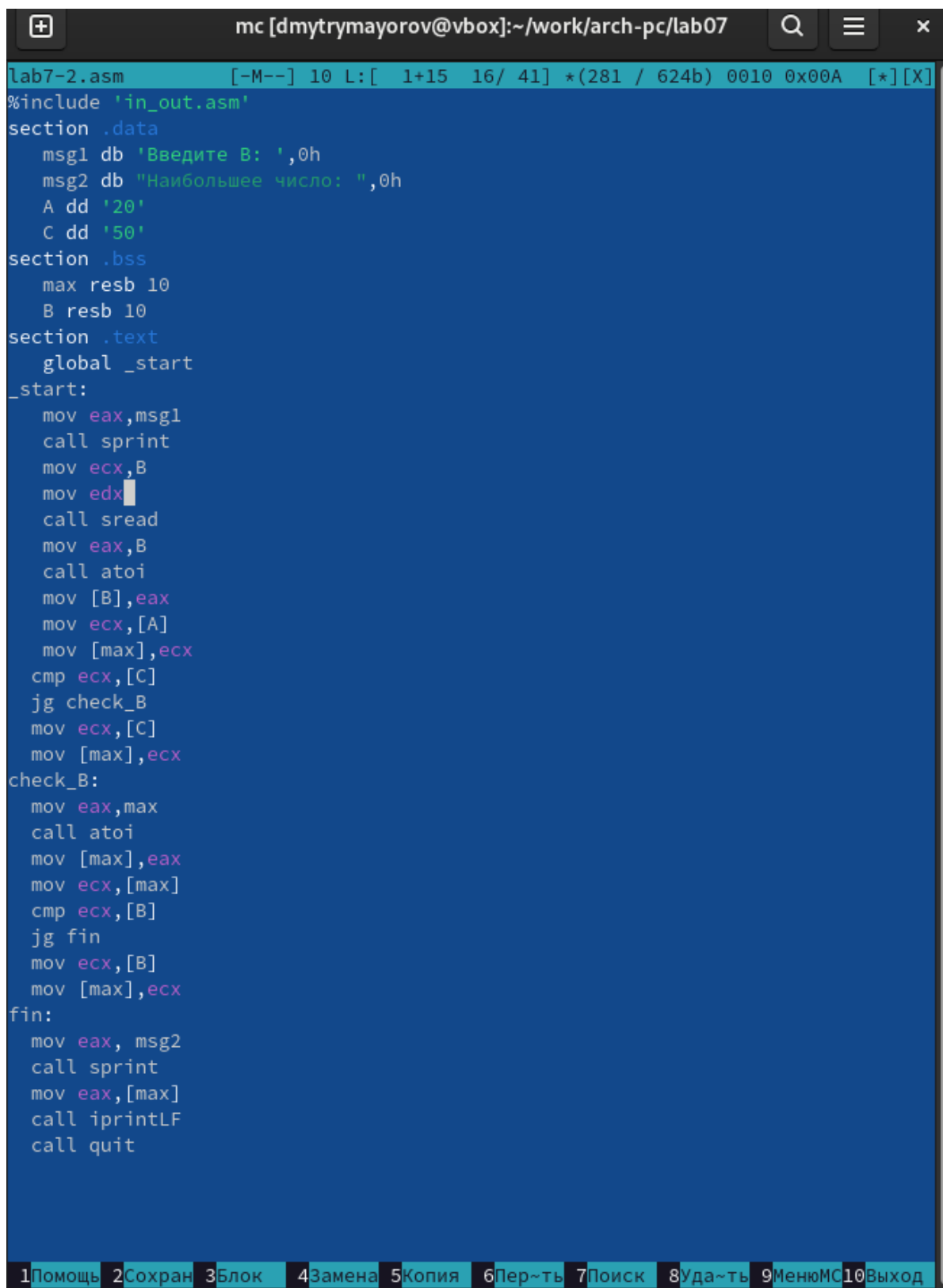
```

Рис. 4.2: Открываем файл листинга и изучаем его

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, mov ebx,1-присвоение переменной ebx значения 1

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4

Строка 35: 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.



```
lab7-2.asm [-M--] 10 L: [ 1+15 16/ 41] *(281 / 624b) 0010 0x00A [*] [X]
#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:
    mov eax,msg1
    call sprint
    mov ecx,B
    mov edx
    call sread
    mov eax,B
    call atoi
    mov [B],eax
    mov ecx,[A]
    mov [max],ecx
    cmp ecx,[C]
    jg check_B
    mov ecx,[C]
    mov [max],ecx
check_B:
    mov eax,max
    call atoi
    mov [max],eax
    mov ecx,[max]
    cmp ecx,[B]
    jg fin
    mov ecx,[B]
    mov [max],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[max]
    call iprintLF
    call quit
```

Рис. 4.3: Открываем файл и удаляем один операнд


```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:16: error: invalid combination of opcode and operands
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 4.4: Транслируем файл

При трансляции файла, выдается ошибка, но создаются файлы lab7-2 и lab7-2.lst

```

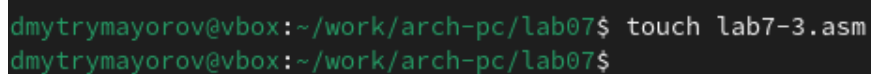
dmytrymayorov@vbox:~/work/arch-pc/lab07
lab7-2.lst [----] 30 L:[108+21 129/218] *(8105/13110b) 0032 0x020 [*][X]
107 <1> .
108 <1> ;----- iprintLF -----
109 <1> ; Функция вывода на экран чисел в формат
110 <1> ; входные данные: mov eax,<int>
111 <1> iprintLF:
112 00000086 E8C9FFFFFF <1> call iprint.....
113 <1> .
114 0000008B 50 <1> push eax.....
115 0000008C B80A000000 <1> mov eax, 0Ah.....
116 00000091 50 <1> push eax.....
117 00000092 89E0 <1> mov eax, esp.....
118 00000094 E876FFFFFF <1> call sprint.....
119 00000099 58 <1> pop eax.....
120 0000009A 58 <1> pop eax.....
121 0000009B C3 <1> ret
122 <1> .
123 <1> ;----- atoi -----
124 <1> ; Функция преобразования ascii-код символа
125 <1> ; входные данные: mov eax,<int>
126 <1> atoi:
127 0000009C 53 <1> push ebx.....
128 0000009D 51 <1> push ecx.....
129 0000009E 52 <1> push edx.....
130 0000009F 56 <1> push esi.....
131 000000A0 89C6 <1> mov esi, eax.....
132 000000A2 B800000000 <1> mov eax, 0.....
133 000000A7 B900000000 <1> mov ecx, 0.....
134 <1> .
135 <1> .multiplyLoop:
136 000000AC 31DB <1> xor ebx, ebx.....
137 000000AE 8A1C0E <1> mov bl, [esi+ecx]
138 000000B1 80FB30 <1> cmp bl, 48.
139 000000B4 7C14 <1> jl .finished.
140 000000B6 80FB39 <1> cmp bl, 57..
141 000000B9 7F0F <1> jg .finished.
142 <1> .
143 000000BB 80EB30 <1> sub bl, 48.
144 000000BE 01D8 <1> add eax, ebx
145 000000C0 BB0A000000 <1> mov ebx, 10..
146 000000C5 F7E3 <1> mul ebx..
147 000000C7 41 <1> inc ecx...
148 000000C8 EBE2 <1> jmp .multiplyLoop..
149 <1> .
150 <1> .finished:
151 000000CA 83F900 <1> cmp ecx, 0..
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход

```

Рис. 4.5: Открываем файл с ошмбкой и изучаем его

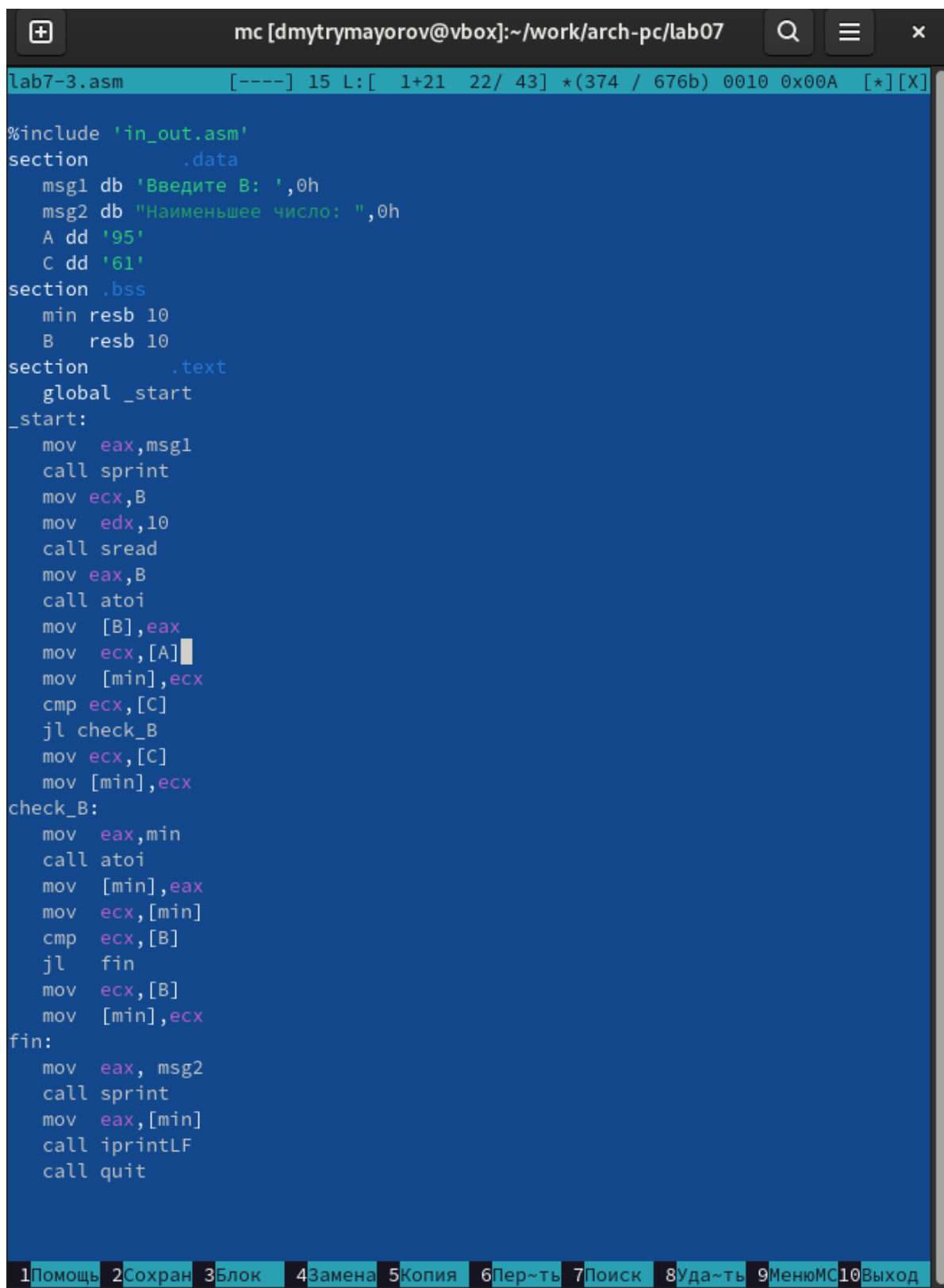
5 Задание для самостоятельной работы (Вариант 20)

Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.



```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ touch lab7-3.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 5.1: Создаем новый файл



```
lab7-3.asm [----] 15 L: [ 1+21 22/ 43] *(374 / 676b) 0010 0x00A [*] [X]

#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd '95'
    C dd '61'
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]
    jl check_B
    mov ecx,[C]
    mov [min],ecx
check_B:
    mov eax,min
    call atoi
    mov [min],eax
    mov ecx,[min]
    cmp ecx,[B]
    jl fin
    mov ecx,[B]
    mov [min],ecx
fin:
    mov eax, msg2
    call sprint
    mov eax,[min]
    call iprintLF
    call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС10Выход

Рис. 5.2: Открываем файл и пишем программу

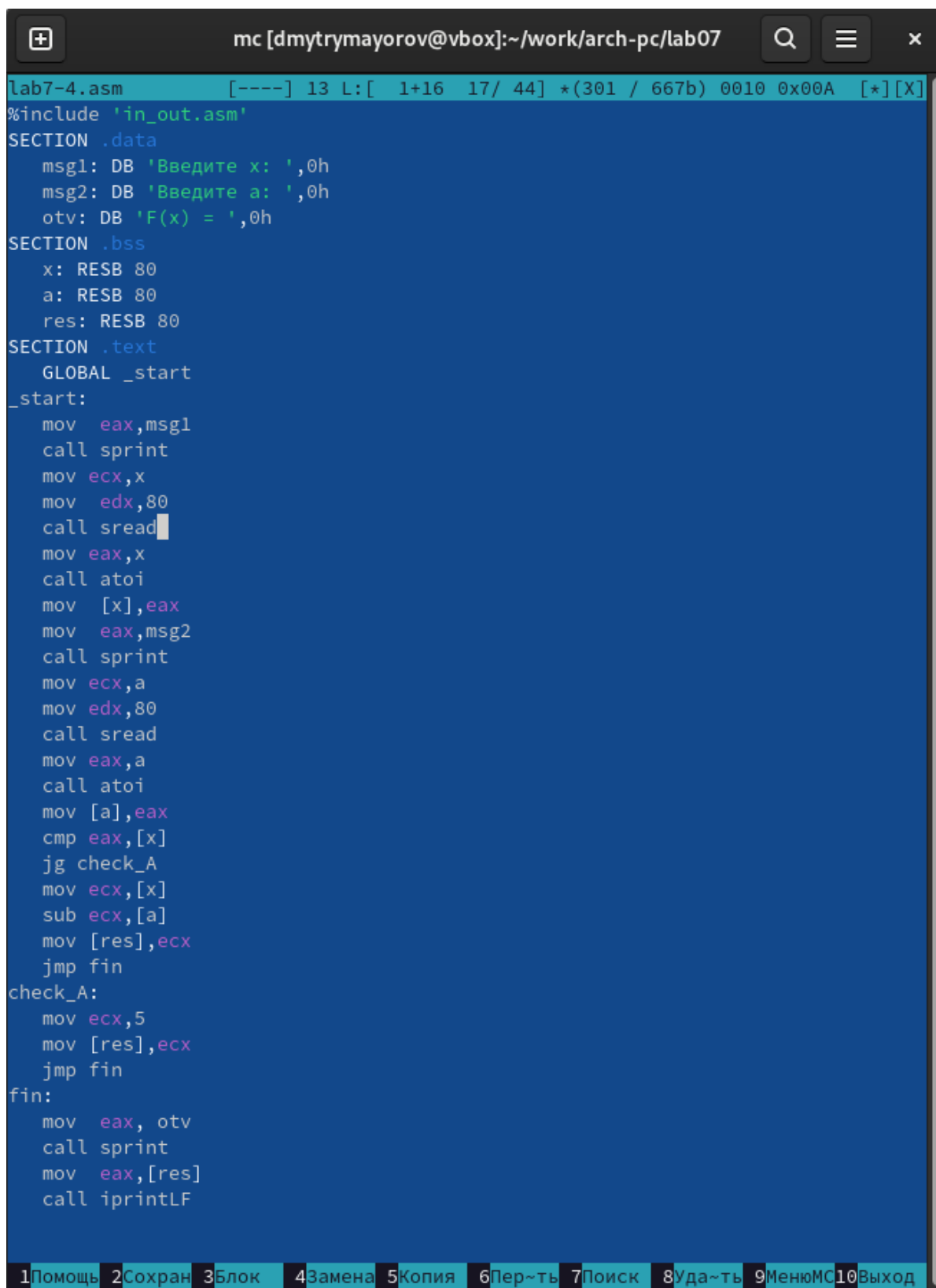
```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-3
Введите В: 95
Наименьшее число: 61
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 5.3: Танслируем файл и смотрим на его работу

Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ touch lab7-4.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$
```

Рис. 5.4: Создаем новый файл



```
lab7-4.asm [----] 13 L: [ 1+16 17/ 44] *(301 / 667b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .data
    msg1: DB 'Введите x: ',0h
    msg2: DB 'Введите a: ',0h
    otv: DB 'F(x) = ',0h
SECTION .bss
    x: RESB 80
    a: RESB 80
    res: RESB 80
SECTION .text
    GLOBAL _start
_start:
    mov eax,msg1
    call sprint
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    mov [x],eax
    mov eax,msg2
    call sprint
    mov ecx,a
    mov edx,80
    call sread
    mov eax,a
    call atoi
    mov [a],eax
    cmp eax,[x]
    jg check_A
    mov ecx,[x]
    sub ecx,[a]
    mov [res],ecx
    jmp fin
check_A:
    mov ecx,5
    mov [res],ecx
    jmp fin
fin:
    mov eax, otv
    call sprint
    mov eax,[res]
    call iprintLF
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход

Рис. 5.5: Открываем файл и пишем программу

```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
F(x) = 5
```

Рис. 5.6: Транслируем файл и проверяем его работу при $x=1$ и $a=2$

```
dmytrymayorov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
dmytrymayorov@vbox:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
F(x) = 1
```

Рис. 5.7: Транслируем файл и проверяем его работу при $x=2$ и $a=1$

6 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.

Список литературы