

Отчёт по лабораторной работе 7

Архитектура компьютера

Хаоладар Шаханеоядж НПИ-01-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	13
2.3	Задание для самостоятельной работы	15
3	Выводы	20

Список иллюстраций

2.1	Программа в файле lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Программа в файле lab7-1.asm	9
2.4	Запуск программы lab7-1.asm	9
2.5	Программа в файле lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	11
2.7	Программа в файле lab7-2.asm	12
2.8	Запуск программы lab7-2.asm	12
2.9	Файл листинга lab7-2	13
2.10	Ошибка трансляции lab7-2	14
2.11	Файл листинга с ошибкой lab7-2	15
2.12	Программа в файле task7-1.asm	16
2.13	Запуск программы task7-1.asm	17
2.14	Программа в файле task7-2.asm	18
2.15	Запуск программы task7-2.asm	19

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

Создал каталог для программам лабораторной работы № 7 и файл lab7-1.asm

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Написал в файл lab7-1.asm текст программы из листинга 7.1.



```
mc [haolader69@vbox]:~/work/arch-pc/lab0
lab7-1.asm [----] 9 L: [ 1+24 25/ 25]
#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

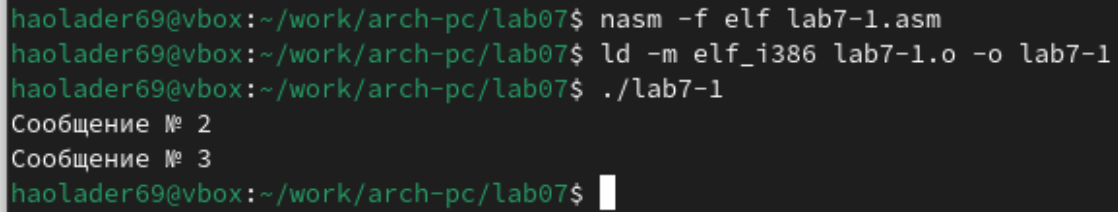
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его.

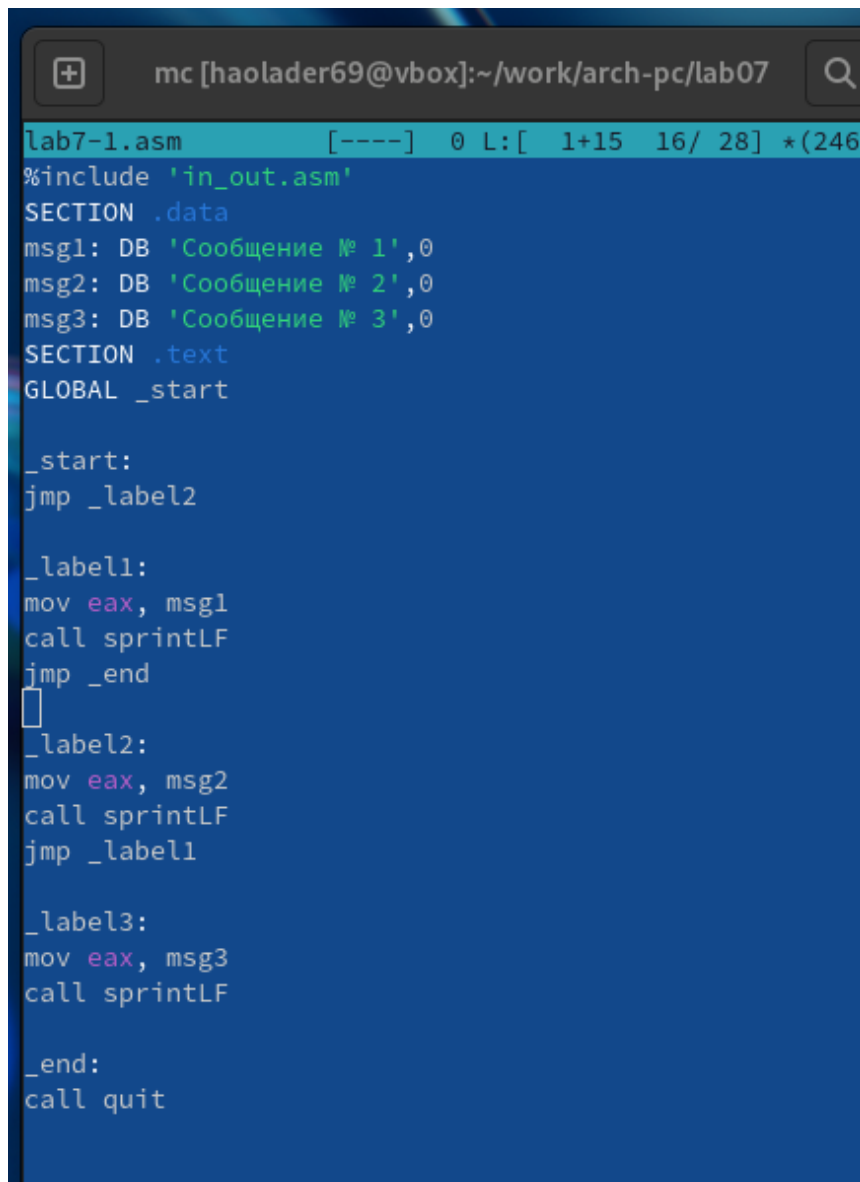
A terminal window with a dark background and green text. The prompt is 'haolader69@vbox:~/work/arch-pc/lab07\$'. The first command is 'nasm -f elf lab7-1.asm'. The second command is 'ld -m elf_i386 lab7-1.o -o lab7-1'. The third command is './lab7-1'. The output shows 'Сообщение № 2' and 'Сообщение № 3' on separate lines. The prompt returns after the execution.

```
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
haolader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
haolader69@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
haolader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.2: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2.



```
lab7-1.asm [-----] 0 L: [ 1+15 16/ 28] *(246)
#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

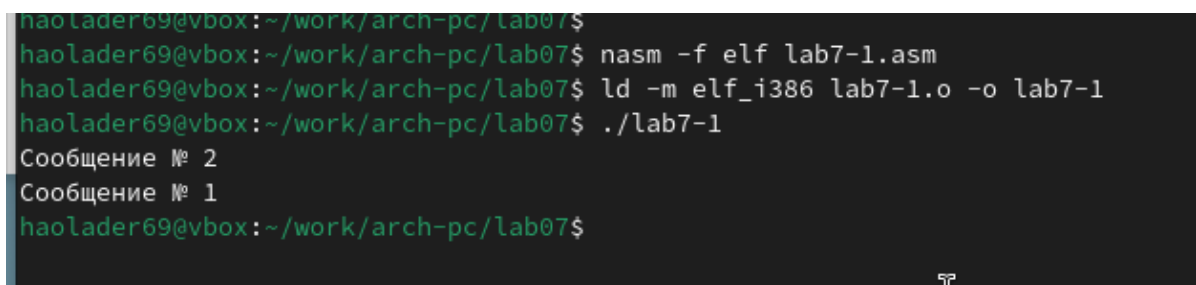
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Программа в файле lab7-1.asm



```
haolader69@vbox:~/work/arch-pc/lab07$
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
haolader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
haolader69@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
haolader69@vbox:~/work/arch-pc/lab07$
```

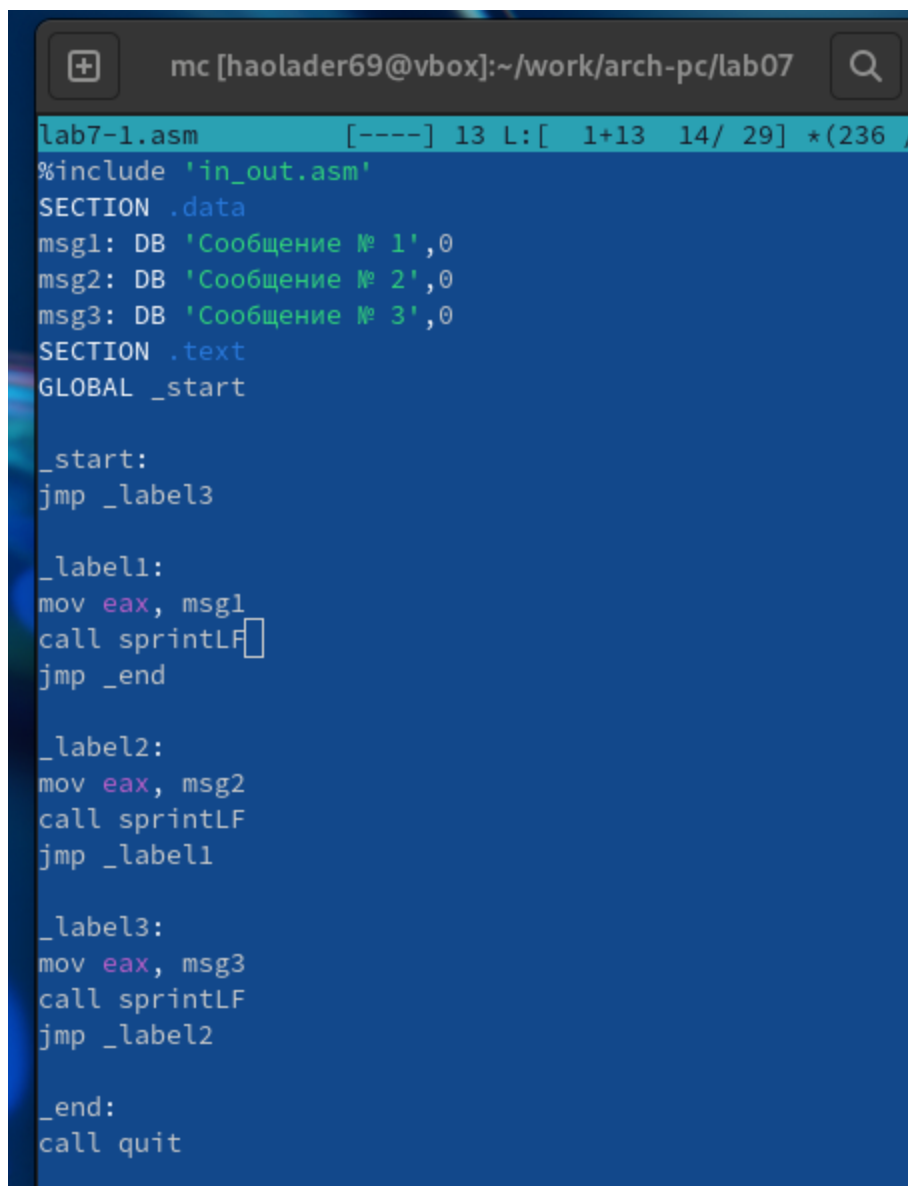
Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab7-1.asm [-----] 13 L: [ 1+13 14/ 29] *(236 /
#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 sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

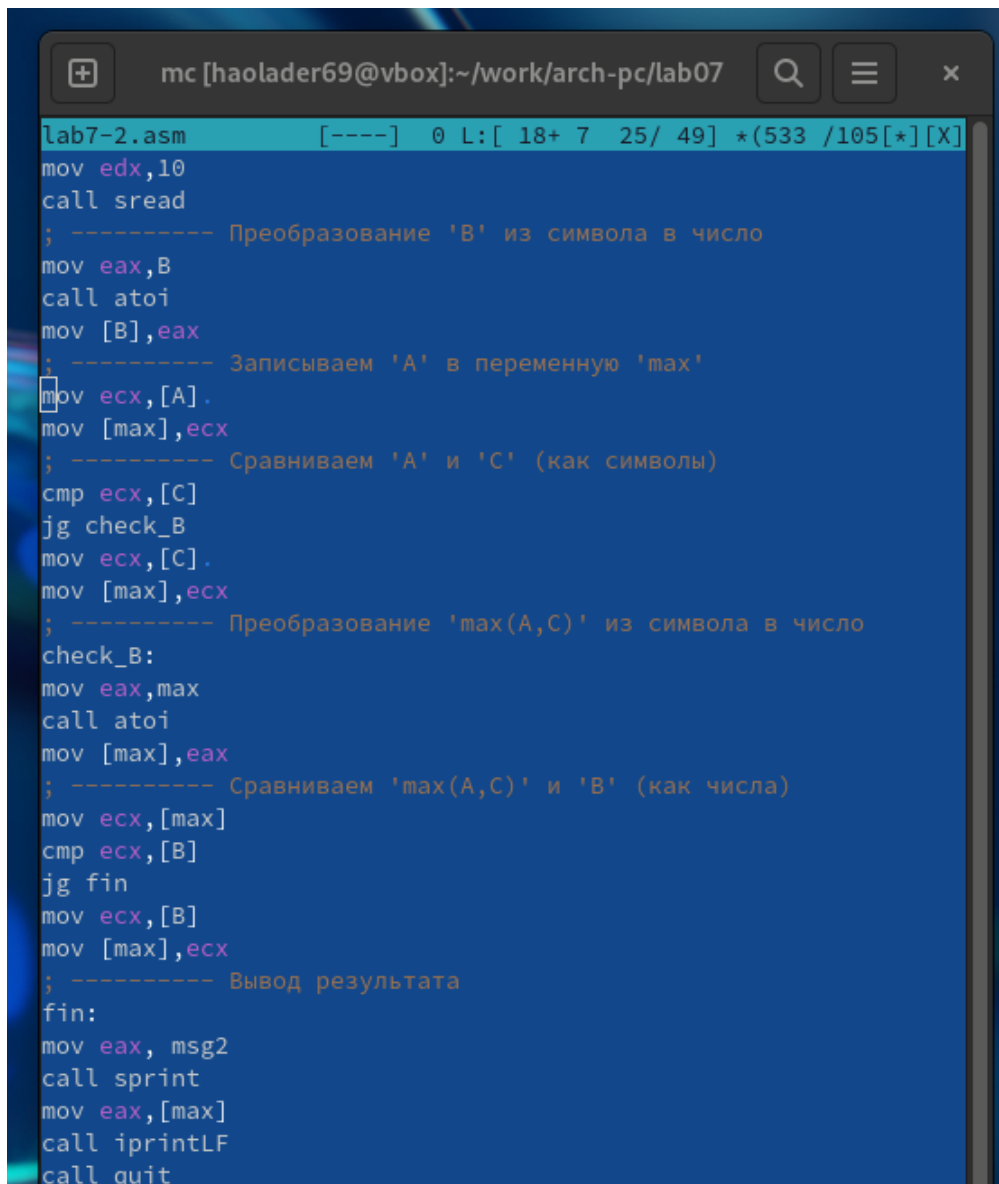
Рис. 2.5: Программа в файле lab7-1.asm

```
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
haolader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
haolader69@vbox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
haolader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.6: Запуск программы lab7-1.asm

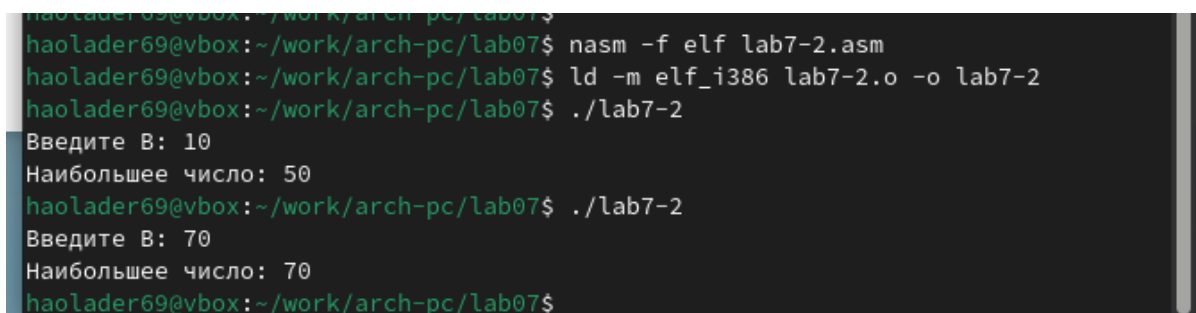
Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений В.



```
lab7-2.asm [----] 0 L: [ 18+ 7 25/ 49] *(533 /105[*] [X]
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.7: Программа в файле lab7-2.asm



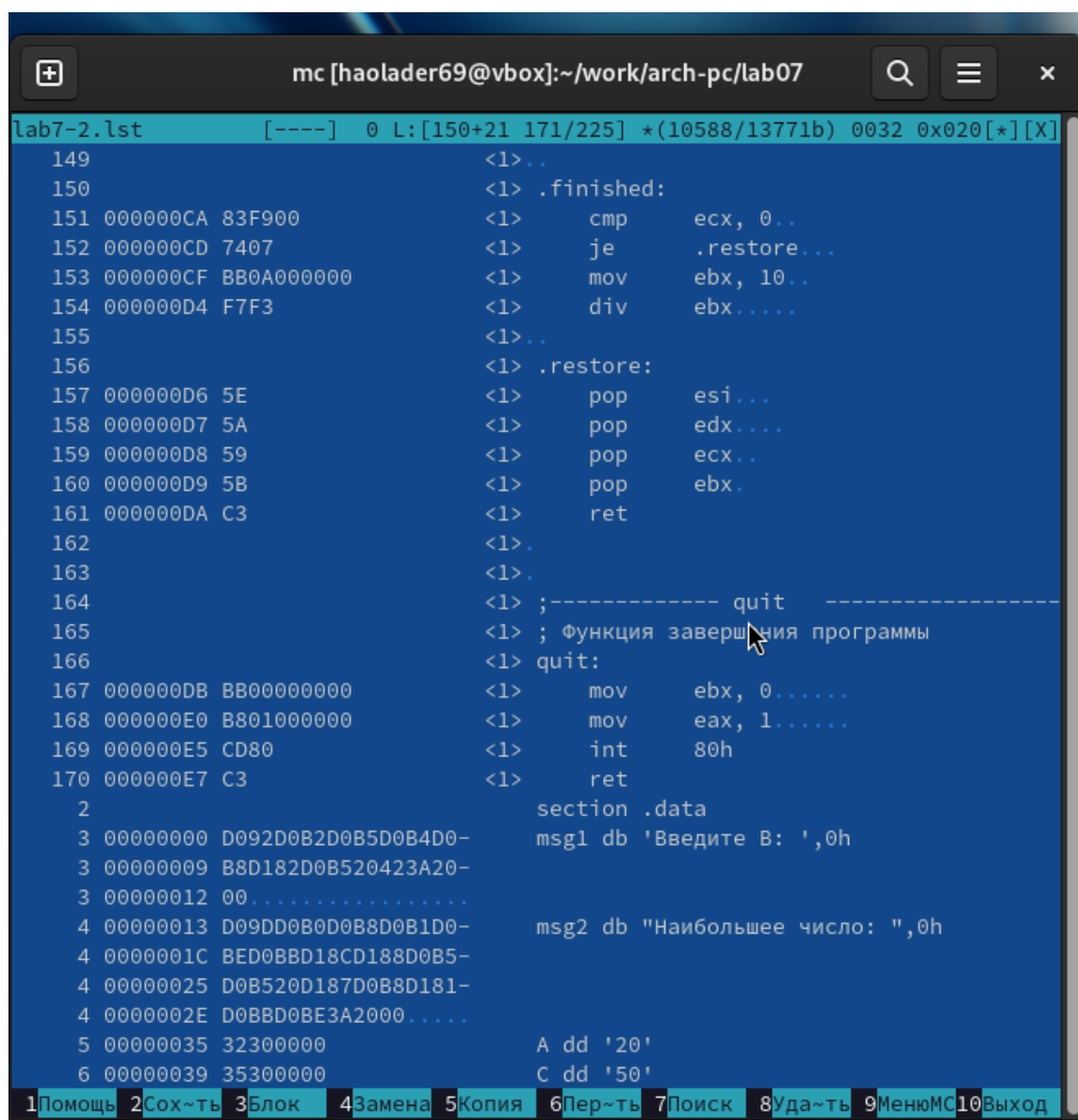
```
hao1ader69@vbox:~/work/arch-pc/lab07$
hao1ader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
hao1ader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
hao1ader69@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
hao1ader69@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
hao1ader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск программы lab7-2.asm

2.2 Изучение структуры файлы листинга

Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm`



```
lab7-2.lst [----] 0 L:[150+21 171/225] *(10588/13771b) 0032 0x020[*][X]
149 <1>..
150 <1> .finished:
151 000000CA 83F900 <1> cmp ecx, 0..
152 000000CD 7407 <1> je .restore...
153 000000CF BB0A000000 <1> mov ebx, 10..
154 000000D4 F7F3 <1> div ebx....
155 <1>..
156 <1> .restore:
157 000000D6 5E <1> pop esi...
158 000000D7 5A <1> pop edx....
159 000000D8 59 <1> pop ecx..
160 000000D9 5B <1> pop ebx.
161 000000DA C3 <1> ret
162 <1>..
163 <1>..
164 <1> ;----- quit -----
165 <1> ; Функция завершения программы
166 <1> quit:
167 000000DB BB00000000 <1> mov ebx, 0.....
168 000000E0 B801000000 <1> mov eax, 1.....
169 000000E5 CD80 <1> int 80h
170 000000E7 C3 <1> ret
2 section .data
3 00000000 D092D0B2D0B5D0B4D0- msg1 db 'Введите B: ',0h
3 00000009 B8D182D0B520423A20-
3 00000012 00.....
4 00000013 D09DD0B0D0B8D0B1D0- msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000....
5 00000035 32300000 A dd '20'
6 00000039 35300000 C dd '50'
```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню

содержимое трёх строк файла листинга по выбору.

строка 203

- 28 - номер строки в подпрограмме
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `cmp esx,[C]` - код программы - сравнивает регистр `esx` и переменную `C`

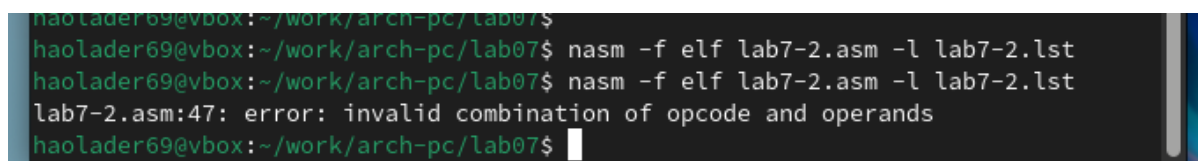
строка 204

- 29 - номер строки в подпрограмме
- 00000122 - адрес
- 7F0C - машинный код
- `jb check_B` - код программы - если `>`, то переход к метке `check_B`

строка 205

- 30 - номер строки в подпрограмме
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- `mov esx,[C]` - код программы - перекладывает в регистр `esx` значение переменной `C`

Открыл файл с программой `lab7-2.asm` и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.



```
haolader69@vbox:~/work/arch-pc/lab07$  
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:47: error: invalid combination of opcode and operands  
haolader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции `lab7-2`

```
lab7-2.lst  [----]  0  L:[195+31 226/226]  *(13857/13857b)  <EOF>  [*] [X]
20                                     ; ----- Преобразование 'B' из симво
21 00000101 B8[0A000000]             mov eax,B
22 00000106 E891FFFFFF               call atoi
23 0000010B A3[0A000000]             mov [B],eax
24                                     ; ----- Записываем 'A' в переменную
25 00000110 8B0D[35000000]           mov ecx,[A].
26 00000116 890D[00000000]           mov [max],ecx
27                                     ; ----- Сравниваем 'A' и 'C' (как с
28 0000011C 3B0D[39000000]           cmp ecx,[C]
29 00000122 7F0C                     jg check_B
30 00000124 8B0D[39000000]           mov ecx,[C].
31 0000012A 890D[00000000]           mov [max],ecx
32                                     ; ----- Преобразование 'max(A,C)' и
33                                     check_B:
34 00000130 B8[00000000]             mov eax,max
35 00000135 E862FFFFFF               call atoi
36 0000013A A3[00000000]             mov [max],eax
37                                     ; ----- Сравниваем 'max(A,C)' и 'B'
38 0000013F 8B0D[00000000]           mov ecx,[max]
39 00000145 3B0D[0A000000]           cmp ecx,[B]
40 0000014B 7F0C                     jg fin
41 0000014D 8B0D[0A000000]           mov ecx,[B]
42 00000153 890D[00000000]           mov [max],ecx
43                                     ; ----- Вывод результата
44                                     fin:
45 00000159 B8[13000000]             mov eax,msg2
46 0000015E E8ACFFFFFF               call sprint
47                                     mov eax,
47                                     ***** error: invalid combination of opcode and
48 00000163 E81EFFFFFF               call iprintLF
49 00000168 E86EFFFFFF               call quit
```

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

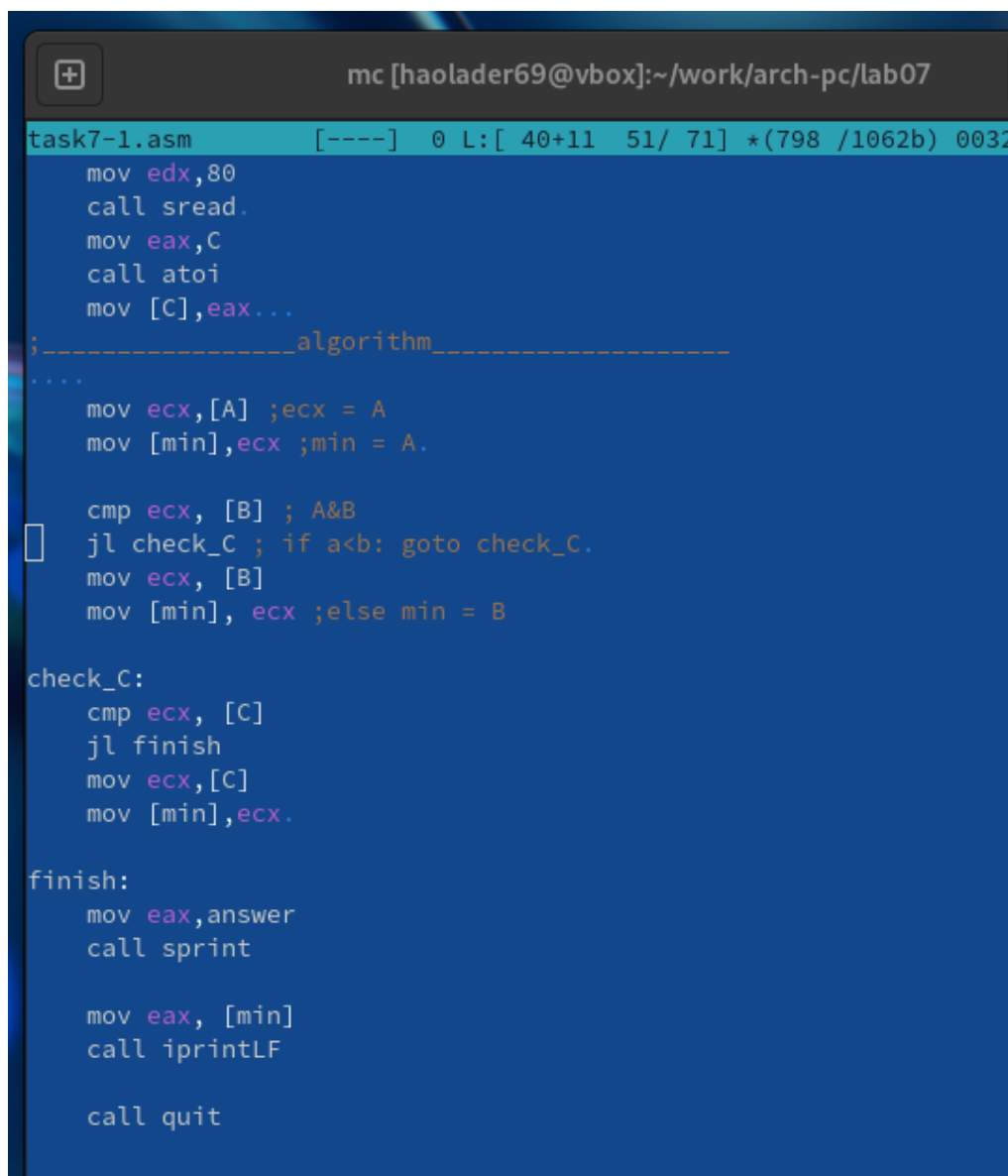
Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

2.3 Задание для самостоятельной работы

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 № 6. Создайте исполняемый файл и проверьте его работу для варианта 3 - 94,5,58



```
mc [haolader69@vbox]:~/work/arch-pc/lab07
task7-1.asm  [----]  0 L: [ 40+11  51/ 71] *(798 /1062b) 0032
    mov edx,80
    call sread.
    mov eax,C
    call atoi
    mov [C],eax...
;-----algorithm-----
....
    mov ecx,[A] ;ecx = A
    mov [min],ecx ;min = A.

    cmp ecx, [B] ; A&B
    jl check_C ; if a<b: goto check_C.
    mov ecx, [B]
    mov [min], ecx ;else min = B

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx.

finish:
    mov eax,answer
    call sprint

    mov eax, [min]
    call iprintLF

    call quit
```

Рис. 2.12: Программа в файле task7-1.asm


```
hao1ader69@vbox:~/work/arch-pc/lab07$  
hao1ader69@vbox:~/work/arch-pc/lab07$ nasm -f elf task7-1.asm  
hao1ader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task7-1.o -o task7-1  
hao1ader69@vbox:~/work/arch-pc/lab07$ ./task7-1  
Input A: 94  
Input B: 5  
Input C: 58  
Smallest: 5  
hao1ader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы task7-1.asm

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.

для варианта 3

$$\begin{cases} 3x, x = 3 \\ a + 1, x \neq 3 \end{cases}$$

```
mc [haolader69@vbox]:~/work/arch-pc/l
task7-2.asm [----] 7 L: [ 21+27 48/ 52] *(695 /
    call atoi.
    mov [A],eax

    mov eax,msgX
    call sprint
    mov ecx,X
    mov edx,80
    call sread.
    mov eax,X
    call atoi
    mov [X],eax...
;-----algorithm-----

    mov ebx, [X]
    mov edx, 3
    cmp ebx, edx
    je first
    jmp second

first:
    mov eax,[X]
    mov ebx,3
    mul ebx
    call iprintLF.
    call quit
second:
    mov eax,[A]
    add eax,1
    call iprintLF.
    call quit
```

Рис. 2.14: Программа в файле task7-2.asm

```
haolader69@vbox:~/work/arch-pc/lab07$  
haolader69@vbox:~/work/arch-pc/lab07$  
haolader69@vbox:~/work/arch-pc/lab07$ nasm -f elf task7-2.asm  
haolader69@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 task7-2.o -o task7-2  
haolader69@vbox:~/work/arch-pc/lab07$ ./task7-2  
Input A: 4  
Input X: 3  
9  
haolader69@vbox:~/work/arch-pc/lab07$ ./task7-2  
Input A: 4  
Input X: 1  
5  
haolader69@vbox:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы task7-2.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.