

Отчёт по лабораторной работе 7

Архитектура компьютера

Соловьев Серафим

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание для самостоятельной работы	16
3	Выводы	21

Список иллюстраций

2.1	Код программы lab7-1.asm	7
2.2	Компиляция и запуск программы lab7-1.asm	8
2.3	Код программы lab7-1.asm	9
2.4	Компиляция и запуск программы lab7-1.asm	9
2.5	Код программы lab7-1.asm	10
2.6	Компиляция и запуск программы lab7-1.asm	11
2.7	Код программы lab7-2.asm	12
2.8	Компиляция и запуск программы lab7-2.asm	13
2.9	Файл листинга lab7-2	14
2.10	Ошибка трансляции lab7-2	15
2.11	Файл листинга с ошибкой lab7-2	16
2.12	Код программы lab7-3.asm	17
2.13	Компиляция и запуск программы lab7-3.asm	18
2.14	Код программы lab7-4.asm	19
2.15	Компиляция и запуск программы lab7-4.asm	20

Список таблиц

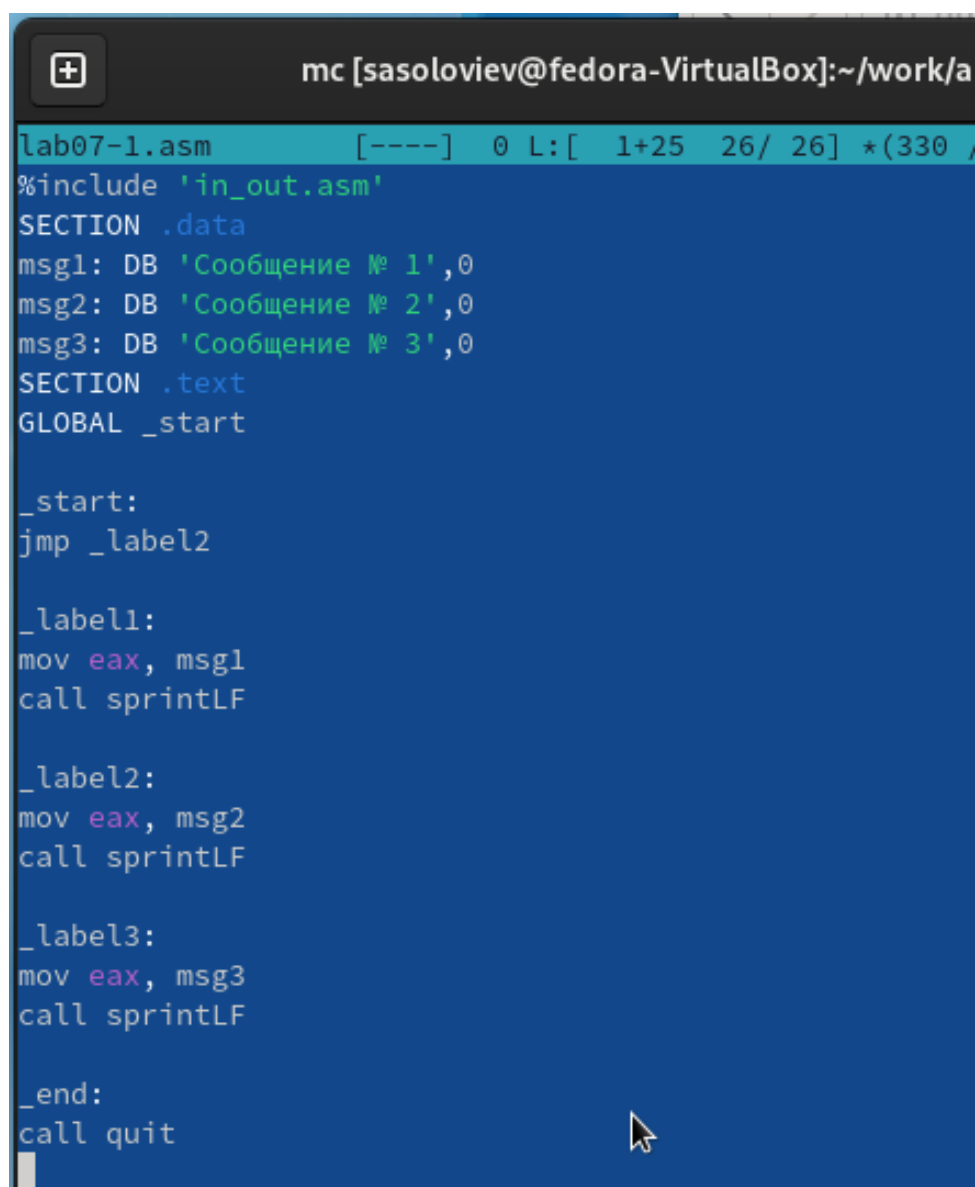
1 Цель работы

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

Я организовал папку для работы над седьмой лабораторной и создал файл с исходным кодом lab7-1.asm.

В NASM команда `jmp` применяется для безусловного перехода. Изучил пример кода с этой командой и внёс его в файл lab7-1.asm.



```
mc [sasoloviev@fedora-VirtualBox]:~/work/a
lab07-1.asm [----] 0 L:[ 1+25 26/ 26] *(330 /
#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: Код программы lab7-1.asm

Скомпилировал и запустил полученную программу.

```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-1.asm
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-1.o -o lab07-1
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-1
Сообщение № 2
Сообщение № 3
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.2: Компиляция и запуск программы lab7-1.asm

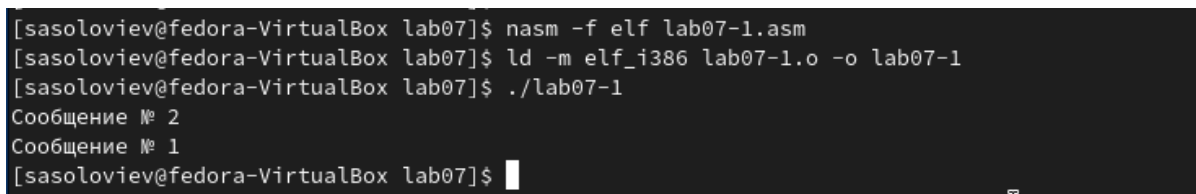
Команда `jmp` позволяет переходить как вперёд, так и назад в коде. Модифицировал программу так, чтобы она сначала показывала “Сообщение № 2”, а потом “Сообщение № 1”, и после этого завершалась. Это было достигнуто добавлением команды `jmp` с меткой `_label1` после “Сообщения № 2” для перехода к выводу “Сообщения № 1”, и команды `jmp` с меткой `_end` после “Сообщения № 1” для завершения работы через вызов функции `quit`.

Внёс изменения в код, соответствующие листингу 7.2.



```
mc [sasoloviev@fedora-VirtualBox]:~/work/arch-  
lab07-1.asm [----] 8 L: [ 1+16 17/ 28] *(255 / 35  
%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: Код программы lab7-1.asm



```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-1.asm  
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-1.o -o lab07-1  
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-1  
Сообщение № 2  
Сообщение № 1  
[sasoloviev@fedora-VirtualBox lab07]$
```

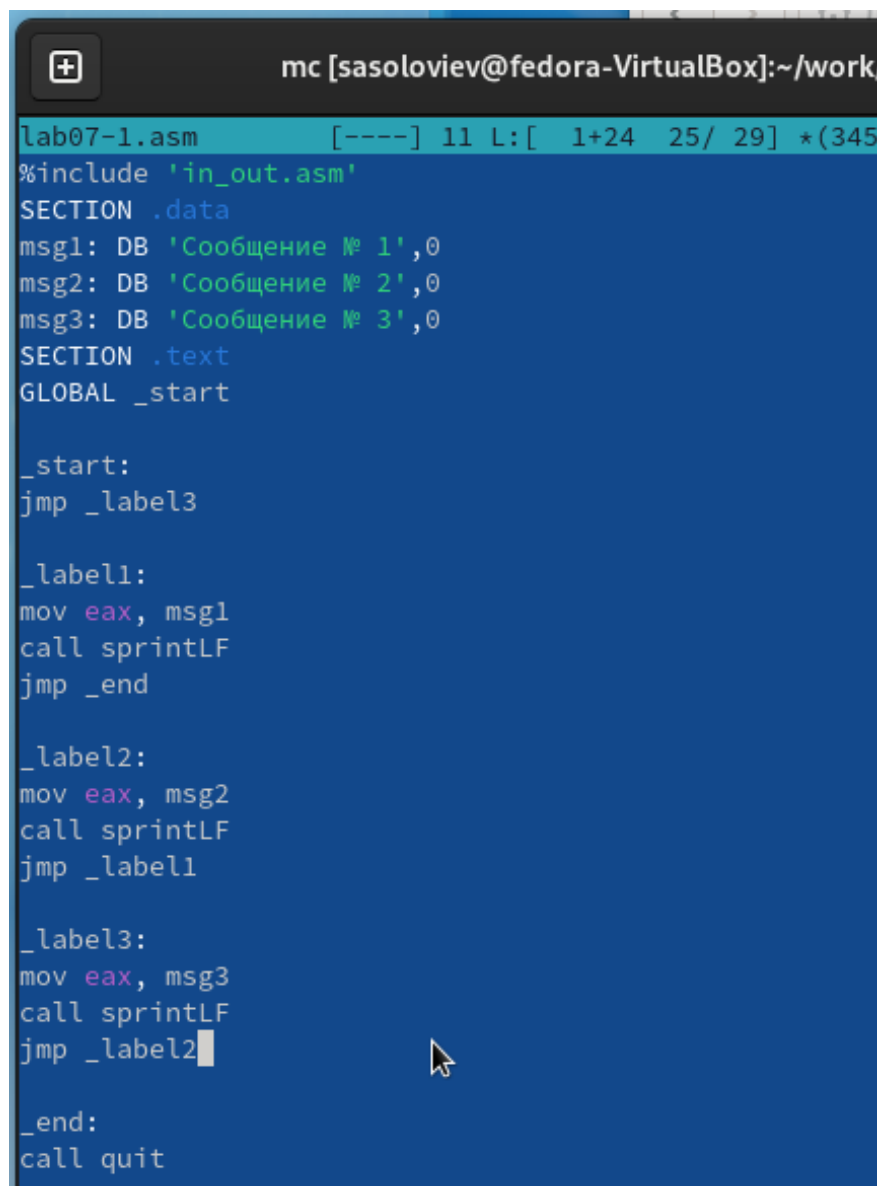
Рис. 2.4: Компиляция и запуск программы lab7-1.asm

Программа была переписана с изменёнными командами `jmp` для изменения порядка вывода.

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
mc [sasoloviev@fedora-VirtualBox]:~/work
lab07-1.asm [----] 11 L: [ 1+24 25/ 29] *(345
%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: Код программы lab7-1.asm

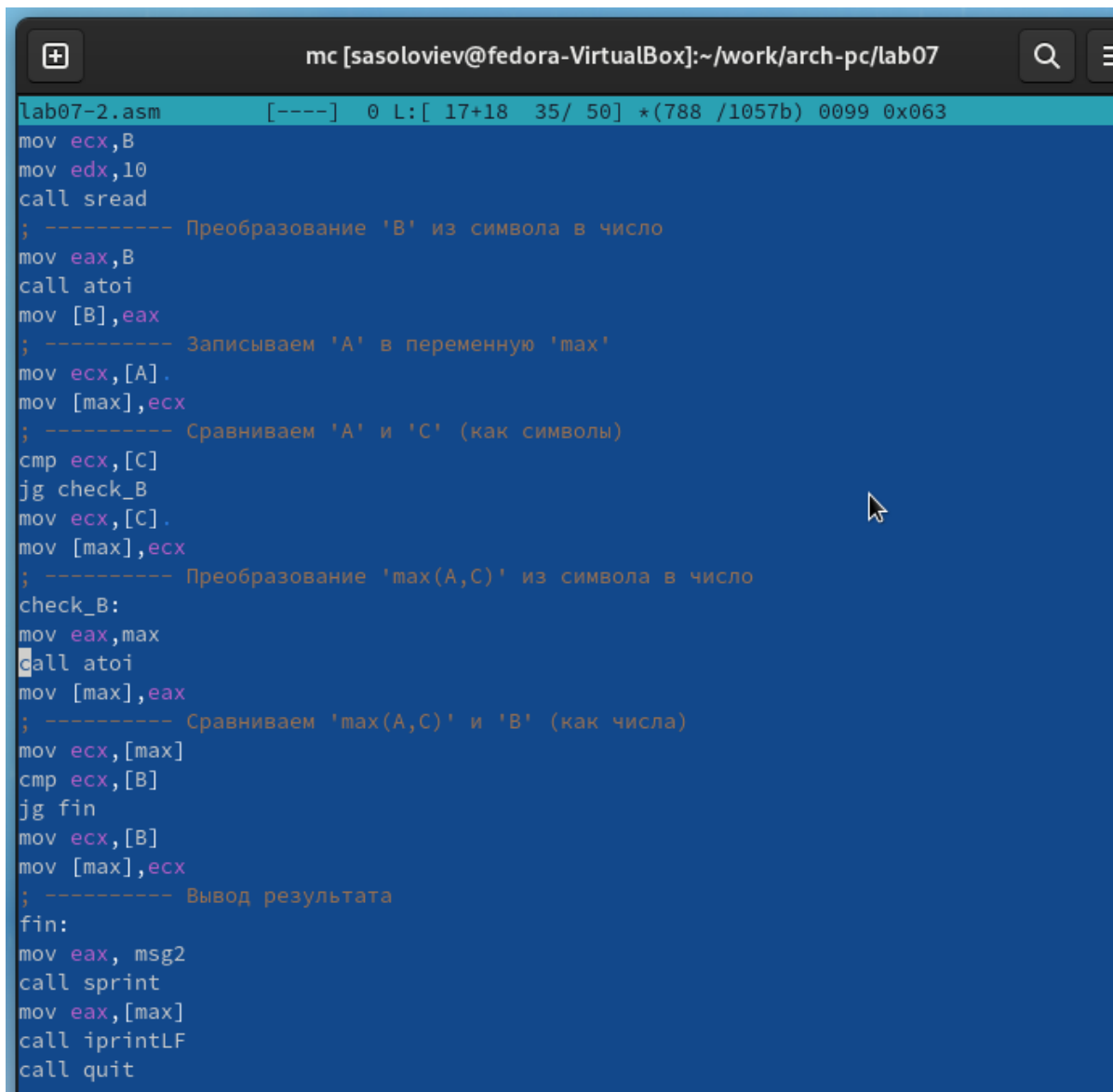
```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-1.asm
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-1.o -o lab07-1
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.6: Компиляция и запуск программы lab7-1.asm

Команда `jmp` всегда приводит к переходу. Однако в программировании часто требуются условные переходы, когда переход выполняется только при определённом условии.

Рассмотрим программу, которая вычисляет и выводит наибольшее из трёх чисел: А, В и С. Значения А и С заданы в коде, а значение В вводится пользователем.

Скомпилировал программу и провёл тестирование с различными вводимыми значениями В.



```
lab07-2.asm [----] 0 L: [ 17+18 35/ 50] *(788 /1057b) 0099 0x063
mov ecx,B
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: Код программы lab7-2.asm

```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-2.asm
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-2.o -o lab07-2
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-2
Введите B: 15
Наибольшее число: 50
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-2
Введите B: 40
Наибольшее число: 50
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-2
Введите B: 51
Наибольшее число: 51
[sasoloviev@fedora-VirtualBox lab07]$
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-2
Введите B: 60
Наибольшее число: 60
[sasoloviev@fedora-VirtualBox lab07]$
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.8: Компиляция и запуск программы lab7-2.asm

Обычно при компиляции с помощью `nasm` получается лишь объектный файл. Однако, чтобы сформировать файл листинга, следует использовать опцию `-l` и определить имя файла листинга через командную строку.

Сформировал листинг для кода, находящегося в `lab7-2.asm`.

```

Открыть + lab07-2.lst Стр. 1, Поз. 1
~/work/arch-pc/lab07

189 13 ; ----- Вывод сообщения 'Введите B: '
190 14 000000E8 B8[00000000] mov eax,msg1
191 15 000000ED E81DFFFFFF call sprint
192 16 ; ----- Ввод 'B'
193 17 000000F2 B9[0A000000] mov ecx,B
194 18 000000F7 BA0A000000 mov edx,10
195 19 000000FC E842FFFFFF call sread
196 20 ; ----- Преобразование 'B' из символа в число
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E801FFFFFF call atoi
199 23 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 00000130 B8[00000000] mov eax,max
211 35 00000135 E862FFFFFF call atoi
212 36 0000013A A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 0000013F 8B0D[00000000] mov ecx,[max]
215 39 00000145 3B0D[0A000000] cmp ecx,[B]
216 40 0000014B 7F0C jg fin
217 41 0000014D 8B0D[0A000000] mov ecx,[B]
218 42 00000153 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000159 B8[13000000] mov eax,msg2
222 46 0000015E E8ACFFFFFF call sprint
223 47 00000163 A1[00000000] mov eax,[max]
224 48 00000168 E819FFFFFF call iprintLF
225 49 0000016D E869FFFFFF call quit

```

Рис. 2.9: Файл листинга lab7-2

Я внимательно ознакомился с форматом и содержимым файла листинга. Подробно объясню содержимое трёх строк из этого файла.

строка 213

- 38 - номер строки в подпрограмме
- 0000013F - адрес
- 8B0D[00000000] - машинный код
- mov ecx,[max] - код программы - копирует MAX в ecx

строка 214

- 39 - номер строки в подпрограмме
- 00000145 - адрес
- 3B0D[0A000000] - машинный код
- `cmp esx,[B]` - код программы - сравнивает `esx` и `B`

строка 215

- 40 - номер строки в подпрограмме
- 0000014B - адрес
- 7F0C - машинный код
- `jg fin` - код программы - если больше перейти к метке `fin`

Затем я открыл исходный код в `lab7-2.asm` и удалил один операнд из команды, содержащей два операнда. После этого произвел компиляцию с целью создания файла листинга.

```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
[sasoloviev@fedora-VirtualBox lab07]$
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
lab07-2.asm:34: error: invalid combination of opcode and operands
[sasoloviev@fedora-VirtualBox lab07]$
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.10: Ошибка трансляции `lab7-2`

```

Открыть  + lab07-2.lst ~\work\arch-pc\lab07 Стр. 1, Поз. 1
190 14 000000E8 B8[00000000] mov eax,msg1
191 15 000000ED E81DFFFFFF call sprint
192 16 ; ----- Ввод 'B'
193 17 000000F2 B9[0A000000] mov ecx,B
194 18 000000F7 BA0A000000 mov edx,10
195 19 000000FC E842FFFFFF call sread
196 20 ; ----- Преобразование 'B' из символа в число
197 21 00000101 B8[0A000000] mov eax,B
198 22 00000106 E801FFFFFF call atoi
199 23 0000010B A3[0A000000] mov [B],eax
200 24 ; ----- Записываем 'A' в переменную 'max'
201 25 00000110 8B0D[35000000] mov ecx,[A]
202 26 00000116 890D[00000000] mov [max],ecx
203 27 ; ----- Сравниваем 'A' и 'C' (как символы)
204 28 0000011C 3B0D[39000000] cmp ecx,[C]
205 29 00000122 7F0C jg check_B
206 30 00000124 8B0D[39000000] mov ecx,[C]
207 31 0000012A 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 mov eax,
211 34 ***** error: invalid combination of opcode and operands
212 35 00000130 E867FFFFFF call atoi
213 36 00000135 A3[00000000] mov [max],eax
214 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 38 0000013A 8B0D[00000000] mov ecx,[max]
216 39 00000140 3B0D[0A000000] cmp ecx,[B]
217 40 00000146 7F0C jg fin
218 41 00000148 8B0D[0A000000] mov ecx,[B]
219 42 0000014F 890D[00000000] mov [max],ecx
220 43 ; ----- Вывод результата
221 44 fin:
222 45 00000154 B8[13000000] mov eax,msg2
223 46 00000159 E8B1FFFFFF call sprint
224 47 0000015E A1[00000000] mov eax,[max]
225 48 00000163 E81EFFFFFF call iprintf
226 49 00000168 E86EFFFFFF call quit

```

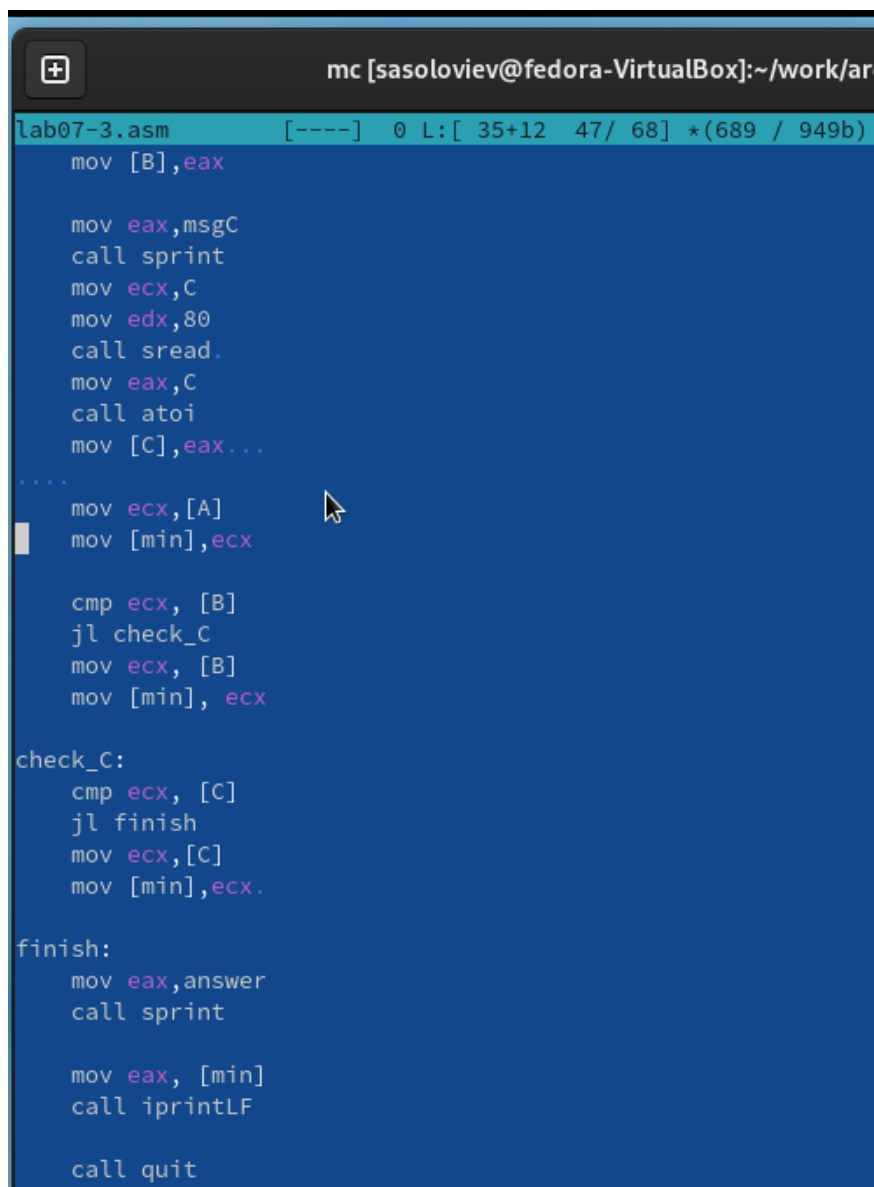
Рис. 2.11: Файл листинга с ошибкой lab7-2

В итоге из-за синтаксической ошибки не удалось сгенерировать объектный файл, но был получен листинг программы, где было указано место возникновения ошибки.

2.1 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

Мой вариант 18 - числа: 83,73,30



```
lab07-3.asm [-----] 0 L: [ 35+12 47/ 68] *(689 / 949b)
    mov [B],eax

    mov eax,msgC
    call sprint
    mov ecx,C
    mov edx,80
    call sread
    mov eax,C
    call atoi
    mov [C],eax...
...
    mov ecx,[A]
    mov [min],ecx

    cmp ecx, [B]
    jl check_C
    mov ecx, [B]
    mov [min], ecx

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: Код программы lab7-3.asm

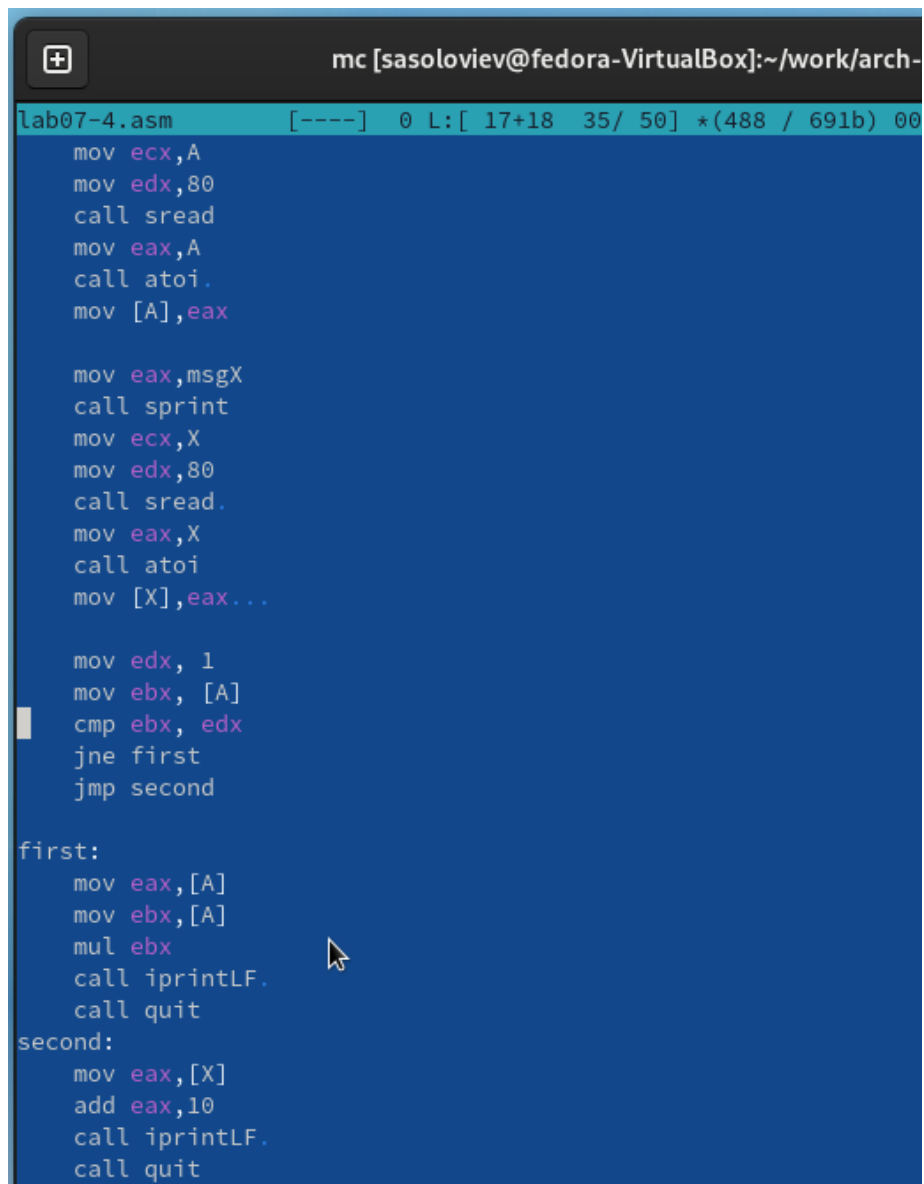
```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-3.asm
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-3.o -o lab07-3
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-3
Input A: 83
Input B: 73
Input C: 30
Smallest: 30
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.13: Компиляция и запуск программы lab7-3.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.

Мой вариант 18

$$\begin{cases} a^2, a \neq 1 \\ 10 + x, a = 1 \end{cases}$$



```
mc [sasoloviev@fedora-VirtualBox]:~/work/arch-  
lab07-4.asm [----] 0 L: [ 17+18 35/ 50] *(488 / 691b) 00  
mov ecx,A  
mov edx,80  
call sread  
mov eax,A  
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...  
  
mov edx, 1  
mov ebx, [A]  
cmp ebx, edx  
jne first  
jmp second  
  
first:  
mov eax,[A]  
mov ebx,[A]  
mul ebx  
call iprintLF.  
call quit  
  
second:  
mov eax,[X]  
add eax,10  
call iprintLF.  
call quit
```

Рис. 2.14: Код программы lab7-4.asm

При $x = 1, a = 2f(x) = 4$

При $x = 2, a = 1f(x) = 12$

```
[sasoloviev@fedora-VirtualBox lab07]$ nasm -f elf lab07-4.asm
[sasoloviev@fedora-VirtualBox lab07]$ ld -m elf_i386 lab07-4.o -o lab07-4
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-4
Input A: 2
Input X: 1
4
[sasoloviev@fedora-VirtualBox lab07]$ ./lab07-4
Input A: 1
Input X: 2
12
[sasoloviev@fedora-VirtualBox lab07]$
```

Рис. 2.15: Компиляция и запуск программы lab7-4.asm

3 Выводы

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