

Отчёт по лабораторной работе 7

Архитектура компьютеров и операционные системы

Алексей Белов НПИбд-01-23

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	17
3	Выводы	21

Список иллюстраций

2.1	Создал каталог и файл	6
2.2	Программа в файле lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	8
2.4	Программа в файле lab7-1.asm	9
2.5	Запуск программы lab7-1.asm	10
2.6	Программа в файле lab7-1.asm	11
2.7	Запуск программы lab7-1.asm	12
2.8	Программа в файле lab7-2.asm	13
2.9	Запуск программы lab7-2.asm	14
2.10	Файл листинга lab7-2	14
2.11	Ошибка трансляции lab7-2	16
2.12	Файл листинга с ошибкой lab7-2	16
2.13	Программа в файле lab7-3.asm	17
2.14	Запуск программы lab7-3.asm	18
2.15	Программа в файле lab7-4.asm	19
2.16	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

Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.
(рис. [2.1])

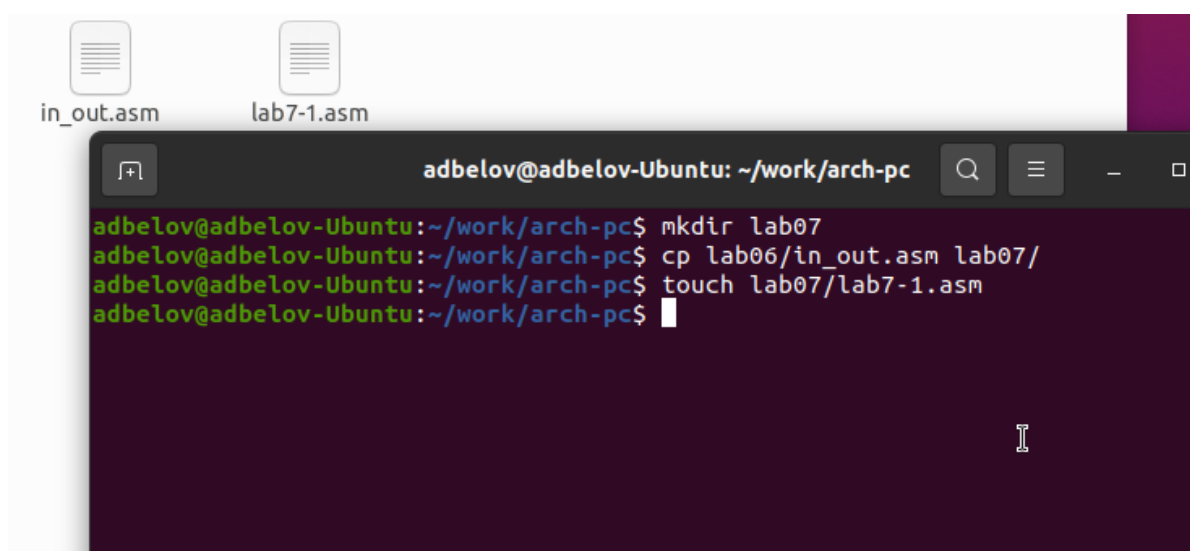
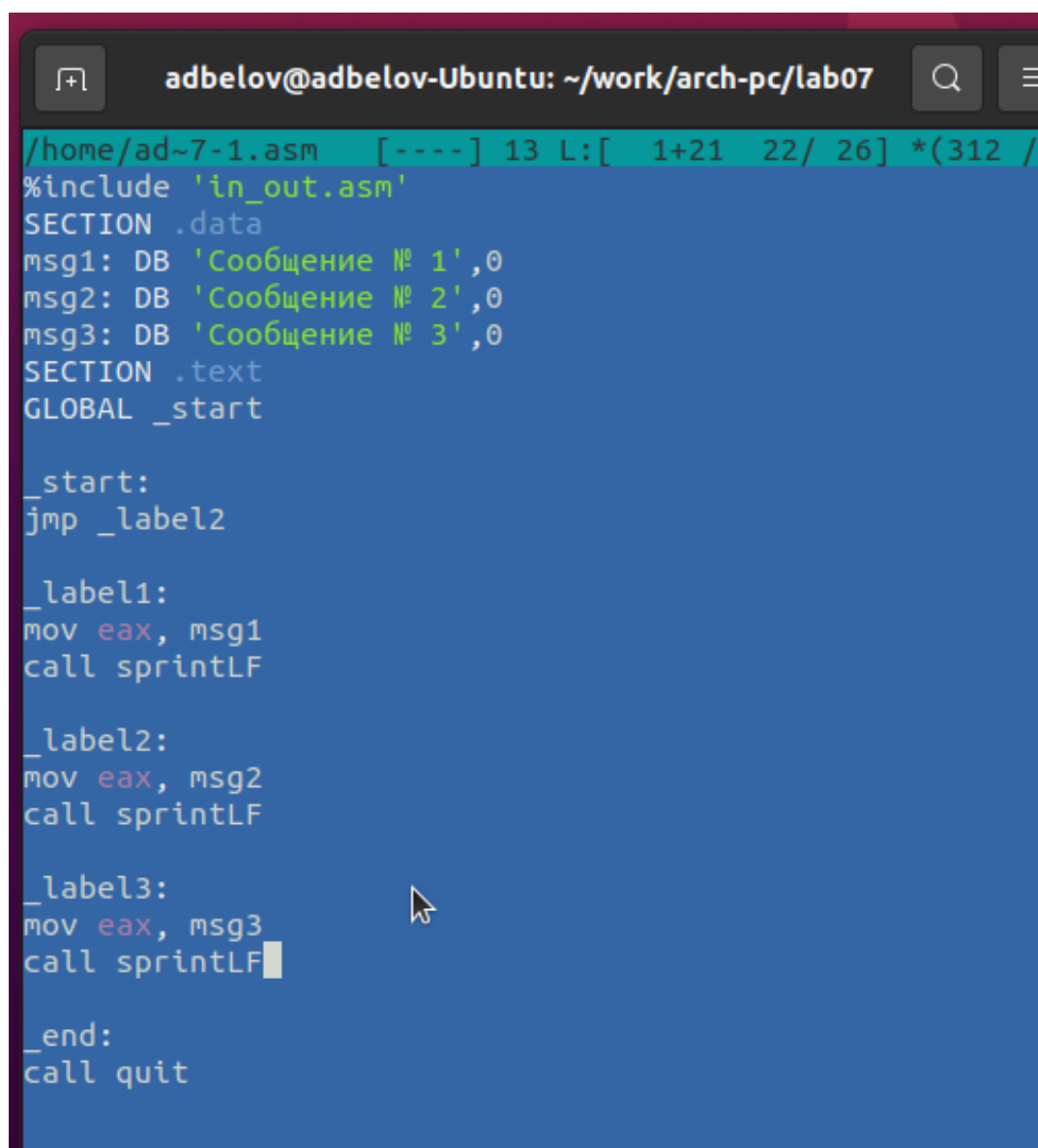


Рис. 2.1: Создал каталог и файл

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. (рис. [2.2])



```
adbelov@adbelov-Ubuntu: ~/work/arch-pc/lab07
/home/ad~7-1.asm [----] 13 L:[ 1+21 22/ 26] *(312 /
#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. 2.2: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его. (рис. [2.3])

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

Рис. 2.3: 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. (рис. [2.4]) (рис. [2.5])


```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/..  
/home/ad~7-1.asm [----] 13 L:[ 1+17 18/ 2  
%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. 2.4: Программа в файле lab7-1.asm

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

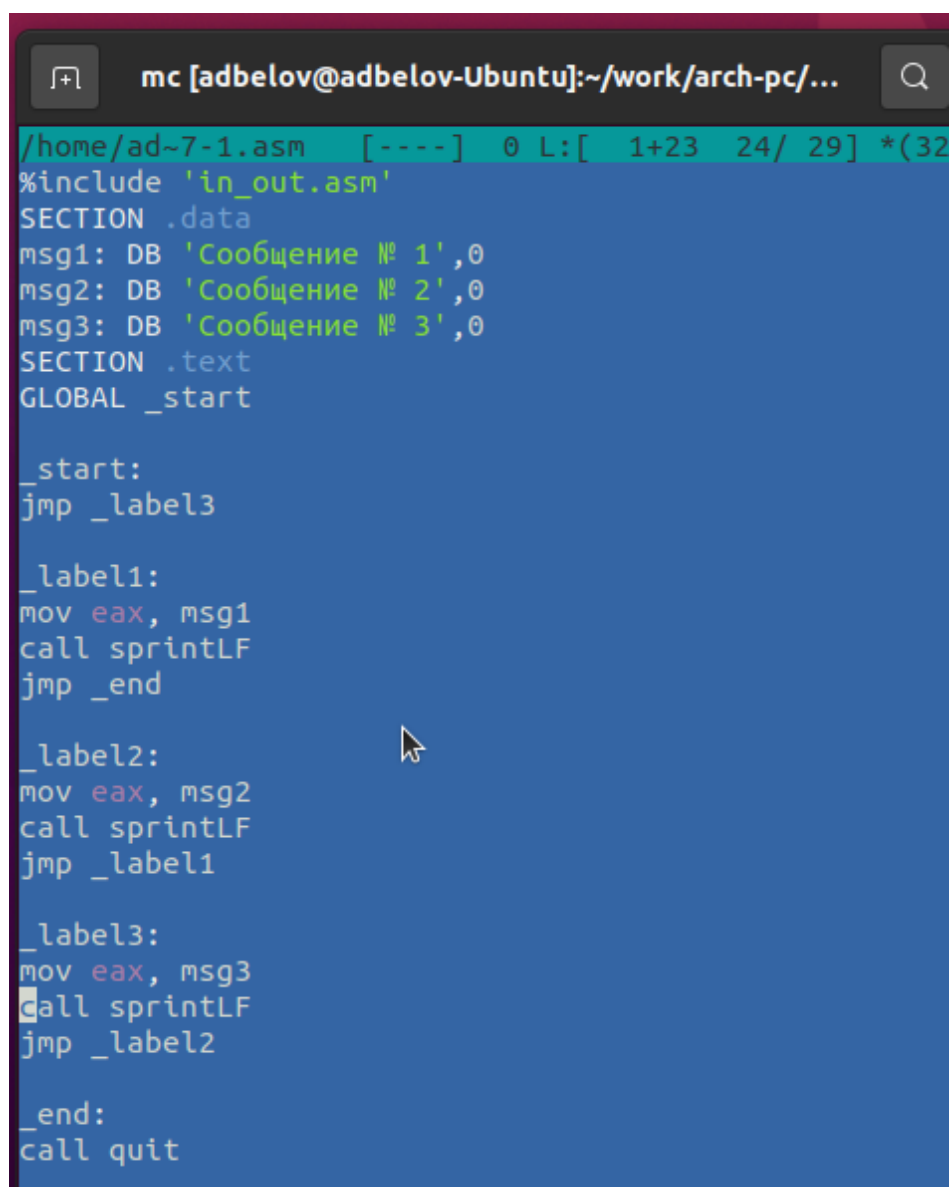
Рис. 2.5: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. [2.6]) (рис. [2.7]):

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/...  
/home/ad~7-1.asm [---] 0 L:[ 1+23 24/ 29] *(32  
%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. 2.6: Программа в файле lab7-1.asm

```

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

```

Рис. 2.7: 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 (рис. [2.8]) (рис. [2.9]).

```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/...
/home/ad~7-2.asm [----] 0 L: [ 19+ 8 27/ 50] *(560 /1057b) 00[*][X
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. 2.8: Программа в файле lab7-2.asm

```

adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ █

```

Рис. 2.9: Запуск программы lab7-2.asm

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 (рис. [2.10])

```

186 10 section .text
187 11 global _start
188 12 _start:
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 E891FFFFFF 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:

```

Рис. 2.10: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 190

- 14 - номер строки в подпрограмме
- 000000E8 - адрес
- B8[00000000] - машинный код
- mov eax,msg1 - код программы - перекладывает msg1 в eax

строка 191

- 15 - номер строки в подпрограмме
- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 193

- 17 - номер строки в подпрограмме
- 000000F2 - адрес
- B9[0A000000] - машинный код
- mov ecx,B - код программы - перекладывает B в ecx

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. [2.11]) (рис. [2.12])

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:30: error: invalid combination of opcode and operands
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
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 E891FFFFFF 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 7F06 jg check_B
206 30 mov ecx,
207 30 ***** error: invalid combination of opcode and operands
208 31 00000124 890D[00000000] mov [max],ecx
209 32 ; ----- Преобразование 'max(A,C)' из символа в число
210 33 check_B:
211 34 0000012A B8[00000000] mov eax,max
212 35 0000012F E868FFFFFF call atoi
213 36 00000134 A3[00000000] mov [max],eax
214 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
215 38 00000139 8B0D[00000000] mov ecx,[max]
216 39 0000013F 3B0D[0A000000] cmp ecx,[B]
217 40 00000145 7F0C jg fin
218 41 00000147 8B0D[0A000000] mov ecx,[B]
219 42 0000014D 890D[00000000] mov [max],ecx
220 43 ; ----- Вывод результата
221 44 fin:
222 45 00000153 B8[13000000] mov eax, msg2
223 46 00000158 E8B2FFFFFF call sprint
224 47 0000015D A1[00000000] mov eax,[max]
225 48 00000162 E81FFFFFFF call iprintLF
226 49 00000167 E86FFFFFFF call quit
```

Рис. 2.12: Файл листинга с ошибкой lab7-2

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

2.1 Самостоятельное задание

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 (рис. [2.13]) (рис. [2.14])

для варианта 8 - 52, 33, 40



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/...
/home/ad~7-3.asm [----] 0 L:[ 38+ 7 45/ 69] *(668 /1060b) 00[*][X]
call sprint
mov ecx,C
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. 2.13: Программа в файле lab7-3.asm

```

adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Input A: 52
Input B: 33
Input C: 40
Smallest: 33
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.14: Запуск программы lab7-3.asm

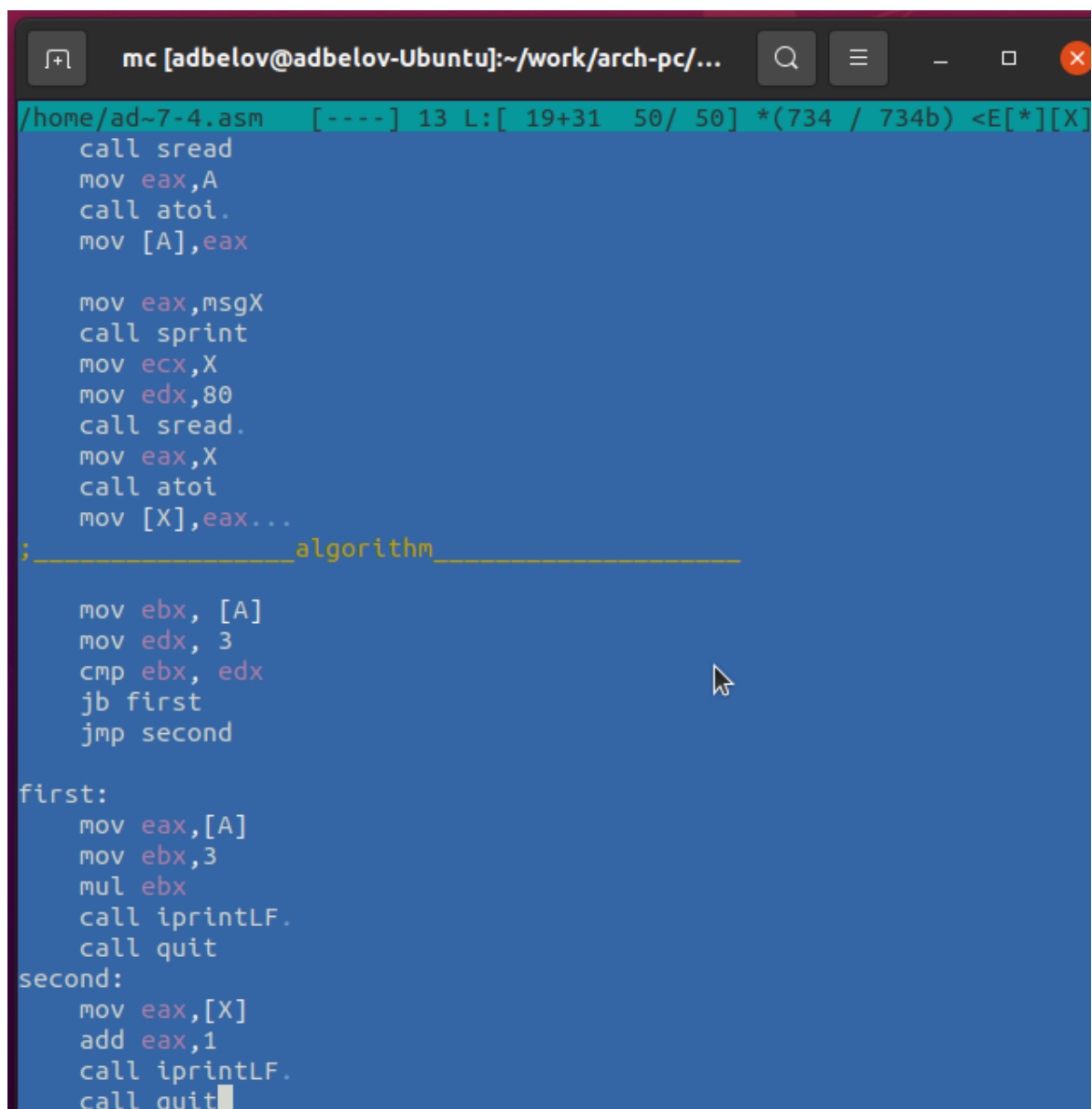
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений X и a из 7.6. (рис. [2.15]) (рис. [2.16])

для варианта 8

$$\begin{cases} 3a, a < 3 \\ x + 1, a \geq 3 \end{cases}$$

Если подставить $x = 1, a = 4$ получается $1 + 1 = 2$.

Если подставить $x = 1, a = 2$ получается $3 * 2 = 6$.



```
mc [adbelov@adbelov-Ubuntu]:~/work/arch-pc/...
/home/ad~7-4.asm [ - - - ] 13 L: [ 19+31 50/ 50] *(734 / 734b) <E[*][X]
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...
; _____algorithm_____

mov ebx, [A]
mov edx, 3
cmp ebx, edx
jb first
jmp second

first:
mov eax,[A]
mov ebx,3
mul ebx
call iprintLF.
call quit

second:
mov eax,[X]
add eax,1
call iprintLF.
call quit
```

Рис. 2.15: Программа в файле lab7-4.asm

```
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Input A: 4
Input X: 1
13
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Input A: 4
Input X: 1
2
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 1
6
adbelov@adbelov-Ubuntu:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-4.asm

3 Выводы

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