

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютерных наук

Литвинов Максим Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	14
2.3	Задание для самостоятельной работы	17
3	Выводы	22

Список иллюстраций

2.1	Изменение кода	7
2.2	Запуск программы	8
2.3	Изменение кода	9
2.4	Запуск программы	10
2.5	Изменение кода	11
2.6	Запуск программы	12
2.7	Изменение кода	13
2.8	Запуск программы	14
2.9	Файл листинга	15
2.10	Ошибка трансляции	16
2.11	Файл листинга	17
2.12	Изменение кода	18
2.13	Запуск программы	19
2.14	Изменение кода	20
2.15	Запуск программы	21

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

Я создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

Инструкция `jmp` в NASM используется для выполнения безусловных переходов. Рассмотрим пример программы, в которой используется инструкция `jmp`. Написал текст программы из листинга 7.1 в файле lab7-1.asm.

```
Открыть ▾ + lab07-1.asm
~/.work/lab07

%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: Изменение кода

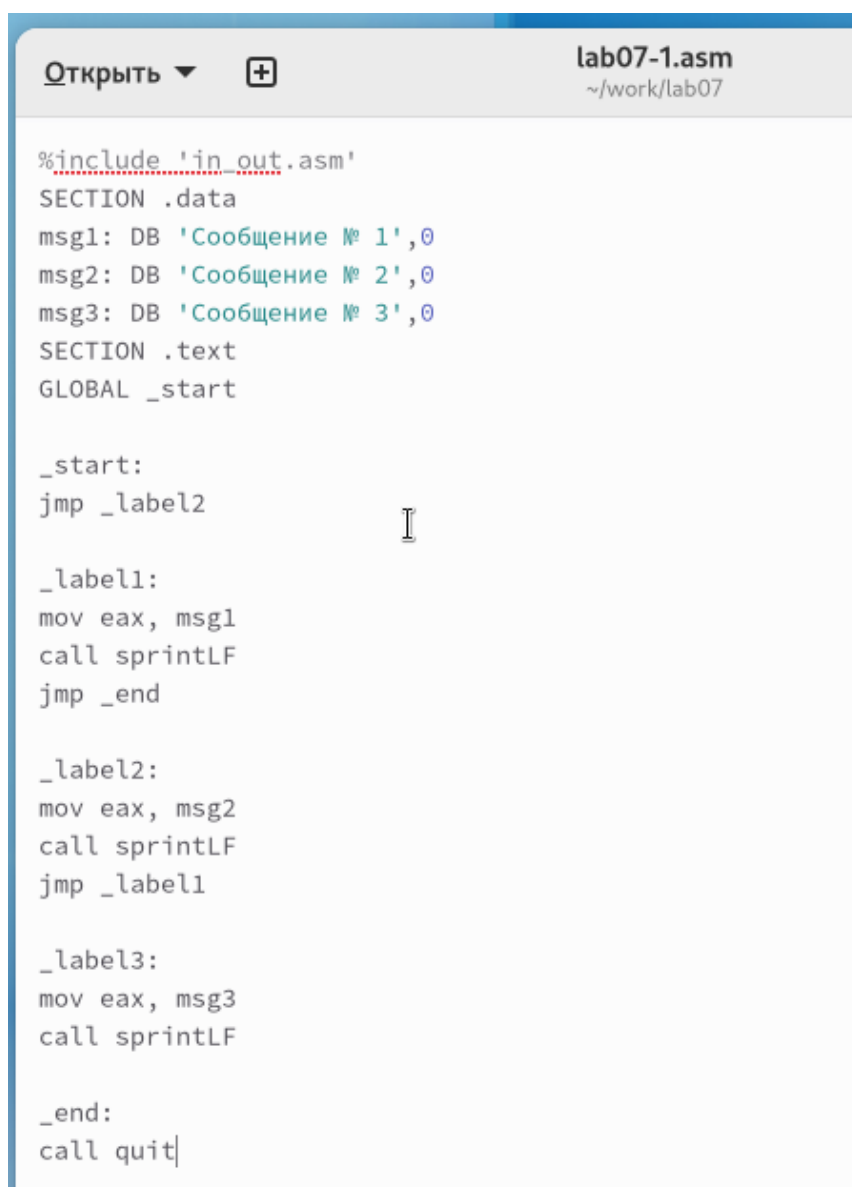
Создал исполняемый файл и запустил его.

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

Рис. 2.2: Запуск программы

Инструкция `jmp` позволяет осуществлять переходы не только вперед, но и назад. Мы изменим программу так, чтобы она сначала выводила “Сообщение № 2”, затем “Сообщение № 1” и завершала работу. Для этого мы добавим в текст программы после вывода “Сообщения № 2” инструкцию `jmp` с меткой `_label1` (чтобы перейти к инструкциям вывода “Сообщения № 1”) и после вывода “Сообщения № 1” добавим инструкцию `jmp` с меткой `_end` (чтобы перейти к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2.



The image shows a screenshot of an assembly code editor. The title bar at the top indicates the file is 'lab07-1.asm' located at '~/.work/lab07'. The editor has a menu bar with 'Открыть' (Open) and a plus icon. The code is written in assembly language and includes comments in Russian. It defines three data segments: 'Сообщение № 1', 'Сообщение № 2', and 'Сообщение № 3'. It then defines three labels: '_label1', '_label2', and '_label3', each followed by instructions to move the message address into 'eax', call 'sprintf', and jump to '_end'. The '_end' label calls 'quit'.

```
%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
```

Рис. 2.3: Изменение кода

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

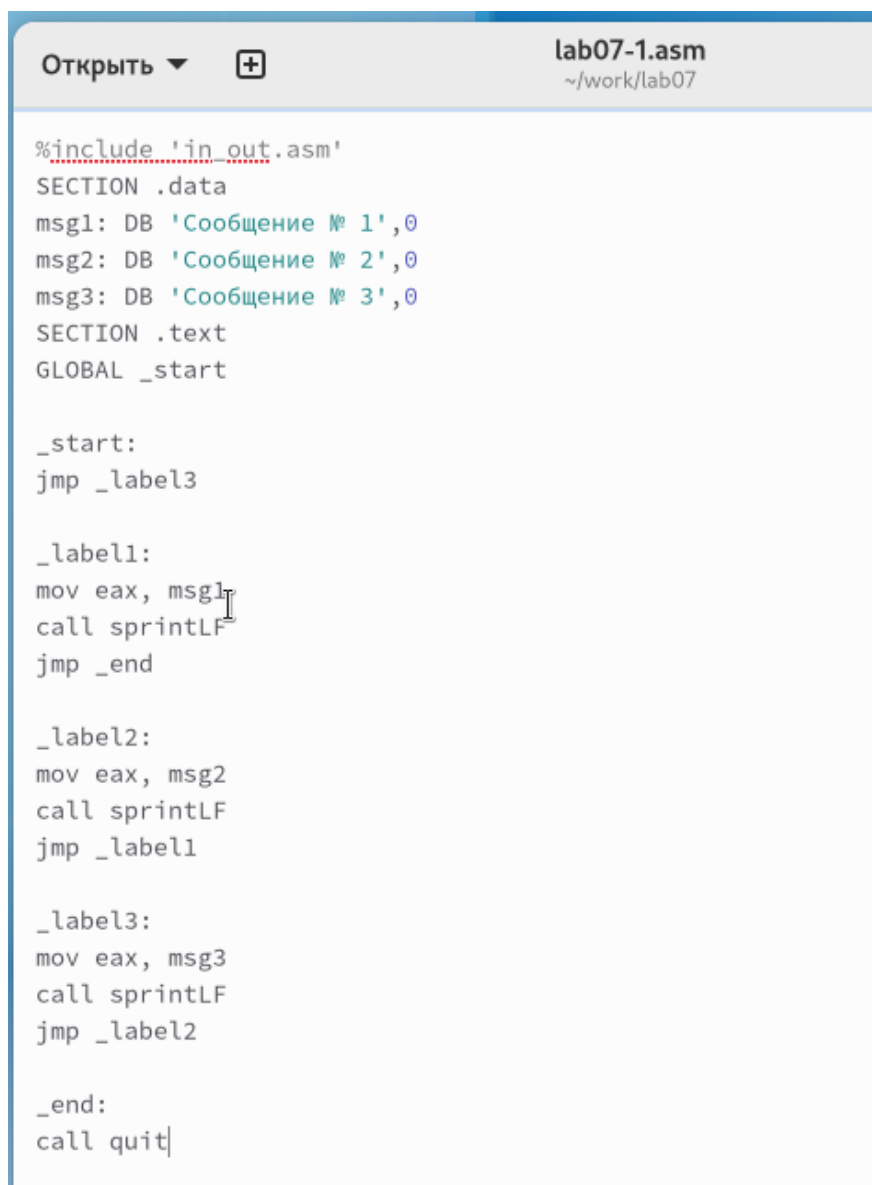
Рис. 2.4: Запуск программы

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
%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: Изменение кода

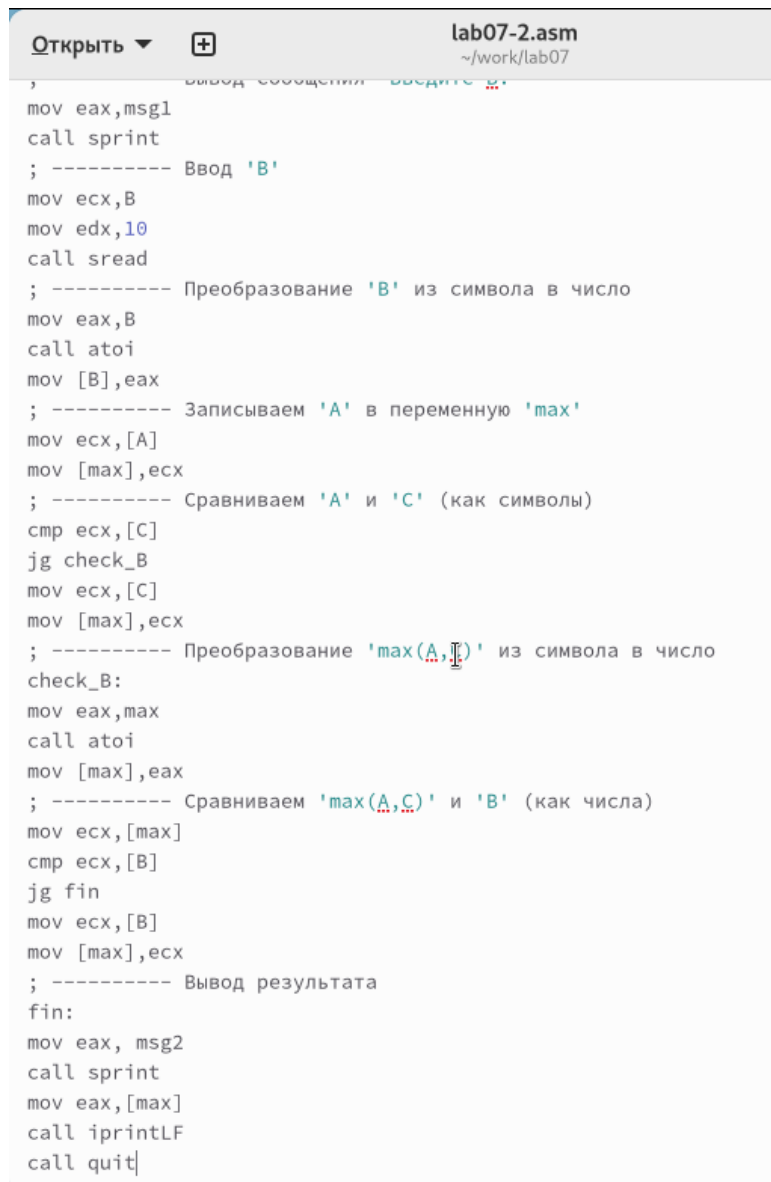
```
[malitvinov@fedora lab07]$  
[malitvinov@fedora lab07]$ nasm -f elf lab07-1.asm  
[malitvinov@fedora lab07]$ ld -m elf_i386 lab07-1.o -o lab07-1  
[malitvinov@fedora lab07]$ ./lab07-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[malitvinov@fedora lab07]$
```

Рис. 2.6: Запуск программы

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, то есть переход должен происходить, если выполнено какое-либо условие.

Давайте рассмотрим программу, которая определяет и выводит на экран наибольшую из трех целочисленных переменных: А, В и С. Значения для А и С задаются в программе, а значение В вводится с клавиатуры.

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



```
lab07-2.asm
~/work/lab07

; ----- Вывод сообщения -----
mov eax,msg1
call sprint
; ----- Ввод 'B' -----
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: Изменение кода

```
[malitvinov@fedora lab07]$ nasm -f elf lab07-2.asm
[malitvinov@fedora lab07]$ ld -m elf_i386 lab07-2.o -o lab07-2
[malitvinov@fedora lab07]$ ./lab07-2
Введите В: 20
Наибольшее число: 50
[malitvinov@fedora lab07]$ ./lab07-2
Введите В: 60
Наибольшее число: 60
[malitvinov@fedora lab07]$
```

Рис. 2.8: Запуск программы

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`

lab07-2.asm			lab07-2.lst
191	16		; ----- Ввод 'B'
192	17	000000F2 B9[0A000000]	mov ecx,B
193	18	000000F7 BA0A000000	mov edx,10
194	19	000000FC E842FFFFFF	call sread
195	20		; ----- Преобразование 'B' из символа в число
196	21	00000101 B8[0A000000]	mov eax,B
197	22	00000106 E891FFFFFF	call atoi
198	23	0000010B A3[0A000000]	mov [B],eax
199	24		; ----- Записываем 'A' в переменную 'max'
200	25	00000110 8B0D[35000000]	mov ecx,[A]
201	26	00000116 890D[00000000]	mov [max],ecx
202	27		; ----- Сравниваем 'A' и 'C' (как символы)
203	28	0000011C 3B0D[39000000]	cmp ecx,[C]
204	29	00000122 7F0C	jg check_B
205	30	00000124 8B0D[39000000]	mov ecx,[C]
206	31	0000012A 890D[00000000]	mov [max],ecx
207	32		; ----- Преобразование 'max(A,C)' из символа в число
208	33		check_B:
209	34	00000130 B8[00000000]	mov eax,max
210	35	00000135 E862FFFFFF	call atoi
211	36	0000013A A3[00000000]	mov [max],eax
212	37		; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
213	38	0000013F 8B0D[00000000]	mov ecx,[max]
214	39	00000145 3B0D[0A000000]	cmp ecx,[B]
215	40	0000014B 7F0C	jg fin
216	41	0000014D 8B0D[0A000000]	mov ecx,[B]
217	42	00000153 890D[00000000]	mov [max],ecx
218	43		; ----- Вывод результата
219	44		fin:
220	45	00000159 B8[13000000]	mov eax,msg2
221	46	0000015E E8ACFFFFFF	call sprint
222	47	00000163 A1[00000000]	mov eax,[max]
223	48	00000168 E819FFFFFF	call iprintf
224	49	0000016D E869FFFFFF	call quit

Рис. 2.9: Файл листинга

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

строка 203

- 28 - номер строки в подпрограмме *
- 0000011C - адрес *
- 3B0D[39000000] - машинный код *
- str ecx,[C] - код программы - сравнивает ecx и переменную *

строка 204

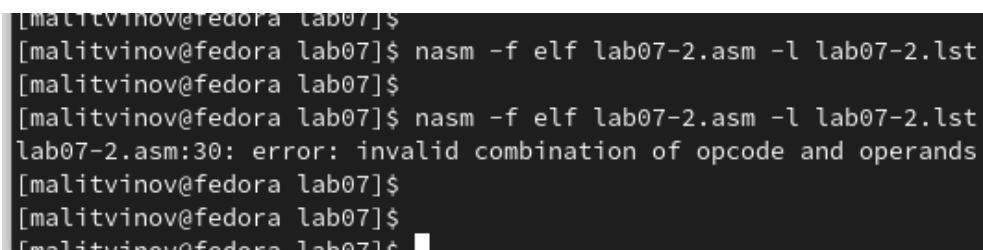
- 29 - номер строки в подпрограмме *
- 00000122 - адрес *

- 7F0C - машинный код *
- `jb check_B` - код программы - если сравнение покажет что одно число больше то переход к метке `check_B`*

строка 205

- 30 - номер строки в подпрограмме *
- 00000124 - адрес *
- `8B0D[39000000]` - машинный код *
- `mov ecx,[C]` - код программы - копирует переменную C в `ecx` *

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



```
[malitvinov@fedora lab07]$
[malitvinov@fedora lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
[malitvinov@fedora lab07]$
[malitvinov@fedora lab07]$ nasm -f elf lab07-2.asm -l lab07-2.lst
lab07-2.asm:30: error: invalid combination of opcode and operands
[malitvinov@fedora lab07]$
[malitvinov@fedora lab07]$
[malitvinov@fedora lab07]$
```

Рис. 2.10: Ошибка трансляции



```
Открыть ▾  lab07-2.lst ~\work\lab07 Стр. 1
lab07-2.asm lab07-2.lst
192 17 000000F2 B8[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC E842FFFFFF call spread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 E891FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8B0D[35000000] mov ecx,[A]
201 26 00000116 890D[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B0D[39000000] cmp ecx,[C]
204 29 00000122 7F06 jg check_B
205 30 mov ecx
206 30 ***** error: invalid combination of opcode and operands
207 31 00000124 890D[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33
210 34 0000012A B8[00000000] mov eax,max
211 35 0000012F E868FFFFFF call atoi
212 36 00000134 A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 00000139 8B0D[00000000] mov ecx,[max]
215 39 0000013F 3B0D[0A000000] cmp ecx,[B]
216 40 00000145 7F0C jg fin
217 41 00000147 8B0D[0A000000] mov ecx,[B]
218 42 0000014D 890D[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000153 B8[13000000] mov eax,msg2
222 46 00000158 E8B2FFFFFF call sprint
223 47 0000015D A3[00000000] mov eax,[max]
224 48 00000162 E81FFFFFFF call iprintLF
225 49 00000167 E86FFFFFFF call quit
```

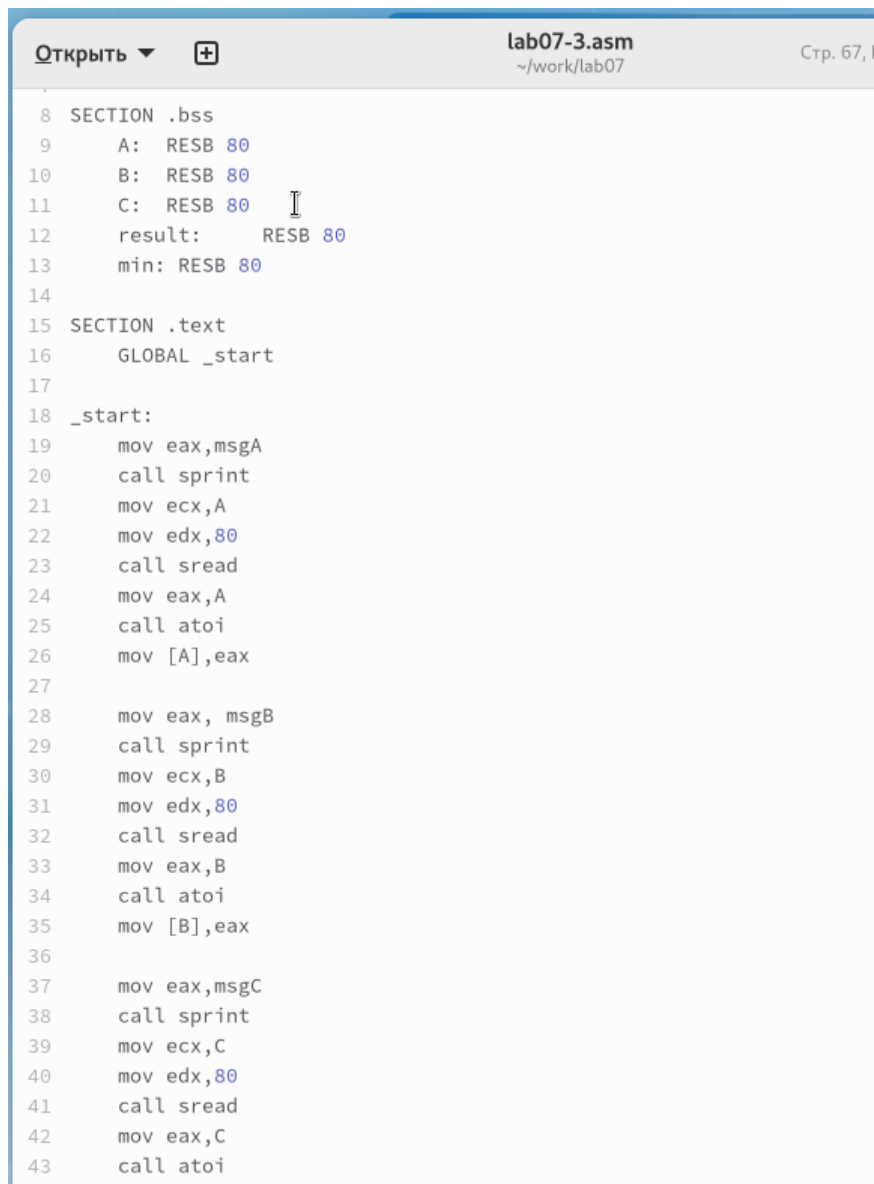
Рис. 2.11: Файл листинга

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 13 - числа: 84,32,77



```
8 SECTION .bss
9     A: RESB 80
10    B: RESB 80
11    C: RESB 80
12    result: RESB 80
13    min: RESB 80
14
15 SECTION .text
16     GLOBAL _start
17
18 _start:
19     mov eax,msgA
20     call sprint
21     mov ecx,A
22     mov edx,80
23     call sread
24     mov eax,A
25     call atoi
26     mov [A],eax
27
28     mov eax, msgB
29     call sprint
30     mov ecx,B
31     mov edx,80
32     call sread
33     mov eax,B
34     call atoi
35     mov [B],eax
36
37     mov eax,msgC
38     call sprint
39     mov ecx,C
40     mov edx,80
41     call sread
42     mov eax,C
43     call atoi
```

Рис. 2.12: Изменение кода

```
[malitvinov@fedora lab07]$  
[malitvinov@fedora lab07]$ nasm -f elf lab07-3.asm  
[malitvinov@fedora lab07]$ ld -m elf_i386 lab07-3.o -o lab07-3  
[malitvinov@fedora lab07]$ ./lab07-3  
Input A: 84  
Input B: 32  
Input C: 77  
Smallest: 32  
[malitvinov@fedora lab07]$
```

Рис. 2.13: Запуск программы

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 13

$$\begin{cases} a - 8, a \geq 7 \\ ax, a < 7 \end{cases}$$



```
lab07-4.asm
~/work/lab07
Стр. 47, По

13
14 _start:
15     mov eax,msgA
16     call sprintf
17     mov ecx,A
18     mov edx,80
19     call sread
20     mov eax,A
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
25     call sprintf
26     mov ecx,X
27     mov edx,80
28     call sread
29     mov eax,X
30     call atoi
31     mov [X],eax
32
33     mov edx, 7
34     mov ebx, [A]
35     cmp ebx, edx
36     jge first
37     jmp second
38
39 first:
40     mov eax,[A]
41     sub eax, 8
42     call iprintLF
43     call quit
44 second:
45     mov eax,[A]
46     mov ebx,[X]
47     mul ebx
48     call iprintLF
49     call quit
```

Рис. 2.14: Изменение кода

```
[malitvinov@fedora lab07]$  
[malitvinov@fedora lab07]$ nasm -f elf lab07-4.asm  
[malitvinov@fedora lab07]$ ld -m elf_i386 lab07-4.o -o lab07-4  
[malitvinov@fedora lab07]$ ./lab07-4  
Input A: 9  
Input X: 3  
1  
[malitvinov@fedora lab07]$ ./lab07-4  
Input A: 4  
Input X: 6  
24  
[malitvinov@fedora lab07]$
```

Рис. 2.15: Запуск программы

3 Выводы

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