

Отчёт по лабораторной работе 7

**Компьютерные науки и технологии программирования. Раздел
Архитектура компьютеров**

Фахми Джакси Гамал Адли

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

Список иллюстраций

2.1	Программа lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	7
2.3	Программа lab7-1.asm	8
2.4	Запуск программы lab7-1.asm	9
2.5	Программа lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	10
2.7	Программа 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 lab7-3.asm	16
2.13	Запуск программы lab7-3.asm	16
2.14	Программа lab7-4.asm	18
2.15	Запуск программы lab7-4.asm	19

Список таблиц

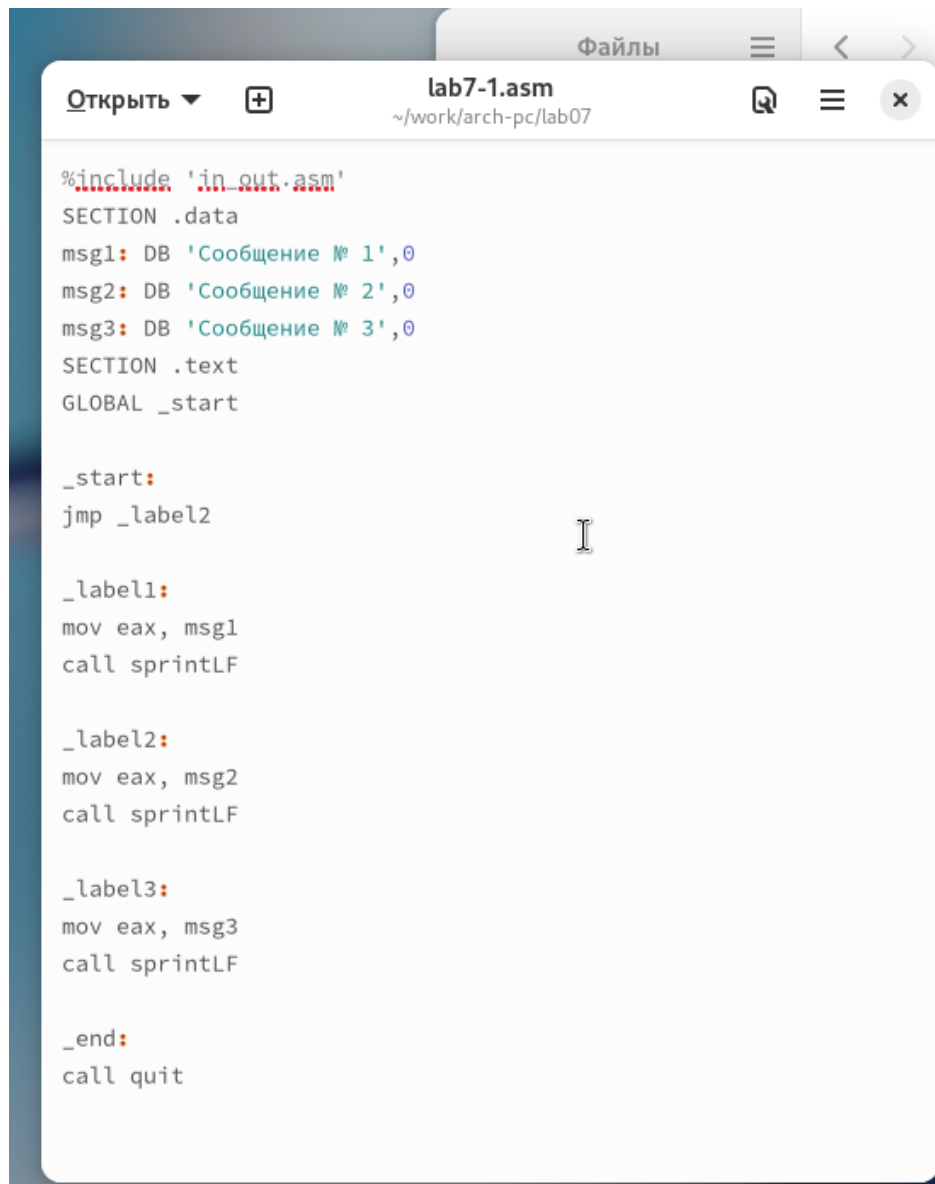
1 Цель работы

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

1. Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm
2. 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.



```
lab7-1.asm
~/work/arch-pc/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

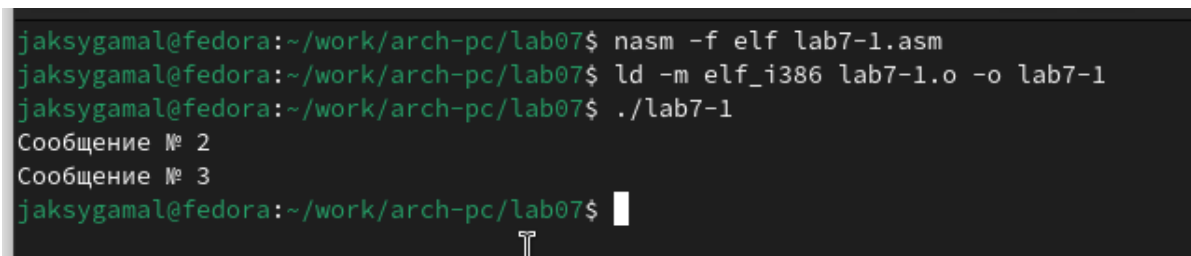
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.1: Программа lab7-1.asm

Создал исполняемый файл и запустил его.

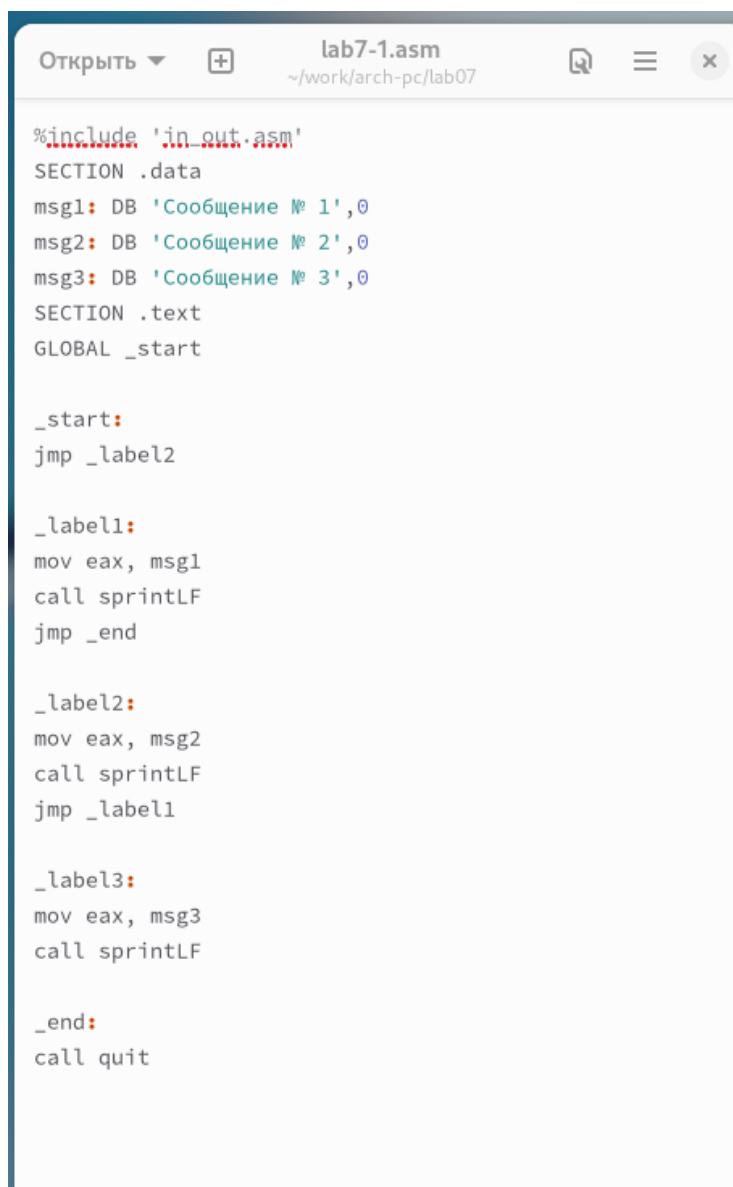


```
jaksygamal@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
jaksygamal@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
jaksygamal@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
jaksygamal@fedora:~/work/arch-pc/lab07$
```

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.



```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/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
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Программа lab7-1.asm


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

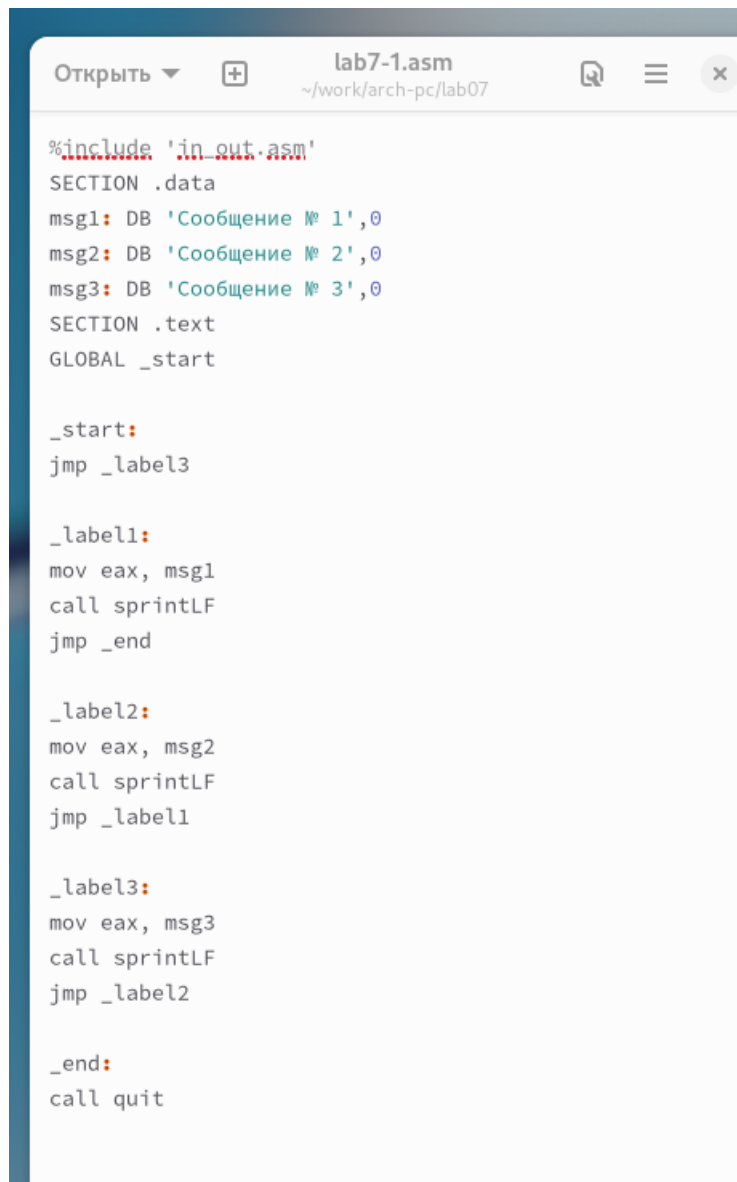
Рис. 2.4: Запуск программы lab7-1.asm

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

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/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 _label3

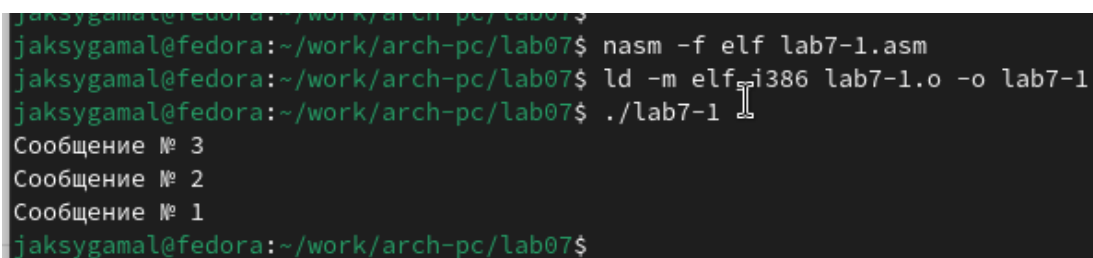
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.5: Программа lab7-1.asm



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

Рис. 2.6: Запуск программы lab7-1.asm

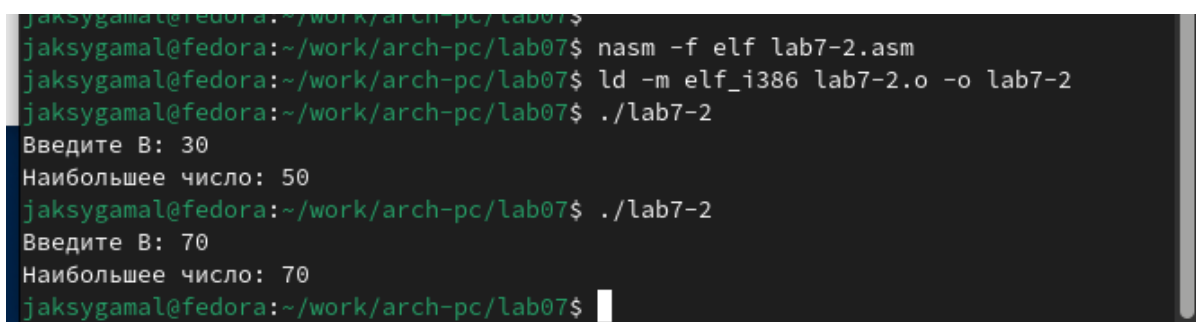
3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую 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.



```
Открыть + lab7-2.asm ~/work/arch-pc/lab07
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
```

Рис. 2.7: Программа lab7-2.asm

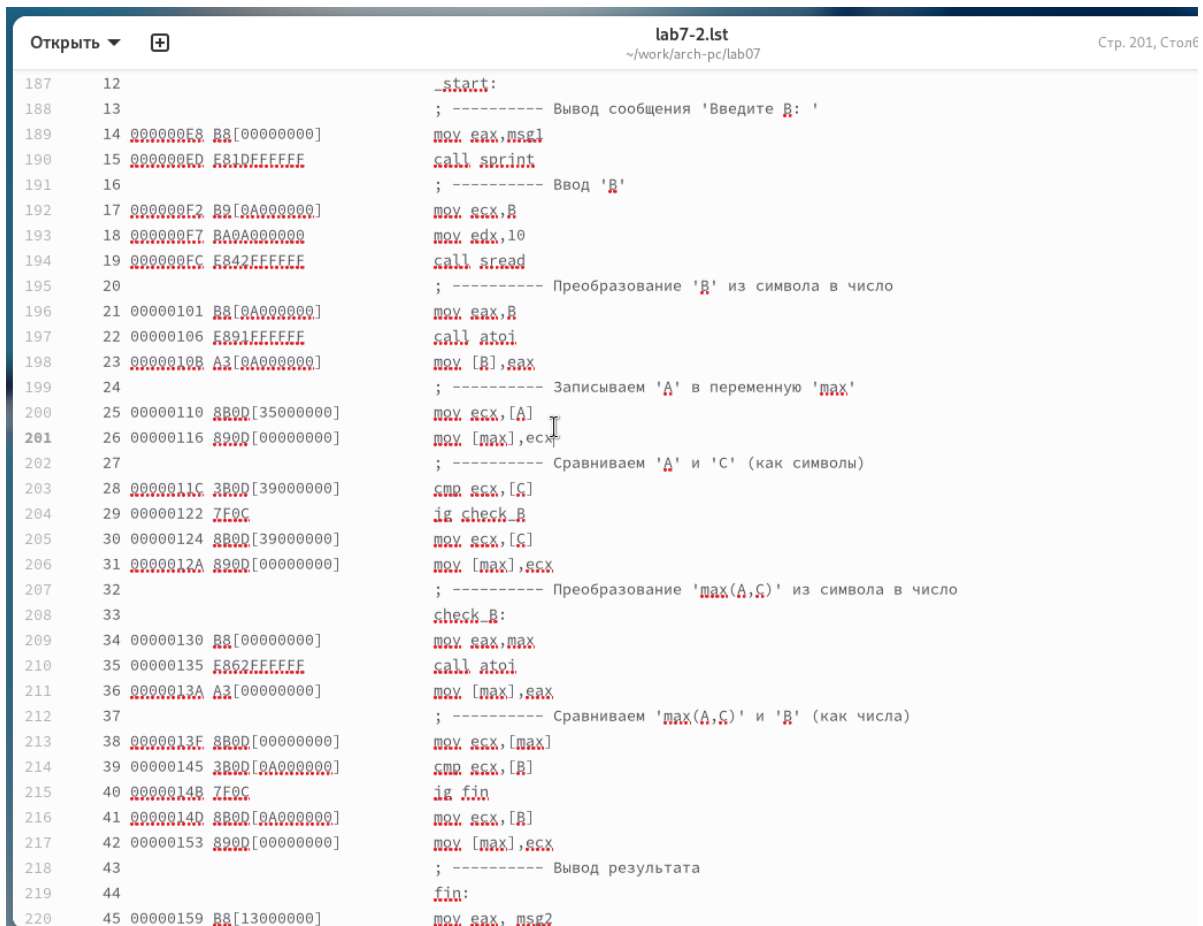


```
jaksygamal@fedora:~/work/arch-pc/lab07$
jaksygamal@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
jaksygamal@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
jaksygamal@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
jaksygamal@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
jaksygamal@fedora:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск программы lab7-2.asm

4. 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`



```
187 12      _start:
188 13      ; ----- Вывод сообщения 'Введите R: '
189 14  000000F8  B8[00000000]      mov eax,msg1
190 15  000000FD  F81DFFFFFF      call sprintf
191 16      ; ----- Ввод 'R'
192 17  000000E2  B9[0A000000]      mov ecx,R
193 18  000000F7  BA0A000000      mov edx,10
194 19  000000FC  F842FFFFFF      call scanf
195 20      ; ----- Преобразование 'R' из символа в число
196 21  00000101  B8[0A000000]      mov eax,R
197 22  00000106  F801FFFFFF      call atoi
198 23  0000010B  A3[0A000000]      mov [R],eax
199 24      ; ----- Записываем 'A' в переменную 'max'
200 25  00000110  8B0D[35000000]      mov ecx,[A]
201 26  00000116  8B0D[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_R
205 30  00000124  8B0D[39000000]      mov ecx,[C]
206 31  0000012A  8B0D[00000000]      mov [max],ecx
207 32      ; ----- Преобразование 'max(A,C)' из символа в число
208 33      check_R:
209 34  00000130  B8[00000000]      mov eax,max
210 35  00000135  F852FFFFFF      call atoi
211 36  0000013A  A3[00000000]      mov [max],eax
212 37      ; ----- Сравниваем 'max(A,C)' и 'R' (как числа)
213 38  0000013F  8B0D[00000000]      mov ecx,[max]
214 39  00000145  3B0D[0A000000]      cmp ecx,[R]
215 40  0000014B  7F0C              jg fin
216 41  0000014D  8B0D[0A000000]      mov ecx,[R]
217 42  00000153  8B0D[00000000]      mov [max],ecx
218 43      ; ----- Вывод результата
219 44      fin:
220 45  00000159  B8[13000000]      mov eax,msg2
```

Рис. 2.9: Файл листинга lab7-2

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

строка 200

- 25 - номер строки
- 00000110 - адрес

- 8B0D[35000000] - машинный код
- mov esx,[A] - код программы

строка 201

- 26 - номер строки
- 00000116 - адрес
- 890D[00000000] - машинный код
- mov [max],esx - код программы

строка 203

- 28 - номер строки
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- cmp esx,[C] - код программы

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

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

Рис. 2.10: Ошибка трансляции lab7-2

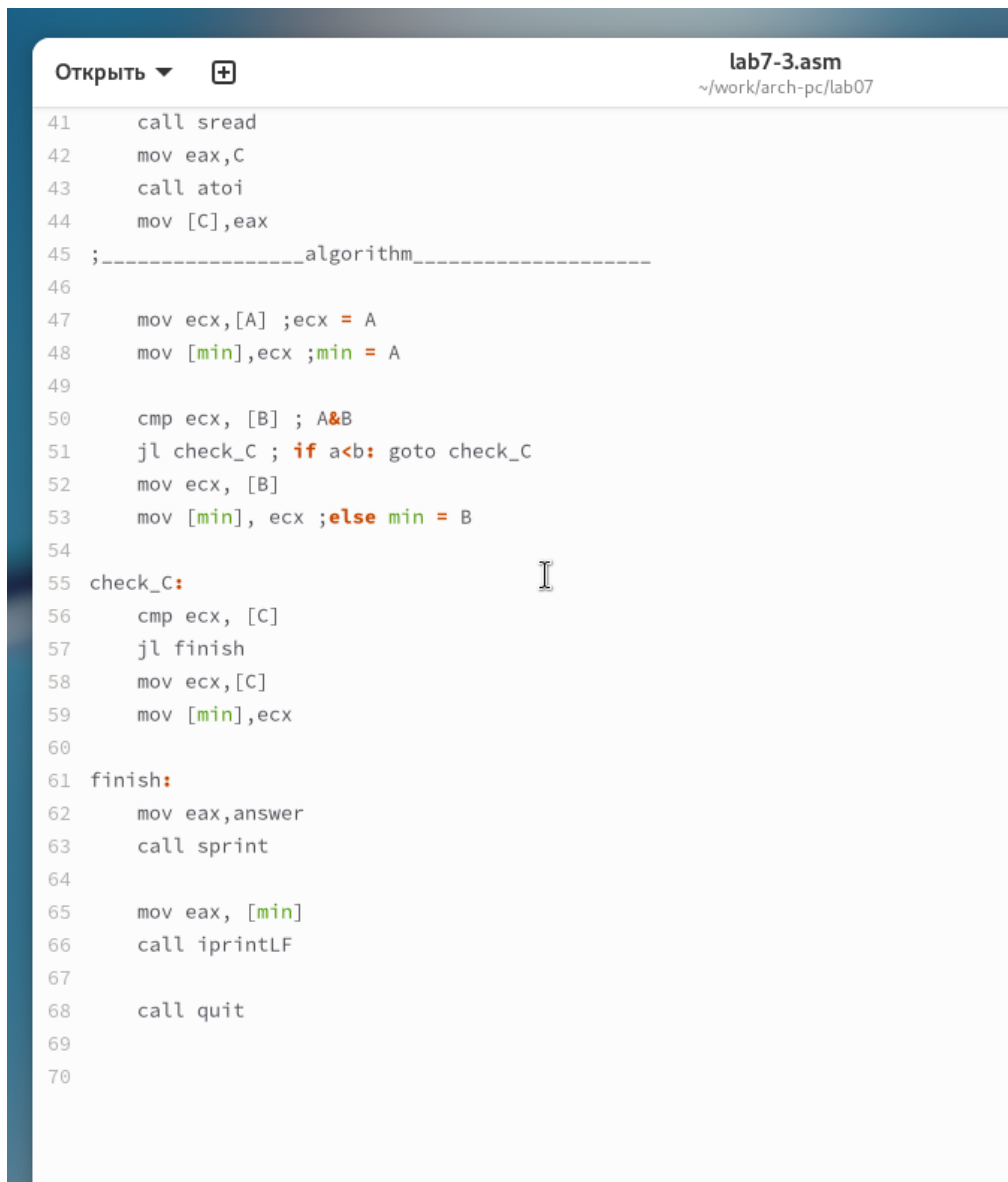
```
Открыть + lab7-2.lst ~\work\arch-pc\lab07 Стр. 1, Столб. 1
189 14 000000F8 B8[00000000] mov eax,msg1
190 15 000000FD F81DFFFFFF call sprint
191 16 ; ----- Ввод 'B'
192 17 000000F2 B8[0A000000] mov ecx,B
193 18 000000F7 BA0A000000 mov edx,10
194 19 000000FC F842FFFFFF call sread
195 20 ; ----- Преобразование 'B' из символа в число
196 21 00000101 B8[0A000000] mov eax,B
197 22 00000106 F891FFFFFF call atoi
198 23 0000010B A3[0A000000] mov [B],eax
199 24 ; ----- Записываем 'A' в переменную 'max'
200 25 00000110 8800[35000000] mov ecx,[A]
201 26 00000116 8900[00000000] mov [max],ecx
202 27 ; ----- Сравниваем 'A' и 'C' (как символы)
203 28 0000011C 3B00[39000000] cmp ecx,[C]
204 29 00000122 7F06 ig check_B
205 30 mov ecx,
206 30 ***** error: invalid combination of opcode and operands
207 31 00000124 8900[00000000] mov [max],ecx
208 32 ; ----- Преобразование 'max(A,C)' из символа в число
209 33 check_B:
210 34 0000012A B8[00000000] mov eax,max
211 35 0000012F F868FFFFFF call atoi
212 36 00000134 A3[00000000] mov [max],eax
213 37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
214 38 00000139 8800[00000000] mov ecx,[max]
215 39 0000013F 3B00[0A000000] cmp ecx,[B]
216 40 00000145 7F0C ig fin
217 41 00000147 8800[0A000000] mov ecx,[B]
218 42 0000014D 8900[00000000] mov [max],ecx
219 43 ; ----- Вывод результата
220 44 fin:
221 45 00000153 B8[13000000] mov eax,msg2
```

Рис. 2.11: Файл листинга с ошибкой lab7-2

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

5. 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

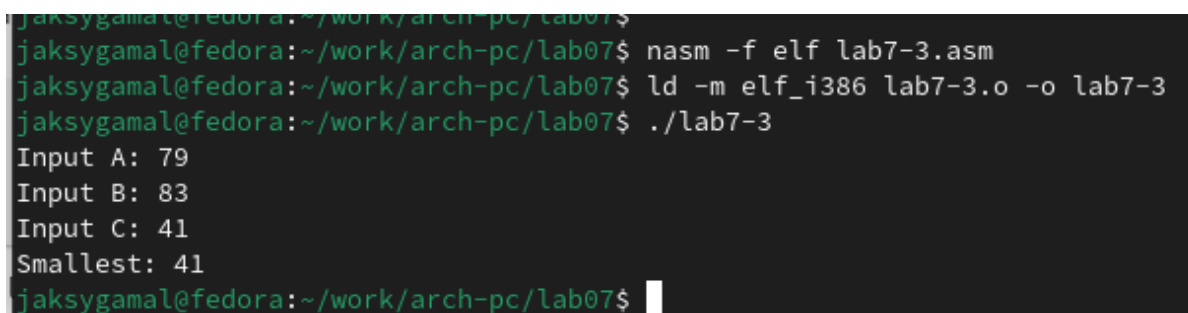
для варианта 6 - 79,83,41



```
lab7-3.asm
~/work/arch-pc/lab07

41  call sread
42  mov  eax,C
43  call atoi
44  mov  [C],eax
45  ;-----algorithm-----
46
47  mov  ecx,[A] ;ecx = A
48  mov  [min],ecx ;min = A
49
50  cmp  ecx, [B] ; A&B
51  jl   check_C ; if a<b: goto check_C
52  mov  ecx, [B]
53  mov  [min], ecx ;else min = B
54
55  check_C:
56  cmp  ecx, [C]
57  jl   finish
58  mov  ecx,[C]
59  mov  [min],ecx
60
61  finish:
62  mov  eax,answer
63  call sprint
64
65  mov  eax, [min]
66  call iprintLF
67
68  call quit
69
70
```

Рис. 2.12: Программа lab7-3.asm




```
jaksygamal@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
jaksygamal@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
jaksygamal@fedora:~/work/arch-pc/lab07$ ./lab7-3
Input A: 79
Input B: 83
Input C: 41
Smallest: 41
jaksygamal@fedora:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск программы lab7-3.asm

6. 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 6

$$\begin{cases} x + a, x = a \\ 5x, x \neq a \end{cases}$$

Открыть ▾ 

lab7-4.asm
~/work/arch-pc/lab07

```
22     mov [A],eax
23
24     mov eax,msgX
25     call sprint
26     mov ecx,X
27     mov edx,80
28     call sread
29     mov eax,X
30     call atoi
31     mov [X],eax
32 ;-----algorithm-----
33
34     mov ebx, [X]
35     mov edx, [A]
36     cmp ebx, edx
37     je first
38     jmp second
39
40 first:
41     mov eax,[X]
42     add eax,[A]
43     call iprintLF
44     call quit
45 second:
46     mov eax,[X]
47     mov ebx,5
48     mul ebx
49     call iprintLF
50     call quit
51
52
```

Рис. 2.14: Программа lab7-4.asm

```
jaksygama@fedora:~/work/arch-pc/lab07$  
jaksygama@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
jaksygama@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
jaksygama@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 2  
4  
jaksygama@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 1  
Input X: 2  
10  
jaksygama@fedora:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

3 Выводы

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