

Отчёт по лабораторной работе 7

дисциплина: Архитектура компьютера

Чернятин Артём Андреевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	13
2.3	Самостоятельное задание	15
3	Выводы	20

Список иллюстраций

2.1	Создан каталог	6
2.2	Программа lab7-1.asm	7
2.3	Запуск программы lab7-1.asm	8
2.4	Программа lab7-1.asm	8
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	10
2.8	Программа lab7-2.asm	12
2.9	Запуск программы lab7-2.asm	13
2.10	Файл листинга lab7-2	13
2.11	Ошибка трансляции lab7-2	14
2.12	Файл листинга с ошибкой lab7-2	15
2.13	Программа lab7-3.asm	16
2.14	Запуск программы lab7-3.asm	16
2.15	Программа lab7-4.asm	18
2.16	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

2.1 Реализация переходов в NASM

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

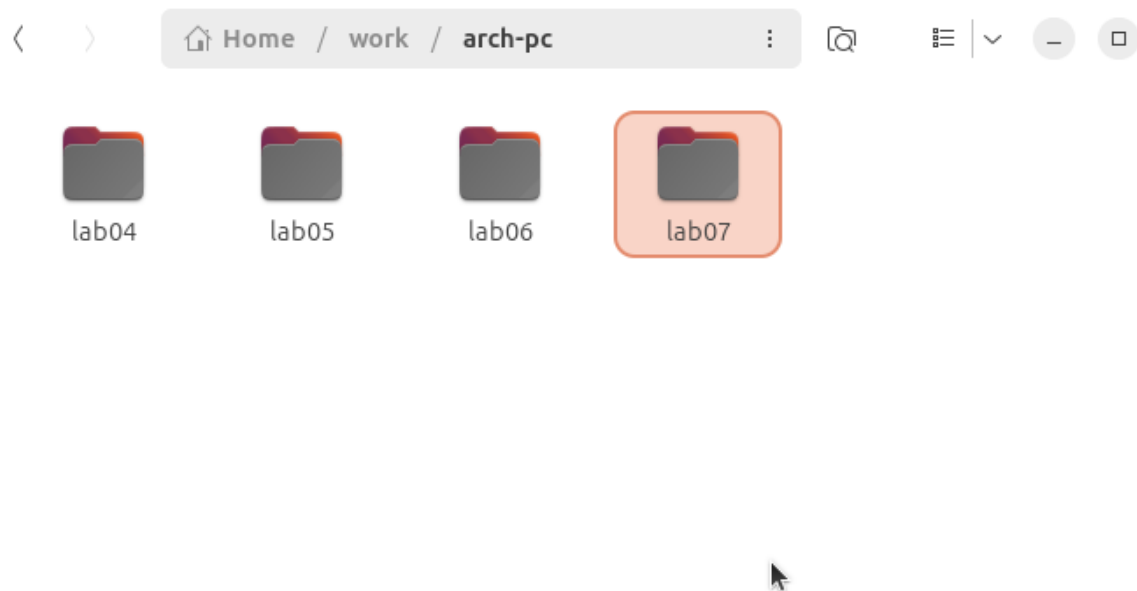
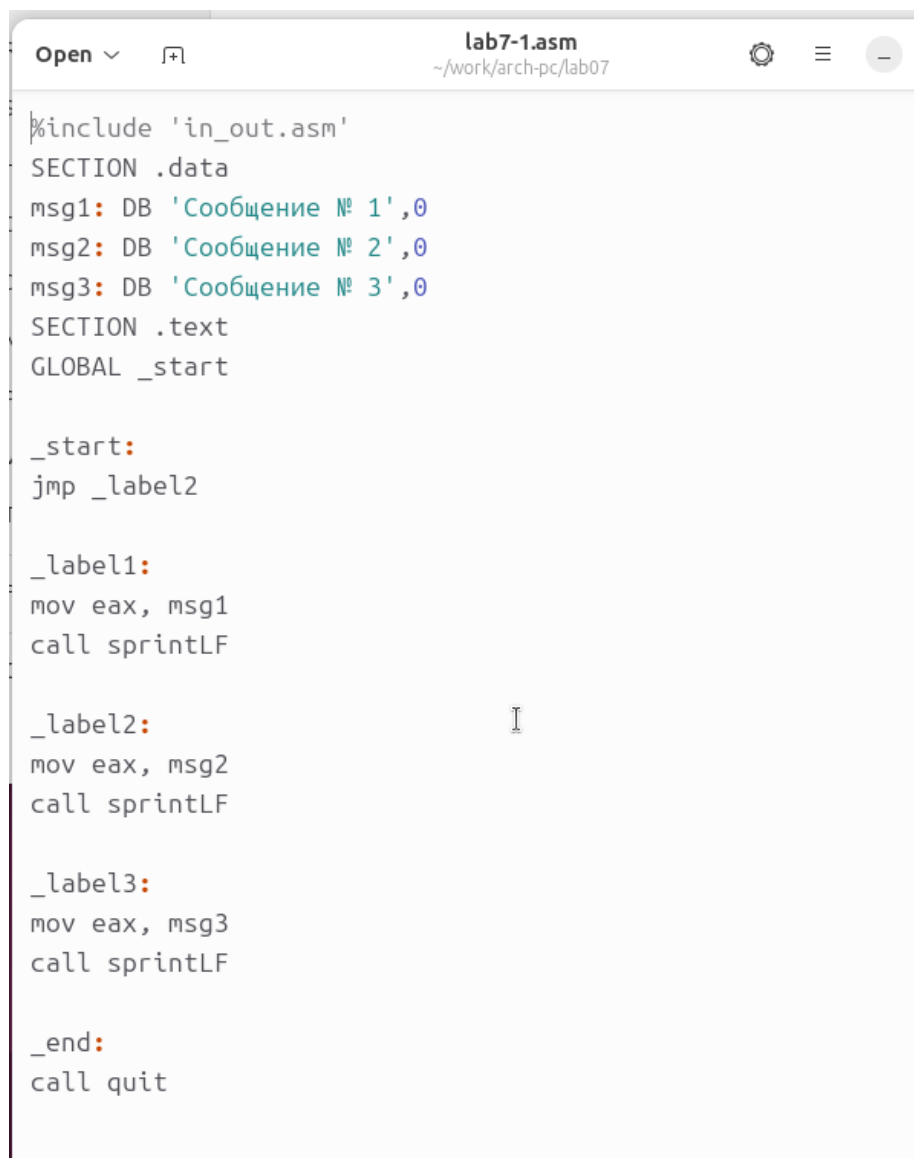


Рисунок 2.1: Создан каталог

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)



```
Open ▾ [icon] 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
```

Рисунок 2.2: Программа lab7-1.asm

Создаю исполняемый файл и запускаю его. (рис. 2.3)

```
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1.1
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1.1
Сообщение № 2
Сообщение № 3
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
```

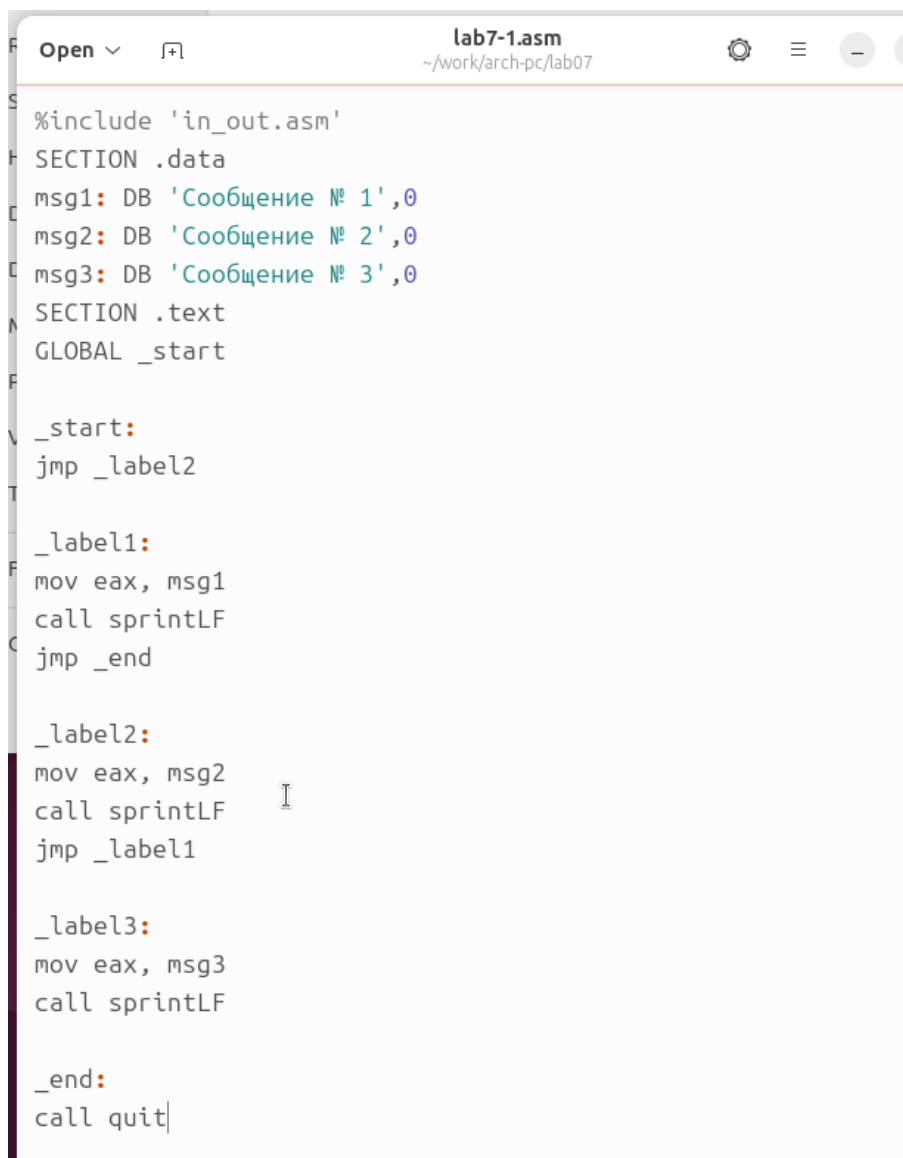
Рисунок 2.3: Запуск программы lab7-1.asm

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

Изменяю текст программы в соответствии с листингом 7.2. (рис. 2.4) (рис. 2.5)

```
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1.1
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1.1
Сообщение № 2
Сообщение № 3
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.4: Программа lab7-1.asm



```
Open ▾ [icon] 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
```

Рисунок 2.5: Запуск программы lab7-1.asm

Изменил текст программы, чтобы вывод программы был следующим (рис. 2.6) (рис. 2.7):

Сообщение № 3
Сообщение № 2
Сообщение № 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 _label3

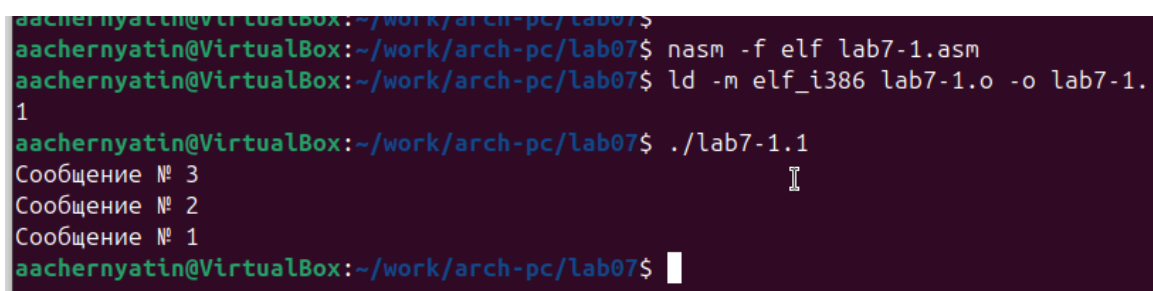
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рисунок 2.6: Программа lab7-1.asm

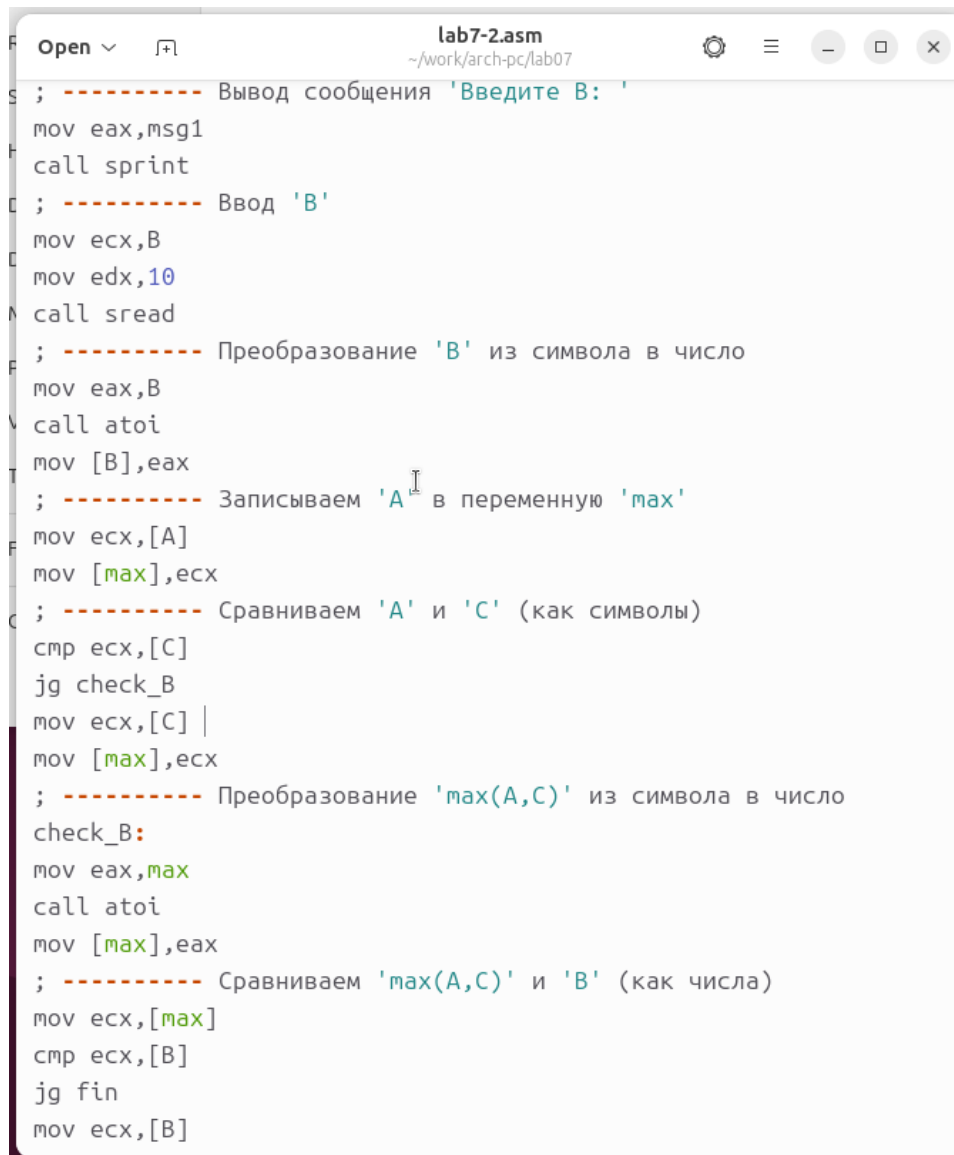


```
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1.1
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-1.1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 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.

Создаю исполняемый файл и проверяю его работу для разных значений В (рис. 2.8) (рис. 2.9).



```
Open ▾ [icon] lab7-2.asm
~/work/arch-пс/lab07

; ----- Вывод сообщения 'Введите B: '
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]
```

Рисунок 2.8: Программа lab7-2.asm

```

aachernyatin@VirtualBox:~/work/arch-pc/lab07$
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
aachernyatin@VirtualBox:~/work/arch-pc/lab07$

```

Рисунок 2.9: Запуск программы lab7-2.asm

2.2 Изучение структуры файла листинга

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

Создаю файл листинга для программы из файла `lab7-2.asm` (рис. 2.10)

```

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 E8ACFEFFFF    call sprint

```

Рисунок 2.10: Файл листинга lab7-2

Ознакомимся с его форматом и содержимым.

строка 203

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

строка 204

- 29 - номер строки
- 00000122 - адрес
- 7F0C - машинный код
- jg check_B - код программы

строка 205

- 30 - номер строки
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- mov esx,[C] - код программы

Открываю файл с программой lab7-2.asm и в инструкции с двумя операндами удаляю один операнд. Выполняю трансляцию с получением файла листинга. (рис. 2.11) (рис. 2.12)

```
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.l
st
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.l
st
lab7-2.asm:28: error: invalid combination of opcode and operands
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.11: Ошибка трансляции lab7-2

```

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 cmp ecx,
205 28 ***** error: invalid combination of opcode and operands
206 29 0000011C 7F0C jg check_B
207 30 0000011E 8B0D[39000000] mov ecx,[C]
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

```

Рисунок 2.12: Файл листинга с ошибкой lab7-2

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

2.3 Самостоятельное задание

Напишите программу нахождения наименьшей из 3 целочисленных переменных а, b и с. 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 4 - 8,88,68

```

Open v
36     mov [B],eax
37
38     mov eax,msgC
39     call sprint
40     mov ecx,C
41     mov edx,80
42     call sread
43     mov eax,C
44     call atoi
45     mov [C],eax
46
47     mov ecx,[A]
48     mov [min],ecx
49
50     cmp ecx, [B]
51     jl check_C
52     mov ecx, [B]
53     mov [min], ecx
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

```

Рисунок 2.13: Программа lab7-3.asm

```

aachernyatin@VirtualBox:~/work/arch-pc/lab07$
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Input A: 8
Input B: 88
Input C: 68
Smallest: 8
aachernyatin@VirtualBox:~/work/arch-pc/lab07$

```

Рисунок 2.14: Запуск программы lab7-3.asm

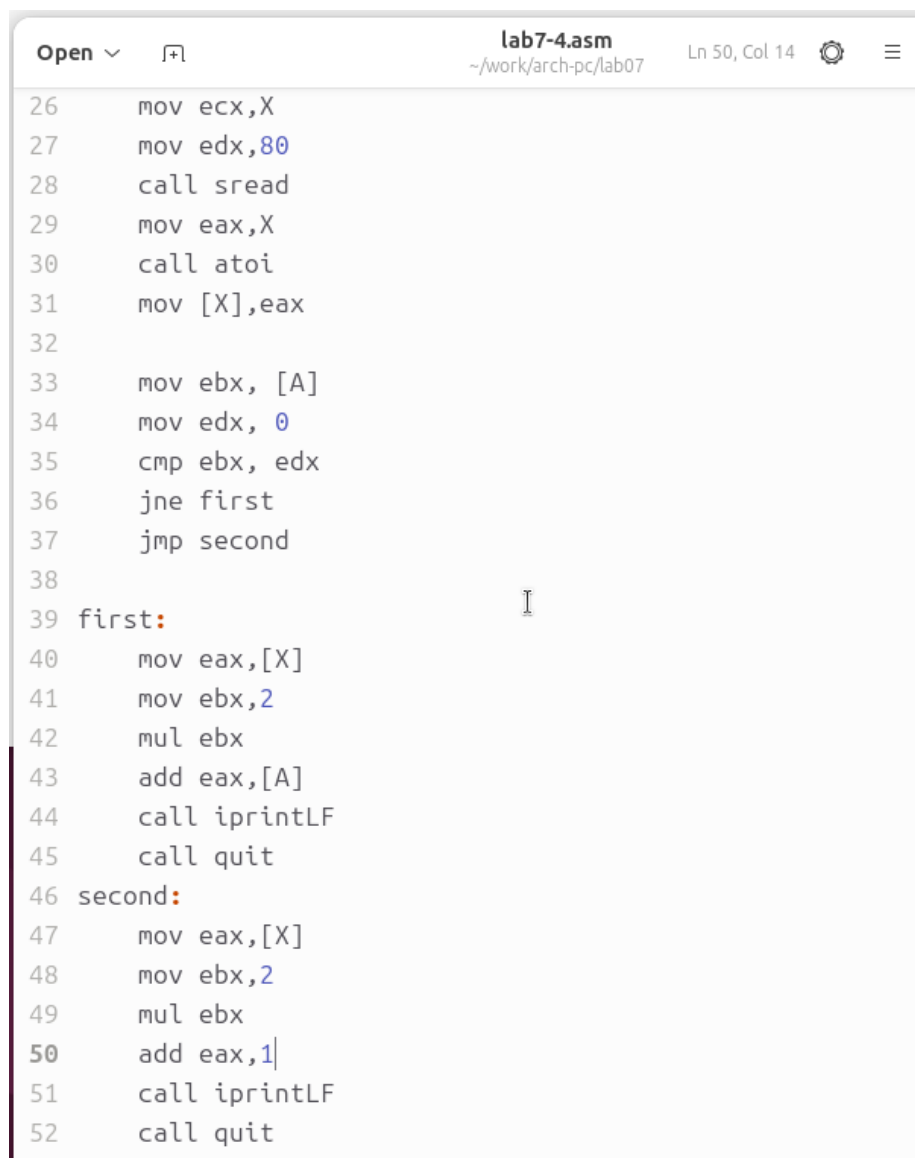
Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый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

для варианта 4

$$\begin{cases} 2x + a, a \neq 0 \\ 2x + 1, a = 0 \end{cases}$$

При $x = 3, a = 0$ получается 7.

При $x = 3, a = 2$ получается 8.



```
lab7-4.asm
~/work/arch-pc/lab07  Ln 50, Col 14

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 ebx, [A]
34     mov edx, 0
35     cmp ebx, edx
36     jne first
37     jmp second
38
39 first:
40     mov eax,[X]
41     mov ebx,2
42     mul ebx
43     add eax,[A]
44     call iprintLF
45     call quit
46 second:
47     mov eax,[X]
48     mov ebx,2
49     mul ebx
50     add eax,1
51     call iprintLF
52     call quit
```

Рисунок 2.15: Программа lab7-4.asm

```
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 0
Input X: 3
7
aachernyatin@VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Input A: 2
Input X: 3
8
aachernyatin@VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 2.16: Запуск программы lab7-4.asm

3 Выводы

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