

Отчёт по лабораторной работе 7

Архитектура компьютера

Сувд Адисурэн

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Условные переходы	11
2.3	Изучение структуры файла листинга	13
2.4	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	9
2.5	Запуск программы lab7-1.asm	9
2.6	Программа lab7-1.asm	10
2.7	Запуск программы lab7-1.asm	11
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	15
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)

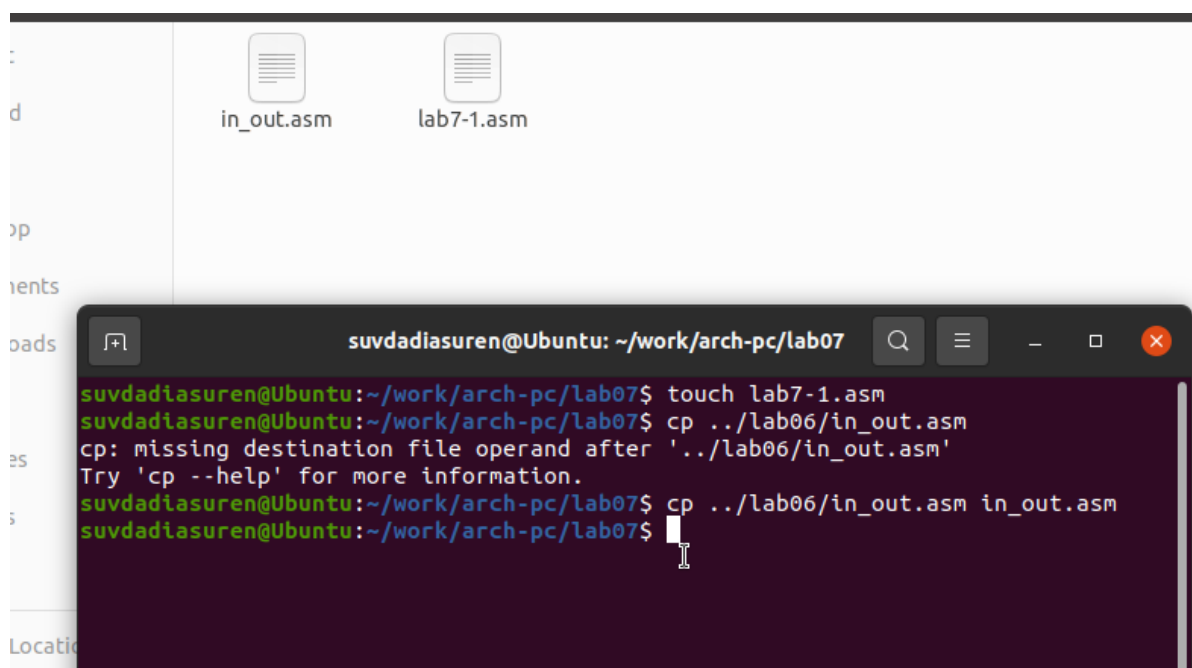


Рис. 2.1: Создан каталог

Инструкция `jmp` в NASM используется для реализации безусловных переходов. Пример программы, демонстрирующей эту инструкцию, приведен в файле lab7-1.asm. (рис. 2.2)

```

lab7-1.asm
1  %include 'in_out.asm'
2  SECTION .data
3  msg1: DB 'Сообщение № 1',0
4  msg2: DB 'Сообщение № 2',0
5  msg3: DB 'Сообщение № 3',0
6  SECTION .text
7  GLOBAL _start
8
9  _start:
10  jmp _label2
11
12  _label1:
13  mov eax, msg1
14  call sprintf
15
16  _label2:
17  mov eax, msg2
18  call sprintf
19
20  _label3:
21  mov eax, msg3
22  call sprintf
23
24  _end:
25  call quit

```

Рис. 2.2: Программа lab7-1.asm

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

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

I

Рис. 2.3: Запуск программы lab7-1.asm

Инструкция `jmp` позволяет осуществлять переходы как вперед, так и назад. Для изменения последовательности вывода программы добавляю метки `_label1` и `_end`. Таким образом, вывод программы изменится: сначала отобразится сообщение № 2, затем сообщение № 1, и программа завершит работу.

Обновляю текст программы согласно листингу 7.2. (рис. 2.4, рис. 2.5)


```

lab7-1.asm
1  %include 'in_out.asm'
2  SECTION .data
3  msg1: DB 'Сообщение № 1',0
4  msg2: DB 'Сообщение № 2',0
5  msg3: DB 'Сообщение № 3',0
6  SECTION .text
7  GLOBAL _start
8
9  _start:
10 _label2:
11
12 _label1:
13 mov eax, msg1
14 call sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25
26 _end:
27 call quit

```

Рис. 2.4: Программа lab7-1.asm

```

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

```

Рис. 2.5: Запуск программы lab7-1.asm

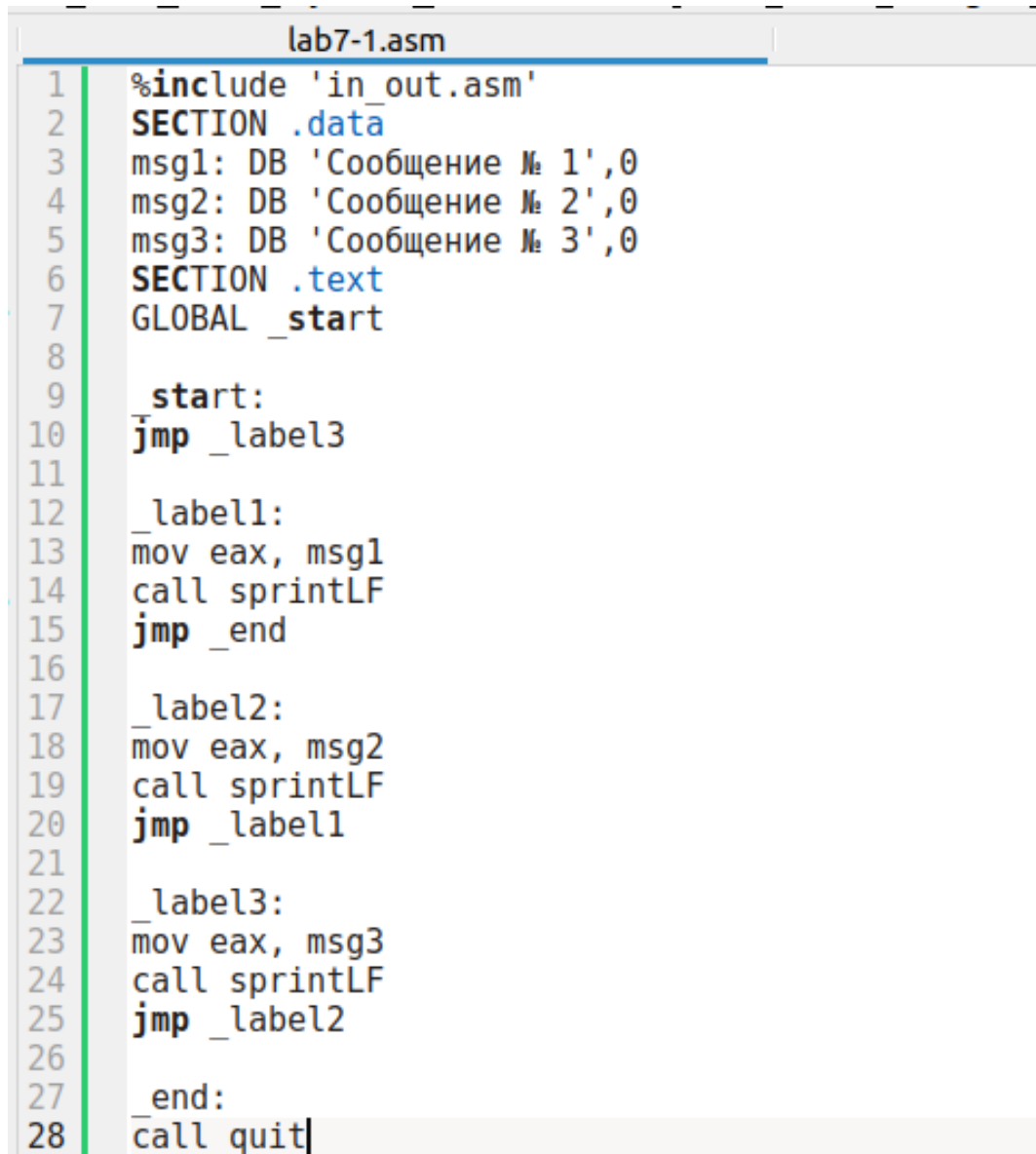
Дорабатываю текст программы для вывода следующих сообщений:

Сообщение № 3

Сообщение № 2

Сообщение № 1

Результат показан на рисунках (рис. 2.6, рис. 2.7).



```
lab7-1.asm
1  %include 'in_out.asm'
2  SECTION .data
3  msg1: DB 'Сообщение № 1',0
4  msg2: DB 'Сообщение № 2',0
5  msg3: DB 'Сообщение № 3',0
6  SECTION .text
7  GLOBAL _start
8
9  _start:
10 jmp _label3
11
12 _label1:
13 mov eax, msg1
14 call sprintf
15 jmp _end
16
17 _label2:
18 mov eax, msg2
19 call sprintf
20 jmp _label1
21
22 _label3:
23 mov eax, msg3
24 call sprintf
25 jmp _label2
26
27 _end:
28 call quit
```

Рис. 2.6: Программа lab7-1.asm

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

Рис. 2.7: Запуск программы lab7-1.asm

Использование инструкции `jmp` обеспечивает переходы независимо от условий. Однако для реализации условных переходов требуется использование дополнительных инструкций.

2.2 Условные переходы

Для демонстрации условных переходов создаю программу, определяющую максимальное значение среди трех переменных: А, В и С. Значения А и С задаются в программе, а В вводится с клавиатуры. Результаты работы программы представлены на рисунках (рис. 2.8, рис. 2.9).

```
lab7-2.asm
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A]
26 mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C]
29 jg check_B
30 mov ecx,[C]
31 mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа
   в число
33 check_B:
34 mov eax,max
35 call atoi
36 mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как
   числа)
38 mov ecx,[max]
39 cmp ecx,[B]
40 jg fin
41 mov ecx,[B]
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprintf
47 mov eax,[max]
48 call iprintLF
49 call quit
```

Рис. 2.8: Программа lab7-2.asm

```

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

```

Рис. 2.9: Запуск программы lab7-2.asm

2.3 Изучение структуры файла листинга

Для получения файла листинга указываю ключ -l при ассемблировании. Результат ассемблирования программы lab7-2.asm представлен на рисунке (рис. 2.10).

	lab7-2.lst	lab7-2.asm
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[39000000]	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

Анализируя структуру листинга, можно увидеть соответствие строк кода и их машинного представления. Например:

- **Строка 203:**

- Номер строки: 28
- Адрес: 0000011C
- Машинный код: 3B0D[39000000]
- Команда: stp esx,[C]

- **Строка 204:**

- Номер строки: 29
- Адрес: 00000122
- Машинный код: 7F0C
- Команда: jg check_B

- **Строка 205:**

- Номер строки: 30
- Адрес: 00000124
- Машинный код: 8B0D[39000000]
- Команда: mov esx,[C]

Далее изменяю инструкцию с двумя операндами, удаляя один, и повторяю трансляцию. Возникает ошибка, результат которой отображен на рисунках (рис. 2.11, рис. 2.12).

```

suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$
suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$
suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:34: error: invalid combination of opcode and operands
suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$
suvdadiasuren@Ubuntu:~/work/arch-pc/lab07$

```

Рис. 2.11: Ошибка трансляции lab7-2

lab7-2.lst			
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		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	0000014E	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 iprintLF

Рис. 2.12: Файл листинга с ошибкой lab7-2

2.4 Самостоятельное задание

1. Напишите программу, которая находит наименьшее значение из трех переменных a, b и c для следующих значений:

Вариант 4: 8,88,68.

Результат работы программы показан на рисунках (рис. 2.13, рис. 2.14).

```

lab7-3.asm
43      call atoi
44      mov [C],eax
45
46      mov ecx,[A]
47      mov [min],ecx
48
49      cmp ecx, [B]
50      jl check_C
51      mov ecx, [B]
52      mov [min], ecx
53
54  check_C:
55      cmp ecx, [C]
56      jl finish
57      mov ecx,[C]
58      mov [min],ecx
59
60  finish:
61      mov eax,answer
62      call sprint
63
64      mov eax, [min]
65      call iprintLF
66
67      call quit
68
69
70

```

Рис. 2.13: Программа lab7-3.asm

```

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

```

Рис. 2.14: Запуск программы lab7-3.asm

2. Напишите программу для вычисления функции $f(x)$ для введенных значений x и a :

Вариант 4:

$$f(x) = \begin{cases} 2x + a, & \text{если } a \neq 0 \\ 2x + 1, & \text{если } a = 0 \end{cases}$$

При $x = 3, a = 0$ результат: 7.

При $x = 3, a = 2$ результат: 8.

Результаты программы представлены на рисунках (рис. 2.15, рис. 2.16).

```

lab7-4.asm
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
53
54

```

Рис. 2.15: Программа lab7-4.asm

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