

Отчёт по лабораторной работе №7

дисциплина: Архитектура вычислительных систем

Мосолов Александр Денисович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	10
2.3	Задание для самостоятельной работы	13
3	Выводы	17

Список иллюстраций

2.1	Создаем каталог для работы	6
2.2	Текст программы в соответствии с листингом 7.1	7
2.3	Вывод программы из листинга 7.1	7
2.4	Текст программы в соответствии с листингом 7.2	8
2.5	Вывод программы из листинга 7.2	8
2.6	Текст программы lab7-1	9
2.7	Вывод программы lab7-1	9
2.8	Текст программы из листинга 7.3	10
2.9	Проверяем работу программы из листинга 7.3	10
2.10	Файл листинга lab7-2.lst	11
2.11	Текст файла листинга 7.3	11
2.12	Используем iprint	12
2.13	Листинг lab7-2.lst	12
2.14	Текст листинга после удаления из текста программы операнда В .	12
2.15	Текст программы нахождения наименьшего из трёх чисел	13
2.16	Результат работы программы поиска наименьшего из трёх чисел .	14
2.17	Результат запуска изменённого файла lab6-3	14
2.18	Текст программы для вычисления $f(x)$	15
2.19	Результат работы программы по вычислению $f(x)$	15
2.20	Проверка программы для а и х из таблицы	16

Список таблиц

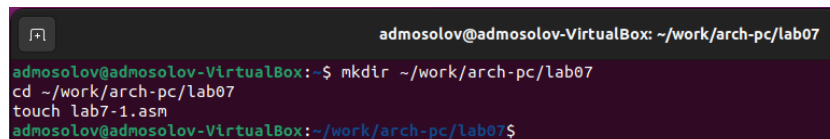
1 Цель работы

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

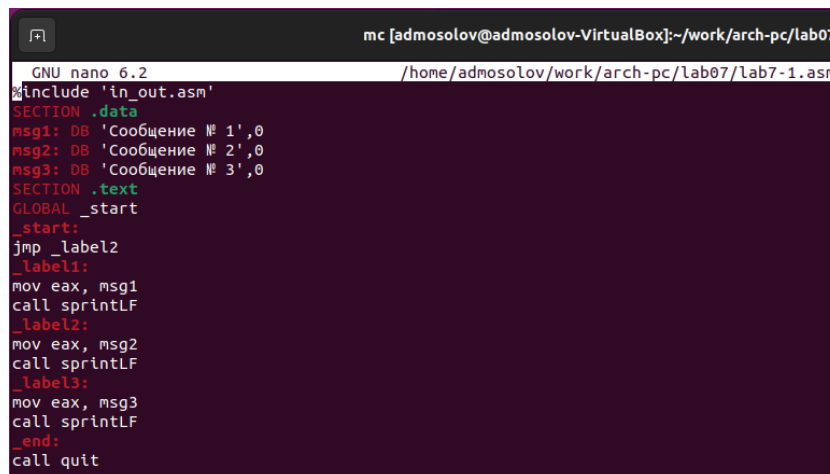
Создаём каталог для программ лабораторной работы № 7, переходим в него и создаём файл *lab7-1.asm*.



```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
admosolov@admosolov-VirtualBox:~$ mkdir ~/work/arch-pc/lab07
cd ~/work/arch-pc/lab07
touch lab7-1.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.1: Создаем каталог для работы

Инструкция *jmp* в *NASM* используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции *jmp*. Введите в файл *lab7-1.asm* текст программы из листинга 7.1.



```

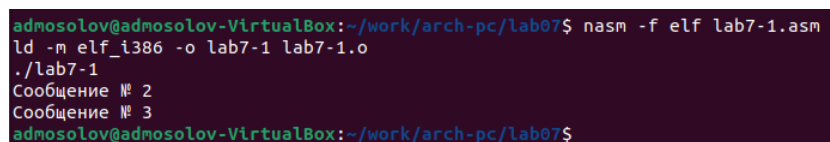
mc [admosolov@admosolov-VirtualBox]:~/work/arch-pc/lab07
GNU nano 6.2 /home/admosolov/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1
call sprintf
_label2:
mov eax, msg2
call sprintf
_label3:
mov eax, msg3
call sprintf
_end:
call quit

```

Рис. 2.2: Текст программы в соответствии с листингом 7.1

Для того, чтобы программа транслировалась без ошибок перенесем файл *in_out.asm* в *~/work/arch-pc/lab06*.

Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:



```

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

```

Рис. 2.3: Вывод программы из листинга 7.1

Таким образом, использование инструкции **jmp _label2** меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки **_label2**, пропустив вывод первого сообщения.

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*).

Изменим текст программы в соответствии с листингом 7.2

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
GNU nano 6.2 /home/admosolov/work/arch-pc/lab07/lab7-1.asm
%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. 2.4: Текст программы в соответствии с листингом 7.2

Создадим исполняемый файл и проверим его работу.

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

Рис. 2.5: Вывод программы из листинга 7.2

Изменим текст программы добавив или изменив инструкции *jmp*, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1


```

admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
GNU nano 6.2 /home/admosolov/work/arch-pc/lab07/lab7-1.asm
%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: Текст программы lab7-1

Транслируем текст, создаём объектный файл, компилируем его и запускаем программу *lab7-1*:

```
nasm -f elf lab7-1.asm
```

```
ld -m elf_i386 -o lab7-1 lab7-1.o
```

```
./lab7-1
```

```

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

```

Рис. 2.7: Вывод программы lab7-1

Создаём файл *lab7-2.asm* в каталоге *~/work/arch-pc/lab07*. Изучаем текст программы из листинга 7.3 и вводим его в *lab7-2.asm*.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
GNU nano 6.2 /home/admosolov/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax
```

Рис. 2.8: Текст программы из листинга 7.3

Создаём исполняемый файл и проверяем его работу для разных значений *B*. Проверим работу программы для чисел: 1, 50, 21, 100, 20.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 50
Наибольшее число: 50
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 21
Наибольшее число: 50
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 100
Наибольшее число: 100
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
```

Рис. 2.9: Проверяем работу программы из листинга 7.3

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. Создаём файл листинга для программы из файла `lab7-2.asm`.

Вводим команду:

```
nasm -f elf -l lab7-2.lst lab7-2.asm
```

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst  lab7-2.o
```

Рис. 2.10: Файл листинга lab7-2.lst

Откроем файл листинга *lab7-2.lst* с помощью текстового редактора *nano*.

Задание: внимательно ознакомиться с форматом листинга и содержимым.

Подробно объяснить содержимое трёх строк файла листинга по выбору.

Строка 34: 0000001D - адрес в сегменте кода, B801000000 - машинный код, mov ebx, 1 - присвоение ebx значение 1

Строка 35: 00000022 - адрес в сегменте кода, B804000000 - машинный код, mov eax, 4 - присвоение eax значение 1

Строка 34: 00000027 - адрес в сегменте кода, CD80 - машинный код, int 80h - обращение к ядру

```
19          <1>
20          <1> ;----- sprint -----
21          <1> ; Функция печати сообщения
22          <1> ; входные данные: mov eax,<message>
23          <1> sprint:
24 0000000F 52          <1>      push    edx
25 00000010 51          <1>      push    ecx
26 00000011 53          <1>      push    ebx
27 00000012 50          <1>      push    eax
28 00000013 E8E8FFFFFF    <1>      call    slen
29          <1>
30 00000018 89C2        <1>      mov     edx, eax
31 0000001A 58          <1>      pop     eax
32          <1>
33 0000001B 89C1        <1>      mov     ecx, eax
34 0000001D B801000000    <1>      mov     ebx, 1
35 00000022 B804000000    <1>      mov     eax, 4
36 00000027 CD80        <1>      int     80h
37          <1>
38 00000029 5B          <1>      pop     ebx
39 0000002A 59          <1>      pop     ecx
40 0000002B 5A          <1>      pop     edx
41 0000002C C3          <1>      ret
```

Рис. 2.11: Текст файла листинга 7.3

А теперь на вывод программы с использованием *iprint*.

```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab06$
```

Рис. 2.12: Используем iprint

Задание: откройте файл с программой *lab7-2.asm* и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга.

Проверим файл изначального листинга.

```
17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread
```

Рис. 2.13: Листинг lab7-2.lst

Уберём из него один операнд - *B* в 17 строке. Посмотрим файл листинга после повторного получения.

```
GNU nano 6.2 lab7-2.lst
5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7                          section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10                         section .text
11                         global _start
12                         _start:
13
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E810FFFFFF      call sprint
16
17                         mov ecx
17                         *****
17                         error: invalid combination of opcode and operands
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi
23 00000106 A3[0A000000]      mov [B],eax
```

Рис. 2.14: Текст листинга после удаления из текста программы операнда B

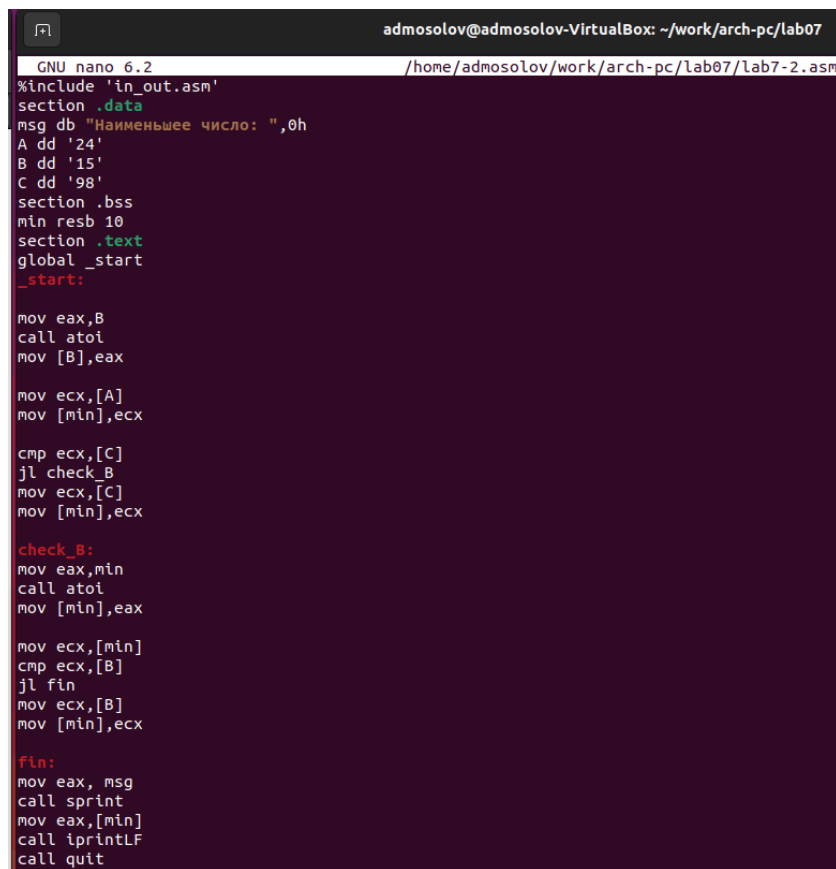
В листинге добавляется текст, сообщающий об ошибке. Выходные данные: текст об ошибке компиляции.

2.3 Задание для самостоятельной работы

Напишите программу нахождения наименьшей из 3 целочисленных переменных a , b , 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.

В моём случае вариант 9 - сравнение чисел 15, 24, 98.

Запишем текст программы в *lab7-2.asm*



```
admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
GNU nano 6.2 /home/admosolov/work/arch-pc/lab07/lab7-2.asm
#include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '24'
B dd '15'
C dd '98'
section .bss
min resb 10
section .text
global _start
_start:

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [min],ecx

cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx

check_B:
mov eax,min
call atoi
mov [min],eax

mov ecx,[min]
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx

fin:
mov eax, msg
call sprint
mov eax,[min]
call iprintLF
call quit
```

Рис. 2.15: Текст программы нахождения наименьшего из трёх чисел

Проверим программу на работоспособность. Транслируем полученный текст программы *lab7-2.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
ld -m elf_i386 -o lab7-2 lab7-2.o
./lab7-2
Наименьшее число: 15
```

Рис. 2.16: Результат работы программы поиска наименьшего из трёх чисел

Транслируем полученный текст программы *lab6-3.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
ld -m elf_i386 -o lab6-3 lab6-3.o
./lab6-3
Результат: 5
Остаток от деления: 1
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab06$
```

Рис. 2.17: Результат запуска изменённого файла lab6-3

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 № 6. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создадим файл *lab7.asm*, напомним текст программы.

```

admosolov@admosolov-VirtualBox: ~/work/arch-pc/lab07
GNU nano 6.2 lab7.asm *
#include 'in_out.asm'
section .data
msg1 DB 'Введите x: ',0h
msg2 DB "Введите a: ", 0h
otv DB 'f(x)= ', 0h
section .bss
x: RESB 80
a: RESB 80
res: RESB 80
section .text
section .text
global _start
_start:
mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov [x], eax
mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov [a], eax
mov ebx, [x]
cmp ebx, eax
jbe _less_than_x
mov [res], eax
jmp fin
_less_than_x:
add eax, ebx
mov [res], eax
jmp fin
fin:
mov eax, otv
call sprint
mov eax, [res]
call iprintLF
call quit

```

Рис. 2.18: Текст программы для вычисления $f(x)$

Транслируем текст программы *lab7.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.

```

admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7.asm
ld -m elf_i386 -o lab7 lab7.o
./lab7
Введите x: 10
Введите a: 1
f(x)= 1
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ ./lab7
Введите x: 0
Введите a: 120
f(x)= 120
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ ./lab7
Введите x: 2
Введите a: 2
f(x)= 4
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$

```

Рис. 2.19: Результат работы программы по вычислению $f(x)$

Проверим работу программы подставив $a=5, x=7$ и $a=6, x=4$.

```
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ ./lab7
Введите x: 5
Введите a: 7
f(x)= 12
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$ ./lab7
Введите x: 6
Введите a: 4
f(x)= 4
admosolov@admosolov-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.20: Проверка программы для a и x из таблицы

3 Выводы

В ходе лабораторной работы были изучены команды условного и безусловного переходов. Приобретены навыки написания программ с использованием переходов. Помимо этого, была усвоена информация, связанная с назначением и структурой файла листинга.