

Лабораторная работа №7

НПИбд-01-25 №1032252598

Иванова Ангелина Олеговна

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
2.1 Задания лабораторной работы	6
2.2 Задание для самостоятельной работы	15
3 Выводы	19

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Измененный файл lab7-1.asm	7
2.3	Создание исполняемого файла и вывод его работы	8
2.4	Измененный текст программы	9
2.5	Создание исполняемого файла и вывод его работы	9
2.6	Измененный текст программы	11
2.7	Создание исполняемого файла и вывод его работы	11
2.8	Создание файла lab7-2.asm	12
2.9	Создание исполняемого файла и вывод его работы	13
2.10	Создание листинга	13
2.11	Открытый листинг	13
2.12	Удаление в инструкции mov esx,B операнд B	14
2.13	Ошибка в терминале	14
2.14	Ошибка в листинге	15
2.15	Работа созданного нами файла	16
2.16	Работа созданного нами файла	18

Список таблиц

1 Цель работы

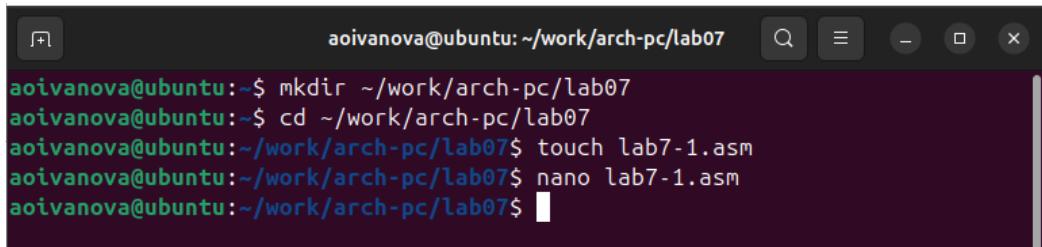
Целью данной лабораторной работы является изучение команд условного и безусловного переходов, а также приобретение навыков написания программ с использованием переходов и знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Задания лабораторной работы

2.1.1 Реализация переходов в NASM

Создали каталог для программам лабораторной работы № 7, перешли в него и создали файл lab7-1.asm



```
aoivanova@ubuntu:~/work/arch-pc/lab07$ mkdir ~/work/arch-pc/lab07
aoivanova@ubuntu:~/work/arch-pc/lab07$ cd ~/work/arch-pc/lab07
aoivanova@ubuntu:~/work/arch-pc/lab07$ touch lab7-1.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ nano lab7-1.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 2.1: Создание каталога и файла

Рассмотрели пример программы с использованием инструкции jmp. Ввели в файл lab7-1.asm текст программы из первого листинга

The screenshot shows a terminal window titled "aoivanova@ubuntu: ~/work/arch-pc/lab07" with the file "lab7-1.asm" open. The code is as follows:

```
GNU nano 7.2          lab7-1.asm *
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

At the bottom of the terminal window, there is a menu bar with the following options:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Paste
- ^J Justify

Рисунок 2.2: Измененный файл lab7-1.asm

Создали исполняемый файл и запустили его. Получили корректный результат работы данной программы

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

Рисунок 2.3: Создание исполняемого файла и вывод его работы

Далее изменили программу таким образом, чтобы она выводила сначала „Сообщение № 2“, потом „Сообщение № 1“ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили 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). Измените текст программы в соответствии со вторым листингом

The screenshot shows a terminal window titled "aoivanova@ubuntu: ~/work/arch-pc/lab07" displaying assembly code in a file named "lab7-1.asm". The code defines three messages in the .data section and prints them in the .text section using the sprintLF instruction. It includes labels for jumping between messages and exiting the program.

```
GNU nano 7.2                               lab7-1.asm
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

At the bottom of the terminal window, there is a menu bar with keyboard shortcuts:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^X Exit
- ^R Read File
- ^\\ Replace
- ^U Paste
- ^J Justify

Рисунок 2.4: Измененный текст программы

Создали исполняемый файл и проверили его работу

The terminal session shows the compilation of the assembly code into an ELF executable and its execution. The output of the program is displayed in the terminal.

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

Рисунок 2.5: Создание исполняемого файла и вывод его работы

Далее изменили текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим: Сообщение № 3 Сообщение № 2 Сообщение № 1

Листинг 1:

```
%include „in_out.asm“ ; подключение внешнего файла  
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 ; Вывод на экран строки  
call sprintLF ; „Сообщение № 1“  
  
jmp _end _label2: mov eax, msg2 ; Вывод на экран строки  
call sprintLF ; „Сообщение № 2“  
  
jmp _label1  
  
_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintLF ; „Сообщение № 3“  
  
jmp _label2  
  
_end:  
call quit ; вызов подпрограммы завершения
```

The screenshot shows a terminal window titled "aoivanova@ubuntu: ~/work/arch-pc/lab07". The file being edited is "lab7-1.asm". The code is as follows:

```
GNU nano 7.2          lab7-1.asm *
%include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

At the bottom of the terminal window, there is a menu bar with the following options:

- ^G Help
- ^O Write Out
- ^W Where Is
- ^K Cut
- ^T Execute
- ^X Exit
- ^R Read File
- ^\\ Replace
- ^U Paste
- ^J Justify

Рисунок 2.6: Измененный текст программы

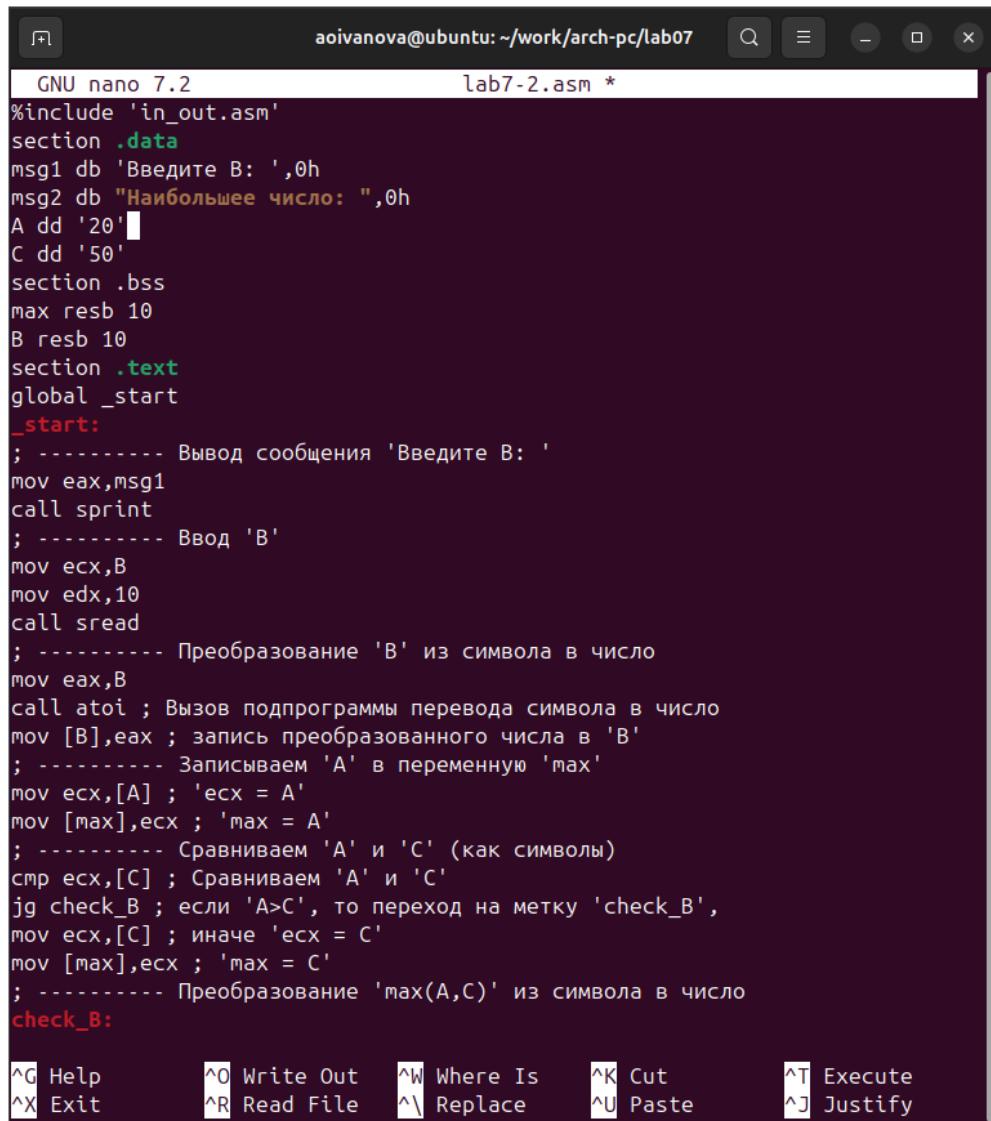
Создали исполняемый файл и проверили его работу

The screenshot shows a terminal window with the following commands and output:

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

Рисунок 2.7: Создание исполняемого файла и вывод его работы

Создали файл lab7-2.asm в каталоге ~/work/arch-pc/lab06 и ввели в него текст программы из второго листинга



The screenshot shows a terminal window titled "aoivanova@ubuntu: ~/work/arch-pc/lab07". The window contains the assembly code for "lab7-2.asm". The code includes sections for .data and .text, with various instructions like mov, cmp, and jg. It also includes comments in Russian explaining the purpose of each section. The assembly code is as follows:

```
GNU nano 7.2                               lab7-2.asm *
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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:
; ----- Вывод сообщения 'Введите В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
```

The keyboard shortcut bar at the bottom of the terminal window includes: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, and ^J Justify.

Рисунок 2.8: Создание файла lab7-2.asm

Создали исполняемый файл и проверили его работу для разных значений В.

```

aoivanova@ubuntu:~/work/arch-pc/lab07$ nano lab7-2.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 3
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 8
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 1
Наибольшее число: 50
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 128
Наибольшее число: 128

```

Рисунок 2.9: Создание исполняемого файла и вывод его работы

2.1.2 Изучение структуры файлы листинга

Создали файл листинга для программы из файла lab7-2.asm

```

aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm

```

Рисунок 2.10: Создание листинга

Открыли файл листинга lab7-2.lst с помощью текстового редактора mcedit.

```

/home/aoivanova/work/arch-pc/lab07/lab7-2.lst [---] 0 L:[177+0 177/225] *(18921/1445Bb) 0032 0x020
[*][X]
4 0000001C BE00BD18CD18BD0B5-
4 00000025 D0B520D187D0B8D0B1-
4 0000002E D0BBD0B8E3A2000 ....
5 00000035 32300000
6 00000039 35300000
7
8 00000000 <res Ah>
9 000000A <res Ah>
10
11
12
13 ; ----- Вывод сообщения 'Введите В: '
14 000000E8 B8[00000000]
15 000000ED E81DFFFFFF
16
17 000000F2 B9[0A000000]
18 000000F7 BA0A000000
19 000000FC EB42FFFFF
20
21 00000010 B8[0A000000]
22 00000016 EB91FFFFFF
23 00000010B A3[0A000000]
24
25 000000110 880D[35000000]
26 000000116 890D[00000000]
27
28 00000011C 3B0D[39000000]
29 000000122 7F0C
30 000000124 880D[39000000]
31 00000012A 890D[00000000]
32
33
34 000000130 B8[00000000]
35 000000135 EB62FFFFFF
36 00000013A A3[00000000]

; ----- Ввод 'В'
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'

```

Рисунок 2.11: Открытый листинг

Внимательно ознакомились с его форматом и содержимым. Опишем строки под номерами 14, 15 и 17.

- 1) 14 (номер строки); 000000E8 (адрес, начинается по смещению 000000E8 в сегменте кода); B8[00000000] (машинный код); mov eax,msg1 (исходный текст программы, в котором мы перемещаем адрес метки msg1 в регистр eax. msg1 - адрес строки, которую мы хотим вывести)
- 2) 15 (номер строки); 000000ED (адрес, начинается по смещению 000000ED в сегменте кода); E81DFFFF (машинный код); call sprint (исходный текст программы, в котором мы вызываем подпрограмму печати сообщения)
- 3) 17 (номер строки); 000000F2 (адрес, начинается по смещению 000000F2 в сегменте кода); B9[0A000000] (машинный код); mov ecx,B (исходный текст программы, в котором мы записываем адрес введённой переменной в ecx)

Открыли файл с программой lab7-2.asm и удалили в инструкции mov ecx,B операнд B

```
; ----- Ввод 'B'
mov ecx
mov edx,10
call sread
```

Рисунок 2.12: Удаление в инструкции mov ecx,B операнд B

Выполнили трансляцию с получением файла листинга. Создался исполнительный файл и файл листинга («lab7-2» и «lab7-2.lst»), но при создании в терминале вывело ошибку

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:17: error: invalid combination of opcode and operands
aoivanova@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 2.13: Ошибка в терминале

Также ошибка отобразилась в листинге

```
16 ; ----- Ввод 'B'  
17 mov ecx  
17 *****  
errror: invalid combination of opcode and operands
```

Рисунок 2.14: Ошибка в листинге

2.2 Задание для самостоятельной работы

1. Написали программу нахождения наименьшей из 3 целочисленных переменных а, б и с. Значения переменных выбрали из таблицы в соответствии с 19 вариантом, то есть a=46, b=32, c=74

Листинг 2

```
%include „in_out.asm“  
  
section .data  
  
a dd 46  
b dd 32  
c dd 74  
  
msg db «Наименьшее число:»,0h  
  
section .bss  
  
min resb 10  
  
section .text  
  
global _start  
  
_start:  
  
mov eax, [a]  
mov ebx, [b]  
mov ecx, [c]  
cmp eax, ebx  
jl check_c  
mov eax, ebx  
check_c:
```

```

cmp eax, ecx
jl save_result
mov eax, ecx
save_result:
mov [min], eax
mov eax, msg
call sprint
mov eax, [min]
call iprintLF
call quit

```

```

aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 32

```

Рисунок 2.15: Работа созданного нами файла

Как можно заметить, программа корректно считает наименьшее число

2. Написали программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.
- 19 вариант : если $x > a$ $f(x) = a + x$, если $x \leq a$ $f(x) = x$

Листинг 3

```

%include „in_out.asm“
section .data
msgx db „Ведите x =“,0h
msga db „Ведите a =“,0h
msgresult db „Результат f(x) =“,0h
section .bss
x resb 10

```

```
a resb 10
result resb 10
section .text
global _start
_start:
    mov eax, msgx
    call sprint
    mov ecx, x
    mov edx, 10
    call sread mov eax, x
    call atoi
    mov [x], eax
    mov eax, msga
    call sprint
    mov ecx, a
    mov edx, 10
    call sread mov eax, a
    call atoi
    mov [a], eax
    mov ebx, [x]
    mov ecx, [a]
    cmp ebx, ecx
    jg greater_case
    mov [result], ebx
    jmp output_result

greater_case:
    add ecx, ebx
    mov [result], ecx
output_result:
```

```
mov eax, msgresult
```

```
call sprint
```

```
mov eax, [result]
```

```
call iprintLF
```

```
call quit
```

Создали исполняемый файл и проверили его работу для значений x и a



The screenshot shows a terminal window titled 'aoivanova@ubuntu: ~/work/arch-pc/lab07'. The terminal displays the following command-line session:

```
aoivanova@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aoivanova@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Введите x = 4
Введите a = 5
Результат f(x) = 4
aoivanova@ubuntu:~/work/arch-pc/lab07$ ./lab7-4
Введите x = 3
Введите a = 2
Результат f(x) = 5
aoivanova@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 2.16: Работа созданного нами файла

3 Выводы

Изучили команды условного и безусловного переходов, а также приобрели навыки написания программ с использованием переходов и познакомились с назначением и структурой файла листинга.