

Отчёта по лабораторной работе №7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Кичигина Полина Евгеньевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Задания для самостоятельной работы	13
5	Выводы	18

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	7
3.2	Заполняем файл	8
3.3	Запускаем файл и смотрим на его работу	8
3.4	Изменяем файл	9
3.5	Запускаем файл и смотрим на его работу	9
3.6	Редактируем файл	10
3.7	Проверяем работу файла	10
3.8	Создаем и заполняем файл	11
3.9	Смотрим на работу программ	11
3.10	создаем файл	11
3.11	Транслируем файл и наблюдаем его работу	12
4.1	Создаем и формируем программу	14
4.2	Смотрим на работу программы(всё верно)	14
4.3	Создаем файл и пишем программу	16
4.4	Проверяем работу программы	17

Список таблиц

1 Цель работы

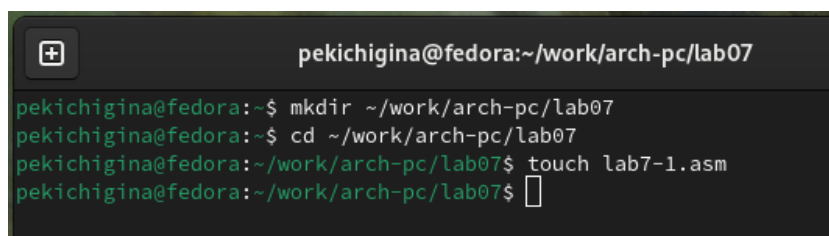
Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

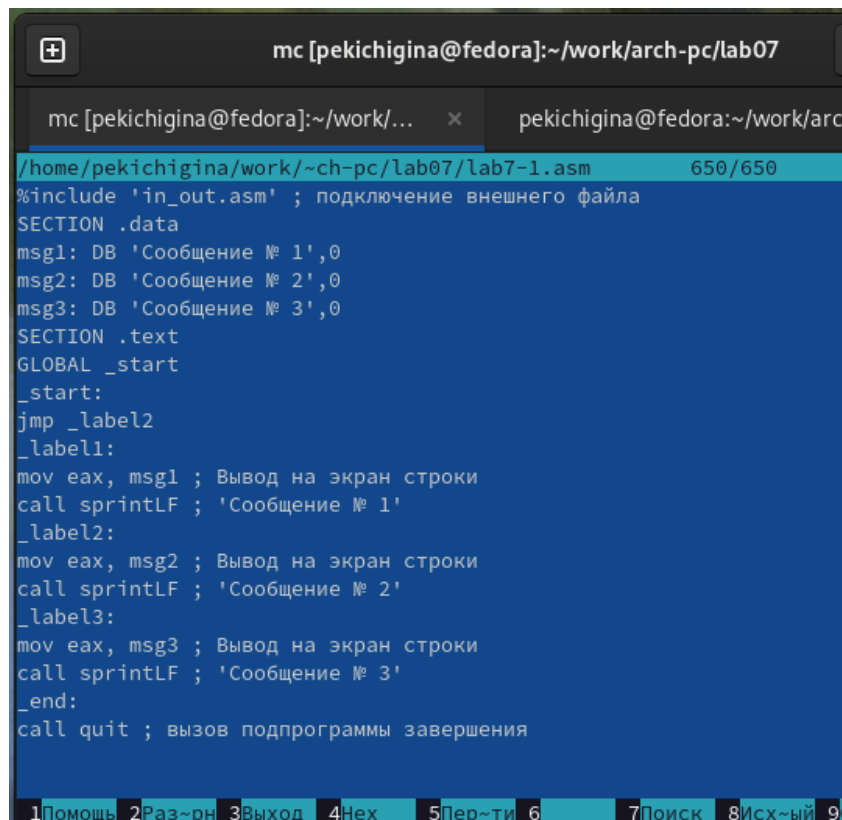
1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm(рис. 3.1)



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

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

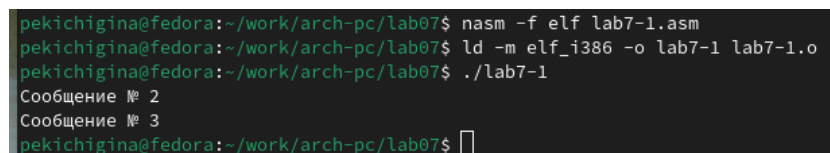
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. Введите в файл `lab7-1.asm` текст программы из листинга 7.1(рис. 3.2)

A screenshot of a terminal window with a dark background. The title bar shows 'mc [pekichigina@fedora]:~/work/arch-pc/lab07'. The terminal content shows the assembly of a file named 'lab7-1.asm'. The assembly code includes a data section with three messages and a text section with a loop that prints each message. The status bar at the bottom shows various menu options in Russian.

```
mc [pekichigina@fedora]:~/work/arch-pc/lab07
/home/pekichigina/work/~ch-pc/lab07/lab7-1.asm 650/650
%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 sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Заполняем файл

Создайте исполняемый файл и запустите его(рис. 3.3)

A screenshot of a terminal window showing the execution of the assembly program. The user runs 'nasm' to assemble the file and 'ld' to link it. The output shows the messages 'Сообщение № 2' and 'Сообщение № 3' being printed.

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

Рис. 3.3: Запускаем файл и смотрим на его работу

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`). Измените текст программы в соответствии с листингом 7.2(рис. 3.4)


```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lab07/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 sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки

```

[Прочитано 23 строки]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Пози
^X Выход	^R ЧитФайл	^_ Замена	^U Вставить	^J Вывод	^/_ К ст

Рис. 3.4: Изменяем файл

Создайте исполняемый файл и проверьте его работу(рис. 3.5)

```

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

```

Рис. 3.5: Запускаем файл и смотрим на его работу

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

Сообщение № 3 Сообщение № 2 Сообщение № 1(рис. 3.6)

```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lab07/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 sprintf ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения

```

Рис. 3.6: Редактируем файл

Теперь проверяем работу программы(рис. 3.7)

```

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

```

Рис. 3.7: Проверяем работу файла

3. Создайте файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.3 и введите в lab7-2.asm(рис. 3.8)

```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lab07/lab7-2.asm Изменён
%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:
; ----- 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 ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке

```

Рис. 3.8: Создаем и заполняем файл

Создайте исполняемый файл и проверьте его работу для разных значений B(рис. 3.9)

```

pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
pekichigina@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
pekichigina@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 4
Наибольшее число: 50
pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
pekichigina@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
pekichigina@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60
pekichigina@fedora:~/work/arch-pc/lab07$ 

```

Рис. 3.9: Смотрим на работу программ

4. Создайте файл листинга для программы из файла lab7-2.asm(рис. 3.10)

```

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

```

Рис. 3.10: создаем файл

Откройте файл листинга lab7-2.lst с помощью любого текстового редактора,

например mcsedit. Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержание трёх строк файла листинга по выбору.

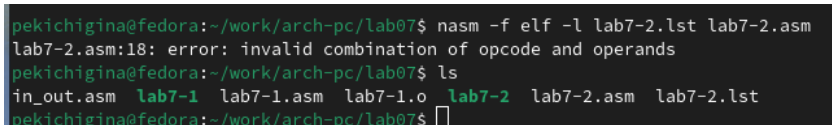
Строка 33: 0000001D-адрес в сегменте кода, B01000000-машинный код, mov ebx,1-присвоение переменной ebx значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4.

Строка 35: 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга.

Какие выходные файлы создаются в этом случае? Что добавляется в листинге? При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst(рис. 3.11)



```
pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
pekichigina@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1  lab7-1.asm  lab7-1.o  lab7-2  lab7-2.asm  lab7-2.lst
pekichigina@fedora:~/work/arch-pc/lab07$
```

Рис. 3.11: Транслируем файл и наблюдаем его работу

4 Задания для самостоятельной работы

1. 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 (рис. 4.1)

```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
    msg1 db 'Введите B: ',0h
    msg2 db "Наименьшее число: ",0h
    A dd '79'
    C dd '41'
section .bss
    min 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
    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, msg2
    call sprint
    mov eax,[min]
    call iprintLF
    call quit

```

Рис. 4.1: Создаем и формируем программу

Создайте исполняемый файл и проверьте его работу(рис. 4.2)

```

pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
pekichigina@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
pekichigina@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 83
Наименьшее число: 41
pekichigina@fedora:~/work/arch-pc/lab07$ 

```

Рис. 4.2: Смотрим на работу программы(всё верно)

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вы-

числений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 6 (рис. 4.3)

```

GNU nano 7.2 /home/pekichigina/work/arch-pc/lat
%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
    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
    cmp eax,[x]
    jg check_A
    mov ecx,[x]
    sub ecx,[a]
    mov [res],ecx
    jmp fin
check_A:
    mov ecx,5
    mov [res],ecx
    jmp fin
fin:
    mov eax, otv
    call sprint
    mov eax,[res]
    call iprintLF
    call quit

```

Рис. 4.3: Создаем файл и пишем программу

Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6(рис. 4.4)

```
pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
pekichigina@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
pekichigina@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 2
F(x) = 5
pekichigina@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
pekichigina@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
pekichigina@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
F(x) = 1
pekichigina@fedora:~/work/arch-pc/lab07$
```

Рис. 4.4: Проверяем работу программы

5 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.