

# **Отчёт по лабораторной работе №7**

**НКАбд-03-25, 1032253497**

Кулаженкова Яна Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файла листинга . . . . .	10
<b>5</b>	<b>Задания для самостоятельной работы</b>	<b>12</b>
5.1	1 . . . . .	12
5.2	2 . . . . .	12
<b>6</b>	<b>Выводы</b>	<b>14</b>

# Список иллюстраций

4.1	Подготовка файла . . . . .	8
4.2	Исполняемый файл . . . . .	9
4.3	Изменим код . . . . .	9
4.4	Запуск файла . . . . .	9
4.5	Второй файл . . . . .	10
4.6	Запуск второго файла . . . . .	10
4.7	Новый код программы . . . . .	10
5.1	Запуск файла . . . . .	12
5.2	Запуск файла . . . . .	13

## **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 Задание

Освоить методы реализации ветвлений в ассемблере NASM путем изучения команд безусловного и условного переходов.

### 3 Теоретическое введение

Данная лабораторная работа посвящена изучению механизмов реализации ветвлений в среде ассемблера NASM. Основным инструментом организации ветвлений являются команды передачи управления (команды переходов), позволяющие изменять естественный поток выполнения программы в зависимости от определенных условий.

Переходы классифицируются на два основных типа:

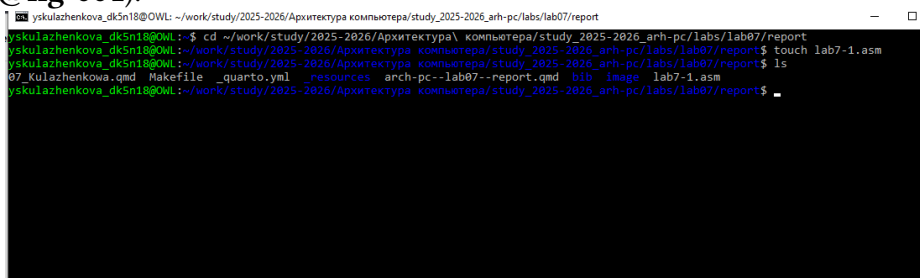
1. Безусловные переходы - передача управления производится без какой-либо предварительной проверки условий. Этот механизм реализуется с помощью инструкции `jmp`, которая принимает в качестве аргумента целевой адрес (метку, абсолютный адрес или содержимое регистра).
2. Условные переходы выполняются только при соблюдении определенного условия, которое определяется состоянием специальных флагов регистра флагов. Эти флаги устанавливаются различными инструкциями, такими как `cmp` (сравнение), в зависимости от результата операций.

Раздел посвящен детальному рассмотрению обоих типов переходов, особенностям реализации ветвлений и формированию соответствующих конструкций в ассемблерных программах. Особое внимание уделяется структуре файла листинга, который создается компилятором NASM и играет важную роль в процессе отладки программ.

## 4 Выполнение лабораторной работы

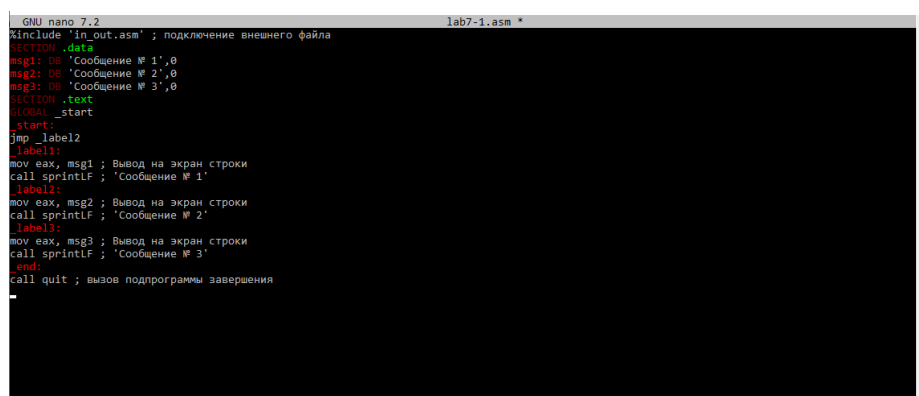
### 4.1 Реализация переходов в NASM

Начнём выполнение лабораторной работы. Создадим каталог для программ (?@fig-001).



```
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arch-pc/labs/lab07/report
yskulazhenkova_dk5n18@OWL: ~$ cd ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arch-pc/labs/lab07/report
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arch-pc/labs/lab07/report$ touch lab7-1.asm
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arch-pc/labs/lab07/report$ ls
07_Kulazhenkova.qmd  Makefile  _quarto.yml  _resources  arch-pc--lab07--report.qmd  bib  image  lab7-1.asm
yskulazhenkova_dk5n18@OWL: ~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arch-pc/labs/lab07/report$
```

Рассмотрим пример программы с использованием инструкции `jmp`. Введем в файл `lab6-1.asm` данный текст программы (рис. 4.1).



```
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 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
```

Рисунок 4.1: Подготовка файла



Создадим исполняемый файл и запустим его (рис. 4.2).

```
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nasm -f elf lab7-1.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ld -m elf_i386 -o lab7-1 lab7-1.o
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-1
Сообщение № 2
Сообщение № 3
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$
```

Рисунок 4.2: Исполняемый файл

Изменим программу таким образом, чтобы она выводила сначала „Сообщение №2“, потом „Сообщение №1“ и завершала работу (рис. 4.3).

```
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 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"
_end:
call quit ; вызов подпрограммы завершения_
```

Рисунок 4.3: Изменим код

Продемонстрируем работу файла (рис. 4.4).

```
Сообщение № 3
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nano lab7-1.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nasm -f elf lab7-1.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ld -m elf_i386 -o lab7-1 lab7-1.o
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-1
Сообщение № 2
Сообщение № 1
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-1
Сообщение № 2
Сообщение № 1
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nano lab7-1.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$
```

Рисунок 4.4: Запуск файла

Теперь рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. Для этого создадим файл lab7-2.asm (рис. 4.5).

```

GNU nano 7.2 lab7-2.asm
mov ecx,8
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'
; ----- 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'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C) > B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call sprintf ; Вывод 'max(A,B,C)'
call printf ; Вывод
call exit ; Выход
File Name to Write: lab7-2.asm
^G Help ^H DOS Format ^M-A Append ^M-B Backup File
^O Mac Format ^R Mac Roman ^T Source

```

Рисунок 4.5: Второй файл

Создадим исполняемый файл и проверим его работу (рис. 4.6).

```

yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ touch lab7-2.asm
yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nano lab7-2.asm
yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nasm -f elf lab7-2.asm
yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ld -m elf_i386 -o lab7-2 lab7-2.o
yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-2
Введите B: 3
Наибольшее число: 50
Введите B: 50
Наибольшее число: 50
Введите B: -9
Наибольшее число: 50
yskulazhenkova_dk5n1@QML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$

```

Рисунок 4.6: Запуск второго файла

## 4.2 Изучение структуры файла листинга

Создадим файл листинга для программы из файла lab7-2.asm (рис. 4.7).

```

GNU nano 7.2 lab7-2.lst
1      1      <I> %include 'in_out.asm'
2      2      <I> ; ----- slen -----
3      3      <I> slen:
4      4 00000000 53      <I> push ebx
5      5 00000001 89C3    <I> mov ebx, eax
6      6      <I>
7      7      <I> nextchar:
8      8 00000003 082800   <I> cmp byte [eax], 0
9      9 00000006 7403    <I> je Finished
10     10 00000008 40      <I> inc eax
11     11 00000009 EBF8    <I> jmp nextchar
12     12      <I>
13     13      <I> Finished:
14     14 0000000B 29D8    <I> sub eax, ebx
15     15 0000000D 5B      <I> pop ebx
16     16 0000000E C3      <I> ret
17     17      <I>
18     18      <I>
19     19      <I> ; ----- sprintf -----
20     20      <I> ; функция печати сообщения
21     21      <I> ; входные данные: mov eax,message
22     22      <I> sprintf:
23     23 0000000F 52      <I> push edx
24     24 00000010 51      <I> push ecx
25     25 00000011 53      <I> push ebx
26     26 00000012 50      <I> push eax
27     27 00000013 E8B8FFFF   <I> call slen
28     28      <I>
29     29 00000018 89C2    <I> mov edx, eax
30     30 0000001A 58      <I> pop eax
31     31      <I>
32     32 0000001B 89C1    <I> mov ecx, eax

```

Рисунок 4.7: Новый код программы

Изучим содержание файла листинга lab7-2.lst с помощью любого текстового редактора.

Анализ выбранного фрагмента листинга

```
10 | 00000000 | B804000000 | mov eax,4
11 | 00000005 | BB01000000 | mov ebx,1
12 | 0000000A | B900000000 | mov ecx,hello
```

Каждая строка листинга состоит из четырех полей:

1. Номер строки.
2. Смещение.
3. Машинный код.
4. Исходный текст программы.

Первая строка листинга: Номер строки: 10 — десятая строка листинга. Смещение: 00000000 — начало сегмента. Машинный код: B804000000 — команда MOV с непосредственной загрузкой значения 4 в регистр EAX. Исходный текст: инструкция `mov eax,4`.

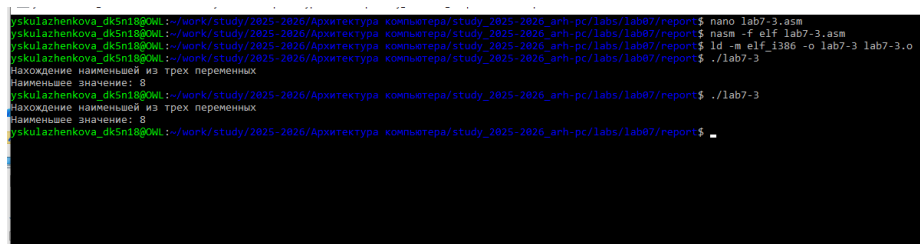
Вторая строка листинга: Номер строки: 11 — следующая строка листинга. Смещение: 00000005 — инструкция располагается по смещению 5. Машинный код: BB01000000 — команда MOV с непосредственной загрузкой значения 1 в регистр EBX. Исходный текст: инструкция `mov ebx,1`.

Третья строка листинга: Номер строки: 12 — очередная строка листинга. Смещение: 0000000A — инструкция размещена по смещению A (десятичное 10). Машинный код: B900000000 — команда MOV с загрузкой адреса метки `hello` в регистр ECX. Исходный текст: инструкция `mov ecx,hello`.

## 5 Задания для самостоятельной работы

### 5.1 1

Напишем программу, которая находит наименьшую величину из трех целочисленных переменных (рис. 5.1).



```
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nano lab7-3.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nasm -f elf lab7-3.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ld -m elf_i386 -o lab7-3 lab7-3.o
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-3
Нахождение наименьшей из трех переменных
Наименьшее значение: 8
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-3
Нахождение наименьшей из трех переменных
Наименьшее значение: 8
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$
```

Рисунок 5.1: Запуск файла

### 5.2 2

Создадим программу, которая вычисляет значение функции для введенных с клавиатуры значений (рис. 5.2).

```

yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ touch lab7-4.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nano lab7-4.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ nasm -f elf lab7-4.asm
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ld -m elf_i386 -o lab7-4 lab7-4.o
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-4
Вычисление функции f(x, a)
f(x,a) = { a/2, если a != 1; 10*x, если a = 1 }
Введите значение x: 1
Введите значение a: 2
Результат f(x, a): 1
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$ ./lab7-4
Вычисление функции F(x, a)
F(x,a) = { a/2, если a != 1; 10*x, если a = 1 }
Введите значение x: 2
Введите значение a: 1
Результат f(x, a): 12
yskulazhenkova_dk5n18@OML:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab07/report$

```

Рисунок 5.2: Запуск файла

## 6 Выводы

В ходе выполнения лабораторной работы были освоены механизмы реализации ветвлений в языке ассемблера NASM посредством команд безусловного и условного переходов. Были изучены особенности работы с этими командами, рассмотрены способы изменения естественного порядка выполнения инструкций в программе.

Все выполненные задания подтвердили успешное усвоение материала, касающегося принципов работы команд переходов и особенностей формирования программы на ассемблере.