

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

**Назармамадов Умед Джамshedович**

# 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>
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	Создание файла . . . . .	8
4.2	Редактирование файла . . . . .	8
4.3	Запуск файла . . . . .	8
4.4	Редактирование файла . . . . .	9
4.5	Запуск файла . . . . .	9
4.6	Редактирование файла . . . . .	10
4.7	Редактирование файла . . . . .	10
4.8	Создание файла . . . . .	10
4.9	Изменение файла . . . . .	11
4.10	Запуск файла . . . . .	11
4.11	Создание файла . . . . .	11
4.12	Изучение файла . . . . .	12
4.13	Выбранные строки . . . . .	12
4.14	Удаление выделенного операнда . . . . .	12
4.15	Запуск файла . . . . .	13
4.16	Решение задачи . . . . .	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

1. Реализация переходов в NASM.
2. Изучение структуры файлы листинга.
3. Задание для самостоятельной работы.

### 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

1. Условный переход - выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
2. Безусловный переход - выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией `jmp`. Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания. Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

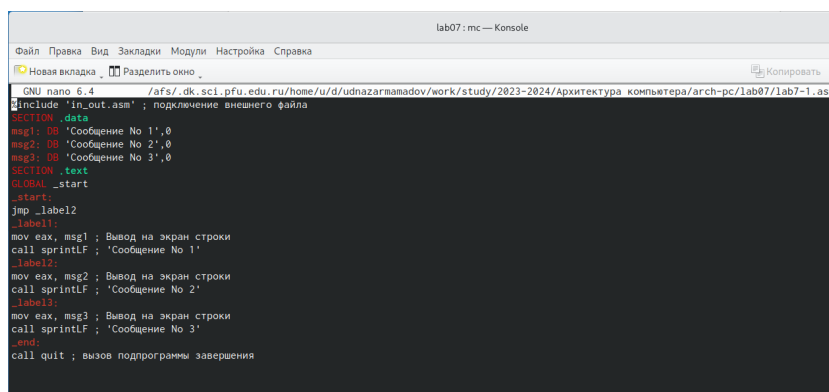
## 4 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. -4.1).

```
udnazarmamadov@dk5n56 ~ $ mkdir ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab07
udnazarmamadov@dk5n56 ~ $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/lab07
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 4.1: Создание файла

Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. -4.1).



```
lab07:mc - Konsole
Файл Правка Вид Закладки Модули Настройка Справка
[+] Новая вкладка [ ] Разделить окно [ ] Копировать
GNU nano 6.4 /afs/.dk.scl.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB "Сообщение No 1",0
msg2: DB "Сообщение No 2",0
msg3: DB "Сообщение No 3",0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; "Сообщение No 1"
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; "Сообщение No 2"
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; "Сообщение No 3"
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. -4.1).

```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-1.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.3: Запуск файла



Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2 (рис. -4.1).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение No 1',0
msg2: DB 'Сообщение No 2',0
msg3: DB 'Сообщение No 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение No 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение No 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение No 3'
_end:
call quit ; вызов подпрограммы завершения
  
```

Рис. 4.4: Редактирование файла

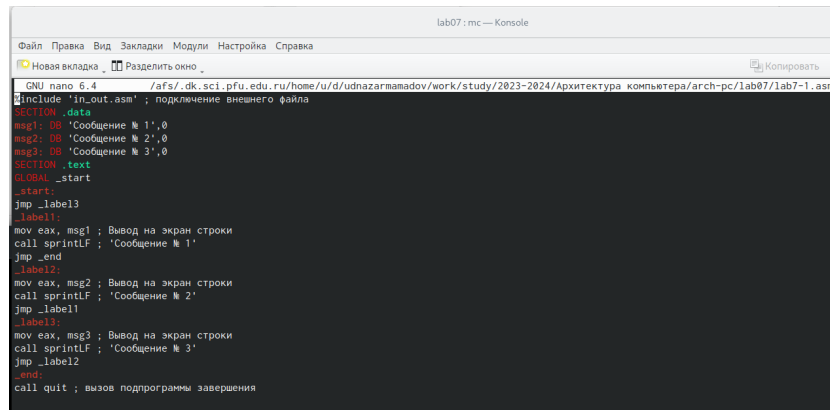
Создаю исполняемый файл и проверяю его работу (рис. -4.1).

```

udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-1.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1
Сообщение No 2
Сообщение No 1
  
```

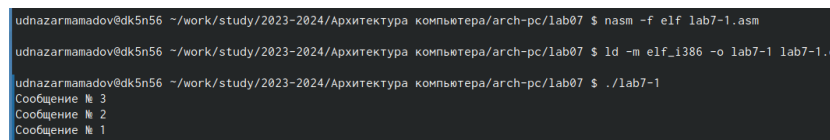
Рис. 4.5: Запуск файла

Изменяю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1` (рис. -4.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/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
```

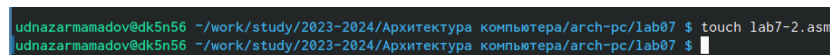
Рис. 4.6: Редактирование файла



```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf lab7-1.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Редактирование файла

Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. -4.1).



```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ touch lab7-2.asm
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $
```

Рис. 4.8: Создание файла

Вставляю текст программы из листинга 7.3 ввожу в lab7-2.asm (рис. -4.1).

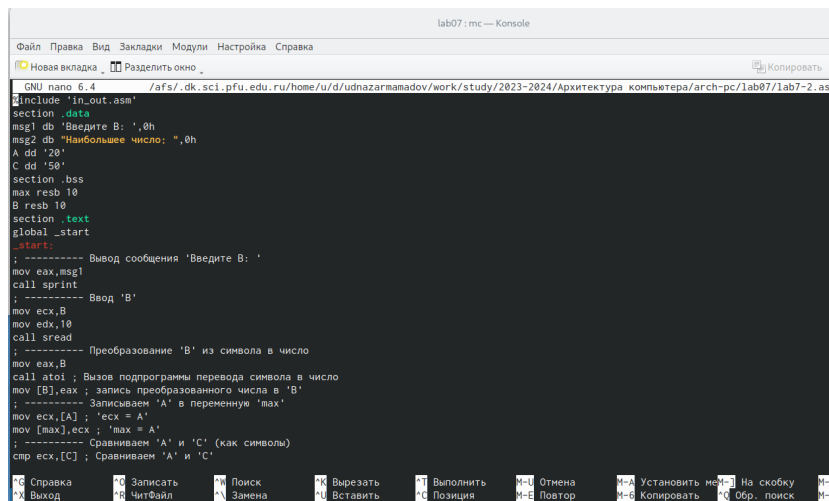


Рис. 4.9: Изменение файла

Создаю исполняемый файл и запускаю его (рис. -4.1).

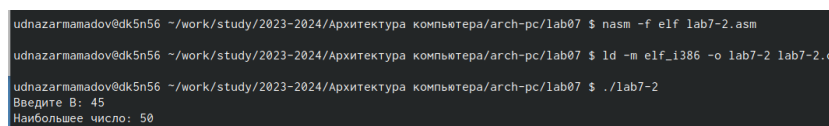


Рис. 4.10: Запуск файла

Создаю файл листинга для программы из файла lab7-2.asm (рис. -4.1).

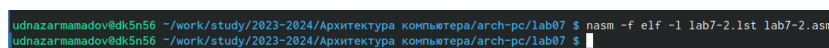


Рис. 4.11: Создание файла

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое (рис. -4.1).

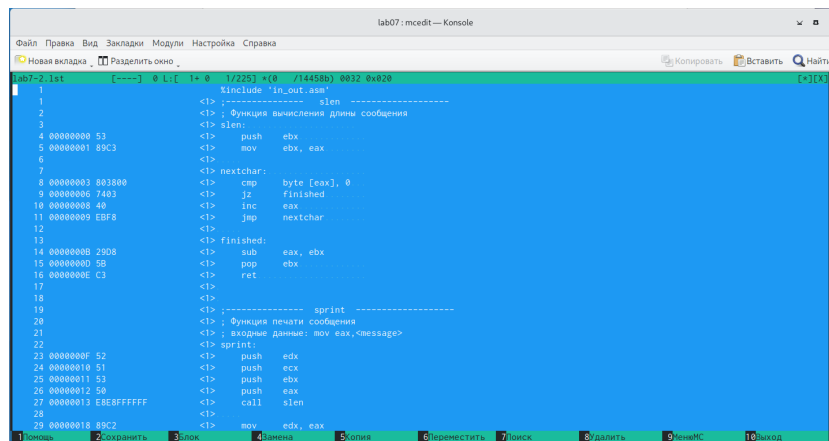


Рис. 4.12: Изучение файла

В представленных трех строчках содержатся следующие данные (рис. -4.1).

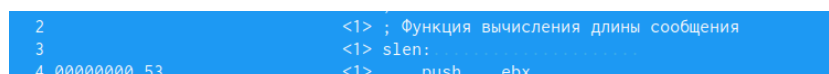


Рис. 4.13: Выбранные строки

2” - номер строки кода, “; Функция вычисления длинны сообщения” “3” - номер строки кода, “slen” - название функции, не имеет адреса и машинного кода. “4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек. Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд (рис. -4.1).

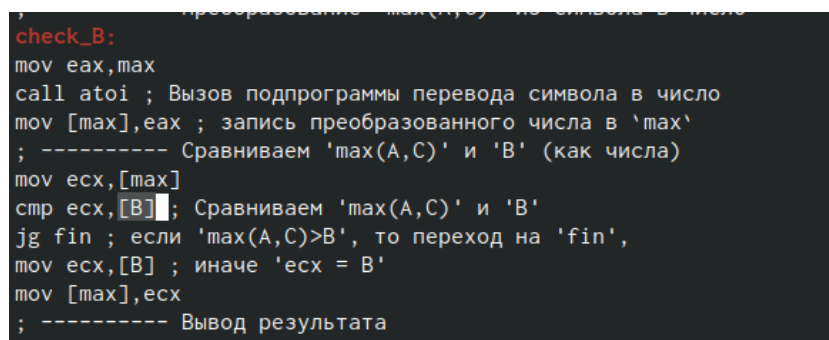


Рис. 4.14: Удаление выделенного операнда

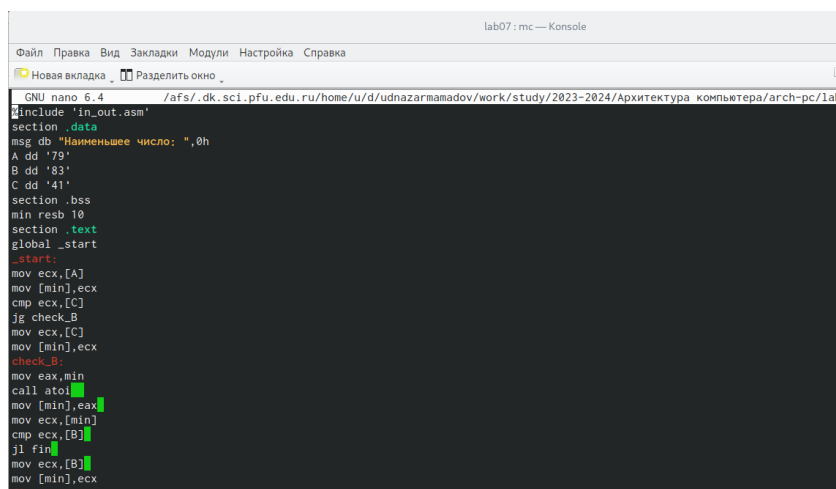
Выполняю трансляцию с получением файла листинга. (рис. -4.1). На выходе я получаю ошибку:инструкция mov (единственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

```
udnazarmamadov@dk5n56 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07 $ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:39: error: invalid combination of opcode and operands
```

Рис. 4.15: Запуск файла

### #Задания для самостоятельной работы

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных а, в и с. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант под номером 6, поэтому мои значения - 79, 83 и 41. (рис. -4.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/u/d/udnazarmamadov/work/study/2023-2024/Архитектура компьютера/arch-pc/lab
#include 'in_out.asm'
section .data
msg db "Наименьшее число: ",0h
A dd '79'
B dd '83'
C dd '41'
section .bss
min resb 10
section .text
global _start
_start:
mov ecx,[A]
mov [min],ecx
cmp ecx,[C]
jg 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:
```

Рис. 4.16: Решение задачи

Код пограммы:

```
%include 'in_out.asm' section .data msg db "Наименьшее число:",0h A dd '79' B dd '83' C dd '41' section .bss min resb 10 section .text global _start _start: mov ecx,[A] mov [min],ecx cmp ecx,[C] jg 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
```

## 5 Выводы

При выполнении данной лабораторной работы я изучил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и ознакомился с назначением и структурой файла листинга.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-



- е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
18. — 1120 с. — (Классика Computer Science).