РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

дисциплина: Архитектура компьютера

Студент: Волчкова Елизавета Дмитриевна

Группа: НКАбд-01-24

МОСКВА

2024 г.

.

Цель работы.

Изучить команды условного и безусловного переходов и приобрести навыки написания программ с использованием переходов, а также знакомство с назначением и структурой файла листинга.

Теоретическое введение.

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода.

Можно выделить 2 типа переходов:

• условный переход – выполнение или не выполнение перехода в опре деленную точку программы в зависимости от проверки условия.

• безусловный переход – выполнение передачи управления в определен ную точку программы без каких-либо условий.

Задание для самостоятельной работы.

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

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

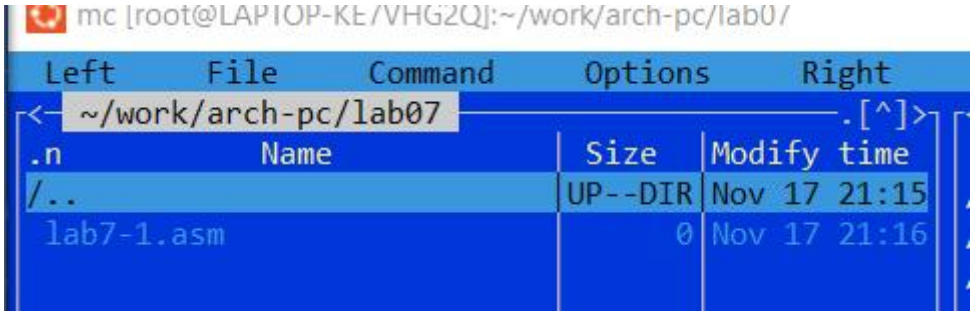
Порядок выполнения лабораторной работы.

1. Сначала создала каталог для программам лабораторной работы № 7, перешла в него и создайте файл lab7-1.asm:

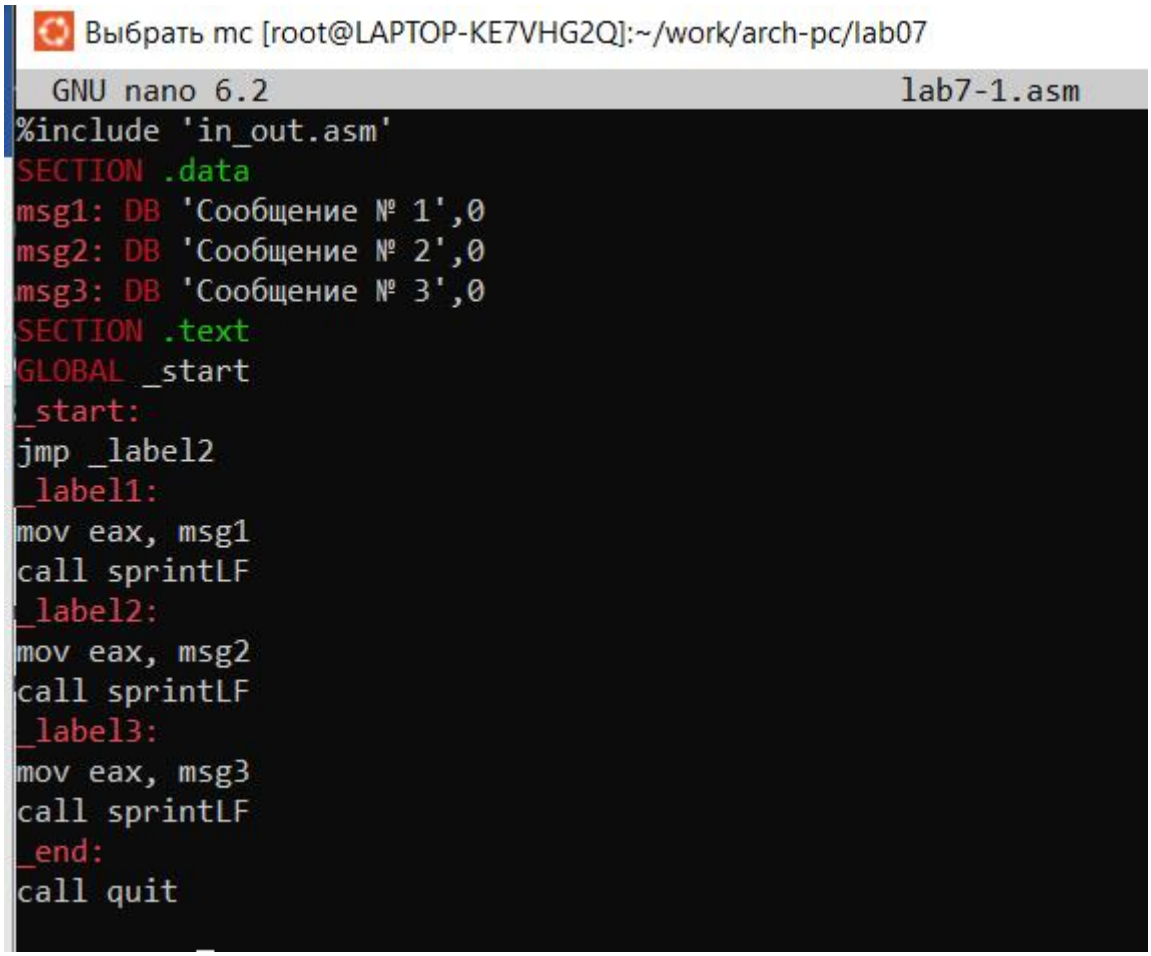
mkdir ~/work/arch-pc/lab07

cd ~/work/arch-pc/lab07

touch lab7-1.asm



Ввела в файл lab7-1.asm текст программы из листинга 7.1.



Создала исполняемый файл и запустила его.

Результат работы данной программы будет следующим:

root@LAPTOP-KE7VHG2Q:~# ./lab7

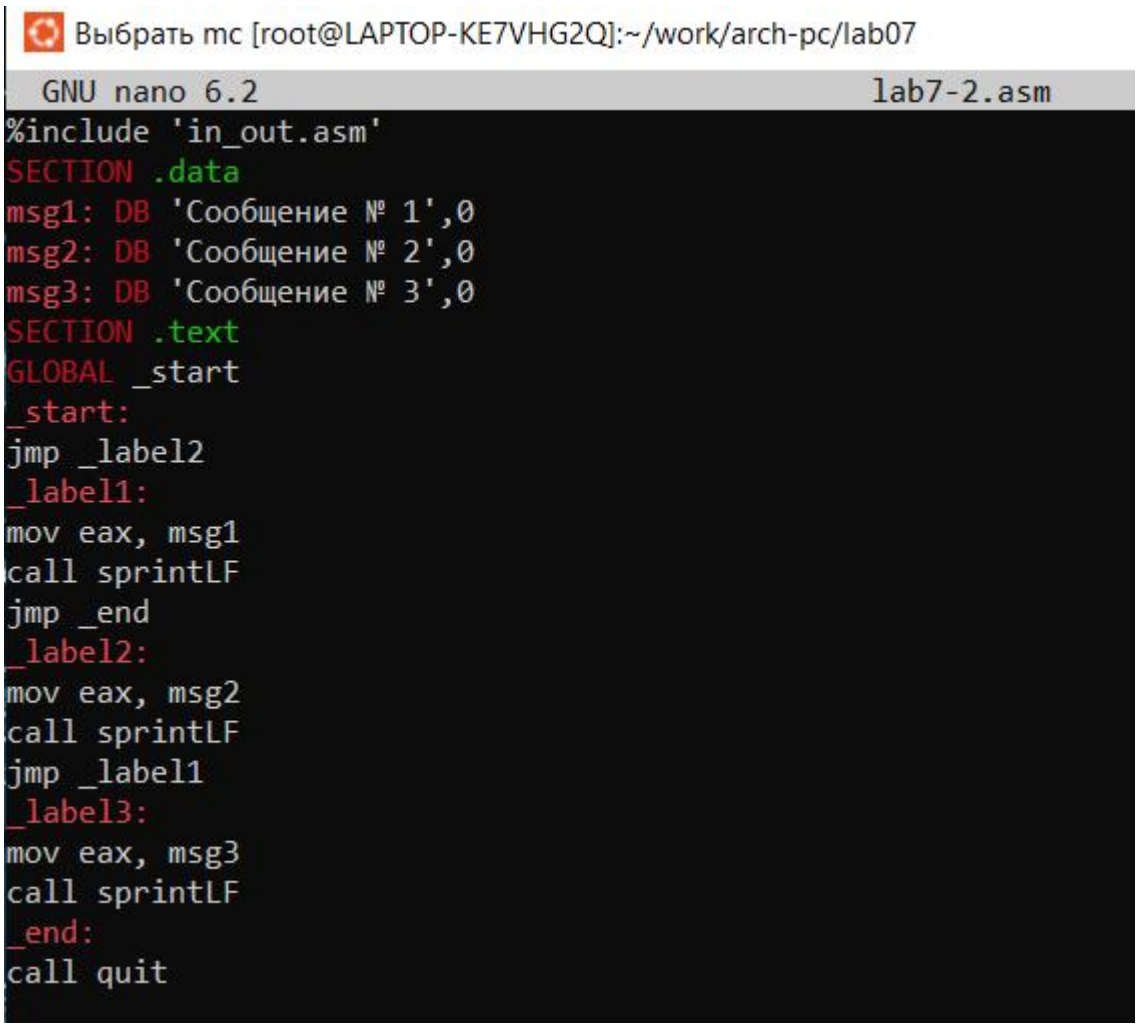
Сообщение № 2

Сообщение № 3

root@LAPTOP-KE7VHG2Q:~#

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# nasm -f elf32 lab7-1.asm -o obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ld -m elf_i386 -o lab7 obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ./lab7
Сообщение № 2
Сообщение № 3
```

Далее в текст программы после вывода сообщения № 2 добавила инструкцию jmp с меткой _label1 и после вывода сообщения № 1 добавила инструкцию jmp с меткой _end .




```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# nasm -f elf32 lab7-2.asm -o obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ld -m elf_i386 -o lab7 obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ./lab7
Сообщение № 2
Сообщение № 1
```

Затем создала исполняемый файл и проверила его работу. Потом изменила текст программы добавив или изменив инструкции jmp, чтобы вывод программы был следующим:

root@LAPTOP-KE7VHG2Q:~# ./lab7

Сообщение № 3

Сообщение № 2

Сообщение № 1

root@LAPTOP-KE7VHG2Q:~#

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ./lab7
Сообщение № 3
Сообщение № 2
Сообщение № 1

root@LAPTOP-KE7VHG2Q:~#
```

После создала файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. И я внимательно изучилла текст программы из листинга 7.3 и ввела в lab7-2.asm

Далее я создала исполняемый файл и проверила его работу для разных значений B.

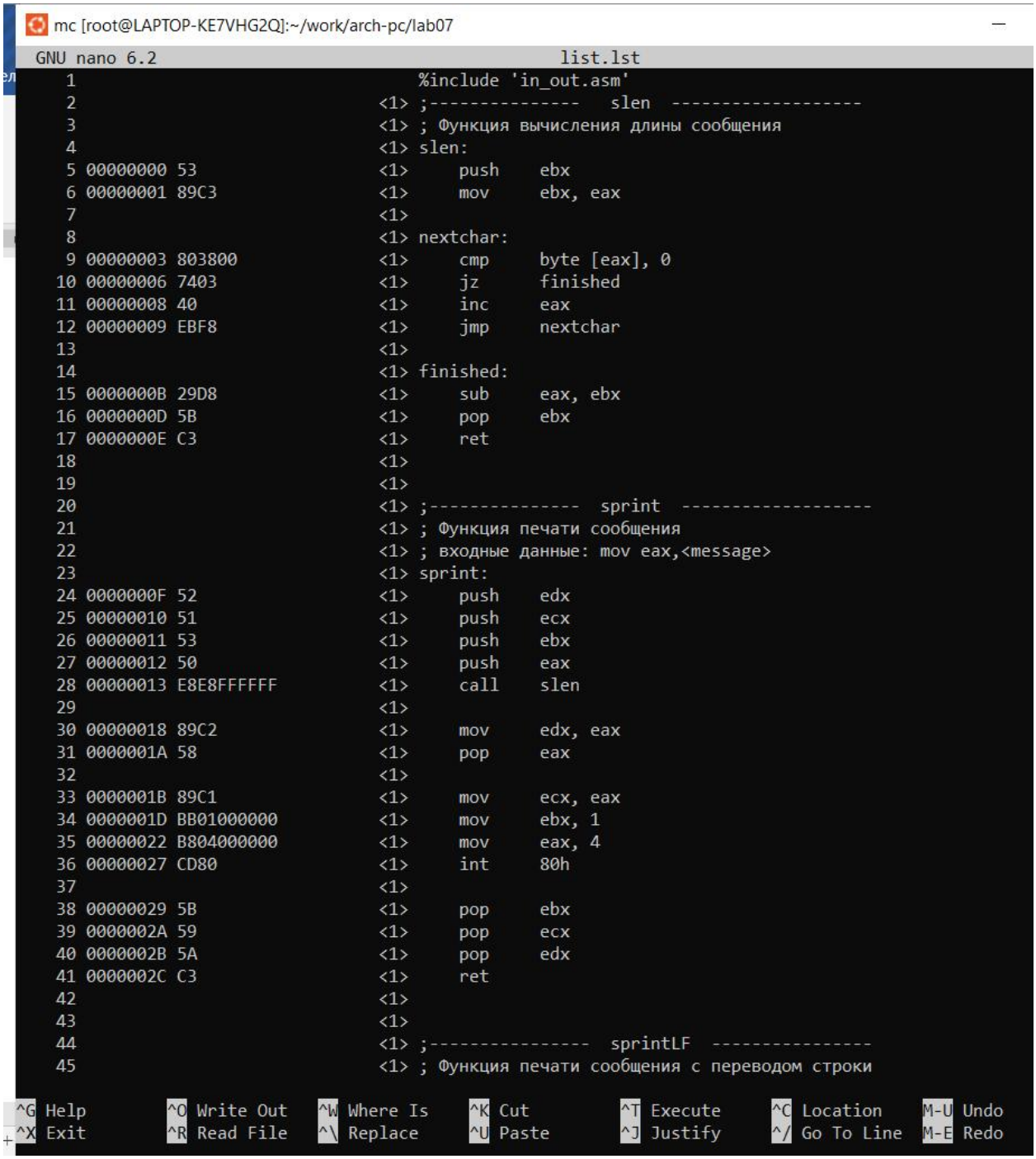
```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# nasm -f elf32 lab7-2.asm -o obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ld -m elf_i386 -o lab7 obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ./lab7
Введите B: 3
Наибольшее число: 50

root@LAPTOP-KE7VHG2Q:~#
```

Потом создала файл листинга для программы из файла lab7-2.asm

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

Открыла файл листинга list.lst с помощью текстового редактора nano:

nano list.lst



Я открыла файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполнила трансляцию с получением файла листинга: nasm -f elf -l list.lst lab7-2.asm -o obj.o.

Какие выходные файлы создаются в этом случае?

list.lst

obj.o

Что добавляется в листинге?

Файл листинга позволяет увидеть работу компилятора FASM: что генерирует каждая строка исходного кода, сколько байт занимают машинные команды, к акие значения присваиваются переменным. Листинг состоит из трех колонок: первая содержит адреса (точнее, смещения в секции), вторая — сгенерирован ный машинный код и третья — соответствующие строчки исходного кода пр ограммы.

1. Написала программу нахождения наименьшей из 3 целочисленных переменных a,b и . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7.

Создала исполняемый файл и проверила его работу.

2. Написала программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрала из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создала исполняемый файл и проверила его работу для значений x и a из 7.6.

с

mc [root@LAPTOP-KE7VHG2Q]:~/work/arch-pc/lab07

GNU nano 6.2 lab7-3.asm

```
%include 'in_out.asm'
section .data
    A db 95
    B db 2
    C db 61
    msg db "Наибольшее число: ", 0

section .bss
    max resb 1

section .text
    global _start

_start:
    ; Сравниваем A и B
    mov al, [A]
    cmp al, [B]
    jge check_C
    mov al, [B]

check_C:
    cmp al, [C]
    jge store_max
    mov al, [C]

store_max:
    mov [max], al

    ; Выводим сообщение
    call iprintLF

    ; Завершаем программу
    call quit
```

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# nasm -f elf32 lab7-3.asm -o obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ld -m elf_i386 -o lab7 obj.o

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab07# ./lab7
95

root@LAPTOP-KE7VHG2Q:~#
```

Вывод.

Целью было изучить команды условного и безусловного переходов и приобрести навыки написания программ с использованием переходов, а также знакомство с назначением и структурой файла листинга, сделав задания, я смога разобраться в данной теме.

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 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — 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, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. 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.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).