**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7.**

дисциплина: *Архитектура компьютеров*

Студент: Подхалюзина Виолетта Михайловна

Группа: НКАбд-04-24

**МОСКВА**

2024 г.

Оглавление

[1 Цель работы 2](#_Toc1821261303)

[2 Введение 3](#_Toc1341810899)

[3 Выполнение лабораторной работы 4](#_Toc780144944)

[3.1 Начало работы 5](#_Toc1698033197)

[3.2 Самостоятельная работа 8](#_Toc326217467)

[4 Контрольные вопросы для самопроверки 10](#_Toc1639664171)

[5 Список литературы 11](#_Toc165038948)

# Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. И познакомиться с назначением и структурой файла листинга.

# 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 определенную точку программы в зависимости от проверки условия.

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

Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. В следующей таблице указано положение битовых флагов в регистре флагов.

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.

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Итак, структура листинга:

• номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);

• адрес — это смещение машинного кода от начала текущего сегмента;

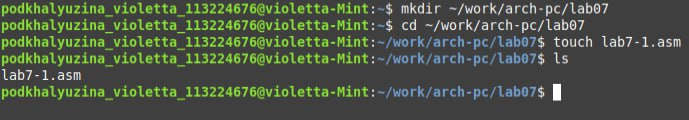
• машинный код представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра);

• исходный текст программы — это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

# Выполнение лабораторной работы

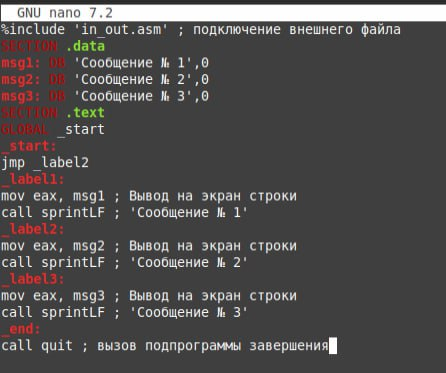
# Начало работы

Я создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm:

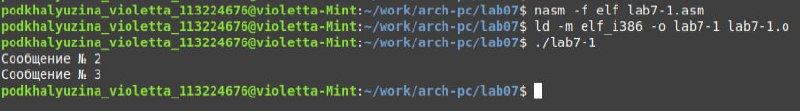


Копирую код из листинга в файл будущей программы:

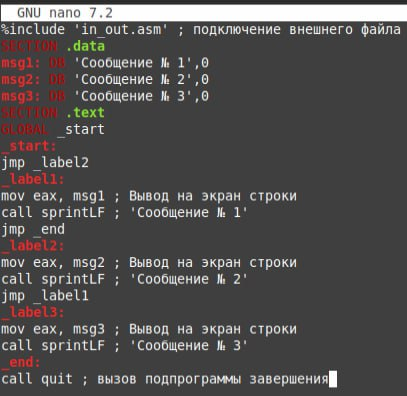




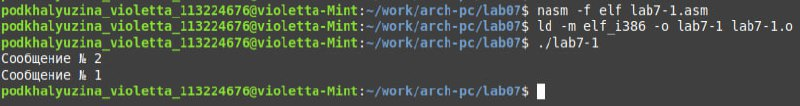
Создаю исполняемый файл, запускаю его и убеждаюсь, что при запуске программы неусловный переход действительно изменяет порядок выполнения инструкций:



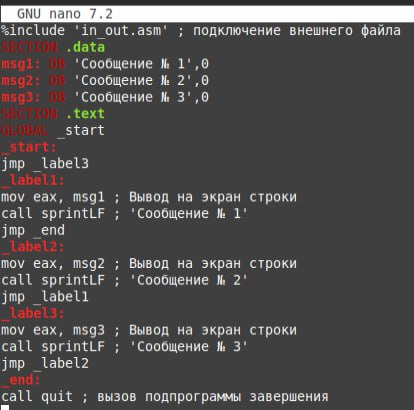
Изменяю программу таким образом, чтобы поменялся порядок выполнения функций:

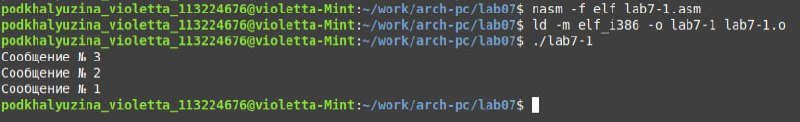


Далее проверяю, работают ли изменения:

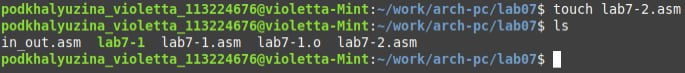


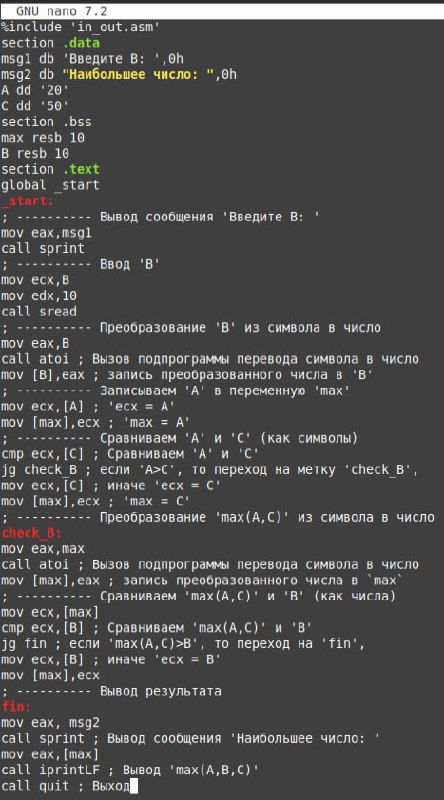
После этого изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке:



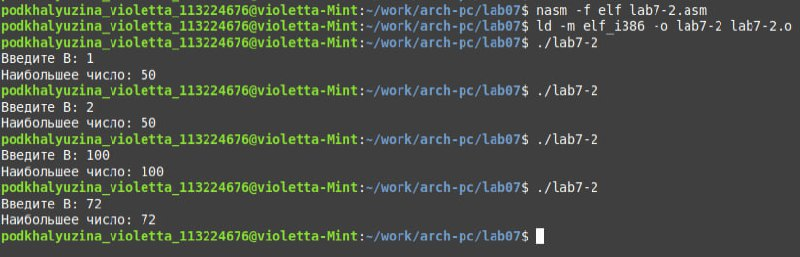


Далее создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07, изучаю текст программы из листинга 7.3 и ввожу в lab7-2.asm:





Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными:



Создаю файл листинга для программы из файла lab7-2.asm и открываю файл листинга lab7-2.lst:



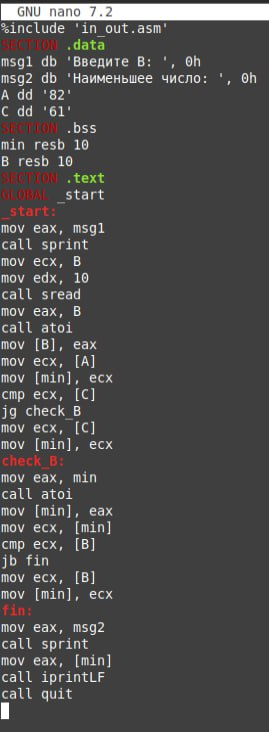
Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст прогарммы с комментариями. Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем:

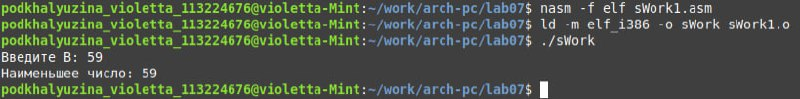
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются:

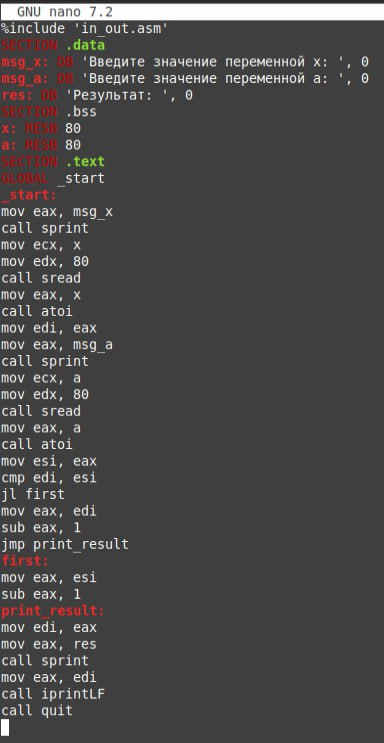
# 3.2 Самостоятельная работа

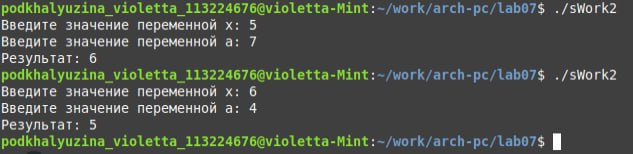
Я написала программу нахождения наименьшей из 3 целочисленных переменных 𝑎,𝑏 и . Значения переменных выбрала из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создала исполняемый файл и проверила его работу.

А затем, я 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 проверьте его работу для значений 𝑥 и 𝑎 из 7.6.









Вывод: я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. И познакомилась с назначением и структурой файла листинга.

# Контрольные вопросы для самопроверки

1. Файл листинга NASM нужен для отладки программы. В отличие от текста программы, он содержит дополнительную информацию: адреса команд, машинный код и их соответствие исходному коду, а также возможные сообщения об ошибках.

2. Формат файла листинга NASM текстовый. Он состоит из номера строки, адреса, машинного кода и исходного текста программы. Например, строка может включать номер строки в листинге, смещение команды от начала сегмента, её машинный код в шестнадцатеричном виде и строку исходного кода программы.

3. В программах на ассемблере ветвление реализуется с помощью команд передачи управления (переходов). Они делятся на условные и безусловные. Условные переходы зависят от значения флагов, безусловные всегда передают управление в указанную точку программы.

4. Безусловный переход выполняется с помощью команды jmp. Условные переходы, такие как je (если равно) или jne (если не равно), проверяют состояние флагов процессора после предыдущей команды.

5. Команда cmp сравнивает два операнда, выполняя их вычитание, но результат не сохраняется. Она лишь выставляет флаги, такие как Zero Flag (ZF) или Sign Flag (SF), для дальнейшего использования командами условного перехода.

6. Синтаксис команд условного перехода: j<мнемоника> <метка>. Например, je label выполнит переход на метку label, если результат сравнения равен нулю (ZF=1).

7. Пример: cmp ax, bx ; сравниваются регистры je equal ; переход на метку equal, если ax = bx ... equal: mov cx, ax ; выполняется, если ax = bx

8. Команды безусловного перехода не анализируют флаги. Они передают управление в указанную точку программы независимо от текущего состояния процессора.



# 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/Learningbash-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, 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).