Шаблон отчёта по лабораторной работе

Архитектура компьютеров и операционных систем

Виктор Ващаев Андреевич

Содержание

# 1 Цель работы

Цель работы:

Разработать программу на языке ассемблера, которая принимает на вход аргументы командной строки, вычисляет значения заданной функции для каждого аргумента, а затем выводит их сумму. Целью является закрепление навыков работы с функциями, стеком, арифметическими операциями, системными вызовами, а также работа с аргументами командной строки в ассемблере.

# 2 Задание

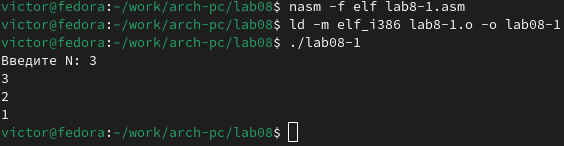
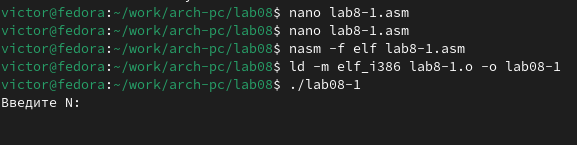
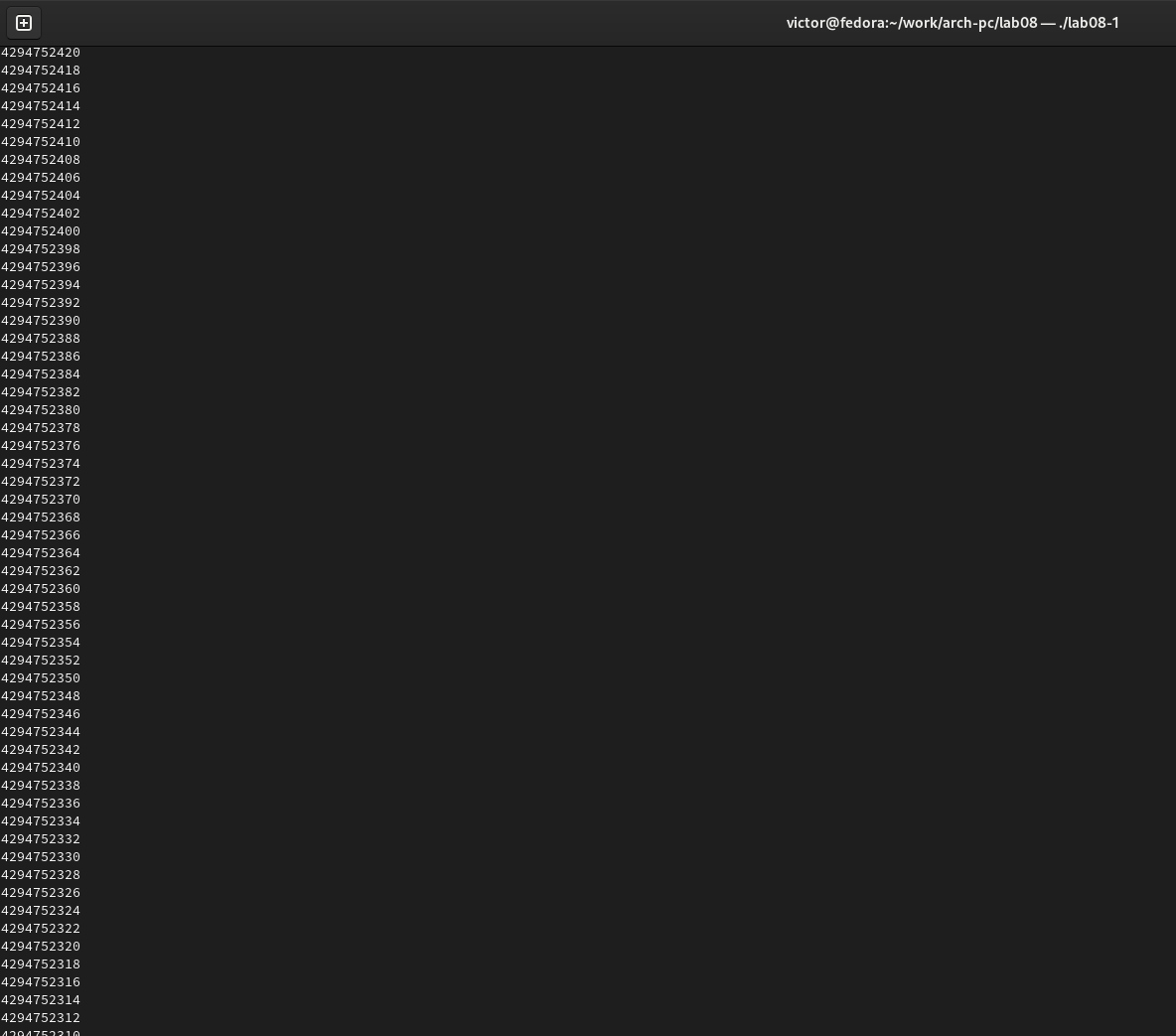
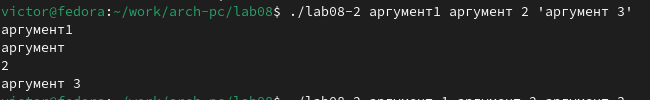
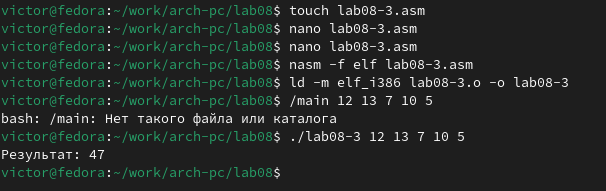
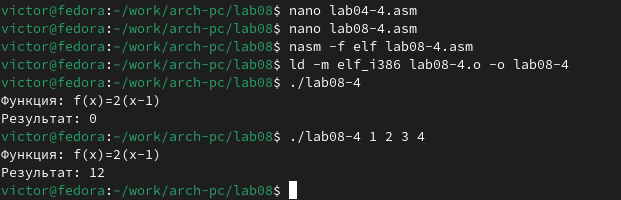
Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

# 3 Теоретическое введение

Теоретическое введение

Ассемблер — это язык низкого уровня, который позволяет напрямую взаимодействовать с аппаратным обеспечением компьютера. Он обеспечивает полный контроль над работой процессора, памяти и других компонентов системы. В данной работе используется синтаксис NASM (Netwide Assembler), одного из самых популярных ассемблеров для разработки приложений под Linux. Ассемблерный код позволяет эффективно выполнять низкоуровневые операции, такие как обработка данных, управление стеком и взаимодействие с операционной системой через системные вызовы. # Выполнение лабораторной работы

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. **¿fig:001?**).

 Первый скриншот: • Выполняется команда nasm -f elf lab8-1.asm для компиляции исходного файла lab8-1.asm в объектный файл lab8-1.o. • Затем команда ld -m elf\_i386 lab8-1.o -o lab8-1 связывает объектный файл в исполняемый файл lab8-1. • Исполняемый файл запускается командой ./lab8-1, после чего программа ожидает ввода значения N.  Второй скриншот: • Происходит открытие файла lab8-1.asm с помощью текстового редактора (nano). • Снова компиляция и связывание с помощью nasm и ld. • Запуск программы ./lab8-1, которая снова ждет ввода значения N.  Третий скриншот: • После ввода значения 3 программа выполняет вывод результатов (возможно, последовательности чисел или значений, связанных с расчетами).  Четвертый скриншот: • Исполняемый файл запускается с несколькими аргументами: аргумент1, аргумент2, аргумент3. • Программа считывает и выводит каждый из аргументов на экран.  Пятый скриншот: • Создается пустой файл lab08-3.asm с помощью команды touch lab08-3.asm. • Файл редактируется через nano lab08-3.asm. • Компиляция выполняется с помощью команды nasm -f elf lab08-3.asm, создавая объектный файл lab08-3.o. • Связывание исполняемого файла выполняется командой ld -m elf\_i386 lab08-3.o -o lab08-3. • Попытка запуска ./main завершается ошибкой, так как файл main отсутствует. • Исполняемый файл запускается с аргументами 12 13 7 10 5 командой ./lab08-3. Программа вычисляет результат (сумму значений функции) и выводит: Результат: 47.  Самостоятельная работа: • Открывается файл lab08-4.asm для редактирования через nano. • Компиляция выполняется командой nasm -f elf lab08-4.asm, создавая объектный файл lab08-4.o. • Связывание исполняемого файла выполняется командой ld -m elf\_i386 lab08-4.o -o lab08-4. • Программа запускается без аргументов: ./lab08-4, функция выводит результат 0, так как аргументов нет. • Программа запускается с аргументами 1 2 3 4 командой ./lab08-4 1 2 3 4. Вычисляется сумма значений функции , и выводится результат: Результат: 12.

# 4 Выводы

Выводы

1. Работа с циклами: Команда loop упрощает реализацию циклов, но аналогичные циклы можно организовать с использованием условных переходов, таких как cmp и jne/jz.
2. Использование стека: Стек предоставляет эффективный способ временного хранения данных. Он работает по принципу LIFO (последним пришел — первым вышел), что особенно полезно для передачи параметров и хранения промежуточных значений.
3. Аргументы командной строки: Работа с аргументами в ассемблере требует их считывания из стека, преобразования и обработки, что демонстрирует низкоуровневое управление данными.
4. Практическое применение: Реализация программы для вычисления функции показала важность модульности кода, использования функций и взаимодействия с операционной системой через системные вызовы. # Список литературы{.unnumbered}