Отчёт по лабораторной работе №10

Дисциплина: Архитектура компьютера

Барсегян Вардан Левонович НПИбд-01-22

Содержание

# Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

# Выполнение лабораторной работы

## Реализация подпрограмм в NASM

1. Создаю каталог для выполнения лабораторной работы № 10, перехожу в него и создаю файл *lab10-1.asm* (рис. 1)

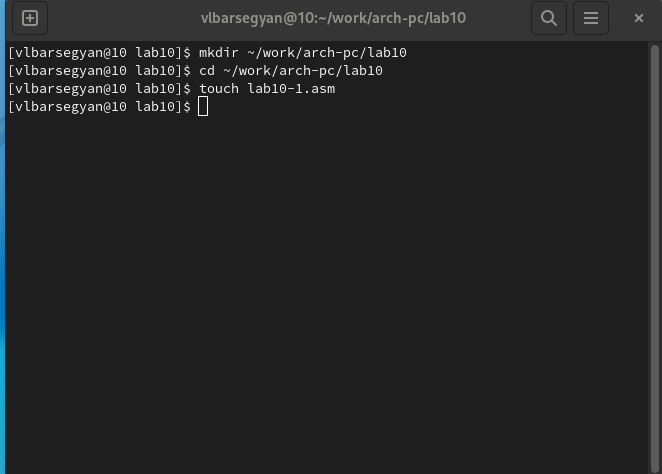


Рис. 1: Создание каталога и файла

1. Ввожу в файл текст программы листинга 1 - программа вычисления арифметического выражения 𝑓(𝑥) = 2𝑥 + 7 с помощью подпрограммы \_calcul. В данном примере 𝑥 вводится с клавиатуры, а само выражение вычисляется в подпрограмме (рис. 2), компилирую и запускаю исполняемый файл (рис. 3)

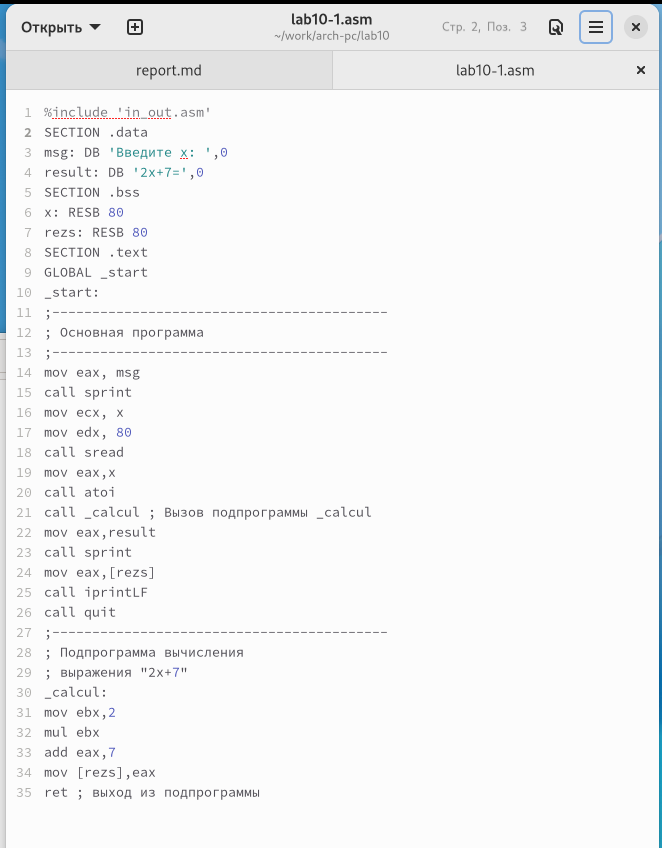


Рис. 2: Текст программы листинга 1

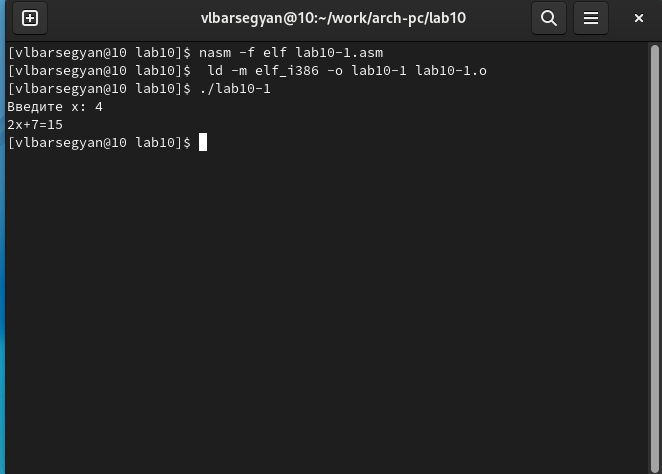


Рис. 3: Компиляция и запуск исполняемого файла

1. Изменяю текст программы, добавив подпрограмму \_subcalcul (рис. 4). Компилирую исполняемый файл и проверяю его работу (рис. 5)

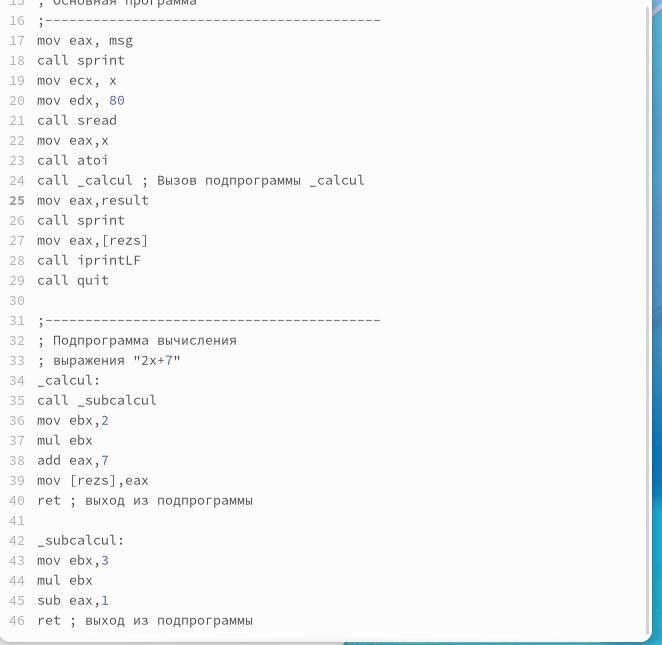


Рис. 4: Обновленный екст программы

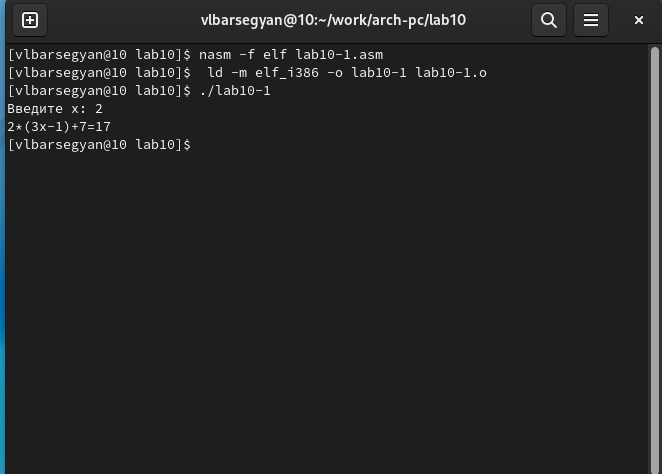


Рис. 5: Создание и запуск исполняемого файла

## Отладка программам с помощью GDB

1. Создаю файл *lab10-2.asm* с помощью команды *touch lab10-2.asm* с текстом программы из Листинга 10.2 (Программа печати сообщения Hello world!) (рис. 6)

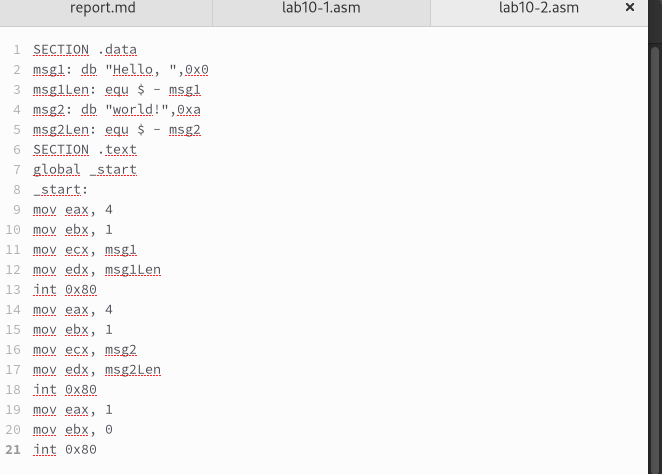


Рис. 6: Текст программы файла lab10-2.asm

1. Создаю исполняемый файл, добавив отладочную информацию с помощью ключа *-g* и загружаю его в отладчик GDB (рис. 7)

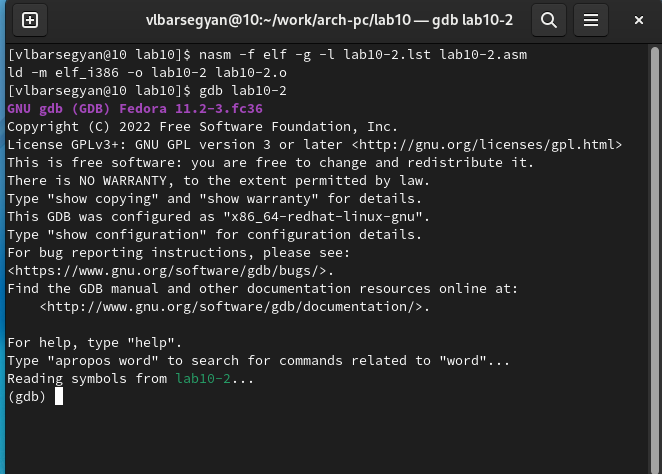


Рис. 7: Компиляция исполняемого файла и его загрузка в отладчик GDB

1. Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 8)

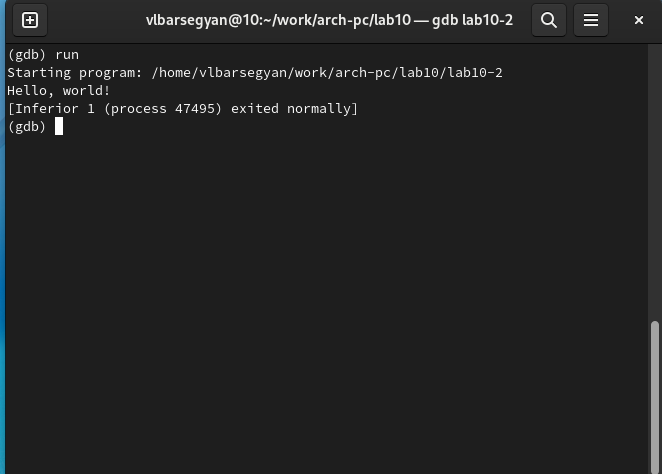


Рис. 8: Запуск программы в оболочке GDB

1. Устанавливаю брейкпоинт на метку \_start и запускаю её (рис. 9)

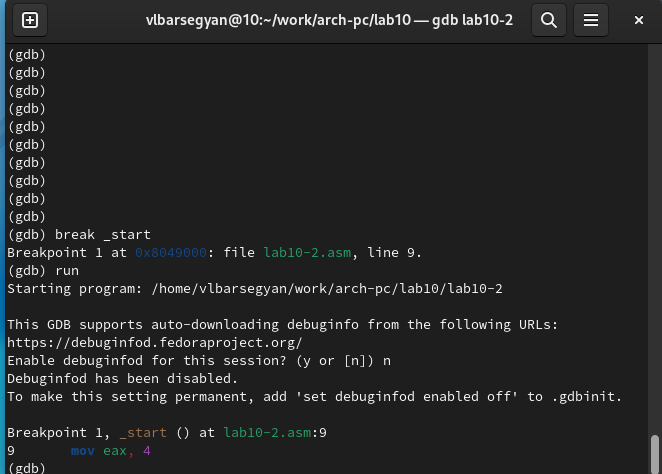


Рис. 9: Установка брейкпоинта и запуск

1. Смотрю дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 10)

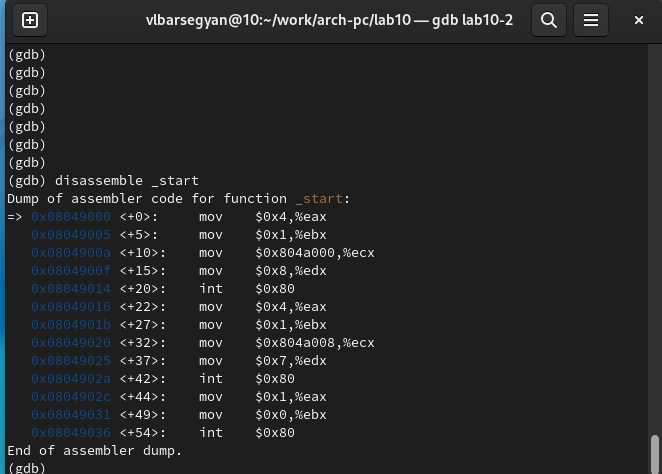


Рис. 10: Дисассимилированный код программы

1. Переключаюсь на отображение команд с Intel’овским синтаксисом, введя команду *set disassembly-flavor intel* (рис. 11). После смены интерфейса команды отображаются с привычным Intel’овским синтаксисом

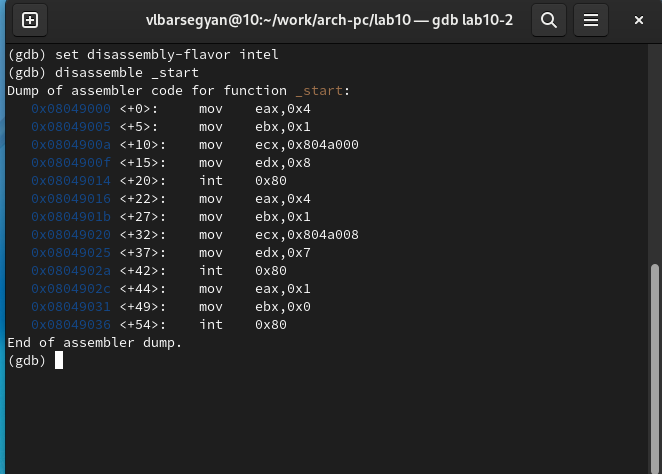


Рис. 11: Дисассимилированный код программы в режиме Intel

1. Включаю режим псевдографики для более удобного анализа программы (рис. 12, 13)

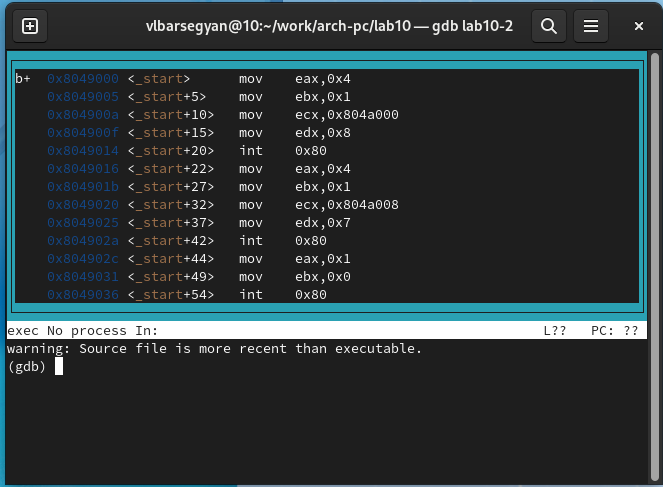


Рис. 12: Включение режима псевдографики

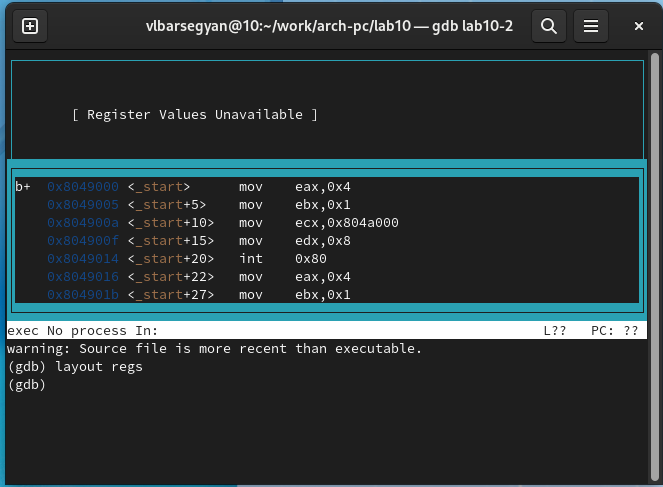


Рис. 13: Включение режима псевдографики

### Добавление точек останова

1. Проверяю установку точки остановы по имени метки (\_start) с помощью команды *i b* (рис. 14)

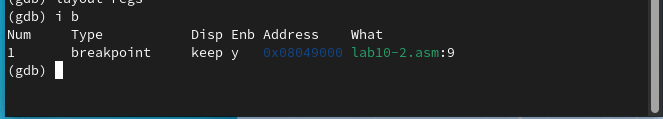


Рис. 14: Проверка установки точки

1. Адрес предпоследней инструкции - 0x8049031. Устанавливаю точку остановы по этому адресу и проверяю все такие точки (рис. 15)

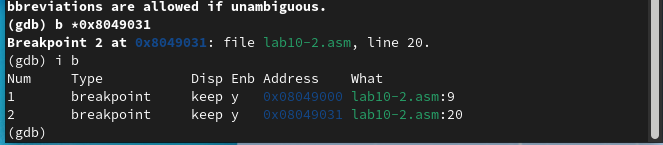


Рис. 15: Установка точки остановы и проверка все таких точек

### Работа с данными программы в GDB

1. Выполняю 5 инструкций stepi (si). Каждый раз выполняется соответствующая инструкция, и подсвечивается соответствующее изменение (рис. 16)

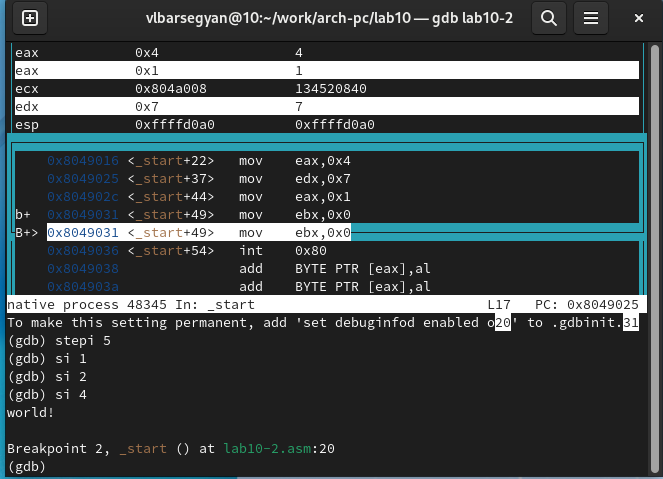


Рис. 16: Выполнение инструкции stepi

1. Посмотреть содержимое регистров также можно с помощью команды info registers (или i r). (рис. 17)

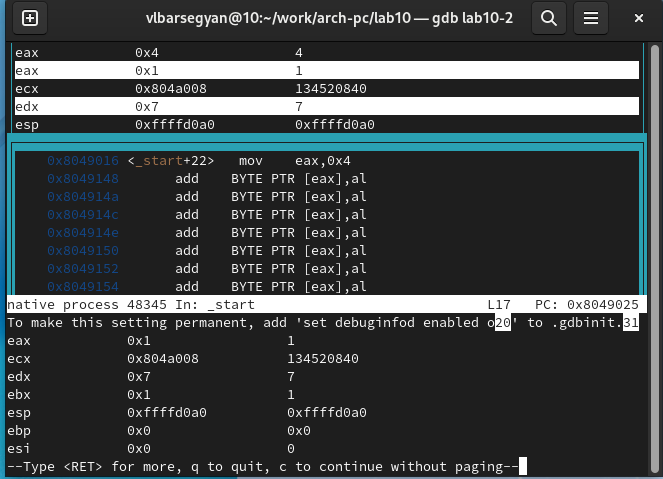


Рис. 17: Просмотр содержимого регистров

1. Смотрю значение переменной msg1 по имени, а значение переменной msg2 - по адресу (рис. 18)

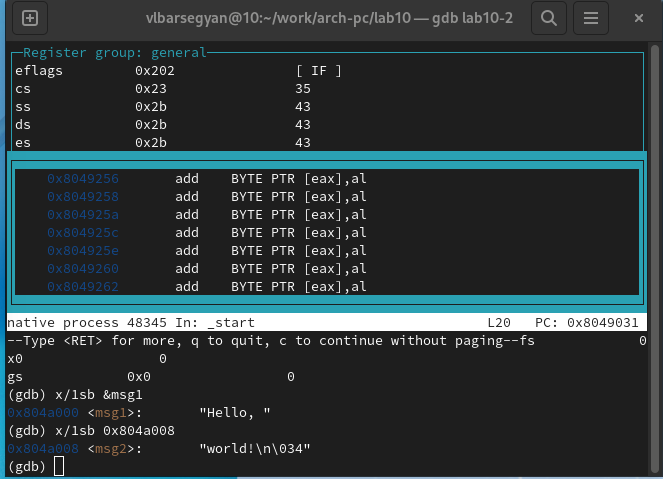


Рис. 18: Просмотр содержимого переменных

1. Изменяю первый символ переменной msg1 и два символа в переменной msg2, используя ее адрес (рис. 19)

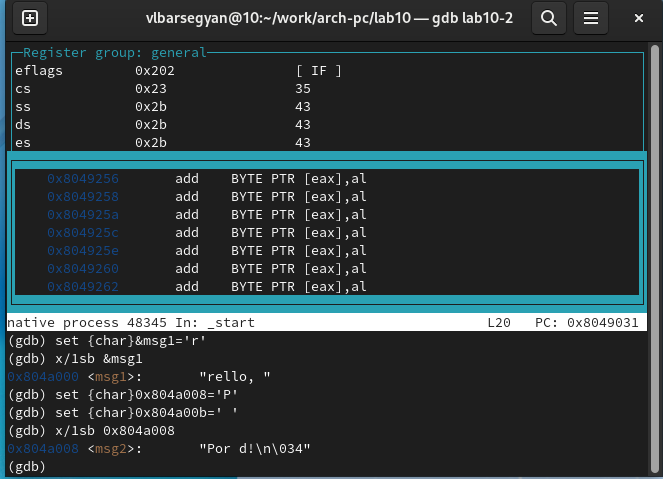


Рис. 19: Изменение символов переменных

1. Вывожу значение регистра edx в различных форматах (p/x - шестнадцатеричный, p/t - двоичный, p/s - символьный) (рис. 20)

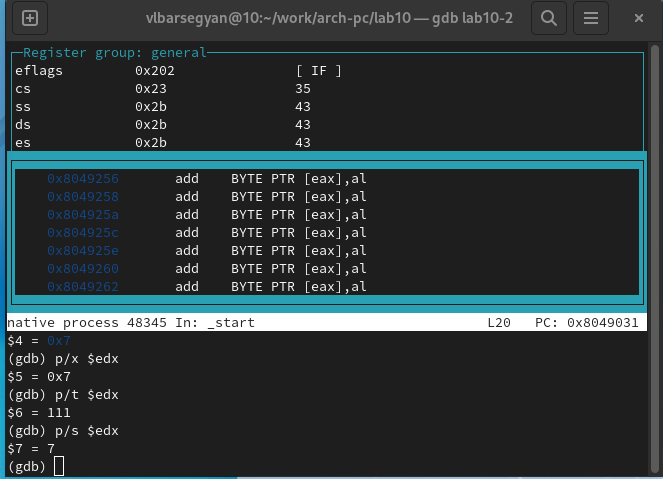


Рис. 20: Вывод значения регистра в различных форматах

1. С помощью команды set изменяю значение регистра ebx (рис. 21). Разница в выводе в том, что в первом случае 2 вводится как символ, и в символьном виде выводится ascii-номер символа, во втором же случае 2 вводится как число и в символьном виде показывается как число

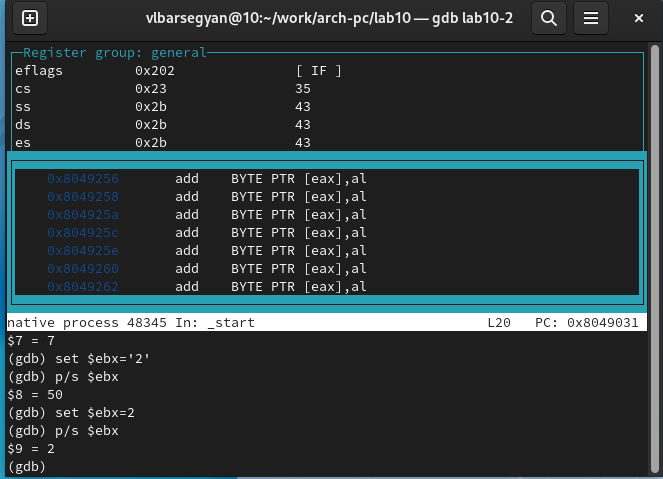


Рис. 21: Изменение значения регистра

1. Продолжаю выполнение программы с помощью команды *с* и выхожу из GDB с помощью команды *q* (рис. 22)

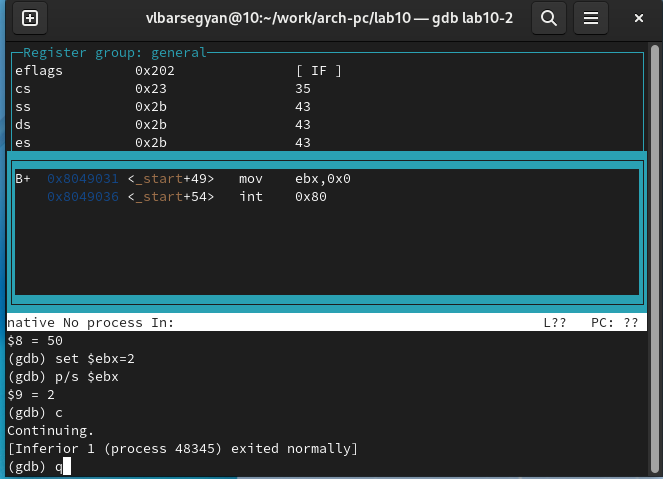


Рис. 22: Продолжение выполнения программы и выход из GDB

### Обработка аргументов командной строки в GDB

1. Копирую файл *lab9-2.asm*, созданный при выполнении лабораторной работы №9 в файл с именем *lab10-3.asm* и создаю исполняемого файла (рис. 23)

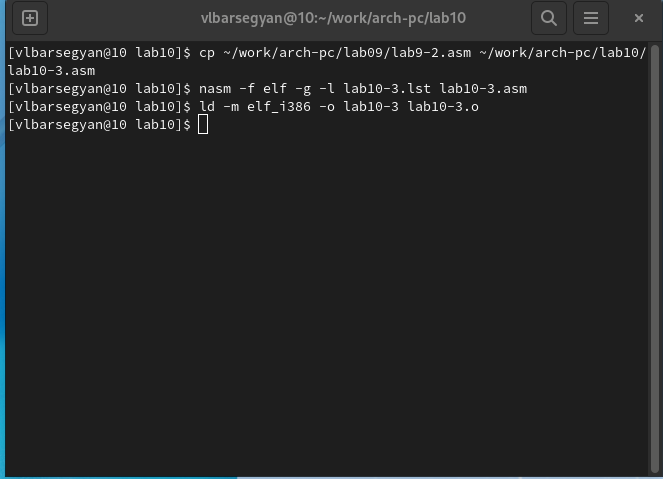


Рис. 23: Копирование файла, создание исполняемого

1. Загружаю исполняемый файл в отладчик, используя следующие аргументы (рис. 24)

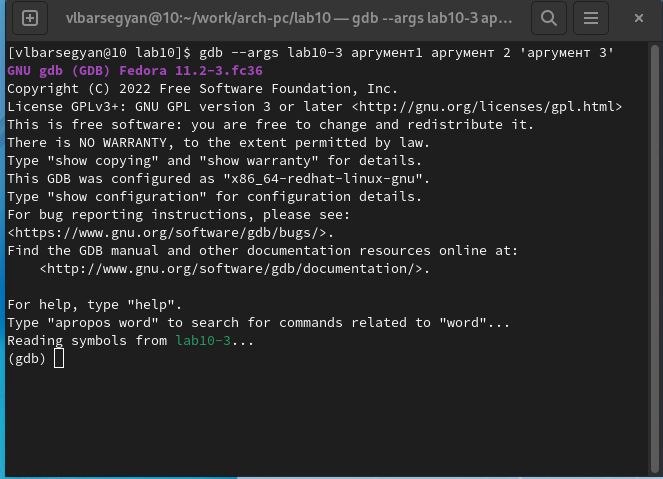


Рис. 24: Загрузка в отладчик

1. Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 25)

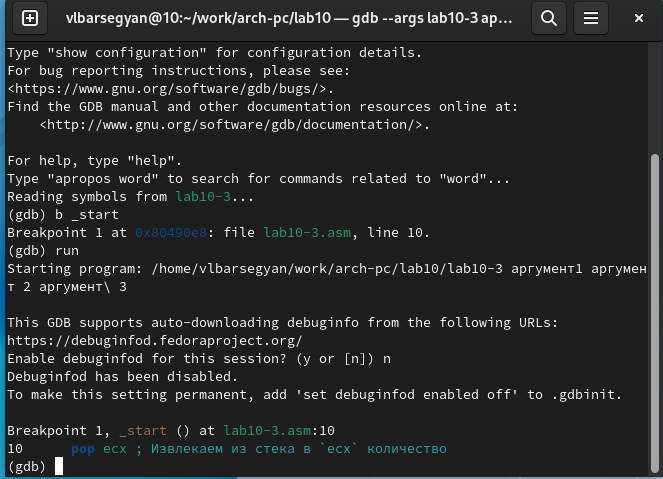


Рис. 25: Установка точки останова и запуск

1. Смотрю позиции стека по их адресам (рис. 26). Шаг изменения адреса равен 4, т.к. размер переменной - 4 байта

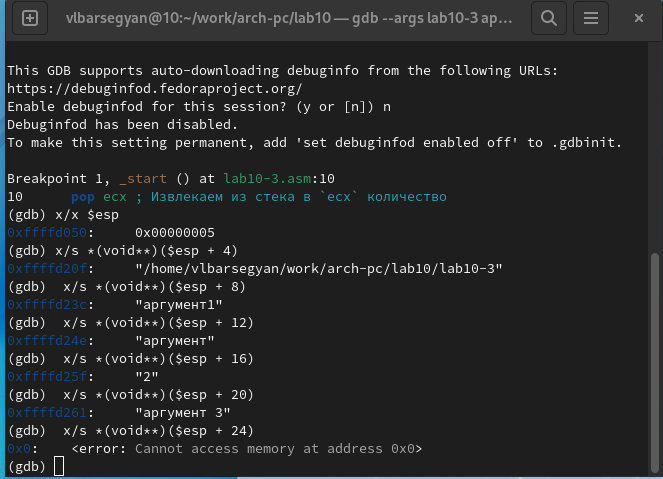


Рис. 26: Позиции стека

# Задания для самостоятельной работы

1. Создаю файл *task1.asm*, пишу текст программы (Преобразуйте программу из лабораторной работы №9 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции 𝑓(𝑥) как подпрограмму) (рис. 27). Компилирую исполняемый файл и проверяю его работу (рис. 28)

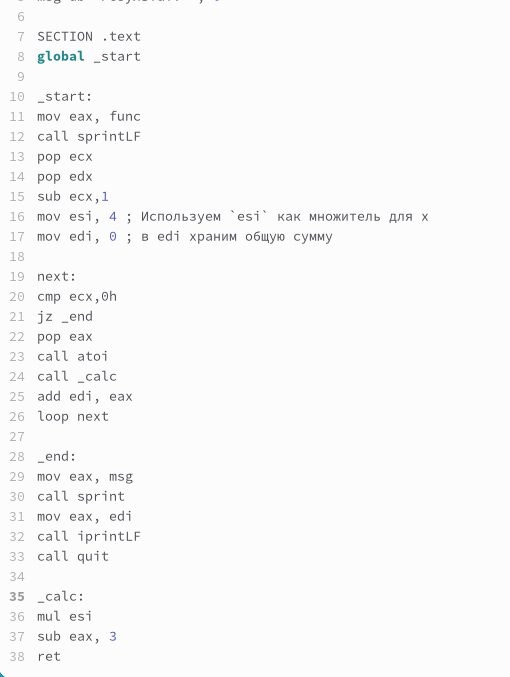


Рис. 27: Текст программы

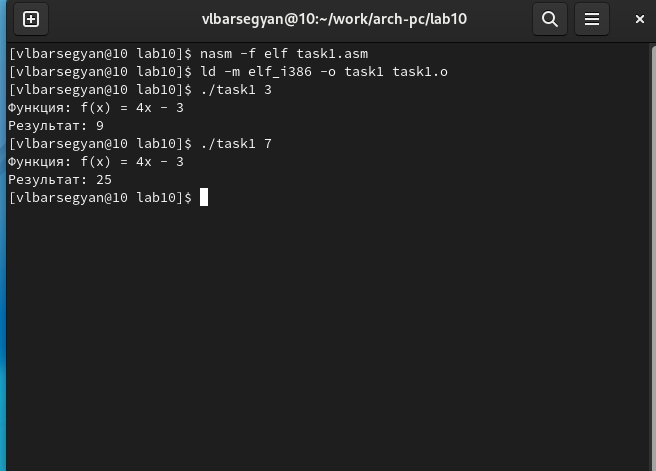


Рис. 28: Проверка программы

1. Создаю файл *task2.asm*, ввожу текст программы из листинга 10.4 (рис. 29)

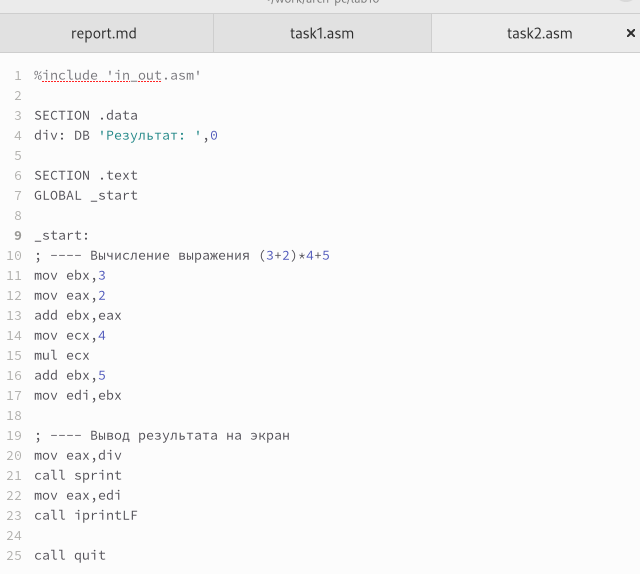


Рис. 29: Текст неправильной программы

1. Открываю отладчик GDB, отлаживаю код (рис. 30). В коде перепутаны местами регистры ebx и eax в команде add, а также в регистр edi копируются данные из регистра ebx, а не eax. Также 5 должно прибавлять к значению регистра eax, а не ebx

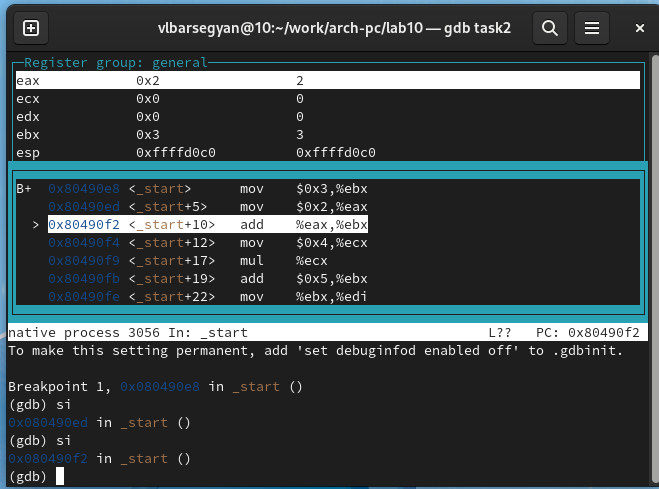


Рис. 30: Отладка программы

1. Исправленный код программы (рис. 31)

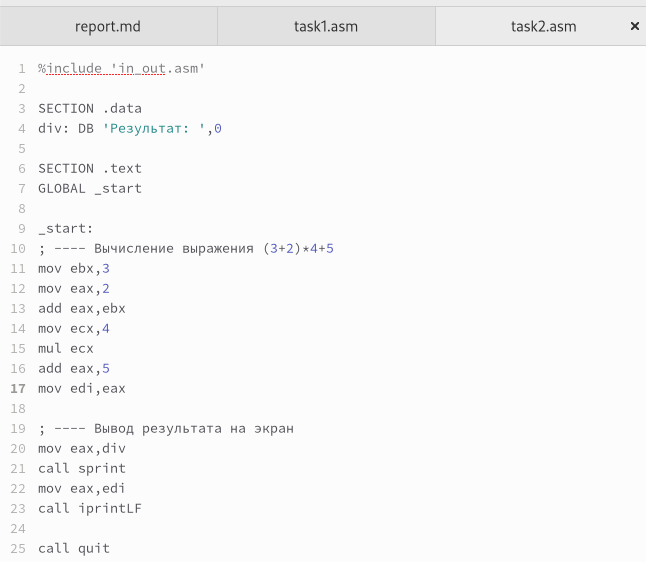


Рис. 31: Исправленный код программы

1. Проверяю правильность работы программы (рис. 32)

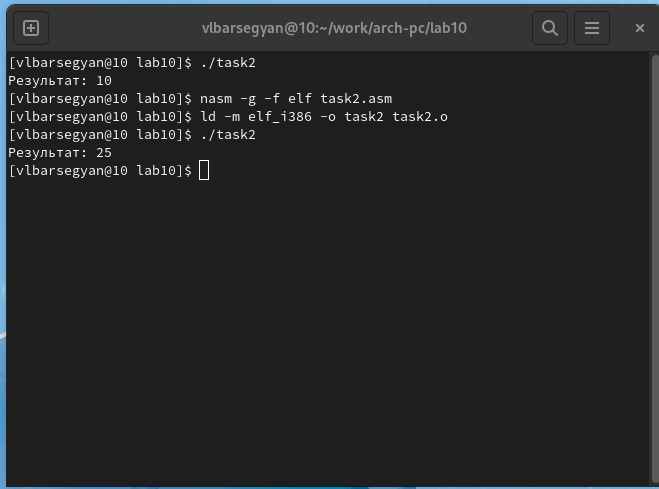


Рис. 32: Проверка работы программы

# Выводы

Я научился писать программы с использованием подпрограмм, переписал старую программу с использованием подпрограммы. Узнал, что такое отладка и как пользоваться отладчиком GDB и познакомился с его основными возможностями