Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютера

Тяпкова Альбина НММбд-04-24

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM

Я создала каталог для выполнения лабораторной работы №9 и перешла в него.

В качестве примера рассмотрим программу, которая вычисляет арифметическое выражение ( f(x) = 2x + 7 ) с использованием подпрограммы calcul. В этом примере значение ( x ) вводится с клавиатуры, а само выражение вычисляется в подпрограмме.

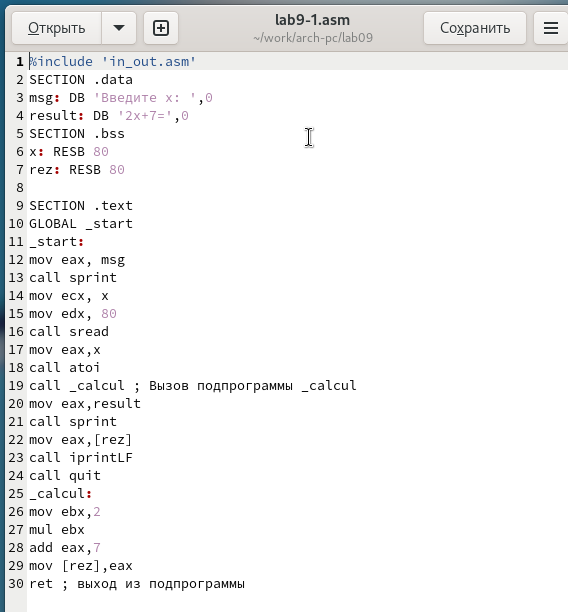


Рис. 1: Программа в файле lab9-1.asm

Первые строки программы отвечают за вывод сообщения на экран (с помощью вызова sprint), чтение данных, введенных с клавиатуры (с помощью вызова sread) и преобразование введенных данных из символьного вида в численный (с помощью вызова atoi).

После инструкции call \_calcul, которая передает управление подпрограмме \_calcul, будут выполнены инструкции, содержащиеся в подпрограмме.

Инструкция ret является последней в подпрограмме и её выполнение приводит к возврату в основную программу к инструкции, следующей за инструкцией call, которая вызвала данную подпрограмму.

Последние строки программы реализуют вывод сообщения (с помощью вызова sprint), вывод результата вычисления (с помощью вызова iprintLF) и завершение программы (с помощью вызова quit).

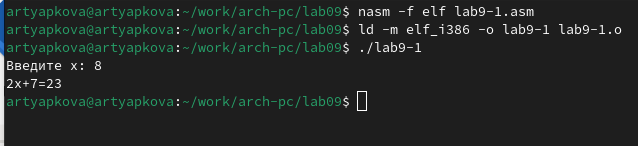


Рис. 2: Запуск программы lab9-1.asm

Я изменила текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения ( f(g(x)) ), где ( x ) вводится с клавиатуры, ( f(x) = 2x + 7, g(x) = 3x - 1 ).

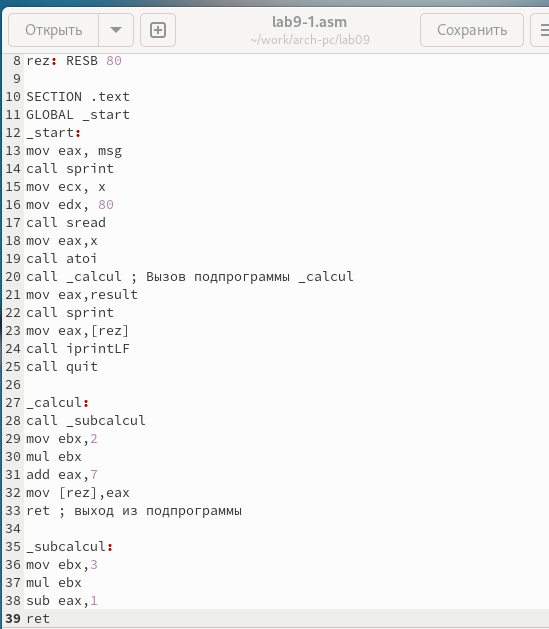


Рис. 3: Программа в файле lab9-1.asm

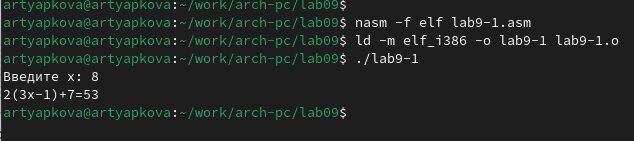


Рис. 4: Запуск программы lab9-1.asm

## 2.2 Отладка программ с помощью GDB

Я создала файл lab9-2.asm с текстом программы из Листинга 9.2 (Программа печати сообщения Hello world!).

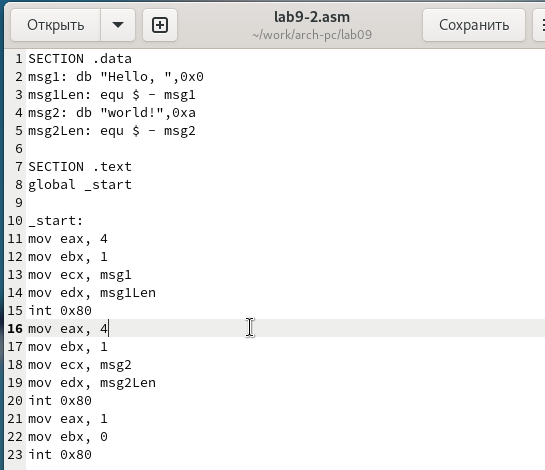


Рис. 5: Программа в файле lab9-2.asm

После того как я получила исполняемый файл, для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для чего трансляцию программ следует проводить с ключом -g.

Загрузила исполняемый файл в отладчик GDB и проверила работу программы, запустив её в оболочке GDB с помощью команды run (сокращенно r).

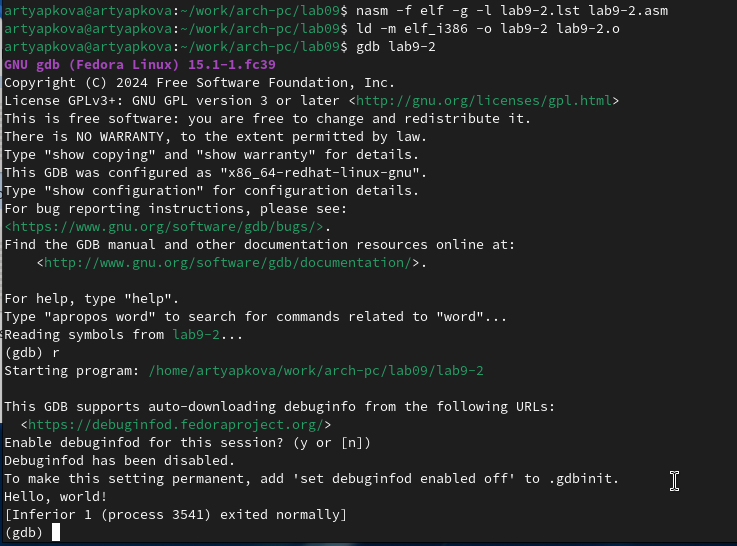


Рис. 6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установила брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустила её. Посмотрела дизассемблированный код программы.

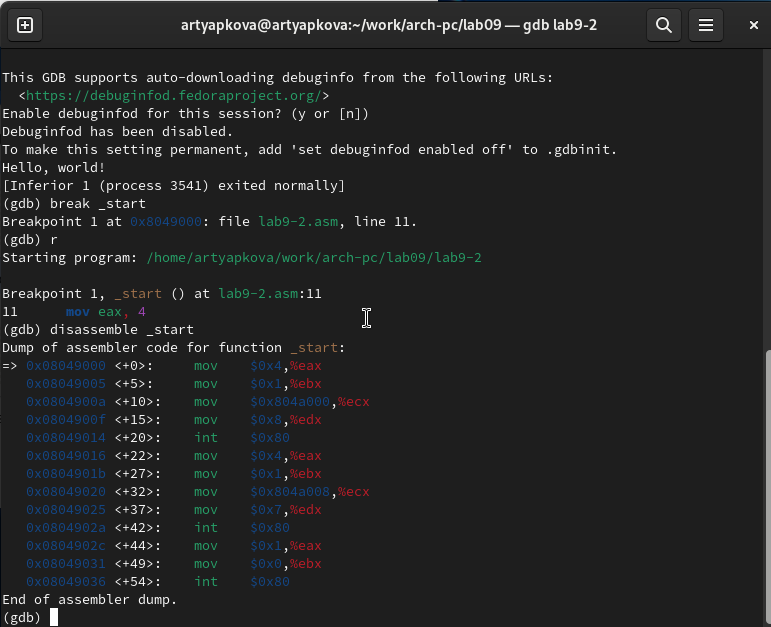


Рис. 7: Дизассемблированный код

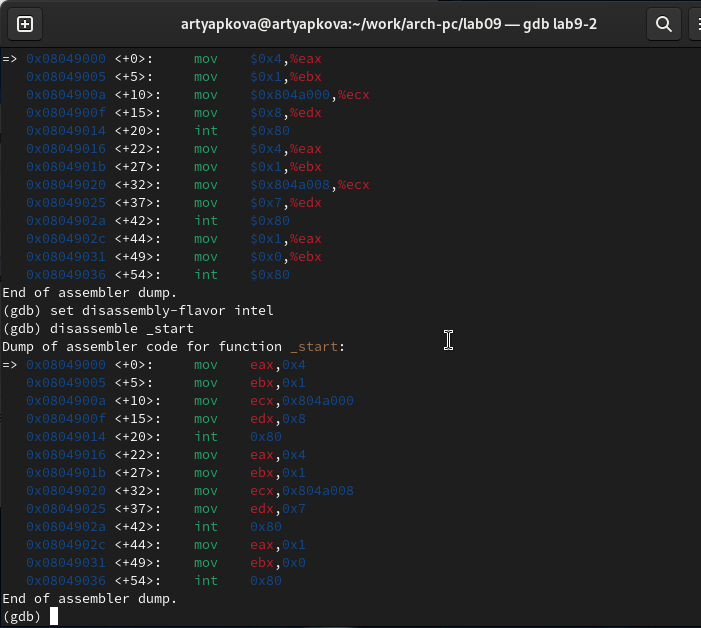


Рис. 8: Дизассемблированный код в режиме интел

Для установки точки останова использовала команду break (кратко b). Типичный аргумент этой команды — место установки. Его можно задать либо как номер строки программы (если есть исходный файл и программа компилировалась с отладочной информацией), либо как имя метки, или как адрес. Чтобы избежать путаницы с номерами, перед адресом ставится «звездочка».

На предыдущих шагах была установлена точка останова по имени метки \_start. Проверила это с помощью команды info breakpoints (кратко i b). Затем установила ещё одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку.

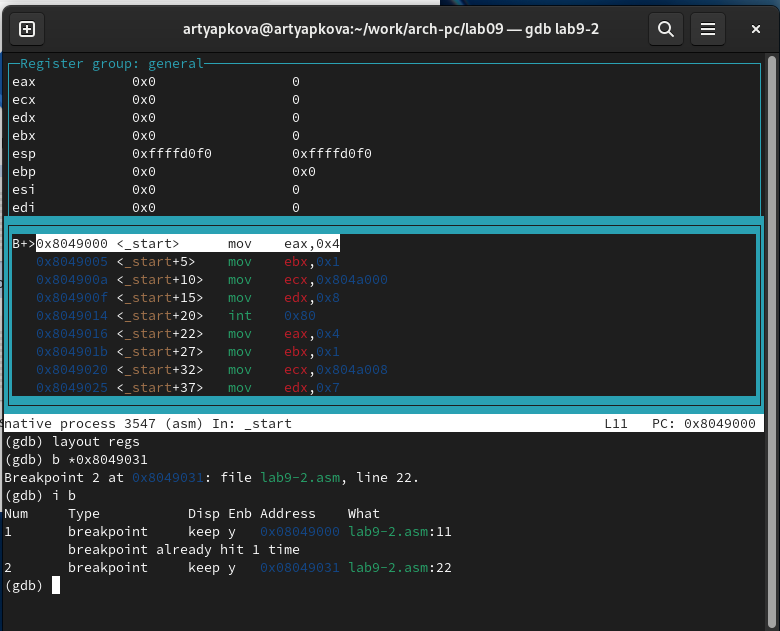


Рис. 9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Я выполнила 5 инструкций с помощью команды stepi (или si) и проследила за изменением значений регистров.

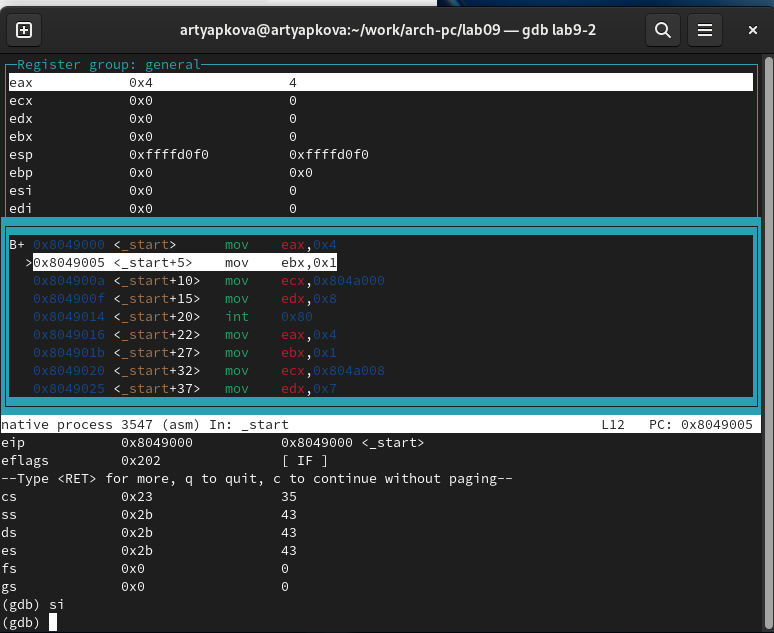


Рис. 10: Изменение регистров

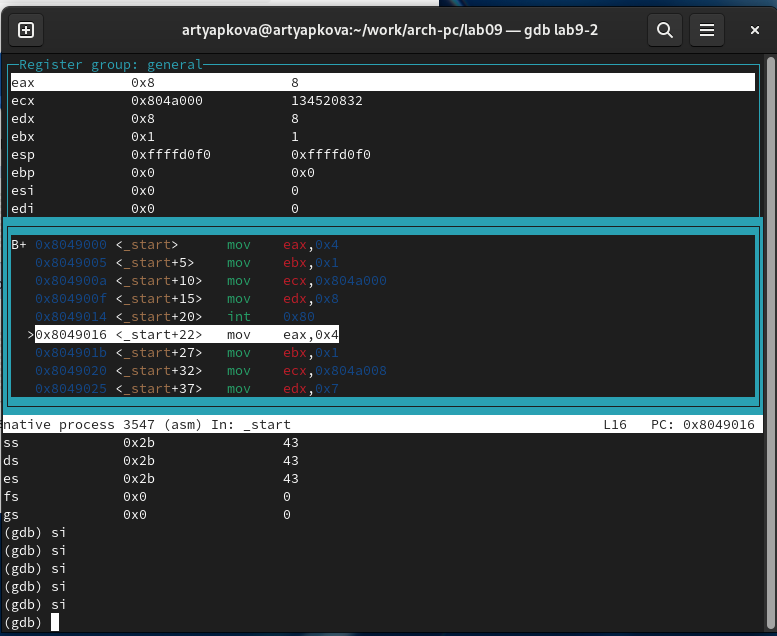


Рис. 11: Изменение регистров

Посмотрела значение переменной msg1 по имени и значение переменной msg2 по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды set, указав имя регистра или адрес. Я изменила первый символ переменной msg1.

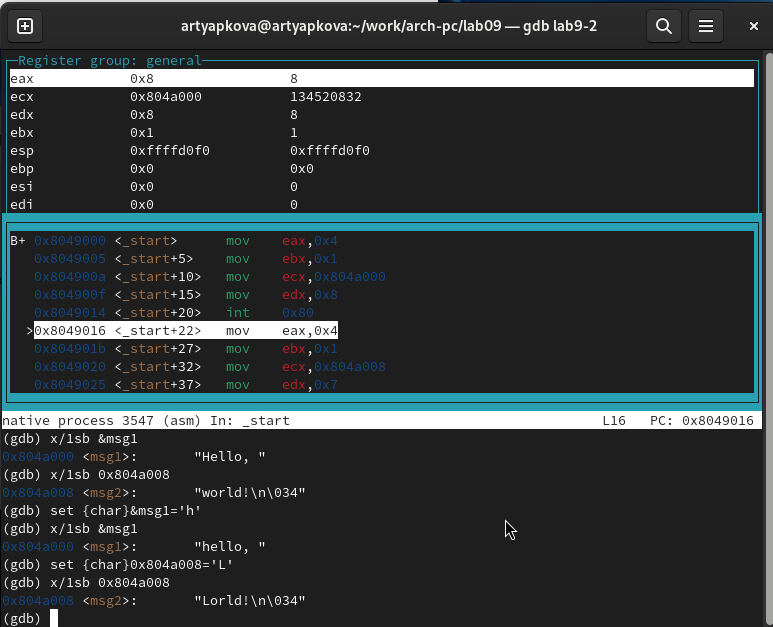


Рис. 12: Изменение значения переменной

Я вывела значение регистра edx в различных форматах (в шестнадцатеричном, двоичном и символьном).

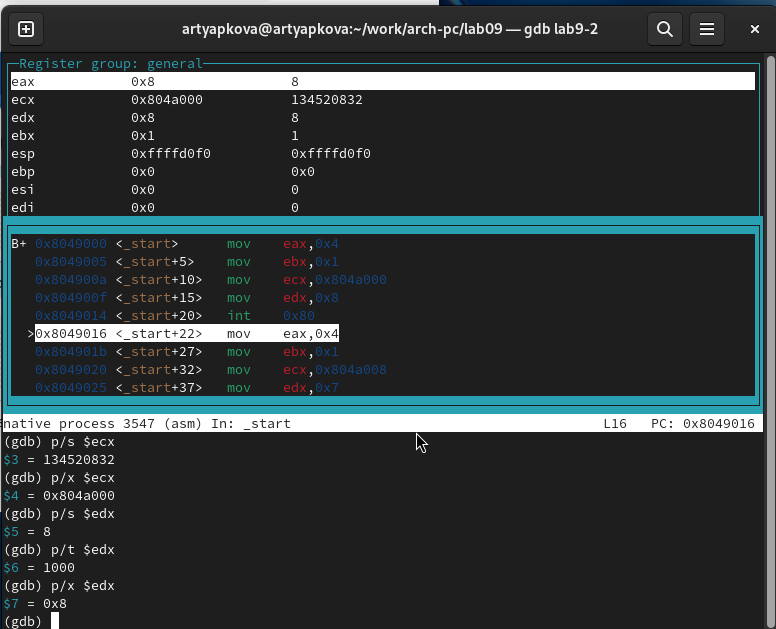


Рис. 13: Вывод значения регистра

С помощью команды set изменила значение регистра ebx.

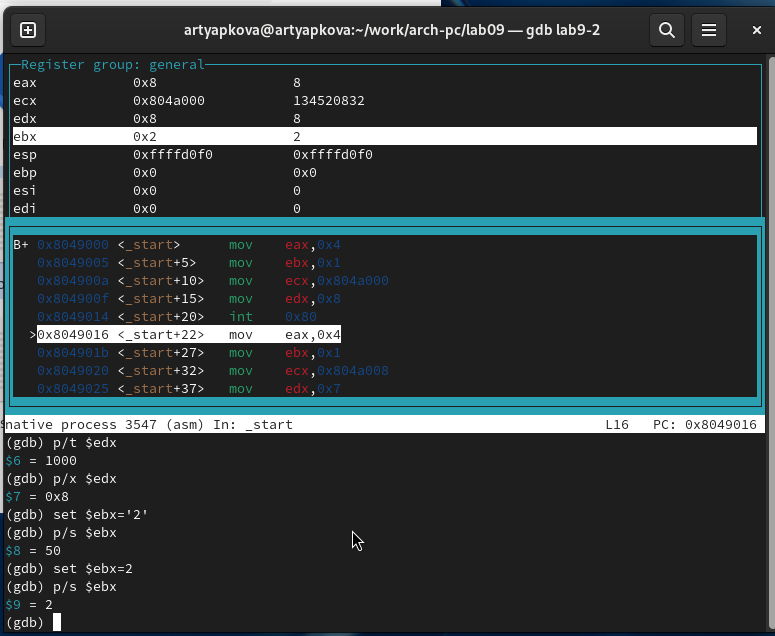


Рис. 14: Вывод значения регистра

Я скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой, выводящей на экран аргументы командной строки. Создала исполняемый файл. Для загрузки программы с аргументами в GDB необходимо использовать ключ --args. Загрузила исполняемый файл в отладчик, указав аргументы.

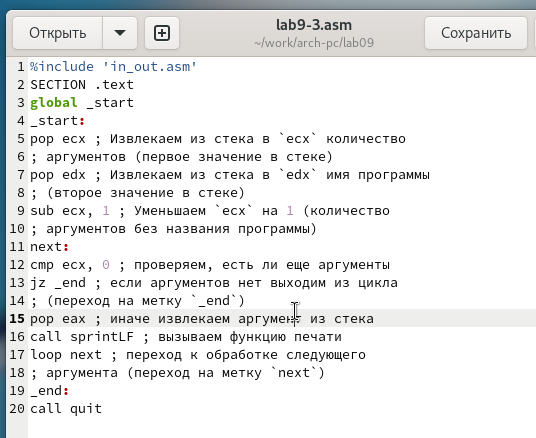


Рис. 15: Программа в файле lab9-3.asm

Для начала установила точку останова перед первой инструкцией в программе и запустила её.

Адрес вершины стека хранится в регистре esp, и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 — это имя программы lab9-3 и непосредственно аргументы: аргумент1, аргумент2 и аргумент 3.

Посмотрела остальные позиции стека — по адресу [esp+4] располагается адрес в памяти, где находится имя программы, по адресу [esp+8] — адрес первого аргумента, по адресу [esp+12] — второго и т.д.

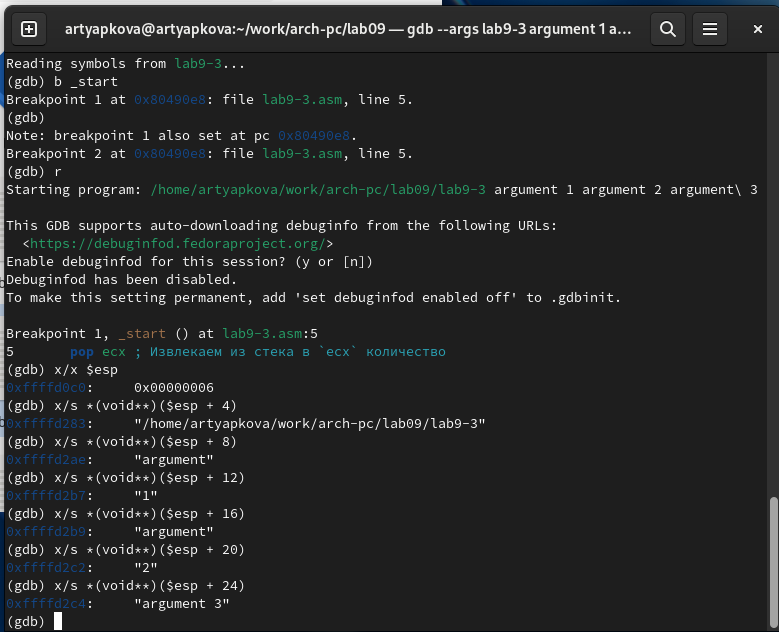


Рис. 16: Вывод значения регистра

Объяснила, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) — шаг равен размеру переменной (4 байта).

## 2.3 Задание для самостоятельной работы

Я переписала программу из лабораторной работы №8, чтобы вычислить значение функции ( f(x) ) в виде подпрограммы.

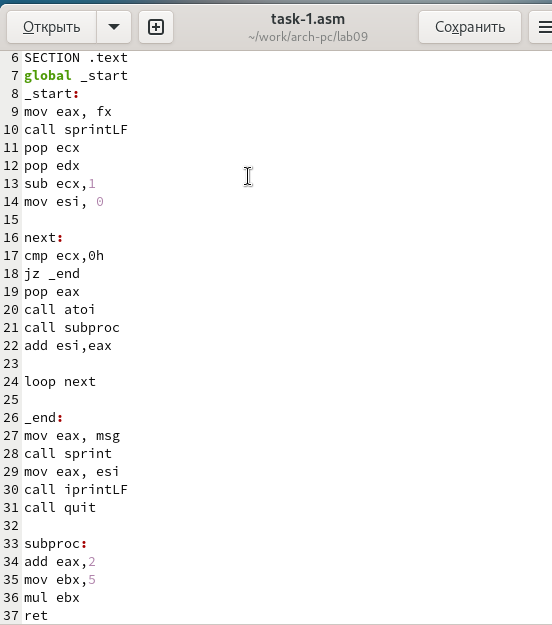


Рис. 17: Программа в файле task-1.asm

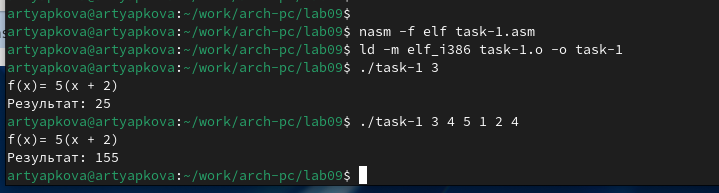


Рис. 18: Запуск программы task-1.asm

Приведенный ниже листинг программы вычисляет выражение ( (3+2)\*4+5 ). Однако при запуске программа дает неверный результат. Я проверила это и решила использовать отладчик GDB для анализа изменений значений регистров и определения ошибки.

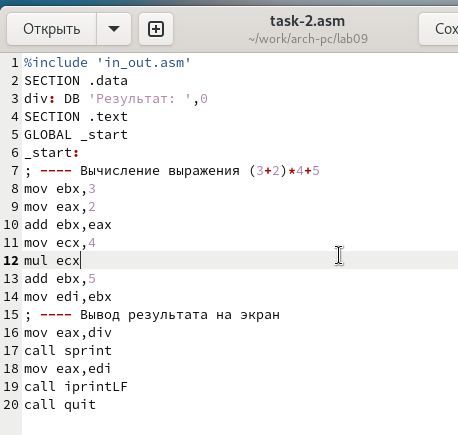


Рис. 19: Код с ошибкой в файле task-2.asm

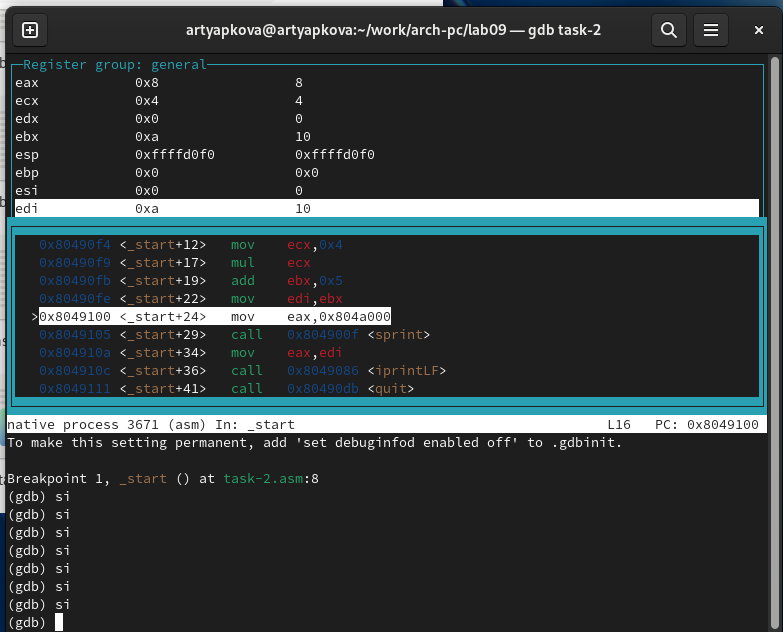


Рис. 20: Отладка task-2.asm

Я заметила, что порядок аргументов в инструкции add был перепутан, и что при завершении работы вместо eax значение отправлялось в edi. Вот исправленный код программы:

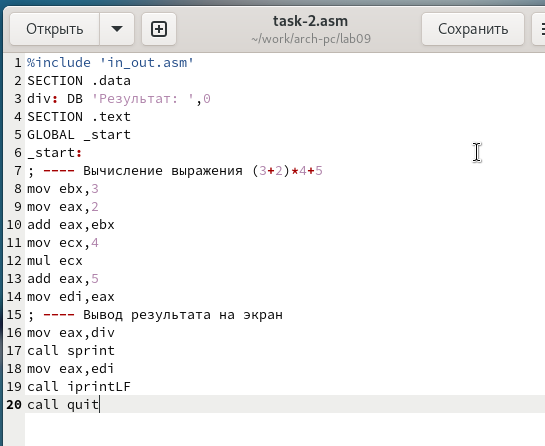


Рис. 21: Код исправлен в файле task-2.asm

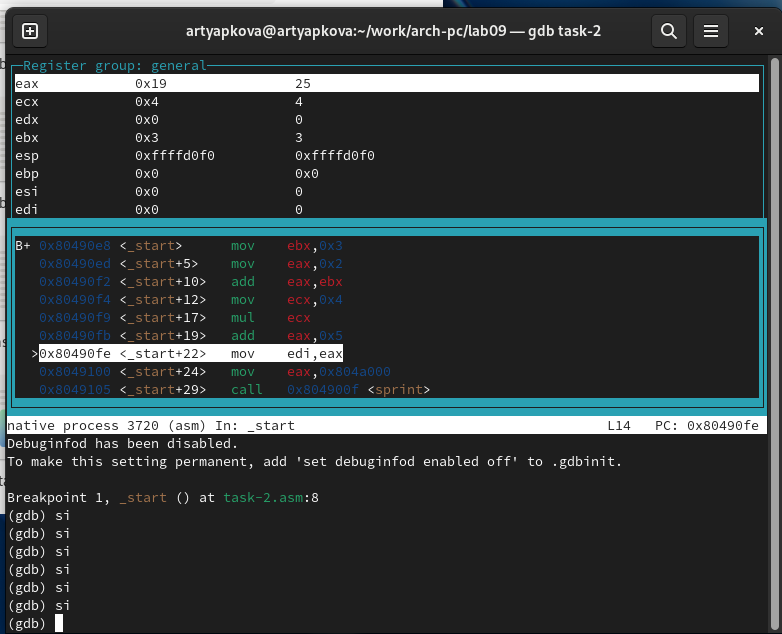


Рис. 22: Проверка работы task-2.asm

# 3 Выводы

Освоили работy с подпрограммами и отладчиком.