Отчёта по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB.

Газизянов Владислав Альбертович

Содержание

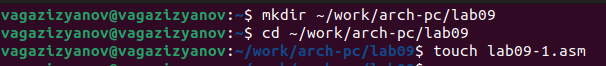
# 1 Цель работы

Познакомиться с методами отладки при помощи GDB, его возможностями.

# 2 Выполнение лабораторной работы

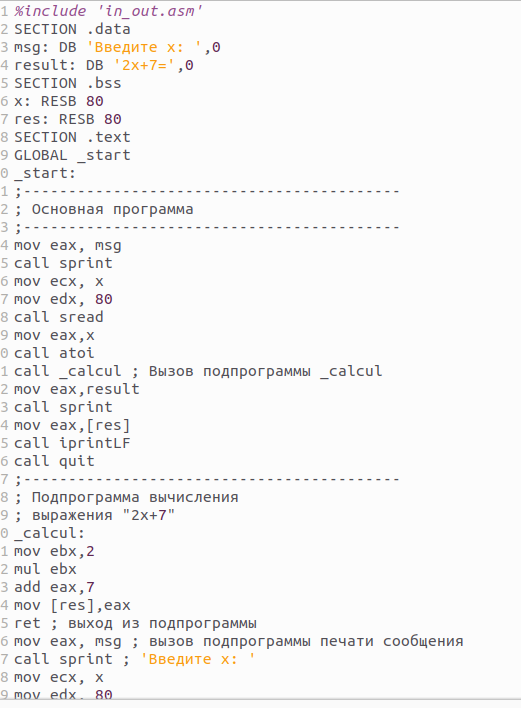
## 2.1 Реализация подпрограмм в NASM

Создайте каталог для выполнения лабораторной работы № 9, перейдите в него и создайте файл lab09-1.asm:

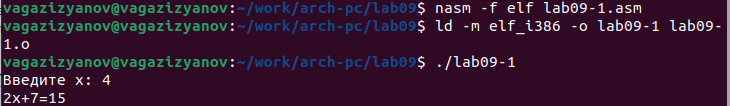


Создаём каталог

В качестве примера рассмотрим программу вычисления арифметического выражения f(x) = 2x + 7 с помощью подпрограммы \_calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Внимательно изучите текст программы (Листинг 9.1). Введите в файл lab09-1.asm текст программы из листинга 9.1. Создайте исполняемый файл и проверьте его работу.

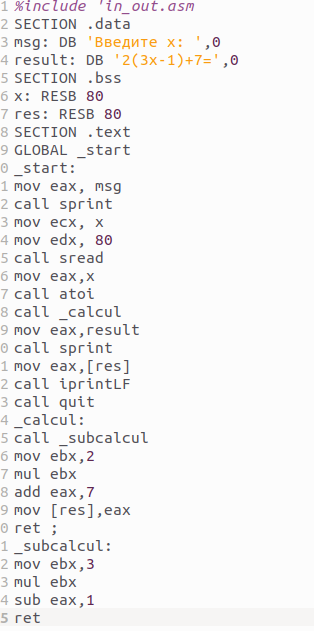


Заполняем файл



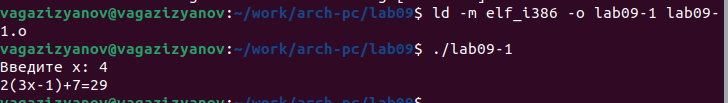
Проверяяем работу

Измените 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.



Изменим текст программы

Выодим результат.



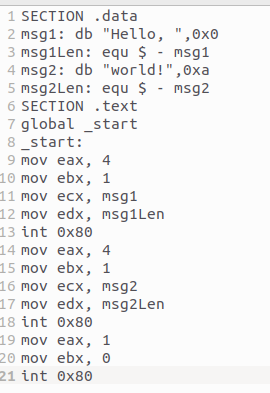
Результат

## 2.2 Отладка программам с помощью GDB

Создайте файл lab09-2.asm с текстом программы из Листинга 9.2.

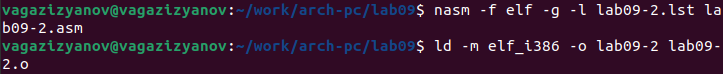
Создание файла

Создание файла



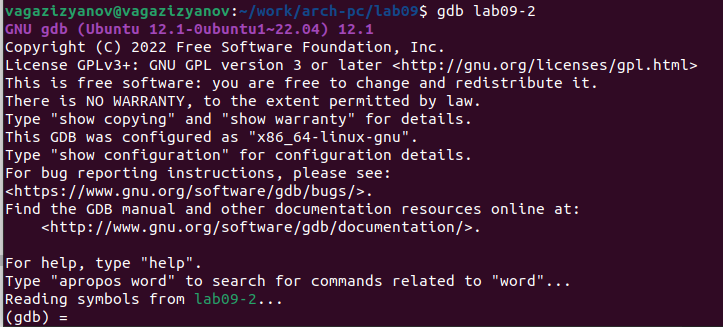
Заполнение

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’.



ключ -g

Загрузите исполняемый файл в отладчик gdb



Загрузка в gdb

Проверьте работу программы, запустив ее в оболочке GDB с помощью команды run

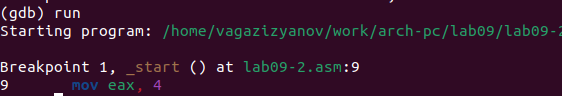
Проверка работы

Проверка работы

Для более подробного анализа программы установите брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустите её.

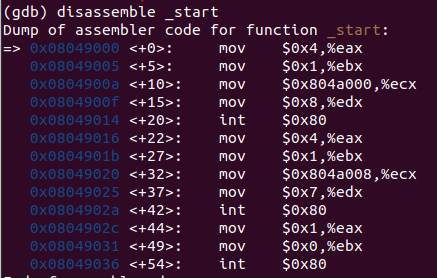
брейкпоинт

брейкпоинт



Запуск

Посмотрите дисассимилированный код программы с помощью команды disassemble начиная с метки \_start

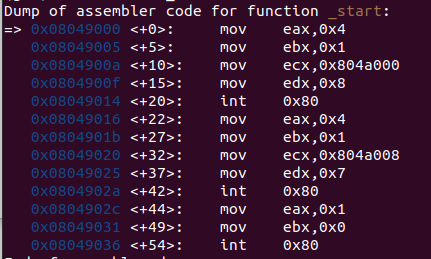


Просмотрим код

Переключитесь на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel

переход на Intel

переход на Intel



проверка работы

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

1.Порядок операндов: В ATT синтаксисе порядок операндов обратный, сначала указывается исходный операнд, а затем - результирующий операнд. В Intel синтаксисе порядок обычно прямой, результирующий операнд указывается первым, а исходный - вторым.

2.Разделители: В ATT синтаксисе разделители операндов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).

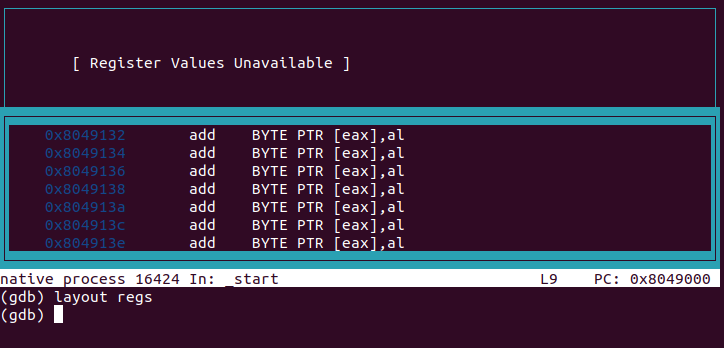
3.Префиксы размера операндов: В ATT синтаксисе размер операнда указывается перед операндом с использованием префиксов, таких как “b” (byte), “w” (word), “l” (long) и “q” (quadword). В Intel синтаксисе размер операнда указывается после операнда с использованием суффиксов, таких как “b”, “w”, “d” и “q”.

4.Знак операндов: В ATT синтаксисе операнды с позитивными значениями предваряются символом “$". В Intel синтаксисе операнды с позитивными значениями могут быть указаны без символа "$”.

5.Обозначение адресов: В ATT синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.

6.Обозначение регистров: В ATT синтаксисе обозначение регистра начинается с символа “%”. В Intel синтаксисе обозначение регистра может начинаться с символа “R” или “E” (например, “%eax” или “RAX”).

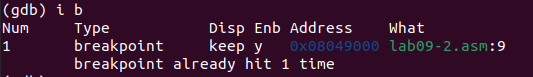
Включите режим псевдографики для более удобного анализа программы



псевдографика

## 2.3 Добавление точек останова

Проверяем точки останова с помощью команды info breakpoints



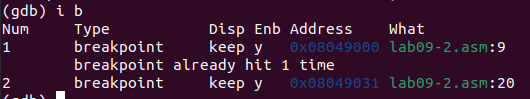
info breakpoints

Установим еще одну точку останова по адресу инструкции

точка останова

точка останова

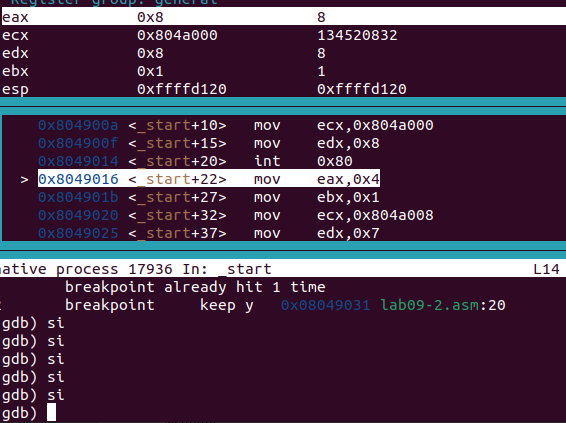
Посмотрите информацию о всех установленных точках останова



информация о точках

## 2.4 Работа с данными программы в GDB

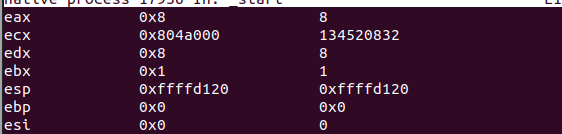
Выполните 5 инструкций с помощью команды stepi (или si) и проследите за изменением значений регистров.



инструкции

Во время выполнения команд менялись регистры: ebx, ecx, edx,eax, eip.

Посмотреть содержимое регистров также можно с помощью команды info registers



содержание регистров

Посмотрите значение переменной msg1 по имени

msg1

msg1

Посмотрите значение переменной msg2 по адресу

msg2

msg2

Измените первый символ переменной msg1

меняем символ

меняем символ

Замените любой символ во второй переменной msg2

меняеем символ

меняеем символ

Выведете в различных форматах значение регистра edx.

значения

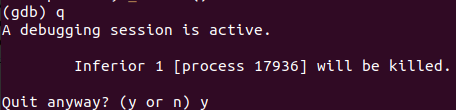
значения

значения

значения

Выводится разные значения, так как команда без кавычек присваивает регистру вводимое значение.

Завершите выполнение программы с помощью команды continue



завершение

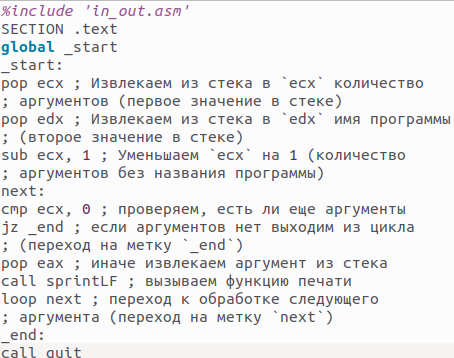
## 2.5 Обработка аргументов командной строки в GDB

Создаём файл.

создание файла

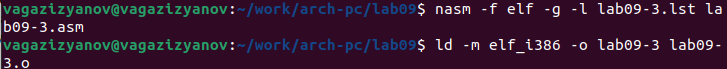
создание файла

Скопируйте файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm



заполнение файла

Создайте исполняемый файл



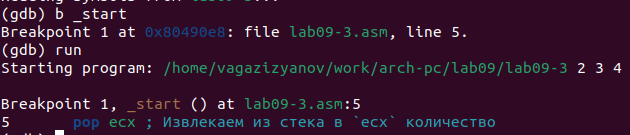
исполняеемый файл

Загрузите исполняемый файл в отладчик, указав аргументы:



загрузка в отладчик

установим точку останова перед первой инструкцией в программе и запустим ее.

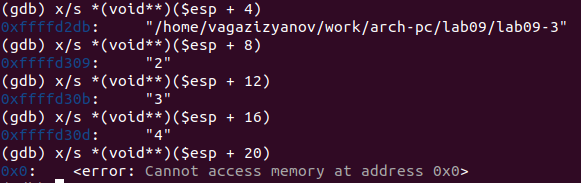


точка останова

Посмотрите остальные позиции стека

позиции стека

позиции стека



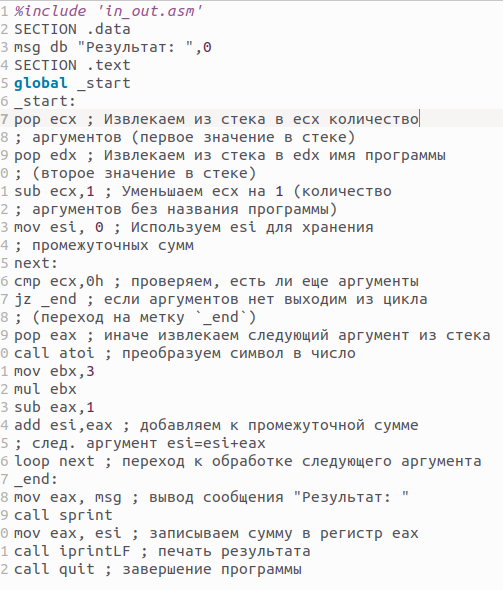
позиции стека

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

## 2.6 Задание для самостоятельной работы

### 2.6.1 Задание 1

Копируем файл lab8-4.asm(ср №1 в ЛБ8) в файл с именем lab09-4.asm



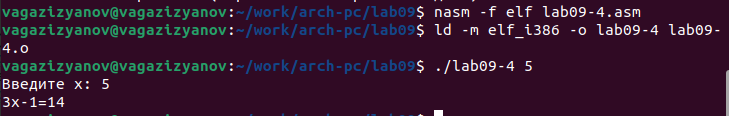
Копируем

Открываем файл и меняем его, создавая подпрограмму



Редатируем

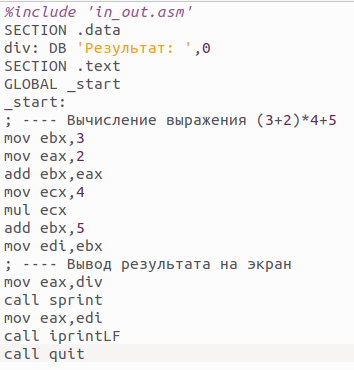
Создаем исполняемый файл и запускаем его



Запуск

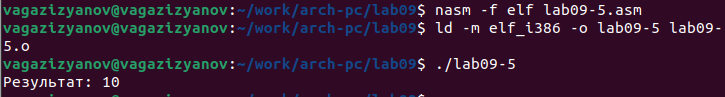
### 2.6.2 Задание 2

Создаём файл и всталвяем текст из листинг 9.3



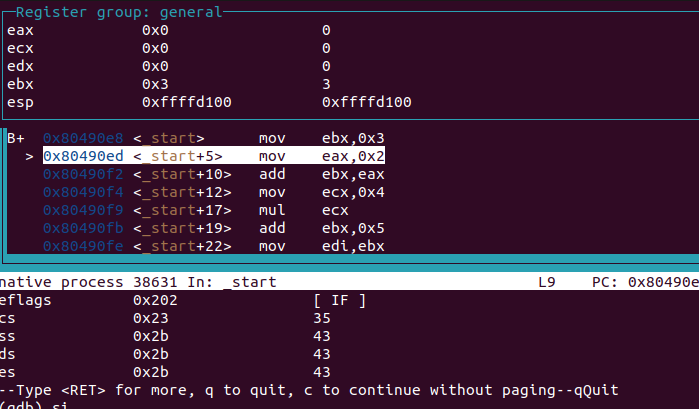
Новый файл

Создаем исполняемый файл и запускаем его



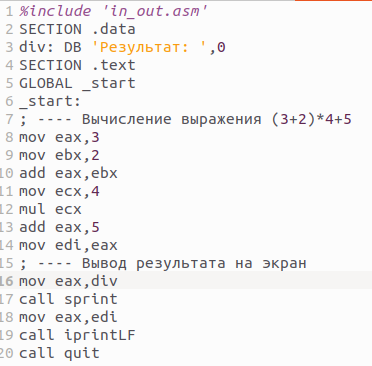
Запуск

Создаем исполняемый файл и запускаем его в отладчике GDB и смотрим на изменение регистров командой si



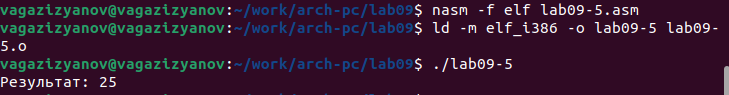
Смотрим изменения

Изменяем программу для корректной работы



корректируем

Создаем исполняемый файл и запускаем его



проверка работы

# 3 Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.