Отчёт по лабораторной работе №10

Компьютерные науки и технология программирования

Сячинова Ксения Ивановна

Содержание

# 1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. познакомиться с методами отладки при помощи GDB и его основными возможностями.

# 2 Выполнение лабораторной работы

1. Создаём каталог для выполнения лабораторной работы. Переходим в него и создаём файл для первого листинга.(рис. 1)

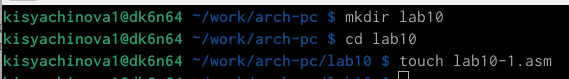


Рис. 1: Создание необходимых файлов

1. Рассмотрим программу вычисления арифмитических выражения с помощью подпрограммы ’\_calcul’. (рис. 2), (рис. 3)

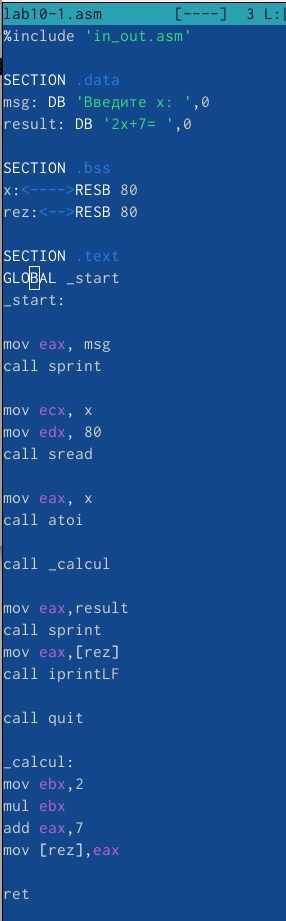


Рис. 2: Текст программы

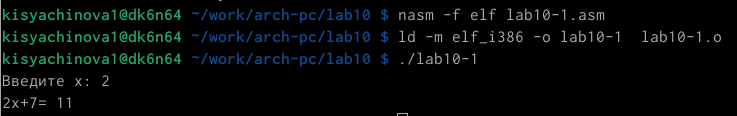


Рис. 3: Результат программы

1. Изменим программу, при этом добавим подпрограмму g(x)=3x-1. (рис. 4), (рис. 5), (рис. 6)

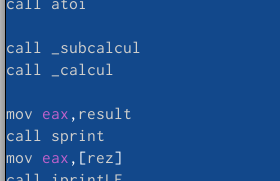


Рис. 4: Изменение программы

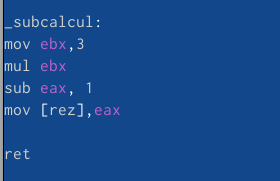


Рис. 5: Изменение программы

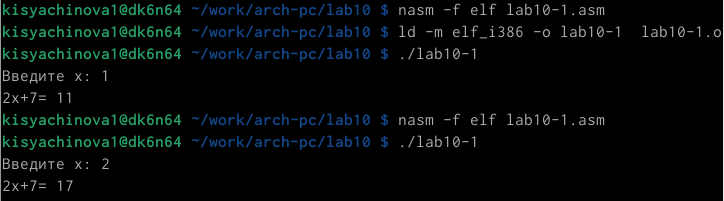


Рис. 6: Результат

Программа работает корректно. При х=1 имеем, что g(x)=2, а f(x)=11.

1. Создаём файл для второго листинга. Напишем программу печати сообщения Hello world!. (рис. 7), (рис. 8)

Рис. 7: Создание файла

Рис. 7: Создание файла

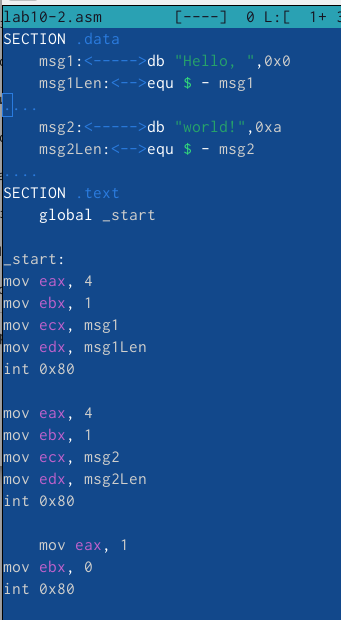


Рис. 8: Текст программы

Затем получаем исполняемый файл. Для работы с GDB нужно добавить отладочную информация, для этого трансляцию программы необходимо проводить с ключом “-g”. Загружаем исполняемый файл в отладчик. (рис. 9)

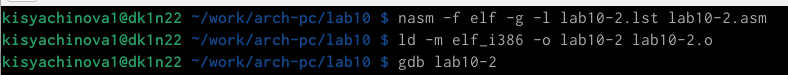


Рис. 9: Компиляция программы

Проверим работу программы, запускаем её в оболочке GDB с помощью комнады ‘run’. (рис. 10)

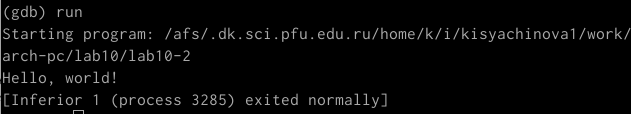


Рис. 10: Проверка работы программы

Затем установим брейкпоинт на метку \_start, с которой начинается выполнение программы и запусим её. (рис. 11)

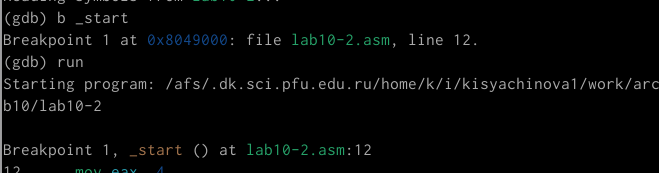


Рис. 11: Установка брейкпоинта

Далее просмотрим дисассимблированный код программы с помощью команды disassemble начиная с метки \_start. (рис. 12)



Рис. 12: Дисассимбилированный код программы

Переключимся на отображение команд с Intel’овским синтаксисом с помощью команды set disassembly-flavor intel. (рис. 13)



Рис. 13: Переключение на другой синтаксис

Для более удобного анализа программы включаем режим псевдографики.(рис. 14)

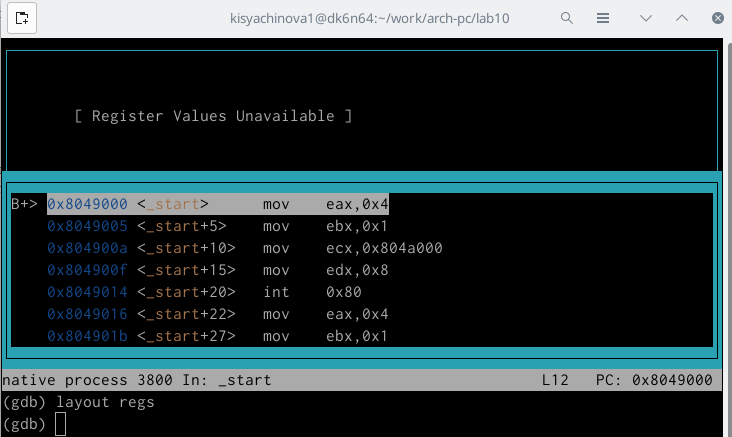


Рис. 14: Режим псевдографики

В этом режиме имеем три окна: - В верхней части названия регистров и их текущее значения - В среденй части виден результат дисассимилирования программы - Нижняя часть доступная для ввода команд

Основные различия синтаксиса машинных команд в режимах ATT и Intel заключаются в том, что в синтаксисе Intel не используются символы $ и %, а так же операнды меняются местами.

1. Для проверки точек останова используем команду ‘info breakpoints’. (рис. 15)

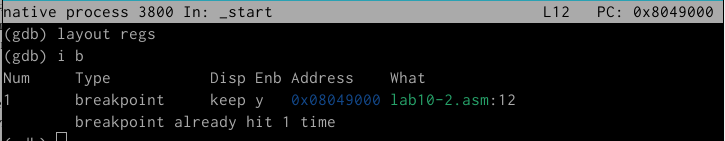


Рис. 15: Проверка точек

Установим ещё одну точку останова по адресу интрукции (ebx, 0x0). (рис. 16)

Рис. 16: Новая точка останова

Рис. 16: Новая точка останова

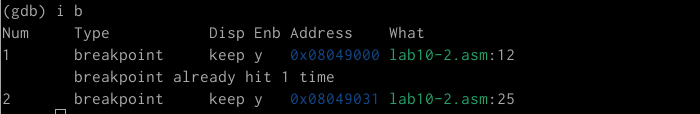


Рис. 17: Точки останова

1. Выполним 5 инструкций с помощью команды “stepi” и проследим значение регистров.(рис. 18). Как мы можем заметить, изменяются значения регистров eax, ebx, ecx, edx.

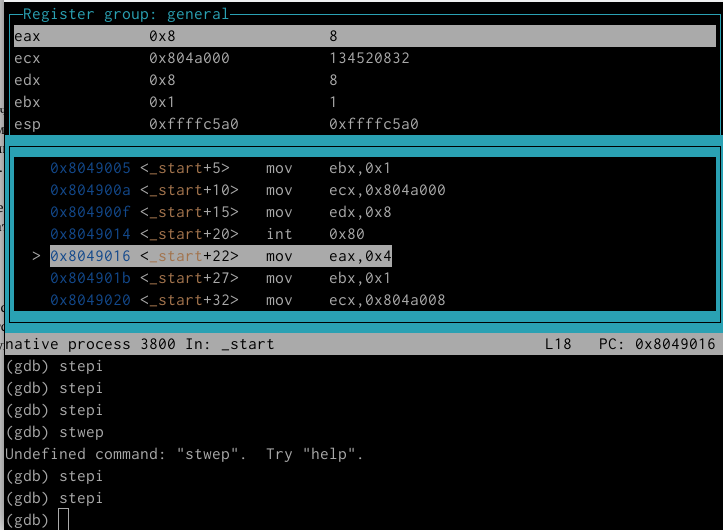


Рис. 18: Инструкция stepi

Команда ‘info registers’ (i r) позволяет посмтреть содержимое регистров. (рис. 19)

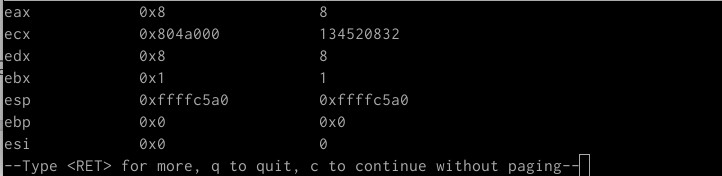


Рис. 19: Просмотр регистров

Команду ‘x ’ можно использовать для отображения памяти. С помощью ‘x &’ так же можно это сделать.(рис. 20)

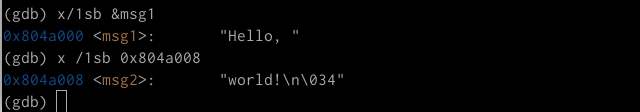


Рис. 20: Содержимое памяти

Далее воспользуемся комнадой ‘sПриобрести навыки написания программ с использованием подпрограмм. познакомиться с методами отладки при помощи GDB и его основными возможностями.et’, которая помогает изменить значение для регистра и ячейки. (рис. 21), (рис. 22)

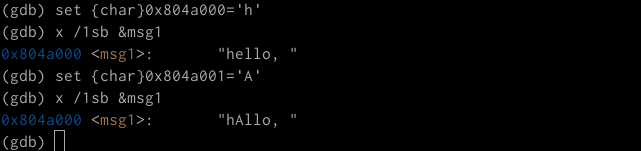


Рис. 21: Изменение значения регистров

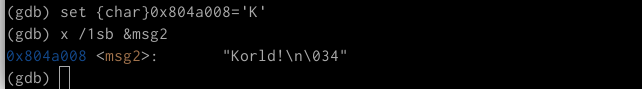


Рис. 22: Изменение значения регистров

Для просмотра значений используем команду ‘print /F ’. (рис. 23)

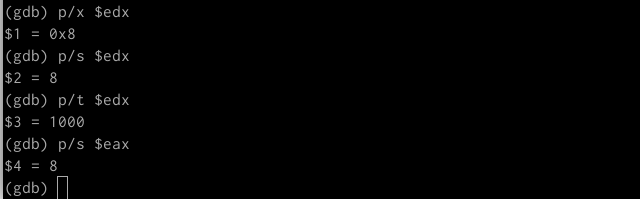


Рис. 23: Просмотр значений

С помощью команды ‘set’ можем изменить значение регистра. Изменим значения регистра ‘ebx’. (рис. 24)

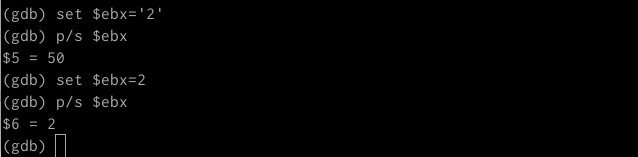


Рис. 24: Изменение регистра

Основное различие в том, что код символа ‘2’ - 110010, а это есть число 50.

Завершим выполнение программы с помощью команды ‘continue’ и выйдем из отдладчика с помощью команды ‘q’. (рис. 25)

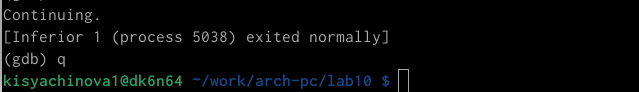


Рис. 25: Завершение программы

1. Копируем файл из лабораторной работы №9 с новым именем.(рис. 26)

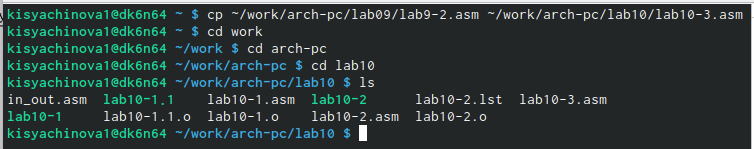


Рис. 26: Копирование файла

Далее создаём имполняемый файл. (рис. 27)

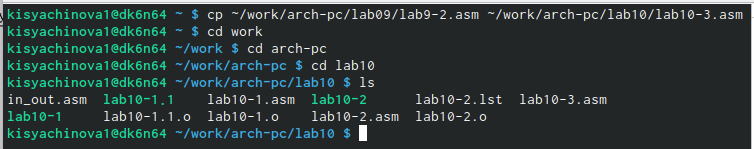


Рис. 27: Создание исполняемого файла

Для загрузки в GDB программы с аргументами необходимо использовать ключ –args.(рис. 28)

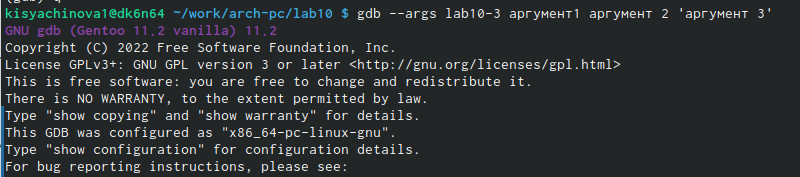


Рис. 28: Загрузка в GDB

Затем исследуем расположение аргументов командной строки. Для начала установим точку останова и запстим команду. (рис. 29)

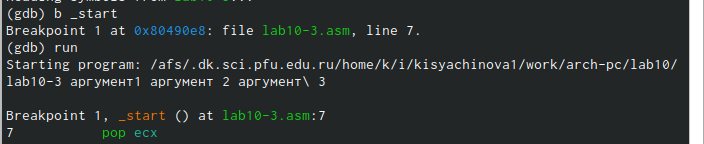


Рис. 29: Точка останова

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки. (рис. 30)

Рис. 30: Адрес вершины стека

Рис. 30: Адрес вершины стека

Посмотрим остальные позиции стека – по адесу [esp+4]. Адрес располагается в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д. (рис. 31)

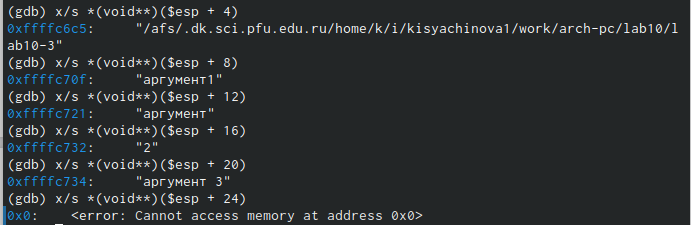


Рис. 31: Позиции стека

В данном случае шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и тд.) т.к. в теле цикла next 4 строки ввода.

#Задания для самостоятельной работы.

1. Копируем файл из 9 лабораторной работы. (рис. 32)

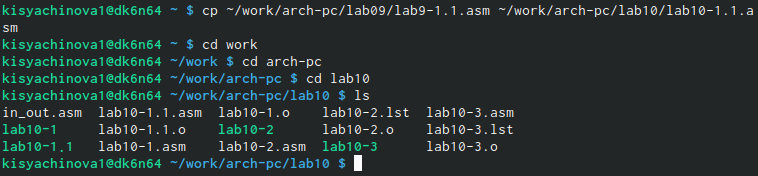


Рис. 32: Копирование файла

Изменим программу из 9 лабораторной с использованием подпрограмм и запустим её. (рис. 33)

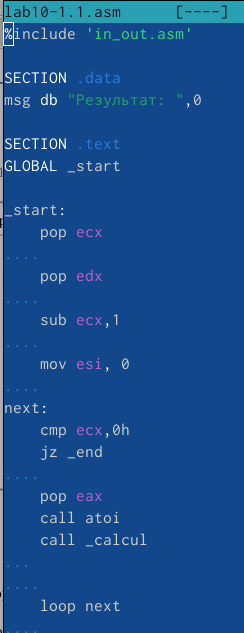


Рис. 33: Текст программы

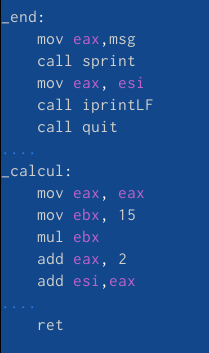


Рис. 34: Текст прогарммы

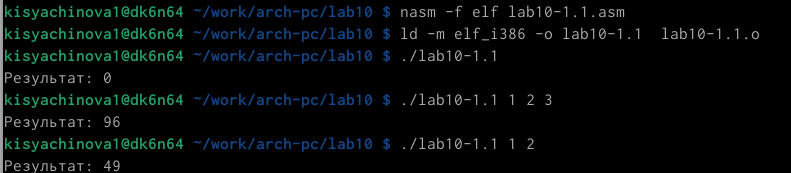


Рис. 35: Проверка файла

1. Напишем программу из листинга лабораторной работы. Проверяем работу с помощью отладчика.(рис. 36), (рис. 37)

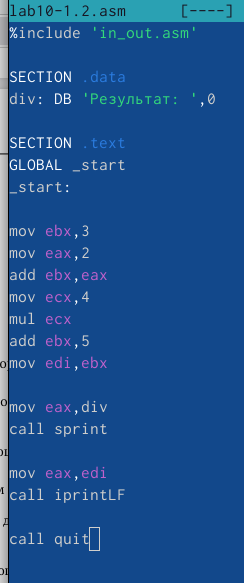


Рис. 36: Текст программы

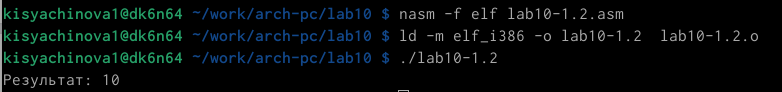


Рис. 37: Результат программы

Видим, что результат программы неверный, т.к (3+2)\*4+5=25, а не 10. Для устранения ошбики запускаем отладчик.(38)

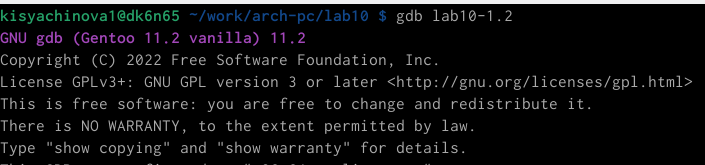


Рис. 38: GDB

Установим точку останова. (рис. 39)



Рис. 39: Точка останова

Запускаем код программы.(рис. 40)

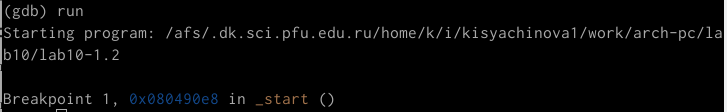


Рис. 40: Запуск программы

Включаем режим псевдографики и пошагово проходим все строчки.(рис. 41)

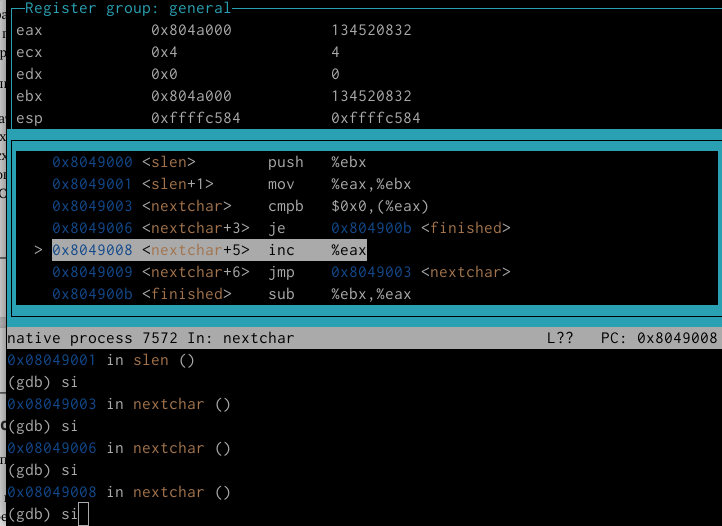


Рис. 41: Проверка программы

Как мы можем заметить, регистр eax должен умножаться на 4, но у нас умножался регистр ebx. Также, число 5 прибавлялось не к произведению, а только к ebx. Исправим ошибки. рис. 42)

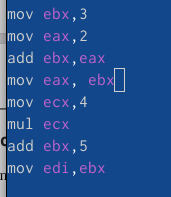


Рис. 42: Исправление ошибки

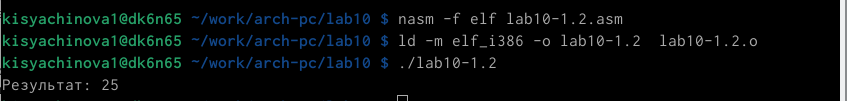


Рис. 43: Проверка

Программа работает корректно.

#Вывод

В ходе выполнения данной лабороторной работы я преобрела навыки написания программ с использованием подпрограмм, а так же познакомилась с методами отладки при помощи GDB и его основными возможностями.