Отчёт по лабораторной работе №9

Дисциплина: Архитектура Компьютера

Азарцова Вероника Валерьевна

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

1. Обнаружение ошибки;
2. Поиск её местонахождения;
3. Определение причины ошибки;
4. Исправление ошибки.

Наиболее часто применяют следующие методы отладки:

1. Создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения);
2. Использование специальных программ-отладчиков.

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом.  
Для вызова подпрограммы из основной программы используется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр eip адрес соответствующей подпрограммы, осуществляя таким образом переход. Затем начинается выполнение подпрограммы, которая, в свою очередь, также может содержать подпрограммы.

# 3 Выполнение лабораторной работы

1. Создаю каталог для выполнения лабораторной работы №9 и файл lab09-1.asm в нём (рис. 1).

Создание каталога lab09

Рис. 1: Создание каталога lab09

Ввожу текст программы из листинга в файл lab09-1.asm (рис. 2).

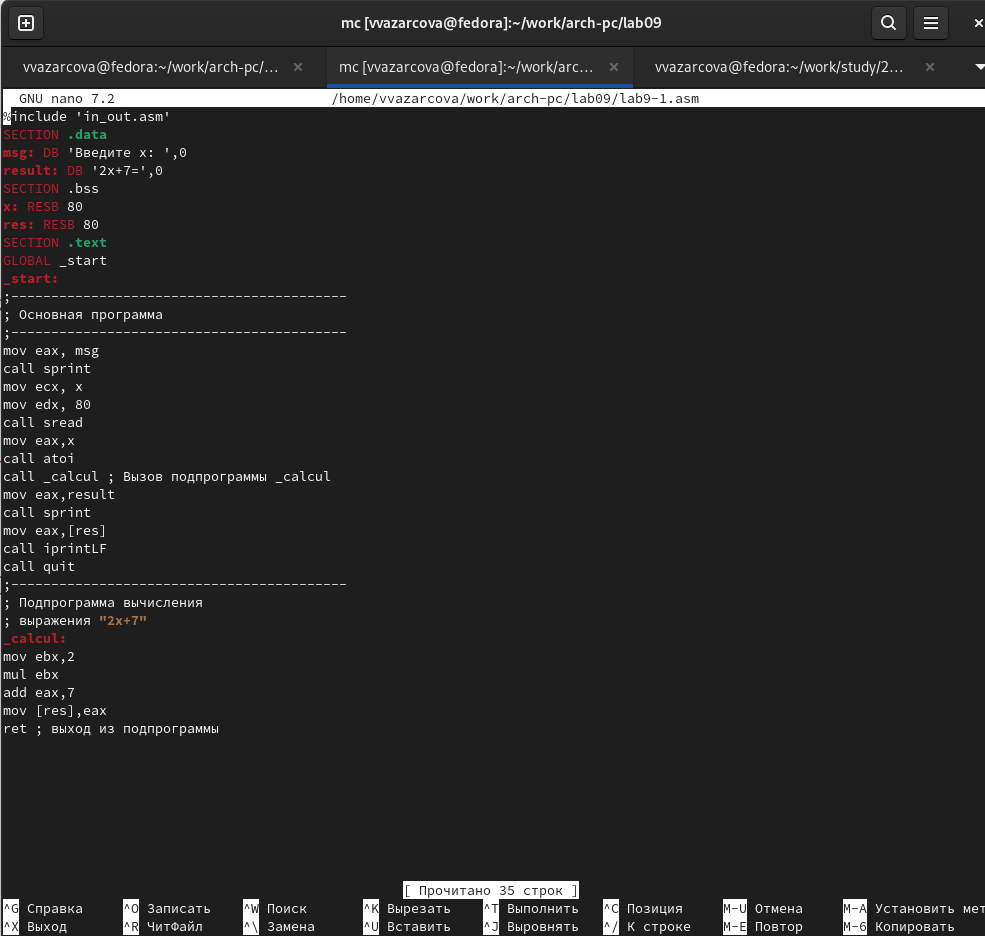


Рис. 2: lab9-1.asm

Создаю исполняемый файл и проверяю его работу (рис. 3).

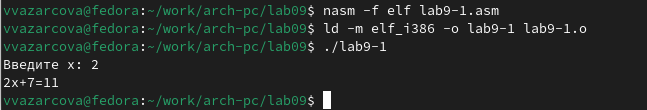


Рис. 3: Запуск lab9-1.asm

1. Создаю файл lab9-2.asm с текстом программа печати сообщения “Hello world!” (рис. 4).

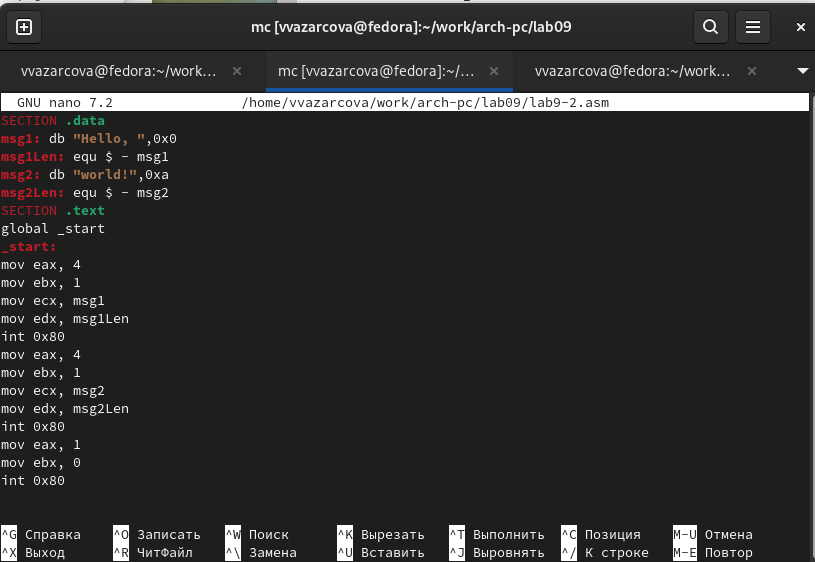


Рис. 4: lab9-2.asm

Получаю исполняемый файл. Для работы с GDB в исполняемый файл добавляю отладочную информацию, для этого трансляцию программы провожу с ключом‘-g’ и загружаю исполняемый файл в отладчик gdb (рис. 5).

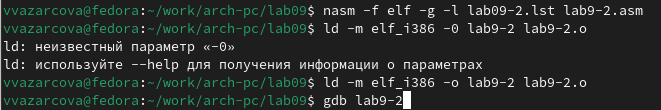


Рис. 5: Загрузка lab9-2.asm в GDB

Запускаю программу в оболочке GDB (рис. 6).

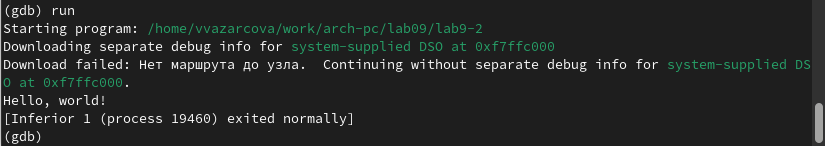


Рис. 6: Запуск в оболочке GDB

Запускаю программу с брейпоинтом на метке \_start (рис. 7).

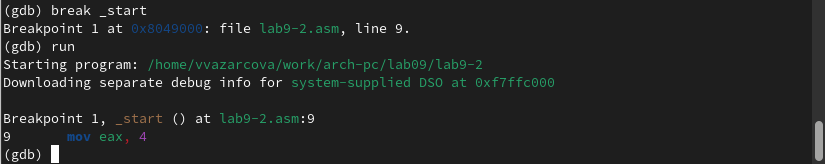


Рис. 7: Брейпоинт на метке \_start

Смотрю на дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 8).

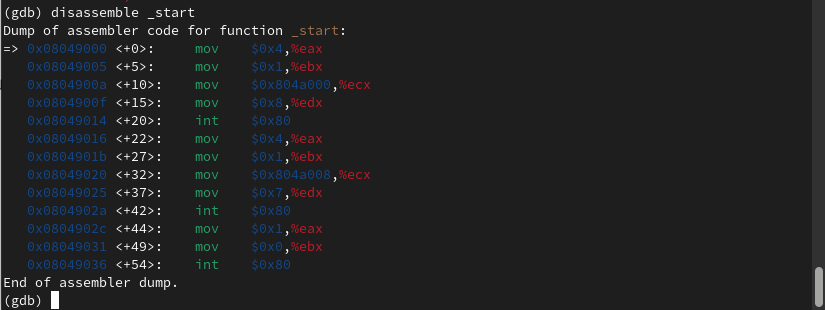


Рис. 8: Дисассимилированный код программы

Смотрю на дисассимилированный код программы с синтаксом intel с помощью команды disassemble начиная с метки \_start (рис. 9).

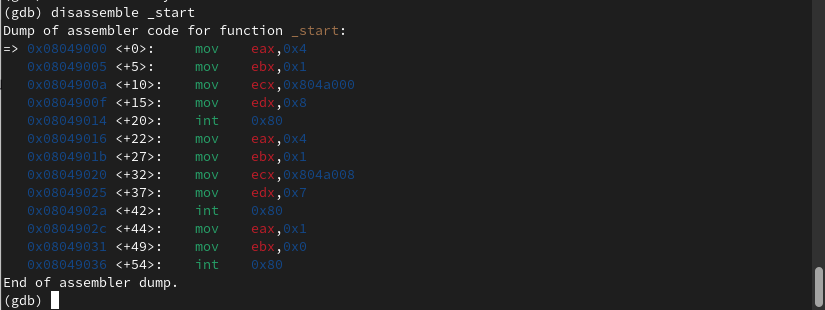


Рис. 9: Дисассимилированный код программы с синтаксом intel

Эти два отображения различаются тем, что в Intel’овском сначала пишется регистр, а потом адрес, причем без символа доллара и без символа процента, т.е. его вид упрощенный.

Включаю режим псевдографики для более удобного анализа программы (рис. 10).

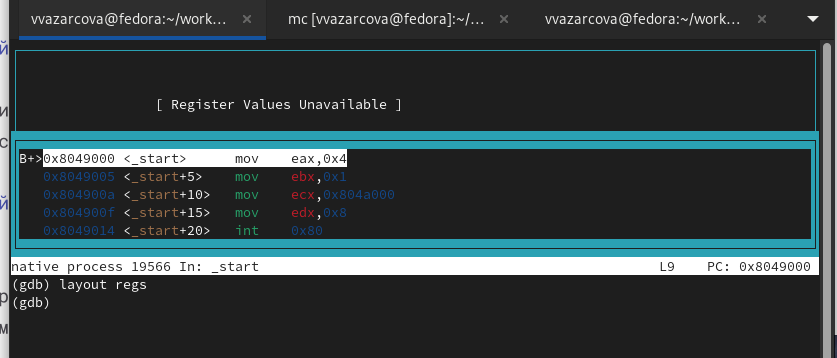


Рис. 10: Режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверяю это с помощью команды info breakpoints (кратко i b) (рис. 11).

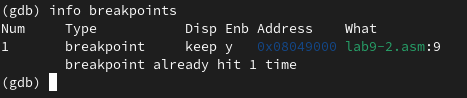


Рис. 11: Проверка меток

Устанавливаю ещё одну точку останова по адресу инструкции и проверяю (рис. 12).

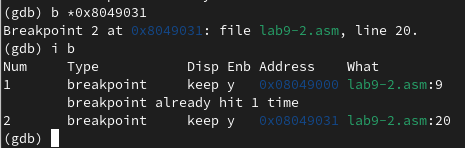


Рис. 12: Установка второй метки и проверка

Выполняю 5 инструкций с помощью команды stepi (или si) и слежу за изменением значений регистров. Значения регистров ebx, ecx, edx и eax изменялись (рис. 13).

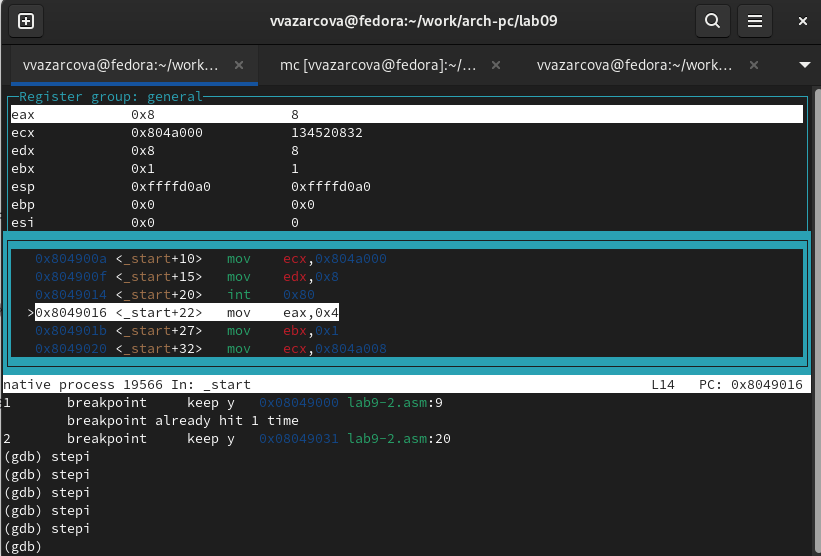


Рис. 13: 5 инструкций с помощью stepi

Смотрю значение переменной msg1 по имени (рис. 14).

Просмотр значения переменной

Рис. 14: Просмотр значения переменной

Изменю первый символ переменной msg1 и msg2 (рис. 15).

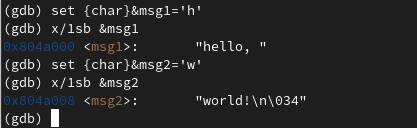


Рис. 15: Изменение символов в переменных

Выведу в разных форматах значение регистра edx (рис. 16).



Рис. 16: Значение регистра edx в разных форматах

С помощью команды set измените значение регистра ebx (рис. 17).

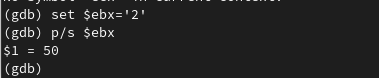


Рис. 17: Смена значения регистра ebx

Завершаю выполнение программы.

1. Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл именем lab9-3.asm, создаю исполняемый файл, и загружаю его в отладчик с ключем –args (рис. 18).

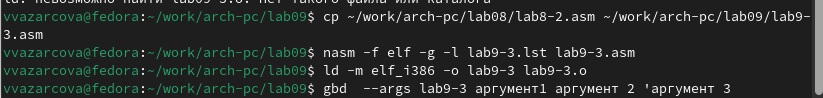


Рис. 18: Копирование, создание исполняемого файла, загрузка

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 19).

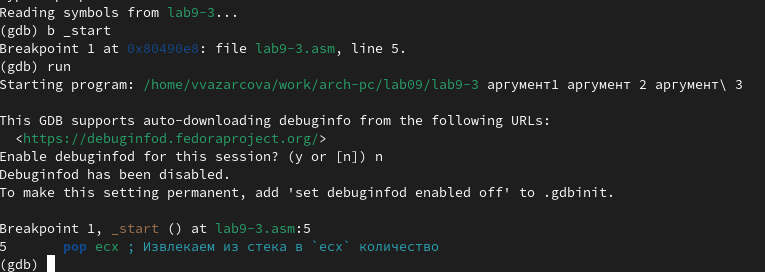


Рис. 19: Запуск с точкой останова

Посмотрю остальные позиции стека – по адесу esp+4 располагается адрес в памяти где находиться имя программы, по адесу esp+8 храниться адрес первого аргумента, по аресу esp+12 – второго и т.д (рис. 20).

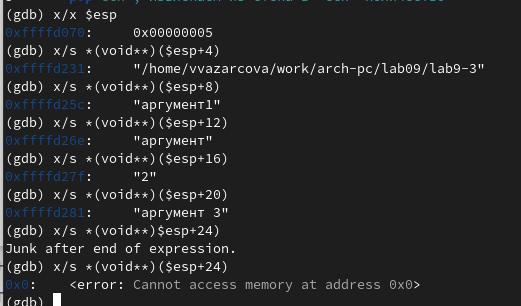


Рис. 20: Просмотр позиций стека

Шаг 4 обусловлен разрядностью системы, а указатель void занимает 4 байта.

# 4 Задания для самостоятельной работы

1. Преобразую программу из лабораторной работы 8, реализоваа вычисление функции как подпрограмму.

%include 'in\_out.asm'  
  
SECTION .data  
msg\_func db "Функция: f(x) = 10x - 4", 0  
msg\_result db "Результат: ", 0  
  
SECTION .text  
GLOBAL \_start  
  
\_start:  
mov eax, msg\_func  
call sprintLF  
  
pop ecx  
pop edx  
sub ecx, 1  
mov esi, 0  
  
next:  
cmp ecx, 0h  
jz \_end  
pop eax  
call atoi  
  
call \_calculate\_fx  
  
add esi, eax  
loop next  
  
\_end:   
mov eax, msg\_result  
call sprint  
mov eax, esi  
call iprintLF  
call quit  
  
\_calculate\_fx:  
mov ebx, 10  
mul ebx  
sub eax, 4

1. Запускаю программу в режике отладичка и ппросматриваю изменение значений регистров. При выполнении инструкции mul ecx результат записывается в eax, но также меняет edx. Значение регистра ebx не обновляется, поэтому результат программа неверно подсчитывает функцию. Исправляю это и функция работает корректно.

%include 'in\_out.asm'  
  
SECTION .data  
div: DB 'Результат: ', 0  
  
SECTION .text  
GLOBAL \_start  
\_start:  
  
mov ebx, 3  
mov eax, 2  
add ebx, eax  
mov eax, ebx  
mov ecx, 4  
mul ecx  
add eax, 5  
mov edi, eax  
  
mov eax, div  
call sprint  
mov eax, edi  
call iprintLF  
  
call quit

# 5 Выводы

Подводя итоги данной лабораторной работы, я получила навыки написания программ с использованием подпрограмм, а так же познакомилась с методами отладки при помощи GDB.

# Список литературы