Отчёта по лабораторной работе №9

Архитектура компьютера

Еремина Оксана Андреевна НКАбд-02-23

Содержание

# Цель работы

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

# Задание

1. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Обработка аргументов командной строки в GDB
4. Выполнение заданий для самостоятельной работы

# Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе.

Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия.

Точки останова — это специально отмеченные места в программе, в которых программаотладчик приостанавливает выполнение программы и ждёт команд.

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки.

# Выполнение лабораторной работы

1. Реализация подпрограмм в NASM

Создаю каталог для программам лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm. (рис.1)

Figure 1: Создание папки и файла

Figure 1: Создание папки и файла

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

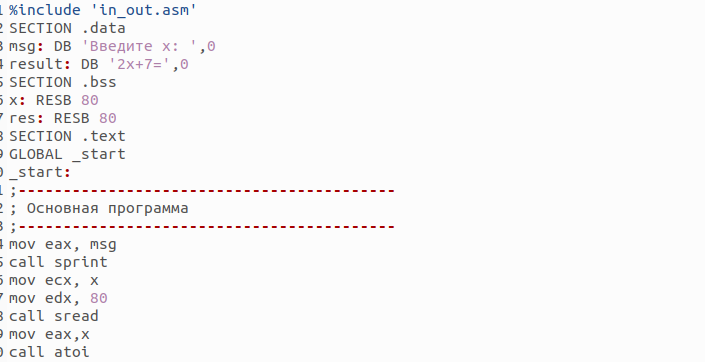


Figure 2: Содержание файла lab09-1.asm

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

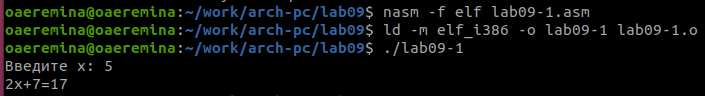


Figure 3: Выподнение программы

Изменяю код 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.4)

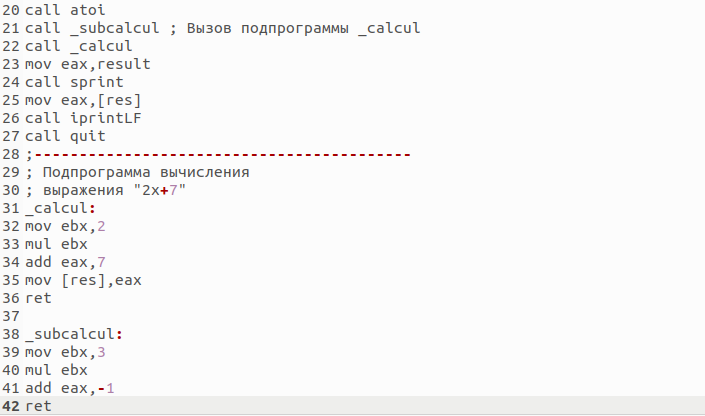


Figure 4: Изменение программы

Создаю исполняемый файл и проверяю работу программы. (рис.5)

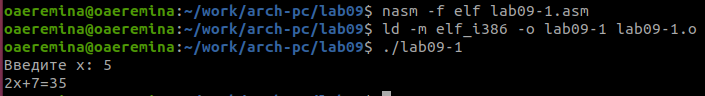


Figure 5: Выподнение программы

1. Отладка программам с помощью GDB

Создаю файл lab09-2.asm в каталоге ~/work/arch-pc/lab08. Ввожу код программы из листинга 9.2 в созданный файл. (рис.6)

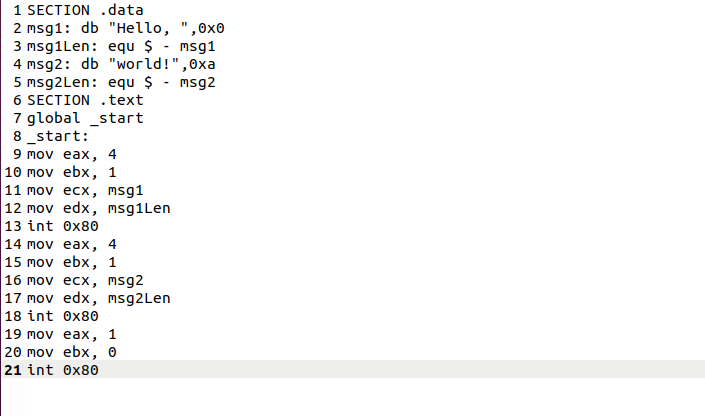


Figure 6: Содержание файла lab09-2.asm

Получаю исполняемый файл для работы с GDB с ключом ‘-g’. Загружаю исполняемый файл в отладчик gdb. Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run. (рис.7)

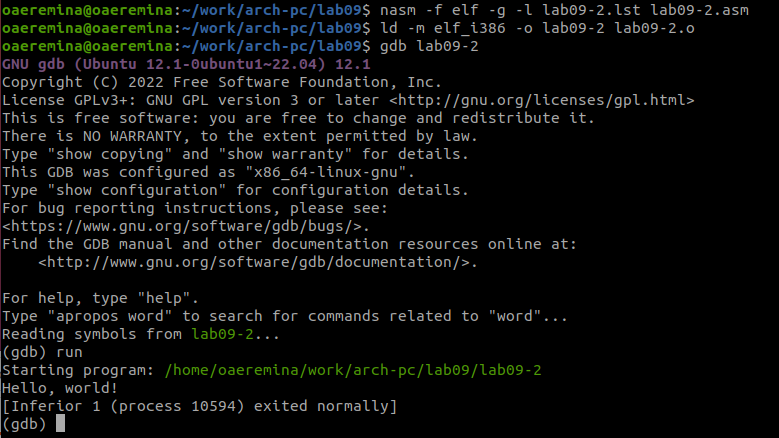


Figure 7: Выполнение программы

Для более подробного анализа программы устанавливаю брейкпоинт на метку \_start и запускаю её. (рис.8)

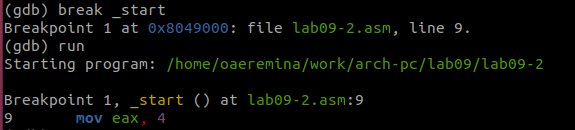


Figure 8: Анализ программы

Просматриваю дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start, и переключаюсь на отображение команд с синтаксисом Intel, введя команду set disassembly-flavor intel. (рис.9)

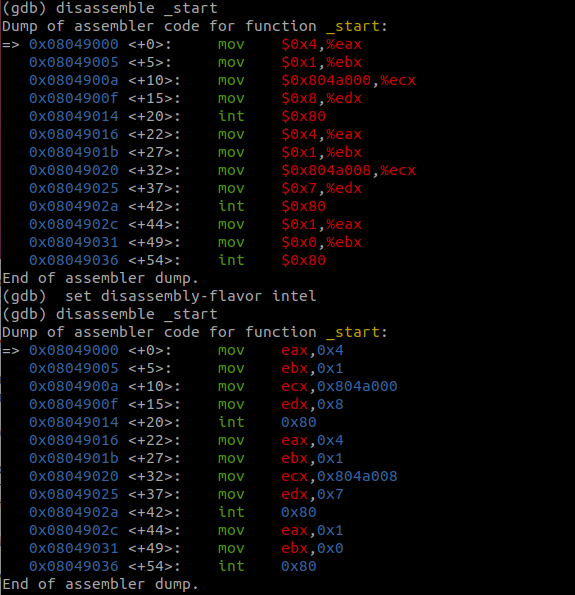


Figure 9: Дисассимилированный код программы

Включаю режим псевдографики для более удобного анализа программы с помощью команд layout asm и layout regs. (рис.10)

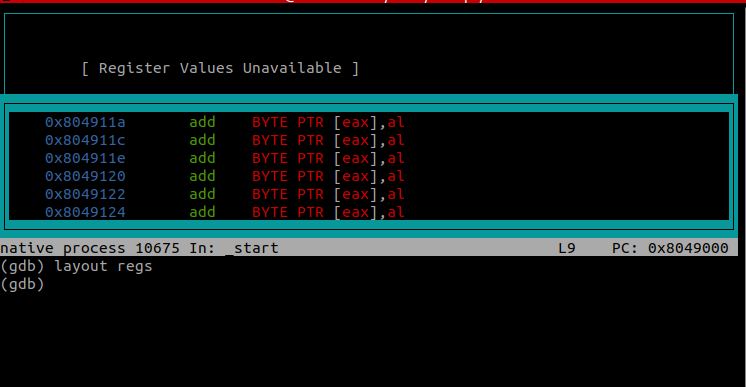


Figure 10: Режим псевдографики

Добавление точек останова

Проверяю, что точка останова по имени метки \_start установлена с помощью команды info breakpoints и устанавливаю еще одну точку останова по адресу инструкции mov ebx,0x0. Просматриваю информацию о всех установленных точках останова. (рис.11)

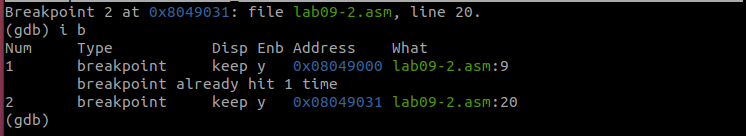
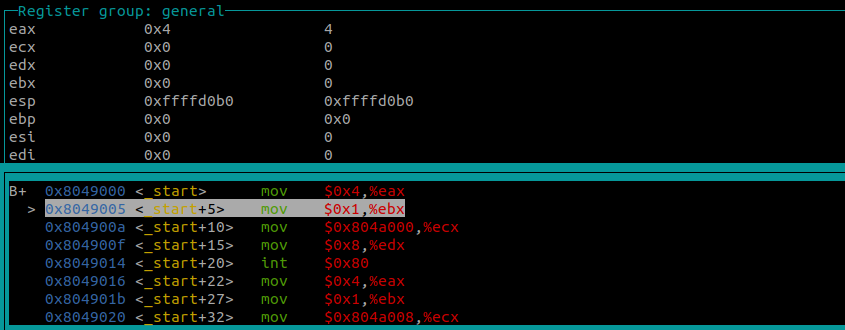
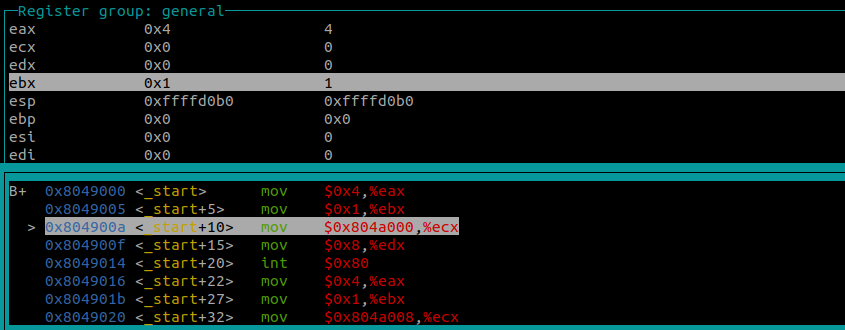
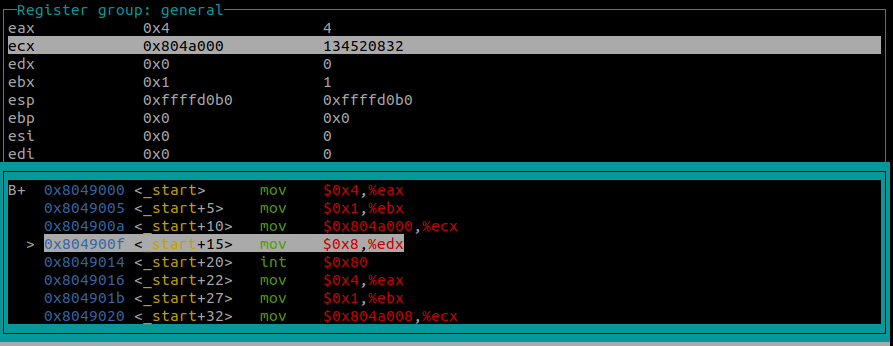
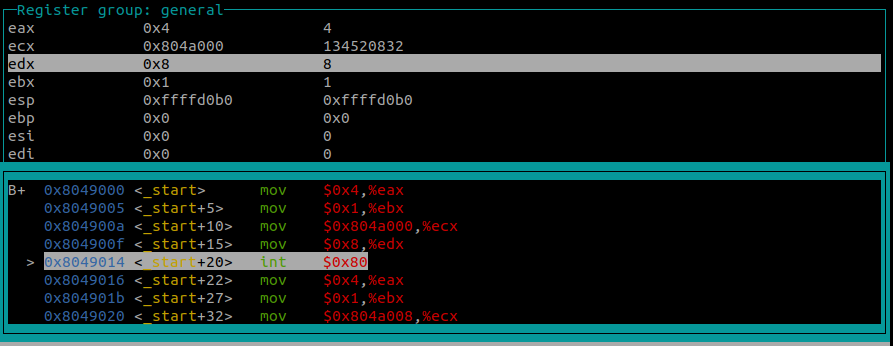
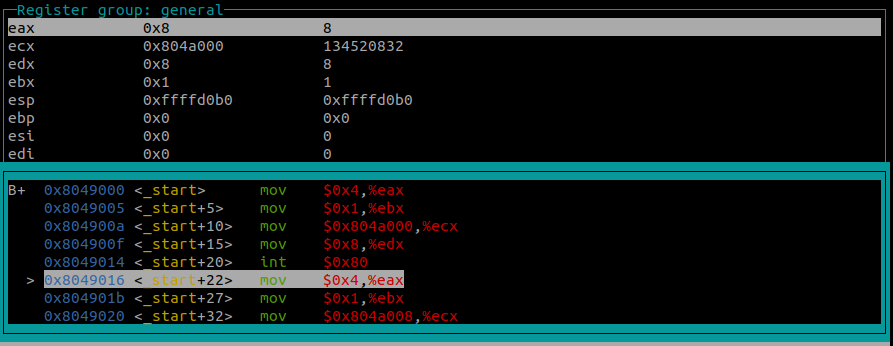


Figure 11: Добавление точек останова

Работа с данными программы в GDB

Выполните 5 инструкций с помощью команды stepi (или si). (рис.12-16)

    ]

Изменились значения регистров eax, ecx, edx и ebx.

Просматриваю значение переменной msg1 по имени с помощью команды x/1sb &msg1 и значение переменной msg2 по ее адресу.(рис.17)

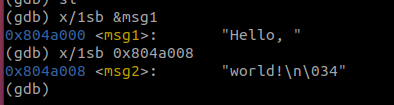


Figure 12: Значение переменных msg1 иmsg2

С помощью команды set изменяю первый символ переменной msg1 и заменяю первый символ в переменной msg2. (рис.18)

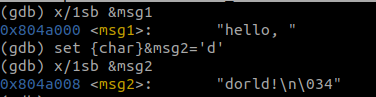


Figure 13: Замена символов переменных msg1 иmsg2

Вывожу значение регистра edx в шестнадцатеричном, двоичном формате и в символьном виде. (рис. 19)

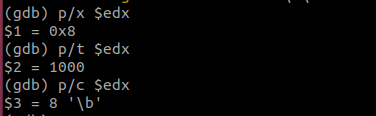


Figure 14: Значения регистра edx

С помощью команды set изменяю значение регистра ebx. (рис.20)

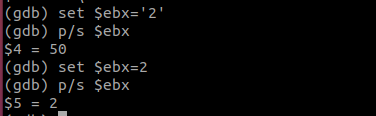


Figure 15: Изменение регистра edx

Разница вывода команд p/s $ebx отличается тем, что в первом случае мы переводим символ в его строковый вид, а во втором случае число в строковом виде не изменяется.

Выхожу из GDB с помощью команды quit.

1. Обработка аргументов командной строки в GDB

Копируйте 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. Создаю исполняемый файл. Загружаю исполняемый файл в отладчик gdb. (рис.21)

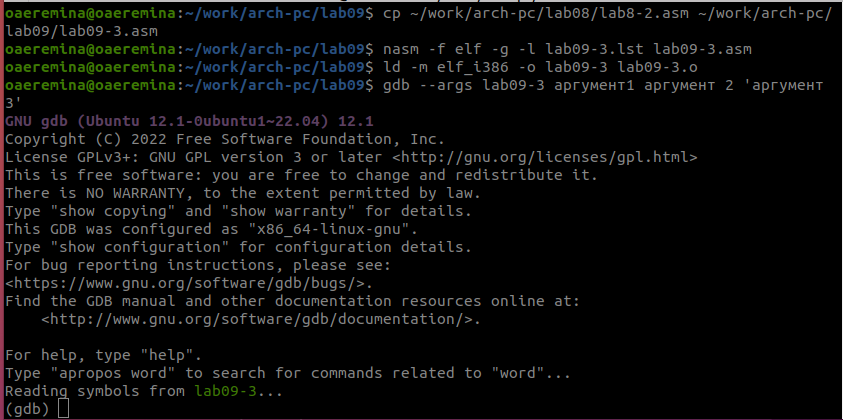


Figure 16: Действия с файлом lab8-2.asm

Устанавливаю точку останова перед первой инструкцией в программе и запускаем ее. ее. Адрес вершины стека, который храниться в регистре esp и по этому адресу располагается число равное количеству аргументов компандной строки. (рис.22)

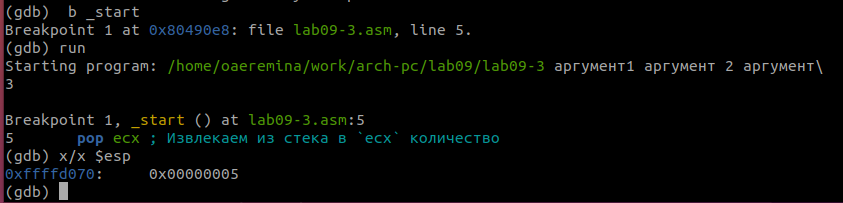


Figure 17:

Посматриваю вершину стека и позиции стека по их адресам. (рис.23)

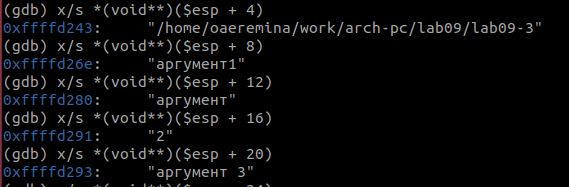


Figure 18: Вершины стека

1. Выполнение заданий для самостоятельной работы
2. Преобразовываю программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. Создаю исполнительный файл и выполняю программу. (рис.24)

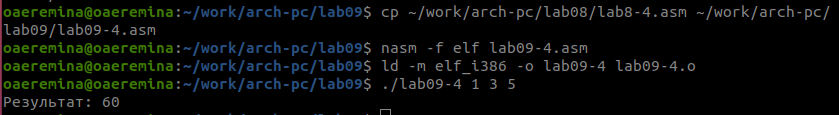


Figure 19: Измененный код программы из лабораторной работы №8

Код программы:

%include ‘in\_out.asm’

SECTION .data

msg db “Результат:”,0

SECTION .text

global \_start

\_start:

pop ecx

pop edx

sub ecx,1

mov esi, 0

mov edi,10

next:

pop eax

call atoi

sub eax,1

mul edi

add esi,eax

cmp ecx,0h

jz .done

loop next

.done:

mov eax, msg

call sprint

mov eax, esi

call iprintLF

call quit

ret

# Вывод

При выполнении данной лабораторной работы я приобрела практические навыки написания программ с использованием подпрограмм, познакомилась с методами отладки при помощи GDB и его основными возможностями.

# Список литературы

Архитектура ЭВМ