Отчет по лабораторной работе 9

Понятие подпрограммы.Отладчик GDB.

Зайцева П.Е.

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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:

• обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

• синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают пре- рывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить доволь- но трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы.

Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

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.

Если в программе встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма бу- дет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы. # Выполнение лабораторной работы Создала каталог для выполнения лабораторной работы No 9, перешла в него и создала файл lab09-1.asm

```
pezayjceva@dk8n54 ~ $ mkdir ~/work/arch-pc/lab09
pezayjceva@dk8n54 ~ $ cd ~/work/arch-pc/lab09
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $ touch lab09-1.asm
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Рис.1

Ввела в файл lab09-1.asm текст программы из листинга 9.1. Создала исполняемый файл и проверила его работу.

```
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 1
2x+7=9
pezayjceva@dk8n54 ~/work/arch-pc/lab09 $
```

Рис. 2.2: Рис.2

Создала файл lab09-2.asm с текстом программы из Листинга 9.2. Получила исполняемый файл. Для работы с GDB в исполняемый файл добавила отладочную информацию, для этого трансляцию программ провела с ключом ‘-g’. Проверила работу программы, запустив ее в оболочке GDB с помощью команды run

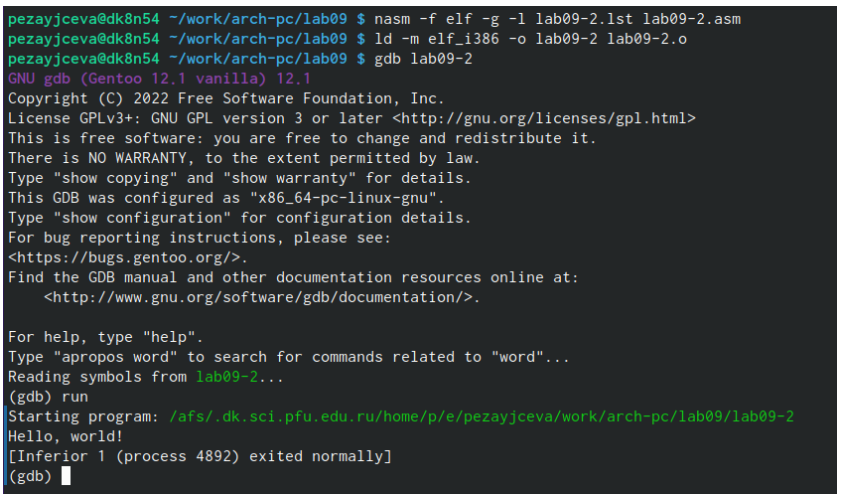

Рис. 2.3: Рис.3

Включила режим псевдографики для более удобного анализа программы. В этом режиме есть три окна:

- В верхней части видны названия регистров и их текущие значения;
- В средней части виден результат дисассимилирования программы;
- Нижняя часть доступна для ввода команд.

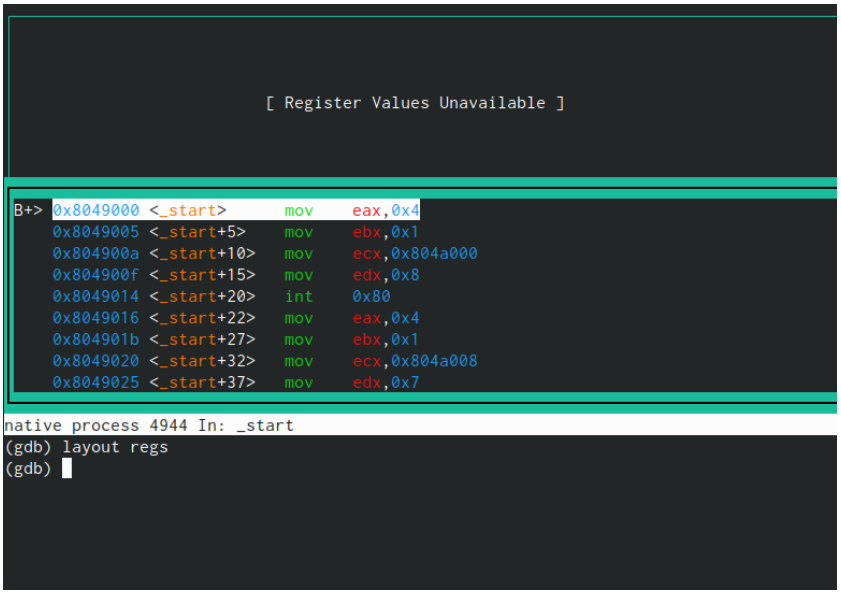

Рис. 2.4: Рис.4

На предыдущих шагах была установлена точка останова по имени метки (_start). Проверила это с помощью команды info breakpoints. Определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку останова.

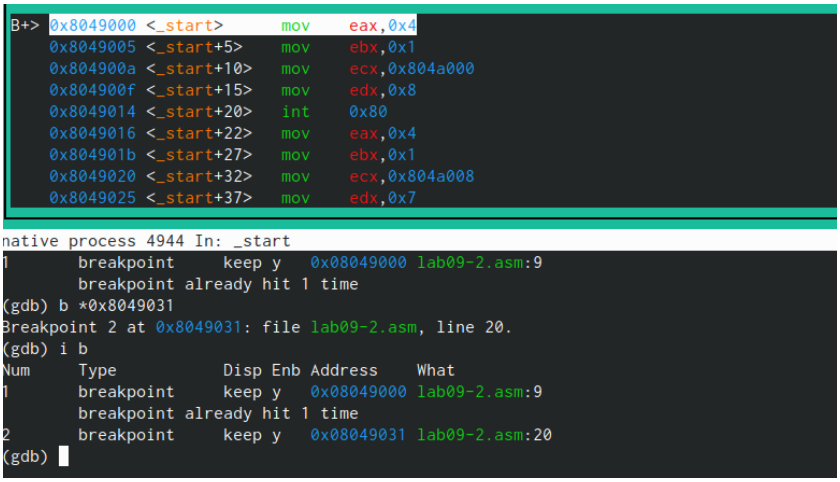

Рис. 2.5: Рис.5

Посмотрела содержимое регистров также можно с помощью команды info registers Посмотрела значение переменной msg2 по адресу.

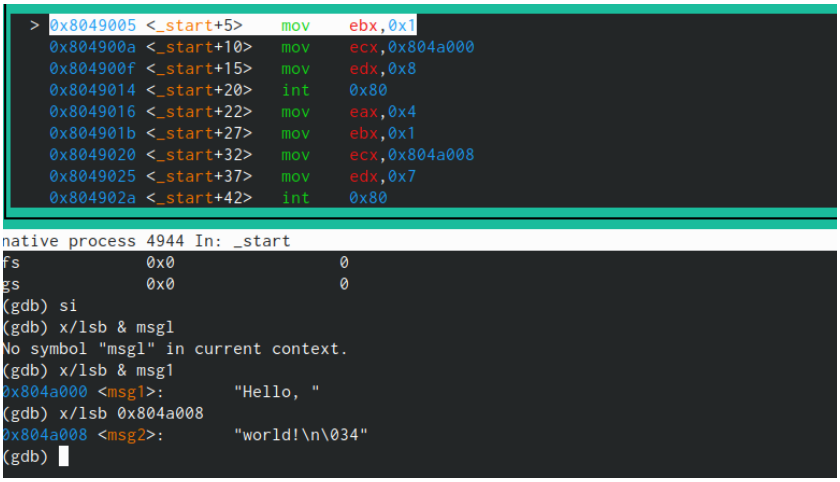

Рис. 2.6: Рис.6

С помощью команды set измените значение регистра ebx.

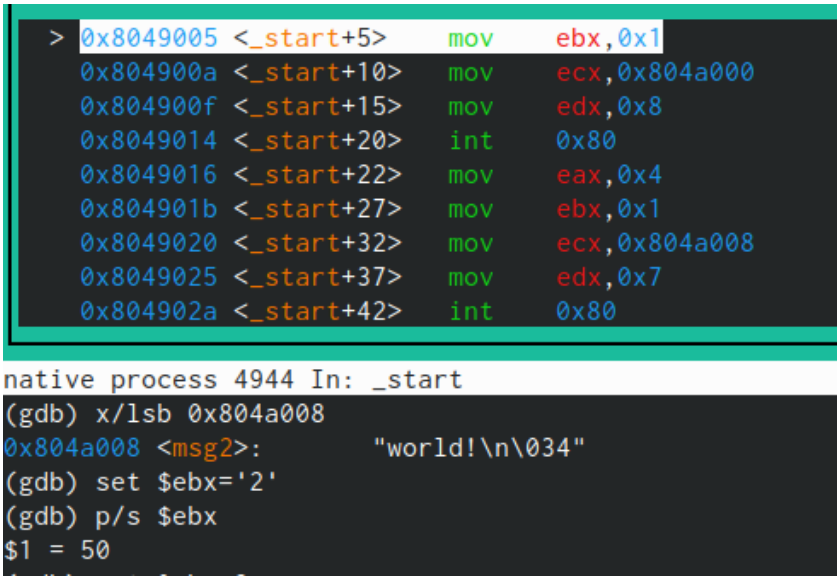

Рис. 2.7: Рис.7

Скопировала файл lab8-2.asm, созданный при выполнении лабораторной работы No8,с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. Создала исполняемый файл.

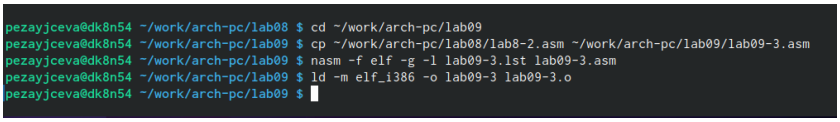

Рис. 2.8: Рис.8

Для загрузки в gdb программы с аргументами необходимо использовать ключ –args. Загрузила исполняемый файл в отладчик, указав аргументы. Для начала установила точку останова перед первой инструкцией в программе и запустила ее.

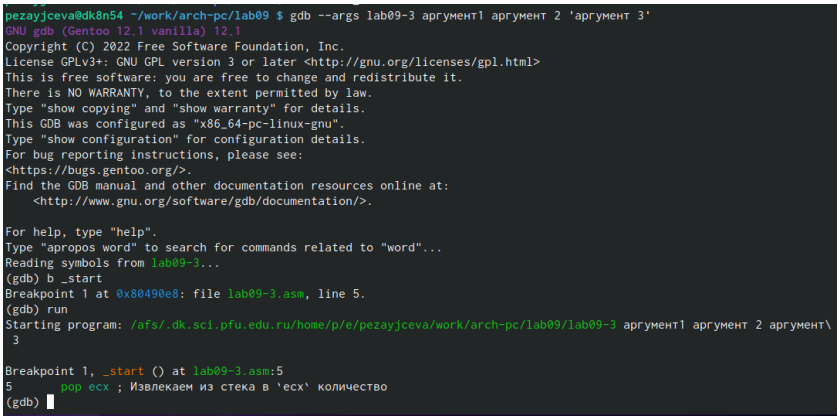

Рис. 2.9: Рис.9

Посмотрела 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.

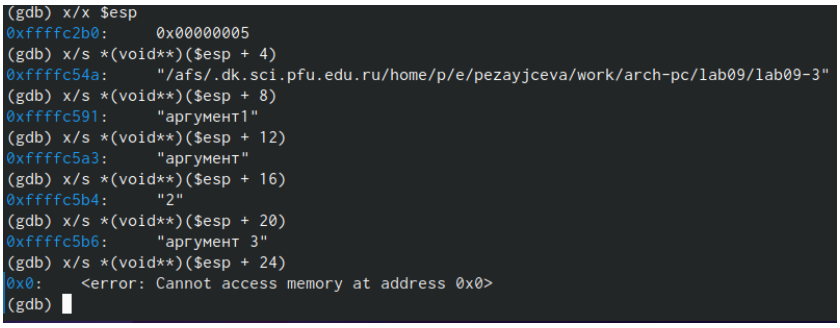

Рис. 2.10: Рис.10

Задание для самостоятельной работы.

Создала файл lab09-1.asm. Ввела программу из листинга 9.3. При запуске данная программа дает неверный результат. Проверила это.

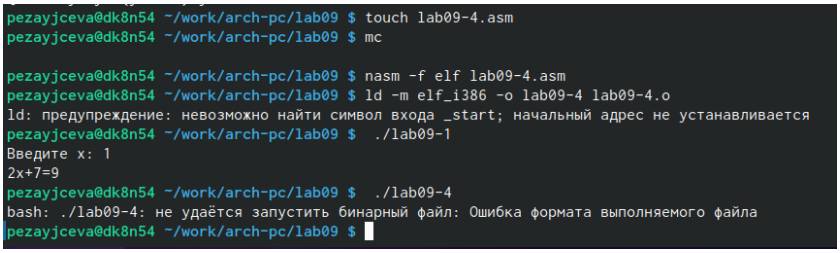

Рис. 2.11: Рис.11

С помощью отладчика GDB, анализируя изменения значений регистров, определила ошибку и исправьте ее. Ошибка была в строках: add ebx,eax mov ecx,4 mul ecx add ebx,5 mov edi,ebx

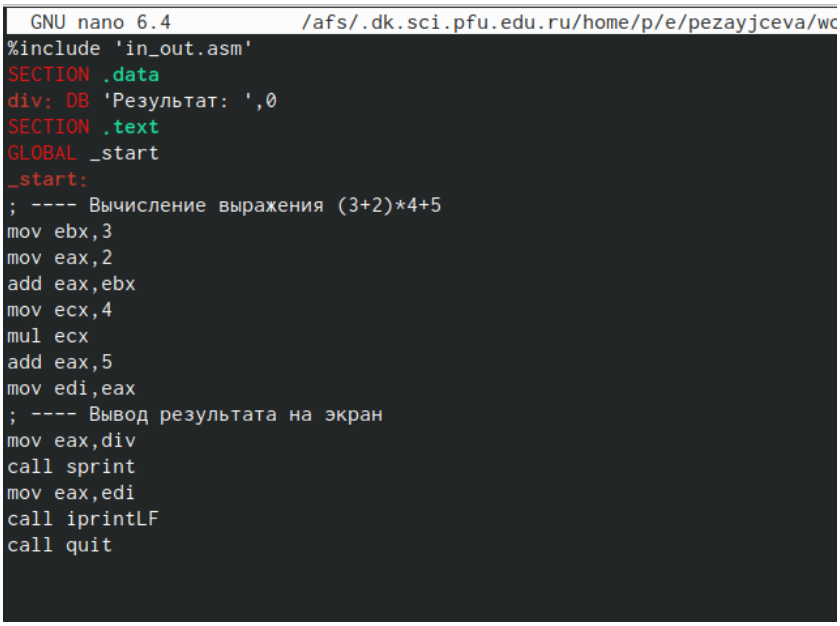

Рис. 2.12: Рис.12

Исправленный листинг: %include ‘in_out.asm’ SECTION .data div: DB ‘Результат:’,0 SECTION .text GLOBAL _start _start: ; —- Вычисление выражения (3+2)*4+5 mov ebx,3 mov eax,2 add eax,ebx mov ecx,4 mul ecx add eax,5 mov edi,eax ; —- Вывод результата на экран mov eax,div call sprint mov eax,edi call iprintLF call quit

Проверила его работу.

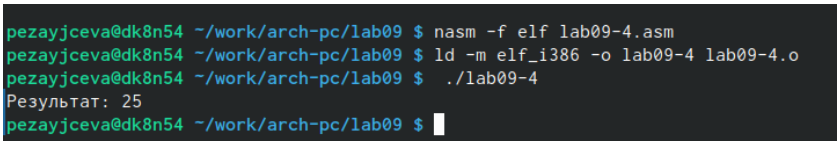

Рис. 2.13: Рис.13

3 Выводы

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