Лабораторная работа No9

Понятие подпрограммы.Отладчик GDB.

Бурыкина Софья Дмитриенва

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм.

2 Теоретическое введение

Отладчики позволяют управлять ходом выполнения программы, контролировать и из- менять 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.

Более подробно об Unix см. в [1–6].

3 Выполнение лабораторной работы

Ввелла программу с листинга 9.1(рис. 3.1).

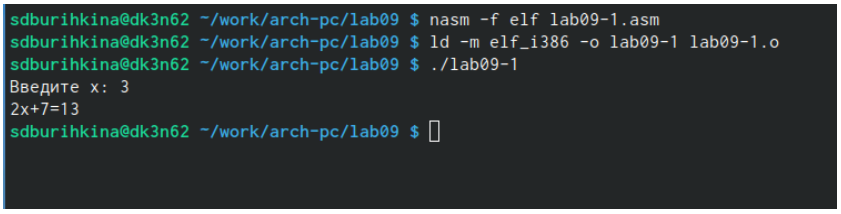

Рис. 3.1: Создала исполняемый файл и проверила его работу

Изменила текст программы добавив подпрограмму _subcalcul (рис. 3.2).(рис. 3.3).

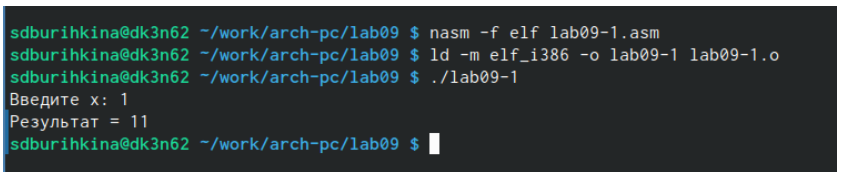

Рис. 3.2: Результат программы

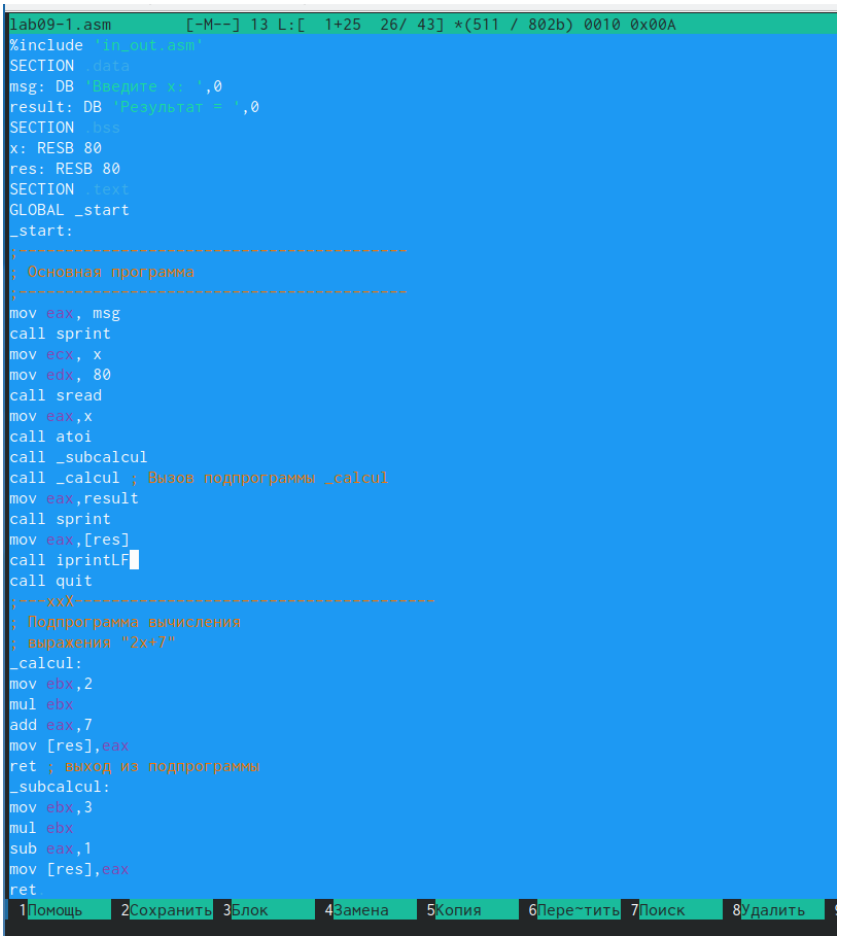

Рис. 3.3: Изменённая программа

Создала файл lab09-2.asm с текстом программы из Листинга 9.2.(рис. **??**).(рис. 3.4).

Рис. 3.4: Проверила его работу, запустив её в оболочке GDB

Установила брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустила её.(рис. 3.5). (рис. **??**). (рис. 3.6). (рис. 3.7).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/s/d/sdburihkina/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9       mov eax, 4
```

Рис. 3.5: Запуск программы

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:     mov    $0x4,%eax
   0x08049005 <+5>:     mov    $0x1,%ebx
   0x0804900a <+10>:    mov    $0x804a000,%ecx
   0x0804900f <+15>:    mov    $0x8,%edx
   0x08049014 <+20>:    int    $0x80
   0x08049016 <+22>:    mov    $0x4,%eax
   0x0804901b <+27>:    mov    $0x1,%ebx
   0x08049020 <+32>:    mov    $0x804a008,%ecx
   0x08049025 <+37>:    mov    $0x7,%edx
   0x0804902a <+42>:    int    $0x80
   0x0804902c <+44>:    mov    $0x1,%eax
   0x08049031 <+49>:    mov    $0x0,%ebx
   0x08049036 <+54>:    int    $0x80
End of assembler dump.
```

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:     mov    eax,0x4
   0x08049005 <+5>:     mov    ebx,0x1
   0x0804900a <+10>:    mov    ecx,0x804a000
   0x0804900f <+15>:    mov    edx,0x8
   0x08049014 <+20>:    int    0x80
   0x08049016 <+22>:    mov    eax,0x4
   0x0804901b <+27>:    mov    ebx,0x1
   0x08049020 <+32>:    mov    ecx,0x804a008
   0x08049025 <+37>:    mov    edx,0x7
   0x0804902a <+42>:    int    0x80
   0x0804902c <+44>:    mov    eax,0x1
   0x08049031 <+49>:    mov    ebx,0x0
   0x08049036 <+54>:    int    0x80
End of assembler dump.
```

Рис. 3.6: Переключилась на отображение команд с Intel’овским синтаксисом,

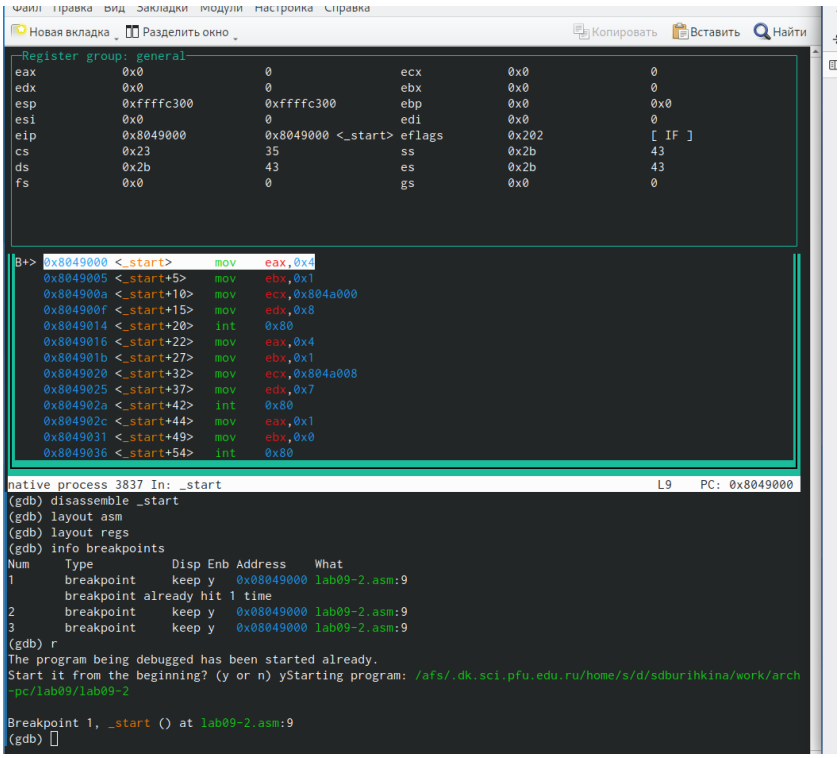

Рис. 3.7: Включила режим псевдографики для более удобного анализа программы

Посмотрела информацию о всех установленных точках (рис. 3.8).

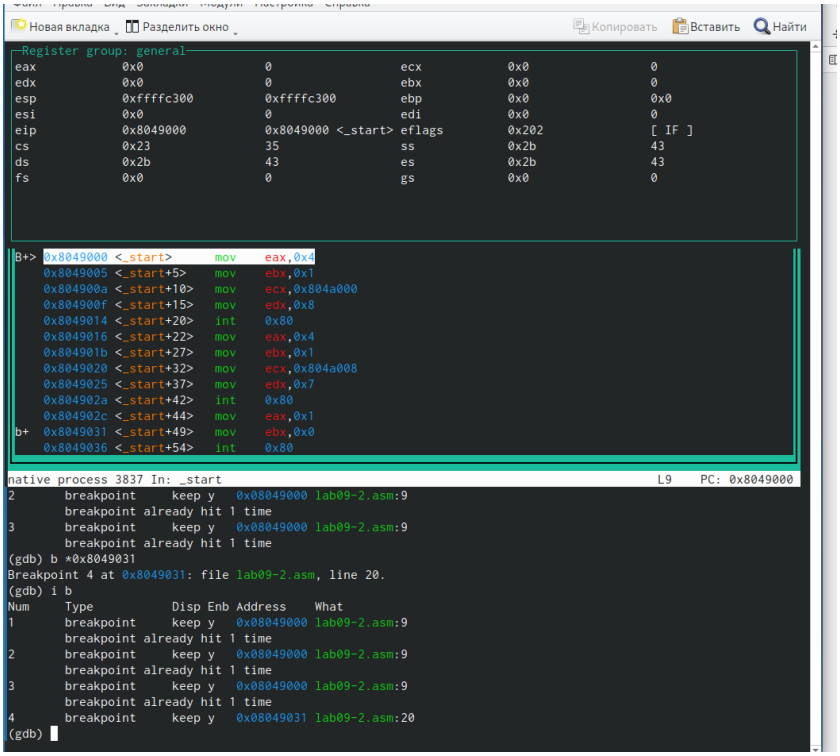

Рис. 3.8: Установка точки останова по адресу инструкции

Посмотрим значение переменной msg1 по имени (рис. 3.9). (рис. 3.10).

Рис. 3.9: Значение

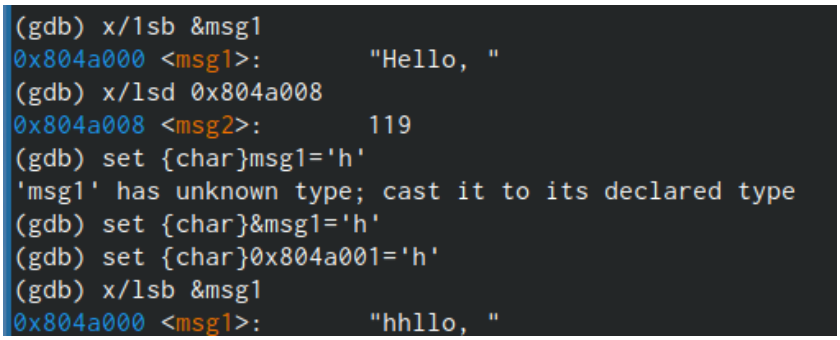

Рис. 3.10: Значение

С помощью команды set изменила значение регистра ebx. (рис. 3.11).

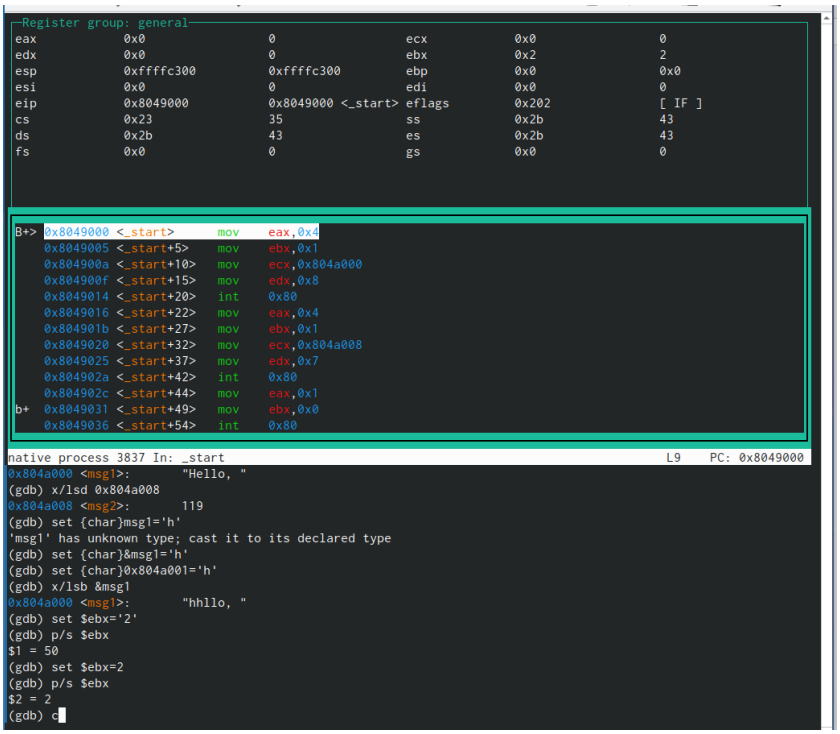

Рис. 3.11: Вывод команды

Я копирую файл lab8-2.asm в папку с лабараторной номер 9 и называю его lab09-3.asm,создаю исполняемый файл и запускаю его через gdb. Эта прогрмма должна находить произведение аргументов. Тауже ставлю точку останова на месту старт (рис. 3.12).

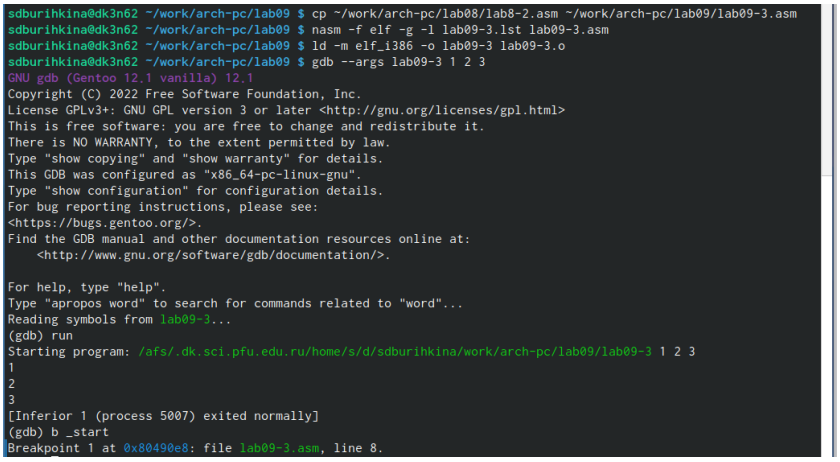

Рис. 3.12: Запуск программы с аргументом 1 2 3, усттановка брейкпоина

Рассматриваю позиции стека, так как у меня всего три аргумента на шаге (+20) выдаёт ошибку (рис. 3.13).

Рис. 3.13: Позиция стека

4 Самостоятельная работа

Создаю файл lab09-3.asm для выполнение первого задания из самостоятельной работы. Ввожу текст листинга 9.1 для удобства. Пишу прграмму согласно 7 варианту лабаратоорной работы номер 8 (рис. 4.1).

```
sdburihkina@dk3n62 ~/work/arch-pc/lab09 $ nasm -f elf lab09-3.asm
sdburihkina@dk3n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
sdburihkina@dk3n62 ~/work/arch-pc/lab09 $ ./lab09-3
Введите x: 1
3(x+2)=9
```

Рис. 4.1: Результат работы

Создаю файл lab09-4.asm для выполнения второго пункто лабараторной работы, ввожу туда текст листинга 9.3, сохраняю, создаю исполняемый файл и запускаю его. Убеждаюсь, что результат неверный. Ответ должен быть 25 (рис. 4.2).

```
ecx            0x0             0
ebx            0x5             5
ebp            0x0             0x0
edi            0x0             0
eflags         0x206           [ PF IF ]
ss             0x2b            43
es             0x2b            43
gs             0x0             0
```

Рис. 4.2: Запуск файла

Запускаю откладчик, смотрю как изменяется регистры eax,ebx,ecx пошагово с

помощью команды si. Замечаю что на третьем шаге регистр ebx имеет значение 5, а на четвёртом шаге ecx и eax перемножаются, что даёт неверный результат. Следовательно изменяю программу так, чтобы результат сложения записывался и eax(рис. 4.3).

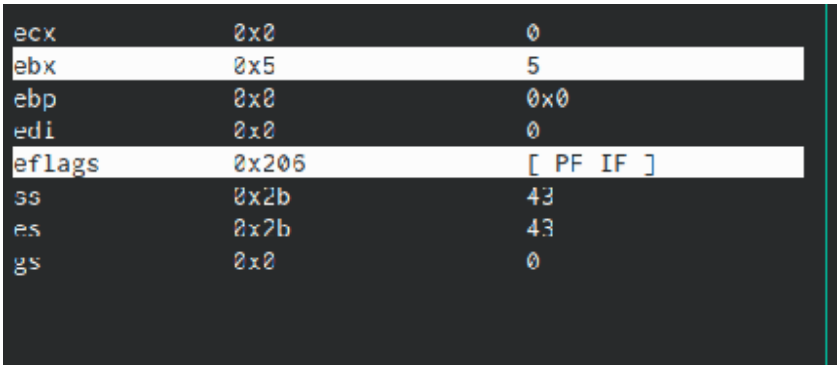

Рис. 4.3: Регистр ebx=5, а должен быть eax=5

Снова неверный результат, нахожу ошибку в именованиях регистрах, вижу ошибку в сложение и меняю это (рис. 4.4).

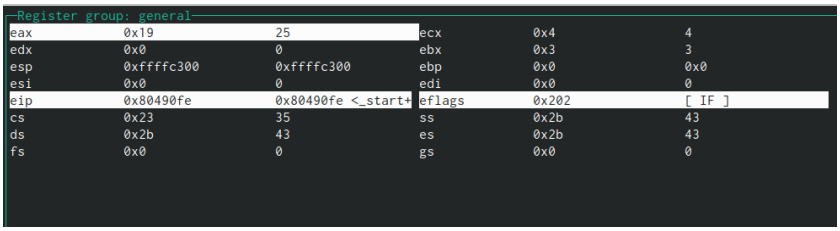

Рис. 4.4: Ошибка в сложение

Создаю исполняемый файл и запускаю его (рис. 4.5).

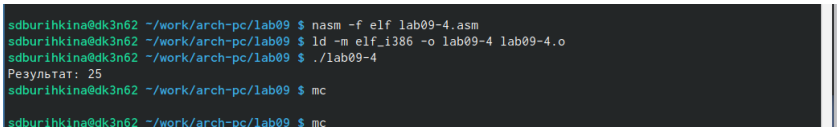

Рис. 4.5: Проверка программы

5 Выводы

Приобрела навыки написание программ с использованием подпрограмм. Познакомилась с мотодами откладки при помощи GDB и его основными возможностями. # Список литературы{.unnumbered}

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: https://www.gnu.org/software/bash/manual/.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.