Лабораторная работа номер 9.

Понятие подпрограммы. Отладчик GDB.

Сорокин Кирилл

Содержание

# 1 Цель работы

Научиться писать подпрограммы и работать с отладчиком GDB.

# 2 Задание

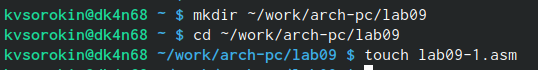
Используя написать подпрограммы внутри программ, а также проверять работу программ с помощью отладчика.

# 3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. 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.

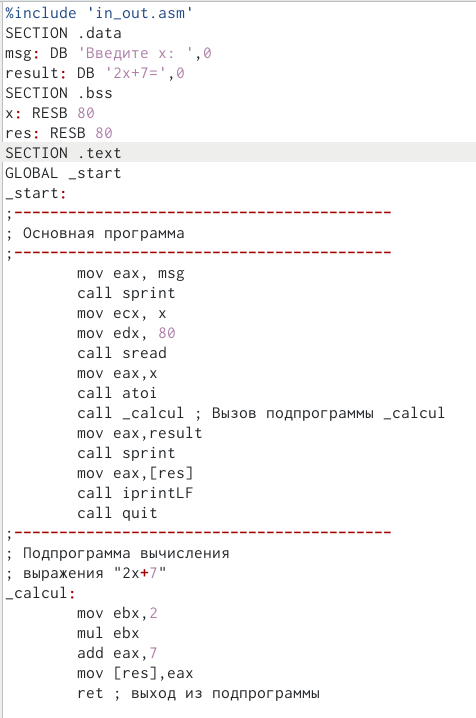
# 4 Выполнение лабораторной работы

Создадим директории и файл для работы (рис. ??).



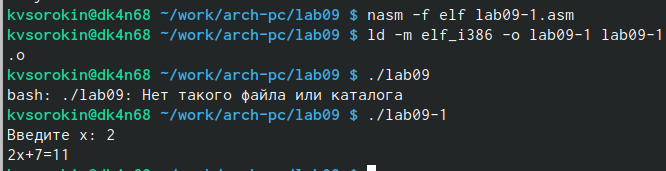
Директории и файл

Запишем листинг 9.1 в файл(рис. ??).



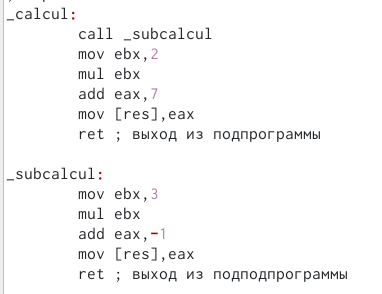
Листинг 9.1

Проверим работу файла(рис. ??).



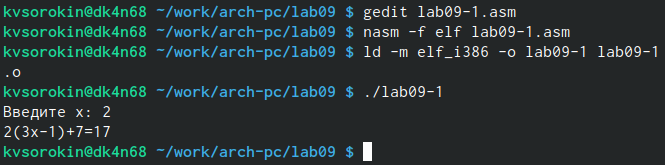
Работа файла

Изменим код программы для вложенных функций (рис. ??).



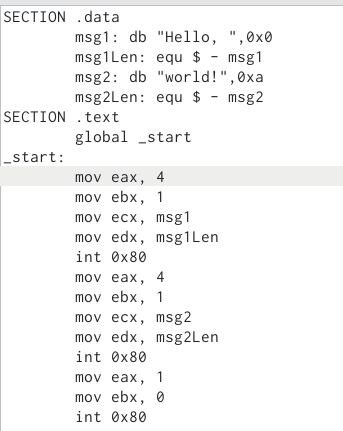
Изменение кода

Проверим работу (рис. ??).



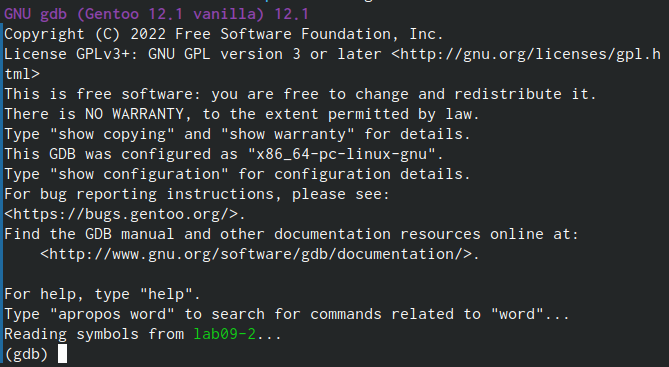
Работа файла

Введём листинг второй программы в новый файл (рис. ??).



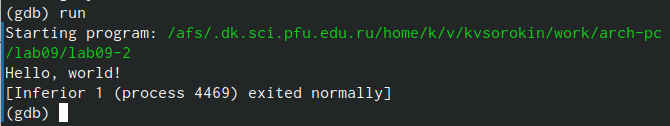
Листинг второй программы

Скомпилируем файлы и запустим в оболочке GDB командой gdb lab09-2(рис. ??).



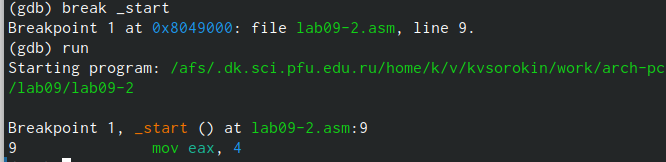
Работа GBD

Запустим программу командой run (рис. ??).



Запуск программы

Поставим брейкпоинт и запустим программу снова (рис. ??).



Брейкпоинт

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. ??).



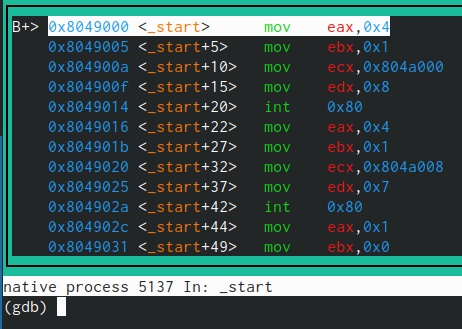
Дисассимилированный код

Переключим на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. ??).



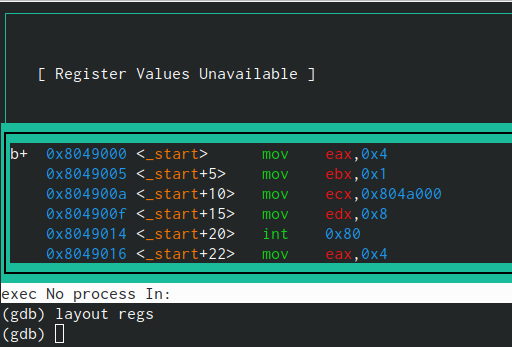
Переключение кода

Командой layout asm, откроем окно с дисассимилированием программы (рис. ??).



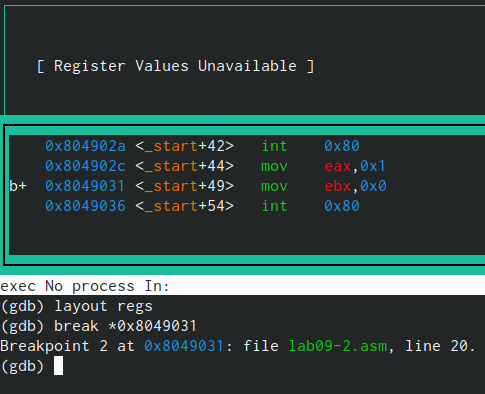
layout asm

Командой layout regs, откроем окно с названия регистров и их текущие значения(рис. ??).



layout regs

Командой break установим ещё один брейкпоинт (рис. ??).



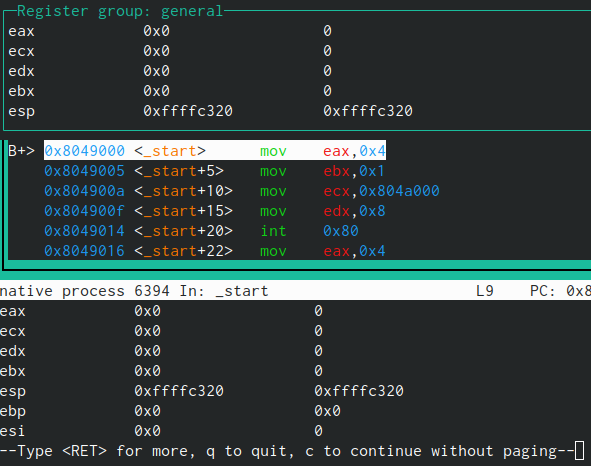
break

Командой i b посморим позиции брейкпоинтов(рис. ??).

Команда i b

Команда i b

Командой info registers, увидим информацию о регистах(рис. ??).



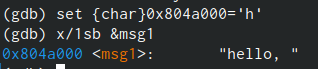
Команда info registers

Посмотрим значение переменной msg1 по имени (рис. ??).

Значение msg1

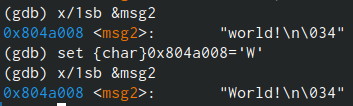
Значение msg1

Изменим значение первой буквы на маленькую версию(рис. ??).



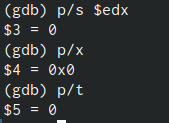
Замена символа

Аналогично сделаем с msg2 (рис. ??).



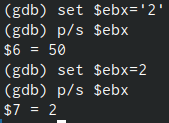
Изменение msg2

Выведим в различных форматах значение регистра edx (рис. ??).



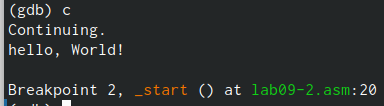
Значение регистра edx

Заменим значение ebx на ‘2’, а затем на 2. Заметим, что выодится разное значене из-за разных форматов данных (рис. ??).



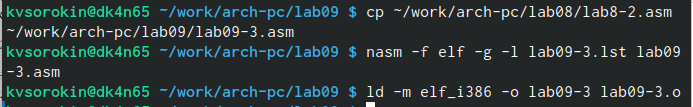
Значение ebx

Довыполним программу командой с, затем выйдем из gdb командой quit(рис. ??).



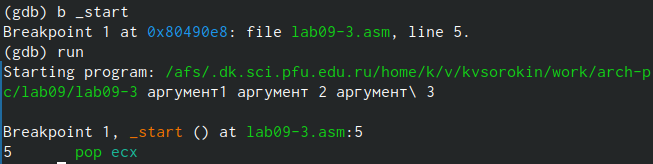
Довыполним программу

Скопируем файл lab8-2.asm с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm и скомпилируем файлы (рис. ??).



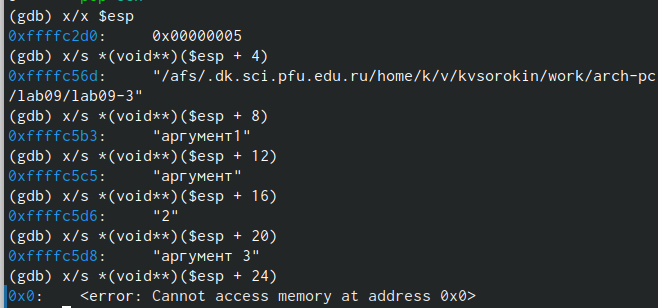
lab09-3.asm

Запустим gdb передав аргументы вместе с файлом, постави брейкпоинт и увидим количество аргументов командной строки (рис. ??).



Запуск через gdb

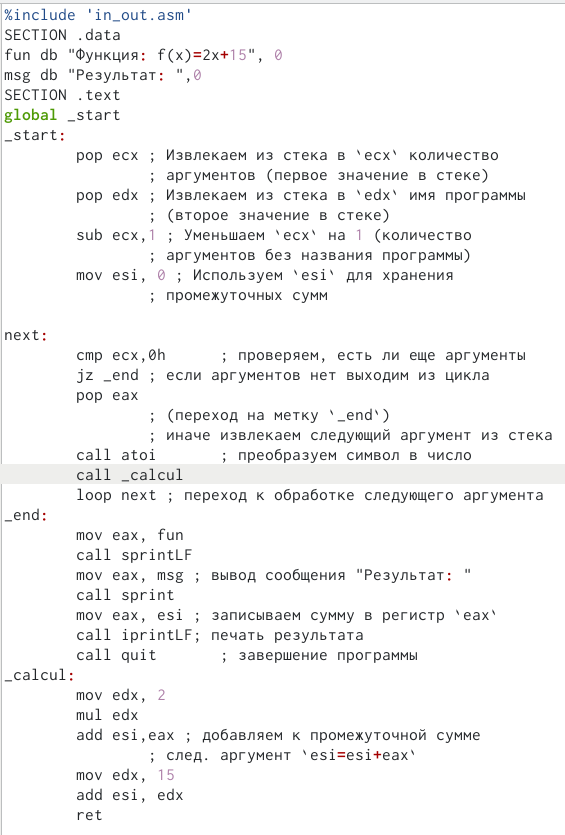
Посморим все позиции стека по адресу. Заметим, что все аргументы расположенны на растоянии в 4 байта друг от друга (рис. ??).



Позиции в стеке

## 4.1 Самостоятельная работа

Изменим текст самостоятельной работы из 8 лабораторной так, чтобы в ней задействовалась подпрограмма(рис. ??).



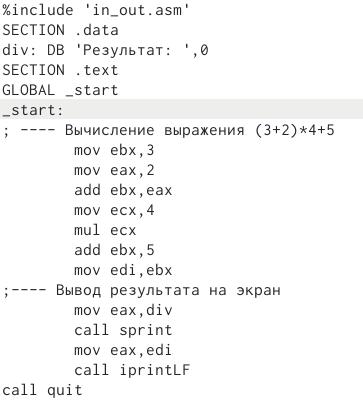
Изменённый текст 8 лабораторной

Проверим, что она всё ещё работает корректно (рис. ??).

Работа самостоятельной

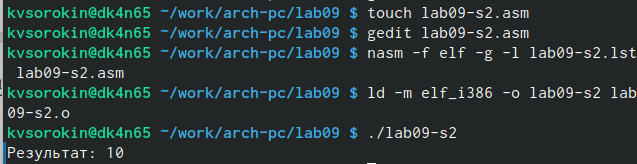
Работа самостоятельной

Введём листинг 9.3 во второй файл самостоятельной работы (рис. ??).



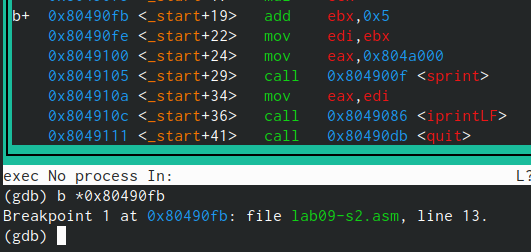
Текст второго файла

Проверим работу и увидим, что результат не корректен (рис. ??).



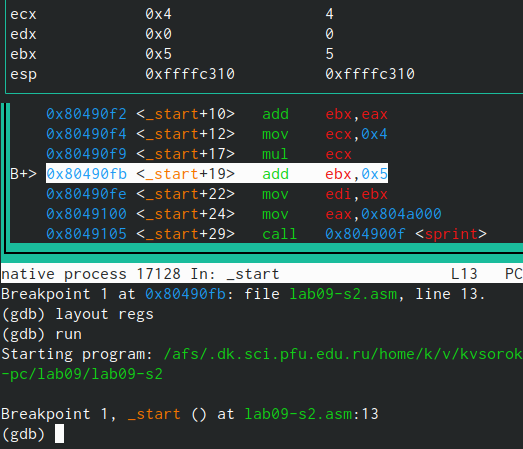
Работа второй программы

Запустим программу в отладчике и поставим брейкпоинт в месте, до того как прибавили 5 (рис. ??).



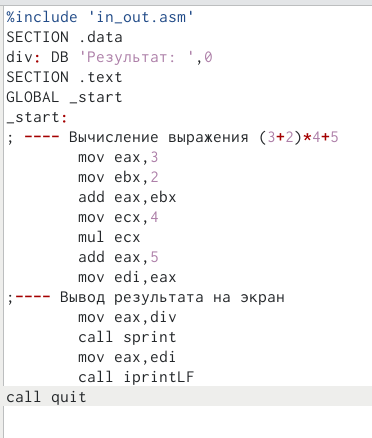
Брейкпоинт

Увидим, что значение ebx не равно 20, значит проблема в умножении(рис. ??).



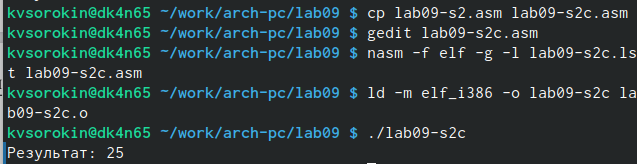
Поиск ошибки

Исправим ошибку(рис. ??).



Код исправленной программы

Удостоверимся, что теперь программа работает корректно (рис. ??).



Работа lab09-s2c

# 5 Выводы

Мы научились GDB, а также писать подпрограммы

# Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnightcommander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com Learning-bash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. :Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. - 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. -СПб. : Питер,
17. — 1120 с. — (Классика Computer Science)