Отчёта по лабораторной работе №9

НПИ-03-23

Махмудов Суннатилло Баходир угли

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

# 2 Задание

1. Реализация подпрограмм в NASM.
2. Отладка программ с помощью GDB.
3. Работа с данными программы в GDB.
4. Обработка аргументов командной строки в GDB.
5. Преобразование программы из лабораторной работы №8, реализовав

вычисление значения функции как подпрограмму.

1. Проверить неправильную работу программы, проанализировав изменения

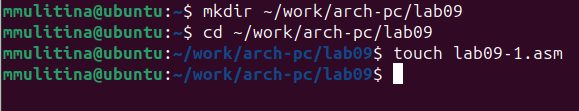
значения регистров. Определить ошибку и исправить ее

# 3 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 прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно 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.

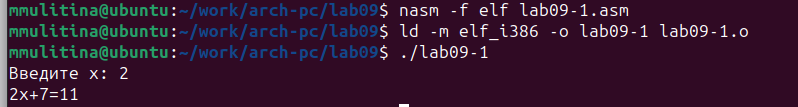
# 4 Выполнение лабораторной работы

Создадим каталог и файл для лабораторной работы(рис. ??).



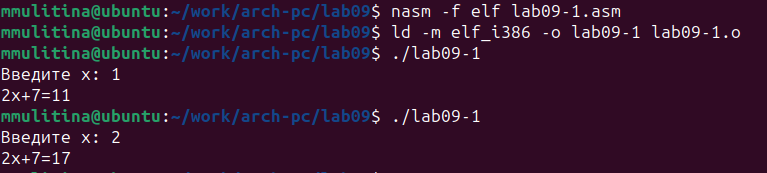
Создание каталога

Введём текст программы и запустим её для проверки(рис. ??).



Программа

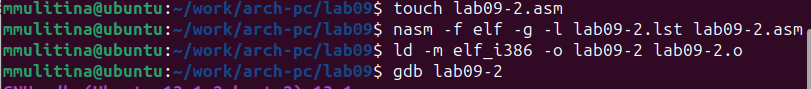
Добавим подпрограмму \_subcalcul, запустим программу для проверки(рис. ??).



Программа

Создадим файл lab09-2.asm, введём в него текст программы, получим исполняемый

файл и загрузим его в отладчик(рис. ??).



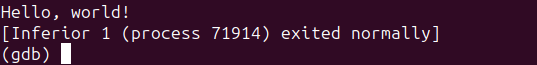
Отладчик

Проверим работу программы, запустим ее в оболочке GDB(рис. ??).

Отладчик

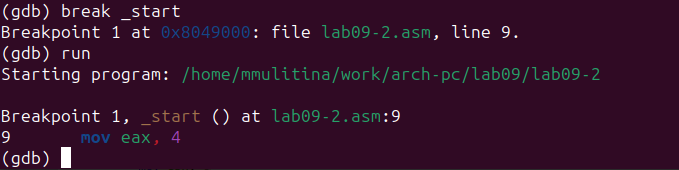
Отладчик

(рис. ??).



Отладчик

Установим брейкпоинт и запустим программу(рис. ??).



Брейкпоинт

Посмотрим дисассимилированный код (рис. ??).



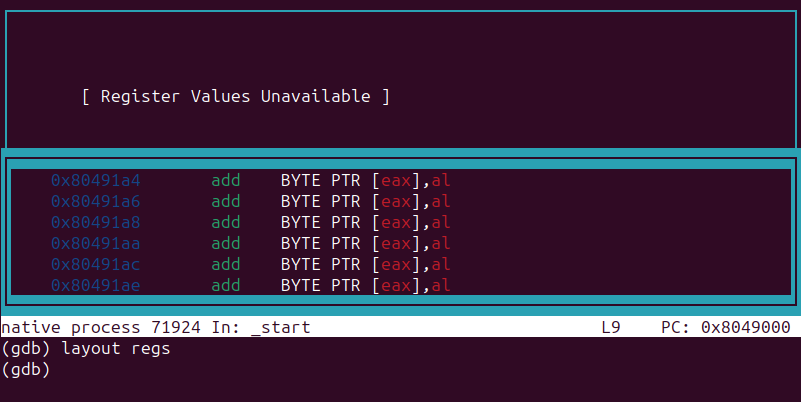
Дисассимилированный код

Переключимся на отображение команд с Intel синтаксисом (рис. ??).



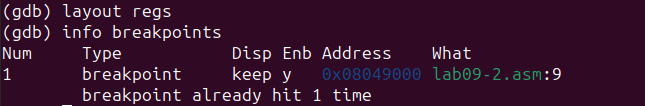
Отображение

Включим режим псевдографики (рис. ??).



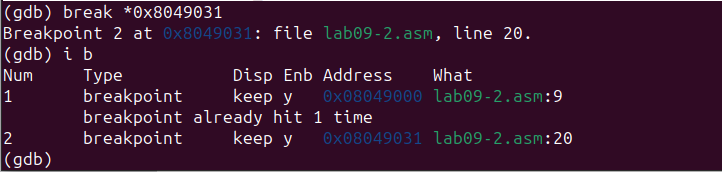
Псевдографика

Проверим точки останова с помощью команды info breakpoints (рис. ??).



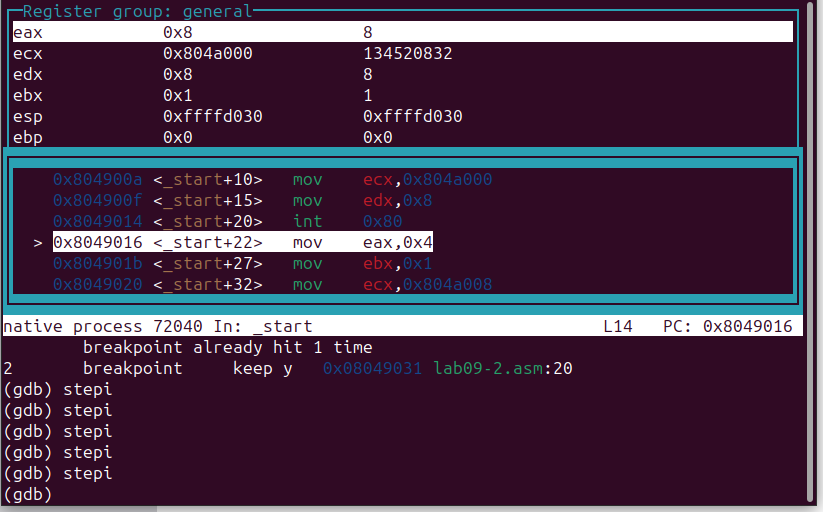
Точки останова

Установим ещё одну точку останова и снова посмотрим информацию о точках останова (рис. ??).



Точки останова

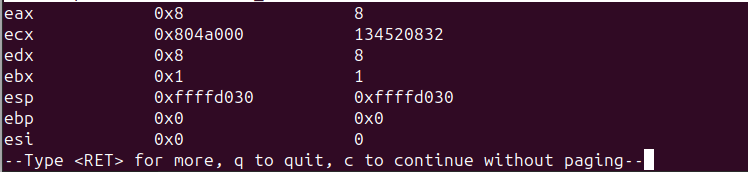
Выполним 5 инструкций с помощью команды stepi и проследим изменения регистров(рис. ??).



stepi

Изменились значения регистров eax, ecx, ebx, edx.

Посмотрим значения регистров с помощью info registers(рис. ??).



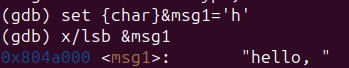
Значения регистров

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

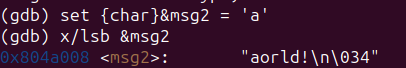
Значение переменной

Значение переменной

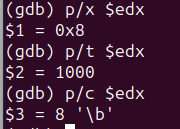
Изменим первый символ переменной (рис. ??).

{#fig:001 width=70%

Изменим любой символ второй переменной msg2 (рис. ??).

{#fig:001 width=70%

Посмотрим значений регистра edx (рис. ??).

{#fig:001 width=70%

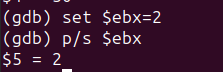
С помощью set изменим значение регистра ebx (рис. ??).

Изменение значения регистра{#fig:001 width=70%

Проверим его значение(рис. ??).

Значение{#fig:001 width=70%

Снова изменим значение ebx(рис. ??).

{#fig:001 width=70%

В первом случае мы ввели символьное значение, во втором цифру.

Скопируем файл из прошлой лабораторной работы(рис. ??).

Копирование файла{#fig:001 width=70%

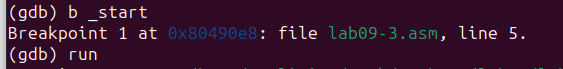
Создадим исполняемый файл (рис. ??).

Исполняемый файл{#fig:001 width=70%

Загрузим исполняемый файл в отладчик, указав аргументы(рис. ??).

Отладчик{#fig:001 width=70%

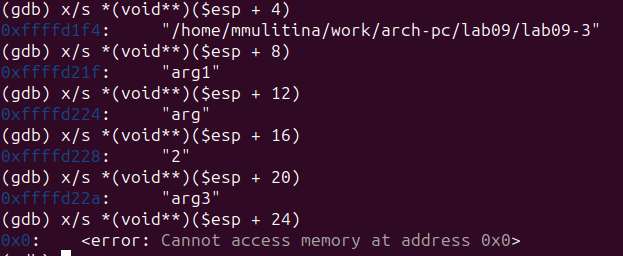
Установим точку останова перед первой инструкцией в программе и запустим её(рис. ??).

{#fig:001 width=70%

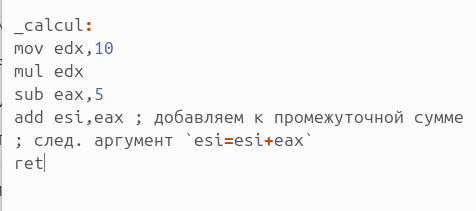
Посмотрим значение регистра esp, где хранится адрес вершины стека(рис. ??).

Регистр esp{#fig:001 width=70%

Посмотрим остальные позиции стека по адресу (рис. ??).

{#fig:001 width=70%

Шаг изменения равен 4, т.к. у нас 4 аргумента.(рис. ??).

{#fig:001 width=70%

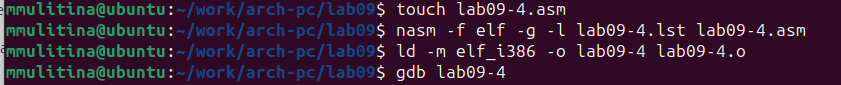
Шаг изменения равен 4, т.к. у нас 4 аргумента.

# 5 4.1 Задания для самостоятельной работы

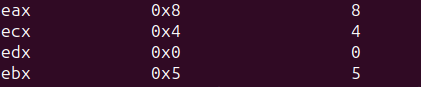
1. Преобразуем программу из лабораторной работы №8, реализовав вычисление значения функции как подпрограмму (рис. ??).

Подпрограмма{#fig:001 width=70%

Запустим программу для проверки(рис. ??).

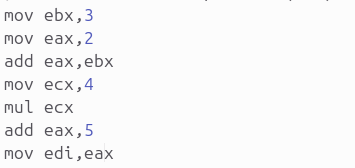
{#fig:001 width=70%

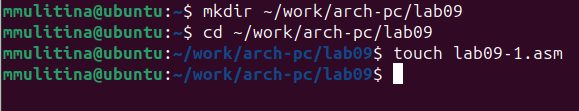
1. Создадим файл для программы, введём в него текст программы, запустим его в отладчике GDB (рис. ??).

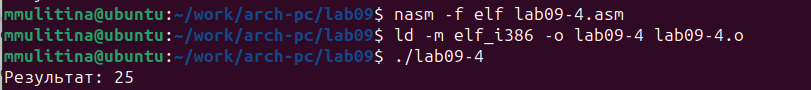
{#fig:001 width=70%

При умножении с помощью mul, мы умножаем eax на ecx b записываем в eax. Получаем 24=9 вместо (3+2)4(рис. ??).

Проверка{#fig:001 width=70% Потом скалдываем с регистром ebx 5 и получаем 10. Проверим это, запустив программу(рис. ??).

{#fig:001 width=70% Исправим программу(рис. ??).

{#fig:001 width=70% Запустим её для проверки (рис. ??).

{#fig:001 width=70%

# 6 Выводы

В процессе выполнения работы я приобрела навыки написания программ с использованием подпрограмм и познакомилась с методами отладки при помощи GDB и его основными возможностями.

# 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://midnight-commander. 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,
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).