Отчет по лабораторной работе №9

Дисциплина: Архитектура компьютера

Абронина Алиса Кирилловна

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. Выполнение лабораторной работы
2. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Отладка — это когда программист ищет и устраняет ошибки в коде. Этот процесс состоит из четырёх шагов:

1. Обнаружение ошибки — вы видите, что что-то пошло не так.
2. Поиск места ошибки — нужно понять, где именно в коде проблема.
3. Определение причины ошибки — почему произошла эта ошибка?
4. Исправление ошибки — устранение проблемы.

Есть несколько типов ошибок:

* Синтаксические ошибки — они возникают, когда код написан неправильно для языка программирования. Эти ошибки легко заметить, потому что программа просто не запустится.
* Семантические ошибки — эти ошибки сложнее. Программа работает, но делает не то, что вы хотели.
* Ошибки времени выполнения — такие ошибки появляются уже после запуска программы. Например, деление на ноль или переполнение памяти могут остановить работу программы.

Чтобы найти ошибку, лучше всего разбивать программу на маленькие части и проверять каждую часть отдельно. Когда находишь проблему, обычно становится понятно, почему она возникла. Исправляешь ошибку, снова запускаешь программу, и если всё хорошо, продолжаешь дальше. Если нет — ищем следующую ошибку!

# 4 Выполнение лабораторной работы

Создаю каталог для выполнения лабораторной работы №9 (рис. -fig. 1).

![Рис. 1: Создание рабочего каталога](data:application/octet-stream;base64,)

Рис. 1: Создание рабочего каталога

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. -fig. 2).

![Рис. 2: Запуск программы из листинга](data:application/octet-stream;base64,)

Рис. 2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения f(g(x)) (рис. -fig. 3).

![Рис. 3: Изменение программы первого листинга](data:application/octet-stream;base64,)

Рис. 3: Изменение программы первого листинга

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компилирую и запускаю в отладчике (рис. -fig. 4).

![Рис. 4: Запуск программы в отладчике](data:application/octet-stream;base64,)

Рис. 4: Запуск программы в отладчике

Запустив программу командой run, я убедилась в том, что она работает исправно (рис. -fig. 5).

![Рис. 5: Проверка программы отладчиком](data:application/octet-stream;base64,)

Рис. 5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку \_start и снова запускаю отладку. Далее смотрю дисассимилированный код программы, перевожу на команд (рис. -fig. 6). Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - pазмер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом $; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

![Рис. 6: Запуск отладичка с брейкпоинтом](data:application/octet-stream;base64,)

Рис. 6: Запуск отладичка с брейкпоинтом

Включаю режим псевдографики для более удобного анализа программы (рис. -fig. 7).

![Рис. 7: Режим псевдографики](data:application/octet-stream;base64,)

Рис. 7: Режим псевдографики

Проверяю в режиме псевдографики, что брейкпоинт сохранился.Устаналиваю еще одну точку останова по адресу инструкции (рис. -fig. 8).

![Рис. 8: Список брейкпоинтов; Добавление второй точки останова](data:application/octet-stream;base64,)

Рис. 8: Список брейкпоинтов; Добавление второй точки останова

Просматриваю содержимое регистров командой info registers (рис. -fig. 9).

![Рис. 9: Просмотр содержимого регистров](data:application/octet-stream;base64,)

Рис. 9: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. -fig. 10).

![Рис. 10: Просмотр содержимого переменных](data:application/octet-stream;base64,)

Рис. 10: Просмотр содержимого переменных

Вывожу в различных форматах значение регистра edx (рис. -fig. 11).

![Рис. 11: Просмотр значения регистра разными представлениями](data:application/octet-stream;base64,)

Рис. 11: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. -fig. 12).

![Рис. 12: Примеры использования команды set](data:application/octet-stream;base64,)

Рис. 12: Примеры использования команды set

Копирую программу из предыдущей лабораторной работы в текущий каталог и и создаю исполняемый файл с файлом листинга и отладки. Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились. (рис. -fig. 13).(рис. -fig. 14)

![Рис. 13: Подготовка новой программы](data:application/octet-stream;base64,)

Рис. 13: Подготовка новой программы

![Рис. 14: Проверка работы стека](data:application/octet-stream;base64,)

Рис. 14: Проверка работы стека

# 5 Выполнение заданий для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. -fig. 15).

![Рис. 15: Измененная программа предыдущей лабораторной работы](data:application/octet-stream;base64,)

Рис. 15: Измененная программа предыдущей лабораторной работы

Запускаю программу в режике отладичка и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию. Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. -fig. 16).

![Рис. 16: Проверка корректировок в программме](data:application/octet-stream;base64,)

Рис. 16: Проверка корректировок в программме

# 6 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, а так же познакомилась с методами отладки при поомщи GDB и его основными возможностями.