Содержание

Введение

В настоящее время обучение языку ассемблера является важной составляющей многих программистских курсов.

Очень часто студенты, изучающие язык ассемблера, сталкиваются с проблемами при настройки среды разработки, при использовании инструментов компиляции и отладки.

Преподаватели таких курсов также сталкиваются с проблемами организации учебного процесса.

Создание удобного, интерактивного и производительного программного инструмента удалённой сборки и отладки програм на ассемблере, представляет собой актуальную задачу.

Цель данной работы состоит в разработке и инструмент удалённого запуска, отладки и проверки программ на языке ассемблера, удовлетворяющий следующим свойствам:

1. Не ожидает, что студент хорошо разбирается в инструментах компиляции, отладки, не требует опыта работы с командной строкой;
2. Минимизирует количество элементарных шагов, требуемых для запуска программ;
3. Не требует установки дополнительного ПО на устройстве студента;

Задачи данной работы:

1. Исследование существующих решений для запуска и отладки программ на языке ассемблера, а также решений для обучения языку ассемблера;
2. Формирование требований к разрабатываемому инструменту;
3. Исследование возможности создания инструмента;
4. Разработка программной архитектуры инструмента;
5. Реализация инструмента;

Объектом исследования являются системы запуска и отладки программ на языке ассемблера.

Предметом исследования является интерактивность и удобство использования таких систем.

Практическая ценность работы состоит в том, что разработанный инструмент позволит проводить обучение языку ассемблера более эффективно для студентов.

Глава 1. Обзор существующих решений

1.1 Критерии сравнения

Смотрим на следующие критерии:

1. Поддержка запуска ассемблерного кода на разных диалектах и на разных архитектурах;
2. Поддержка отладки: выполнение по шагам, поддержка точек останова, редактирования регистров/памяти, визуализация стека вызовов;
3. Поддержка задач и их автоматической проверки;
4. Поддержка интеграции с системами управления обучением;
5. Возможность работы без установки дополнительного программного обеспечения на устройстве пользователя;
6. Возможность самостоятельной установки и развёртывания системы на выделенном сервере, доступность исходного кода;

1.2 Существующие решения

1.2.1 Ideone

Ideone является онлайн компилятором и средой разработки, поддерживающей более 60 языков программирования, в том числе несколько диалектов ассемблера. Система доступна по адресу `https://ideone.com`.

Поддерживается запуск ассемблерного кода на архитектурах x86 (NASM и GNU диалекты) и x86-64 (только NASM диалект). Отладка не поддерживается.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Взаимодействие с системой происходит через веб-интерфейс, установки дополнительного ПО не требуется.

Система имеет закрытый исходный код, самостоятельно установить систему на выделенный сервер не представляется возможным.

1.2.2 OneCompiler

Примерно то же самое, что и Ideone, правда поддерживает только x86 с NASM диалектом. Система доступна по адресу `https://onecompiler.com/assembly`. Не уверен, нужно ли включать в список аналогов, или же хватит Ideone.

1.2.3 ASM Debugger

ASM Debugger является инструментом для пошаговой отладки простых программ на языке ассемблера. Инструмент доступен по адресу `http://asmdebugger.com`.

Особенностью инструмента является то, что он не использует запуск программ на реальном аппаратном обеспечении. Вместо этого, на языке Javascript реализовано подмножество инструкций x86 ассемблера.

Поддерживается запуск ассемблерного кода на архитектуре x86 с NASM диалектом. Поддерживается пошаговое исполнение, просмотр значений регистров.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Взаимодействие с инструменетом происходит через веб-интерфейс, установки дополнительного ПО не требуется.

Инструмент имеет открытый исходный код, доступный по адресу `https://github.com/dinoqqq/asmdebugger`. Соответственно, есть возможность уста-

новить его на выделенный сервер.

1.2.4 SASM (SimpleASM)

SASM представляет из себя кроссплатформенную среду разработки на языке ассемблера для архитектур x86 и x86-64 с использованием диалектов NASM, GNU, FASM, MASM. Инструмент доступен по адресу `https://dman95.github.io/SASM/index.html`.

Поддерживается запуск ассемблерного кода, поддерживается выполнение по шагам, точки останова, просмотр и редактирование регистров и памяти, а также произвольные команды GDB.

Поддержки задач, их автоматической проверки нет, соответственно нет и интеграции в системы управления обучением.

Для использования инструмента необходима его установка на компьютер пользователя. Исходный код инструмента доступен по адресу `https://github.com/Dman95/SASM`.

1.2.5 JetBrains Clion + EduTools

Clion — это интегрированная среда разработки от компании JetBrains, предназначенная, в первую очередь, для разработки приложений на языках C и C++. Язык ассемблера не поддерживается ни в каком виде, но существуют сторонние плагины, которые решают эту проблему. Здесь я рассмотрю самый популярный такой плагин, он доступен по адресу `https://plugins.jetbrains.com/plugin/9759-nasm-assembly-language`.

Компиляция и запуск кода на языке ассемблера возможны, если модифицировать должным образом файлы системы описания сборки CMake. Отладка ассемблерного кода не поддерживается.

Плагин EduTools позволяет создавать и писать задачи с автоматическими тестами, что упрощает проверку решений. Отсутствует поддержка задач с закрытыми (недоступными для обучающегося) тестами. Интеграция с системами управления обучением отсутствует.

Для использования данной среды разработки необходима её установка на компьютер пользователя. Она имеет закрытый исходный код.

1.2.6 GitHub Classroom + Visual Studio Code

GitHub Classroom — это сервис, позволяющий давать учебные задания в виде git-репозиториев. GitHub Classroom позволяет добавить кнопку «открыть в Visual Studio Code», которая позволяет открыть репозиторий с преднастроенными плагинами в этом редакторе.

Для того, чтобы настроить поддержку языка ассемблера в Visual Studio Code, требуется установка дополнительных плагинов. Также преподавателю в шаблонном репозитории необходимо будет настроить компиляцию и запуск в файлах `tasks.json` и `launch.json`. Отладка не поддерживается.

GitHub Classroom позволяет добавлять тесты через веб-интерфейс преподавателя. В качестве теста может выступать набор входных данных и эталонных ответов к ним, так и путь до скрипта для автоматической проверки. В первом случае входные данные передаются программе через стандартный поток ввода, а вывод программы сравнивается с эталонным ответом.

Необходима установка Visual Studio Code, компилятора и отладчика.

GitHub Classroom имеет закрытый исходный код, установить свою копию на выделенный сервер не представляется возможным.

1.2.7 Stepik

В системе управления обучением Stepik есть режим задания Code Challenge, который позволяет проверять код, написанный на разных языках программирования.

Поддерживается NASM диалект x86 и x86-64 ассемблера. Отладка не поддерживается.

Поддерживаются задачи и их автоматическая проверка на скрытых тестах. Тесты должны иметь вид набора входных данных и эталонных ответов. Входные данные передаются программе через стандартный поток ввода, а вывод программы сравнивается с эталонным ответом.

взаимодействие с системой происходит через веб-интерфейс, установка дополнительного ПО не требуется. Система имеет закрытый исходный код.

1.2.8 Moodle + Virtual Programming Lab

1.2.9 Git репозиторий с заданиями и скриптами для проверки

1.3 Сравнение существующих решений

Тут табличка с ранее описанными критериями сравнения.

1.4 Формирование требований к разрабатываемому инструменту

Глава 2. Исследование возможности создания инструмента

2.1 Компиляция ассемблерных программ

2.1.1 Использование компилятора GCC

Ну да, используем GCC, что тут ещё можно сказать.

2.1.2 Использование ассемблера YASM

Необходимо сказать, что в YASM нет альтернативы директиве `#line`, из-за чего при добавлении префикса/суффикса к коду строчки становятся непонятными.

2.2 Запуск ассемблерных программ

2.2.1 Изоляция с помощью seccomp

Seccomp — механизм ядра Linux, позволяющий процессу перейти в «безопасный режим», в котором запрещены все системные вызовы, кроме `exit`, `sigreturn`, `read` и `write`.

Мы просто хотим мувать регистры туда-сюда, зачем нам системные вызовы?

2.2.2 Ограничение потребляемых ресурсов с помощью prlimit

Ограничиваем процессорное время в секундах через `RLIM_CPU`.

2.2.3 Ограничение потребляемых ресурсов с помощью Docker

Docker (через механизм контрольных групп) позволяет ограничивать использование процессорного времени, используемую память в контейнерах.

2.3 Отладка ассемблерных программ

2.3.1 Использование GDB

GDB — консольный инструмент отладки программ. Позволяет отлаживать программы на самых разных языках программирования на разных платформах. Поддерживает отладочную информацию в формате DWARF.

2.3.2 GDB/MI

GDB/MI (GDB Machine Interface, машинный интерфейс GDB) позволяет взаимодействовать с процессом отладки в машиночитаемом виде. Это нам пригодится.

Тут можно вкратце описать формат взаимодействия, поставить референс на мануал GDB/MI.

2.3.3 GDB server

GDB server — программа, с которой GDB может взаимодействовать, чтобы организовать отладку кода на удалённой машине. Пригодится для разделения полномочий и ограничения ресурсов.

2.4 Выводы

Пишем выводы.