Mike Glukhov

i.am.rigidus@gmail.com

Навыки и технологии

Хорошо знаком с: Solidity, Go, C, Javascript, Common Lisp, Emacs Lisp, Forth, Assembler x86, Git, Linux
Также работал с: Java, Python, C++, Rust, Clojure, Erlang, C#, PHP, Bash

Опыт работы

Реализация системы визуального программирования

JavaScript, Emacs-lisp, Golang

Реализовал парсинг graphml-формата файлов и генератор кода, как средство для быстрого прототипирования web-приложений в микросервисной архитектуре.
Общая идея:

- При прототипировании из сценариев использования (data-flow) формируется потоки событий (event-flow), которые переносятся на граф, где узлами являются компоненты, а ребрами - API-вызовы
- С помощью этого графа просиходит линковка сниппетов кода, которые обрабатывают API
- Сниппеты выгружаются в симуляционный и "боевой" исходный код. В результате получаем симуляцию для целей тестирования и scaffolding для микросервисов
- Реинжиниринг и отладка занимает меньше времени, количество ошибок уменьшается

Полученные результаты:

- Наблюдаемые симуляции сценариев использования (отображаются на графе в ответ на действия пользователя при выполнении сценария)
- Линковка компонентов клиентского дизайна (кнопки, вызывающие API-вызовы) с соответствующими routes бакенд-сервисов
- Автоматическое построение графа зависимостей сервисов и проверка консистентности API (соответствие сценарию, совпадение сигнатур параметров, отсутствие побочных эффектов)
- Генерация интеграционных тестов для каждого сценария использования

Система опробована при прототипировании двух коммерческих проектов.

Реализация экспертной системы

Lisp

Реализовал экспертную систему на базе машины Поста для автоматического распределения регистров и некоторых компиляторных оптимизаций:

- Встраивание (инлайнинг) функций
- Удаление неиспользуемого кода
- Раскрутка циклов
- Вынесение дублирующегося кода

Экспертная система работает в паре с JIT и семплирующим профайлером и обрабатывает только "горячий" код. Работает как подключаемый Lisp-package.

Реализация языка программирования Lisp с уровня Forth

Forth

Реализоовал минимального подмножества Lisp (Без CLOS и стандартной библиотеки, но с поддержкой окружений и семантики, а также с REPL)

- Реализована на языке Forth, который я реализовал ранее на ассемблере. Таким образом получилась реализация высокоуровневого языка с самого низкого уровня.
- JIT и AOT компиляция
- Система рестартов (более совершенная чем исключения)
- REPL
- Интерактивный отладчик

Реализация языка программирования Forth

Assembler x86

Реализовал Forth VM с уровня ассемблера

- Поддержка исключений
- Объектно-ориентированное программирование
- Поддержка системных вызовов
- Форт-ассемблер (неполный, без команд сопроцессора и векторных расширений)
- Декомпилятор форт-слов

Парсер языка Solidity

Common Lisp

Реализовал парсер языка смарт-контрактов блокченйа Etherium, для использования в экспертной системе поиска уязвимых смарт-контрактов.

- Полностью реализована грамматика Solidity 0.8.0
- OpenSource проект

Система моделирования схем дискретной логики

Tcl/Tk

"Фотошоп" для проектирования цифровых схем. Вдохновлен соответствующей главой SICP. Предназначался как учебное пособие по курсу проектирования цифровых схем по книге "Д. М. Харрис и С. Л. Харрис Цифровая схемотехника и архитектура компьютера".

- Графический интерфейс для размещения разнообразных схем и связывания их проводниками
- Эмуляторы для каждой из этих схем с поддержкой задержек
- Механизм плагинов, для подключения библиотек микросхем
- Эмулятор выполнения с автоматическим рассчетом временных диаграмм
- Поддержка языка визуального проектирования "ДРАКОН"

Эмулятор Spectrum-48k (Z80-процессор)

Assembler x86, TASM

Эмулятор для запуска классических спектрумовских игр. Неоконченый проект, увяз в отладке и рефакторинге из-за не самой лучшей архитектуры. Тем не менее некоторые игры работали успешно.

Декомпилятор Borland Pascal

Turbo C++, TASM, Turbo Debugger

- Эвристический анализатор для восстановления циклов, CASE-операторов и других конструкций языка
- Система плагинов, в том числе для поддержки упакованных UPX-like архиваторами исполняемых файлов
- Интерактивный режим (вдохновленный IDA)

Отладчик Ring-0

Assembler x86, TASM

- Использование отладочных регистров x86
- Внутренний язык DebugScript для автоматизации работы

OS защищенного режима для Intel x86

Assembler x86, TASM

Минимальная демонстрационная операционная система защищенного режима с графическим интерфейсом пользователя, защитой памяти программ, вытесняющей многозадачностью и поддержкой мыши

- Разработал загрузчик (512 байт), который размещался в Master Boot Record дискеты и загружался в 0000:7c00, а потом загружал ядро, используя прерывния дисковой подсистемы BIOS (13h)
- Разработка ядра ОС
- Разработка менеджера памяти
- Разработка Program Dispatcher с вытесняющей многозадачностью
- Переключение в защищенный режим, настройка GDT и IDT
- Разработка графической библиотеки для VESA-видеорежима
- Разработка драйвера мыши (COM-порт) графического режима
- Разработка демонстрационной игры "тетрис"

Резидентный файловый монитор

Assembler x86, TASM

Программа перехватывала файловые операции сохраняя предыдущие версии файлов и запрещая редактирование исполняемых файлов

- Сплайсинг в DOS-функций прерывания 21h
- Автоматическая пошаговая трассировка прерывания до нахождения точки ветвления функций открытия/записи файлов и сплайсинг через jmp в резидентный код
- Разработка простого LZ-архиватора для инкрементального сжатия предыдущих версий файла
- Разработка интерфейса просмотра/восстановления предыдущих версий в стиле Norton Commander

Резидентный скринсейвер, будильник и часы

Assembler x86

Программа отслеживала периоды неактивности, переводила монитор в графический режим и рисовала стрелочные часы. При нажатии любой кнопки возвращала исходный режим. В конфигурационном файле можно было настроить срабатывание будильников (несколько мелодий, проигрываемых через PC-Speaker)

- Перехват вектора прерывания таймера (08h) для расчета времени неактивности
- Перехват прерывания клавиатуры (09h)
- Настройка программируемого контроллера прерываний для управления PC-Speaker
- Настройка графического видеорежима VESA
- Отрисовка графических часов