RISC-V на практике

Оглавление

Обзор главы

В этой главе мы получим некоторый практический опыт программирования на языке ассемблера RISC-V для операционной системы Linux. Даже если вы никогда раньше не программировали на ассемблере, в этой главе вы получите всю необходимую информацию, чтобы начать писать свое первое приложение "Hello World".

Цели обучения

К концу этой главы вы научитесь:

- Понимать, как эмулировать простую систему Linux с помощью QEMU.
- Писать простую программу "Hello World" на 64-битном языке ассемблера RISC-V.
- Компилировать и запускать приложение RISC-V в эмуляции.

Обзор языка ассемблера RISC-V

Необходимая документация

Во-первых, в главе 2 спецификации RISC-V Unprivileged Specification подробно рассматривается набор команд RV32I Base Integer Instruction Set, включая модель программирования и объяснение форматов команд. Хотя эта информация не является обязательной для данного курса, она, безусловно, полезна для понимания того, как архитектура RISC-V выполняет инструкции.

Для программирования инструкций ассемблера мы можем использовать справочную документацию ABI и руководство ASM, чтобы ответить на любые вопросы, которые могут возникнуть в процессе работы. Вы можете найти эти документы здесь:

- Спецификации RISC-V[1]
- Документация ABI[2]
- Руководство по ASM[3]

Опять же, ни один из этих документов не является обязательным для изучения в данной главе, но вы можете обратиться к ним, если у вас возникнут вопросы, на которые здесь нет ответов.

Обзор языка ассемблера

Эта глава будет представлять собой очень высокоуровневый обзор инструкций ассемблера RISC-V и лишь некоторые из них будут рассмотрены на практике. Мы надеемся, что этот учебник даст вам инструменты, необходимые для продолжения вашего путешествия по программированию на языке ассемблера. Если ваша цель - просто понять основы и разрабатывать приложения на языке более высокого уровня, этот курс, скорее всего, покроет большую часть необходимой вам информации.

RISC-V – это архитектура с "сокращенным набором инструкций", и поэтому в ней не так много инструкций, которые нужно изучать. В этом учебнике мы используем только 3 инструкции: LA (загрузка абсолютного адреса), ADDI (немедленное добавление) и ECALL. Инструкция ECALL используется для выполнения служебного запроса к среде выполнения. В нашем приложении Hello World мы будем использовать только два вызова, один для "записи" и один для "выхода".

---

[1] https://riscv.org/technical/specifications/

[2] https://riscv.org/technical/specifications/

[3] https://github.com/riscv-non-isa/riscv-asm-manual/blob/master/riscv-asm.md

Полный список инструкций можно найти в спецификации RISC-V Unprivileged Specification в главе 24 "RV32/64G Instruction Set Listings". Если вы хотите узнать больше о программировании на языке ассемблера, существует множество книг и курсов. Для получения дополнительной информации посетите веб-сайт[4] RISC-V.

Начало работы с QEMU

Компиляция необходимых двоичных файлов

Если вы хотите следить за видео, вы, конечно, можете это делать. Однако следует отметить, что создание эмуляционной среды - задача не из легких. Мы бы настоятельно рекомендовали вам пока просто следовать за нами, если у вас нет опыта компиляции ядра Linux.

Инструкции по компиляции необходимых двоичных файлов можно найти в документе "RISC-V - Руководство по началу работы"[5].

Создание пользовательской системы RISC-V

Если вы уже умеете компилировать ядро Linux, QEMU и такие программные комплексы, как BusyBox, возможно, вы захотите сделать еще один шаг вперед. Существует система сборки для создания корневых файловых систем на базе Linux и их эмуляции под названием Yocto Project. В RISC-V есть "слой", который можно использовать для создания полностью собственного дистрибутива Linux. Для получения более подробной информации смотрите meta-riscv[6] на GitHub.

[4] https://riscv.org/learn/

[5] https://risc-v-getting-started-guide.readthedocs.io/en/latest/linux-qemu.html

[6] https://github.com/riscv/meta-riscv

RISC-V Hello World

Обзор окружения

Вот приложение hello world, которое мы будем использовать:

```
-----code-----
# Simple RISC-V Hello World

.global _start

_start: addi  a0, x0, 1
    la    a1, helloworld
    addi  a2, x0, 13
    addi  a7, x0, 64
    ecall

    addi    a0, x0, 0
    addi    a7, x0, 93
    ecall

.data
helloworld:     .ascii "Hello World!\n"
-----end code-----
```

Существует также два способа компиляции этого кода: либо с помощью GCC, либо вызывая "as" и "ld" напрямую:

```
-----code-----
# GCC
riscv64-linux-gnu-gcc -o rv-hello rv-hello.s -nostdlib -static

# AS & LD
riscv64-linux-gnu-as -march=rv64imac -o rv-hello.o rv-hello.s
riscv64-linux-gnu-ld -o rv-hello rv-hello.o
-----end code-----
```