Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б10-мм

Запуск Embox на RISC-V в Supervisor режиме

Фролов Тимофей Сергеевич

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
старший преподаватель кафедры ИАС К. К. Смирнов

Санкт-Петербург
2024

Оглавление

Введение

RISC-V[1] — набирающая популярность открытая процессорная архитектура. В ней есть несколько режимов привилегий: (M (Machine), S (Supervisor), U (User)), в которых может работать исполняемый код. У исполняемого кода может быть или не быть доступ к тем или иным аппаратным функциям в зависимости от режима, в котором он работает.

Согласно стандарту [1], аппаратные потоки процессора запускаются в M режиме, в котором у запущенного кода есть доступ ко всем аппаратным функциям процессора. Далее, запущенный код может перевести поток в другой режим привилегий. Пользовательские приложения обычно запускаются в U режиме, а S режим может быть использован для ядра ОС.

Любая аппаратная реализация RISC-V должна поддерживать M режим, а поддержка S и U режима опциональна, но если система поддерживает S режим — она должна поддерживать U режим [1].

В S режиме у запущенного кода нет доступа к части аппаратных возможностей процессора, которые необходимы многим ОС для корректной работы. Для получения доступа к этой функциональности необходимо взаимодействовать с прошивкой, запущенной в M режиме. OpenSBI [3] (Supervisor Binary Interface) — рекомендованный интерфейс между платформозависимой прошивкой, запущенной в M режиме и операционной системой, запущенной в S режиме.

Embox[1] — операционная система реального времени с открытым исходным кодом. В ОС Embox есть поддержка многих процессорных архитектур (x86, arm, и т.д.), однако поддержка RISC-V ограничена: поддерживается запуск только в M режиме процессора. На некоторых платах производитель предполагает, что операционная система запускается в S режиме. Пример такой платы — Lichee RV 86 panel[2].

Целью учебной практики является добавление в ОС Embox поддержки запуска в S режиме процессора из-под OpenSBI.

[1] https://embox.github.io/

[2] https://wiki.sipeed.com/hardware/en/lichee/RV/86_panel.html

1. Постановка задачи

Целью работы является добавление в ОС Embox поддержки запуска в S режиме процессора с использованием OpenSBI. Для её выполнения были поставлены следующие задачи:

1. определить места в исходном коде ОС Embox, в которых предполагается, что ОС запущена в M режиме;

2. реализовать во всех местах, в которых предполагается, что ОС запущена в M режиме, аналогичную функциональность для S режима.

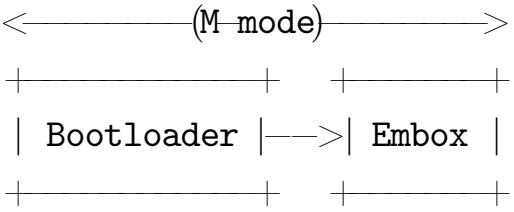
2. Обзор предметной области

В архитектуре RISC-V есть несколько режимов привилегий: M (Machine), S (Supervisor) и U (User). В данной работе имеют значение только M и S режимы.

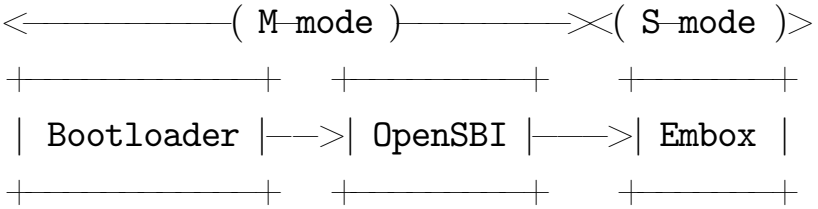
Согласно стандарту [1], в каждом из двух режимов есть инструкции и регистры, которые можно использовать только в этом режиме. Например, есть инструкция mret и инструкция sret. Обе инструкции используются для возврата из обработчика прерывания. Для многих, но не для всех, инструкций и регистров, доступных в M режиме, существуют аналоги для S режима. Однако существуют инструкции и регистры M режима, для которых нет альтернатив в S режиме. Например, регистр mhartid, в котором находится идентификатор текущего аппаратного потока. Также есть отличие в обработке прерываний: по умолчанию все прерывания обрабатываются в M режиме, однако код может перенаправить обработку отдельных прерываний из M режима в S режим. OpenSBI перенаправляет обработку всех прерываний в S режим [3].

На диаграмме 1 проиллюстрирован поддерживаемый ОС Embox процесс запуска в M режиме. На диаграмме 2 показан процесс запуска ОС Embox в S режиме, поддержку которого добавляет данная работа.

Листинг 1: Диаграмма загрузки ОС Embox в M режиме



Листинг 2: Диаграмма загрузки ОС Embox в S режиме (Добавлено в данной работе)



В следующих местах в ОС Embox[3] используется функциональность архитектуры, доступная только в M режиме процессора.

1. Использование инструкций и регистров M режима. В частности, следующие инструкции и регистры не имеют аналогов в S режиме.

   (a) Регистр mhartid — регистр, в котором находится идентификатор текущего аппаратного потока.

   (b) Регистр mtime — регистр, в котором находится текущее значение системного таймера.

   (c) Регистр mtimecmp — регистр, в котором находится значение системного таймера, при достижении которого будет сгенерировано прерывание таймера.

   (d) Регистры mstatus, mie — используются для обработки прерываний. Существуют sstatus и sie, но биты, в которых расположена информация о прерываниях M режима отличается от места, в котором расположена информация о прерываниях S режима.

2. PLIC (Platform-Level Interrupt Controller). Между M режимом и S режимом есть разница в адресах, в которых хранится информация об прерываниях M режима и S режима.

Для получения доступа к функциональности, которую предоставляют инструкции и регистры M режима, для которых есть аналоги, эти инструкции и регистры можно заменить на аналоги. Для получения доступа к функциональности остальных инструкций и регистров программе необходимо взаимодействовать с интерфейсом, предоставляемый системой, запущенной в M режиме. Исключение составляет чтение из регистра mtime, вместо которого можно использовать инструкцию rdtime.

OpenSBI [3] предоставляет системный вызов для записи в регистр mtimecmp, а идентификатор текущего аппаратного потока передаёт при старте программы в регистре a0.

[3] https://github.com/embox/embox

В PLIC [2] отображенные области памяти разделены на контексты, где контекст — это конкретный режим привилегий на конкретном аппаратном потоке процессора. Документация определяет, в каком месте в памяти находятся контексты, но не определяет соответствие между номером контекста и аппаратными потоками с конкретными режимами привилегий, а оставляет определение этого соответствия реализации.

3. Выбор технологии

Тестирование проводится с использованием эмулятора QEMU, потому что для запуска на реальной плате, помимо поддержки S режима, требуется наличие драйверов устройств (как минимум UART), находящихся на плате.

Выбрана платформа virt[4], являющаяся виртуальной реализацией RISC-V без привязки к конкретной реальной плате.

[4]https://www.qemu.org/docs/master/system/riscv/virt.html

4. Реализация

1. Добавлен конфигурационный параметр для сборки в S режим процессора, при включении которого:
   (a) Все инструкции и регистры M режима, для которых есть аналогичные инструкции и регистры в S режиме, заменены на аналоги.
   (b) Идентификатор аппаратного потока получается при старте ОС от OpenSBI.
   (c) Для настройки таймера используется системный вызов OpenSBI.
2. Добавлены параметры конфигурации, определяющие области памяти, по которым необходимо взаимодействовать с PLIC.
3. Создана конфигурация riscv64/qemu-smode для сборки Embox под QEMU virt для запуска в S режиме. В качестве основы конфигурации выступает существующая конфигурация riscv64/qemu. Отличие заключается в адресах для PLIC и использовании S режима вместо M режима.

5. Тестирование

Embox собирается компилятором riscv64-unknown-linux-gnu-gcc версии 13.2.0. Команда сборки:

```
make confload-riscv64/qemu-smode; make
```

Запуск проводится на платформе virt эмулятора QEMU версии 8.1.1. Команда запуска операционной системы в S режиме:

```
qemu-system-riscv64 -nographic -M virt -kernel embox
```

При запуске успешно прошёл стандартный тест Embox на работу точек останова. На рисунках 1 и 2 показан вывод в консоль при запуске ОС Embox.

```
[tima@fedora] edu/embox2 ᚠ master $ qemu-system-riscv64 -nographic -M virt -kernel build/base/bin/embox

OpenSBI v1.3.1
   ____                    _____ ____ _____
  / __ \                  / ____|  _ \_   _|
 | |  | |_ __   ___ _ __ | (___ | |_) || |
 | |  | | '_ \ / _ \ '_ \ \___ \|  _ < | |
 | |__| | |_) |  __/ | | |____) | |_) || |_
  \____/| .__/ \___|_| |_|_____/|___/_____|
        | |
        |_|

Platform Name             : riscv-virtio,qemu
Platform Features         : medeleg
Platform HART Count       : 1
Platform IPI Device       : aclint-mswi
Platform Timer Device     : aclint-mtimer @ 10000000Hz
Platform Console Device   : uart8250
Platform HSM Device       : ---
Platform PMU Device       : ---
Platform Reboot Device    : sifive_test
Platform Shutdown Device  : sifive_test
Platform Suspend Device   : ---
Platform CPPC Device      : ---
Firmware Base             : 0x80000000
Firmware Size             : 194 KB
Firmware RW Offset        : 0x20000
Firmware RW Size          : 66 KB
Firmware Heap Offset      : 0x28000
Firmware Heap Size        : 34 KB (total), 2 KB (reserved), 9 KB (used), 22 KB (free)
Firmware Scratch Size     : 4096 B (total), 760 B (used), 3336 B (free)
Runtime SBI Version       : 1.0

Domain0 Name              : root
Domain0 Boot HART         : 0
Domain0 HARTs             : 0*
Domain0 Region00          : 0x0000000002000000-0x000000000200ffff M: (I,R,W) S/U: ()
Domain0 Region01          : 0x0000000080000000-0x000000008001ffff M: (R,X) S/U: ()
Domain0 Region02          : 0x0000000080020000-0x000000008003ffff M: (R,W) S/U: ()
Domain0 Region03          : 0x0000000000000000-0xffffffffffffffff M: (R,W,X) S/U: (R,W,X)
Domain0 Next Address      : 0x0000000084000000
Domain0 Next Arg1         : 0x0000000087e00000
Domain0 Next Mode         : S-mode
Domain0 SysReset          : yes
Domain0 SysSuspend        : yes

Boot HART ID              : 0
Boot HART Domain          : root
Boot HART Priv Version    : v1.12
Boot HART Base ISA        : rv64imafdch
Boot HART ISA Extensions  : time,sstc
Boot HART PMP Count       : 16
Boot HART PMP Granularity : 4
Boot HART PMP Address Bits: 54
Boot HART MHPM Count      : 16
Boot HART MIDELEG         : 0x0000000000001666
Boot HART MEDELEG         : 0x0000000000f0b509
```

Рис. 1: Пример запуска Embox под QEMU. Показан процесс запуска и использование встроенных команд ls и cat. Часть 1

```
Interrupt controller: riscv_plic

Embox kernel start
        unit: initializing embox.mem.phymem:
                        start=0x00000000848f0000 end=0x00000000a4000000 size=527499264
                 done
runlevel: init level is 0
        unit: initializing embox.mem.static_heap: done
        unit: initializing embox.kernel.task.task_resource: done
        unit: initializing embox.kernel.task.task_table: done
        unit: initializing embox.kernel.task.kernel_task: done
        unit: initializing embox.kernel.sched.sched: done
runlevel: init level is 1
        unit: initializing embox.fs.buffer_cache: done
        unit: initializing embox.arch.riscv.kernel.interrupt: done
        unit: initializing embox.mem.slab: done
        unit: initializing embox.driver.clock.riscv_clk: done
        unit: initializing embox.driver.tty.serial: done
        test: running embox.lib.breakpoint_test.sw_breakpoint_test . done
        unit: initializing embox.fs.rootfs_oldfs: done
runlevel: init level is 2
runlevel: init level is 3
Default IO device[ttyS0]
>export PWD=/
>export HOME=/
>tish
root@embox:/#ls
 ./adduser.conf
 ./group
 ./hosts
 ./passwd
 ./shadow
 ./dev
root@embox:/#cat passwd
root:x:0:0:root:/:/bin/sh
smac_administrator:x:1:0::/:/bin/sh
user:x:1000:1000:First proud embox user:/:/bin/sh
guest:x:1001:1001:guest:/:/bin/false
unclassified:x:1002:1002:unclassified:/:/bin/false
service:x:1003:1003:service:/:/bin/false
secret:x:1004:1004:secret:/:/bin/false
confidentially:x:1005:1005:confidentially:/:/bin/false
root@embox:/#
```

Рис. 2: Пример запуска Embox под QEMU. Показан процесс запуска и использование встроенных команд ls и cat. Часть 2

Заключение

Были достигнуты следующие результаты по поставленным задачам.

- Определены все места в исходном коде ОС Embox, в которых предполагается, что ОС запущена в M режиме.
- Во всех местах, в которых предполагается, что ОС запущена в M режиме, добавлены альтернативные реализации для S режима. Выбор между компиляцией для M режима и для S режима управляется конфигурационным параметром.

По итогу учебной практики в Embox добавлена поддержка запуска в Supervisor режиме. Добавленная функциональность проверена с использованием QEMU. Создан Pull Request[5] в проект Embox, впоследствии принятый в основную ветку проекта.

[5] https://github.com/embox/embox/pull/3039

Список литературы

[1] The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20211203 / Ed. by Andrew Waterman, Krste Asanović, John Hauser. — 2021. — Dec. — URL: https://github.com/riscv/riscv-isa-manual/releases/tag/Priv-v1.12 (дата обращения: 11 декабря 2023 г.).

[2] RISC-V Platform-Level Interrupt Controller Specification. — 2023. — Mar. — URL: https://github.com/riscv/riscv-plic-spec/releases/tag/1.0.0 (дата обращения: 11 декабря 2023 г.).

[3] RISC-V Supervisor Binary Interface Specification. — 2022. — Mar. — URL: https://github.com/riscv-non-isa/riscv-sbi-doc/releases/tag/v1.0.0 (дата обращения: 11 декабря 2023 г.).