ИТМО

Системы ввода/вывода

Лабораторная работа №1
«Принципы организации ввода/вывода без операционной системы»

Преподаватели
Сергей Быковский
Сергей Табунщик

2025

О лабораторных работах

- **3 или 4** лабораторные работы (в зависимости от успехов)
- Лабораторные будут проводится 4 дня по две пары в каждый день
- **ПЕРВОЕ** занятие – теория и начало работы, **ВТОРОЕ** занятие – выполнение и демонстрация
- Баллы за лабораторные сданные в срок:
  - **Лабораторная работа 1** - 8 баллов
  - **Лабораторная работа 2** – 8 баллов
  - **Лабораторная работа 3** – 14 баллов (в зависимости от успехов может быть разбита на две)
- Сдача всех лабораторных обязательна для допуска к зачету

Тематики лабораторных работ

- Принципы организации ввода/вывода **без операционной системы**
- Основы написания драйверов устройств **с использованием операционной системы** (Linux)
- Изучение протоколов передачи данных между устройствами с использованием простейших последовательных интерфейсов (UART, SPI, I2C)

Лабораторная работа 1

- **Тема:** «Принципы организации ввода/вывода без операционной системы»

- **Цель:** познакомится с принципами организации ввода/выводе без операционной системы на примере компьютерной системы на базе процессора с архитектурой RISC-V и интерфейсом OpenSBI с использованием эмулятора QEMU.

- По для выполнения работы:
  - qemu-system-riscv32
  - RISC-V C/ASM окружение для сборки проектов (на базе LLVM)

Материалы

ИТМО

- Операционная система в 1 000 строках кода
  - https://habr.com/ru/companies/ruvds/articles/874154/ (рус)
  - https://operating-system-in-1000-lines.vercel.app/en/ (eng)
  - https://github.com/nuta/operating-system-in-1000-lines (репозиторий)

- Документация по QEMU RISCV32 по системе virt
  - https://www.qemu.org/docs/master/system/riscv/virt.html

- Спецификация OpenSBI
  - https://github.com/riscv-non-isa/riscv-sbi-doc

Почему RISC-V

ИТМО

- Открытая архитектура
- Понятная и масштабируемая ISA
- Хорошая документация и много литературы в открытом доступе

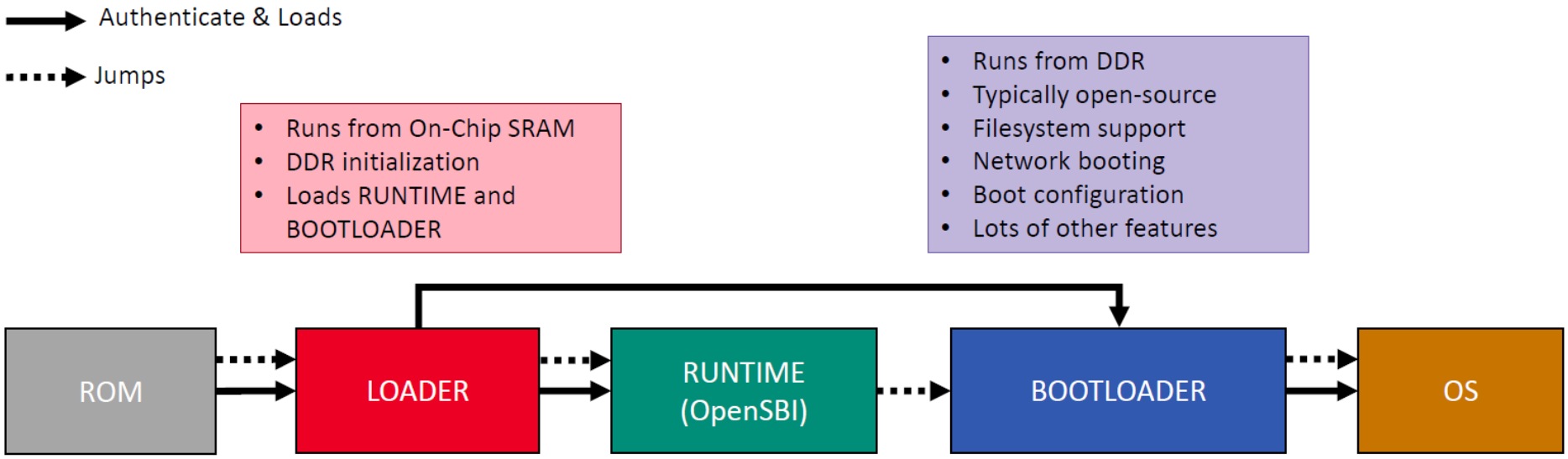
QEMU и система virt

QEMU – открытый проект эмуляции компьютерных систем

Система RISC-V 32 машина virt

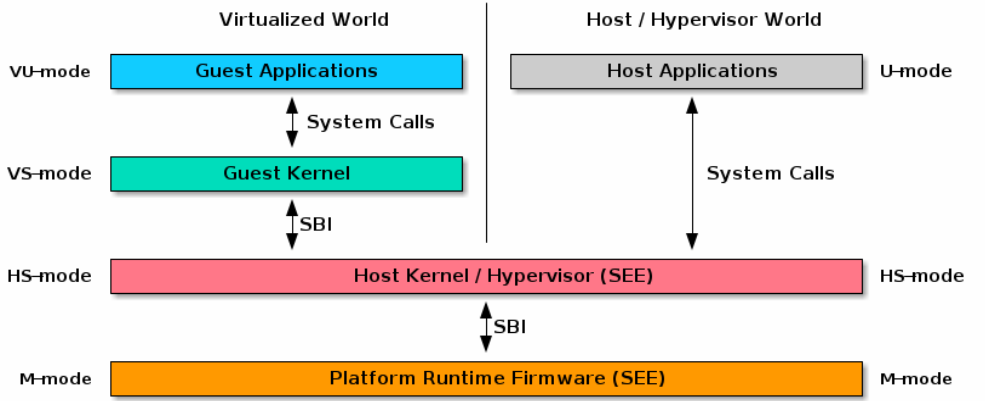
- До 512 ядер RV32GC/RV64GC
- Контроллер прерывания
- NOR Flash-память
- Контроллер UART
- RTC
- Устройства virtio-mmio 8 шт.
- PCIe мост

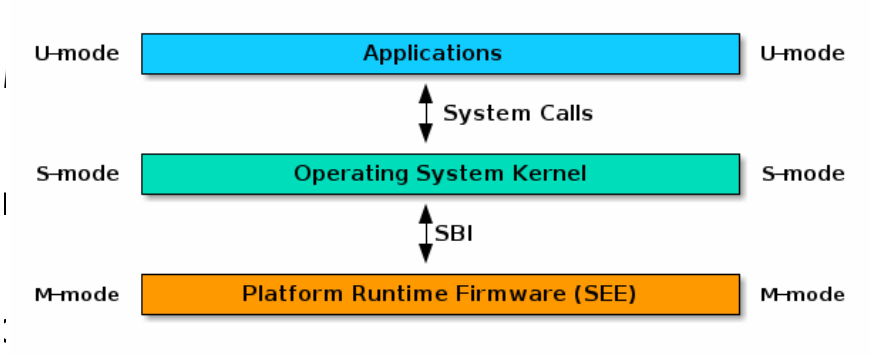
Загрузка компьютерной системы



SBI

- SBI RISC-V (Supervisor Binary Interface) – это соглашение о вызовах в стиле системных вызовов между Supervisor (S-mode OS) и Supervisor Execution Environment (SEE)
- SEE может быть:
  - M-mode RUNTIME firmware для OS в S-mode/Hypervisor в HS-mode
  - HS-mode Hypervisor для Guest OS в VS-mode





SBI

ИТМО

- SBI RISC-V (Supervisor Binary Interface) – это соглашение о вызовах в стиле системных вызовов между Supervisor (S-mode OS) и Supervisor Execution Environment (SEE)
- SEE может быть:
  - M-mode RUNTIME firmware для OS в S-mode/Hypervisor в HS-mode
  - HS-mode Hypervisor для Guest OS в VS-mode
- Вызовы SBI помогают:
  - Уменьшить дублирование кода платформы в разных операционных системах (Linux, FreeBSD и т.д.)
  - Предоставить общие драйверы для ОС, которая может использоваться на нескольких платформах
  - Предоставить интерфейс для прямого доступа к аппаратным ресурсам (M-mode)

OpenSBI

- **Open SBI** (Open Source Supervisor Binary Interface) — это API для ядра ОС, который определяет, какие именно ресурсы системы будут доступны операционной системе.

- В QEMU OpenSBI запускается по умолчанию, выполняет аппаратную инициализацию и загружает ядро для системы RISC-V.

Запуск QEMU (run.sh)

ИТМО

```
#!/bin/bash
set -xue

QEMU=qemu-system-riscv32

# Путь к clang и его флагам
CC=/opt/homebrew/opt/llvm/bin/clang  # Для Ubuntu: используйте CC=clang
CFLAGS="-std=c11 -O2 -g3 -Wall -Wextra --target=riscv32 -ffreestanding -nostdlib"

# Сборка ядра
$CC $CFLAGS -Wl,-Tkernel.ld -Wl,-Map=kernel.map -o kernel.elf \
    kernel.c

# Запуск QEMU
$QEMU -machine virt -bios default -nographic -serial mon:stdio --no-reboot \
    -kernel kernel.elf
```

Скрипт компановщика (kernel.ld)

```
ENTRY(boot)

SECTIONS {
    . = 0x80200000;

    .text :{
        KEEP(*(.text.boot));
        *(.text .text.*);
    }

    .rodata : ALIGN(4) {
        *(.rodata .rodata.*);
    }

    .data : ALIGN(4) {
        *(.data .data.*);
    }

    .bss : ALIGN(4) {
        __bss = .;
        *(.bss .bss.* .sbss .sbss.*);
        __bss_end = .;
    }

    . = ALIGN(4);
    . += 128 * 1024; /* 128KB */
    __stack_top = .;
}
```

- точка входа ядра — функция boot;
- базовый адрес — 0x80200000;
- секция .text.boot всегда размещается в начале;
- все секции размещаются в последовательности .text, .rodata, .data и .bss;
- стек ядра идёт после секции .bss и имеет размер 128 КБ.

Функция boot и точка входа в программу

ИТМО

```
extern char __bss[], __bss_end[], __stack_top[];

void kernel_main(void) {
    for (;;);
}

__attribute__((section(".text.boot")))
__attribute__((naked))
void boot(void) {
    __asm__ __volatile__(
        "mv sp, %[stack_top]\n" // Устанавливаем указатель стека
        "j kernel_main\n"       // Переходим к функции main ядра
        :
        : [stack_top] "r" (__stack_top) // Передаём верхний адрес стека в виде %[stack_top]
    );
}
```

Вызов функций OpenSBI

ИТМО

Все функции SBI используют единую двоичную кодировку, что облегчает смешивание расширений SBI.

- **ECALL** – команда передачи управления между Supervisor и SEE.
- **Регистр a7** – идентификатор расширения SBI (EID).
- **Регистр a6** – идентификатор функции SBI (FID).
- **Регистры a0 - a5** – аргументы для вызова функции SBI. Регистры, которые не определены при вызове функции SBI, не резервируются.
- Все регистры, кроме a0 и a1, должны быть сохранены при вызове SBI вызываемой частью.
- **Регистры a0/a1** – возвращаемые значения функции SBI (код ошибки/значение).

```
struct sbiret {
    long error;
    union {
        long value;
        unsigned long uvalue;
    };
};
```

Описание функции putchar в OpenSBI

5.2. Extension: Console Putchar (EID #0x01)

```
long sbi_console_putchar(int ch)
```

Write data present in **ch** to debug console.

Unlike `sbi_console_getchar()`, this SBI call **will block** if there remain any pending characters to be transmitted or if the receiving terminal is not yet ready to receive the byte. However, if the console doesn't exist at all, then the character is thrown away.

This SBI call returns `0` upon success or an implementation specific negative error code.

Вызов функции OpenSBI

```
struct sbiret sbi_call(long arg0, long arg1, long arg2, long arg3, long arg4,
                       long arg5, long fid, long eid) {
    register long a0 __asm__("a0") = arg0;
    register long a1 __asm__("a1") = arg1;
    register long a2 __asm__("a2") = arg2;
    register long a3 __asm__("a3") = arg3;
    register long a4 __asm__("a4") = arg4;
    register long a5 __asm__("a5") = arg5;
    register long a6 __asm__("a6") = fid;
    register long a7 __asm__("a7") = eid;

    __asm__ __volatile__("ecall"
                         : "=r"(a0), "=r"(a1)
                         : "r"(a0), "r"(a1), "r"(a2), "r"(a3), "r"(a4), "r"(a5),
                           "r"(a6), "r"(a7)
                         : "memory");
    return (struct sbiret){.error = a0, .value = a1};
}

void putchar(char ch) {
    sbi_call(ch, 0, 0, 0, 0, 0, 0, 1 /* Console Putchar */);
}
```

Задание к лабораторной работе

ИТМО

1. Реализовать функцию *putchar* вывода данных в консоль
2. Реализовать функцию getchar для получения данных из консоли
3. На базе реализованных функций *putchar* и *getchar* написать программу, позволяющую вызывать определенные вариантом функции OpenSBI посредством взаимодействия пользователя через меню
4. Запустить программу и выполнить вызов пунктов меню, получив результаты их работы
5. Оформить отчет по работе в электронном формате

Варианты

ИТМО

№ варианта	Пункты меню
1	1. Get SBI specification version 2. Get number of counters 3. Get details of a counter (должно быть возможно задавать номер счетчика) 4. System Shutdown
2	1. Get SBI implementation version 2. Hart get status (должно быть возможно задавать номер ядра) 3. Hart stop 4. System Shutdown

Требования к отчету

ИТМО

1. На титульном листе должны быть приведены следующие данные:
   a. Название дисциплины
   b. Номер и название лабораторной работы
   c. ФИО исполнителя и группа
2. Во введении указываются цели и задачи работы
3. В основной части приводится описание функций *putchar*, *getchar*, а также описывается интерфейс вызова функций OpenSBI, заданных вариантом задания
4. Приводится скриншот вывода в консоль данных при вызове каждого пункта меню.

Спасибо
за внимание!

ITMO more than a UNIVERSITY