ISCAS NIST

循序渐进，学习开发一个 RISC-V 上的操作系统

第 7 章 Hello RVOS

汪辰

2021/4

本章内容

- 系统引导过程
- “Hello，RVOS！”

参考

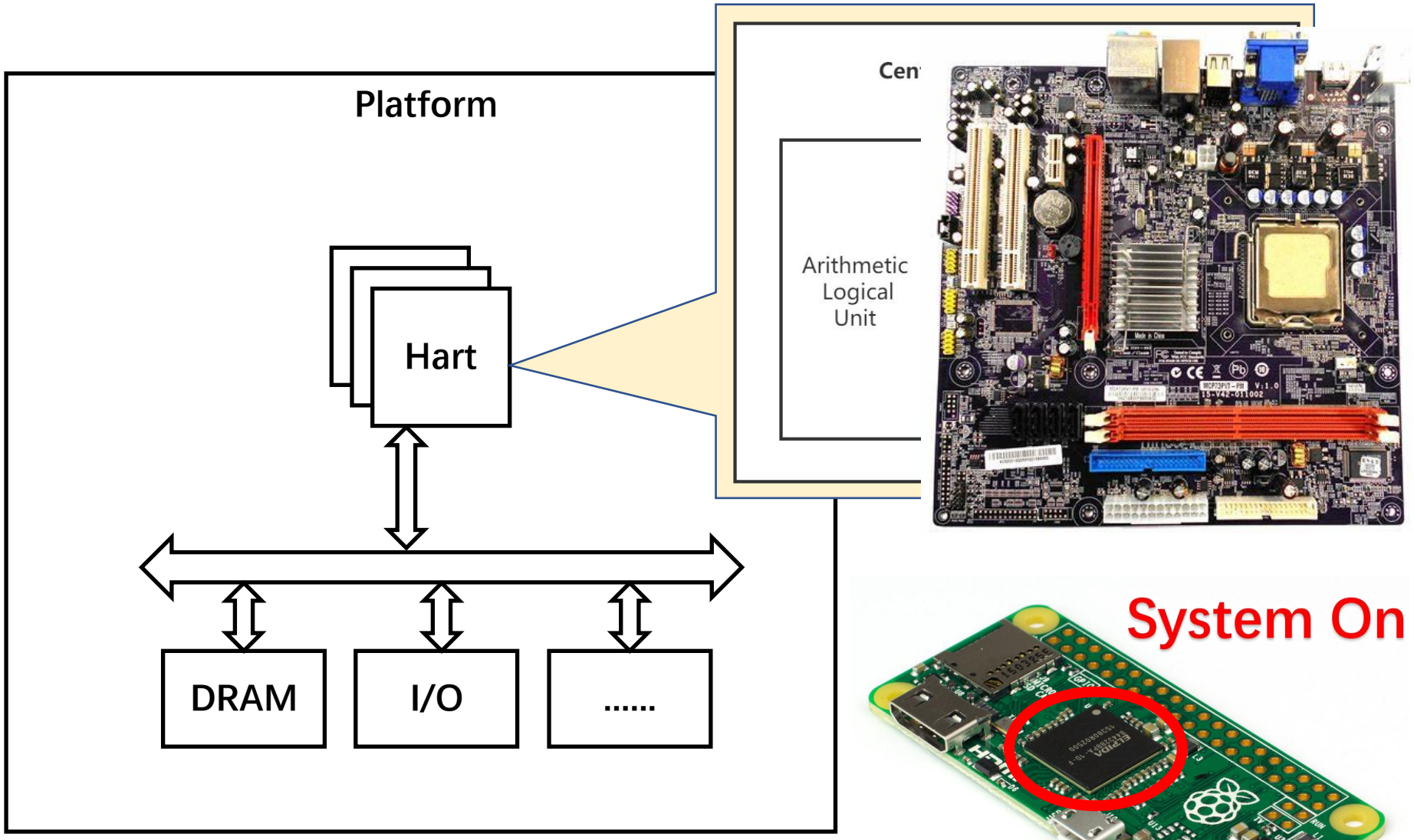
- 【参考 1】：The RISC-V Instruction Set Manual，Volume I: Unprivileged ISA，Document Version 20191213
- 【参考 2】: The RISC-V Instruction Set Manual，Volume II: Privileged Architecture，Document Version 20190608-Priv-MSU-Ratified
- 【参考 3】：TECHNICAL DATA ON 16550：http://byterunner.com/16550.html

本章内容

- **系统引导过程**
  - 硬件的一些基本概念
  - 引导程序要做哪些事情

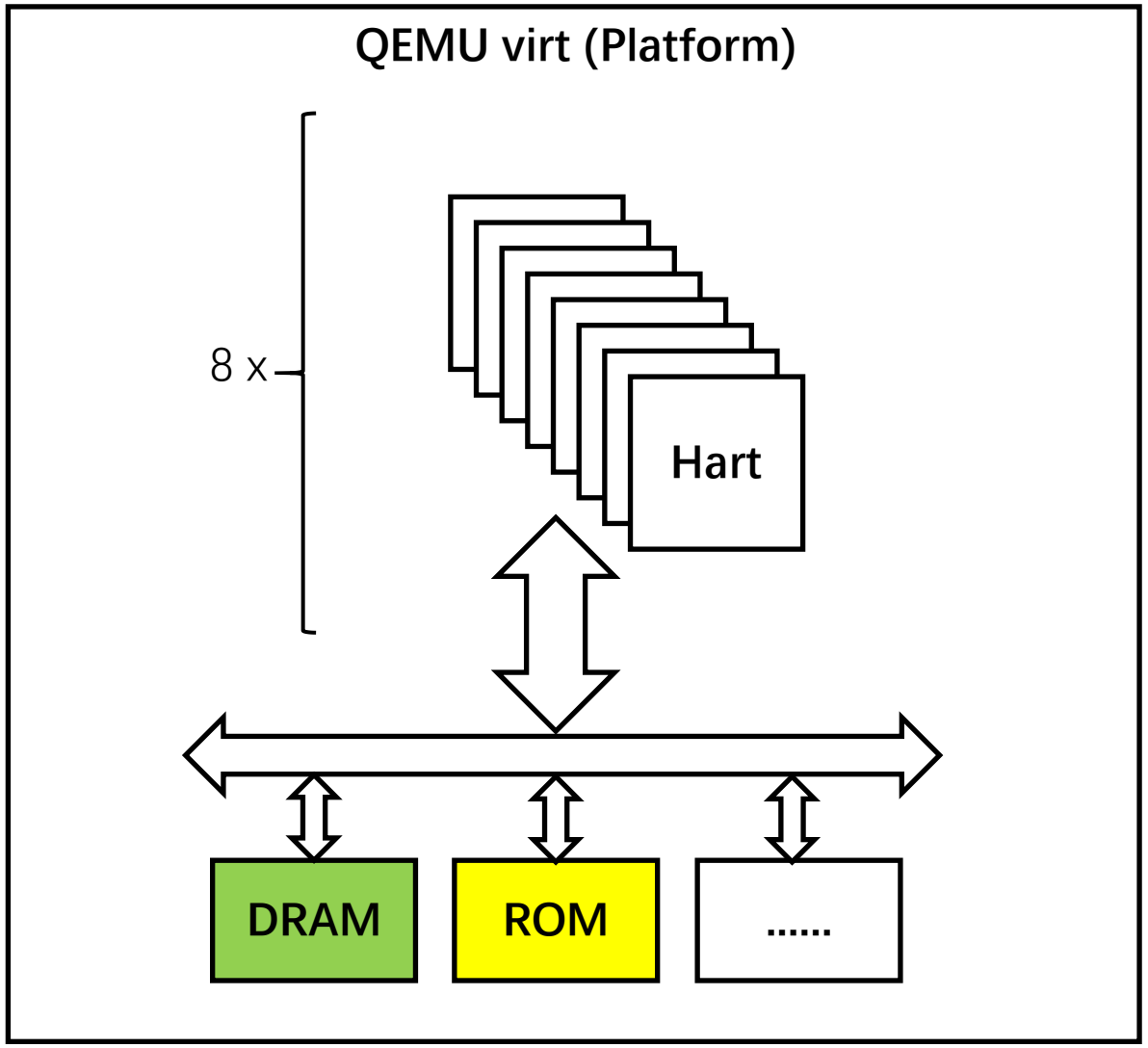
- **“Hello，RVOS！”**

硬件的一些基本概念



System On Chip

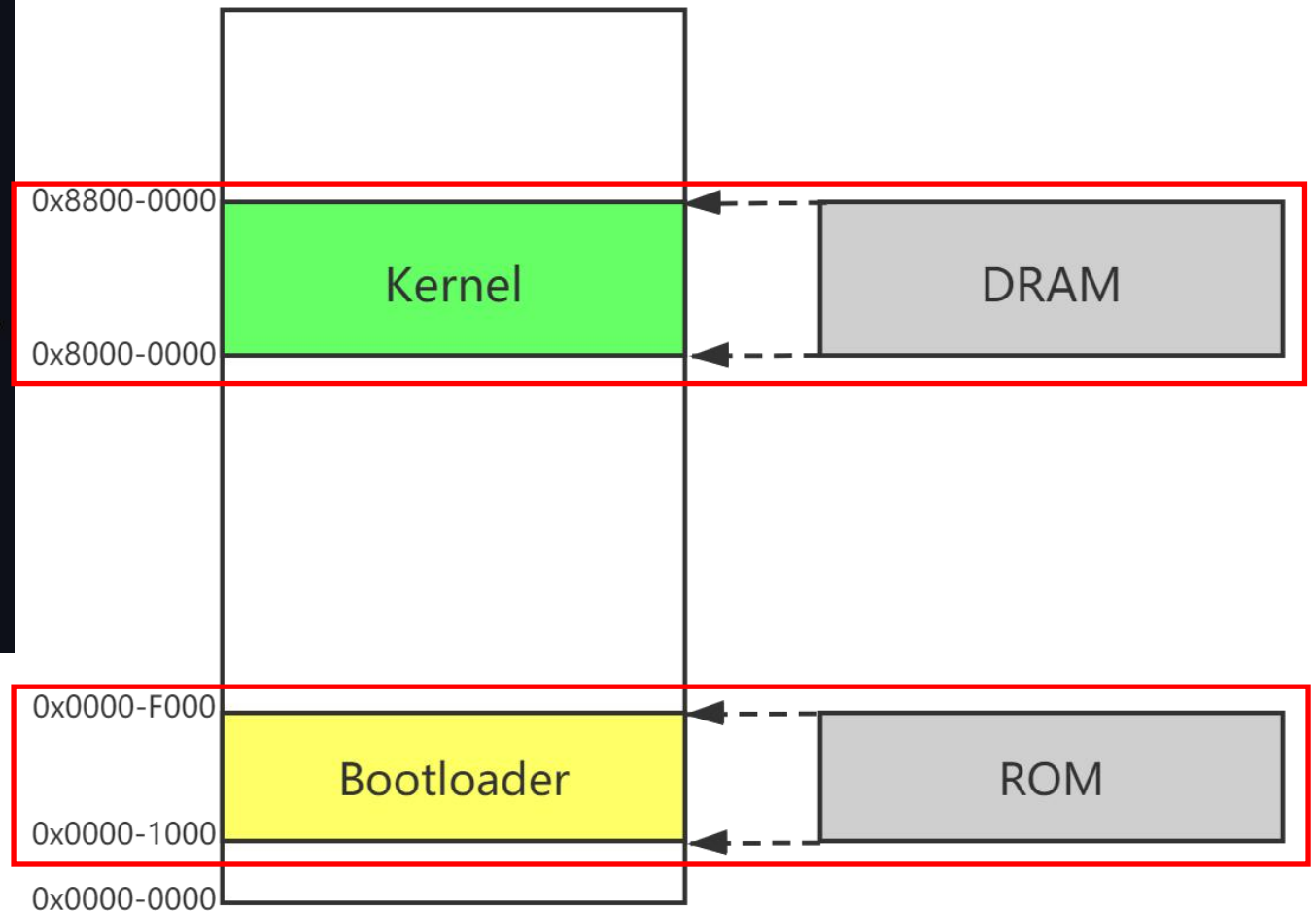
硬件的一些基本概念



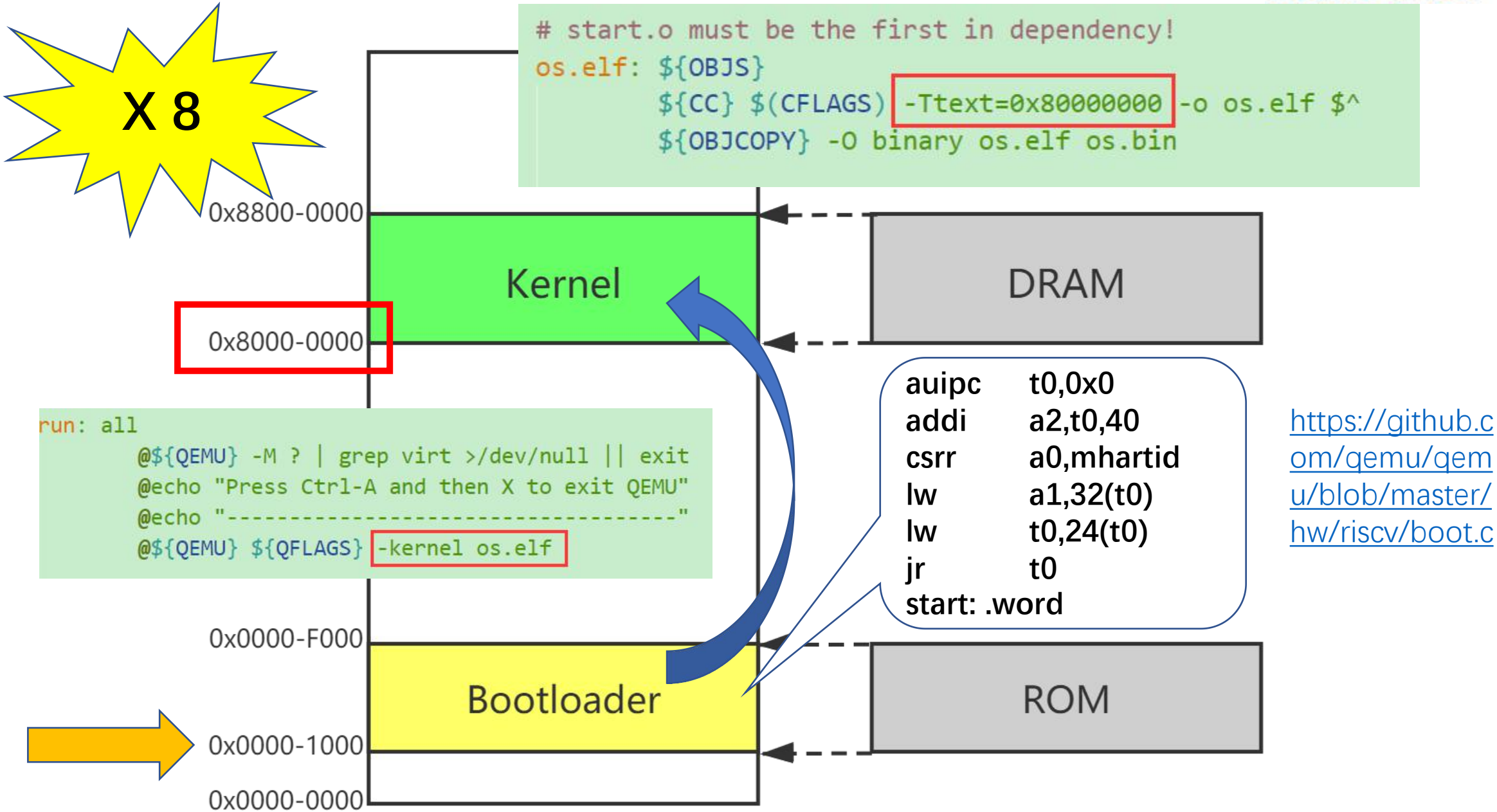
系统引导过程介绍：QEMU-virt 地址映射

https://github.com/qemu/qemu/blob/master/hw/riscv/virt.c

```
static const MemMapEntry virt_memmap[] = {
    [VIRT_DEBUG] =       {        0x0,         0x100 },
    [VIRT_MROM] =        {     0x1000,        0xf000 },
    [VIRT_TEST] =        {   0x100000,        0x1000 },
    [VIRT_RTC] =         {   0x101000,        0x1000 },
    [VIRT_CLINT] =       {  0x2000000,       0x10000 },
    [VIRT_PCIE_PIO] =    {  0x3000000,       0x10000 },
    [VIRT_PLIC] =        {  0xc000000, VIRT_PLIC_SIZE(VIRT_CPUS_MAX * 2) },
    [VIRT_UART0] =       { 0x10000000,         0x100 },
    [VIRT_VIRTIO] =      { 0x10001000,        0x1000 },
    [VIRT_FLASH] =       { 0x20000000,     0x4000000 },
    [VIRT_PCIE_ECAM] =   { 0x30000000,    0x10000000 },
    [VIRT_PCIE_MMIO] =   { 0x40000000,    0x40000000 },
    [VIRT_DRAM] =        { 0x80000000,           0x0 },
};
```

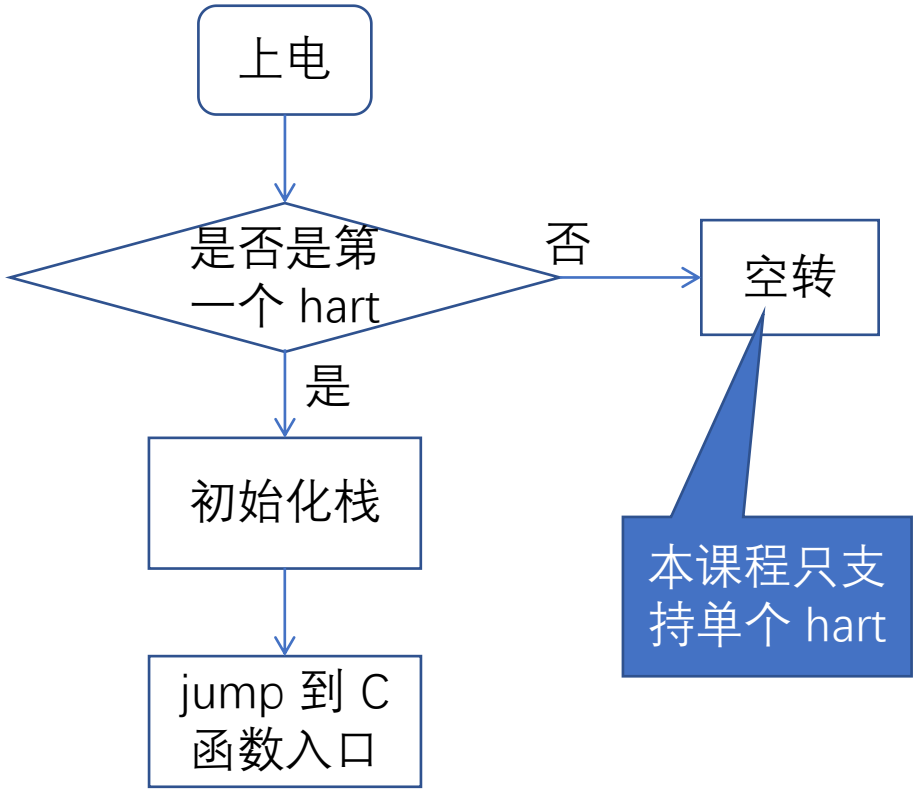


系统引导过程介绍



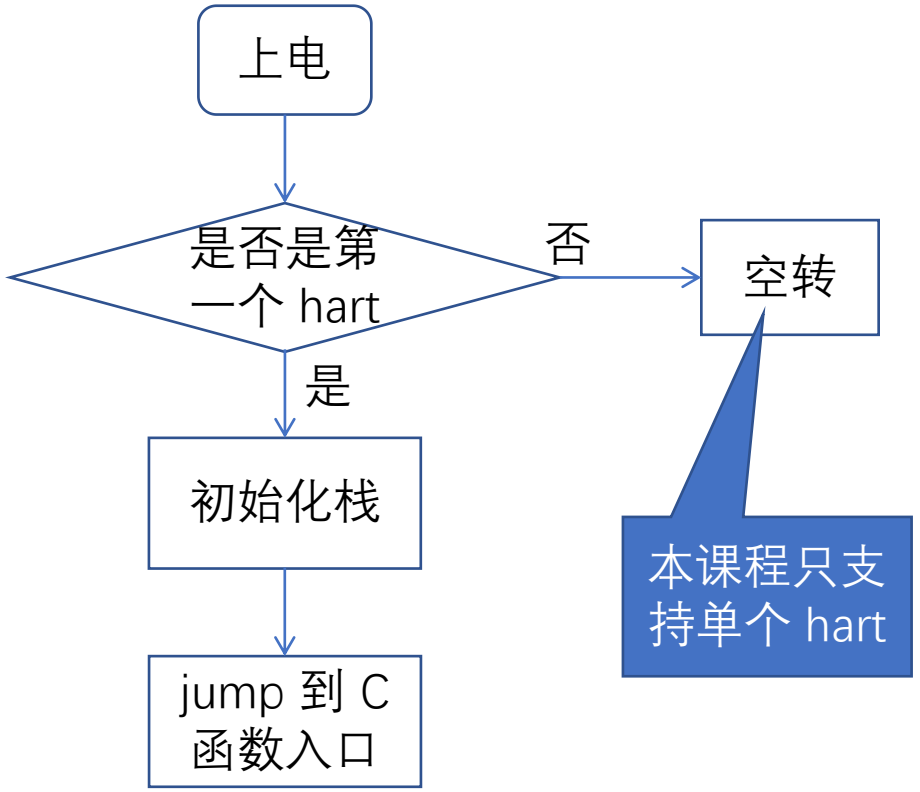
https://github.com/qemu/qemu/blob/master/hw/riscv/boot.c

引导程序要做哪些事情



- 如何判断当前 hart 是不是第一个 hart
- 如何初始化栈
- 如何跳转到 C 语言的执行环境

引导程序要做哪些事情



- 如何判断当前 hart 是不是第一个 hart
- 如何初始化栈
- 如何跳转到 C 语言的执行环境

Control and Status Registers (CSRs)

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	*Reserved*	
3	11	Machine	M

【参考 2】Table 1.1: RISC-V privilege levels.

- 除了所有 Level 下都可以访问的通用寄存器组之外，每个 Level 都有自己对应的一组寄存器。
- 高 Level 可以访问低 Level 的 CSR，反之不可以。
- ISA Specification （“Zicsr”扩展）定义了特殊的 CSR 指令来访问这些 CSR。

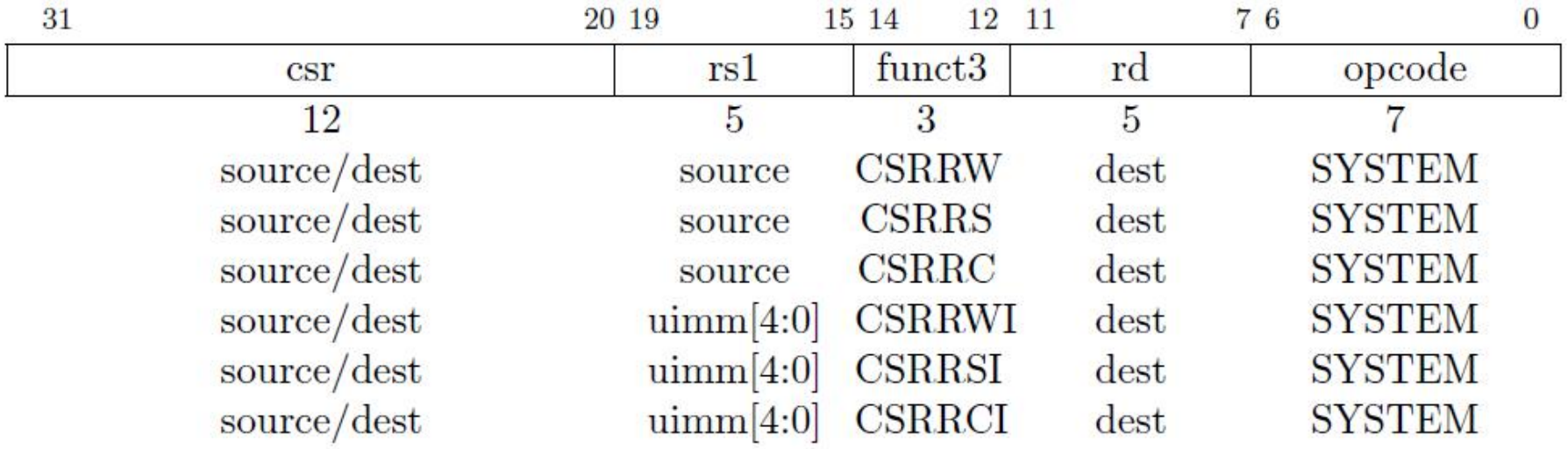
Control and Status Registers (CSRs)

➢ Machine 模式下的 CSR 列表

Number	Privilege	Name	Description
		Machine Information Registers	
0xF11	MRO	mvendorid	Vendor ID.
0xF12	MRO	marchid	Architecture ID.
0xF13	MRO	mimpid	Implementation ID.
0xF14	MRO	mhartid	Hardware thread ID.
		Machine Trap Setup	
0x300	MRW	mstatus	Machine status register.
0x301	MRW	misa	ISA and extensions
0x302	MRW	medeleg	Machine exception delegation register.
0x303	MRW	mideleg	Machine interrupt delegation register.
0x304	MRW	mie	Machine interrupt-enable register.
0x305	MRW	mtvec	Machine trap-handler base address.
0x306	MRW	mcounteren	Machine counter enable.
		Machine Trap Handling	
0x340	MRW	mscratch	Scratch register for machine trap handlers.
0x341	MRW	mepc	Machine exception program counter.
0x342	MRW	mcause	Machine trap cause.
0x343	MRW	mtval	Machine bad address or instruction.
0x344	MRW	mip	Machine interrupt pending.
		Machine Memory Protection	
0x3A0	MRW	pmpcfg0	Physical memory protection configuration.
		⋮	
0x3BF	MRW	pmpaddr15	Physical memory protection address register.

【参考 2】Table 2.4: Currently allocated RISC-V machine-level CSR addresses.

CSR 指令



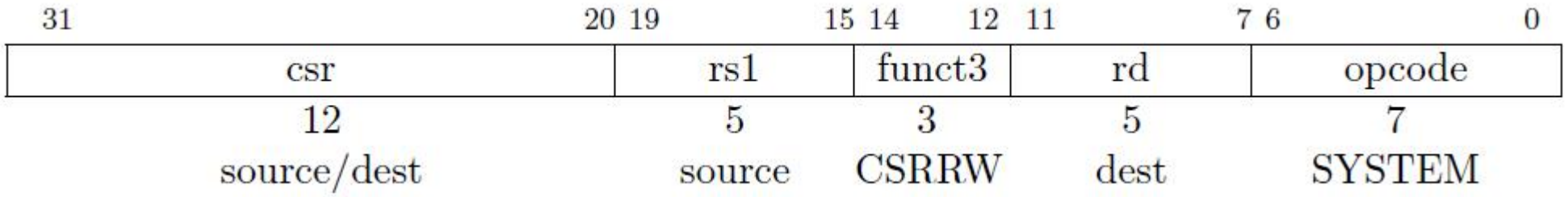
31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
csr	rs1	funct3	rd	opcode
12	5	3	5	7
source/dest	source	CSRRW	dest	SYSTEM
source/dest	source	CSRRS	dest	SYSTEM
source/dest	source	CSRRC	dest	SYSTEM
source/dest	uimm[4:0]	CSRRWI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRSI	dest	SYSTEM
source/dest	uimm[4:0]	CSRRCI	dest	SYSTEM

【参考 1】9.1 CSR Instructions

- **CSRRW (Read/Write CSR)**
- **CSRRS (Read and Set bits in CSR)**
- **CSRRC (Read and Clear bits in CSR)**
- **CSRRWI/CSRRSI/CSRRCI 和以上三个命令的区别是用5 bit 的无符号立即数（zero-extending）代替了 rs1。**
- **opcode 取值为 SYSTEM（值为 1110011）。**

➢ CSRRW（Atomic Read/Write CSR）

语法	CSRRW RD, CSR, RS1	
例子	csrrw t6, mscratch, t6	t6 = mscratch; mscratch = t6



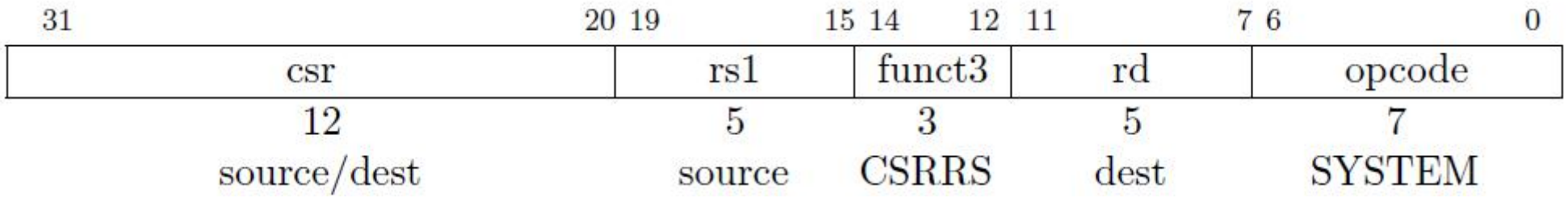
- CSRRW 先读出 CSR 中的值，将其按 XLEN 位的宽度进行“零扩展（zero-extend）”后写入 RD；然后将 RS1 中的值写入 CSR。
- 以上两步操作以“原子性（atomically）”方式完成。
- 如果 RD 是 X0，则不对 CSR 执行读的操作。

pseudoinstruction	Base Instruction	Meaning
`csrw csr, rs`	`csrrw x0, csr, rs`	Write CSR

Table 25.3: RISC-V pseudoinstructions.

➢ CSRRS（Atomic Read and Set Bits in CSR）

语法	CSRRS RD, CSR, RS1	
例子	csrrs x5, mie, x6	x5 = mie; mie |= x6

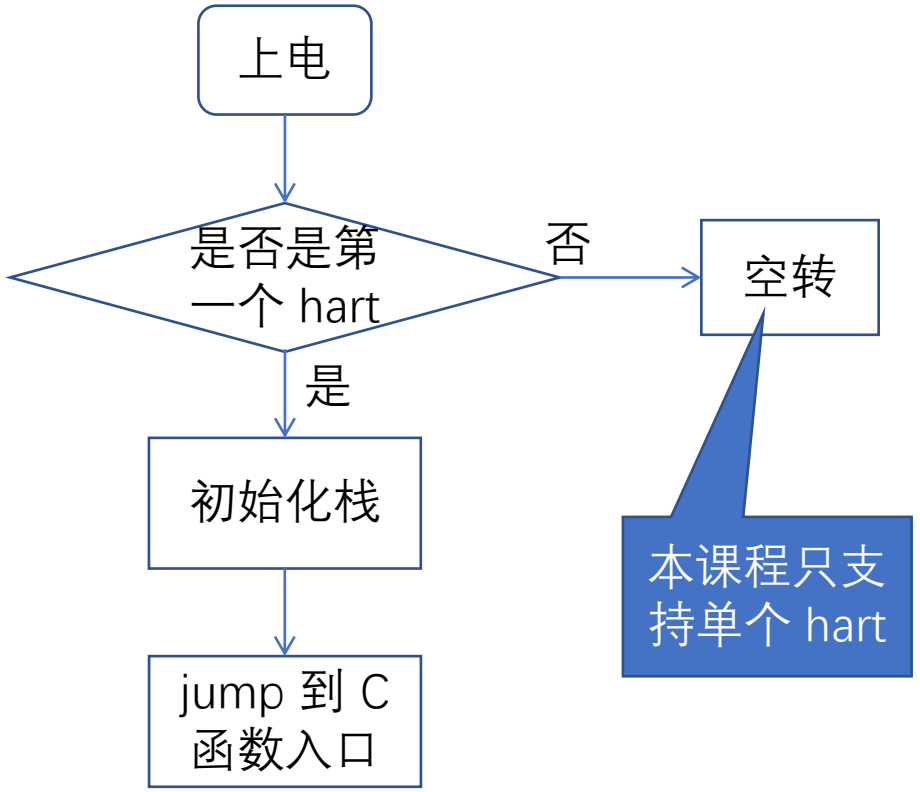


- CSRRS 先读出 CSR 中的值，将其按 XLEN 位的宽度进行“零扩展（zero-extend）”后写入 RD；然后逐个检查 RS1 中的值，如果某一位为 1 则对 CSR 的对应位置 1，否则保持不变。
- 以上两步操作以“原子性（atomically）”方式完成。

pseudoinstruction	Base Instruction	Meaning
`csrr rd, csr`	`csrrs rd, csr, x0`	Read CSR

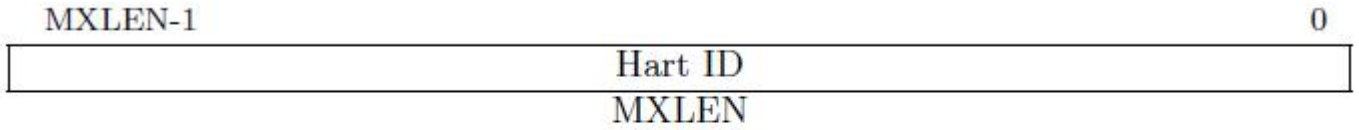
Table 25.3: RISC-V pseudoinstructions.

引导程序要做哪些事情



- 如何判断当前 hart 是不是第一个 hart
- 如何初始化栈
- 如何跳转到 C 语言的执行环境

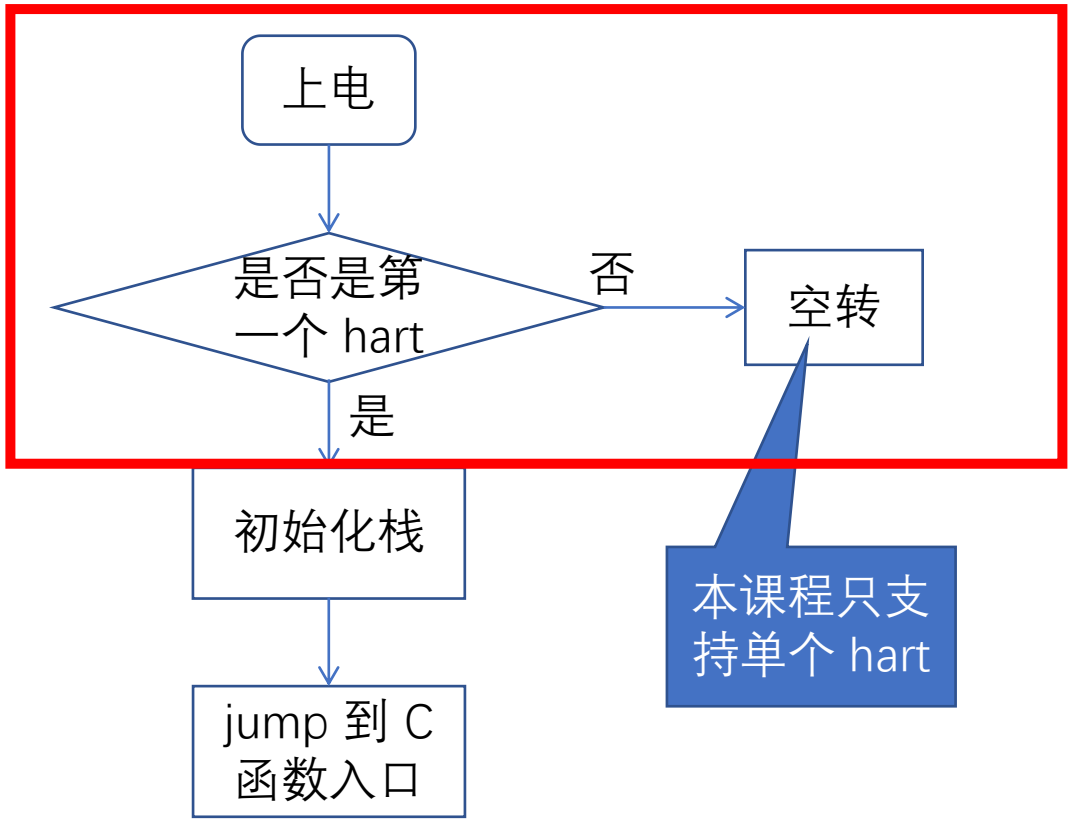
mhartid



【参考 2】Figure 3.5: Hart ID register (mhartid).

- 该 CSR 只读
- 包含了运行当前指令的 hart 的 ID
- 多个 hart 的 ID 必须是唯一的，且必须有一个 hart 的 ID 值为 0 (第一个 hart 的 ID)。

引导程序要做哪些事情

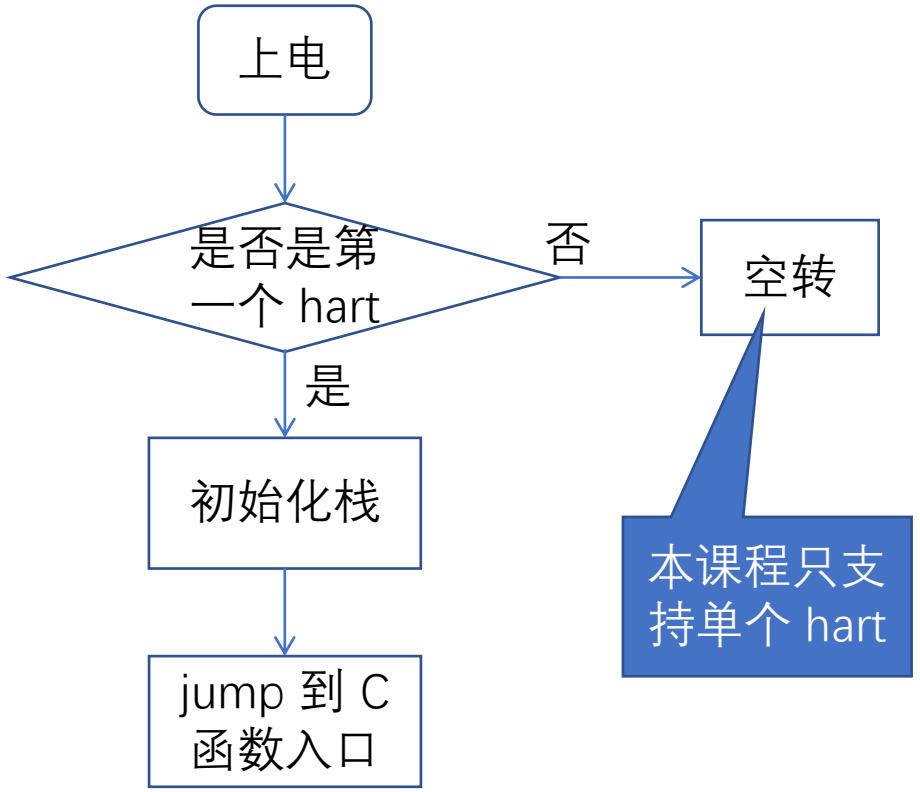



```
_start:
        # park harts with id != 0
        csrr    t0, mhartid
        mv      tp, t0
        bnez    t0, park
```

```
park:
        wfi
        j       park
```

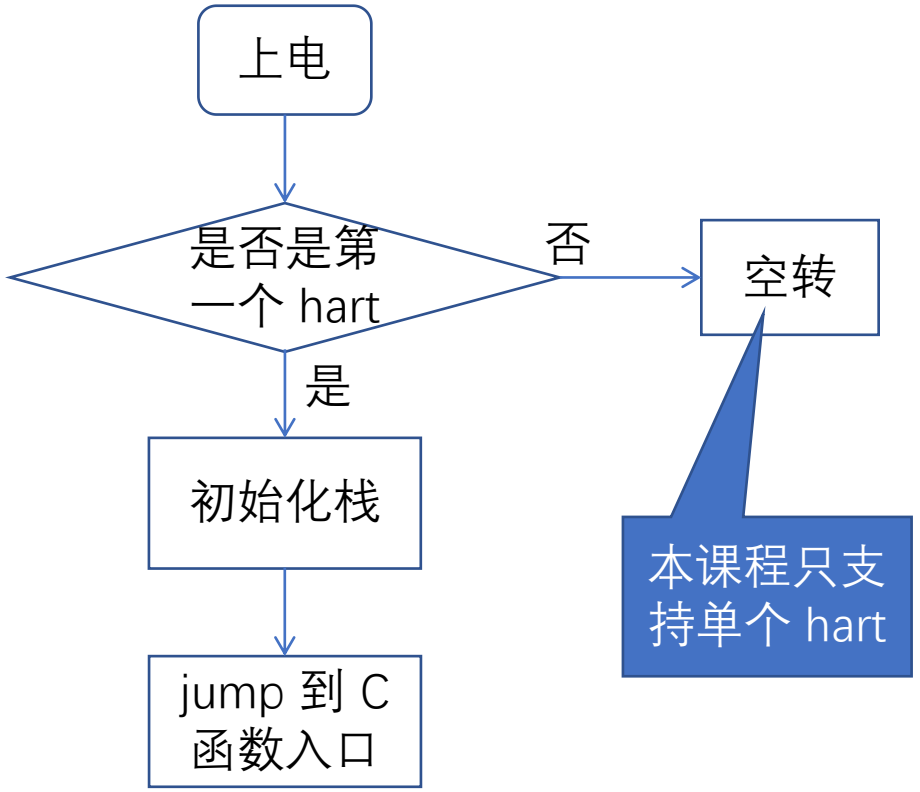
Wait for Interrupt instruction (WFI) 是RISC-V架构定义的一条休眠指令。当处理器执行到 WFI 指令之后，将会停止执行当前的指令流，进入一种空闲状态。这种空闲状态可以被称为“休眠;"状态,直到处理器接收到中断，【参考2】3.2.3

引导程序要做哪些事情



- 如何判断当前 hart 是不是第一个 hart
- **如何初始化栈**
- 如何跳转到 C 语言的执行环境

引导程序要做哪些事情



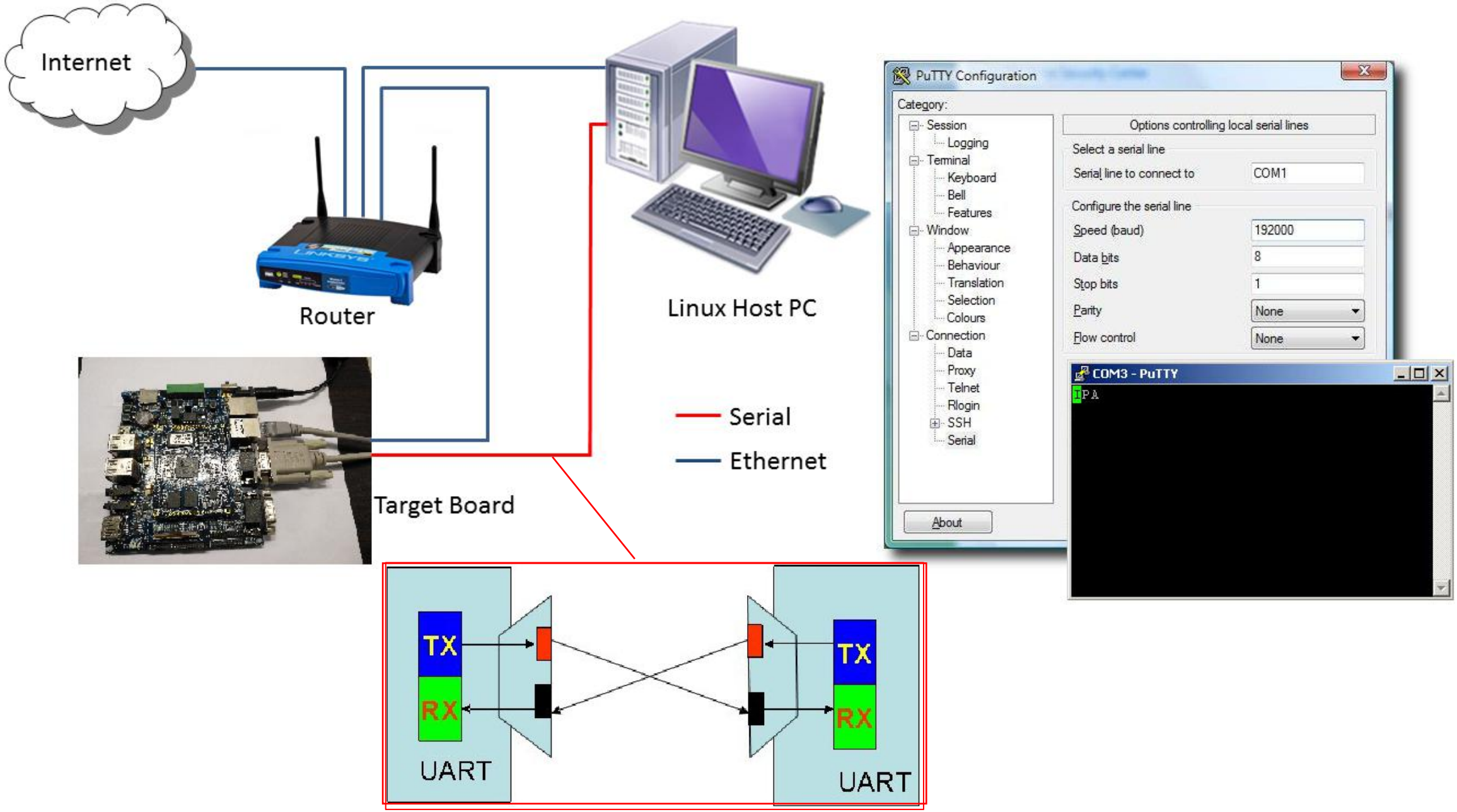
- 如何判断当前 hart 是不是第一个 hart
- 如何初始化栈
- 如何跳转到 C 语言的执行环境

本章内容

- **系统引导过程**

- **“Hello，RVOS！”**
  - UART 的硬件连接方式
  - UART 的 特点
  - UART 的通讯协议
  - NS16550a 编程接口介绍
  - NS16550a 的初始化
  - NS16550a 的数据读写

UART 的硬件连接方式

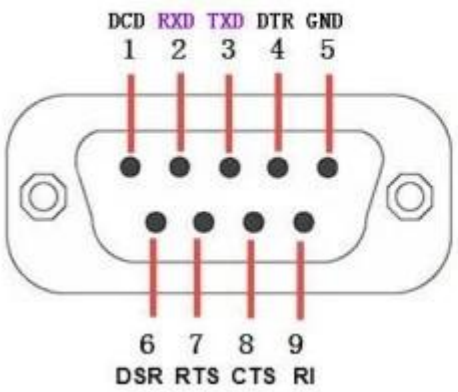


UART 的 特点

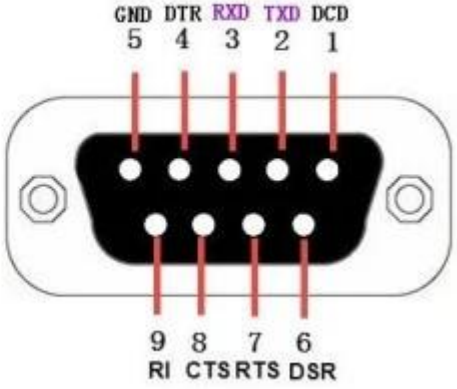
- UART (Universal Asynchronous Receiver and Transmitter)
- **串行**：相对于并行，串行是按位来进行传递，即一位一位的发送和接收。***波特率(baud rate)***，每秒传输的二进制位数，单位为bps(bit per second)。
- **异步**：相对于同步，异步数据传输的过程中，不需要时钟线，直接发送数据，但需要约定通讯协议格式。
- **全双工**：相对于单工和半双工，全双工指可以同时进行收发两方向的数据传递。

UART 的 物理接口

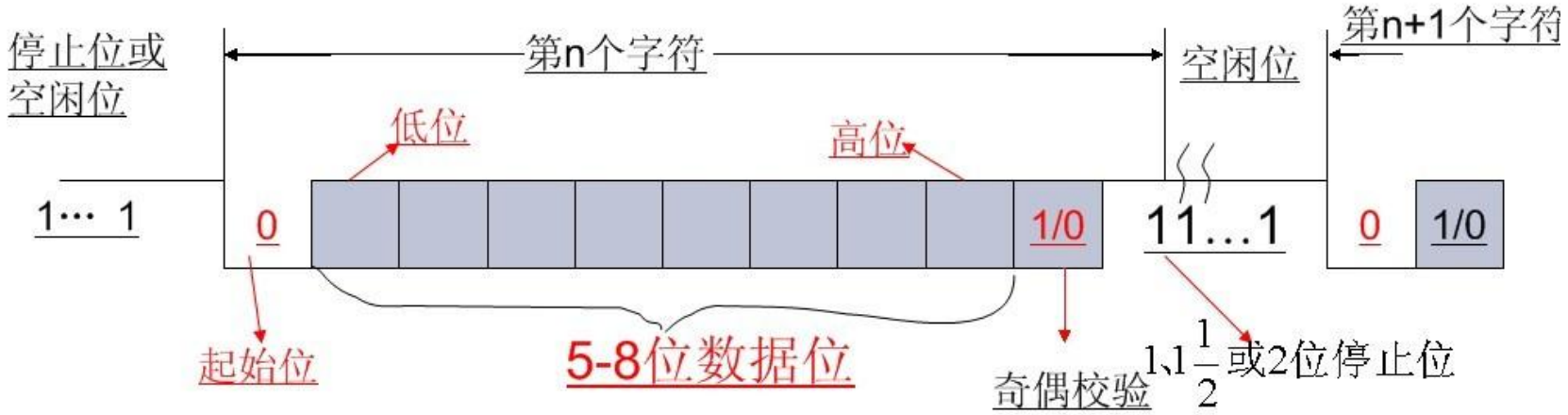
公头



母头



UART 的通讯协议

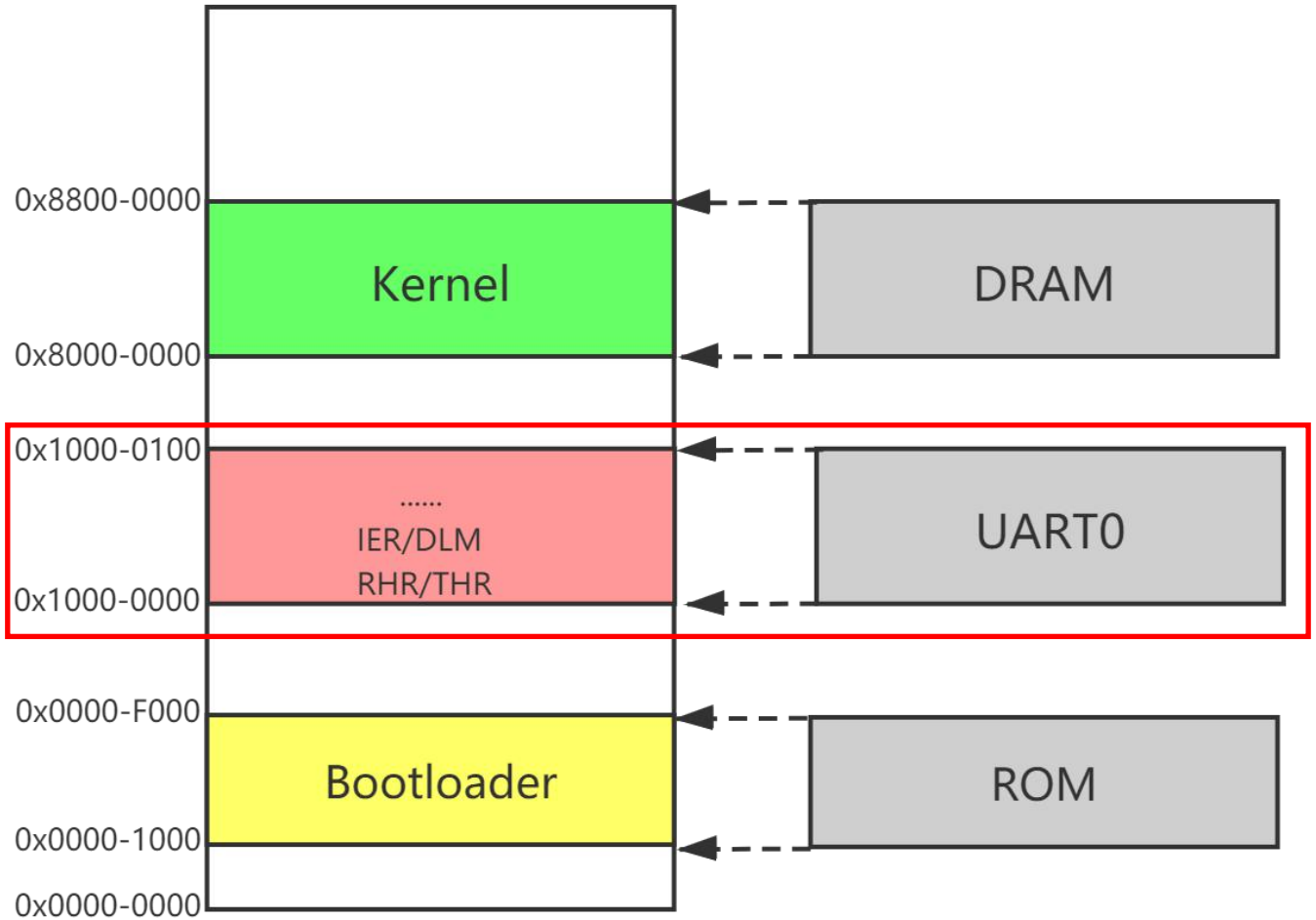


- **空闲位：总线处于空闲状态时信号线的状态为 '1' 即高电平。**
- **起始位：发送方要先发出一个低电平 '0' 来表示传输字符的开始。**
- **数据位：起始位之后就是要传输的数据，数据长度（word length）可以是 5/6/7/8/9 位，构成一个字符，一般都是 8 位。先发送最低位最后发送最高位。**
- **奇偶校验位（parity）：串口校验分几种方式：**
  - ✓ 无校验（no parity）
  - ✓ 奇校验（odd parity）：如果数据位中'1'的数目是偶数，则校验位为'1'，如果'1'的数目是奇数，校验位为'0'。
  - ✓ 偶校验（even parity）：如果数据为中'1'的数目是偶数，则校验位为'0'，如果为奇数，校验位为'1'。
  - ✓ mark parity：校验位始终为1
  - ✓ space parity：校验位始终为0
- **停止（stop）位：数据结束标志，可以是 1位，1.5 位，2 位 的高电平。**

NS16550a 编程接口介绍

https://github.com/qemu/qemu/blob/master/hw/riscv/virt.c

```
static const MemMapEntry virt_memmap[] = {
    [VIRT_DEBUG] =       {        0x0,         0x100 },
    [VIRT_MROM] =        {     0x1000,        0xf000 },
    [VIRT_TEST] =        {   0x100000,        0x1000 },
    [VIRT_RTC] =         {   0x101000,        0x1000 },
    [VIRT_CLINT] =       {  0x2000000,       0x10000 },
    [VIRT_PCIE_PIO] =    {  0x3000000,       0x10000 },
    [VIRT_PLIC] =        {  0xc000000, VIRT_PLIC_SIZE(VIRT_CPUS_MAX * 2) },
    [VIRT_UART0] =       { 0x10000000,         0x100 },
    [VIRT_VIRTIO] =      { 0x10001000,        0x1000 },
    [VIRT_FLASH] =       { 0x20000000,     0x4000000 },
    [VIRT_PCIE_ECAM] =   { 0x30000000,    0x10000000 },
    [VIRT_PCIE_MMIO] =   { 0x40000000,    0x40000000 },
    [VIRT_DRAM] =        { 0x80000000,           0x0 },
};
```

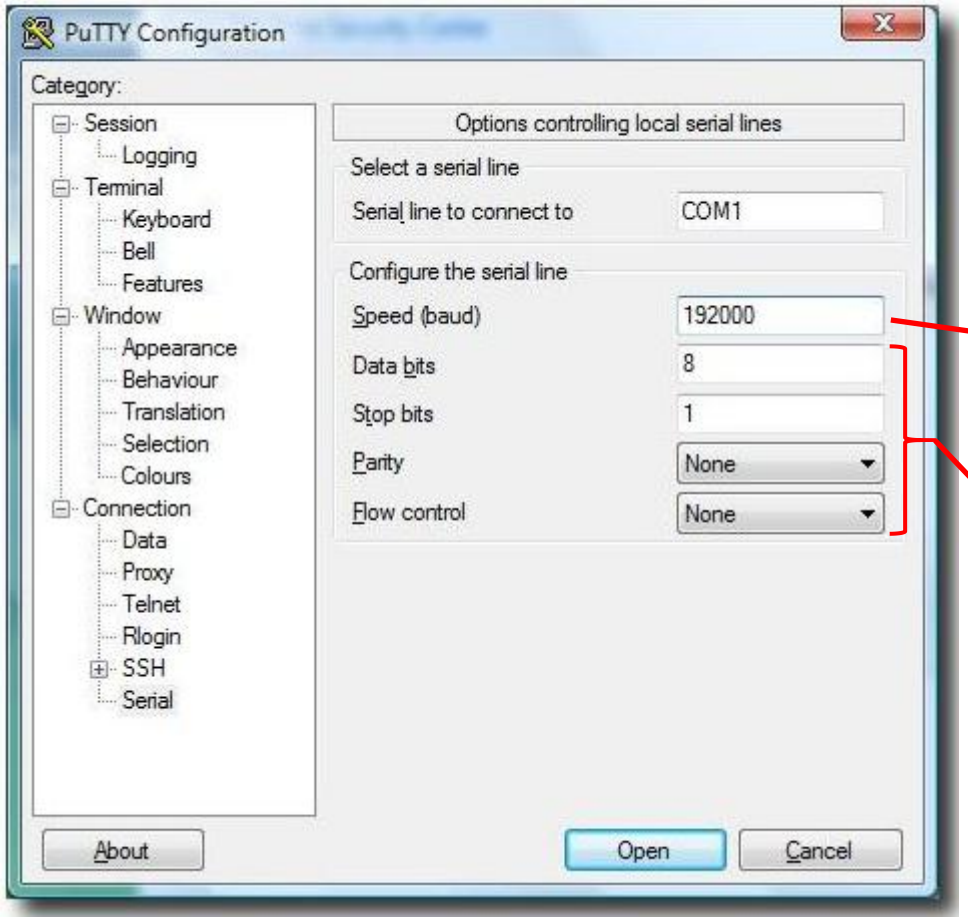


NS16550a 编程接口介绍

A2	A1	A0	READ MODE	WRITE MODE
0	0	0	• Receive Holding Register	• Transmit Holding Register
0	0	0	N/A	• LSB of Divisor Latch when Enabled
0	0	1	N/A	• Interrupt Enable Register
0	0	1	N/A	• MSB of Divisor Latch when Enabled
0	1	0	• Interrupt Status Register	• FIFO control Register
0	1	1	N/A	• Line Control Register
1	0	0	N/A	• Modem Control Register
1	0	1	• Line Status Register	N/A
1	1	0	• Modem Status Register	N/A
1	1	1	• Scratchpad Register Read	• Scratchpad Register Write

【参考 3】PROGRAMMING TABLE

NS16550a 的初始化




```c
void uart_init()
{
        /* disable interrupts. */
        uart_write_reg(IER, 0x00);

        /* Setting baud rate. */
        uint8_t lcr = uart_read_reg(LCR);
        uart_write_reg(LCR, lcr | (1 << 7));
        uart_write_reg(DLL, 0x03);
        uart_write_reg(DLM, 0x00);

        /* Setting communication format */
        lcr = 0;
        uart_write_reg(LCR, lcr | (3 << 0));
}
```

NS16550a 的数据读写

- UART 工作方式为全双工，分发送（TX）和接收（RX）两个独立的方向进行数据传输。

- 对数据的 TX/RX 有两种处理方式：
  - 轮询处理方式
  - 中断处理方式

A2	A1	A0	READ MODE	WRITE MODE
0	0	0	• Receive Holding Register	• Transmit Holding Register
0	0	0	N/A	• LSB of Divisor Latch when Enabled
0	0	1	N/A	• Interrupt Enable Register
0	0	1	N/A	• MSB of Divisor Latch when Enabled
0	1	0	• Interrupt Status Register	• FIFO control Register
0	1	1	N/A	• Line Control Register
1	0	0	N/A	• Modem Control Register
1	0	1	• Line Status Register	N/A
1	1	0	• Modem Status Register	N/A
1	1	1	• Scratchpad Register Read	• Scratchpad Register Write

NS16550a 的数据读写 - putc

A2	A1	A0	READ MODE	WRITE MODE
0	0	0	• Receive Holding Register	• Transmit Holding Register
0	0	0	N/A	• LSB of Divisor Latch when Enabled
0	0	1	N/A	• Interrupt Enable Register
0	0	1	N/A	• MSB of Divisor Latch when Enabled
0	1	0	• Interrupt Status Register	• FIFO control Register
0	1	1	N/A	• Line Control Register
1	0	0	N/A	• Modem Control Register
1	0	1	• Line Status Register	N/A
1	1	0	• Modem Status Register	N/A
1	1	1	• Scratchpad Register Read	• Scratchpad Register Write

```c
#define LSR_TX_IDLE  (1 << 5)
int uart_putc(char ch)
{
        while ((uart_read_reg(LSR) & LSR_TX_IDLE) == 0);
        return uart_write_reg(THR, ch);
}
```

练习 7-1

练习 7-2

谢 谢

欢迎交流合作