操作系统
启动与初始化

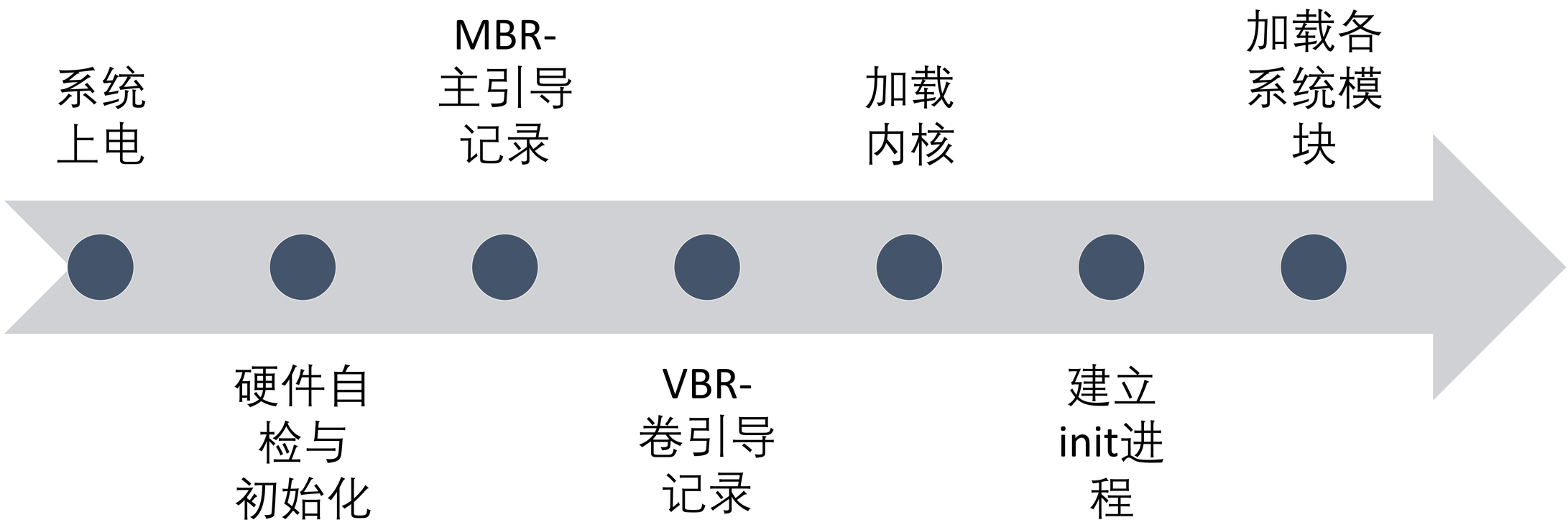
ZJUNIX

目录

系统启动流程

- 传统计算机系统启动流程
- ZJUNIX系统启动流程

传统计算机系统启动流程



ZJUNIX系统启动流程

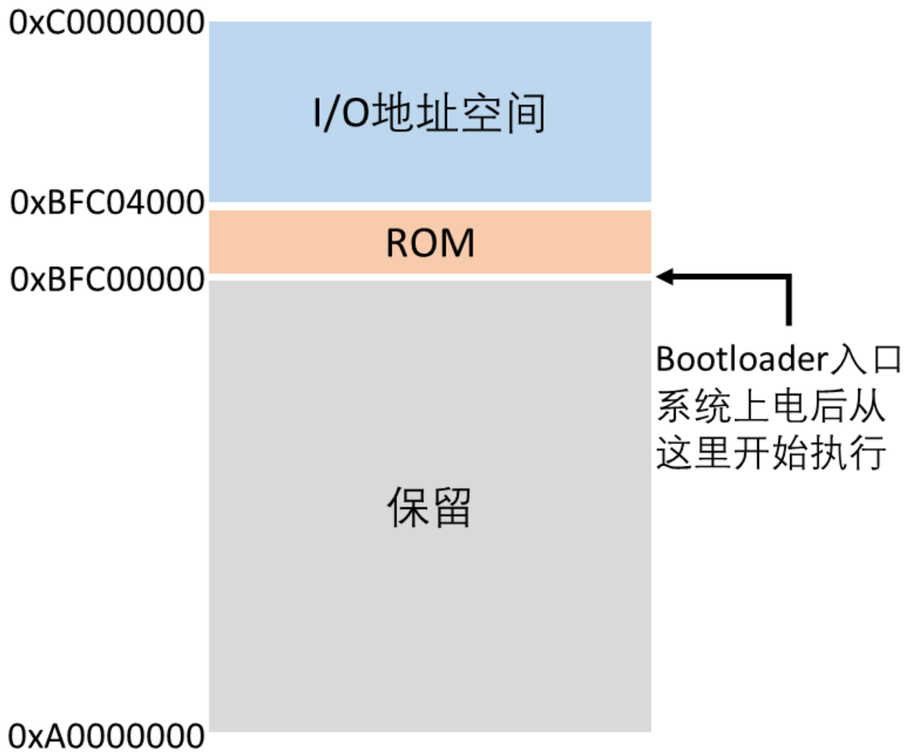
- 跳过MBR和VBR步骤，直接加载内核代码
  - 为了简化启动流程
  - MIPS架构与传统x86架构的区别导致MBR难以存放足够的启动代码

Bootloader设计

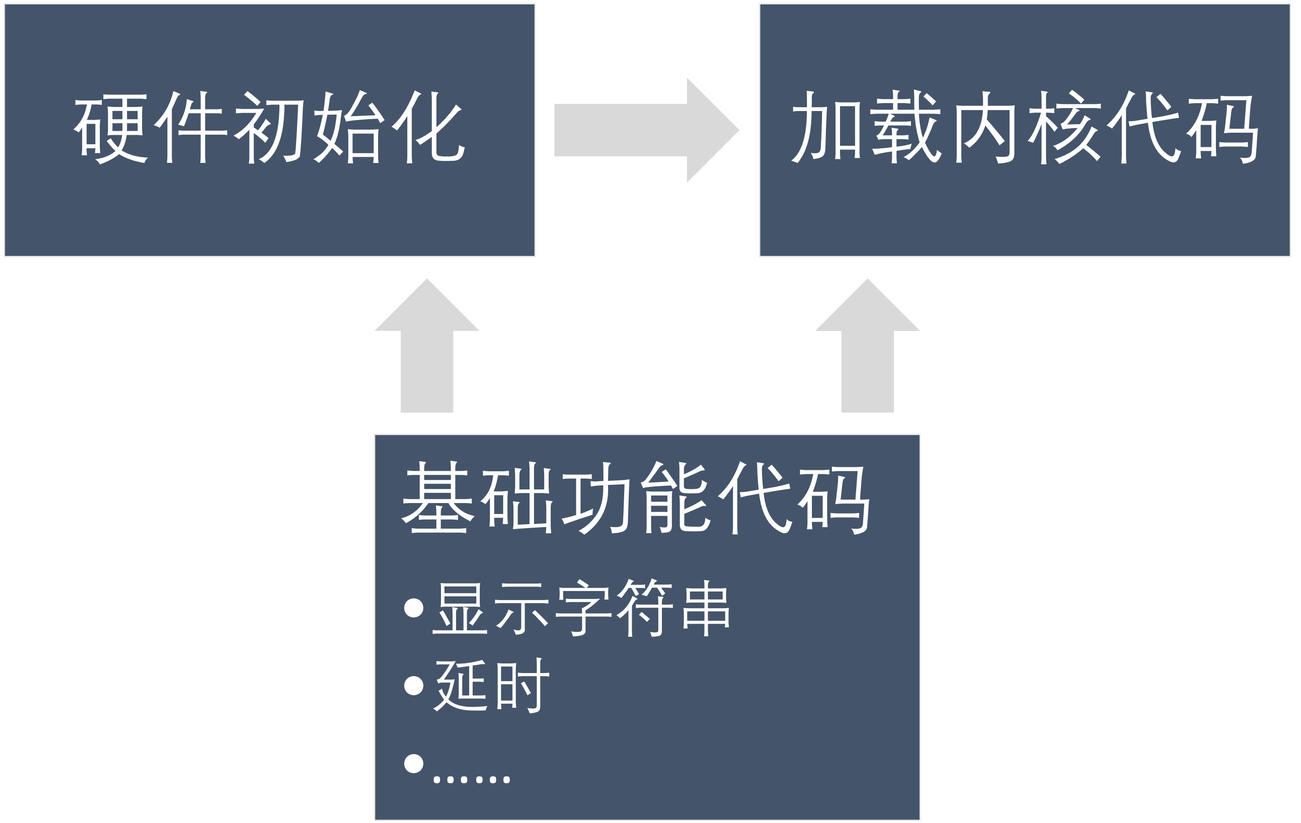
- 功能模块
- 硬件初始化
- 加载内核代码

概述

- Bootloader是固化在ROM里的软件代码
- 是系统上电后启动的第一段程序
- 主要功能：初始化硬件，从SD卡加载内核代码



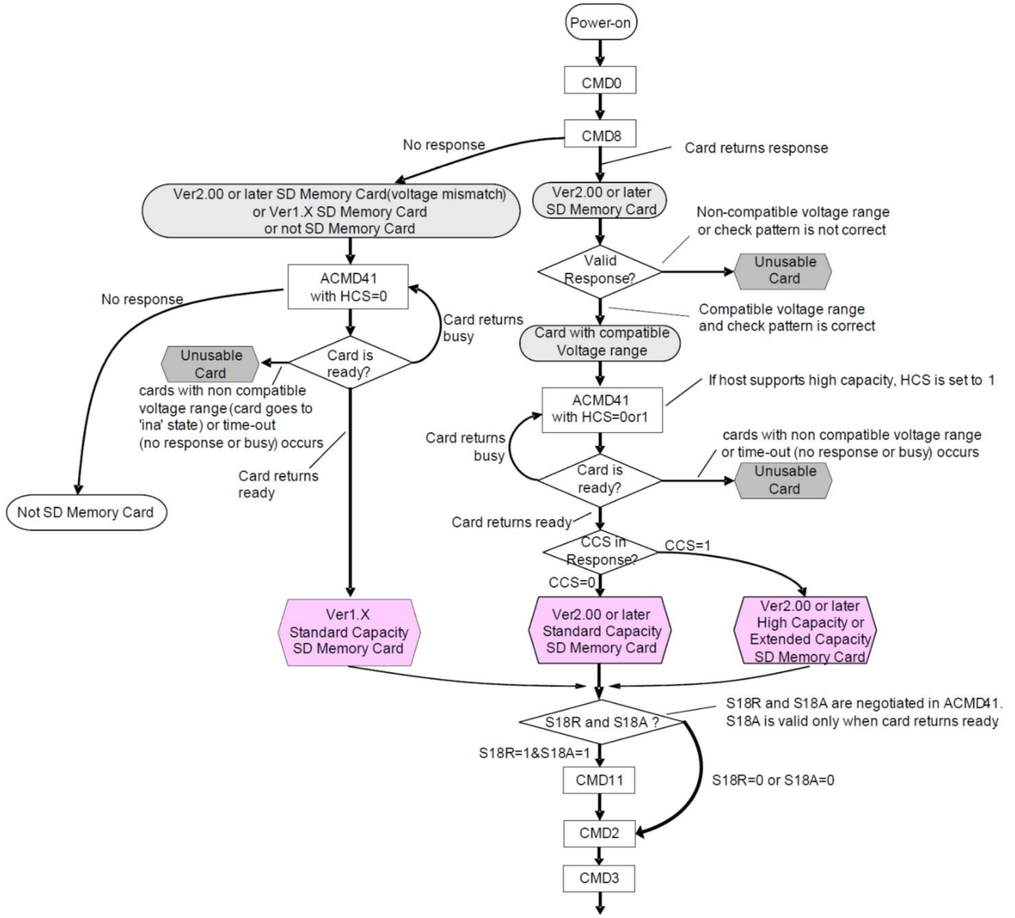
功能模块



ZJUNIX

硬件初始化

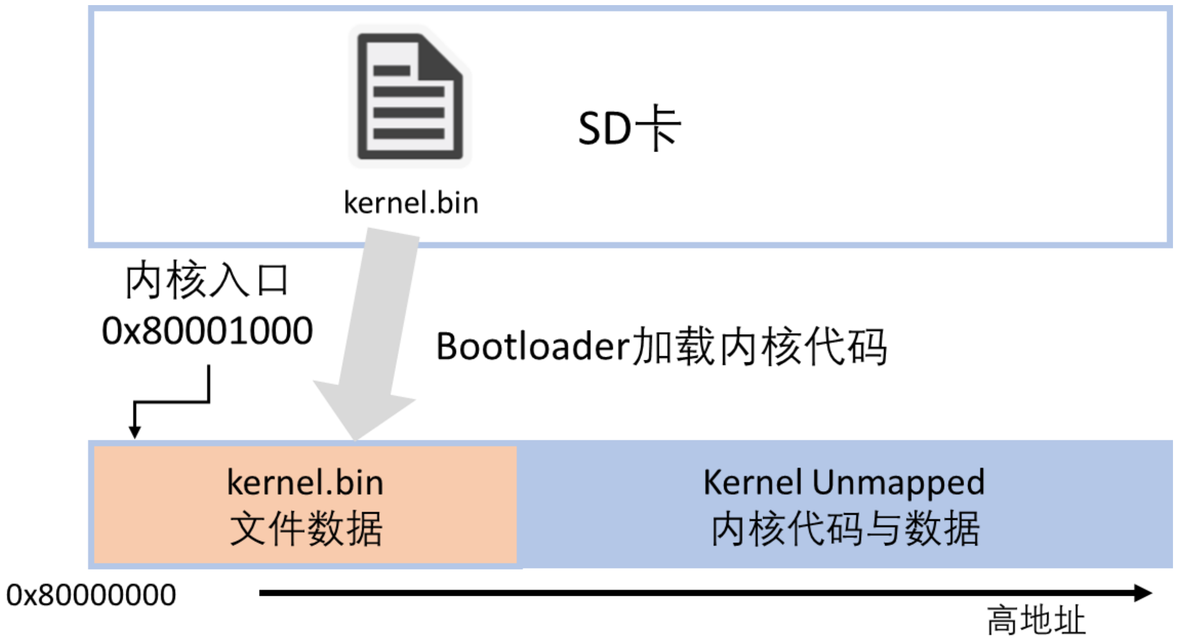
- 初始化CPU各寄存器
  - 通用寄存器清零：
    $at, $v0-$v1, $a0-$a3,
    $t0-t9, $s0-$s7, $k0-$k1,
    $gp, $sp, $fp, $ra
- 初始化SD卡
  - 向SD卡发送一系列指令对其初始化
  - 最终使SD卡工作于4位SD总线模式



SD卡初始化流程图

加载内核代码

- 内核代码文件：SD卡第一个分区名为kernel.bin的文件
- 加载到虚拟地址从0x80000000起始的内存区
  - 对应主存物理地址从0起始的区域
- 内核入口：0x80001000



内核入口与内核初始化

- 内核入口代码
- 内核初始化流程

内核入口代码

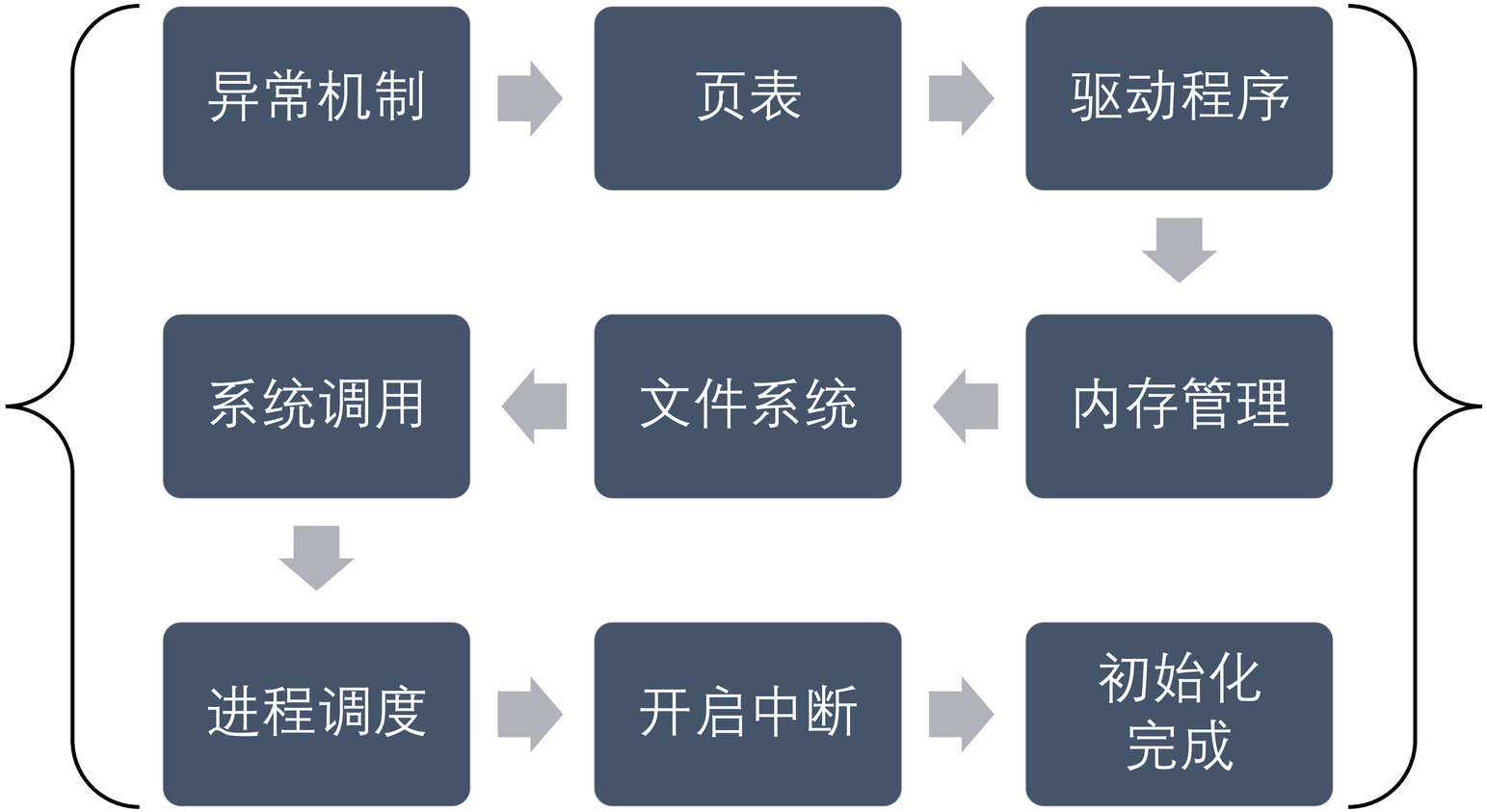
- 内核入口代码：汇编编写
- 设置sp(堆栈指针)及gp(全局指针)寄存器
- 跳转至内核初始化函数
- init_kernel函数调用其他初始化函数完成内核的初始化

```
.org 0x1000
start:
        lui   $sp, 0x8100
        la    $gp, _gp
        j     init_kernel
        nop
```

代码文件：arch/mips32/start.s

内核初始化流程

`void init_kernel()`




```c
void init_kernel() {
    // Exception
    init_exception();
    // Page table
    init_pgtable();
    // Drivers
    init_vga();
    init_ps2();
    // Memory management
    init_bootmm();
    init_buddy();
    init_slab();
    // File system
    init_fs();
    // System call
    init_syscall();
    // Process control
    init_pc();
    create_startup_process();
    // Interrupts
    init_interrupts();
    // Init finished
    machine_info();
```

THANK YOU