硬件视角的操作系统

计算机硬件的状态机模型；

目录

什么是计算机硬件?

计算机硬件 = 数字电路

数字电路模拟器（Logisim）

- 基本构件：wire, reg, NAND
- 每一个时钟周期
  - 先计算 wire 的值
  - 在周期结束时把值锁存至 reg

计算机硬件的状态机模型

不仅是程序，整个计算机系统也是一个状态机

- 状态：内存和寄存器数值
- 初始状态：手册规定（CPU Reset）
- 状态迁移
  - 任意选择一个处理器 cpu
  - 响应处理器外部中断
  - 从 cpu.PC 取指令执行

为了让“操作系统”这个程序能够正确启动，计算机硬件系统必定和程序员之间存在约定-----首先就是 Reset 的状态，然后是 Reset 以后执行的程序应该做什么。

计算机硬件和程序员之间是如何约定的?

Bare-metal 与程序员的约定

Bare-metal 与厂商的约定

- CPU Reset 后的状态（寄存器值）
  - 厂商自由处理这个地址上的值
  - Memory-mapped I/O

厂商为操作系统开发者提供 Firmware

- 管理硬件和系统配置
- 把存储设备上的代码加载到内存
  - 例如存储介质上的第二级 loader（加载器）
  - 或者直接加载操作系统（嵌入式系统）

x86 Family: CPU Reset

9.1.1 Processor State After Reset

Following power-up, The state of control register CR0 is 60000010H (see Figure 9-1). This places the processor is in real-address mode with paging disabled.

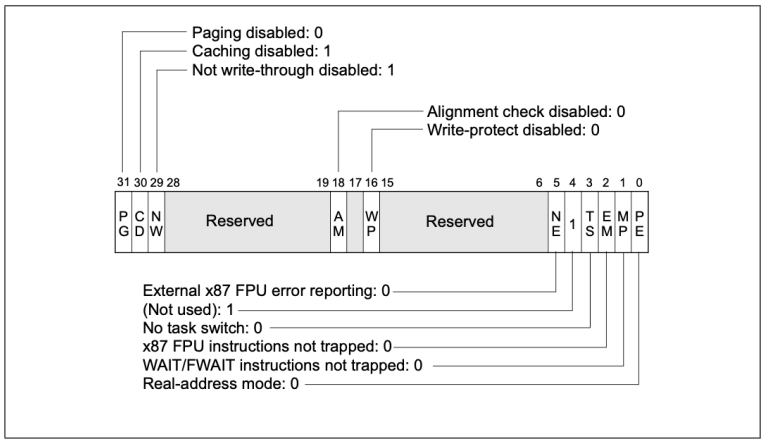


Figure 9-1. Contents of CR0 Register after Reset

The state of the flags and other registers following power-up for the Pentium 4, Pentium Pro, and Pentium processors are shown in Section 22.39, "Initial State of Pentium, Pentium Pro and Pentium 4 Processors" of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B*.

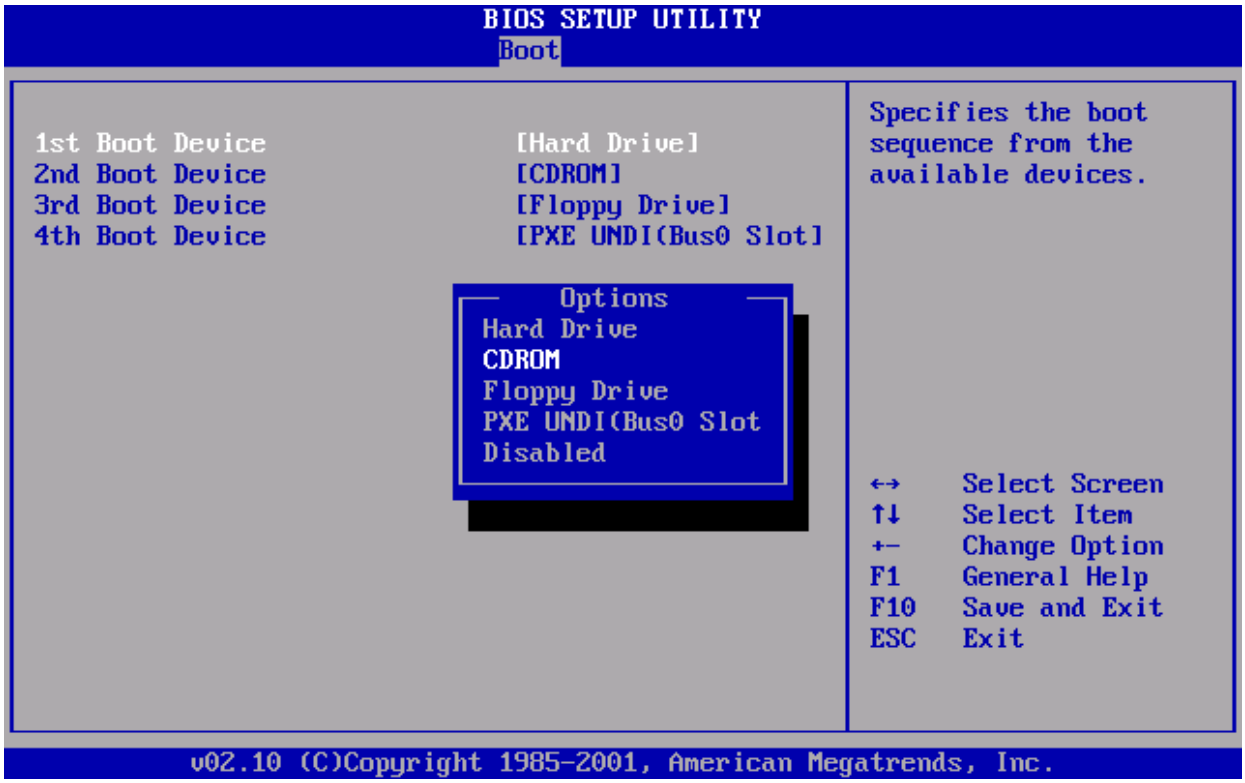
Table 9-1 shows processor states of IA-32 and Intel 64 processors with CPUID DisplayFamily signature of 06H at the following events: power-up, RESET, and INIT. In a few cases, the behavior of some registers behave slightly different across warm RESET, the variant cases are marked in Table 9-1 and described in more detail in Table 9-2.

Table 9-1. IA-32 and Intel 64 Processor States Following Power-up, Reset, or INIT

Register	Power up	Reset	INIT
EFLAGS[1]	00000002H	00000002H	00000002H
EIP	0000FFF0H	0000FFF0H	0000FFF0H
CR0	60000010H[2]	60000010H[2]	60000010H[2]
CR2, CR3, CR4	00000000H	00000000H	00000000H
CS	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = F000H Base = FFFF0000H Limit = FFFFH AR = Present, R/W, Accessed
SS, DS, ES, FS, GS	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed	Selector = 0000H Base = 00000000H Limit = FFFFH AR = Present, R/W, Accessed
EDX	000n06xxH[3]	000n06xxH[3]	000n06xxH[3]
EAX	0[4]	0[4]	0[4]
EBX, ECX, ESI, EDI, EBP, ESP	00000000H	00000000H	00000000H
ST0 through ST7[5]	+0.0	+0.0	FINIT/FNINIT: Unchanged

CPU Reset (Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A/3B)

- 寄存器会有确定的初始状态
  - `EIP = 0x0000fff0`
  - ```
    1 CR0 = 0x60000010
    ```
    ```

        - 处理器处于 16-bit 模式
    - ```
      EFLAGS = 0x00000002
      ```
        - Interrupt disabled
- TFM (5,000 页+)
    - 最需要的 Volume 3A 只有 ~400 页（我们更需要 AI）

其他平台上的 CPU Reset

Reset 后处理器都从固定地址（Reset Vector）启动

- MIPS: 0xbfc00000
    - Specification 规定
- ARM: 0x00000000
    - Specification 规定
    - 允许配置 Reset Vector Base Address Register
- RISC-V: Implementation defined
    - 给厂商最大程度的自由

Firmware 负责加载操作系统

- 开发板：直接把加载器写入 ROM
- QEMU：`-kernel` 可以绕过 Firmware 直接加载内核（RTFM）

x86 CPU Reset 之后：到底执行了什么?

状态机（初始状态）开始执行

- 从 PC 取指令、译码、执行……
- 开始执行厂商“安排好”的 Firmware 代码
    - x86 Reset Vector 是一条向 Firmware 跳转的 jmp 指令

      ```



Firmware: BIOS vs. UEFI

- 一个小“操作系统”
  - 管理、配置硬件；加载操作系统
- Legacy BIOS (Basic I/O System)
  - IBM PC 所有设备/BIOS 中断是有 specification 的（成就了“兼容机”）
- UEFI (Unified Extensible Firmware Interface)

为什么需要 UEFI?

今天的 Firmware 面临麻烦得多的硬件：

- 指纹锁、USB 转接器上的 Linux-to-Go 优盘、山寨网卡上的 PXE 网络启动、USB 蓝牙转接器连接的蓝牙键盘、……
  - 这些设备都需要“驱动程序”才能访问
  - 解决 BIOS 逐渐碎片化的问题

回到 Legacy BIOS：约定

BIOS 提供机制，将程序员的代码载入内存

- Legacy BIOS 把第一个可引导设备的第一个 512 字节加载到物理内存的`7c00`位置
  - 此时处理器处于 16-bit 模式
  - 规定`CS:IP = 0x7c00 (R[CS] << 4) | R[IP] == 0x7c00`
    - 可能性1：`CS = 0x07c0, IP = 0`
    - 可能性2：`CS = 0, IP = 0x7c00`

    - 其他没有任何约束

虽然最多只有 446 字节代码（64B 分区表 + 2B 标识）

- 但控制权已经回到程序员手中了！

🌶️ UEFI 上的操作系统加载

标准化的加载流程

- 磁盘必须按 GPT（GUID Partition Table）方式格式化
- 预留一个 FAT32 分区（lsblk/fdisk 可以看到）
- Firmware 能够加载任意大小的 PE 可执行文件.efi
    - 没有 legacy boot 512 字节限制
    - EFI 应用可以返回 firmware

更好的程序支持

- 设备驱动框架
- 更多的功能，例如 Secure Boot，只能启动“信任”的操作系统

听说操作系统也是程序。那到底是鸡生蛋还是蛋生鸡?

总结

Take-away Messages

计算机系统是严格的数学对象：没有魔法；计算机系统的一切行为都是可观测、可理解的。

- 处理器是无情的执行指令的机器
- 厂商配置好处理器 Reset 后的行为：先运行 Firmware，再加载操作系统
- 厂商逐渐形成了达成共识的 Firmware Specification（IBM PC “兼容机”、UEFI、……）
- 操作系统真的就是个 C 程序，只是能直接访问计算机硬件

课后习题/编程作业

1. 编程实践

在完成实验的过程中，如果不使用本次课展示的调试技巧（而是用 “蛮力” printf 等），将会有更多的挫败感、付出多的多的时间。某种程度上说，课堂上的阅读代码和调试技术是大家对一定规模代码系统的入门必备习题。因此务必动手探索计算机系统的启动过程，和 Hello World 程序是如何在计算机硬件上运行的。