Operating Systems II

Booting and Kernel Initialization

Objectives

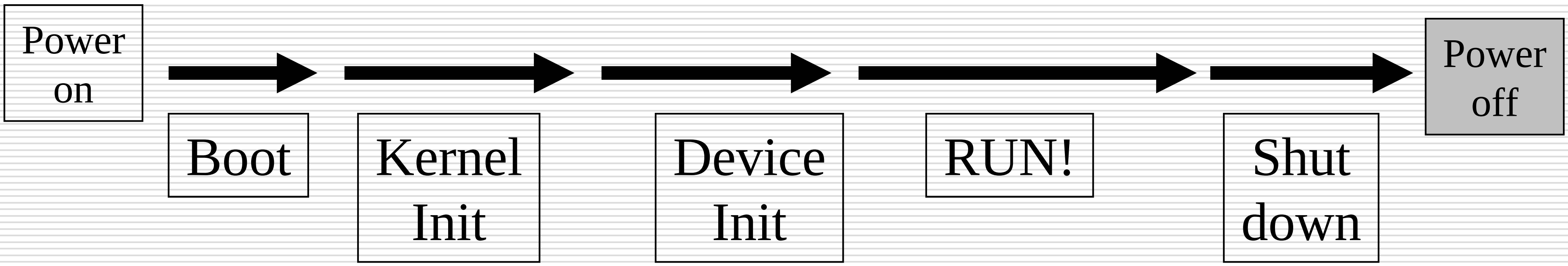
- Describe general principles involved in **booting** a system.
  - Some specific details of a standard LILO disk-based boot on Intel architecture.
- Motivate and clarify the **transfer of control** from hardware, to firmware, to software during system boot.
- Trace significant events in **kernel initialization**.

Objectives

- Demonstrate role and importance of the **initial** process.
- Review **shutdown** procedures.
- Briefly survey a variety of **advanced boot concepts**.
- Briefly consider **power management** issues.

System Lifecycle: Ups & Downs

- Booting
- Kernel initialization
- Device management initialization
- Full operation
- Shutdown

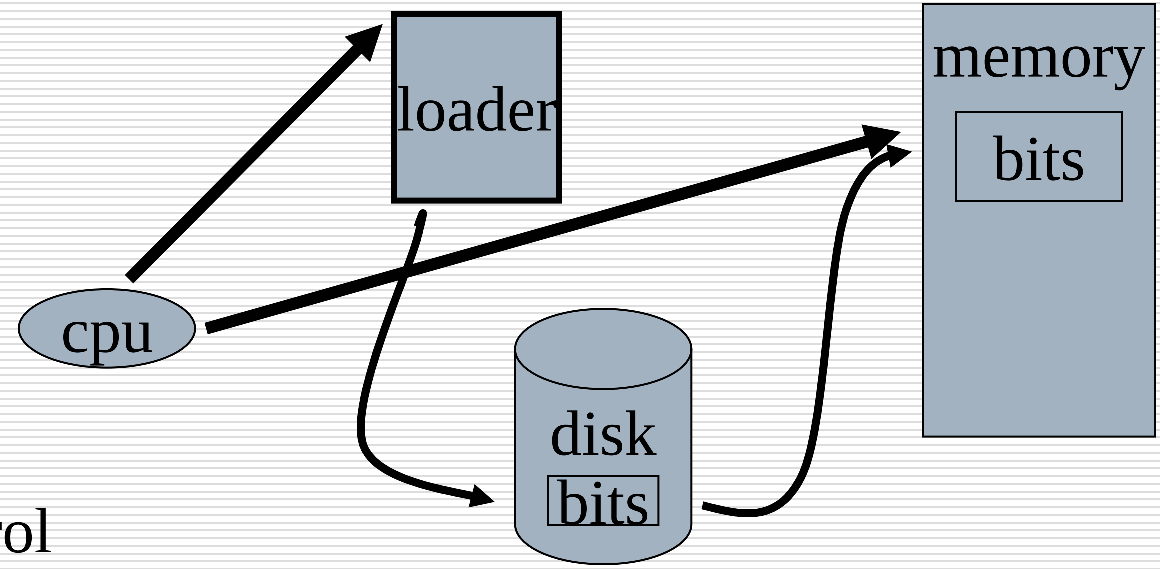


Boot Terminology

- Loader
  - Program that moves bits from disk (usually) to memory and then transfers CPU control to the newly “loaded” bits (executable)
- Bootloader / Bootstrap
  - Program that loads the “first program” (kernel)
- Boot PROM / PROM Monitor / BIOS
  - Persistent code that is “already loaded” on power-up
- Boot Manager
  - Program that lets you choose the “first program” to load

What’s a Loader?

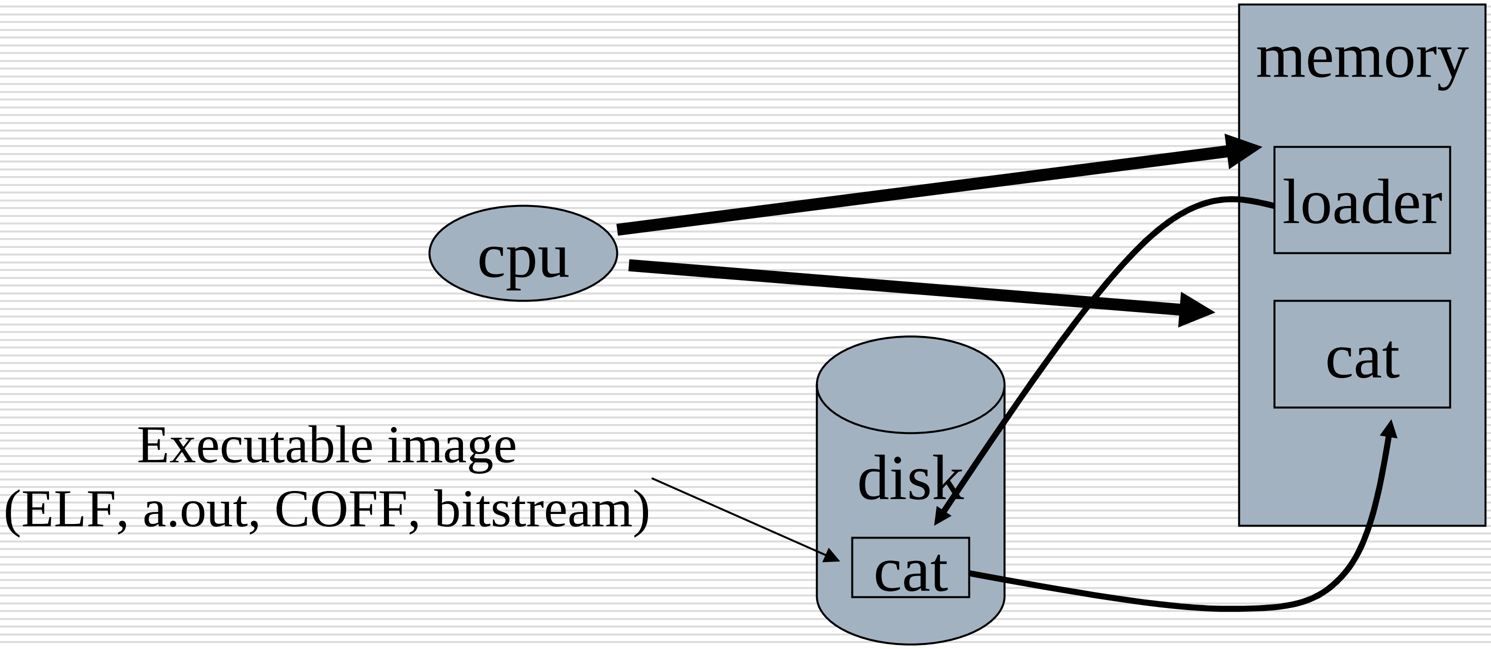
- A program that moves bits (usually) from disk to memory and then transfers control to the newly loaded bits (executable).



1) Move bits
2) Transfer control

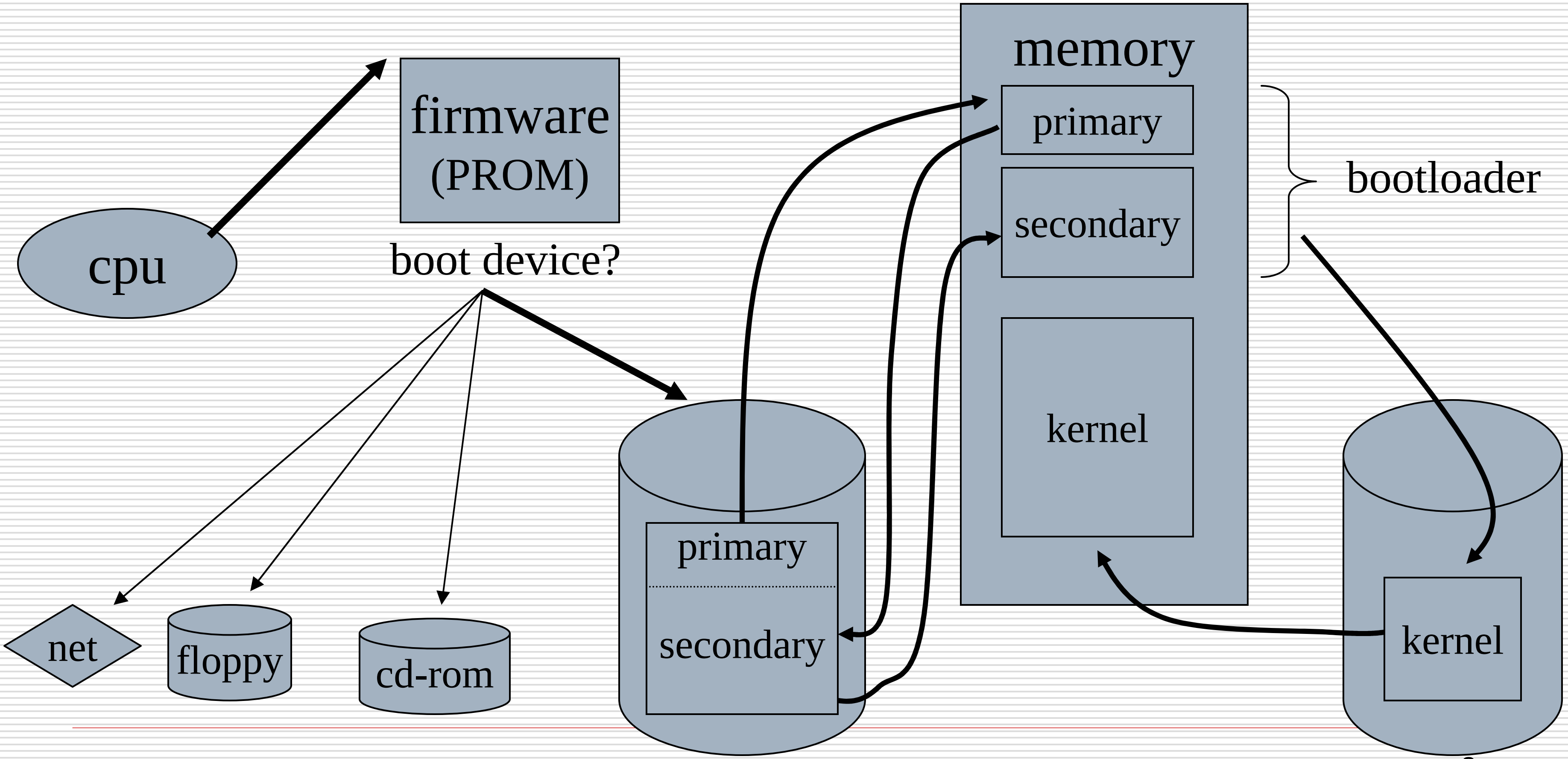
Who Loads the Loader?

- Of course, the loader is just a program and it resides in memory too. How did it get there?



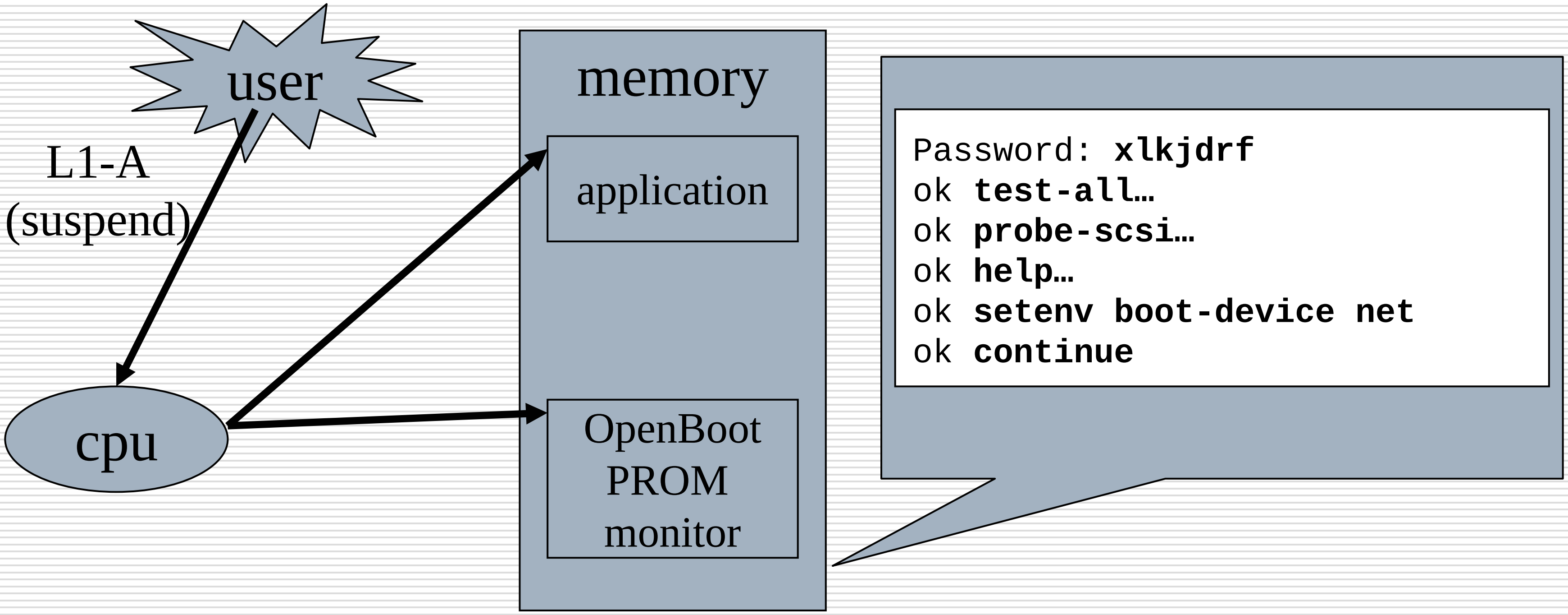
Bootstrap Loader (Bootloader)

- The program that loads the "first program"
- Usually "staged": primary, secondary
- Requires firmware support ("hardware bootstrap")



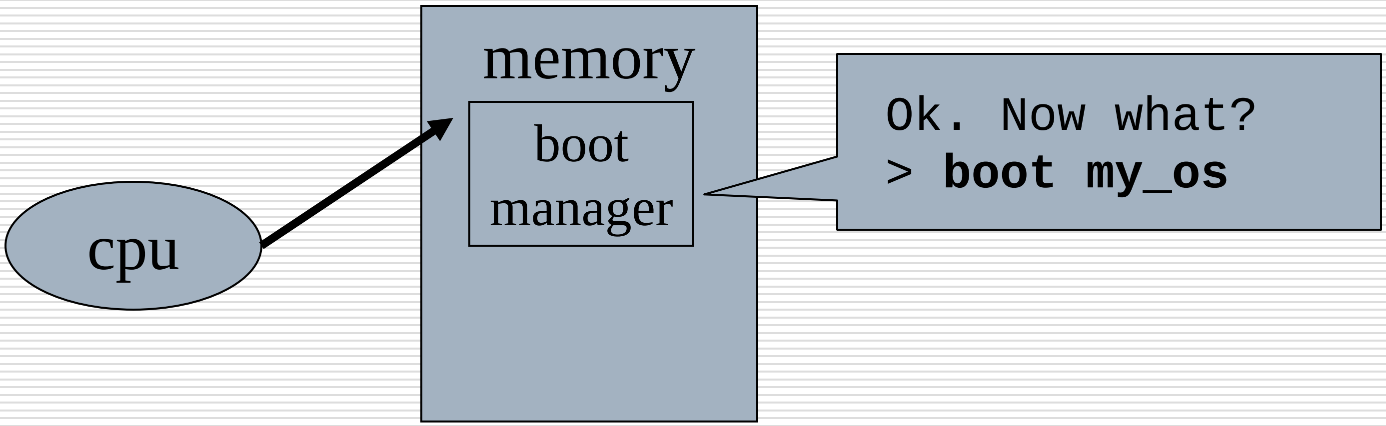
PROM Monitors vs. BIOS

- BIOS: limited setup via DEL or F1 at boot
- Monitor: continuously accessible command interpreter
- Examples: Intel (BIOS) and Sparc (Monitor)



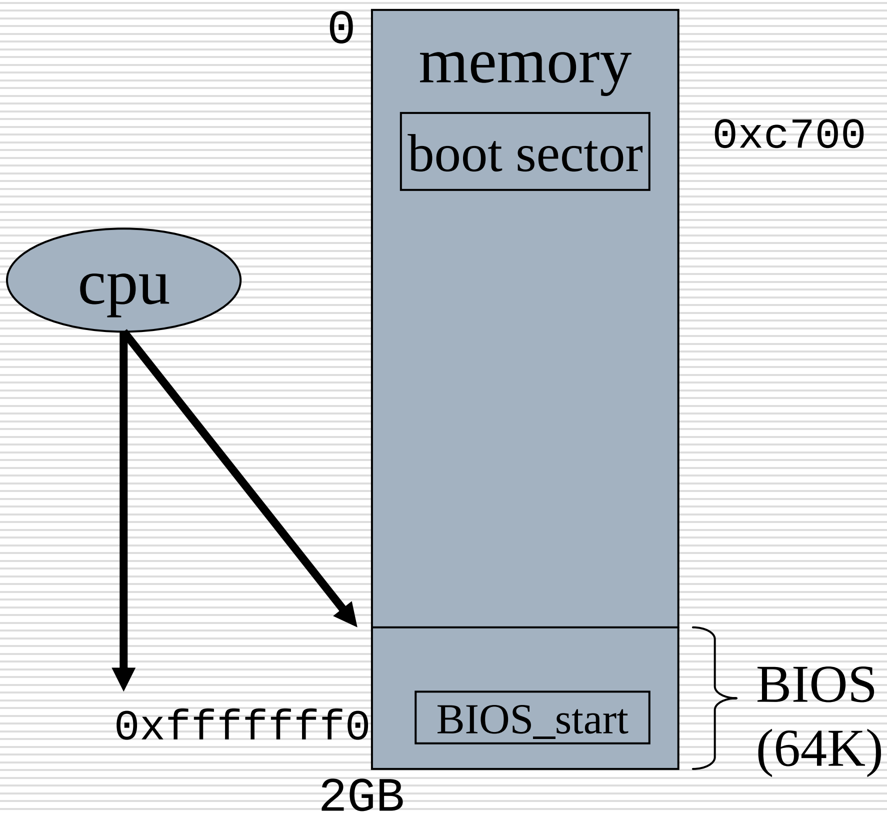
Boot Managers

- Code loaded by firmware bootstrap that allows choice of boot image, specification of boot parameters, etc.
- Adds another "layer" to boot process but increases flexibility, supports "multiboot" configurations
- Examples: LILO, System Commander



Booting a PC

- Intel X86 firmware loads a 512 byte “boot sector” at 0x7C00 and transfers control in real-mode (640K limit)



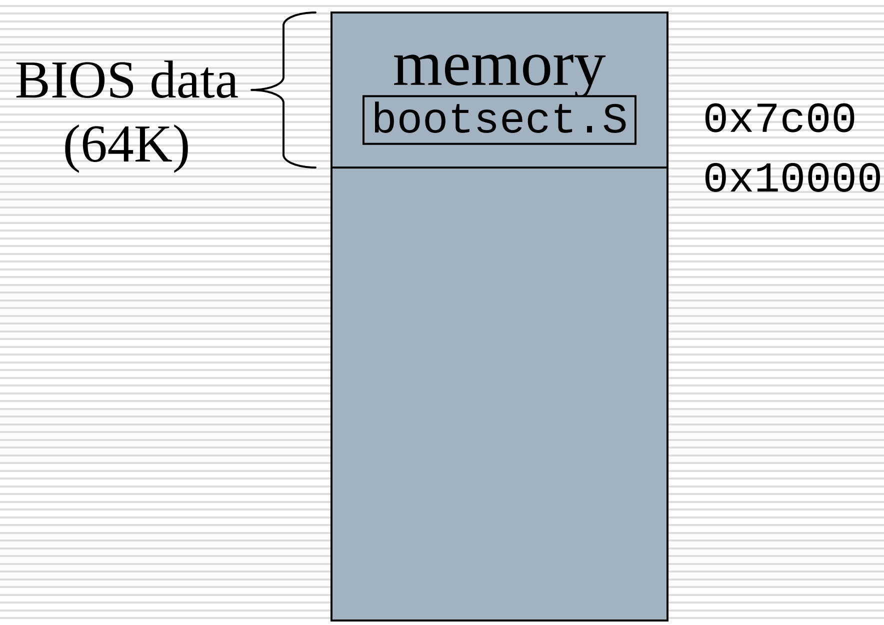
1. Power On Self Test (POST)
2. Generate INT 19h (bootstrap)
3. Select boot device
4. Load boot sector
    1. floppy: first sector
    2. hard disk: MBR or partition boot block
5. Verify “magic number”
6. Execute boot sector (primary bootloader)

Booting from a Floppy 1

- Compressed kernel dumped directly to floppy
- Boot complicated by real-mode, BIOS, compression

1

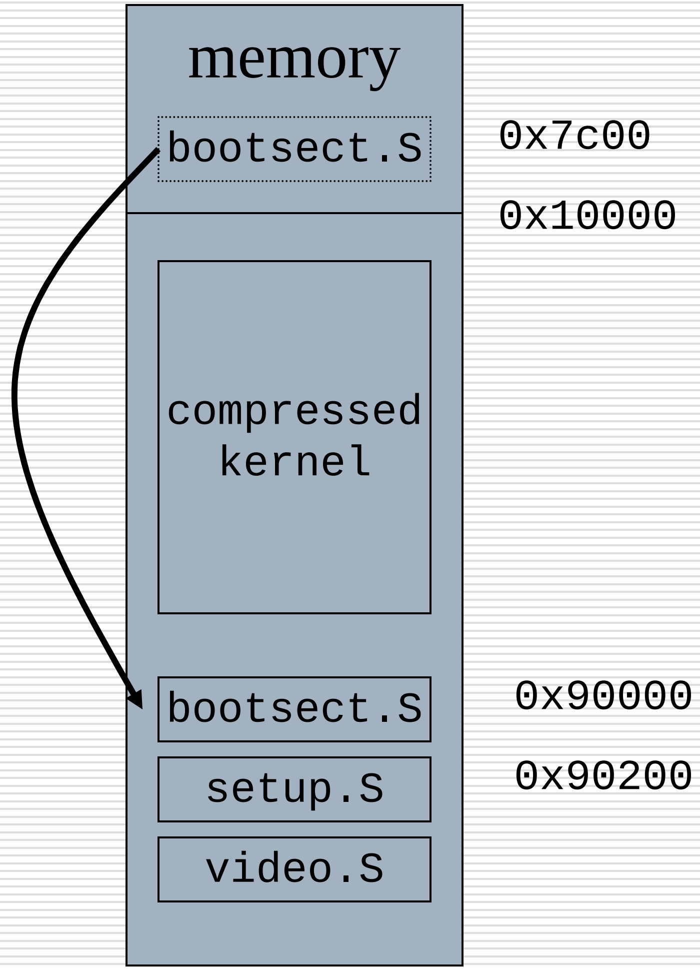
- BIOS loads boot sector
- Transfers control



Booting from a Floppy 2

2

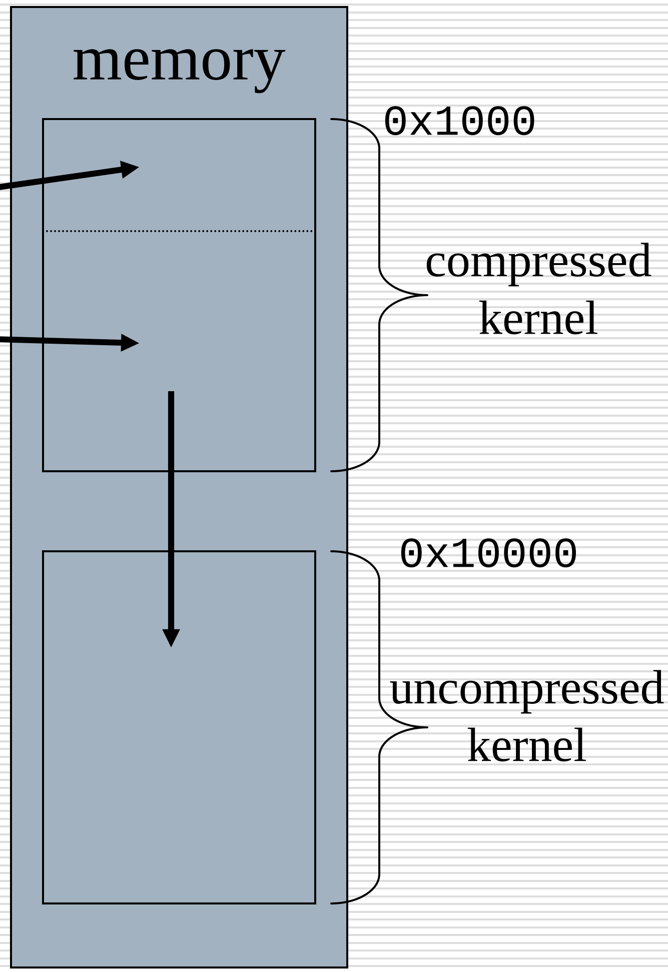
- Boot sector moves itself (!) to 0x90000
  - Limited access to memory
- Loads additional bootstrapping code
  - two more sectors at 0x90200
    - arch/i386/boot/setup.S
    - arch/i386/boot/video.S
- Loads compressed kernel after BIOS data
- Transfers control to setup.S
- Performs real-mode hardware Initialization



Booting from a Floppy 3

3

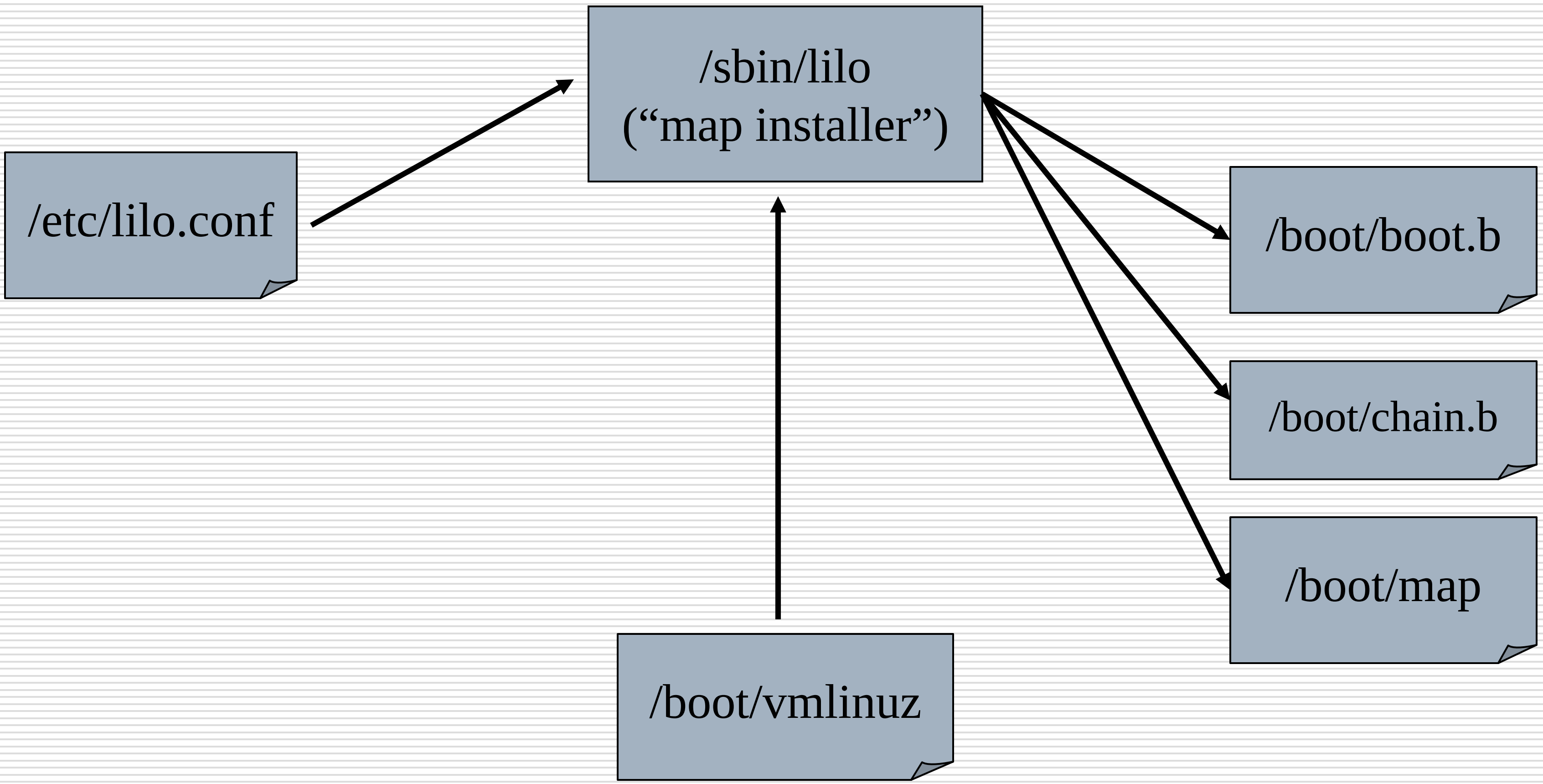
- setup.S copies kernel to 0x1000 (4K)
  - Avoids wasted space but trashes BIOS data
- Kernel consists of two pieces:
  - head.S
  - compressed tail.S
- Enters **protected mode**
- Jumps to 0x1000 (head.S)
- head.S
  - Sets up stack
  - “Self-extracts” kernel
  - Jumps to 0x10000 (kernel head.S)
- kernel head.S sets up paging
- Jumps to main.c: start_kernel (!)



LILO: LInux LOader

- A versatile boot manager that supports:
  - Choice of operating systems / kernels
  - Boot time kernel parameters
  - Booting non-Linux kernels
  - A variety of configurations
- Characteristics:
  - Lives in MBR or partition boot sector
  - Has no knowledge of filesystem structure so…
  - Builds a sector “map file” (block map) to find kernel
- /sbin/lilo – “map installer”
  - Builds map file, boot sector
  - Run after change to kernel or /etc/lilo.conf

LILO Components



Example lilo.conf File

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=linux

image=/boot/vmlinuz-2.2.12-20
       label=linux
       initrd=/boot/initrd-2.2.12-20.img
       read-only
       root=/dev/hda1
```

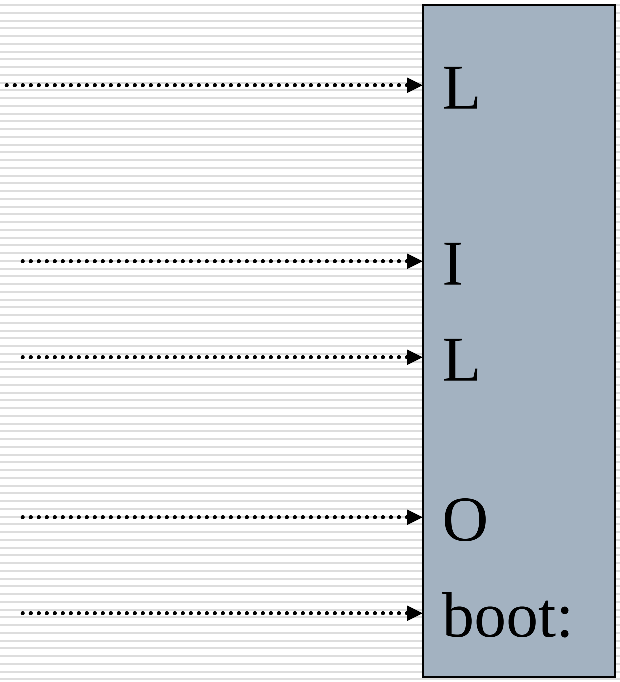
Booting from Disk with LILO

- LILO prints a progress string "LILO boot:"
  1. BIOS loads boot sector at 0x7c00

     Moves itself to 0x9a00
  2. Sets up stack

     Loads secondary bootloader at 0x9b00
  3. Transfers control to secondary
  4. Loads "block map" at 0x9d200

     Loads default command line at 0x9d600
  5. Waits for user input or timeout



Starting a Kernel

- Many different steps are involved in starting a kernel.
- General steps exist for what is to occur, but the actual execution is often **VERY** specific to the basic hardware platform (processor & busses).
- Key point is that order matters
  - Features of the OS are loaded/started as support is gradually increased.

Kernel Starting Jobs

- Identify Bootstrap Processor
- Initialize architecture
- Initialize crucial subsystems
  - Low-level device drivers
  - Scheduling
  - Memory Management
- Parse boot options
- Setup kernel profiling
- Calibrate time quantum

Kernel Starting Jobs (cont.)

- Enable Interrupts
- Initialize secondary subsystems / services
  - Typically systems that require a delay
- Check for bugs
- Initialize multiple processors (if any)
- Alter execution to reflect a “thread” in the OS
- Become the **idle** process – as a user thread!

BogoMIPS

- BogoMIPS is roughly the number of times per second the CPU can execute a short delay loop
- Used by device init code for short waits
  - Can be used as scheduling time quantum
- Misused to measure performance

Bogus!

  - Wait for next clock tick
  - Make initial estimate
  - Verify estimate, adjusting as necessary

Kernel Boot Options

- Passing boot options to the kernel can provide flexibility.
- Options can be used by either the kernel itself or passed to subsystem initialization routines.
- Most options are device specific.

Initialize Crucial Subsystems

- Perform “high-level” initialization requiring memory and process management to be setup
  - Do conditional bus initialization (i.e. pci)
  - Initialize socket communication
  - Memory management (paging, swap)
  - Set up basic devices
  - Filesystem initialization
  - Setup recognized file/executable formats
  - Mount filesystems

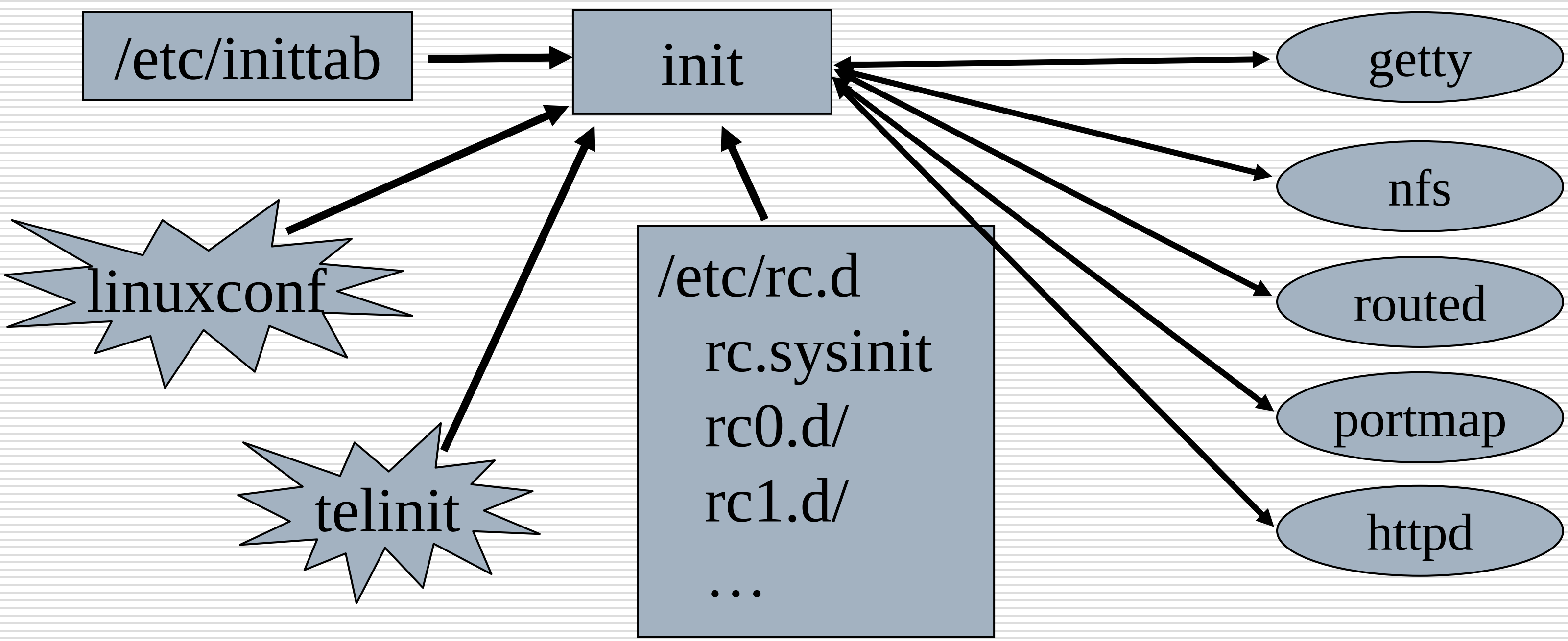
Secondary Subsystems

- Many operating systems rely upon individual processes to provide resources.
- In this hierarchy, the initial kernel process (eventually the **idle** thread) is the “parent” of all these processes.
- In modern operating systems the kernel can operate in multiple *modes*.
  - Modes dictate the overall system support.

Example: UNIX

- Ancestor of all processes (but idle); “reaps” children
- Controls transitions between “runlevels”
  - 0: shutdown   1: single-user   2: multi-user (no NFS)
  - 3: full multi-user   5: X11   6: reboot
- Executes startup/shutdown scripts for each runlevel

Example: UNIX (cont.)



Shutdown

- Any user issued command to the kernel to shutdown should result in a “graceful” termination of services.
  - Prevent any further creation of user processes.
  - Flush any pending I/O the kernel may be buffering.
    - Especially key for maintaining file integrity.
  - The kernel process (**idle**) sends a terminate signal to all processes in the system.
    - Remember: It is the overall parent!

Shutdown (cont.)

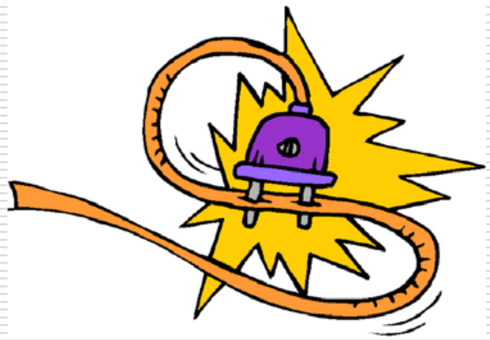
- Hardware oriented shutdowns may not be as graceful
  - CTRL-ALT-DEL or power switch
- This is dependent on how the kernel handles the hardware interrupt.
  - CTRL_ALT_DEL is often serviced as a regular shutdown user command.
  - Power switch is often a “hard” shutdown!
- Important to offer both hard and soft shutdown mechanisms to support user needs!

Advanced Boot Concepts

- Booting from a remote kernel
  - Requires network device communication BEFORE loading the kernel!
  - Similar to booting from a CDROM.
- RAID root
  - Trick for high-performance root
- OpenPROM – open-source BIOS; burn your own!
- Check out linux forums for more interesting advances in booting research

Power Management

- Power-management is essential for mobile systems
- Halting in the idle process
  - Idle process executes hlt on Intel
  - low-power consumption mode



- Suspending the system
  - Dump the state of the system to secondary storage
- APM: Advance Power Management
  - Laptop standard power management
- ACPI: Advanced Configuration and Power Interface
  - Comprehensive standard from Intel-Microsoft

Summary

- Bootstrapping a system is a complex, device-dependent process that involves transition from hardware, to firmware, to software.
- Booting within the constraints of the Intel architecture is especially complex and usually involves firmware support (BIOS) and a boot manager (LILO).
- So many options that are VERY specific to the hardware!