Robotics and Embedded Systems

Technische Universität München

Concept of Booting A Computer System

Kai Huang

Why is Booting Required ?

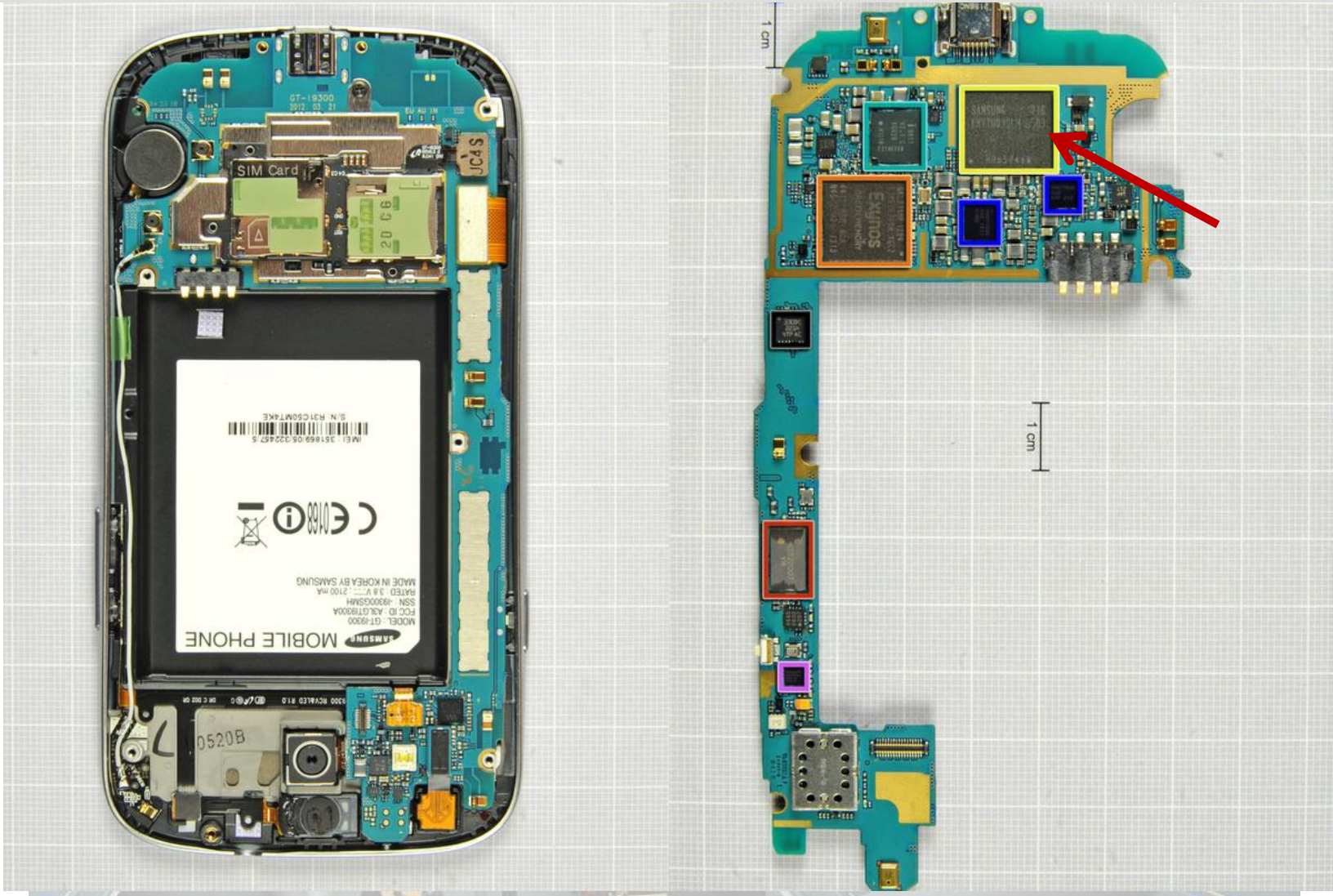
- Hardware doesn't know where the operating system resides and how to load it.
- Need a special program to do this job – Bootstrap loader.
  - E.g. BIOS – Boot Input Output System.

- Bootstrap loader locates the kernel, loads it into main memory and starts its execution.
- In some systems, a simple bootstrap loader fetches a more complex boot program from disk, which in turn loads the kernel.

How Boot Process Occurs ?

- Reset event on CPU (power up, reboot) causes instruction register to be loaded with a predefined memory location. It contains a jump instruction that transfers execution to the location of Bootstrap program.
- This program is form of ROM, since RAM is in unknown state at system startup. ROM is convenient as it needs no initialization and can’t be affected by virus.

Samsung SIII



Tasks performed at Boot Up

- Run diagnostics to determine the state of machine. If diagnostics pass, booting continues.
- Runs a Power-On Self Test (*POST*) to check the devices that the computer will rely on, are functioning.
- BIOS goes through a preconfigured list of devices until it finds one that is bootable. If it finds no such device, an error is given and the boot process stops.
- Initializes CPU registers, device controllers and contents of the main memory. After this, it loads the OS.

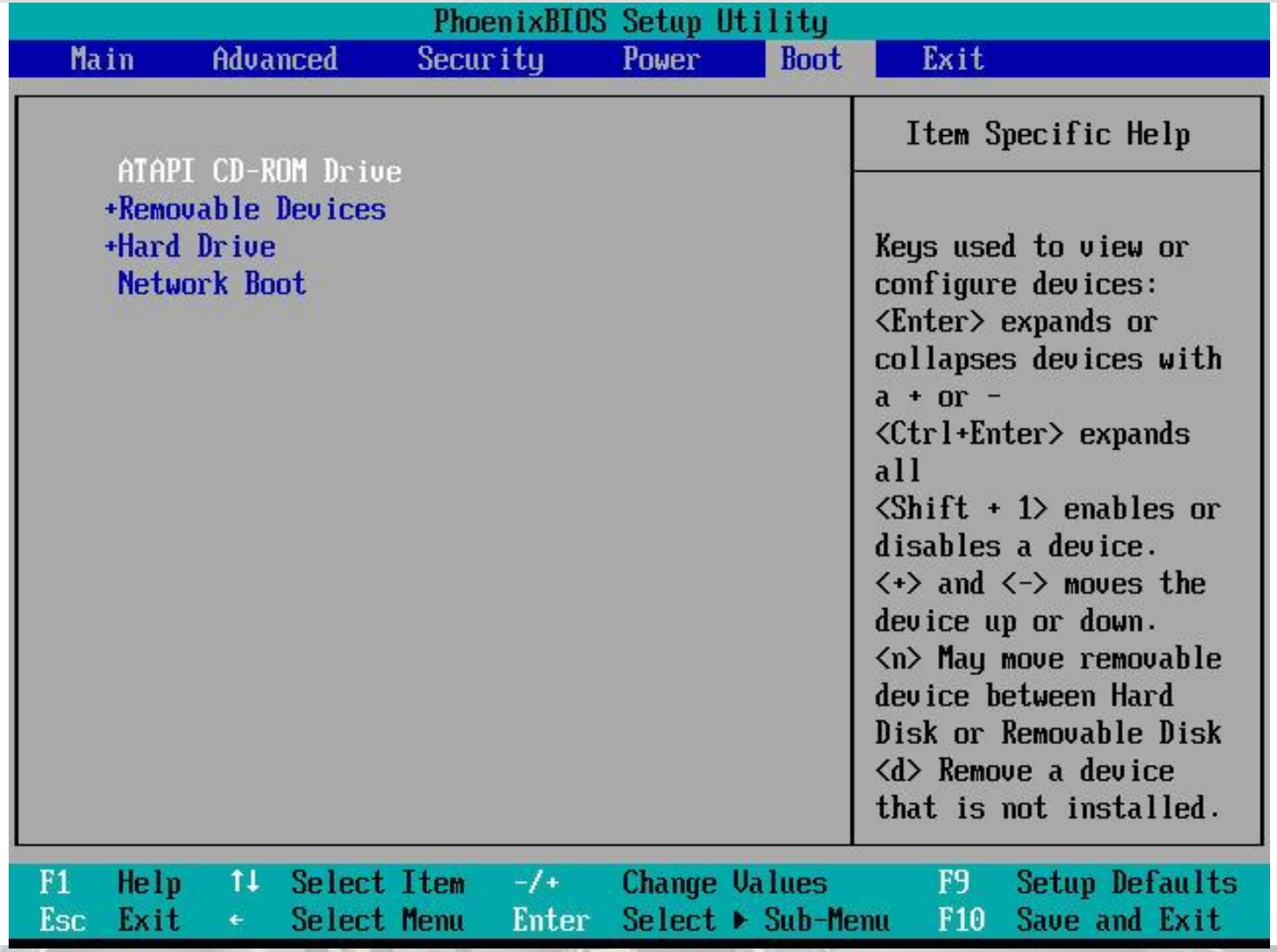
BIOS POST

```
CPU Type          : AMD Athlon(tm) XP     Base Memory       :    640K
CPU ID            : 0681                  Extended Memory   :1047552K
CPU Clock         : 2000MHz               L1 Cache Size     :    128K
                                          L2 Cache Size     :    256K

Diskette Drive A  : 1.44M, 3.5 in.        Display Type      : EGA/VGA
Pri. Master Disk  : LBA,ATA 100,40022MB   Serial Port(s)    : 3F8 2F8
Pri. Slave  Disk  : LBA,ATA 100,40062MB   Parallel Port(s)  : 378
Pri. Master Disk  : DVD,ATA 33            DDR DIMM at Rows  : 2 3 4 5
Sec. Slave  Disk  : CHS,PIO 4,  512MB

PCI device listing ...
Bus No. Device No. Func No. Vendor/Device Class Device Class                IRQ
  0        2         0        10DE      0067    0C03  USB 1.0/1.1 OHCI Controller 10
  0        2         1        10DE      0067    0C03  USB 1.0/1.1 OHCI Controller 11
  0        2         2        10DE      0068    0C03  USB 2.0 EHCI Controller      5
  0        9         0        10DE      0065    0101  IDE Controller              14
  0       13         0        10DE      006E    0C00  Serial Bus Controller       10
  1        8         0        1106      3043    0200  Network Controller          11
  1        9         0        1102      0002    0401  Multimedia Device           11
```

BIOS Booting Priority



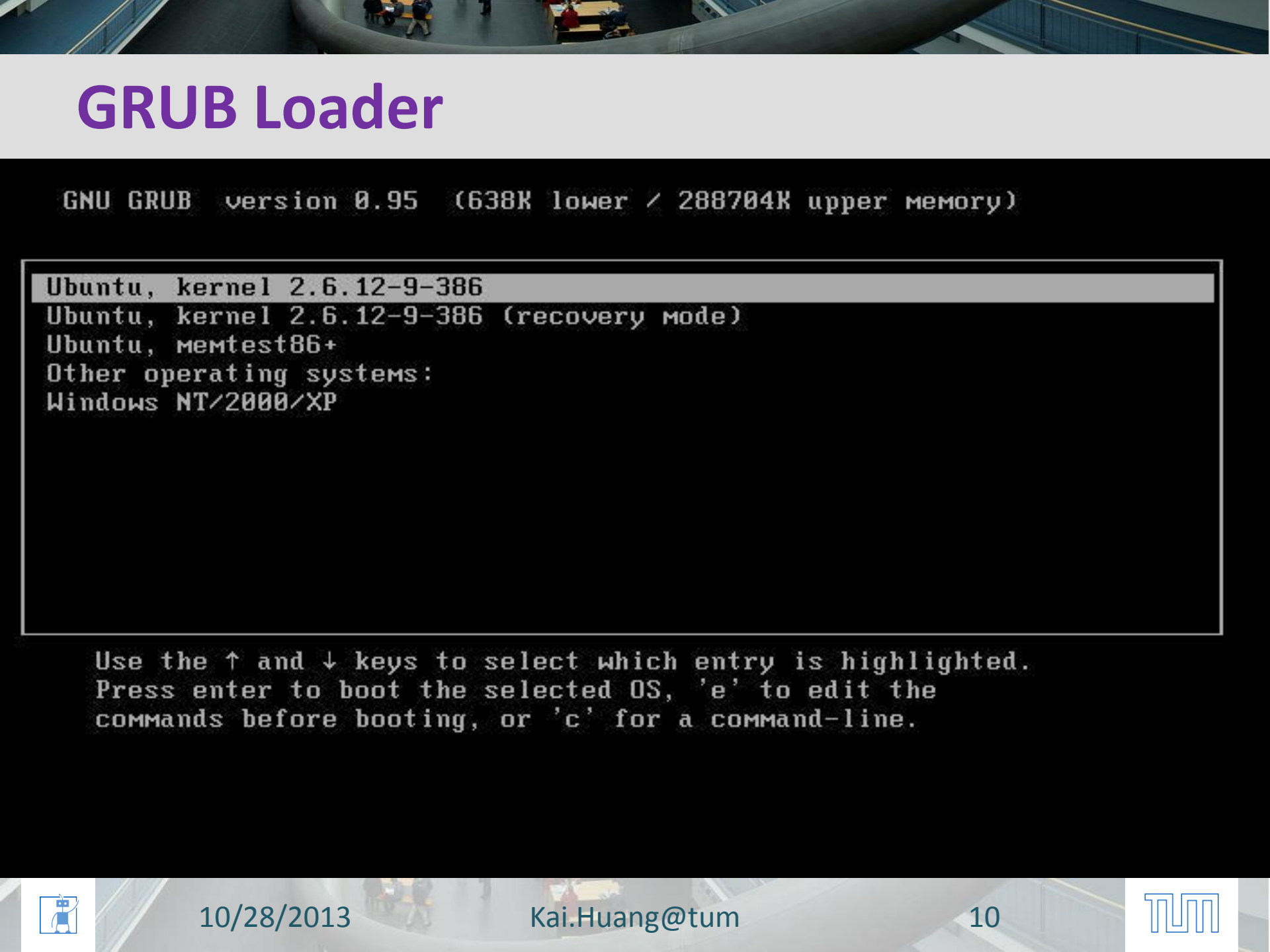
Tasks performed at boot up (Contd)

- On finding a bootable device, the BIOS loads and executes its boot sector. In the case of a hard drive, this is referred to as the master boot record (*MBR*) and is often not OS specific.
- The MBR code checks the partition table for an active partition. If one is found, the MBR code loads that partition's boot sector and executes it.
- The boot sector is often operating system specific, however in most operating systems its main function is to load and execute a kernel, which continues startup.

Secondary Boot Loaders

- If there is no active partition or the active partition's boot sector is invalid, the MBR may load a secondary boot loader and pass control to it and this secondary boot loader will select a partition (often via user input) and load its boot sector.
- Examples of secondary boot loaders
  - GRUB – GRand Unified Bootloader
  - LILO – LInux LOader
  - NTLDR – NT Loader

GRUB Loader


GNU GRUB version 0.95 (638K lower / 288704K upper memory)
Ubuntu, kernel 2.6.12-9-386
Ubuntu, kernel 2.6.12-9-386 (recovery mode)
Ubuntu, memtest86+
Other operating systems:
Windows NT/2000/XP
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

Booting and ROM

- System such as cellular phones, PDAs and game consoles stores entire OS on ROM. Done only for small OS, simple supporting hardware, and rugged operation.
- Changing bootstrap code would require changing ROM chips.
  - EPROM – Erasable Programmable ROM.
- Code execution in ROM is slower. Copied to RAM for faster execution.

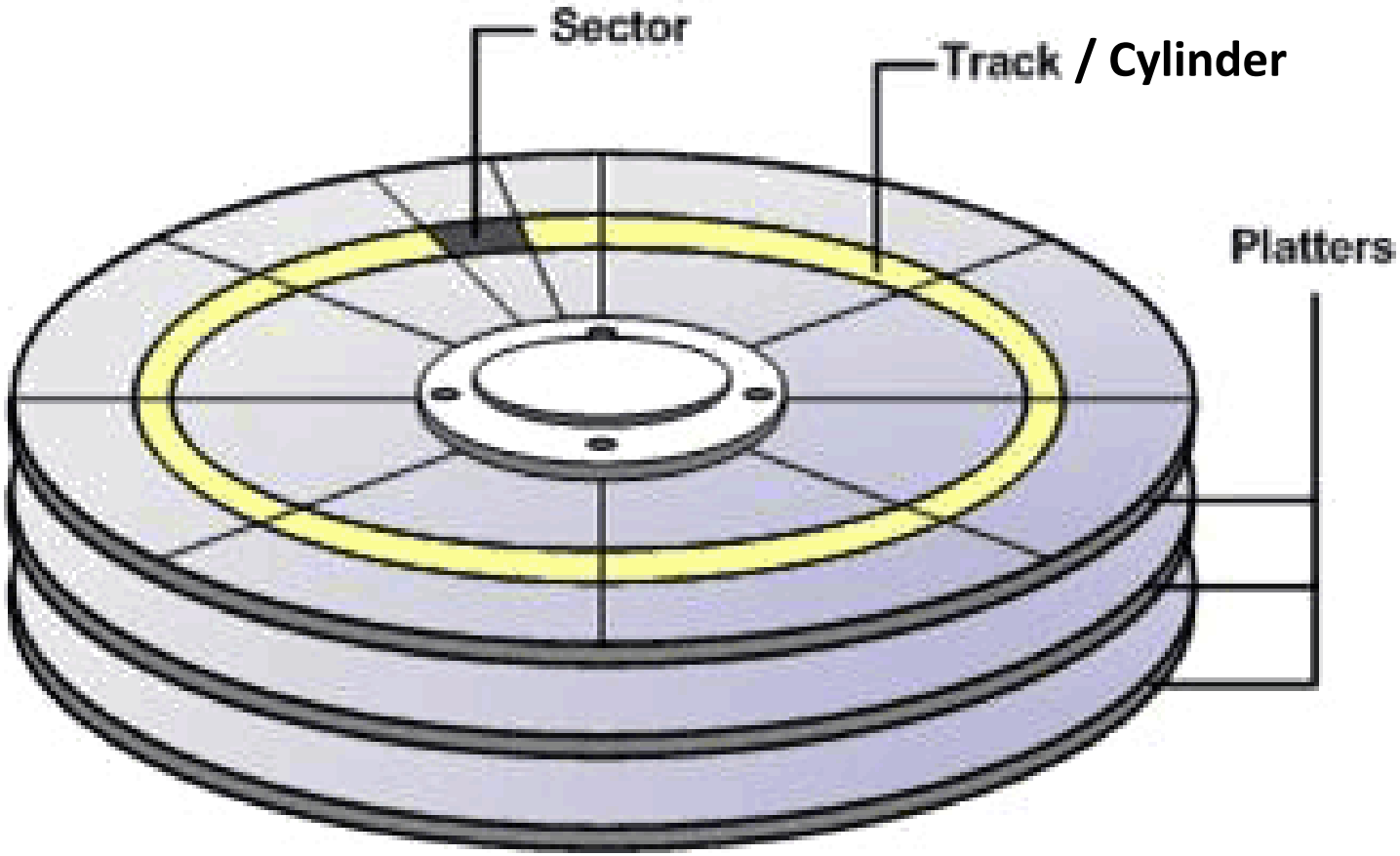
Storage Media

- Hard disks, floppy disk, thumb drives etc.
- Hard disks are the richest in digital evidence
- Integrated Disk Electronics (IDE) or Advanced Technology Attachment (ATA)
- Higher performance SCSI drives
- Fireware is an adaptation of SCSI standards that provides high speed access to a chain of devices
- All hard drives contain platters made of light, rig-hid material such aluminum, ceramic, or glass

Hard Disks

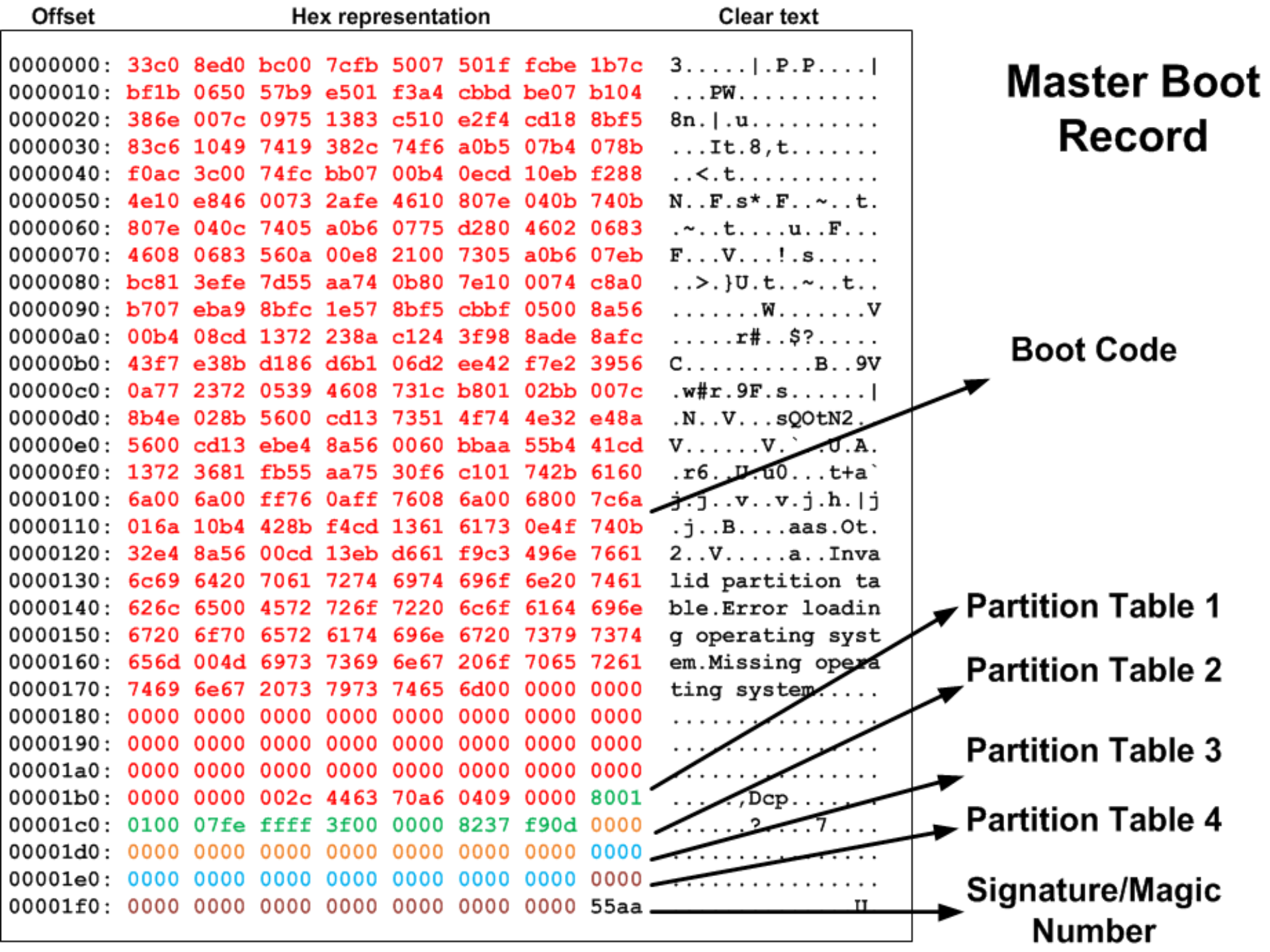
Hard Disk Platters

Tracks and Sectors



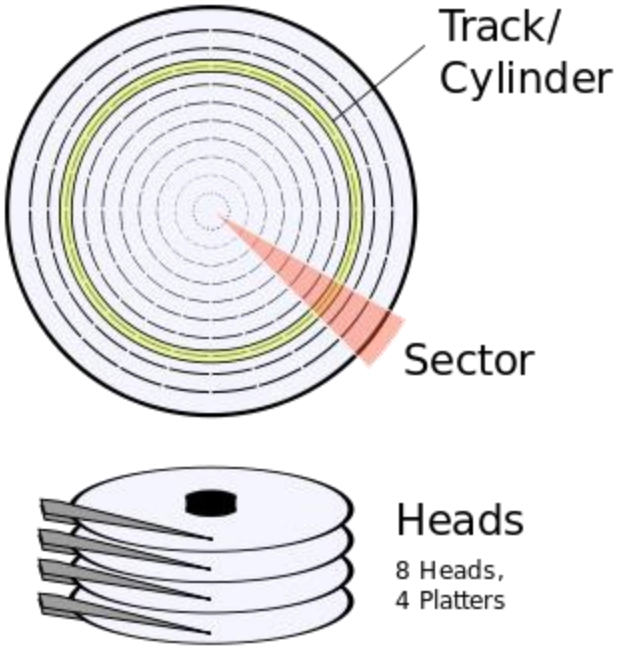
Master Boot Record

- Master Boot Record in first sector (1st 512 byte)
  - Boot Code
  - Partition Table
  - Signature Value
- MBR Supports a maximum of 4 partitions


Offset
Hex representation
Clear text
0000000: 33c0 8ed0 bc00 7cfb 5007 501f fcbe 1b7c 3.....|.P.P....|
0000010: bf1b 0650 57b9 e501 f3a4 cbbd be07 b104 ...PW.........
0000020: 386e 007c 0975 1383 c510 e2f4 cd18 8bf5 8n.|.u........
0000030: 83c6 1049 7419 382c 74f6 a0b5 07b4 078b ...It.8,t.....
0000040: f0ac 3c00 74fc bb07 00b4 0ecd 10eb f288 ..<.t.........
0000050: 4e10 e846 0073 2afe 4610 807e 040b 740b N..F.s*.F..~..t.
0000060: 807e 040c 7405 a0b6 0775 d280 4602 0683 .~..t....u..F...
0000070: 4608 0683 560a 00e8 2100 7305 a0b6 07eb F...V...!.s.....
0000080: bc81 3efe 7d55 aa74 0b80 7e10 0074 c8a0 ..>.}U.t..~..t..
0000090: b707 eba9 8bfc 1e57 8bf5 cbbf 0500 8a56 ....W.......V
00000a0: 00b4 08cd 1372 238a c124 3f98 8ade 8afc .....r#..$?.....
00000b0: 43f7 e38b d186 d6b1 06d2 ee42 f7e2 3956 C..........B..9V
00000c0: 0a77 2372 0539 4608 731c b801 02bb 007c .w#r.9F.s......|
00000d0: 8b4e 028b 5600 cd13 7351 4f74 4e32 e48a .N..V...sQOtN2..
00000e0: 5600 cd13 ebe4 8a56 0060 bbaa 55b4 41cd V......V.`..U.A.
00000f0: 1372 3681 fb55 aa75 30f6 c101 742b 6160 .r6..U.u0...t+a`
0000100: 6a00 6a00 ff76 0aff 7608 6a00 6800 7c6a j.j..v..v.j.h.|j
0000110: 016a 10b4 428b f4cd 1361 6173 0e4f 740b .j..B....aas.Ot.
0000120: 32e4 8a56 00cd 13eb d661 f9c3 496e 7661 2..V.....a..Inva
0000130: 6c69 6420 7061 7274 6974 696f 6e20 7461 lid partition ta
0000140: 626c 6500 4572 726f 7220 6c6f 6164 696e ble.Error loadin
0000150: 6720 6f70 6572 6174 696e 6720 7379 7374 g operating syst
0000160: 656d 004d 6973 7369 6e67 206f 7065 7261 em.Missing opera
0000170: 7469 6e67 2073 7973 7465 6d00 0000 0000 ting system.....
0000180: 0000 0000 0000 0000 0000 0000 0000 0000 ....
0000190: 0000 0000 0000 0000 0000 0000 0000 0000 ....
00001a0: 0000 0000 0000 0000 0000 0000 0000 0000 ....
00001b0: 0000 0000 002c 4463 70a6 0409 0000 8001 .....,Dcp.......
00001c0: 0100 07fe ffff 3f00 0000 8237 f90d 0000 ......?....7....
00001d0: 0000 0000 0000 0000 0000 0000 0000 0000 ....
00001e0: 0000 0000 0000 0000 0000 0000 0000 0000 ....
00001f0: 0000 0000 0000 0000 0000 0000 0000 55aa ....U.
Master Boot Record
Boot Code
Partition Table 1
Partition Table 2
Partition Table 3
Partition Table 4
Signature/Magic Number

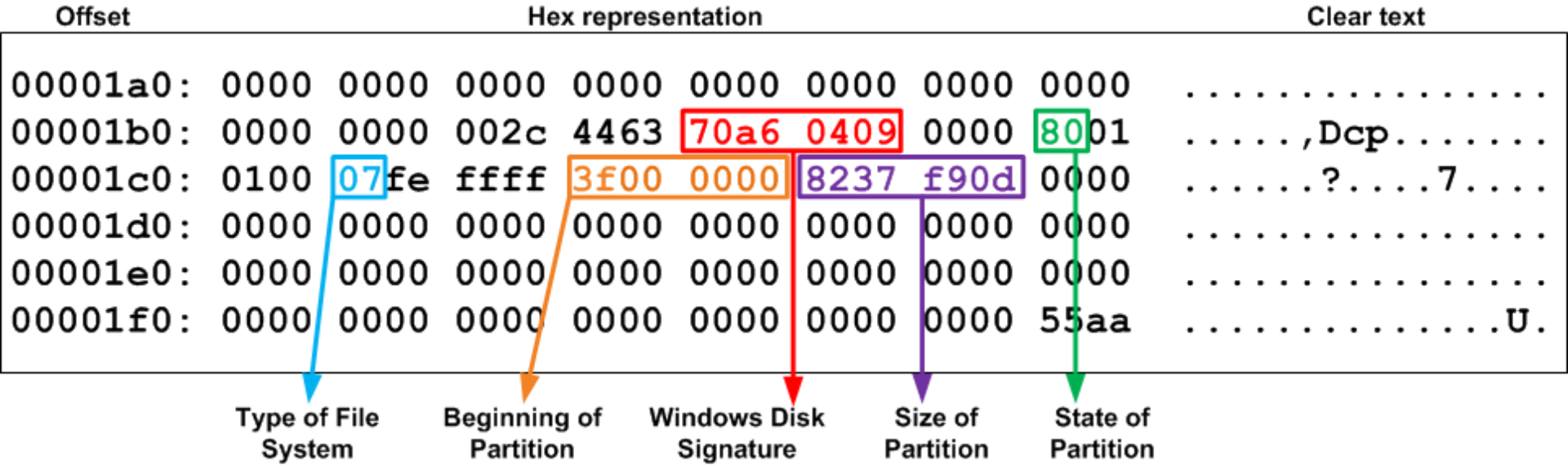
Partition Entry

- Starting CHS Address
- Ending CHS Address
- Starting LBA Address
- Number of Sectors in Partition
- Type of Partition
- Flags

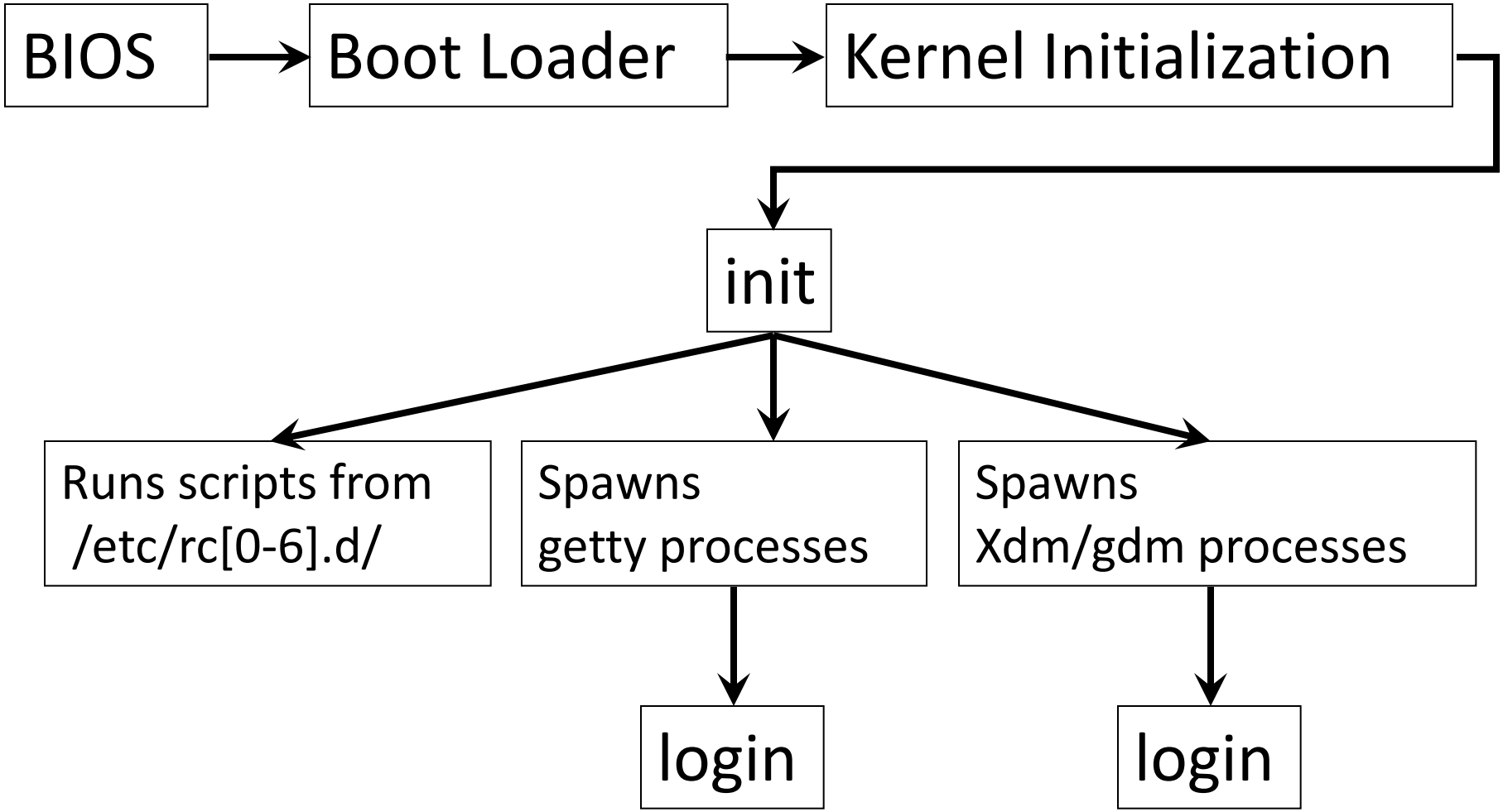


- Limitation
  - 2 Terabyte Disk Partition Limitation
  - MBR Partition size field is 32 bits

Layout of Partition Entry



Booting Linux



Overview of Booting Linux

- Firmware (bootloader)
  - Hardware probing
  - Hardware initialization
  - Kernel load and decompression
- Kernel execution
  - Core init (start_kernel)
  - Driver init (initcalls)
- User-space init
  - /sbin/init
  - RC scripts
  - Graphics start (First Impression)
- Application start
  - Application load and link
  - Application initialization
- First use

Kernel Initialization

- A program itself
  - /vmlinuz or /boot/vmlinuz
- Two-stage loading process
  - initrd (init RAM disk)
    - A transient root filesystem in RAM before a real root filesystem is available
    - E.g., it is used to install file system modules into the kernel
  - The real root filesystem
- Device detection and configuration
  - You tell the kernel what to expect
  - The kernel probes the H/W itself
- Kernel threads creation
  - E.g., init (a user process), kjournald, kswapd

Startup/Init Scripts

- After Kernel initialization, a process called init is created with PID 1
- init runs startup scripts (normal shell scripts) to perform specific tasks, e.g.,
  - Setting the hostname, time zone, etc
  - Checking and mounting the disks
  - Configuring network interfaces
  - Starting up daemons and network services

Startup/Init Scripts (cont' 1)

- Startup scripts (rc files) are run based on run levels
  - 0 the level in which the system is completely shut down
  - 1 single-user mode
  - 2 multiuser mode w/out NFS
  - 3 full multiuser mode
  - 4 unused
  - 5 X11
  - 6 reboot level
- /etc/init/rc-sysinit.conf -> the place to config the run level (usually 2 or3)
- /etc/inittab tells init what to do at each level (obsolete)
- /etc/init (SysVinit) V.S. /etc/init.d (Upstart)
- To find out which run level the system is current in
  - $ runlevel

Startup/Init Scripts (cont' 2)

- init runs the scripts from /etc/rc.d/rc[0-6].d/
  - /etc/rc.d/rc0/K25sshd → /etc/init.d/sshd
  - /etc/rc.d/rc3/S55sshd → /etc/init.d/sshd
- Each server/daemon provides a master script
  - Stored in /etc/init.d
  - Understands the arguments: start, stop, restart
    - /etc/init.d/sshd start
- run level 0 → 3
  - /etc/rc.d/rc3/S* start
- run level 3 → 0
  - /etc/rc.d/rc0/K* stop
- Pretty ugly!

- $ man init

Startup/Init Scripts (cont' 3)

- Use chkconfig instead. E.g.,
  - $ chkconfig --add sshd
  - $ chkconfig --del sshd
- Before that, need to add/modify /etc/init.d/sshd
  - # chkconfig: 2345 55 25
    - sshd should be started/stopped at run level 2, 3, 4 and 5 with the start priority of 55 and the stop priority of 25
- $ man chkconfig

Principle for Altera booting

- Both the hardware design and the software design are stored in flash memory
  - In the case of DE0-nano, Altera erasable programmable configurable serial (EPCS) flash
  - Hardware image: SOF file
  - Software image: ELF file

Tasks (1-4)

- Check BIOS of the Virtual Machine
- Check GRUB
  - How many kernel images installed in the virtulal machine?
- Check first 512 bytes of your virtual machine
  - sudo dd if=/dev/sda of=sda.dump count=1 bs=512
  - hexdump sda.dump
  - How many partitions in the virtual machine?
- Check slides 22-26