# CS 3204
# Operating Systems

Lecture 22

Godmar Back

## Announcements

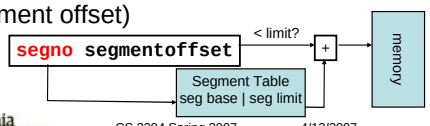
- Project 3 due tomorrow Wed Apr 11, 11:59pm
  - See forum for additional office hours
- Check curator for P2 scores
  - (tonight/early tomorrow)
- Project 4 Help Sessions
  - Th, Fr: 5:30-7:30 in McB 223



## Segmentation

## Segmentation

- Historical alternative to paging
- Instead of dividing virtual address space in many small, equal-sized pages, divide into a few, large segments
- Virtual address is then (segment number, segment offset)






## Segmentation (2)

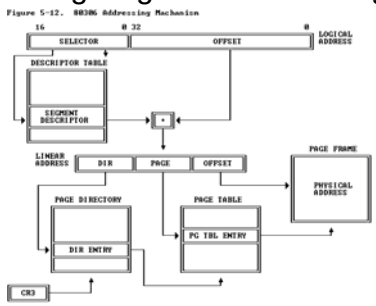
- Advantages:
  - little internal fragmentation "segments can be sized just right"
  - easy sharing – can share entire code segment
  - easy protection – only have to set access privileges for segment
  - small number of segments means small segment table sizes
- Disadvantages:
  - external fragmentation (segments require physically continuous address ranges!)
  - if segment is partially idle, can't swap out



## Segmentation (3)

- Pure segmentation is no longer used
  - (Most) RISC architectures don't support segmentation at all
  - Other architectures combine segmentation & paging
- Intel x86 started out with segmentation, then added paging
  - Segment number is carried in special set of registers (GS, ES, FS, SS), point to "selectors" kept in descriptor tables
  - Instruction opcode determines with segment is used
  - Today: segmentation unit is practically unused (in most 32-bit OS, including Pintos): all segments start at 0x00000000 and end at 0xFFFFFFFF (Exception: for thread-local data!)
  - Do not confuse with Pintos's code/data segments, which are linear subregions of virtual addresses spanning multiple virtual pages
- Note: superpages are somewhat of a return to segmentation

## Combining Segmentation & Paging






## Mem Mgmt Without Virtual Memory

- Book discusses this as motivation
  - Historically important, and still important for VM-less devices (embedded devices, etc.)
- Imagine if we didn't have VM, it would be hard or impossible to
  - Retain the ability to load a program anywhere in memory
  - Accommodate programs that grow or shrink in size
  - Use idle memory for other programs quickly
  - Move/relocate a running program in memory
- VM *drastically* simplifies systems design



## Disks & Filesystems

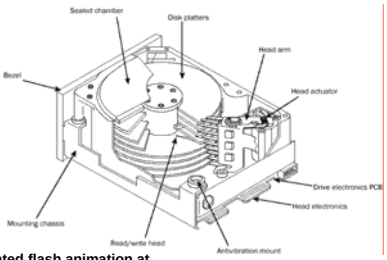


## What Disks Look Like


Hitachi Deskstar T7K500 SATA



## Disk Schematics

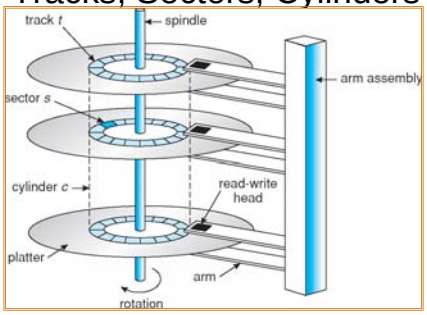



**See narrated flash animation at**
http://cis.poly.edu/cs2214rvs/disk.swf

***Source: Micro House PC Hardware Library Volume I: Hard Drives***



## Tracks, Sectors, Cylinders



## Typical Disk Parameters

- 2-30 heads (2 per platter)
  - Modern disks: no more than 4 platters
- Diameter: 2.5” – 14”
- Capacity: 20MB-500GB
- Sector size: 64 bytes to 8K bytes
  - Most PC disks: 512 byte sectors
- 700-20480 tracks per surface
- 16-1600 sectors per track

Virginia Tech

## What’s important about disks from OS perspective

- Disks are big & slow - compared to RAM
- Access to disk requires
  - Seek (move arm to track) – to cross all tracks anywhere from 20-50ms, on average takes 1/3.
  - Rotational delay (wait for sector to appear under track) 7,200rpm is 8.3ms per rotation, on average takes ½: 4.15ms rot delay
  - Transfer time (fast: 512 bytes at 998 Mbit/s is about 3.91us)
- Seek+Rot Delay dominates
- Random Access is expensive
  - and unlikely to get better
- Consequence:
  - avoid seeks
  - seek to short distances
  - amortize seeks by doing bulk transfers

Virginia Tech

## Disk Scheduling

- Can use priority scheme
- Can reduce avg access time by sending requests to disk controller in certain order
  - Or, more commonly, have disk itself reorder requests
- SSTF: shortest seek time first
  - Like SJF in CPU scheduling, guarantees minimum avg seek time, but can lead to starvation
- SCAN: “elevator algorithm”
  - Process requests with increasing track numbers until highest reached, then decreasing etc. – repeat
- Variations:
  - LOOK – don’t go all the way to the top without passengers
  - C-SCAN: - only take passengers when going up

Virginia Tech

## Accessing Disks

- Sector is the unit of atomic access
- Writes to sectors should always complete, even if power fails
- Consequence of sector granularity:
  - Writing a single byte requires read-modify-write

```
void set_byte(off_t off, char b) {
    char buffer[512];
    disk_read(disk, off/DISK_SECTOR_SIZE, buffer);
    buffer[off % DISK_SECTOR_SIZE] = b;
    disk_write(disk, off/DISK_SECTOR_SIZE, buffer);
}
```

Virginia Tech