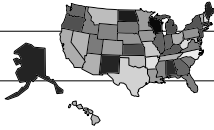


### Memory Mapping



• **Simple systems**

- Virtual address == physical address
- One address space for all processes

• **More typical approach**

- Virtual address maps to physical
- Mapping may be incomplete

---

---

---

---

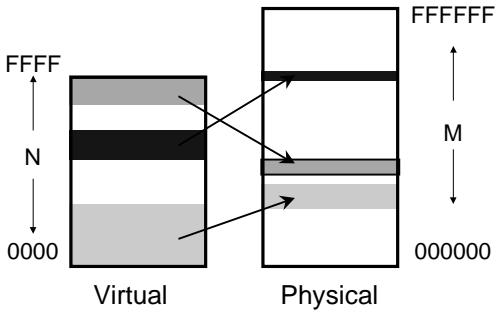
---

---

---

---

### Memory Mapping Example




---

---

---

---

---

---

---

---

### Implementing Mapping (1)

• **Lookup table**

- Maps each virtual address
- To a physical address (or nothing)

• **Table size**

- N entries if each address separate
- Smaller if blocks (pages)

---

---

---

---

---

---

---

---

Implementing Mapping (2)

- **Table entry**
  - Base address of physical page
- **Effective physical address**
  - Physical page base + offset into page
- **Page sizes**
  - 64, 512, 2048, or larger (tradeoffs?)

---

---

---

---

---

---

---

---

Implementing Mapping (3)

- **Unmapped virtual addresses**
  - Not valid for memory access
  - Generate system trap?
- **To change memory mapping**
  - Reload page table?
  - More than one table?

---

---

---

---

---

---

---

---

Secondary Storage

- **Magnetic media** Primary = main memory
  - Rigid disks (fixed, removable)
  - Flexible disks (diskettes)
  - Tape (reel, cartridge)
- **Optical media**
  - Read-only (CDROM), write-once (CD-R), or R/W (MO, CD-RW)

---

---

---

---

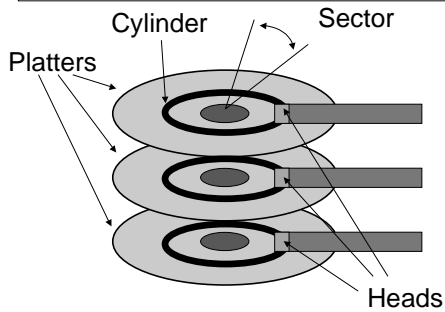
---

---

---

---

Disk Geometry




---

---

---

---

---

---

---

---

Disk Characteristics

- **Capacity**
  - Formatted
  - Unformatted
- **Speed**
  - Access time
    - Seek time
    - Rotational latency




---

---

---

---

---

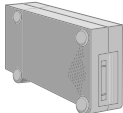
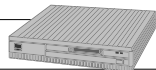
---

---

---

Magnetic Tape

- **Reel to reel**
  - 1/2-inch (1600 - 6250 bpi?)
- **Cartridge tape**
  - QIC, DLT
- **Helical scan**
  - 4mm DAT, 8mm video



Sequential access  
(block replaceable?)

---

---

---

---

---

---

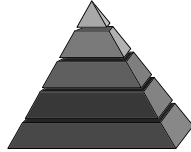
---

---

### Storage Hierarchies

• **What we want**

- Large capacity
- High speed
- Low cost



• **What we can have**

- Some compromise or combination

---

---

---

---

---

---

---

---

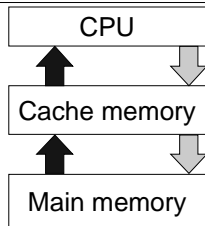
### Caching

• **Best of both worlds**

- Capacity
- Speed
- At reasonable cost

• **Transfer data as needed**

- Between different types of media



- Locality of reference
- Hit ratio
- Cache sizing

---

---

---

---

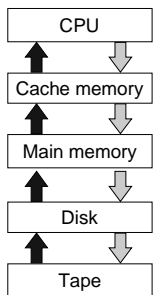
---

---

---

---

### Hierarchical Storage Management



- Known as HSM
- Extend caching concept
- Migrate most-used data to fast memory
- Move least-used data to slower (off-line?) storage

---

---

---

---

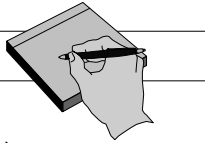
---

---

---

---

Cache Writes



• Write strategy

- Write-through (immediate)
- Delayed (write when flushed)

• Consistency

- One shared memory
- Multiple caches (invalidation)

---

---

---

---

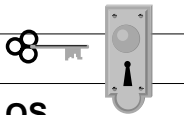
---

---

---

---

Hardware Protection



• Supports multi-process OS

- OS shielded from user processes

• Processor modes

- Dual: supervisor, user
- Rings: kernel, supervisor, user, ...
- Privileged instructions

---

---

---

---

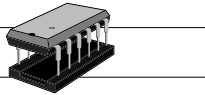
---

---

---

---

Memory Protection



• Part of memory mapping

- Implemented in hardware

• Specifies access by page

- Read/write, read, or none

• Prevents illegal access

- Triggers system trap

---

---

---

---

---

---

---

---

System Timer



- **Hardware-generated interrupt**
  - Handled by OS routine
- **Ensures OS access to CPU**
  - Interrupts even “hung” process
- **Can be used for scheduling**
  - To switch between processes

---

---

---

---

---

---

---

---

System Call Support



- **User process must call OS**
- **But not permitted access**
  - Not mapped to OS memory
- **Special user-mode opcode**
  - Acts as interrupt (INT, SWI?)
  - Switches to OS (system) state

---

---

---

---

---

---

---

---

System Call Issues

- **Memory mapping by mode?**
  - OS has its own page table
- **What if user stack corrupt?**
  - OS has own stack
  - Used by interrupts and system calls
    - System call passes parameters

---

---

---

---

---

---

---

---