Operating Systems W7L1 - Memory Management I

Class	Operating Systems
Created	@Oct 12, 2020 1:07 PM
Materials	05 - Memory Management I.pdf
Reviewed	☐
Type	Lecture

> ! Approximately first five minutes of class was spent discussing the midterm exam format and such.

General Info on Memory

- Creating the illusion of all memory is a more difficult task
- There are seeral types of memory (on all machines) that are designed with different purposes in mind (i.e. cache memory and disk memory are different)
- The two big things with memory are *size* and *speed*
- Memory is 1000x slower than processor and disks are 1000x slower than memory, so there's a bit of a mismatch
- Privacy of memory is also something we want to keep in mind

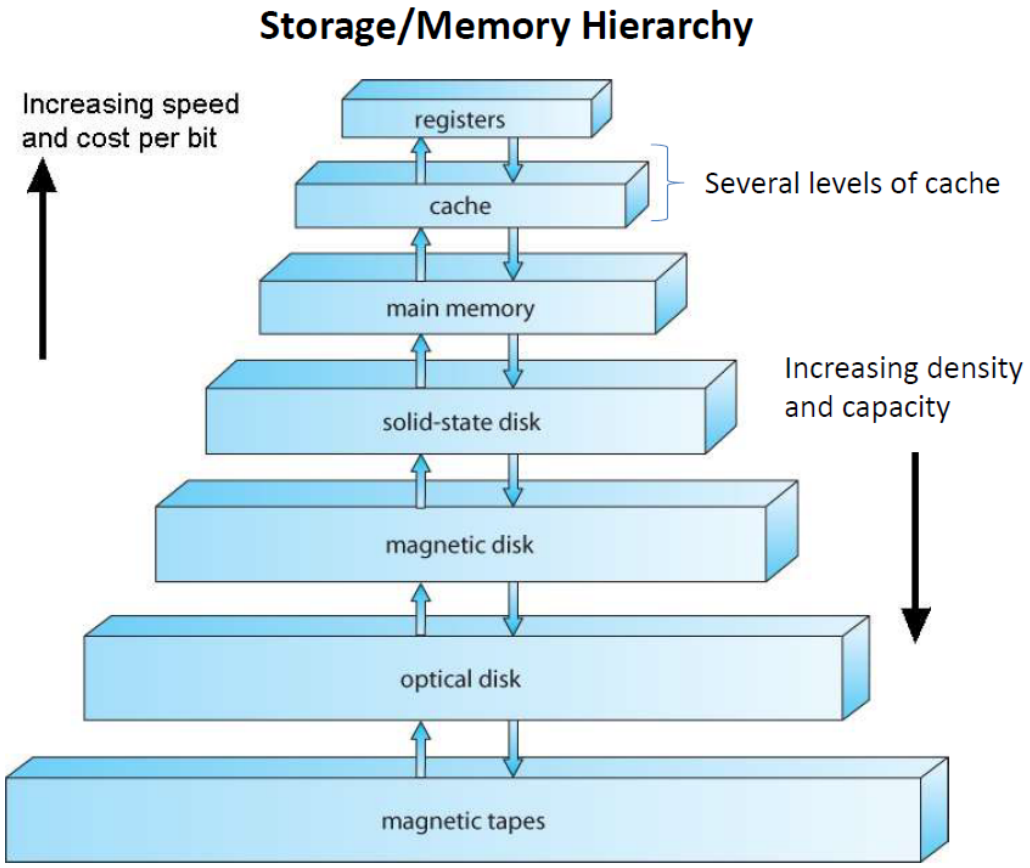
▼ What are the 5 concepts of memory that would be a programmer's dream / exist in an ideal world?

- Private

    - Infinitely large
    - Infinitely fast
    - Non-volatile
    - Inexpensive
- All computers have Dynamic RAM (DRAM) and Static RAM (SRAM)
    - **DRAM:** High capacity, but a bit slower (due to refreshing)
    - **SRAM:** Low capacity, but high speed (no refreshing)
- Non-volatile memory sounds good, but it is also a bit of a hassle for the OS to deal with (i.e. stored info may not be the most recent)

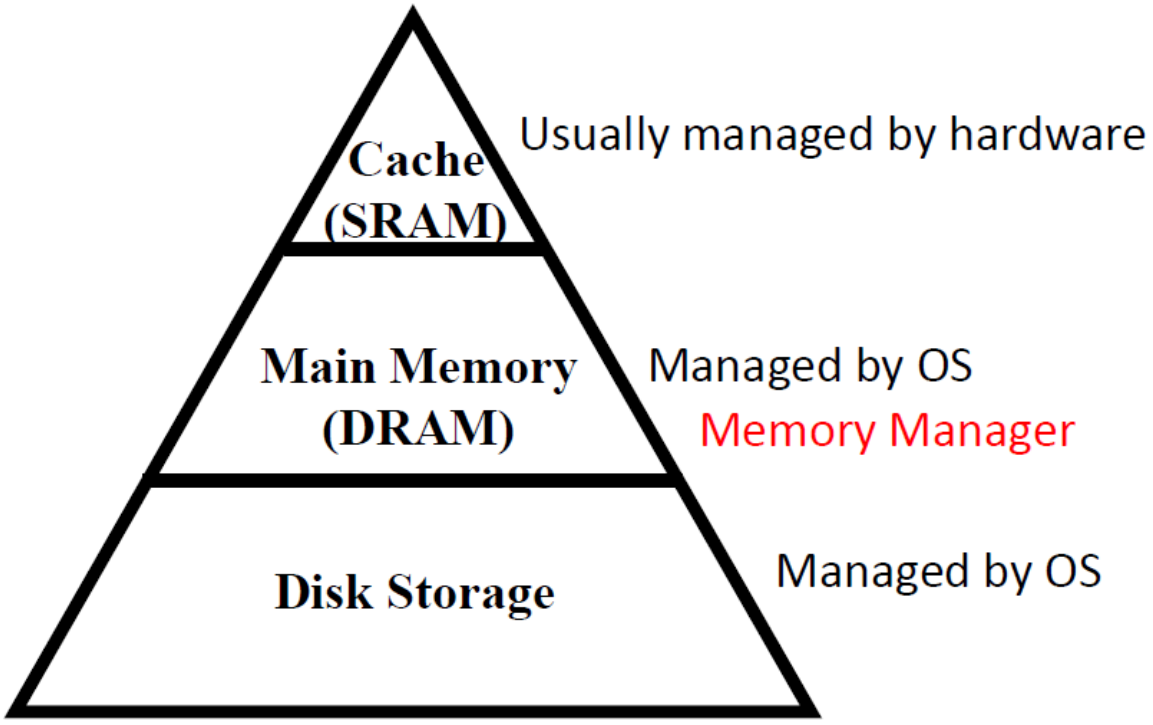
▼ Storage and Memory Hierarchy

Main memory is DRAM, cache is SRAM



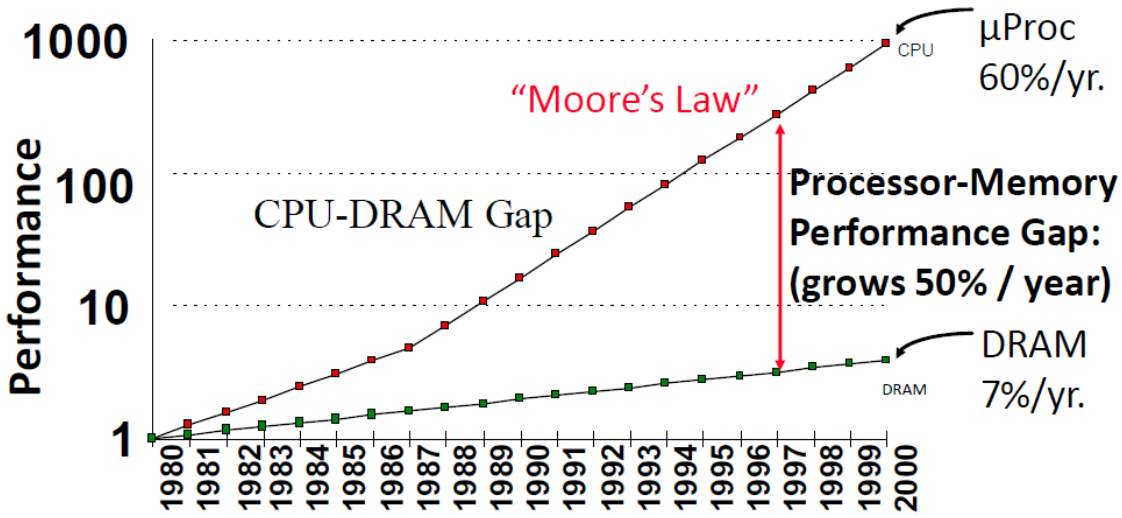
- SSDs are faster because there is no mechanical movements → This is quite literally where the name "solid state" is derived from

▼ Memory hierarchy we are "concentrating on" (i.e. simplified)



- Recall Moore's law. The speed of memory improvements are nowhere *near* as fast as processor speeds

  ▼ Graph Comparison Gap



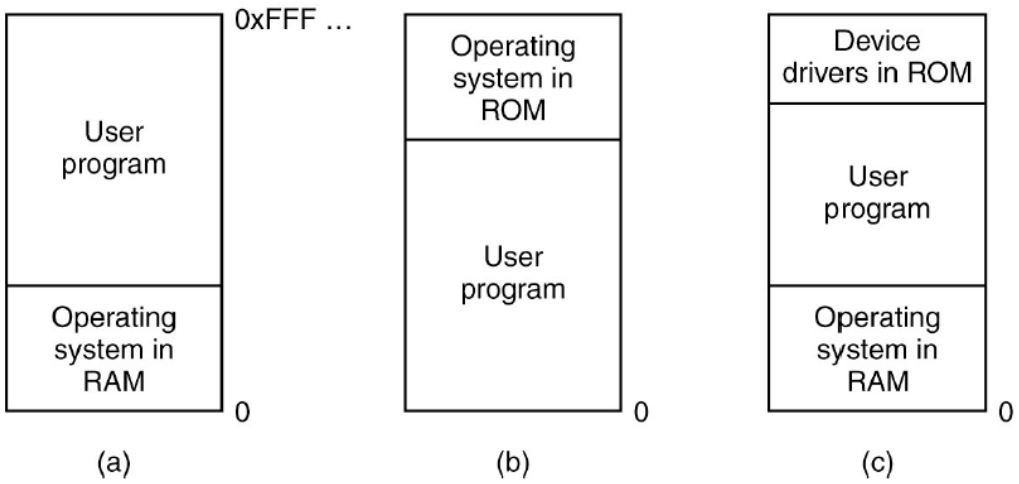
  - The focus on speed is partially economical and psychological → People will buy bigger memory, regardless of brand or such, so brands don't feel more pressure to make "reliable" or "the best" memory

- Speed of memory is bringing data into processor, reading and writing from memory, which is why it's so crucial

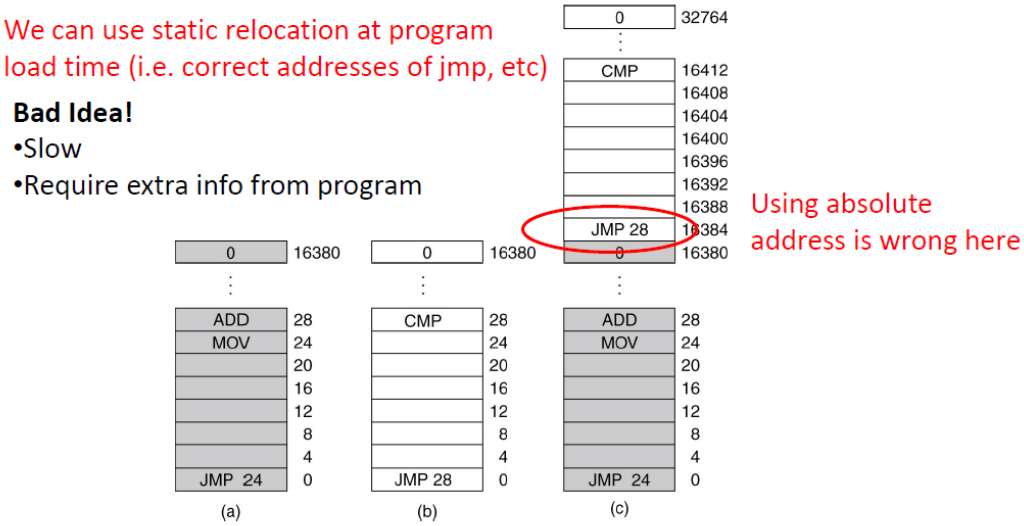
Memory Abstraction

- Hardware and OS memory manager sees memory as a contiguous entity → **Abstraction**
- Memory access and wired connections are the two slowest things in any machine

▼ No Memory Abstraction Diagrams



Only one process at a time can be running (threads??)



Bottom line: Memory abstraction is needed!

- Compilation, linking, and coding become very complex without abstraction since then these things would have to keep track of what is going on "around" them
- Without abstraction, we would be swapping- an extremely slow process due to disk (more on this in later notes)
- As a result of swapping, memory abstraction is absolutely necessary (see above diagrams)

- **Address Space:** New abstraction of memory. A set of addresses that a process can use to address memory.

Protection

- Despite a process thinking it has all the memory, we need to keep in mind that under the hood you may end up accessing real memory that is in use by another process / outside of the process being run
  - This is how a **seg fault** occurs
- Memory references must be checked at runtime to avoid things such as overwiting

Relocation

- Programmers don't know- and don't need to know- about processes around their process at execution
- To maximize processor utilization, active processes need to be bale to be swapped in/out of main memory
- When a process is brought back into memory, it may not optimally "fit" back in, and thus is "relocated" into other memory addresses

Sharing

- Advantageous to allow processes toa ccess same libraries versus a bunch of individual copies
  - However, there is a risk of mutual exclusion here
- Controlled access to shared areas is necessar without compromising protection

Local Organization

- Memory as lieanr 1D address space
- Program = code + data + ... = module
- These *modules* must be organized in an address space of logical order

Physical Organization

- Memory is a hierarchy

- Making use of physical organization to manage modules of different programs in efficient ways

---

▼ What are 3 things that must *constantly* be ensured about memory management?

1. **Transparency:** The running programs must not know that all of this memory virtualization and re-organization is happening
2. **Efficiency:** Both in terms of time (speed) and space (not wasting a lot of memory)
3. **Protection:** As we saw, protecting processes from each other (i.e. seg faults) is crucial

Address Space: Base and Limit

- Left off on this slide and will start back up here next class!