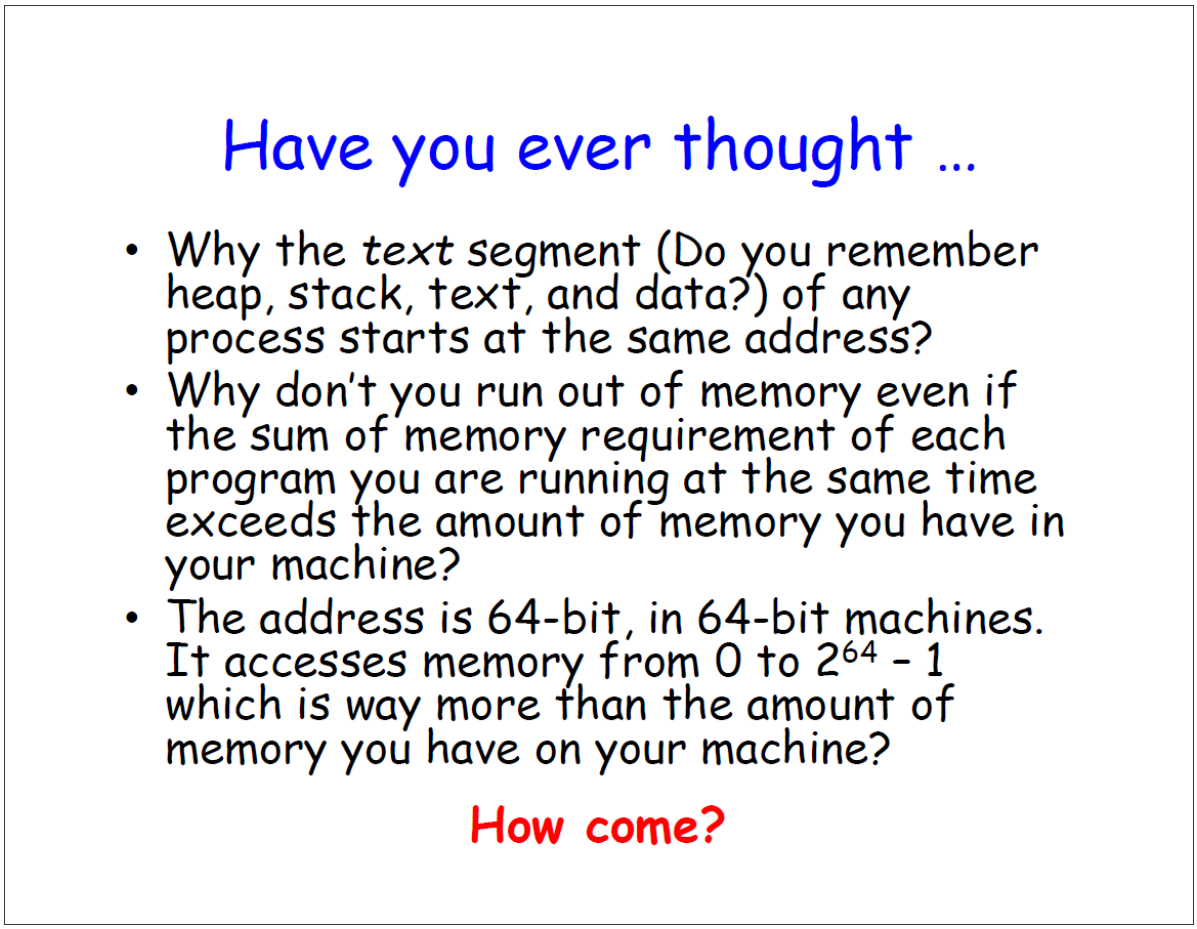
Operating Systems W9L1 - Memory Management II

Class	Operating Systems
Created	@Oct 26, 2020 12:29 PM
Materials	06 - Memory Management II.pdf
Reviewed	☐
Type	Lecture

Starting Discussion

- Each process has its own virtual memory address space
- Having "enough" memory is never really possible
- Processor does not execute anything not in memory

▼ Some food for thought



- All references are logical/virtual addresses then translated into a physical address at runtime
- A process can be broken up / not contiguos in memory
  - The entire code doesn't need to be in memory — **90/10 rule**

Virtual Memory

Mapping from logical/virtual address space to physical address space

- All cache memory access is physical → Virtual memory is OS and hardware, not just OS
  - The hardware is often faster but the software is more flexible, so a combination of both is ideal
- Review cache stuff from CSO as we won't really be going over that in-depth

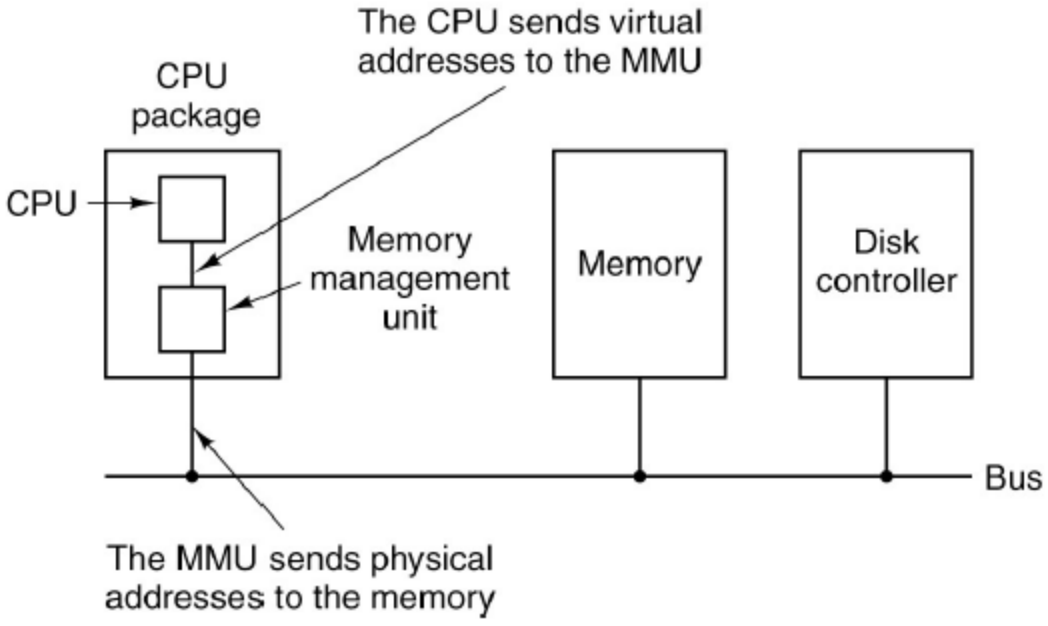
- Virtual memory can be implemented via paging

▼ The Story (and plotholes)

1. Operating system brings into main memory a few pieces of the program. What do you mean by "pieces"?
2. An interrupt is generated when an address is needed that is not in main memory. How do you know it isn't in memory?
3. Operating system places the process in a blocking state.
4. Operating system issues a disk I/O Read request. Why?
5. Another process is dispatched to run while the disk I/O takes place.
6. An interrupt is issued when disk I/O is complete, which causes the operating system to place the affected process in the Ready state
7. Piece of process that contains the logical address is brought into main memory. What if memory is full?

- Address space (per program) is divided into *pages*

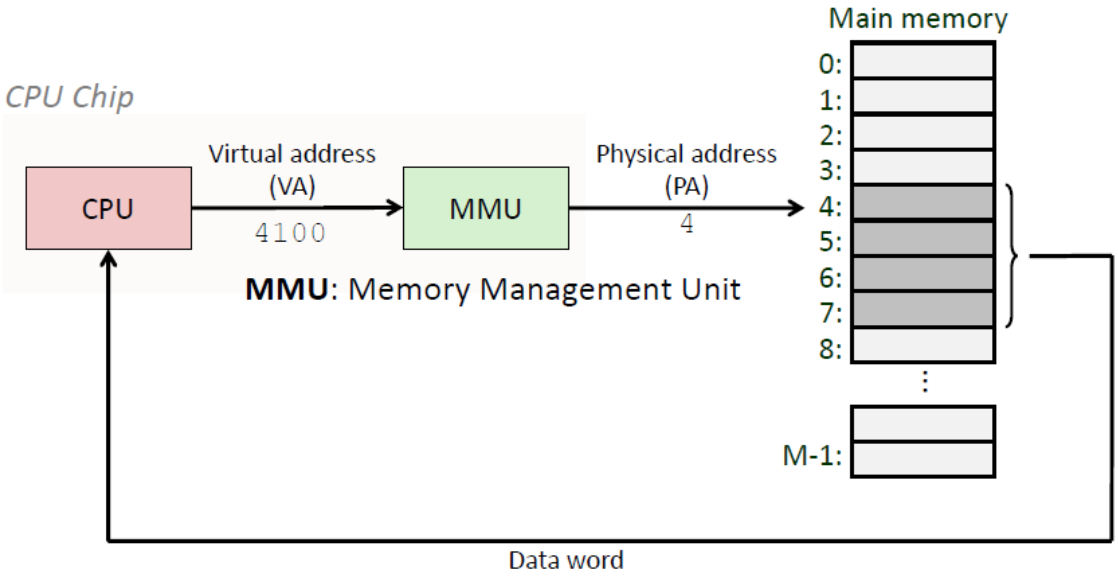
▼ Pages mapped to physical memory diagram



- If, for every virtual address, we have a table for translating every byte, that table would be huge

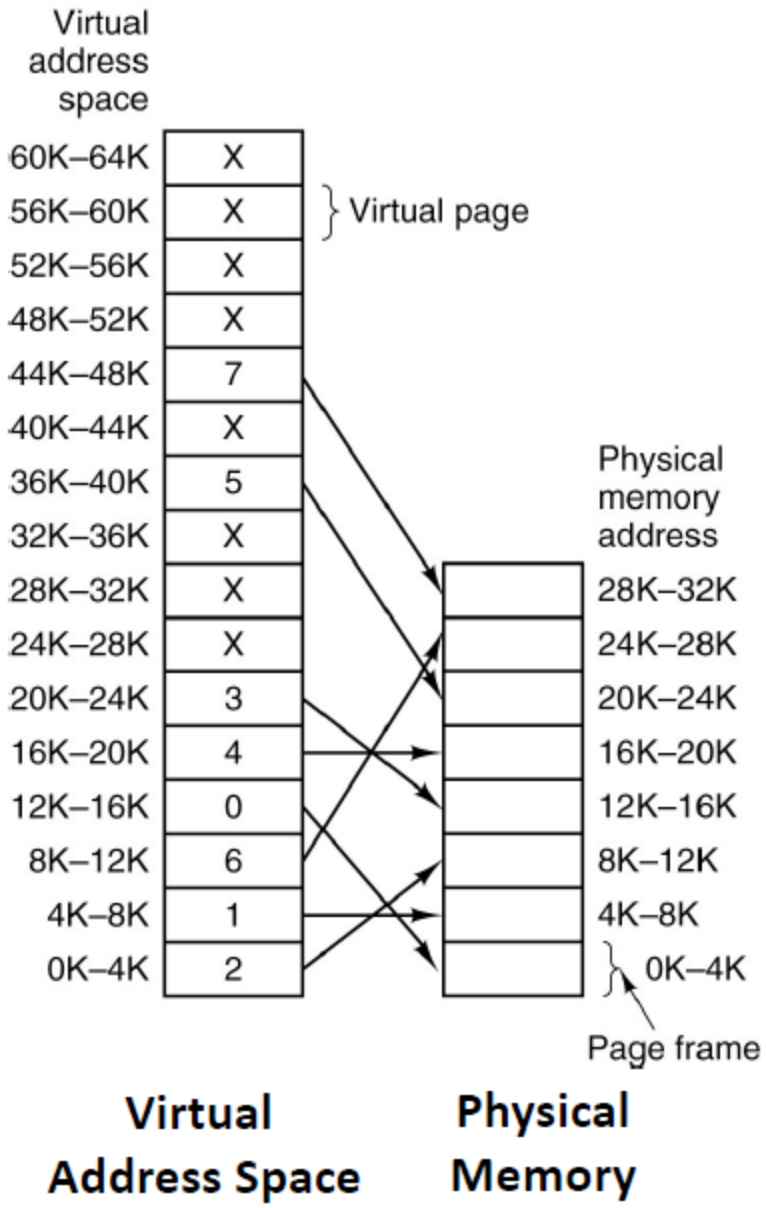
    - Thus, it is done each consecutive bytes using pages, i.e. arrays are stored so you need (1) index and (2) what array (i.e. *page*) you are accessing in the table
    - We are concerned both about memory used by the table *and* how long it would take to search through
- The MMU is hardware that helps the OS search the table
    - In what way is hardware advantageous over OS? OS over hardware? *See earlier note for answer*

▼ Virtual Addressing Diagram



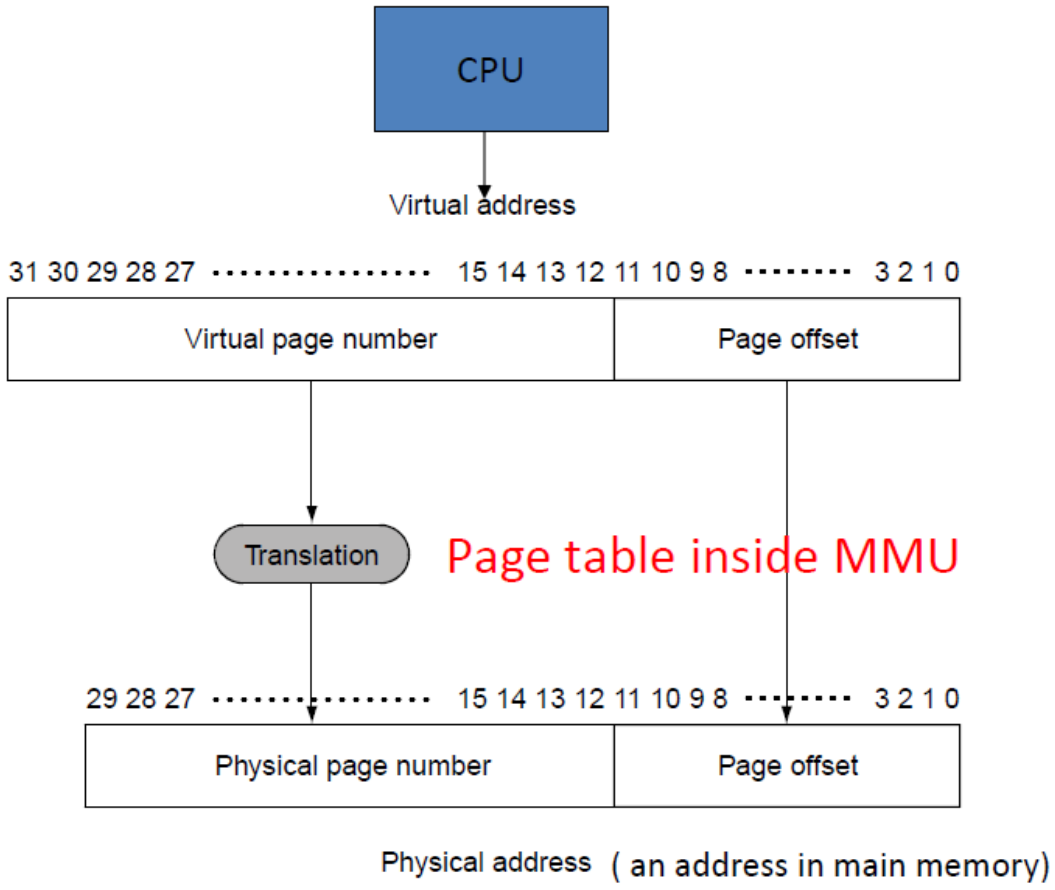
- Used in all modern machines.

▼ Virtual to Physical Address Space



- Memory is, analagously, a cache to secondary storage (i.e. disk)
  - ▼ What are the 3 advantages to this virtualized memory?
    1. Illusion of more physical memory
    2. Program relocation is smoother
    3. Protection
- ▼ CPU t ovirtual address to page translation diagram

Page offset differs from system to system.



MMU = Memory Management Unit