CS 240
Foundations of Computer Systems

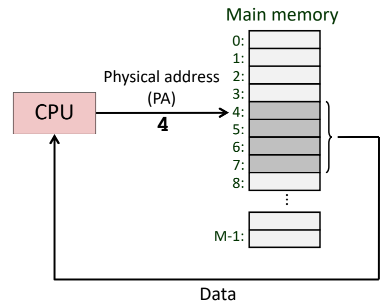
Virtual Memory

Process Abstraction, Part 2: Private Address Space

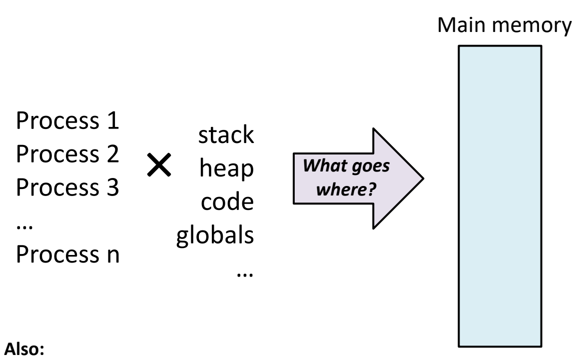
Motivation: why not direct physical memory access?
Address translation with pages
Extra benefits: sharing and protection

Memory as a contiguous array of bytes is a lie! Why?

Problems with physical addressing

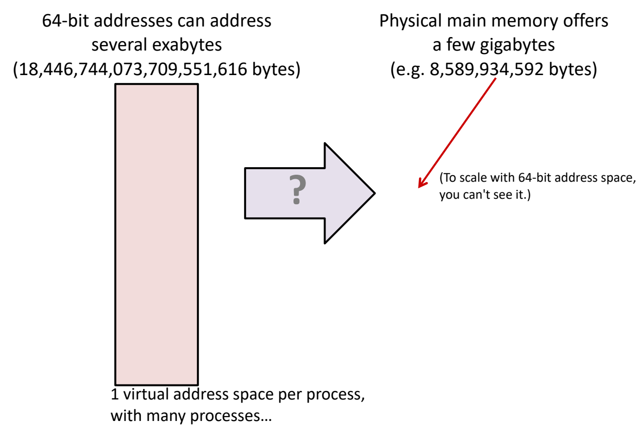


Problem 1: memory management

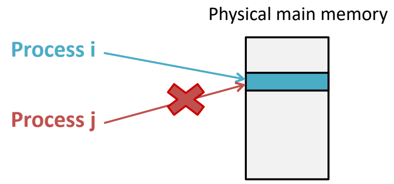


Also:
Context switches must swap out entire memory contents.
Isn't that **expensive**?

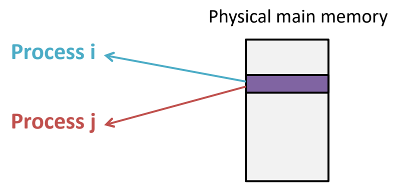
Problem 2: capacity



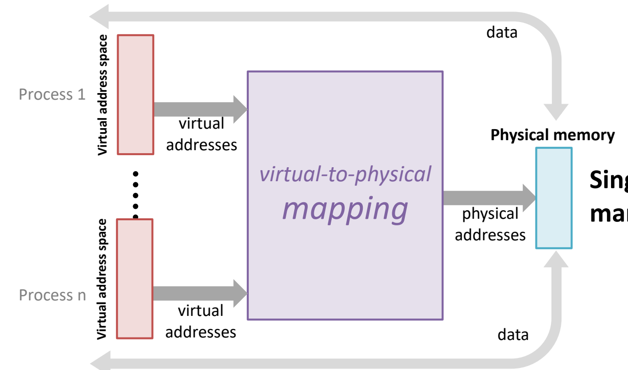
Problem 3: protection



Problem 4: sharing



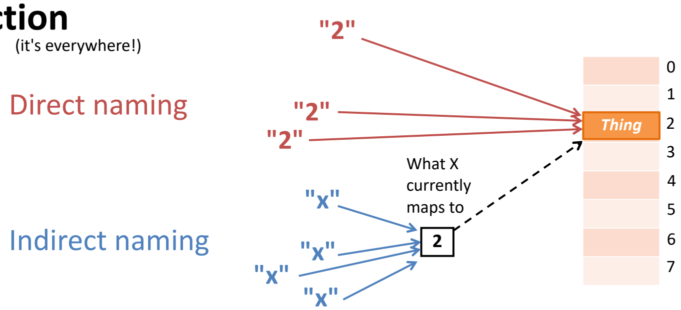
Solution: Virtual Memory (address *indirection*)



Single physical address space managed by OS/hardware.

Indirection

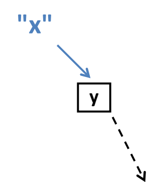
(it's everywhere!)



What if we move ***Thing***?

Tangent: **indirection everywhere**

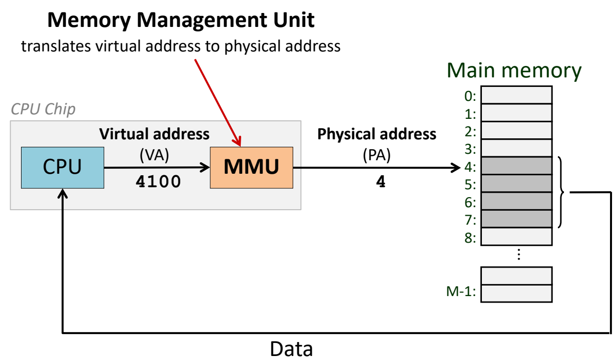
- Pointers
- Constants
- Procedural abstraction
- Domain Name Service (DNS)
- Dynamic Host Configuration Protocol (DHCP)
- Phone numbers
- 911
- Call centers
- Snail mail forwarding
- …



"Any problem in computer science can be solved by adding another level of indirection."
–David Wheeler, inventor of the subroutine, or Butler Lampson

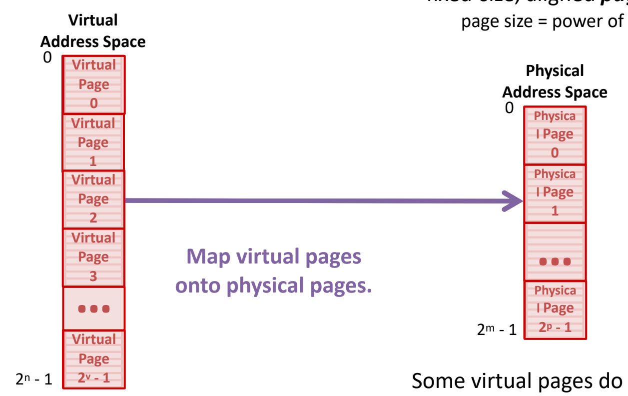
Another Wheeler quote? "Compatibility means deliberately repeating other people's mistakes."

Virtual addressing and address translation



Page-based mapping

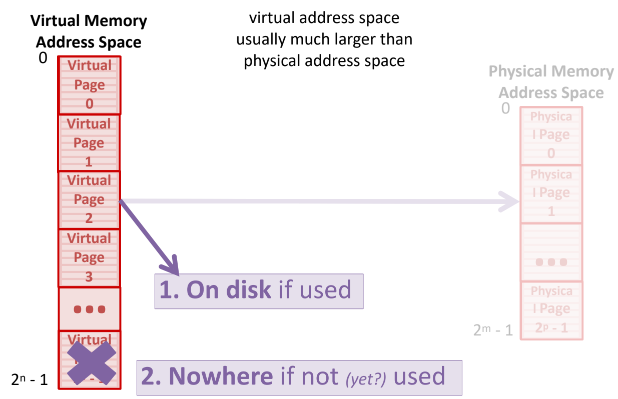
fixed-size, aligned ***pages***
page size = power of two



Some virtual pages do not fit!
Where are they stored?

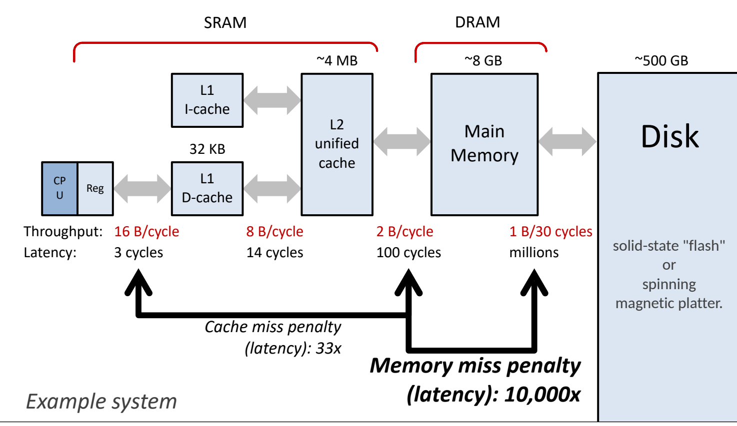
Cannot fit all virtual pages! Where are the rest stored?

virtual address space
usually much larger than
physical address space

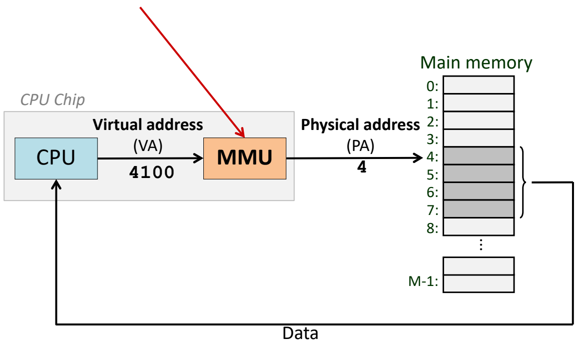


Virtual memory: cache for disk?

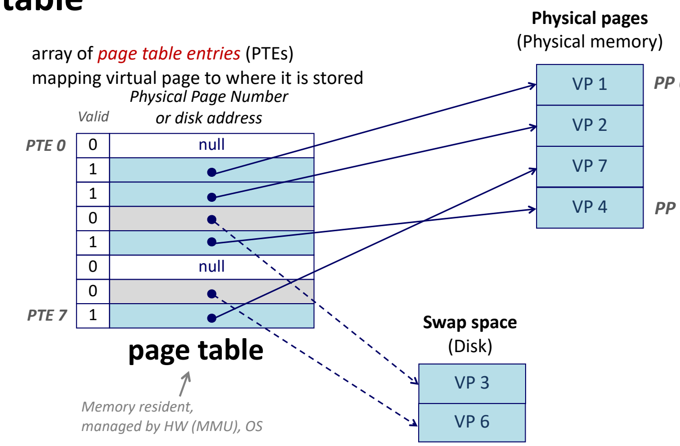
Not drawn to scale!



Address translation

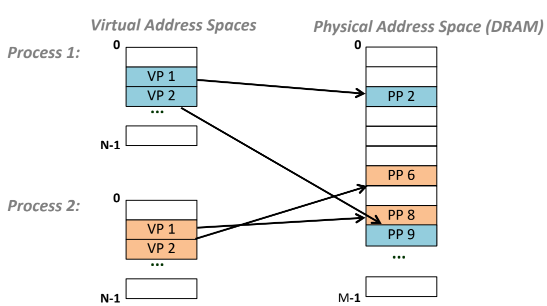


Page table



Virtual memory benefits:
Simple address space allocation

Process needs private *contiguous* address space.



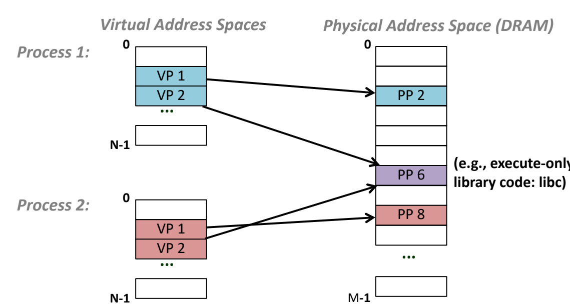
Virtual memory benefits:

Protection:

All accesses go through translation.
Impossible to access physical memory not mapped in virtual address space.

Sharing:

Map virtual pages in separate address spaces to same physical page (PP 6).



Virtual memory benefits:
Memory permissions

MMU checks on every access.
Exception if not allowed.

permission bits

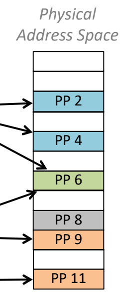
Process 1:	Valid	READ	WRITE	EXEC	Physical Page Num
VP 0:	Yes	No	No	Yes	PP 6
VP 1:	Yes	No	No	Yes	PP 4
VP 2:	Yes	Yes	Yes	No	PP 2

Page Table

permission bits

Process 2:	Valid	READ	WRITE	EXEC	Physical Page Num
VP 0:	Yes	Yes	Yes	No	PP 9
VP 1:	Yes	No	No	Yes	PP 6
VP 2:	Yes	Yes	No	No	PP 11

Page Table



Summary: **virtual memory**

Programmer's view of virtual memory

Each process has its own private linear address space
Cannot be corrupted by other processes

System view of virtual memory

Uses memory efficiently (due to locality) by caching virtual memory pages
Simplifies memory management and sharing
Simplifies protection -- easy to interpose and check permissions
More goodies:

- Memory-mapped files
- Cheap `fork()` with copy-on-write pages (COW)