

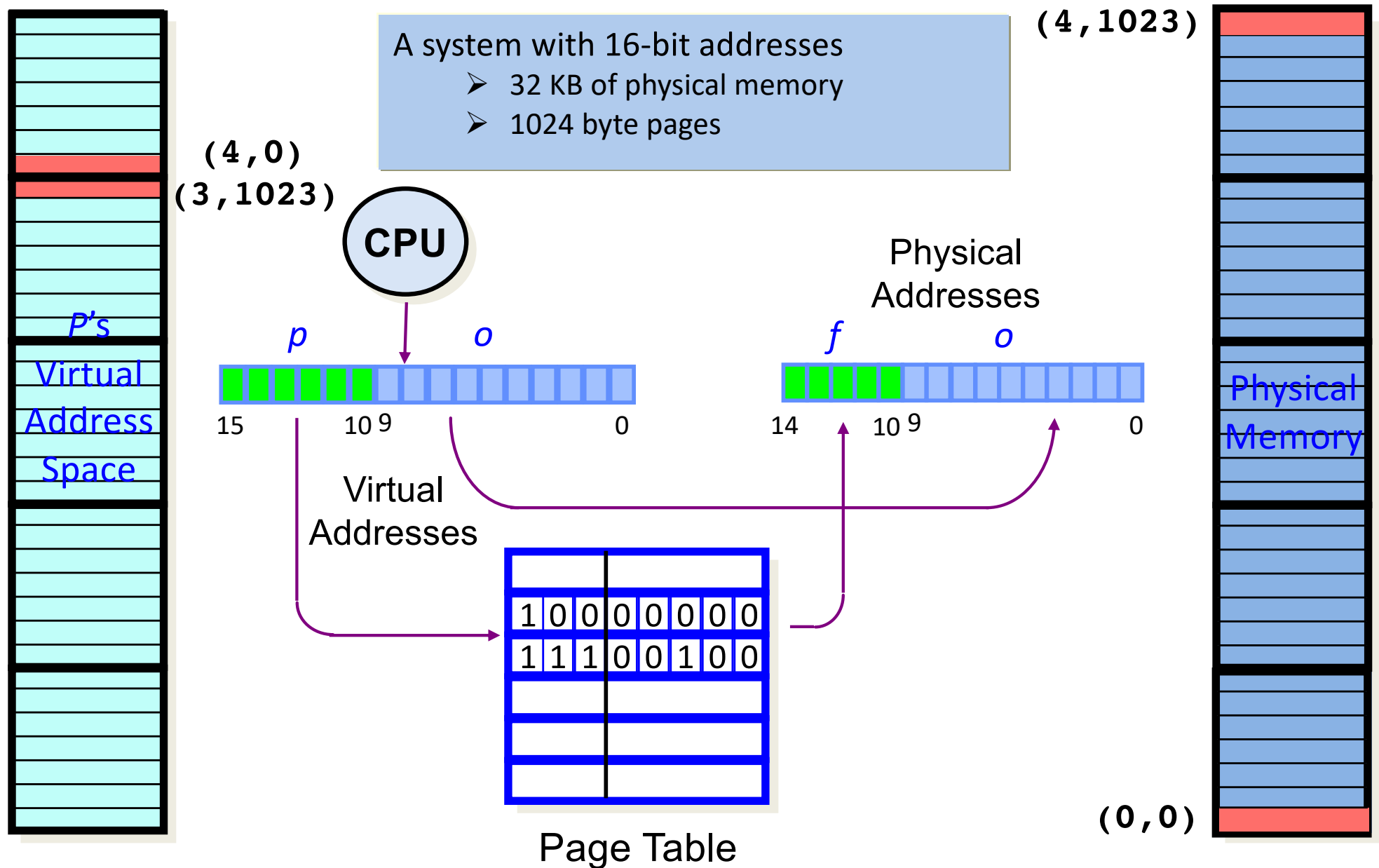
Virtual Memory

CS439: Principles of Computer Systems

Bringing It Together

- Processes are the OS's main abstraction for protection
- A process defines an address space---but this is a virtual/logical address space
- Virtual address spaces are mapped onto physical address spaces in physical memory
- All virtual addresses range from 0->max for that process
- *Paging* describes how the OS provides the illusion of infinite memory

Address Translation: Current Picture



Vocabulary

transparency	offset	page tables
safety	address translation	frames
frame number	segmentation fault	dirty bit
memory exception	PTBR	swapping
virtual address	paging	physical address space
disk	page size	pages
resident bit	physical address	efficiency
internal fragmentation	shared memory	page number
bitmap	memory	external fragmentation
frame size	swap partition	
process	swap slot	
clock/reference bit	virtual address space	

In-Class Problem

Assuming you are in a byte-addressed system, and given a virtual address size of 12 bits, a page size of 256 bytes, and a physical address space sized 2KB:

1. How many bits of the virtual address are devoted to the offset?
2. How many frames are contained in physical memory?
3. Assuming this is a single-level paged system, how many pages does each process have?

Performance Issues: Speed

Problem: Virtual memory references require two memory references!

- One access to get the page table entry
- One access to get the data

Performance Issues: Space

- Two sources:
 - data structure overhead (the page table)
 - fragmentation (*How large should a page be?*)
- Page table can be very large!
 - 32-bit addresses, 4096-byte pages... 1 million pages!
 - 32-bit addresses imply a VAS of 2^{32} bytes
 - Divide VAS into pages ($2^{32}/2^{12}$)
 - 1 million pages (2^{20})
- AND page tables are *per process*.

How can we improve the space efficiency of paging?

Right now:

- 1 page table per process
- Page table is a one-dimensional array
- 1 entry for every page possible in the virtual address space
 - Index is the page number
 - Allocated or not! (When might a page be unallocated?)

What can we do?

Consider

- What changes could we make to the page table data structure in order to improve space efficiency? Consider:
 - Do we need entries for unallocated pages?
 - How do we maintain time efficiency as well?
- For our solution, how would we support address translation?