

Goals

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
- Relocation techniques place a process contiguously in physical memory
- Virtual addresses must be translated to physical addresses
- The *-fit algorithms are used to choose *where* to allocate a process's address space in physical memory
 - All had problems with external fragmentation
- Compaction reduces external fragmentation
- Swapping allows more processes to run than would otherwise fit in memory

Goals

- To allow processes to use more memory than that which is physically available
- Reduce or eliminate external fragmentation
 - We can use compaction, but is there another way?
- Easy to allocate and de-allocate memory to/from processes
 - The ability to grow processes easily
- Ability to easily share memory between processes
- Fine-grained protection

Paging Fundamentals

Virtual Memory: Today's Version

- Process uses a virtual address space
 - May be much larger than available physical memory
- Only portions of the virtual address space are in physical memory at any one time
 - Virtual address space is automatically divided into equal pieces
 - Needed pieces are automatically brought into the memory (from where?)
- This technique is known as *paging*

Paging

1. Divides a process's virtual address space into fixed-sized *pages*
2. Views the physical memory as a series of equal-sized *page frames* (or *frames*)
3. Stores the pages in frames in memory
 - *When* that happens is a policy decision
4. Manages the pages in memory
 - Is it time for it to be removed? Is a page still in use? is it written? ...

Virtual Memory: Pages

$$2^n - 1 =$$

$$(p_{MAX} - 1, o_{MAX} - 1)$$

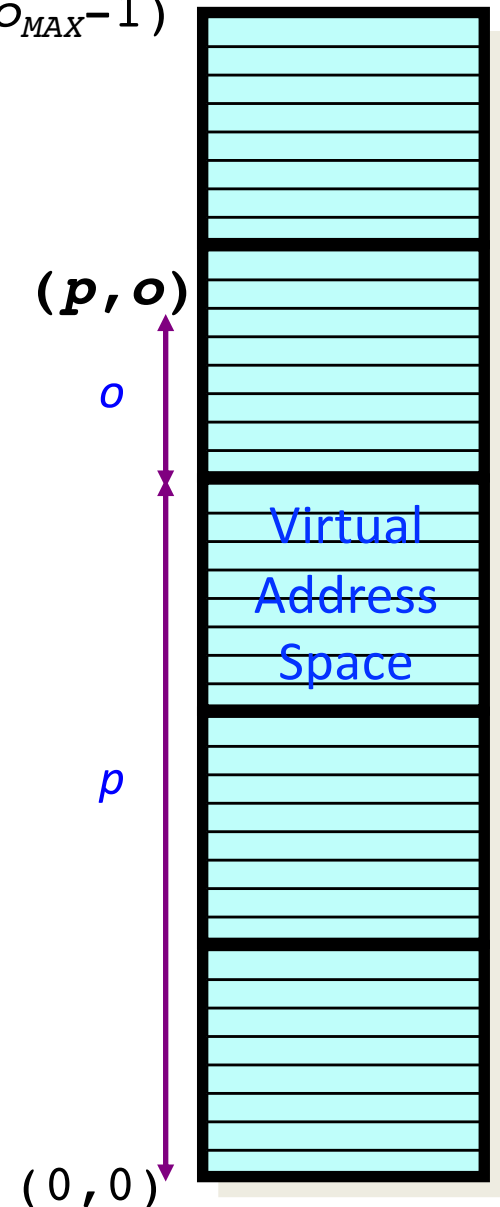
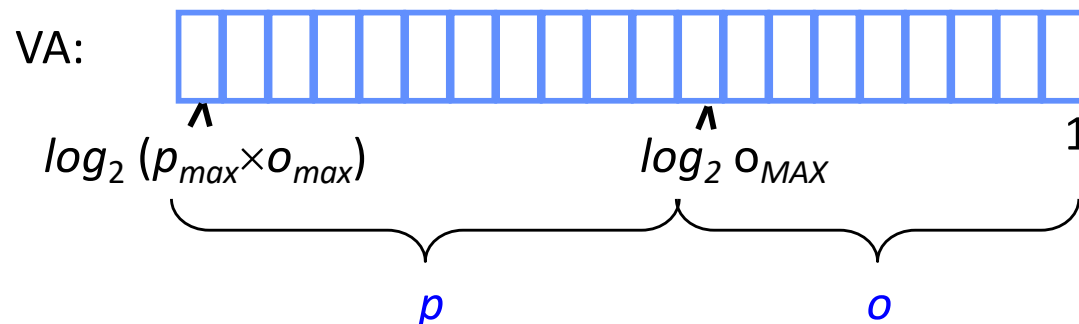
A process's virtual address space is partitioned into equal sized *pages*.

A virtual address is a pair (p, o)

p — page number (p_{max} pages)

o — page offset (o_{max} bytes/pages)

$$\text{Virtual address} = o_{max} \times p + o$$



Page Frames

Physical memory is partitioned into equal sized *page frames*. $(f_{MAX}-1, o_{MAX}-1)$

$$|page| = |page\ frame|$$

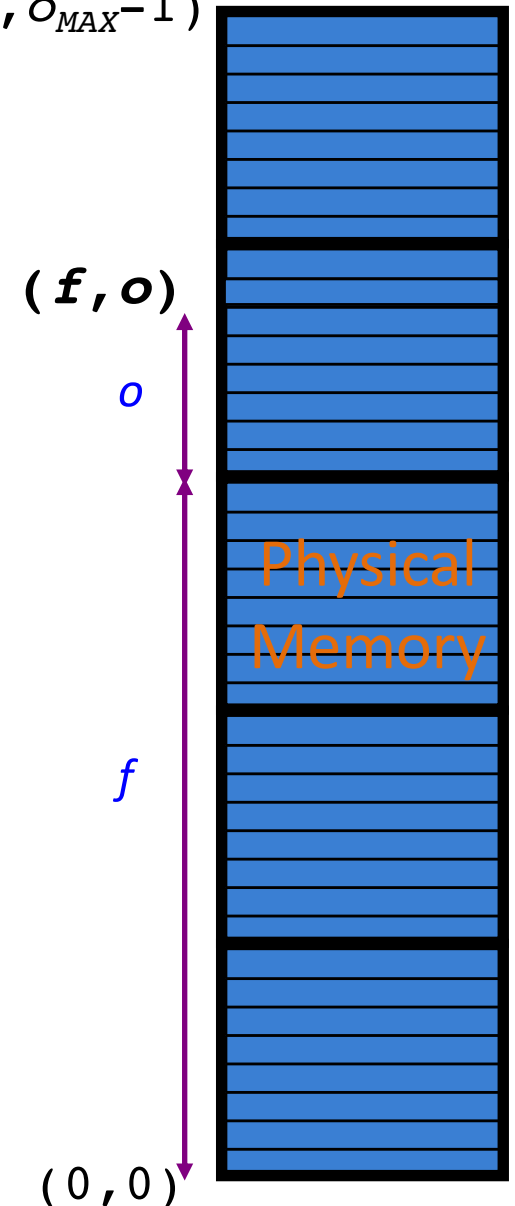
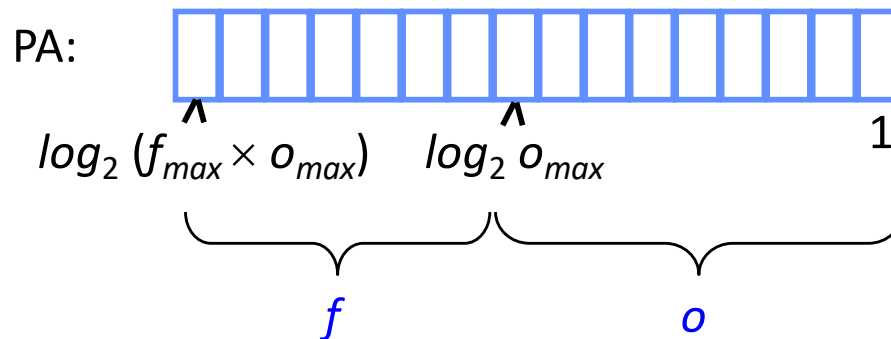
The system's page size is equivalent to its frame size.

A memory address is a pair (f, o)

f — frame number (f_{max} frames)

o — frame offset (o_{max} bytes/frames)

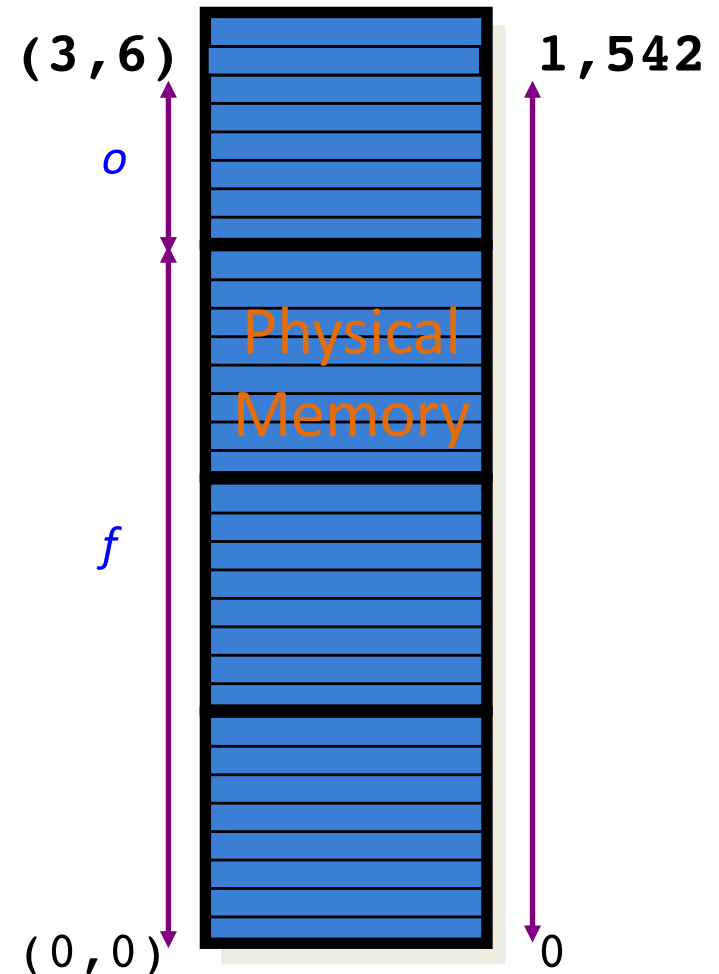
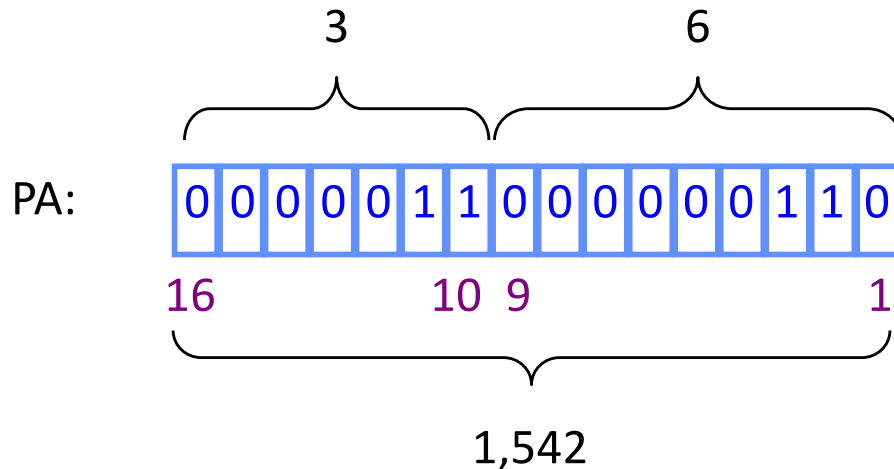
$$\text{Physical address} = o_{max} \times f + o$$



Physical Addresses (Frame/Offset Pairs)

Example: A 16-bit address space with
(o_{max} =) 512-byte page frames

Addressing location (3, 6) = 1,542



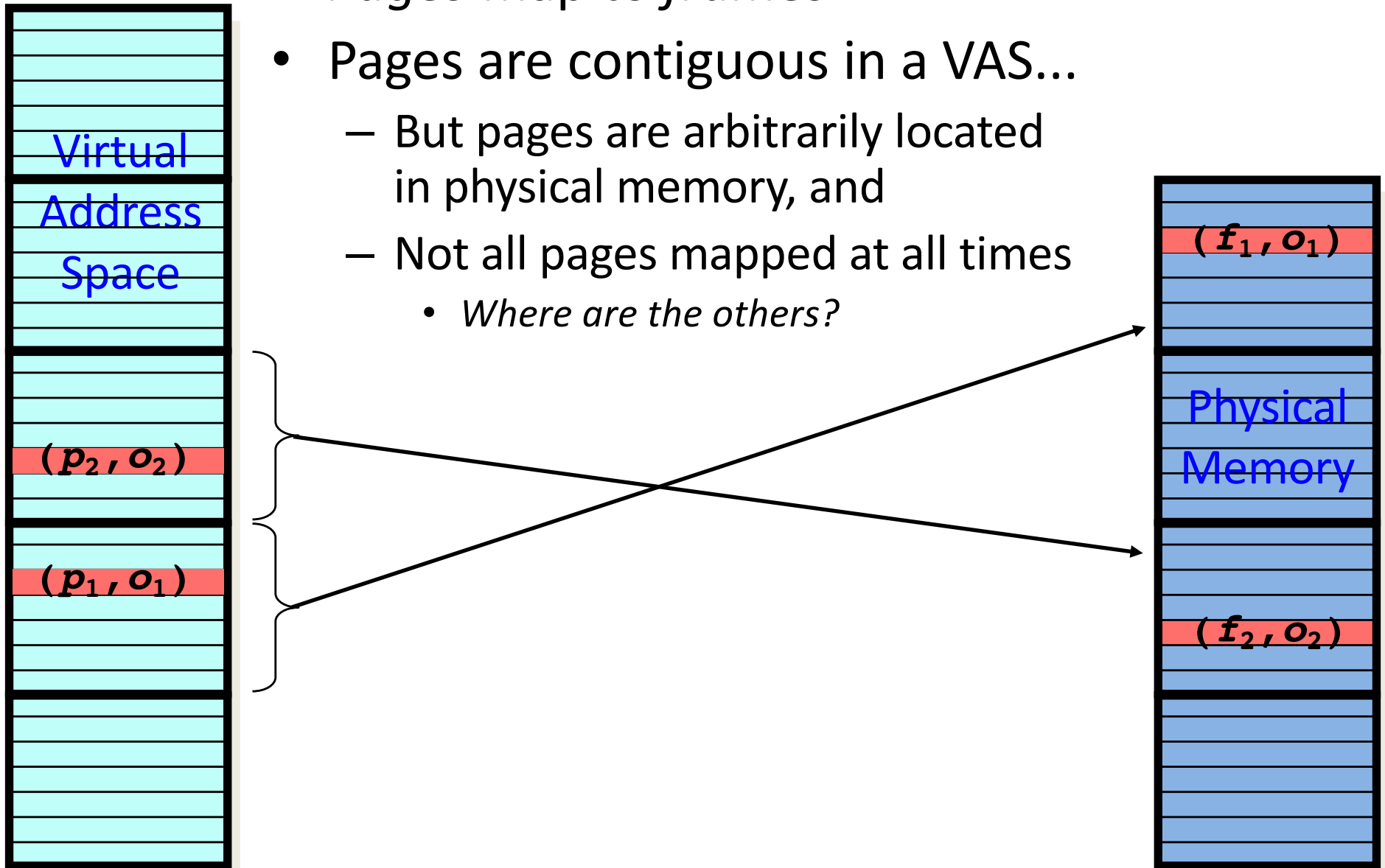
iClicker Question

Paging has what advantage over relocation?

- A. Easier to manage transfer of pages from disk
- B. Requires less hardware support
- C. No external fragmentation

From Virtual to Physical: Allocation Policy

- *Pages map to frames*
- Pages are contiguous in a VAS...
 - But pages are arbitrarily located in physical memory, and
 - Not all pages mapped at all times
 - *Where are the others?*



Ummmm...

How do we find the data when pages are not allocated contiguously in memory?

Address Translation

Ummm...

How do we find the data when pages are not allocated contiguously in memory?

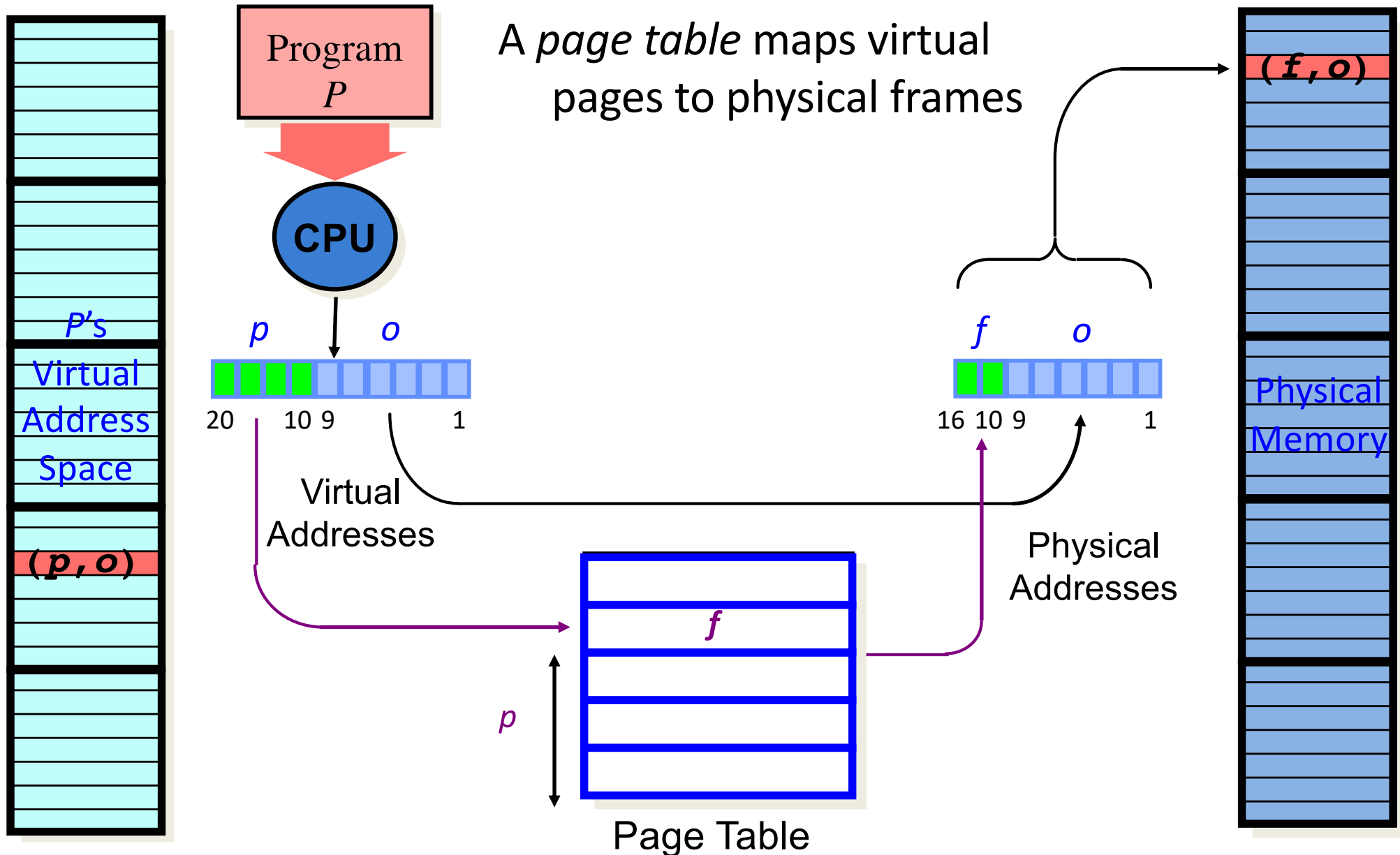
How do we translate the virtual address to a physical one?

How do we track where a page is in physical memory?

A page table keeps track of the mapping of pages to page frames

- You can think of the page table as a set of relocation registers, one for each frame
- Mapping is invisible to the process
- Protection is provided with the same mechanisms as used in dynamic relocation

Virtual Address Translation



Advantages of Paging

So... How are we doing?

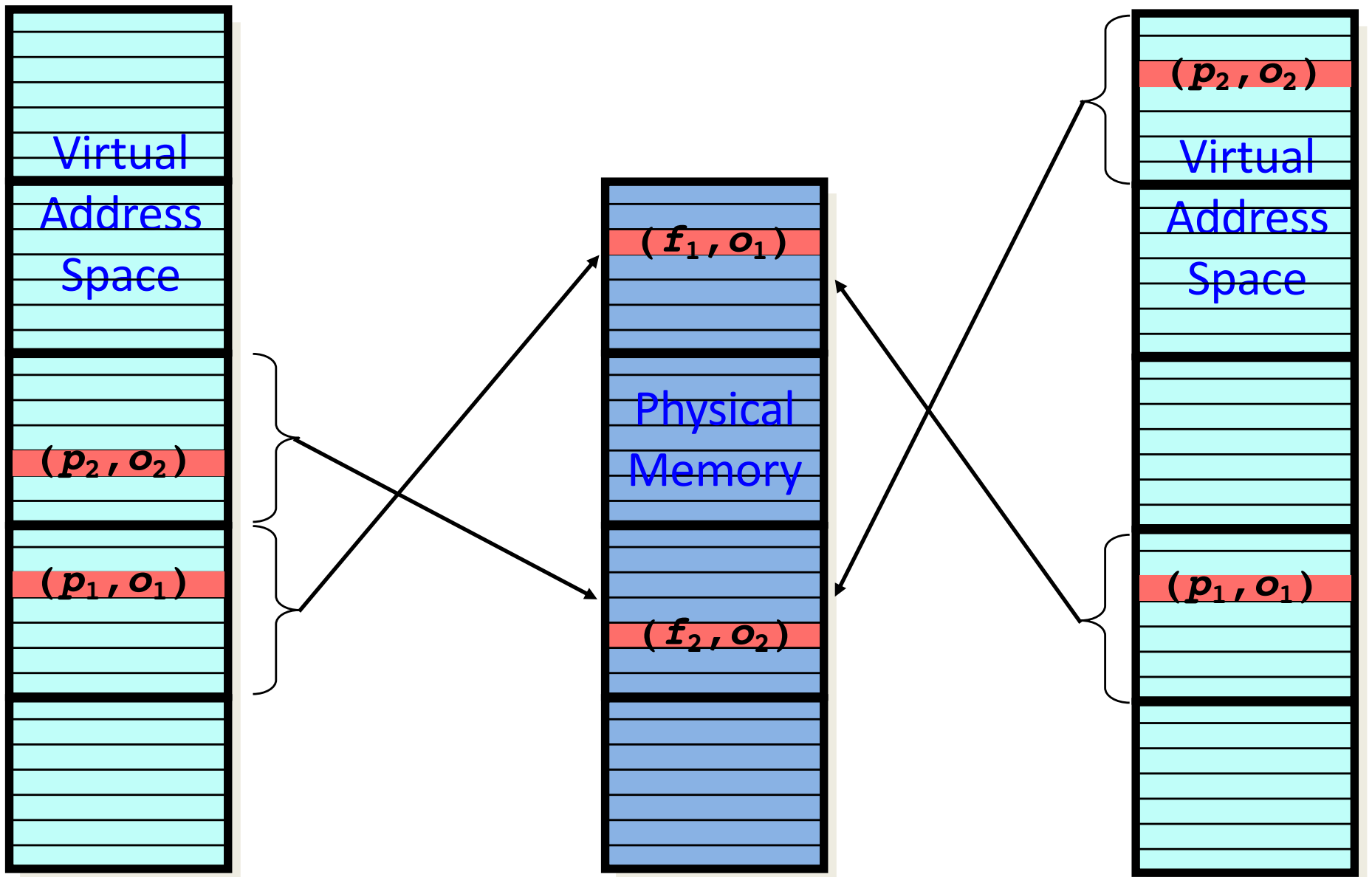
Wanted:

- reduce or eliminate external fragmentation
- to allow processes to use more memory than that which is physically available
- easy to allocate and de-allocate memory to/from processes
 - the ability to grow processes easily
- ability to easily share memory between processes
- protection

Advantages of Paging: Sharing

- Paging allows shared memory
 - the memory used by a process no longer needs to be contiguous
- A shared page may exist in different parts of the virtual address space of each process, but the virtual addresses map to the same physical address

Sharing

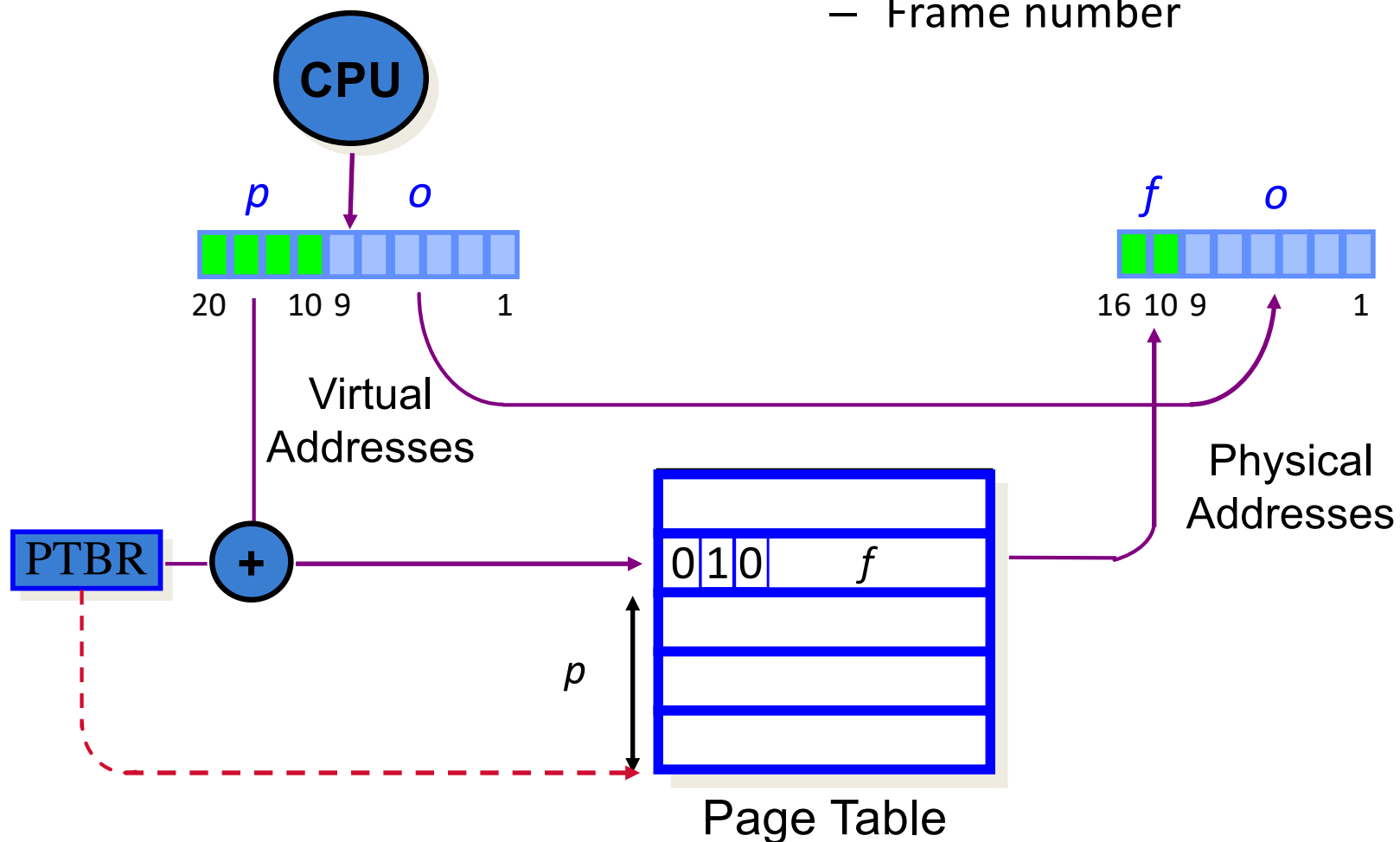


Virtual Address Translation: Protection and Consistency

1 table per process
Part of process's state

Page Table Entry Contents:

- Flags: dirty bit, resident bit, clock/reference bit
- Frame number



Address Translation: Current Picture

