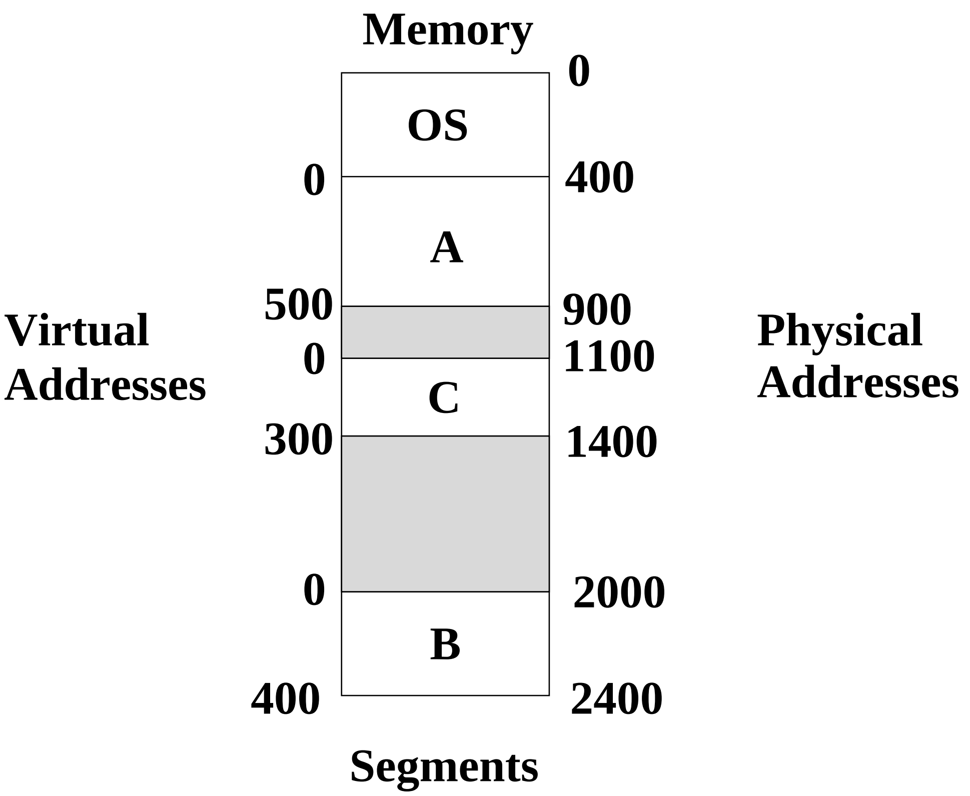
Announcements/Reminders

- Lab 3 due Wednesday
- Lab 3 demos are due no later than 1 week from Wednesday
- I am out of town next week (so this is the week to ask questions in preparation for the exam)

Last Class: Memory Management

- Uniprogramming
- Static Relocation
- Dynamic Relocation
- Monolithic memory segments



Next: Paging

Observation: Processes typically do not use their entire space in memory all the time.

Paging:

1. Divides and assigns processes to fixed sized *pages* (logical blocks of memory),
2. then selectively allocates pages to *frames* in the physical memory, and
3. manages (moves, removes, reallocates) pages in memory.

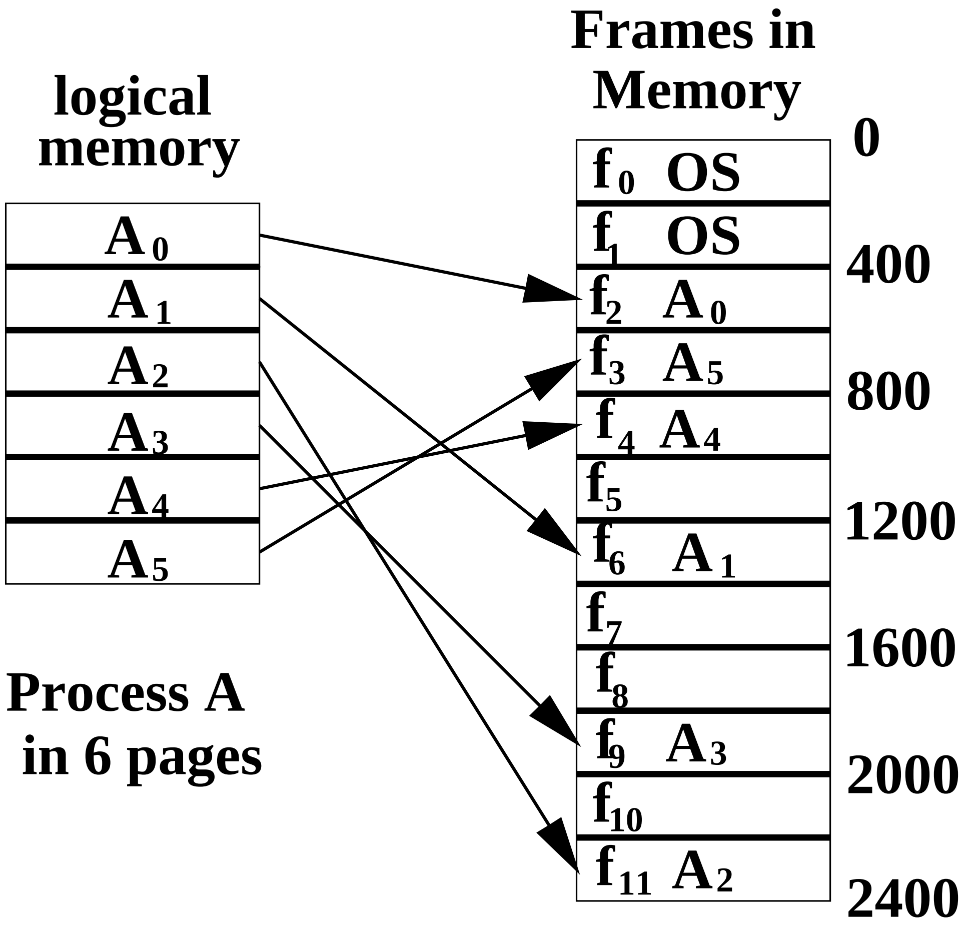
Paging: Motivation & Features

90/10 rule: Processes spend 90% of their time accessing 10% of their space in memory.

$\Rightarrow$ Keep only those parts of a process in memory that are actually being used

- Pages greatly simplify the hole fitting problem: all pages are interchangeable
- The logical memory of the process is contiguous, but pages need not be allocated contiguously in memory.
- By dividing memory into fixed size pages, we can eliminate external fragmentation.
- Paging does not eliminate internal fragmentation ($\sim 1/2$ page per process)

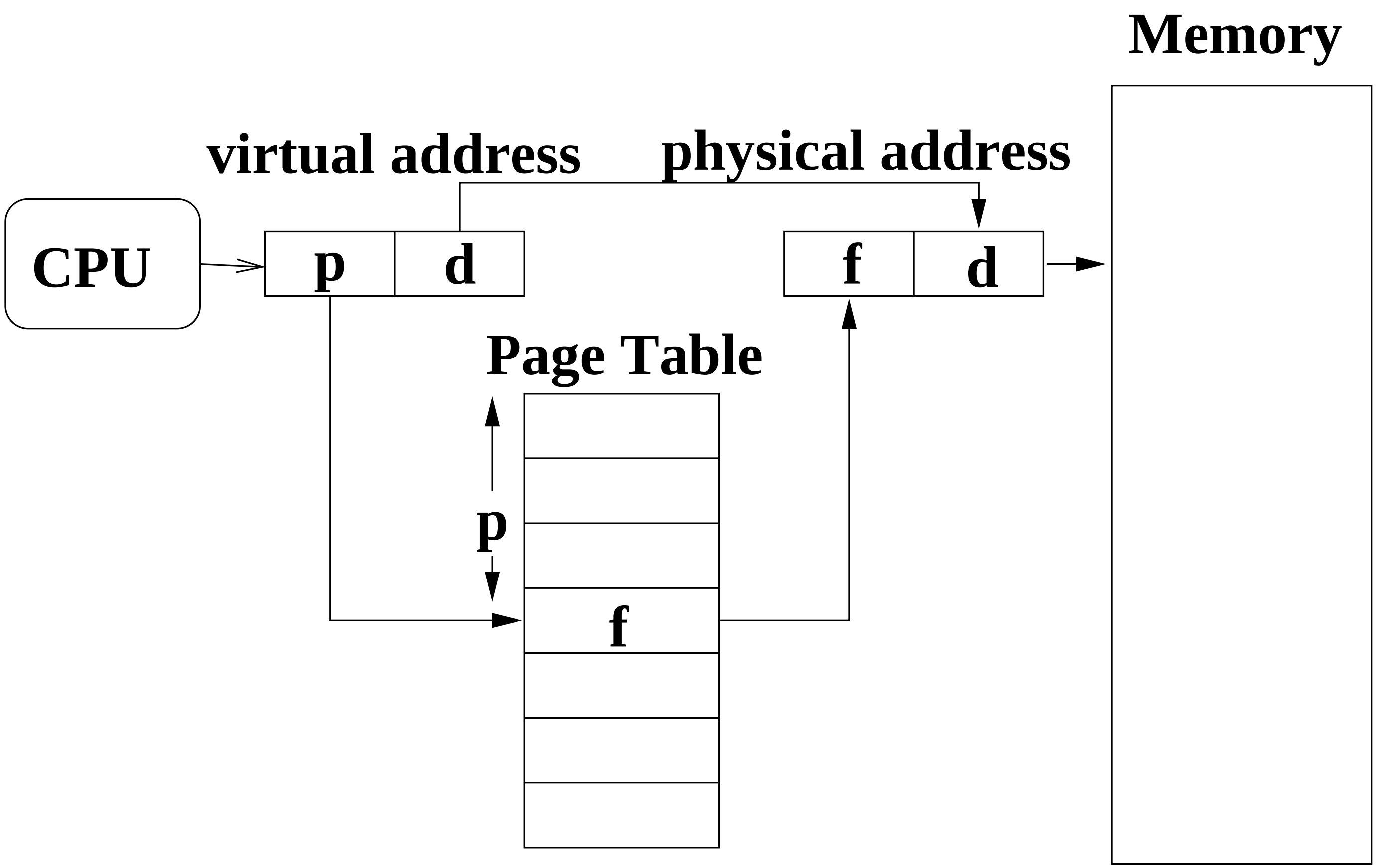
Paging: Example



Paging Hardware

- **Problem:** How do we find addresses when pages are not allocated contiguously in memory?
- **Virtual Address:**
  - Processes use a virtual (logical) address to name memory locations.
  - Process generates contiguous, virtual addresses from 0 to size of the process.
  - The OS lays the process down on pages and the paging hardware translates virtual addresses to actual physical addresses in memory.
  - In paging, the virtual address identifies the page and the page offset.
  - A *page table* keeps track of the frame in memory in which the page is located.

Paging Hardware



Paging Hardware

- Paging is a form of dynamic relocation, where each virtual address is bound by the paging hardware to a physical address.
- Think of the page table as a set of relocation registers, one for each frame.
- Mapping is invisible to the process; the OS maintains the mapping and the hardware does the translation.
- Protection is provided with the same mechanisms as used in dynamic relocation.

Paging Hardware: Practical Details

- Page size (frame sizes) are typically a power of 2 between 512 bytes and 8192 bytes per page.
- Powers of 2 make the translation of virtual addresses into physical addresses easier. Why?

Paging Hardware: Practical Details

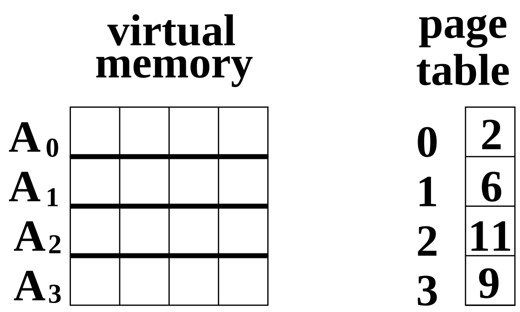
- Powers of 2 make the translation of virtual addresses into physical addresses easier. For example, given:

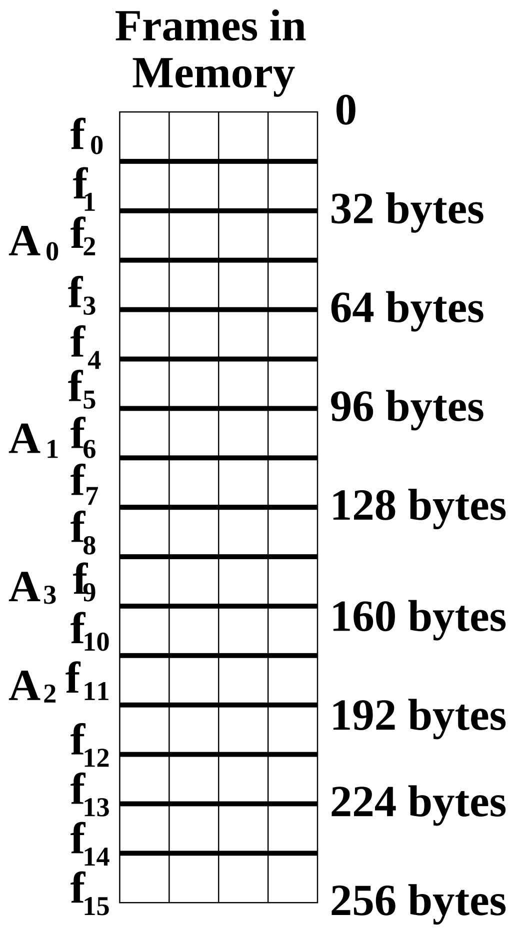
  1. virtual address space of size 2^m bytes and a page of size 2^n, then
  2. the high order $m - n$ bits of a virtual address select the page, and
  3. the low order n bits select the offset in the page

p	d
m-n	n

p: page number
d: page offset

Address Translation Example





memory size = 256 bytes
page size = 16 bytes

Address Translation Example

- How big is the page table?

- How many bits for a physical address? Assume we can address in 1-byte increments.

- What part is p, and d?

- Given virtual address 24 ($0x18$), what is the virtual to physical translation?

Address Translation Example

- How big is the page table?

  4 entries

- How many bits for a physical address? Assume we can address in 1-byte increments.

  8 bits: 4 for page, 4 for offset

- What part is p, and d?

  p: most significant bits; d: least significant

- Given virtual address 24 ($0x18$), what is the virtual to physical translation?

  p = 1, d = 8 (virtual)

  f = 6, d = 8 (physical)

Address Translation Example

- How many bits for an address? Assume we can only address in 1-word (4 byte) increments?
- What part is p, and d?
- Given virtual address 13 ($0xD$), what is the virtual to physical translation?
- What needs to happen on a process context switch?

Address Translation Example

- How many bits for an address? Assume we can only address in 1-word (4 byte) increments?

  6 bits: 4 for page, 2 for offset

- What part is p, and d?

  (again): p is most significant, d is least.

- Given virtual address 13 ($0xD$), what is the virtual to physical translation?

  p $=$ 3, d $=$ 1 (virtual)

  f $=$ 9, offset $=$ 1 (physical)

Address Translation Example (cont)

- What needs to happen on a process context switch?

  Need to save page table in PCB, and then restore page table of the new process.

Making Paging Efficient

How should we store the page table?

- **Registers:**

  Advantages?

  Disadvantages?

- **Memory:**

  Advantages?

  Disadvantages?

Making Paging Efficient

How should we store the page table?

- **Registers:**

  Advantages? Fast.

  Disadvantages? If lots of pages, need many registers. Context switch would require saving/restoring registers which would be slow.

- **Memory:**

  Advantages? Lots of memory. Could just save/restore a pointer to the page table on context switch.

  Disadvantages? Each memory address requires 2 memory accesses: one to translate from virtual to physical memory, one to actually access memory.

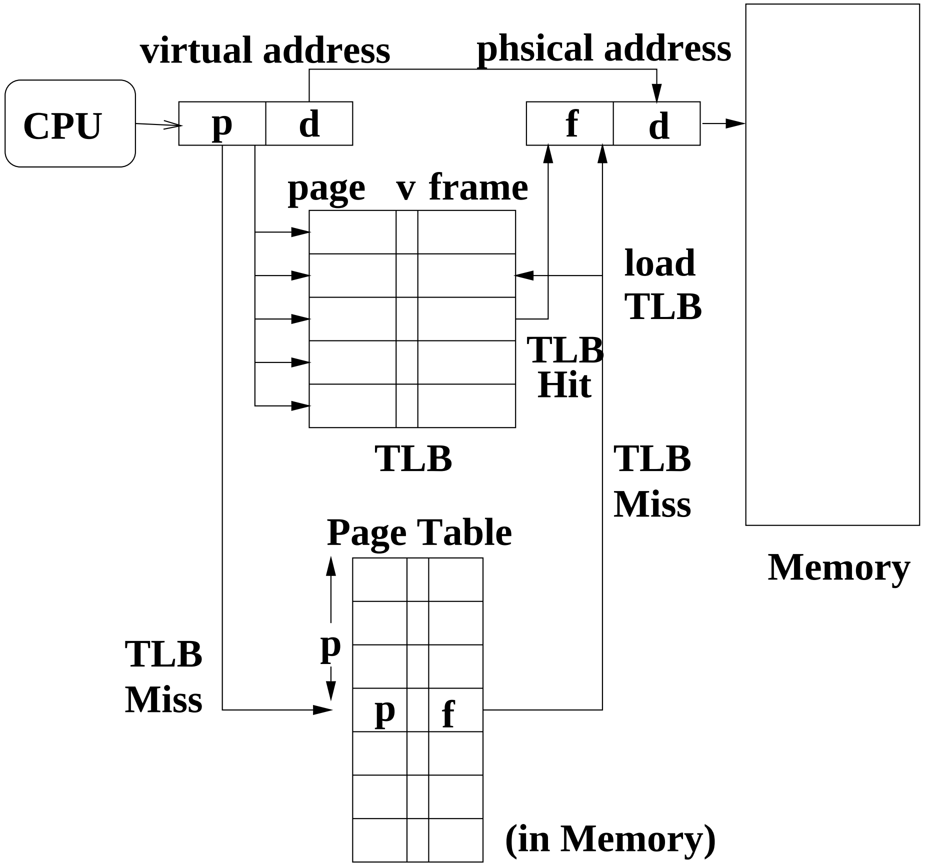
Translation Look-aside Buffers (TLB)

A fast, fully associative memory that stores page numbers (the key) and the frame (the value) in which they are stored.

- If memory accesses have locality, address translation has locality too.
- Typical TLB sizes range from 8 to 2048 entries.

TLB Implementation

v: valid bit that says the entry is up-to-date



Costs of Using The TLB

1. What is the effective memory access cost if the page table is in memory?

2. What is the effective memory access cost with a TLB?

Costs of Using The TLB

1. What is the effective memory access cost if the page table is in memory?

$$ema = 2 * ma$$

2. What is the effective memory access cost with a TLB?

$$ema = (ma + TLB) * p + (1 - p) * (2ma + TLB)$$

A large TLB improves hit ratio, and thus decreases average memory cost.

Initializing Memory when Starting a Process

1. Process needing k pages arrives.
2. If k page frames are free, then allocate these frames to pages. Else free frames that are no longer needed.
3. The OS puts each page in a frame and then puts the frame number in the corresponding entry in the page table.
4. OS marks all TLB entries as invalid (flushes the TLB).
5. OS starts process.
6. As process executes, OS loads TLB entries as each page is accessed, replacing an existing entry if the TLB is full.

Saving/Restoring Memory on a Context Switch

- The Process Control Block (PCB) must be extended to contain:
  - The page table
  - Possibly a copy of the TLB
- On a context switch:
  1. Copy the page table base register value to the PCB.
  2. Copy the TLB to the PCB (optional).
  3. Flush the TLB.
  4. Restore the page table base register.
  5. Restore the TLB if it was saved.
- **Multilevel Paging**: If the virtual address space is huge, page tables get too big. Many systems use a multilevel paging scheme...

Sharing

Paging allows sharing of memory across processes, since memory used by a process no longer needs to be contiguous.

- Shared code must be reentrant, that means the processes that are using it cannot change it (e.g., no data in reentrant code).
- Sharing of pages is similar to the way threads share text and memory with each other.
- 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.
- The user program (e.g., emacs) marks text segment of a program as reentrant with a system call.

Sharing (cont)

- The OS keeps track of available reentrant code in memory and reuses them if a new process requests the same program.
- Can greatly reduce overall memory requirements for commonly used applications.

Summary

- Paging is a big improvement over segmentation:
  - Eliminates the problem of external fragmentation and therefore the need for compaction.
  - Allows sharing of code pages among processes, reducing overall memory requirements.
  - Enables processes to run when they are only partially loaded in main memory.
- However, paging has its costs:
  - Translating from a virtual address to a physical address is more time-consuming.
  - Paging requires hardware support in the form of a TLB to be efficient enough.
  - Paging requires more complex OS to maintain the page table.

Next Time

- Segmented Paging
- Next Week:
  - Monday: Holiday
  - Wednesday: Discussion and exam review
  - Friday: Exam 2 (same place, same time)