Synonyms

- v.p. x, process A
- v.p. y, process B
- index

To avoid synonyms, O.S. or hardware enforces these bits to be the same.

Choosing a Page Size

- Large page size
  - Smaller page table
  - Better coverage in TLB
  - Amortize time to bring in page from disk
  - Allow fast access to larger L1 (virtually addressed, physically tagged)
  - BUT more fragmentation (unused portions of the page)
  - Longer time for start-up

Allowing Multiple Page Sizes

- Recent micros support multiple page sizes
  - Better use of TLB is main reason
- If only two page sizes (one basic, one large)
  - A single bit in the PTE is sufficient (with don't care masks)
- If more than 2 sizes, a field in PTE indicates the size (multiple of basic) of the page
  - Physical frames of various sizes need to be contiguous
  - Need heuristics to coalesce/split pages dynamically

Protection

- Protect user processes from each other
  - Disallow an unauthorized process to access (read, write, execute) part of the addressing space of another process
- Disallow the use of some instructions by user processes
  - E.g., disallow process A to change the access rights to process B
- Leads to
  - Kernel mode (operating system) vs. user mode
  - Only kernel mode can use some privileged instructions (e.g., disabling interrupts, I/O functions)
  - Providing system calls whereby CPU can switch between user and kernel mode

Where to Put Access Rights

- Associate protection bits with PTE
  - E.g., disallow user to read/write/execute in kernel space or read someone else’s program (hence TLB check for v.add. I-caches)
  - E.g., allow one process to share-read some data with another but not modifying it etc.
- User/kernel protection model can be extended with rings of protection
  - E.g., Pentium has 4 levels of protection.
- Further extension leads to the concept of capabilities.
  - Access rights to objects are passed from one process to another

I/O and Caches (primer for cache coherence)

- The “green” page is transferred to/from disk from/to memory. The red “line” is mapped in the cache.
**I/O Passes through the Cache**

- I/O data passes through the cache on its way from/to memory
  - Previous figure does not reflect this possibility.
  - No coherency problem but contention for cache access between the processor and the I/O.
  - Wipes out the entire page area of the cache (i.e. replaces useful lines from other pages that conflict with the page being transferred).
  - Implemented in Amdahl machines (mainframes @ 1980) w/o on-chip caches.

**I/O and Caches – Use of the O.S.**

- I/O interacts directly with main memory.
- Software solution:
  - Output (Memory-disk):
    1. Write-through cache. No problem since correct info is in memory.
    2. Write-back cache: purge the cache of all lines in the page with dirty bits via O.S. interaction (ISA needs a purge instruction).
  - Input (WT and WB):
    - Use a non-cacheable buffer space. Then after input, flush the cache of these addresses and make the buffer cacheable.

**Software solution: Output**

WT case: The contents of the “red” line are already in memory. No problem.

WB case: If the “red” line is dirty, first write it back to memory.

It might also be wise to invalidate all entries belonging to the page in the cache.

**I/O Consistency -- Hardware Approach**

- Subset of the shared-bus multiprocessor cache coherence protocol.
- Cache has duplicate set of tags
  - Allows concurrent checking of the bus and service requests from the processor.
- Cache controller snoops on the bus
  - On input, if there is a match in tags, store in both the cache and main memory.
  - On output, if there is a match in tags: if WT invalidate the entry; if WB take the cache entry instead of the memory contents and invalidate.

**Hardware Solution: Output**

The cache controller snoops on the bus. When it’s time to transfer the red line, its contents come from the cache and the line is invalidated.

**Hardware Solution: Input**

The cache controller snoops on the bus. If there is a match in tags, then the data is stored both in the cache and in main memory.