ECE 2300
Digital Logic & Computer Organization

Fall 2016

Virtual Memory

Cornell University

What's Left

- Virtual Memory
- Exceptions
- Input/Output
- Advanced Topics
- Case Study

Sharing Main Memory

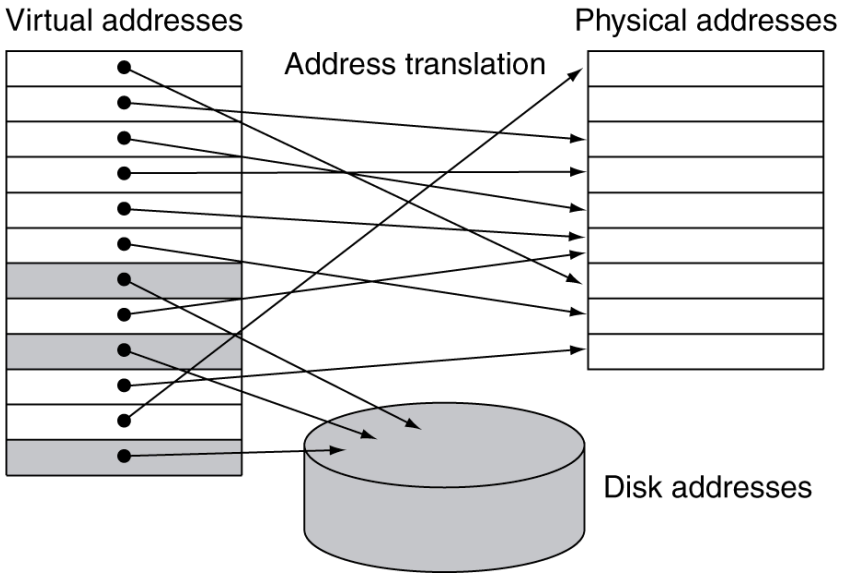
- How do multiple programs share the same main memory (MM) address space?
- What if one program needs more than the amount of installed MM?
- MM is managed similar to a cache
  - Data is brought into MM as it's requested
  - If MM is full, older data gets swapped out
- Each program is dynamically allocated its own set of MM addresses

Virtual Memory

- The hardware and software that dynamically manage the MM
- Moves large blocks (*pages*) between MM and permanent storage as needed
- Provides *protection* among programs that are sharing the MM

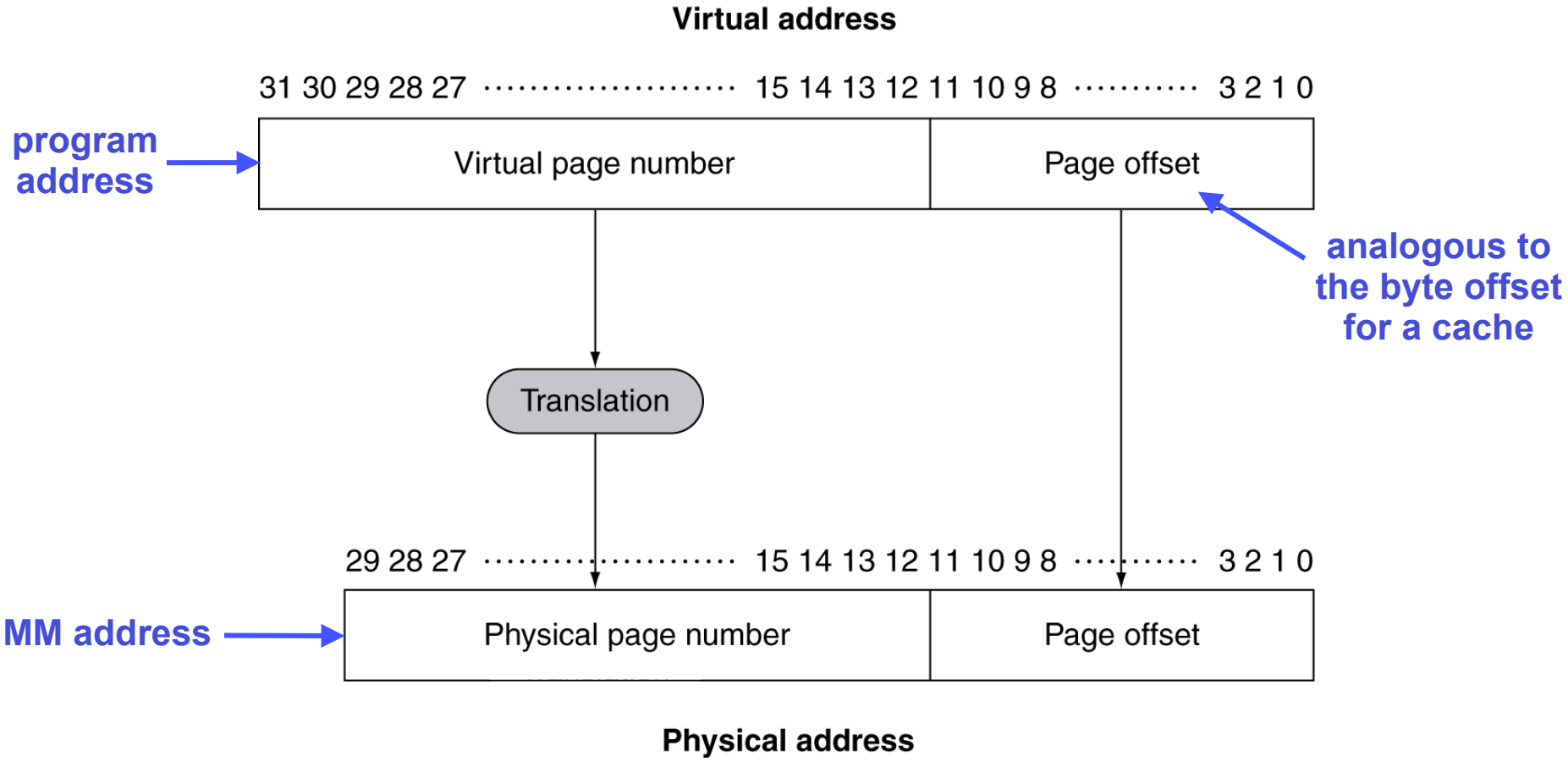
Virtual and Physical Addresses

- When a program is compiled, the instructions and data addresses are *virtual*
  - Do not correspond to where they will be placed in MM

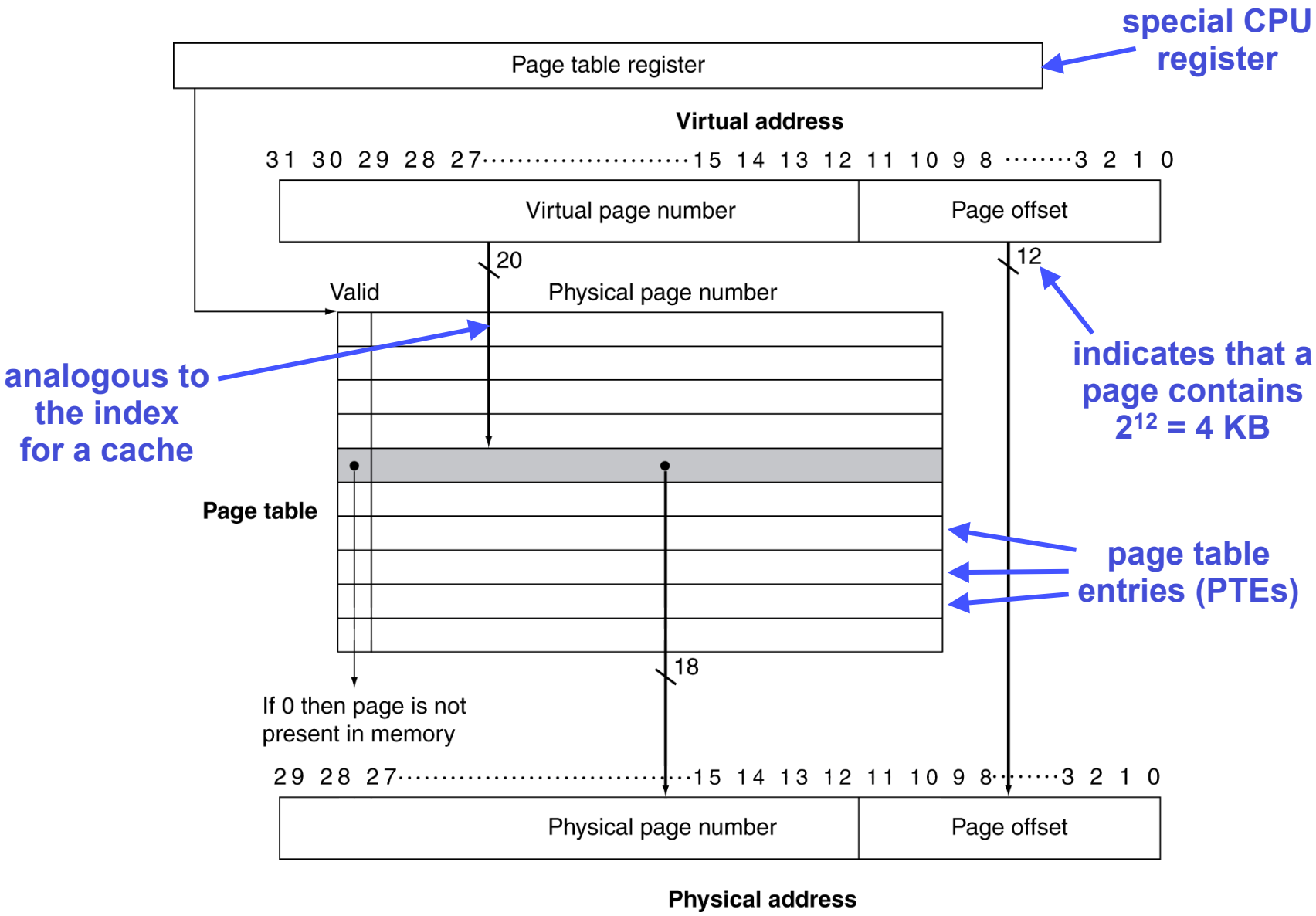


- When requested, a page is brought into a *physical* MM location
- The corresponding *mapping* between virtual to physical addresses is saved
- When the same virtual address is encountered, it is *translated* using this saved information

Virtual and Physical Addresses

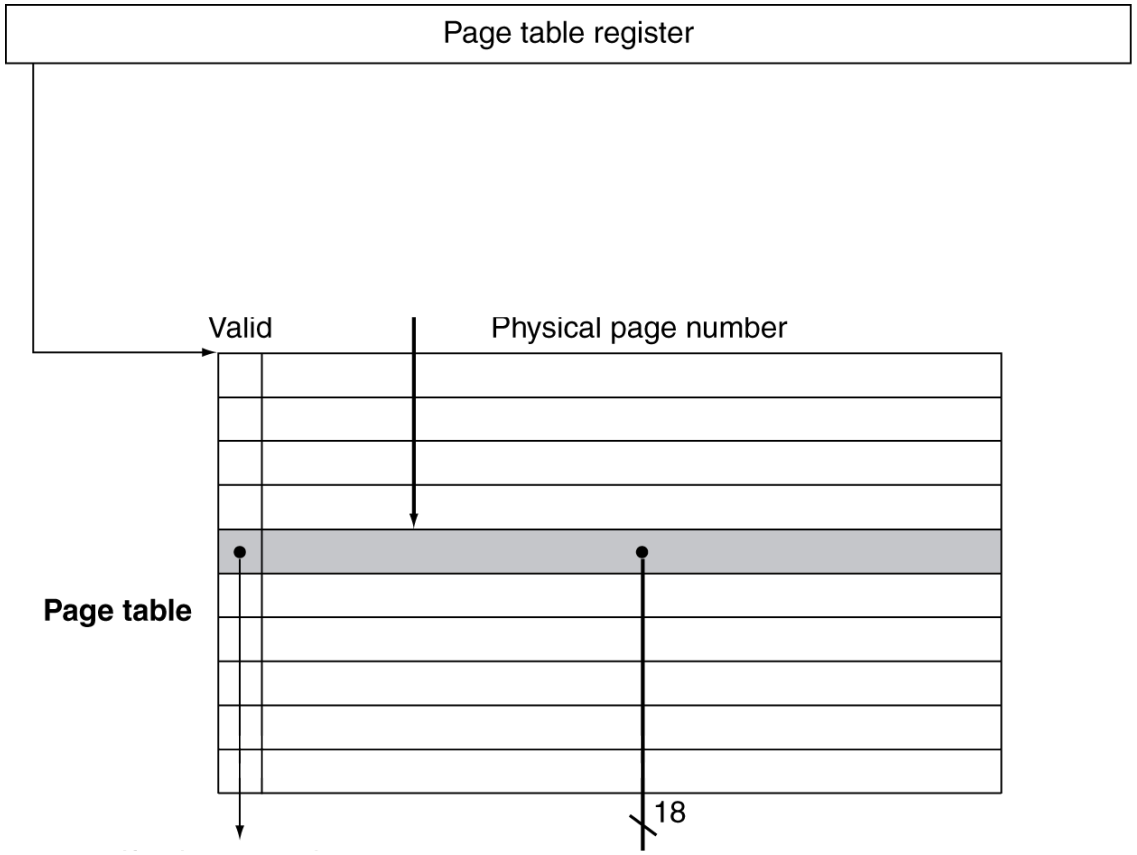


Address Translation Using a Page Table



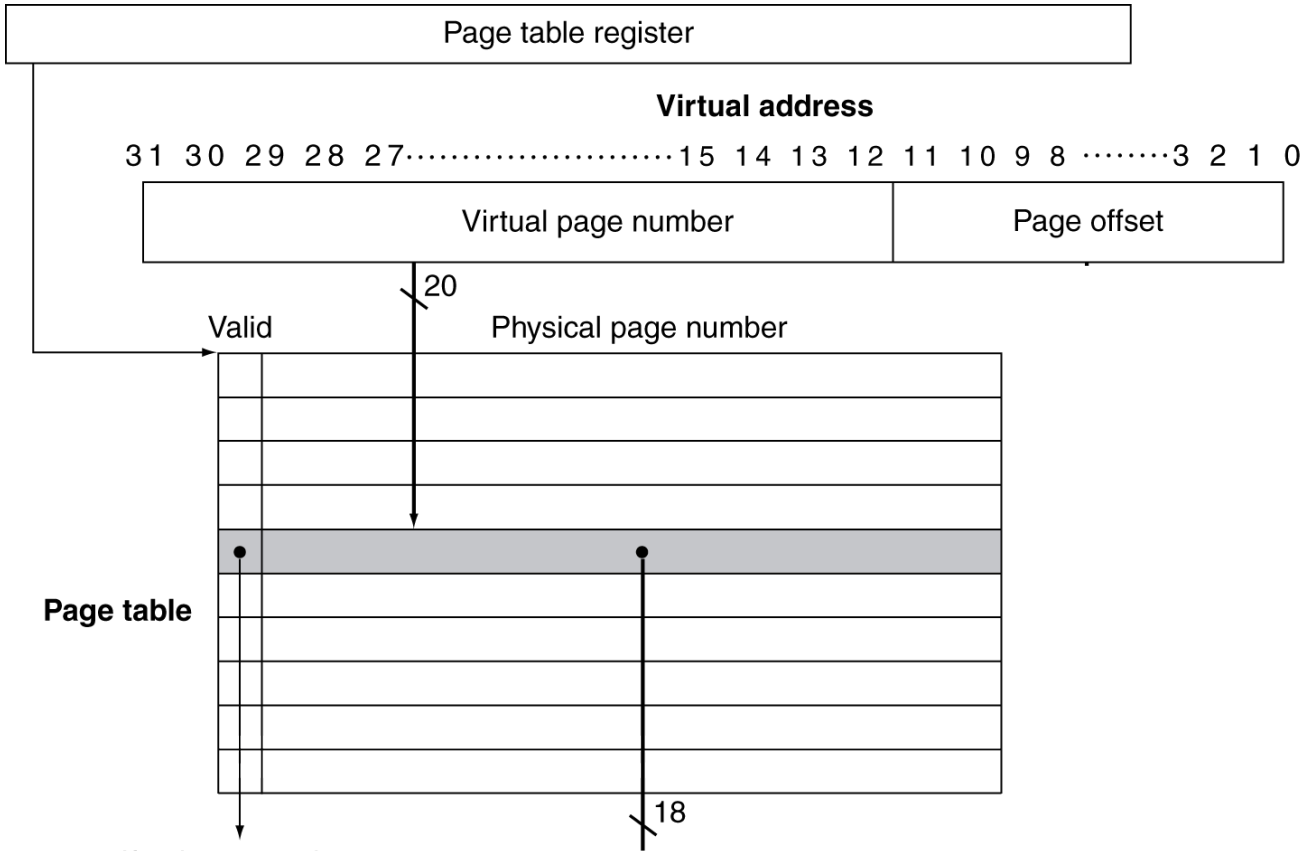
Page Table Operation

The PTR holds the address of the page table in MM for the currently running program



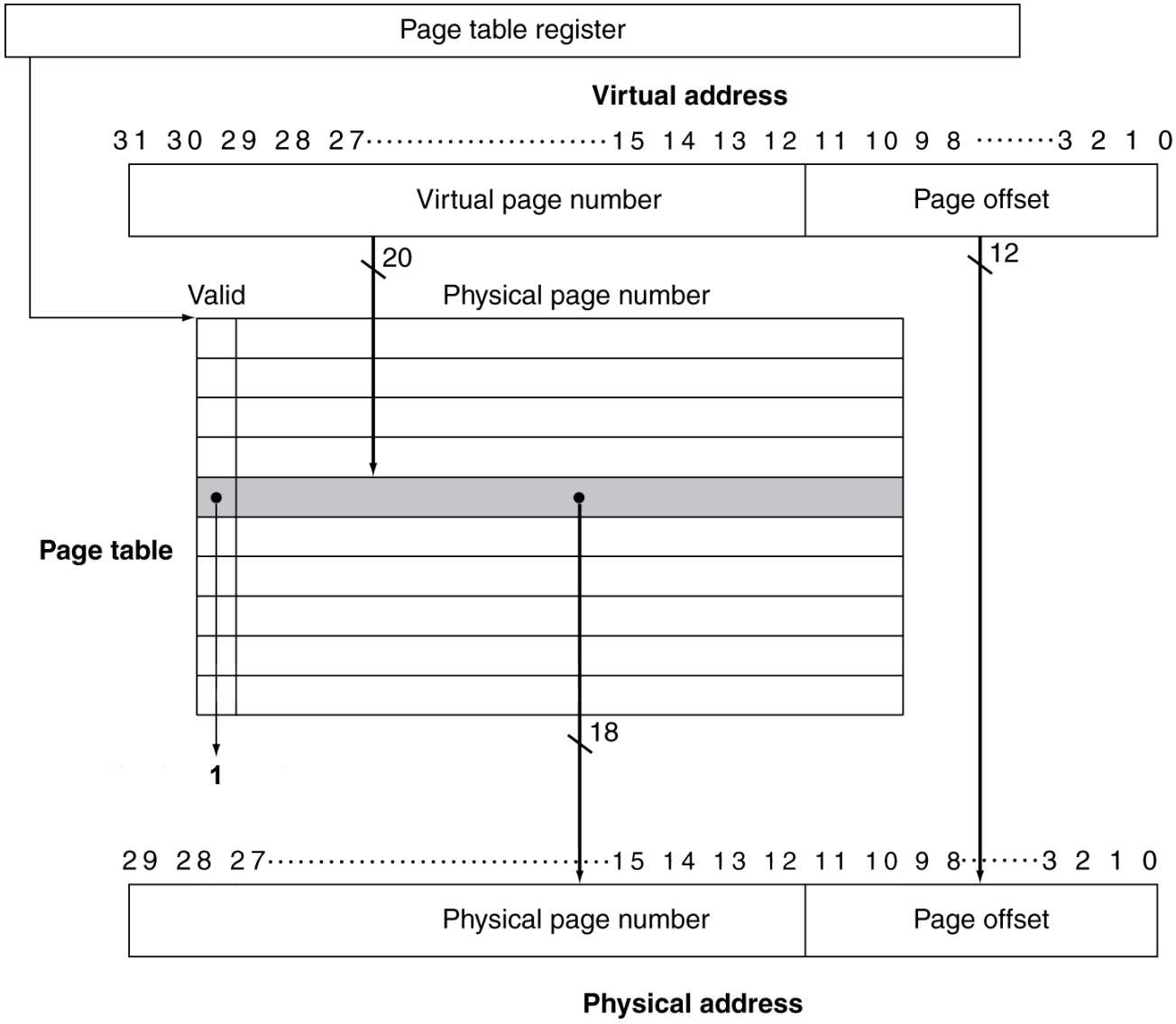
Page Table Operation

PTR+VPN is the address of the PTE in MM



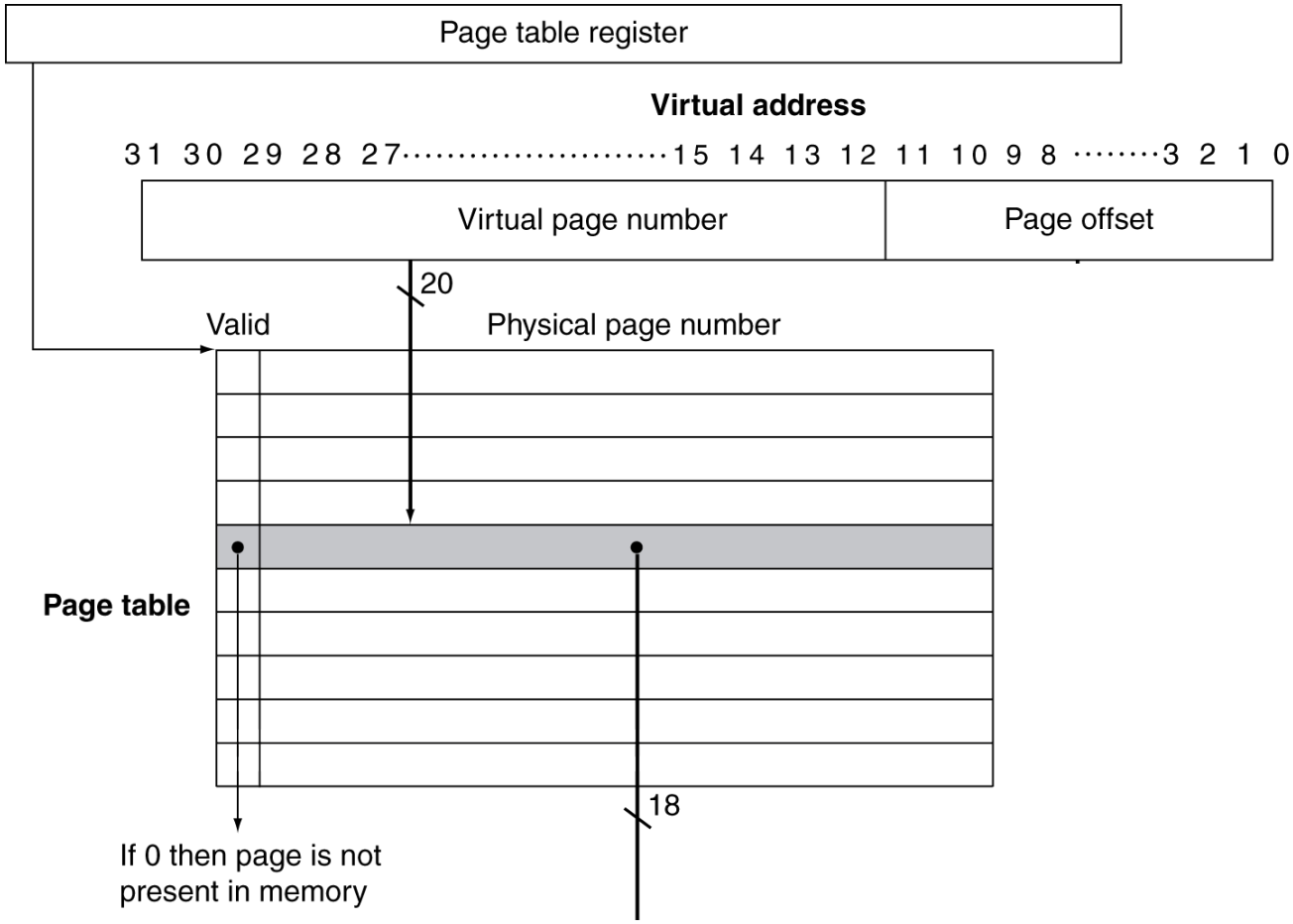
Page Table Operation

If Valid = 1, then the PPN and Page Offset are concatenated to form the physical address



Page Table Operation

If Valid = 0, a miss (*page fault*) has occurred, and the page is read from storage into MM (replacing another page if the MM is full)



Page Faults and Page Replacement

- Miss penalty on a page fault is 10-100M cycles
- Low miss (page fault) rates are essential
  - Fully associative page placement (put anywhere in MM)
  - Least Recently Used (LRU) replacement of a page when MM is full
- The Operating System handles page placement

Page Faults and Page Replacement

- Too expensive to do true LRU (100K-1M pages)

- LRU approximation
  - Each PTE has a *Reference* bit
  - Reference bit is set when a page is accessed
  - OS periodically clears all Reference bits
  - OS chooses a page with a Reference bit of 0

What About Writes to MM?

- **The MM is written**
  - **Write through cache: For every Store instruction**
  - **Write back cache: When a dirty block is replaced**
- **Write through MM policy would significantly degrade performance since writes to storage may take millions of cycles**
- **Write back MM policy is used**
  - ***Dirty* bit in Page Table Entry (PTE) is set on a write to MM**
  - **Page with set Dirty bit is written back to storage when replaced**

Faster Address Translation

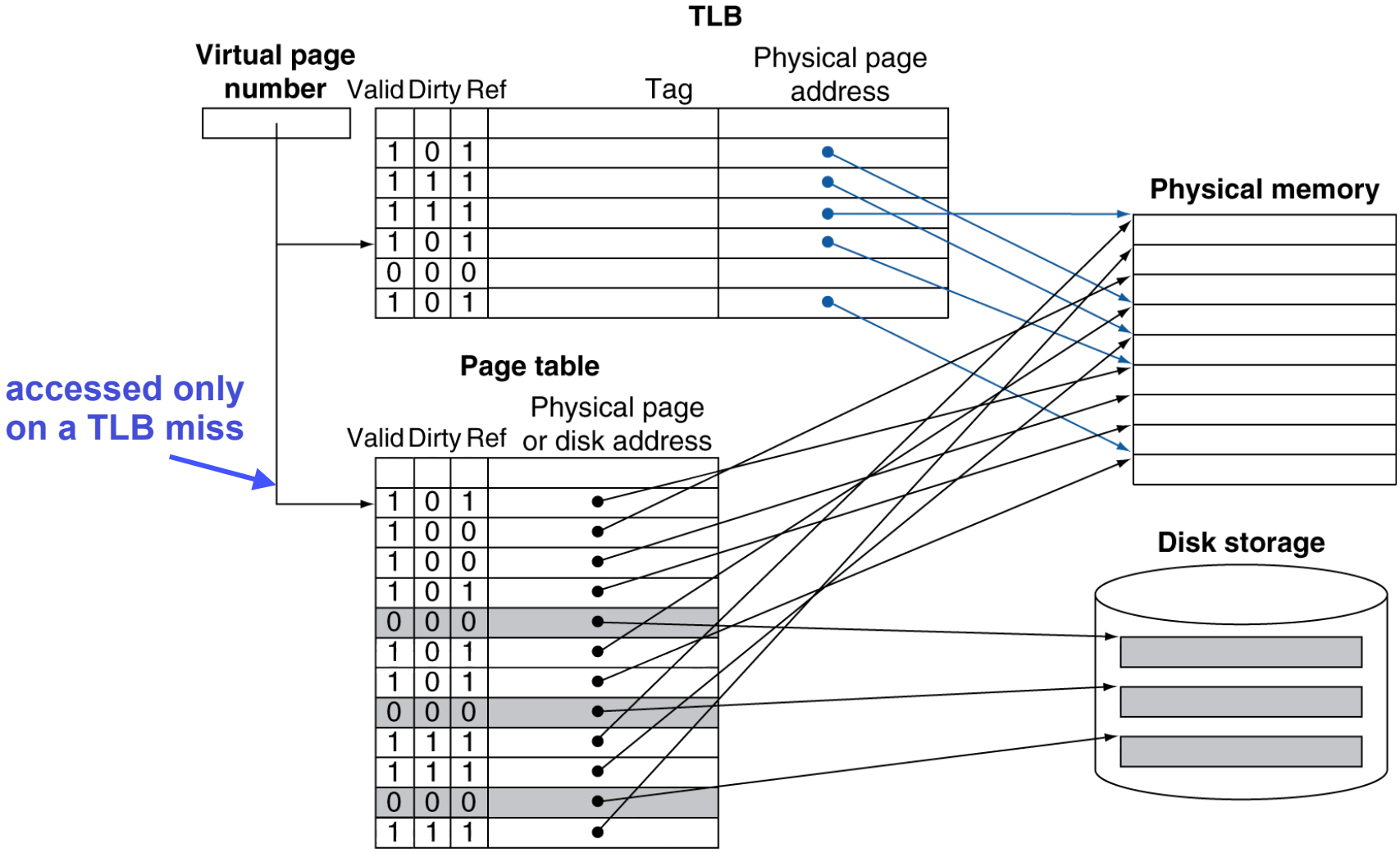
- Have to access the page table in MM before an instruction can be fetched and before data memory can be accessed

- Page table accesses have good locality

$\Rightarrow$ Cache the most recently accessed PTEs within the processor pipeline

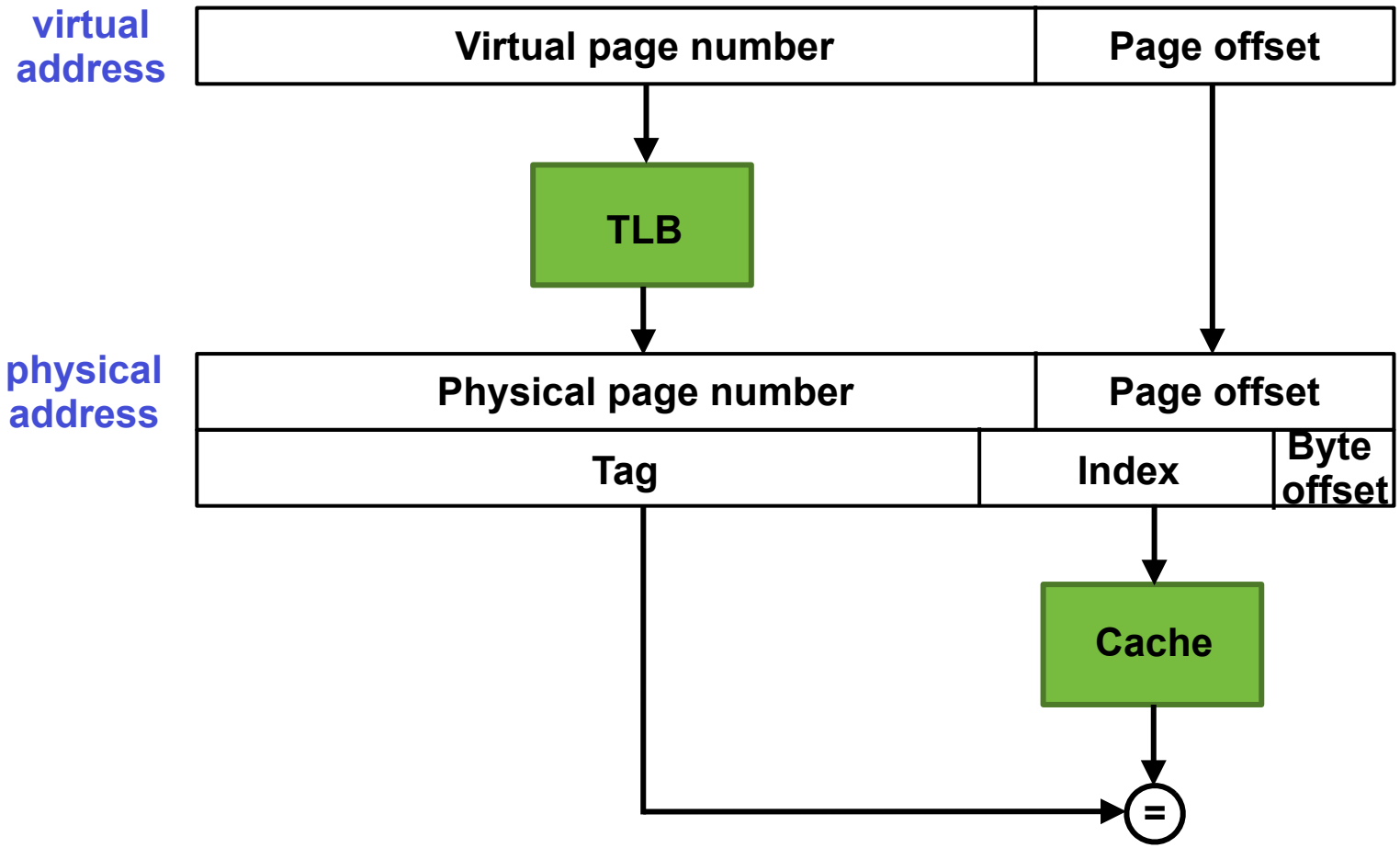
Translation Lookaside Buffer (TLB)

- **Small (typically 16-512 entries) cache of recently accessed PTEs**



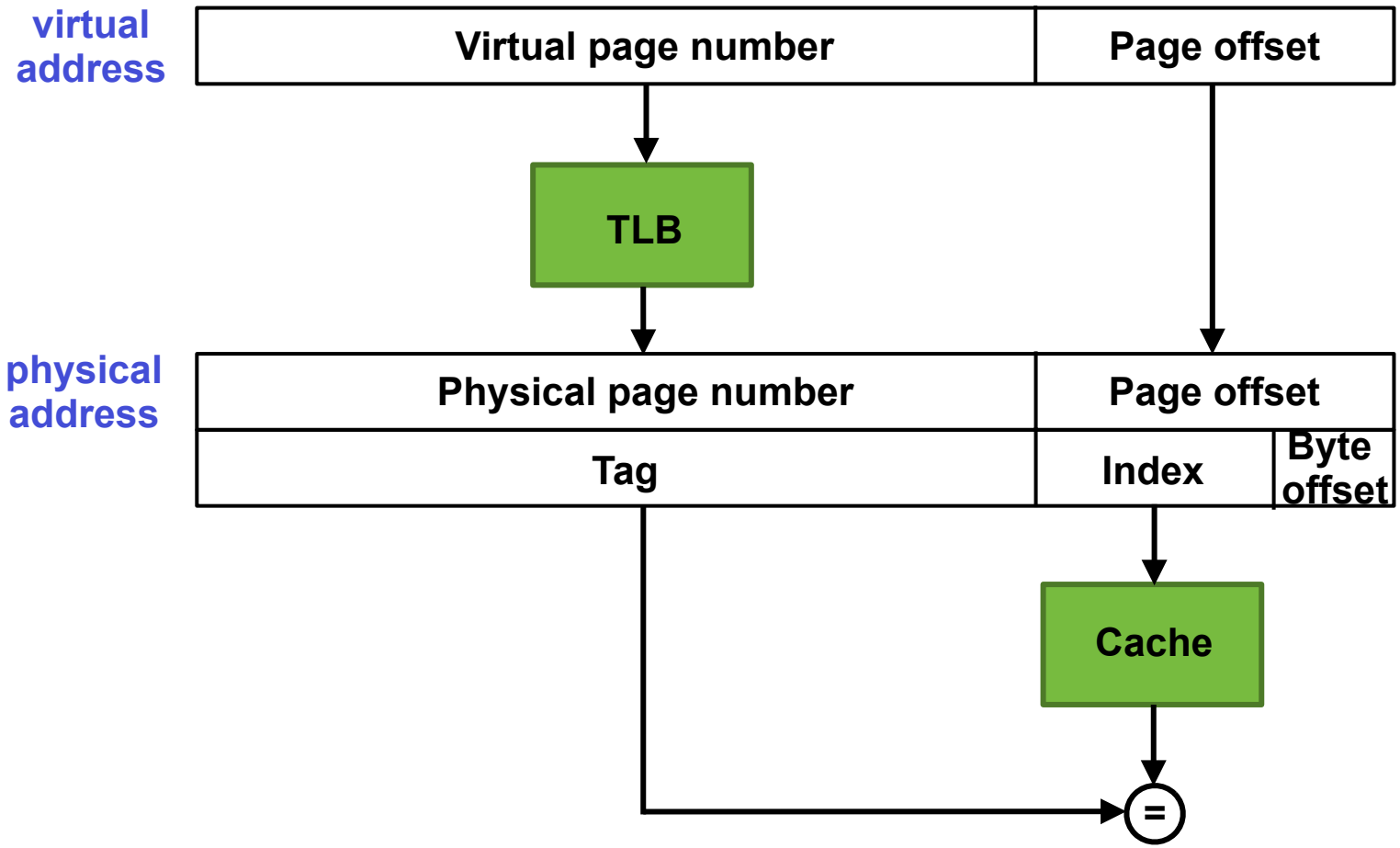
Accessing the TLB and the Cache

- Cache usually uses physical addresses since it holds a subset of what is in MM



Accessing the TLB and the Cache

- What about this situation?



TLB Miss Scenarios

- **TLB miss, page table hit**
  - Bring in the PTE information from page table to TLB
  - Redo the access
  - May be completely performed by hardware
- **TLB miss, page table miss (page fault)**
  - Bring in the page from disk (orchestrated by OS)
  - Load the page table and TLB (orchestrated by OS)
  - Redo the access (cache miss will definitely occur!)

Memory Protection

- MM address space of a running program must be protected from undesired access by other programs
- Special *User/Supervisor* bit is implemented within the processor
- U/S bit must be 1 to unlock special privileges
  - Access to the PTR within the processor
  - Access to the page tables and TLB
  - Access to the U/S bit
  - Special instructions to manipulate this information

Memory Protection

- The U/S bit gets set when a program executes a *system call* (*syscall*) instruction

- A syscall creates an *exception* that kicks out the user program and transfers control to an OS *service routine*

Before Next Class

- H&H 6.7.2, 7.7

Next Time

Exceptions