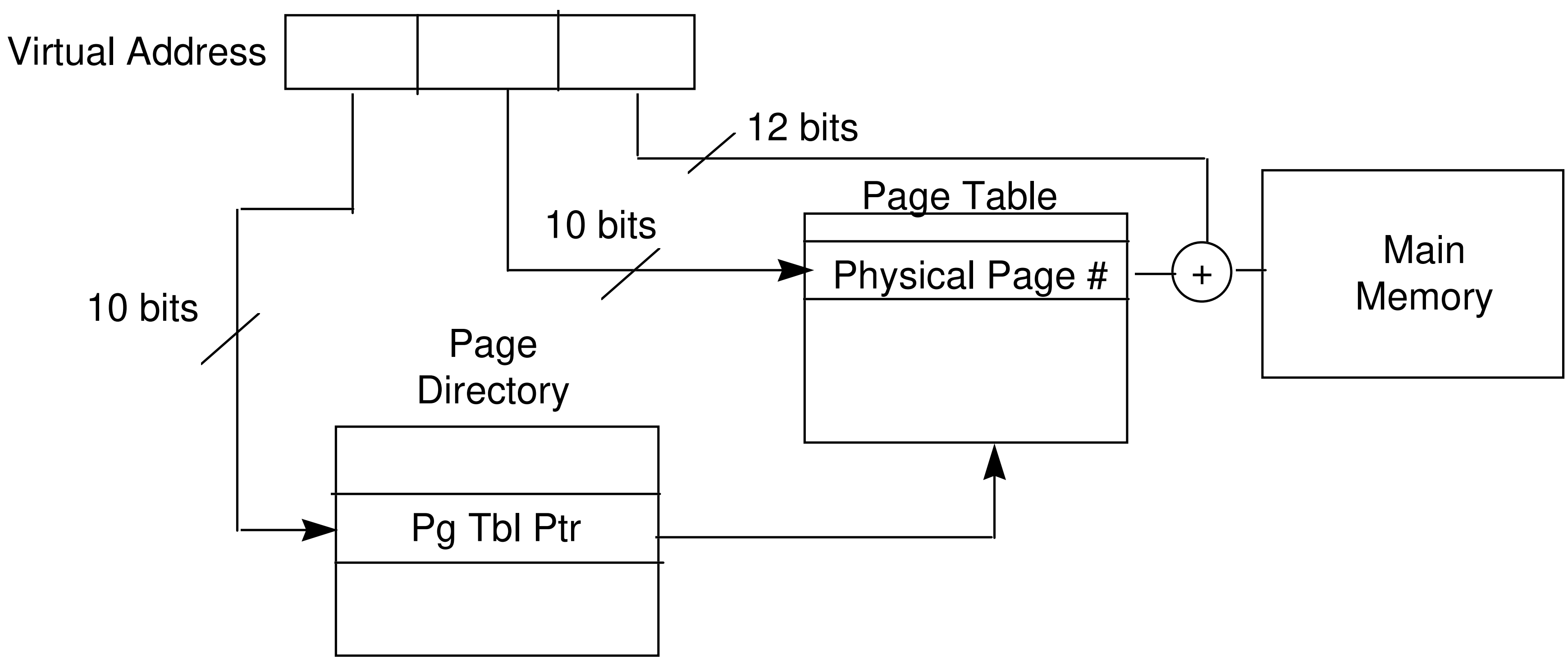
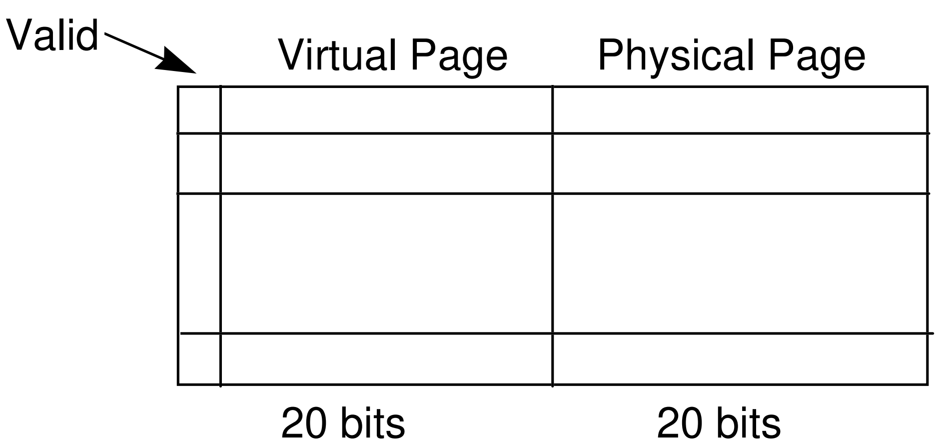
Problems with Page Tables

- One page table can get very big
  - 2^{20} entries (for most programs, most items are empty)
  - solution use a hierarchy of page tables



How to Map from Virtual to Physical Addresses

- need hardware to map between physical and virtual addresses
  - can require multiple memory references
  - this can be slow
- answer: build a cache of these mappings
    - called a translation look-aside buffer (TLB)
    - associative table of virtual to physical mappings
    - typically 16- 64 entries



What Happens when a virtual address has no physical address?

- called a *page fault*
  - a trap into the operating system from the hardware
  - like an interrupt or out of range trap in Mano
- caused by: the first use of a page
  - the operating system allocates a physical page and the process continues
- caused by: a reference to an address that is not valid
  - program is terminated with a “segmentation violation”
- caused by: a page that is currently on disk
  - read page from disk and load it into a physical page, and continue the program

What happens when we fault and there are no more physical pages?

- Need to remove a page from main memory
  - if it is “dirty” we must store it to disk first.
    - dirty pages have been modified since they were last stored on disk.
- How to we pick a page?
  - LIFO (Last In First Out)
    - favors older pages
  - FIFO (First In First Out)
    - favors newer pages
  - LRU (Least Recently Used)
    - favors “popular” pages
    - expensive to implement in hardware

I/O Systems

- Many different types of devices
  - disks
  - networks
  - displays
  - mouse
  - keyboard
  - tapes
- Each have a different expectation for performance
  - bandwidth
    - rate at which data can be moved
  - latency
    - time from request to first data back

I/O Interfaces

- Need to adapt Devices to CPU speeds
- Two major modes
  - Poll
  - Use Interrupts
- Moving the data
  - Programmed I/O
    - Special instructions for I/O
  - Mapped I/O
    - looks like memory only slower
  - DMA (direct memory access)
    - device controller can write to memory
    - processor is not required to be involved
    - can grab bus bandwidth which can slow the processor down

I/O Processors

- Functions required from I/O System can be complex
  - rate adaptation
    - buffering data up until there is enough to use
  - DMA
  - Managing protocols for networks
- Can create a programmable device for I/O
  - really a special purpose processor
  - runs a stored program under the control of the main processor
  - operations limited to one or more I/O devices and moving their data to main memory (or CPU)

Different Requirements lead to Multiple Buses

- Processor Bus (on chip)
  - > 1Gigabyte/sec
- Memory Bus (on processor board)
  - ~500 megabytes per second
- I/O Bus (PCI, MCA)
  - ~100 megabytes per second
  - buses are more complex than we saw in class
    - show PCI spec.
- Device Bus (SCSI)
  - tens of megabytes per second

Issues In Busses

- Performance
  - increase the data bus width
  - have separate address and data busses
  - block transfers
    - move multiple words in a single request
- Who controls the bus?
  - one or more bus masters
    - a bus master is a device that can initiate a bus request
  - need to arbitrate who is the bus master
    - assign priority to different devices
    - use a protocol to select the highest priority item
      - daisy chained
      - central control

Disks

- Several types:
  – Hard Disks - rigid surface with magnetic coating
  – Floppy disks - flexible surface with magnetic coating
  – Optical (read only, write once, multi-write)
- Hard Disk Drives:
  – collect of platters
  – platters contain concentric rings called tracks
  – tracks are divided into fixed sized units called sectors
  – a cylinder is a collection of all tracks equal distant from the center of disk
  – Current Performance:
    - capacity: megabytes to tens of gigabytes
    - throughput: sustained < 10 megabytes/sec
    - latency: mili-seconds