Announcements

- project #3 is due today!
- Project #4 will be on the web today
  - It’s a team project (will email partners)
X86 Segmentation + Paging

Virtual Address

Stored in Segment Register

Selector Offset

Page Directory Page Table Page Frame

Segment Register Virtual Address

Page Directory

Page Table

Page Frame
64 bit processors

- **Problem:** 2 level page tables are too small
- **Solution 1:**
  - Use more levels & larger page size
    - Alpha:
      - 3 level
      - variable size pages
      - w8KB pages
        - 43 bits of virtual address
        - 13 bits page offset
        - 3x10=30 bits in page tables
      - w64KB pages
        - 55 bits of virtual address
        - 16 bits page offset
        - 3x13 = 39 bits in page tables
Sparc & IBM Power 64 bit processors

- **Ultra Sparc 64 bit MMU**
  - 8KB, 16KB, 512KB, 4MB pages supported
  - Software TLB miss handler
  - 44 bit virtual address

- **Power 4**
  - Variable sized pages up to 16MB
  - Inverted page tables
  - TLB
    - 1024 entry 4-way set associate
  - TLB cache
    - Called ERAT
      - 128 entry 2-way set associative
Other 64-bit Designs

- **AMD-64**
  - 54 bit physical memory
  - With 4KB pages
    - 48 bits of virtual address are used
    - 4KB pages
      - 12 bits page
      - $4 \times 9 = 36$ bits via 4-level page tables
    - 2MB pages
      - 21 bits page
      - $3 \times 9 = 27$ bits via 3-level page tables
Inverted Page Tables

- Solution to the page table size problem
- One entry per page frame of physical memory
  
  `<process-id, page-number>`
  
  - each entry lists process associated with the page and the page number
  - when a memory reference:
    
    - `<process-id,page-number,offset>` occurs, the inverted page table is searched (usually with the help of a hashing mechanism)
    
    - if a match is found in entry `i` in the inverted page table, the physical address `<i,offset>` is generated
  
  - The inverted page table does not store information about pages that are not in memory
    
    - page tables are used to maintain this information
    
    - page table need only be consulted when a page is brought in from disk
Inverted Page Table Example (PPC)

Virtual Address

Segment Registers (per process)

Page Table Group
8 page table entries

Page Table (variable size)
one per system

Main Memory

Segment

Page #

Byte

16

4

16

12

24

40

Hash Function

Virtual Segment ID

Status bits

VS ID (40)

Physical page (20)

Page Table Entry (PTE)
Access Large Memory

- **Problem:**
  - Even with Super pages, limited TLB reach

- **Solution:**
  - Add one extra large segment in addition to VM
  - Can be any sized contiguous region of memory
  - Can map into any part of a process's address space
  - Consists of three fields:
    - Virtual base (starting addr in virtual memory, page aligned)
    - Physical base (starting addr in physical memory, page aligned)
    - Length (in multiple of machine's page size)
  - Hardware always consults this mapping regardless of TLB