Announcements

- Midterm #1 was returned
- Project #3 is due Friday
- Project #4 will be on the web later this week
  - It’s a team project (will email partners)
## Midterm #1 Summary

<table>
<thead>
<tr>
<th>Question</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>20</td>
<td>17.8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>20</td>
<td>14.2</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>16</td>
<td>12.5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>12</td>
<td>8.2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>18</td>
<td>12.9</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>14</td>
<td>10.5</td>
</tr>
<tr>
<td>Overall</td>
<td>52</td>
<td>94</td>
<td>76.1</td>
</tr>
</tbody>
</table>

Standard Dev: 11.5
Page State (hardware view)

- **Page frame number** (location in memory or on disk)
- **Valid Bit**
  - indicates if a page is present in memory or stored on disk
- **A modify or dirty bit**
  - set by hardware on write to a page
  - indicates whether the contents of a page have been modified since the page was last loaded into main memory
  - if a page has not been modified, the page does not have to be written to disk before the page frame can be reused
- **Reference bit**
  - set by the hardware on read/write
  - cleared by OS
  - can be used to approximate LRU page replacement
- **Protection attributes**
  - read, write, execute
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
X86 Segmentation + Paging

- Stored in Segment Register
- Virtual Address

Selector + Offset = 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
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 processes 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
Inverted Page Table Example (PPC)

Virtual Address

<table>
<thead>
<tr>
<th>Seg</th>
<th>Page #</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>16</td>
<td>12</td>
</tr>
</tbody>
</table>

16 Segment Registers (per process)

Page Table Group
8 page table entries

Hash Function

Virtual Segment ID

Status bits

Page Table Entry (PTE)

Page Table (variable size)
one per system

Main Memory