Translation, Protection, and Virtual Memory

2/25/16
CS 152 Section 6
Colin Schmidt
Agenda

- Protection
- Translation
- Virtual Memory
- Lab 1 Feedback
- Questions/Open-ended discussion
- Hand back Lab 1
Protection

• Why?

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
<th>Supervisor</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Pointer to next level of page table.</td>
<td>Global</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Pointer to next level of page table—global mapping.</td>
<td>●</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>Supervisor read-only, user read-execute page.</td>
<td>● ● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>3</td>
<td>Supervisor read-write, user read-write-execute page.</td>
<td>● ● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>4</td>
<td>Supervisor and user read-only page.</td>
<td>●</td>
<td>● ●</td>
</tr>
<tr>
<td>5</td>
<td>Supervisor and user read-write page.</td>
<td>● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>6</td>
<td>Supervisor and user read-execute page.</td>
<td>● ● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>7</td>
<td>Supervisor and user read-write-execute page.</td>
<td>● ● ● ●</td>
<td>● ●</td>
</tr>
<tr>
<td>8</td>
<td>Supervisor read-only page.</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Supervisor read-write page.</td>
<td>● ●</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Supervisor read-execute page.</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>11</td>
<td>Supervisor read-write-execute page.</td>
<td>● ●</td>
<td>●</td>
</tr>
<tr>
<td>12</td>
<td>Supervisor read-only page—global mapping.</td>
<td>● ●</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Supervisor read-write page—global mapping.</td>
<td>● ● ●</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Supervisor read-execute page—global mapping.</td>
<td>● ● ●</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Supervisor read-write-execute page—global mapping.</td>
<td>● ● ● ●</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: Encoding of PTE Type field.
Translation

• Why?
  – Multiple users, programs

• Types
  – Base and Bound
  – Paging
  – Segmentation
  – Paged Segements
    • Typical Computer Arch hybrid
Segmentation vs Paging

- words per address
  - 2 vs 1

- programmer visible
  - maybe vs no

- Allocating/replacing block
  - swapping in a new block is hard because size is varied and must be contiguous

- Memory use inefficiency
  - External vs internal fragmentation

- Efficient Disk traffic
  - Not always vs Yes
Page Tables

• Linear vs Hierarchical
  – Linear is simple and inefficient

• How big should a page be?
  – Bigger -> more fragmentation
  – Smaller -> less TLB reach
  – 4KB in RISC-V, why?
    • Legacy
    • Transparent Superpage Support
    • Multi-level TLB hierarchy
What is in a PTE?

<table>
<thead>
<tr>
<th>31</th>
<th>20 19</th>
<th>10 9</th>
<th>7 6 5 4 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPN[1]</td>
<td>PPN[0]</td>
<td>Reserved for software</td>
<td>D</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.14: Sv32 page table entry.

<table>
<thead>
<tr>
<th>63</th>
<th>48 47</th>
<th>28 27</th>
<th>19 18</th>
<th>10 9</th>
<th>7 6 5 4 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>20</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.17: Sv39 page table entry.

<table>
<thead>
<tr>
<th>63</th>
<th>48 47</th>
<th>37 36</th>
<th>28 27</th>
<th>19 18</th>
<th>10 9</th>
<th>7 6 5 4 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>11</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.20: Sv48 page table entry.
Page Table Walk

• Hardware or Software?
  – Hierarchical page table is pretty well agreed upon so makes sense to put in hardware
  – Can always add disable to PTW in hardware that allows fall back to software
TLB

• Why?
• Separate D vs I TLB?
  – Hazards
• How big should your TLB be?
  – Factor TLB miss, and refill into AMAT
  – Think about Iso-Area AMAT curves
  – Increase assoc vs add entry
Address Translation: *putting it all together*

- Virtual Address
  - hardware
  - hardware or software
  - software

  - Page Table Walk
  - TLB Lookup
  - Page Fault (OS loads page)
  - Protection Check
  - Protection Fault
  - Hardware or software
  - SEGFAULT

Where?
Virtual-Address Caches

- one-step process in case of a hit (+)
- cache needs to be flushed on a context switch unless address space identifiers (ASIDs) included in tags (-)
- aliasing problems due to the sharing of pages (-)
- maintaining cache coherence (-) (see later in course)

Alternative: place the cache before the TLB

(StrongARM)
Aliasing

• Physical Cache?
  – How?

• Virtual Cache?
  – How?

• Virtual Tag, Physical Index?
  – How?

• Physical Tag, Virtual Index?
  – How?
Concurrent Access to TLB & Cache (Virtual Index/Physical Tag)

Index L is available without consulting the TLB
=> *cache and TLB accesses can begin simultaneously!*

Tag comparison is made after both accesses are completed

**Cases:** $L + b = k$, $L + b < k$, $L + b > k$
Virtual Index, Physical Tag

• Lets design an 4-way set associative version
Questions

• Any topics want to discuss as a group?
Lab 1 Feedback

• New Requirements
  – NO DUMPS
    • Any copy paste should almost always be in appendix
  – Must be typed
  – Must be digitally submitted
    • Name the file with your first and last name
  – May impose a page limit, but not yet
Lab 1 Feedback

• Make it interesting
• Non-interesting things
  – Big tables of numbers
  – Big list of calculations (1 example is good)
• Interesting Things
  – Comparative graphs
  – Insightful comments
  – Synthesis of ideas
  – Cool open-ended things
Questions and Hand Back