ADDRESS TRANSLATION AND TLB

Mahdi Nazm Bojnordi

Assistant Professor
School of Computing
University of Utah
Virtual Address

- Every virtual address is translated to a physical address with the help of hardware
- Data granularity

What is the table size?
Address Translation Issues

- Where to store the table?
  - Too big for on-chip cache
  - Should be maintained in the main memory

- What to do on a page table miss (page fault)?
  - No valid frame assigned to the virtual page
  - OS copies the page from disk to page frame

- What is the cost of address translation?
  - Additional accesses to main memory per every access
  - Optimizations?
Address Translation Cost

- Page walk: look up the physical address in the page table

- How many pages to store the page table?

![Diagram showing address translation]

<table>
<thead>
<tr>
<th>Virtual Address</th>
<th>Page Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Address</th>
<th>Page frame No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>
Multi-Level Page Table

- The virtual (logical) address space is broken down into multiple pages
  - Example: 4KB pages

![Diagram of virtual and physical address spaces and page table]

- Virtual Address
  - 10 10 12

- Physical Address
  - Page frame No: 12

- base →

- Physical Address
  - Page frame No: 12
Translation Lookaside Buffer

- Exploit locality to reduce address translation time
  - Keep the translation in a buffer for future references
Translation Lookaside Buffer

- Just like any other cache, the TLB can be organized as fully associative, set associative, or direct.
- TLB access is typically faster than cache access.
  - Because TLBs are much smaller than caches.
  - TLBs are typically not more than 128 to 256 entries even on high-end machines.

<table>
<thead>
<tr>
<th>V</th>
<th>Virtual Page #</th>
<th>Physical Page #</th>
<th>Dirty</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CAM Based TLB

- Content addressable memory (CAM)
  - Unlike RAM, data in address out

RAM: Read Operation

CAM: Search Operation
**CAM Based TLB**

- **Content addressable memory (CAM)**
  - Unlike RAM, data in address out

- **CAM based TLB**
  - Both CAM and RAM are used

What if multiple rows match?
On a TLB miss, is the page loaded in memory?

- Yes: takes 10’s cycles to update the TLB
- No: page fault
  - Takes 1,000,000’s cycles to load the page and update TLB

Physically indexed, physically tagged: TLB on critical path!
Problem: increased critical path due to sequential access to TLB and cache

Virtual Page No  Page Offset

TLB

Physical Frame  Page Offset

Tag  Index  Byte

Tag Array

Data Array

Data Block

hit/miss*

Observation: lower address bits (page offset) are not translated
Virtually Indexed Caches

- **Idea:** Index into cache in parallel with page number translation in TLB

![Diagram of Virtually Indexed Caches]

*Question for the final exam:* what if the page offset is not equal to index + byte?