CS 3305A

Memory Management

Lecture 18
Page Table Implementation
Implementation of Page Table

- The simplest approach is to have the page table implemented as a set of dedicated registers.

- Note: Not feasible to keep page table in registers.
  - Why? Page tables can be very large.
  - Would be very expensive.
Implementation of Page Table

- Each process has a page table
- Page table is kept in main memory
- Page-table base register (PTBR) points to the page table
- During a context switch, changing page tables requires changing PTBR
Implementation of Page Table

- Solution: Use a special fast-lookup hardware cache called associative memory or translation look-aside buffers (TLBs)
TLB
TLB

- Associative memory
- Address translation 
  \((p, d)\)
  - If \(p\) is in associative memory, get frame number out
  - Otherwise get frame number from page table in memory

<table>
<thead>
<tr>
<th>Page #</th>
<th>Frame #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Paging Hardware With TLB
TLB

- The TLB contains only a few of the page-table entries.
- When a logical to physical address is requested by the CPU its page number is presented to the TLB.
- If found the frame number is immediately available (TLB Hit)
TLB

- If page number is not in TLB then a **TLB miss** occurs
  - The page table is consulted
  - The page number and frame number is added to the TLB
  - If the TLB is full then one of the entries is replaced
  - Example replacement policy:
    - Least Recently Used (LRU)

- A high hit rate has a high impact can dramatically reduce lookup time
Effective Access Time

- Hit ratio (percentage of times that a particular page is found in the TLB) = 80%
- TLB hit:
  - Time to get data: 120
- TLB miss:
  - Time to get data: 220
- Effective access time:
  - $0.80 \times 120 + 0.2 \times 220 = 140$
Effective Access Time

- Assume hit ratio is 98% (typical)
- Effective access time:
  - 0.98 * 120 + 0.02 * 220 = 122
Protection and Shared Pages
Protection

- Memory protection implemented by associating protection bit with each frame
- One protection bit can define a page to be read-write or read-only
Shared Pages Example

process $P_1$

process $P_2$

process $P_3$