Lost Class

1. Pages
2. Design a VM
3. Page Table
4. Locality

I. VM as a caching tool

II. VM for Memory Protection

A process should not modify any OS pages. Don't access or modify its read-only content (.text)
Page table helps us with this

3 additional bits.

Sup bit / Read bit / Write bit

0 → cannot write
1 → can write

Should the process be running in supervisor mode to access this

PM

<table>
<thead>
<tr>
<th>Page 7</th>
<th>Sup</th>
<th>Read</th>
<th>W</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>VPO</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>VP1</td>
</tr>
</tbody>
</table>

Shared Physical Page!

Eg: Shared Libraries (Print)
III VM for Memory Management

Simplifies → Linking

Sharing

Allocation

Address Translation
Speeding up Address Translation

MMU → PTE from the Physical Memory.

Worst Case

Worst Case → PT not in the cache. Read PM

Best Case → PT is in the cache. Read from cache

Have a small cache → for small number of PTEs

Translation Lookaside Buffer (TLB)
Address Translation

Ganesh Kumar  .  April 20, 2016
Page Hit (Operational View)

- CPU chip
  - Processor
  - MMU
    - PTEA
    - PTE
    - PA
  - Cache/mem
- VA
- Data
- 1
- 2
- 3
- 4
- 5
Page Fault (Operational View)
Cache and Main Memory

- They are distinct entities (duh)
- Caches also use only physical addresses (in most implementations)
- Cache Hits can still happen
- Cache Misses can also happen

**Example:** Requesting a page that is in the Main Memory but not in the Cache?
- Cache miss!
- Load the block(s) onto the cache.
- Now pass along the value requested by the MMU or the CPU.
VM with Cache
TLB Hit
CPU chip

1. Processor
2. VPN
3. PTE
4. PTEA
5. PA
6. Data

TLB Miss

Translation

Cache/memory

VA
Worksheet!
In-Class Problem Walkthrough
Will be solved in class on April 20, 2016

Consider the following memory system snapshot,

Figure A. TLB: Four sets, 16 entries, four-way set associative

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>02</td>
<td>00D</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>07</td>
<td>03</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>02</td>
<td>00D</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN - Physical Page Number

Figure B. Page table: Only the first 16 PTEs are shown

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN - Page Number

TLB entry will be updated
Given Virtual Address: **0x027c**

### A. Virtual Address Format

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPN</td>
<td>0x09</td>
</tr>
<tr>
<td>TLB Index</td>
<td>0x1</td>
</tr>
<tr>
<td>TLB tag</td>
<td>0x2</td>
</tr>
<tr>
<td>TLB Hit? (Y/N)</td>
<td>No.</td>
</tr>
<tr>
<td>Page Fault? (Y/N)</td>
<td>No.</td>
</tr>
<tr>
<td>PPN</td>
<td>0x17</td>
</tr>
</tbody>
</table>

### B. Address translation

![Address translation diagram]

### C. Physical Address Format

![Physical Address Format diagram]
## D. Physical Memory Reference

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte Offset</td>
<td>0x0</td>
</tr>
<tr>
<td>Cache Index</td>
<td>0xF</td>
</tr>
<tr>
<td>Cache tag</td>
<td>0x17</td>
</tr>
<tr>
<td>Cache Hit? (Y/N)</td>
<td>No</td>
</tr>
<tr>
<td>Cache byte returned</td>
<td></td>
</tr>
</tbody>
</table>

Bye Addressable
TLB → tag, set and block offset.

Each block → PTE

VPN → Virtual Page Number

(tag + set).

So far → PPN = 0x17

How do we get the Physical Address?

PA → PPN + PPO

PA → PPN + VPO