Board Notes on Virtual Memory

Part A: Why Virtual Memory?
- Let’s user program size exceed the size of the physical address space
- Supports protection
  - Don’t know which program might share memory at compile time.

Consider the following:

<table>
<thead>
<tr>
<th>Code from P1 (VAs):</th>
<th>16384 = 2^{14}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 add $4, $5, $6</td>
<td>…to…</td>
</tr>
<tr>
<td>1 sub $7, $8, $9</td>
<td>2^{14} + 2^{12} = 20479</td>
</tr>
<tr>
<td>4095 xor $7, $8, $9</td>
<td>20480</td>
</tr>
<tr>
<td>4096 lw $7, 0($10)</td>
<td>20477 + 2^{12} = 24575</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>8191 sub $7, $10, $5</td>
<td></td>
</tr>
</tbody>
</table>

- Above:
  o Assume 4KB pages – therefore, think about “groups of 2^{12} pieces of data”
- Usually, virtual address space is much greater than physical address space
  - (Mapping allows code with virtual address to run on any machine.)

Part B: How do we translate a Virtual Address to a Physical Address
(or alternatively, “How do we know where to start looking in memory?”)
- Good analogy: It’s like finding what cache block a physical address maps to.

Example:
- What if 32-bit virtual address (2^{32} virtual addresses), 4KB pages (like above), 64 MB of main memory (2^{26} physical addresses)

How is this mapping done?

<table>
<thead>
<tr>
<th>VPN (Virtual Page Number)</th>
<th>OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFN (Physical Frame Number)</td>
<td>OFFSET</td>
</tr>
</tbody>
</table>

How do we do VPN → PFN mapping?
- Leverage structure called page table
- To make analogy to cache, “data” = PFN
- To make analogy to cache, also have valid, dirty bits
If no valid mapping, get page fault:
  o Try to avoid
  o Involves lots of disk traffic
  o Placement in memory done fully associative, LRU to minimize
  o Placement = some extra overhead, but small percent – and worth it to avoid M CC penalty

Offset still the same because we go down the same distance

**More specifically:**
The process works like this...

Even more specifically...
  - The page table is stored in memory
  - The beginning of the page table is stored in the page table register
  - OS knows where PT for each program begins; interfaces with architecture to find