What if Virtual Address Space is Big?

• 32 bit addresses can address $2^{32}$ bytes of memory
• Page Size of 4KB
  – low order 12 bits of VA for offset within a page
  – leaves 20 bits for page number $\Rightarrow 2^{20}$ pages VA space
  – $2^{20}$ page table entries for full VA space
    • If each PT entry is 4bytes need:
      $2^{22}$ bytes of contiguous memory space for Page Table
      ($2^{10}$ contiguous pages of memory for Page Table)
      $\Rightarrow$ have the contiguous memory allocation problem again

• Solution: Page the Page Table
  • Allow it to live in non-contiguous memory
  • Don’t need to have whole PT resident
    $\Rightarrow$ now we need a page table for the page table
Two Level Paging

- CPU
- VA
- PA
- TLB
- PTBR
- Outer PT
- Inner PT

Diagram shows the process of two-level paging with TLB miss handling.
How Many Levels of PT?

• Depends on:
  – Virtual address space size
  – Page size
  – PT entry size
  – Number of contiguous pages in the outer most PT
    (assume only 1 unless otherwise specified)

• Example:
  – 4k \((2^{12}\) bytes) page size, 32 bit VA, 8 byte PT entries
  – VA:
    \[
    \begin{array}{ll}
    p & d \\
    20 \text{ bits} & 12 \text{ bits}
    \end{array}
    \]
  – Need \(2^{20}\) PT entries each 8 bytes long \((2^{23}\) bytes of PT space)
  – Can fit \(2^{12}/2^3 \, (2^9)\) PT entries per PT page
    \Rightarrow need \(2^{20}/2^9 \, (2^{11})\) pages of inner-most page tables
Example Continued

Can fit $2^{12}/2^3$ ($2^9$) entries per page (need 9 bits for offset into inner PT page)
$\Rightarrow$ need $2^{20}/2^9$ ($2^{11}$) pages of inner-most page table pages

<table>
<thead>
<tr>
<th>11 bits</th>
<th>9 bits</th>
<th>12 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>InnerPT#</td>
<td>InnerPTOffset</td>
<td>page offset</td>
</tr>
</tbody>
</table>

Outer PT entries (entries point to inner PT pages) don’t fit on single page
$\Rightarrow$ we need another level of page tables (page the outer PT)

Number of outer level PT pages to refer to $2^{11}$ pages of Inner PT:
can fit $2^9$ PT entries per page, so need $2^2$ pages of 2$^{nd}$ outer-level PT

The 2$^{nd}$ outer-level PT entries fit on a single second-level outer PT page

<table>
<thead>
<tr>
<th>2 bits</th>
<th>9 bits</th>
<th>9 bits</th>
<th>12 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>p2</td>
<td>p3</td>
<td>Page Offset</td>
</tr>
</tbody>
</table>

Middle PT Page Num

Inner PT Page Num

Virtual Page Num
Effective Memory Access Time

Depends on Number of Levels of Page Tables:

Effective Access Time:

\[
\text{Effective Access Time} = (H)(\text{TLB access time} + \text{mem access time}) + (1-H)(\text{TLB access} + \text{PTLevels(PT access)} + \text{mem access})
\]

Example: memory access time 100ns

- TLB access time 20 ns
- TLB hit rate = 80%
- Number PT levels = 3

Effective Access Time =

\[
(0.8)(120) + (0.2)(20 + 3(100) + 100) = 204 \text{ ns}
\]

⇒ more than twice as slow due to paging
What if VA space is really big?

• 64 bit architectures: $2^{64}$ bytes of VA space
  ~ 7 levels of Page Tables
  Effective Access Time is way too slow
  Multiple Pi’s PTs take up way too much of mem

• Solution: Inverted Page Tables
  – Idea: instead of mapping from VA to frame #
    keep mappings of frame# to VA
  – PA space much smaller than VA space
    inverted PT small (one entry per page FRAME)
  – Single, global PT for all processes in the system