Paging

Physical Memory Allocation

→ multiple processes need to share the physical memory
→ last class: base & bound

→ virtual memory for each process (own view, independent from where it is in physical memory)
→ address translation is simple: if (vaddr < bound) ? vaddr + bound;

Limitations

→ memory fragmentation (variably sized processes)
→ inefficient use of physical memory (processes don't need all memory at once)
→ hard to implement memory sharing

* paging: divide virtual memory into pages, physical memory as frames, and map a page to a frame
Address Translation For Paging

Virtual address \( (64\text{bit}) \)

\[
\begin{array}{c|c}
\text{page \#} & \text{offset} \\
\hline
\end{array}
\]

\[12\text{ bits} \quad (2^{12} = 4096 = 4\text{KB page})\]

Physical address

\[
\begin{array}{c|c}
\text{frame \#} & \text{offset} \\
\hline
\end{array}
\]

\[12\text{ bits}\]

Need a way to translate page \# to frame \#.

Offset (location) within a page is the same within a frame, no need to translate.

What data structure can we use as the translation (look up) table?

\(\rightarrow\) Simple approach: a single array with an entry for each page.

How many entries do we need?

\[2^{26} \left(2^{64}/2^{32}\right)\text{ entries!}\]
Translation needed - one page table per process!
A single page table is too large in size, especially since most programs don't use that much memory! To save space, we can introduce layer

<table>
<thead>
<tr>
<th>page #</th>
<th>offset</th>
<th>vaddr</th>
</tr>
</thead>
</table>

Multilevel page table

Same vaddr, we just group bits differently

Allocate the page tables as we go
- If we only access 1 page, we only need 1 top level, 1 2nd level and 1 last level page table (each is much smaller than a PT at 2^52 entries)
Is multilevel page table always smaller than single level?

→ What if we use all of the virtual memory?

→ Need to allocate the first level page table, all of 2nd level page table, and all of 3rd level page table. (Last level page tables store the actual page → frame translation, a total of $2^n$ entries)
Now we mostly solved the space problem, next is the cost of page table lookup.

① Cache: Translation Lookaside Buffer (TLB)

→ Caches the result of translation
3. Make hardware walk the page table for you

→ hw needs to understand the page table format.

<table>
<thead>
<tr>
<th>Sign Extend</th>
<th>Page-Map Level-4 Table Offset (PML4)</th>
<th>Page-Directory Pointer Offset</th>
<th>Page-Directory Offset</th>
<th>Page Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>48 47 39 38 30 29 21 20 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- PML4: Page-Map Level-4 Table
- PDPE: Page Directory Pointer Table
- PDE: Page Directory Table
- 2 Mbyte Physical Page

- CR3: Page-Map Level-4 Base Address

- This is an architectural limit. A given processor implementation may support fewer bits

- Page fault
- Miss entry
- Mismatched permission

- Page table base region

- Page table entry

- Page table format

- Page offset