4410 Lecture 12: Page tables

- Computing sizes of data structures
- Page table
- Hierarchical page table
- Inverted / hashed PT
On context switch, flush TLB (points to old proc pages, not new).

Tagged TLB: each entry has a process ID (3, 4, 5 bits) to identify process, only use entries for current process.
2^32 addresses 32-bit logical addr. space

2^32 bytes 30-bit phys.

4 KB page size

PI log addr space


\[
\begin{align*}
\text{phys. mem} & \quad \text{2}^{32} \text{ bytes/frame.} \\
\text{phys. mem} & \quad \text{2}^{32} \text{ bytes} \\
\end{align*}
\]

\[
\frac{2^{32} \text{ bytes}}{\text{log. addr. space}} \cdot \frac{1 \text{ page}}{2^{12} \text{ bytes/addr. space}} = 2^{20} \text{ pages}.
\]

2^{32} bytes

\[
\frac{2^{32} \text{ page}}{2^{12} \text{ bytes/page}} = 2^{20} \text{ page}
\]

Useful

bit: power sizes.

\[
\begin{align*}
2^{10} & \approx 1000 = 1 \text{ k} \\
2^{20} & \approx 1000 \text{ k} = 1 \text{ M (million)} \\
2^{30} & \approx 1000 \text{ M} = 1 \text{ G (billion)}
\end{align*}
\]

Write down units.

TLB

\[
\begin{array}{c|c|c}
\multicolumn{3}{c}{\text{Page #}} \\
\hline
\text{Frame #} & \text{Pens} \\
\hline
<20 \text{ bit frame #>} & \times \times \times \times \\
<20 \text{ bit frame #>} & \times \times \times \times \\
\vdots & \\
\end{array}
\]

Can't store this in TLB (hardware)
Page table (mapping page \( \rightarrow \) frame \#s)

- frame \# + perms, round up
  \[ 2^{20} \text{ entries} \cdot \frac{4 \text{ bytes}}{\text{entry}} = 2^{24} = 16 \text{ Mbytes}. \]

TLB just contains handful of entries (e.g. 16 or 32),
raise HW exception if access page not in TLB (TLB miss)

Page table stored in memory.

OS to handle TLB miss:
- box up page table (in memory),
- find frame \# & permissions,
- stick \# in TLB
- continue.
How to store page table? (mapply of page # => frame #)(perms)

- Hashmap
- Array
- Tree
Store page table in an array?

entries contain frame + pages + other stuff

\[
\frac{2^{20} \text{ entries}}{32 \text{ bits/entry}} = \frac{4 \text{ bytes}}{2^{20} \text{ bytes in pt.}} = 4 \text{ MB/pt.}
\]

Page mem

- area for holding processes' data
- Page tables.

Instead of storing whole page table in one contiguous chunk, page 1.
2nd level page table

(pages of the page table)

PT

log. addr. space

2^10 pages = 2^32 bytes

= 4 kB

41/entry

= 2^12 bytes

(size: 2^10 entries, each entry is frame that holds = 4 bytes corresponding PT)

= 2^12 bytes = 4 kB

don't need to page it.

phys. mem.

32-byte frame

PI pages

frames

PI PT frame

PT and L1 Pt table

log. addr. space

2^32 bytes

= 2^32 / 2^12 pages

byte [2^32]
How to find page \( u \) in 2-LV page table.

**virt address:**
- 32 bits.
- 10 bits
- page
- 10 bits
- entry in P0P
- 12 bits
- offset

**book analogy**
- \( 2^{10} \) bits.
- entry
- 4-byte log. addr.
- face

\[ \begin{align*}
32 \text{ bits} & : \text{32, 476, 325} \\
1000 \text{ words} & : 832, 476, 325
\end{align*} \]
OS algorithm for handling TLB miss (to address a)

- look at first 10 bits of a, index into top level page table.
  \[ \Rightarrow \text{frame \# holds correct PoPT} \]

- use next 10 bits of a to index into PoPT.
  \[ \Rightarrow \text{frame \# holds correct page} \]

- load that frame \# (f:perms) into TLB.

- restart process just before access.
  (process will access a again, TLB looks last 12 bits to offset into frame.)