Paging

- Fixed sized pages (4096)
- Same for physical memory (frames, physical pages)

VAS still contiguous

→ base & bound (contiguous translation)

→ paging: translation for each page

page # -> frame #

0x1000 - 0x1fff pages 1
0x2000 - 0x2fff page 2
Page Table
→ a translation array, indexed by page #

How to speed up translation?
→ cache the result!

- Page table walk (translation) done by the hardware
- Page table structure defined by architecture

Translation Lookaside Buffer (TLB)
\( (pid + \text{page} \#) \Rightarrow \text{frame} \# \)
Virtual Address

Page# Offset

Translation Lookaside Buffer (TLB)

Virtual Page Page Frame Access

Matching Entry

Physical Memory

Frame Offset

Page Table Lookup
Single array: many pages, lots of entries (space), one per process

How to reduce cost?

LargerPages: bigger pagesize \( \Rightarrow \) less pages & less translation overhead (2MB & 1GB supported) *problem: internal fragmentation

Inverted Page Table: frame \# \( \Rightarrow \) page \#, global table

hash function (pid + page\#)

\[ \text{frame \#} \quad \text{pid, page} \# \]

\[ \rightarrow \text{ takes time, hash collision} \]

Multi-level Page Table