The Page Table is part of the virtual memory

- Each process has its own page table stored in memory starting at a specific address indicated in the page address register.
- The page table and page address register are part of the process context (along with the PC, stack pointer, registers …)
- A memory reference (if hits in main memory) requires two memory operations???
- A page fault (main memory miss) requires a disk operation.

Multiple processes share the physical memory

- Multiple processes share the physical memory by using separate page tables for each process. Each process has its own page table that maps virtual pages to physical pages.
- When executing a process, the CPU uses its page table to locate the physical page for a given virtual address.
- If a page fault occurs, a disk operation is required to load the missing page into memory.
Multiple processes share the physical memory

Caching the page table in a Translation Lookaside Buffer
Caching the page table in a TLB

Making Address Translation Fast

With the TLB, we avoid accessing memory twice for each memory reference.

TLB is a cache for the Page Table
• Small
• Highly associative
• Block size = 1

What if we get a TLB miss (page table entry is not in TLB)?
• Get the entry from the page table (from memory) and load it to the TLB.
• may have to replace a TLB entry (the LRU)
Example

Page Tables can be very large

Example: If VS = 32-bit address (byte address) and page size = 4KB
→ VS = 1 million pages, page table = 1 Million entry.
→ if each table entry = 4 bytes → page table occupies 4MB = 1024 pages

The physical memory may contain only a few pages of the page table.
Moreover, a fraction of the page table entries that are in memory are cached in TLB.

Note: The page table is stored in pages within the virtual space. Hence the page table contains entries for its own pages.
Multi level Page Tables (multi level PT)

Example: If VS = 32-bit address (byte address) and page size = 4KB
    ⇒ VS = 1 million pages and page table = 1 Million entry.
    ⇒ if each table entry = 4 bytes ⇒ each page can hold 1024 entries of the PT
    ⇒ page table occupies 4MB = 1024 pages

A level 1 PT occupies one page and contains one entry for each of the 1024 pages of a level 2 PT

Notes:
• A large number of pages in the VS are not used (empty).
• Hence a large number of entries of the PT are never accessed
• Memory foot-print = the part of the VS which is actually used (accessed)
• The level 1 PT is always kept in physical memory and is pointed to by a “base register”
• Pages of the level 2 PT are brought to physical memory on demand.

Alpha 21264 example (3-levels page tables)
**TLBs and caches**

In this example, the TLB is fully associative.

**The whole picture**

CPU (pipeline) stalls if:
- TLB miss (but no page fault)
- Cache miss

The OS is invoked to move a page from disk (where virtual pages reside) to physical memory.
**TLBs and caches**

- Page table walk to get the PT entry into the TLB (hardware)
- If PT indicates that page is not in memory, then service page fault (software – OS)

Note that there cannot be a page fault in case of a TLB hit – there is no reason for the PT entry of a page to be in the TLB if the page is not in memory.

May need to write back a dirty block

or, depending on being write back or write through