Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Informatik
Computer Science

Advanced Operating Systems (263-3800-00L)
Page Tables

Timothy Roscoe & Andrew Baumann

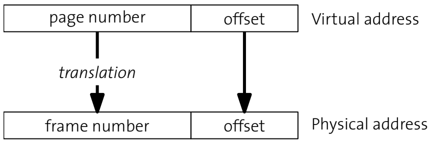
© Systems Group | Department of Computer Science | ETH Zürich Thursday 30 September 2010

Outline

Review of page-based virtual memory

Page table structures
- Linear page table
- Hierarchical page table
- Virtual linear array page table
- Hashed page table
- Software-loaded TLBs

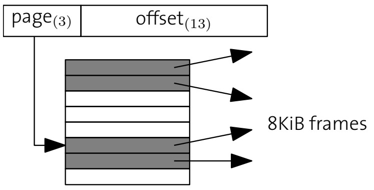
Page-based virtual memory (review)



- **Page table** stores the translation
  - TLB is a cache for the page table(s)
  - May be loaded by hardware or software
- **Base page size** determined by hardware
  - 4KiB, 8KiB typical
- **Superpage** is a multiple (2^n) of the base page size
  - Used to increase TLB coverage, reduce contention

Linear page table

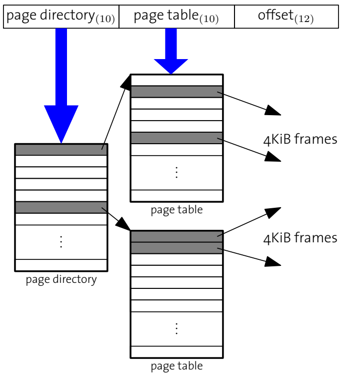
- Array of frame numbers indexed by page number
- Example: PDP-11 (16-bit address, 8KiB pages)



✗ Not feasible for a 32-bit address space

Hierarchical page table

Example: i386 page tables



✔ Saves memory (vs. linear PT) for mostly-empty address spaces
  - ✗ More memory references required for lookup
- Easy to implement in hardware
  - Used by x86, ARM, SPARC (among others)

✔ Natural support for superpages
- Also the generic page-table "abstraction" in Linux

Concrete example: ARM page tables

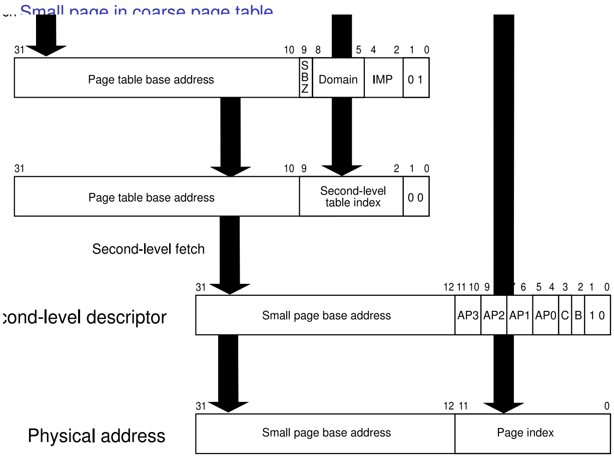
MMU supports:

Sections	1MiB (somewhat like a page)
Large pages	64KiB
Small pages	4KiB
Tiny pages	1KiB

Two-level hierarchical page table:
- First-level table (16KiB) holds section translations and pointers to second-level tables
- Second-level tables may be coarse or fine
  - Coarse tables (1KiB) hold large and small page translations
  - Fine tables (4KiB) can also hold tiny page translations

ARM page-table lookup

Small page in coarse page table



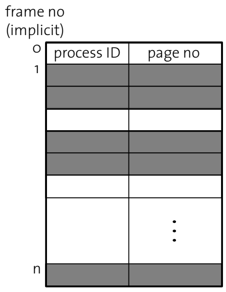
Problems with hierarchical page tables

- Depth scales with size of virtual address space
  - Must be 5–6 levels deep for a full 64-bit address space
  - AMD64 (48-bit virtual address) needs 4 levels
- A sparse address-space layout requires lots of memory for (mostly empty) tables
  - Not a big problem for the traditional UNIX memory model

Virtual linear array page table

- Same as a linear page table, but it is stored in *virtual memory*
- In practice, much like a hierarchical page table
  - Root node located in physical memory (or pinned in TLB)
  - Leaf nodes allocated in contiguous virtual memory region
  - Unused parts are left unmapped
  - ✔ Fewer memory accesses, because TLB caches top-level entries
    - ✗ (Slightly) higher TLB pressure
- Used by Itanium (short format)
  - Hardware walker does not handle nested TLB misses

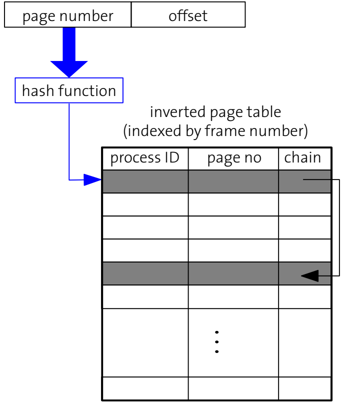
Inverted page table



- Array of page numbers indexed by frame number
  - To lookup page, search for matching frame
- ✔ Size scales with physical memory
- ✔ One table for the whole system, not one per process
- Used in the first virtual memory system (the Atlas computer)
  - Makes sense if the virtual address space is smaller than physical
  - Obviously, this is impractical today

Hashed page table

Inverted page table with a hash for fast lookup



- Used by many 64-bit architectures
  - IBM POWER
  - HP PA-RISC
  - Itanium (long format)
- ✔ Scales with physical memory
- ✔ One table for the whole system
- ✗ Difficult to share memory between processes
- ✗ Difficult to support superpages

Software-loaded TLBs

- On these architectures, every TLB miss is handled by the OS
  - For example: MIPS, Alpha
  - Possible on most others by disabling the hardware walker
- The choice of page-table format is open to the OS designer
  - Clustered page table
  - Guarded page table
  - Variable radix page table
  - …
- Performance issues due to the cost of trapping to software
  - Generally slower than hardware walkers
  - Worsens the impact of TLB thrashing

Summary

- ARM has a hardware-walked page table, but L4 manages this
- You thus have freedom on what you implement in SOS
- We've shown a few common examples,
  many variations and alternatives are possible
  - Check the literature if interested
- You'll need to support demand paging (swap) later on