Page Table Implementations

- Issues in implementing the page table
- Multi-level page table
- Inverted page table
- Hashed page table

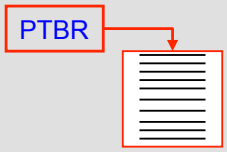
Implementation of Page Table

Observation: Page table must store one entry per page.

Question: How to store the page table?

Naïve solution: Store page table as array in memory.

- Page-Table-Base-Register (PTBR) points to start of array.
- New PTBR loaded during address space switch.



Problem: Page table can become very large.

Example: 32-bit address space (4GB), 4kB pages, 4B entries

=> 2^{20} entries in page table

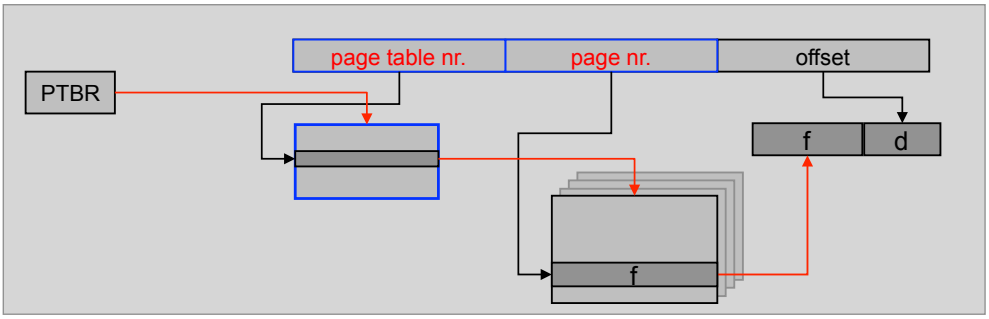
=> we need 4MB per process (!)

Hierarchical (Multilevel) Paging

Problem: Page tables can become very large!

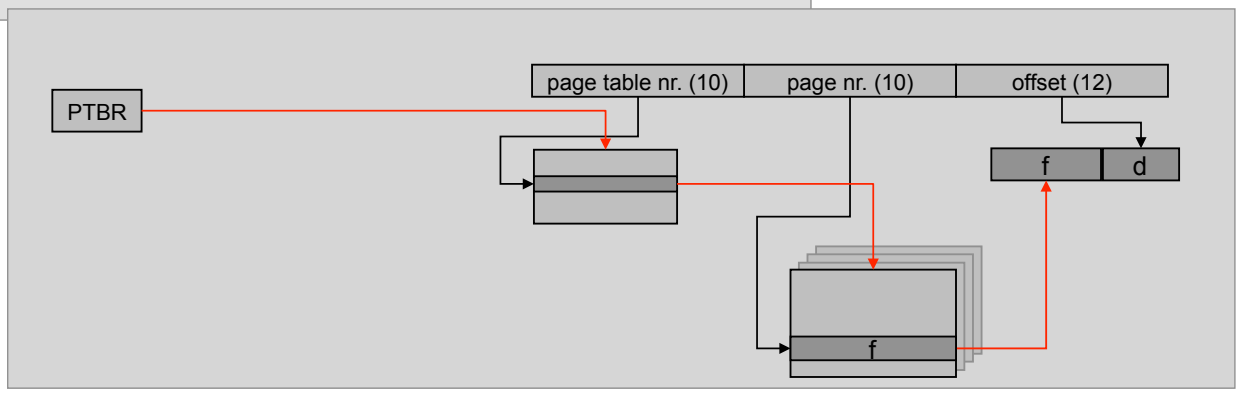
Observation: Most of the address space is not used.
=> Most of the page table is empty.

Solution: Page the page table!



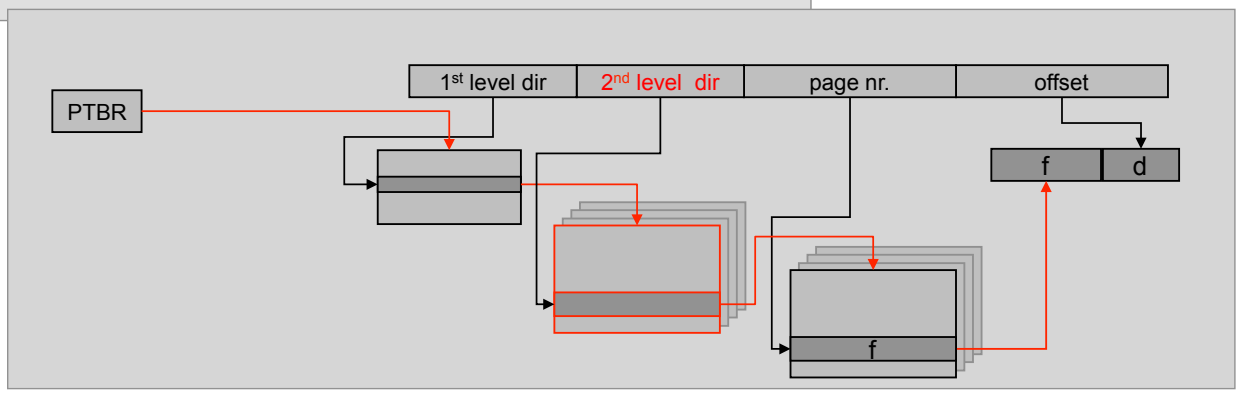
Hierarchical (Multilevel) Paging

Example: Two-level Paging (x86)



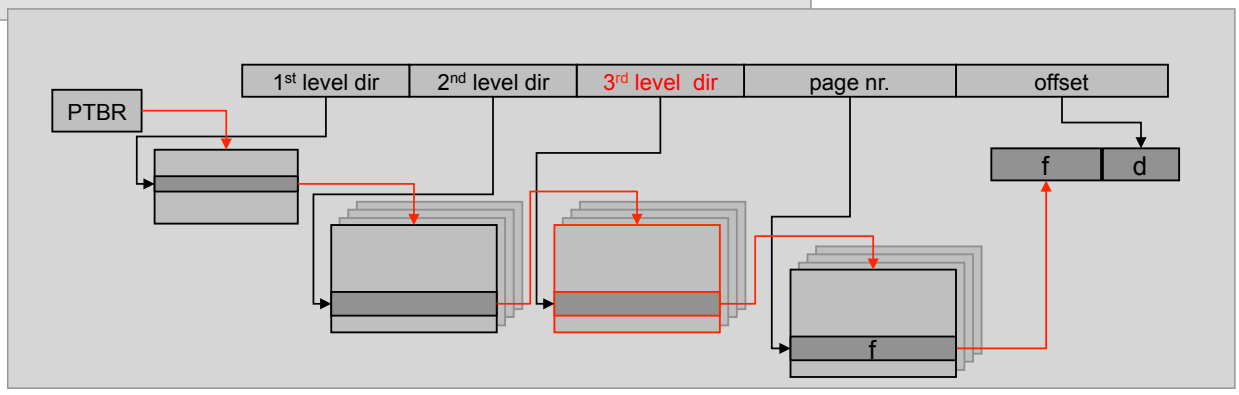
Hierarchical (Multilevel) Paging

Example: Three-level Paging (SPARC)

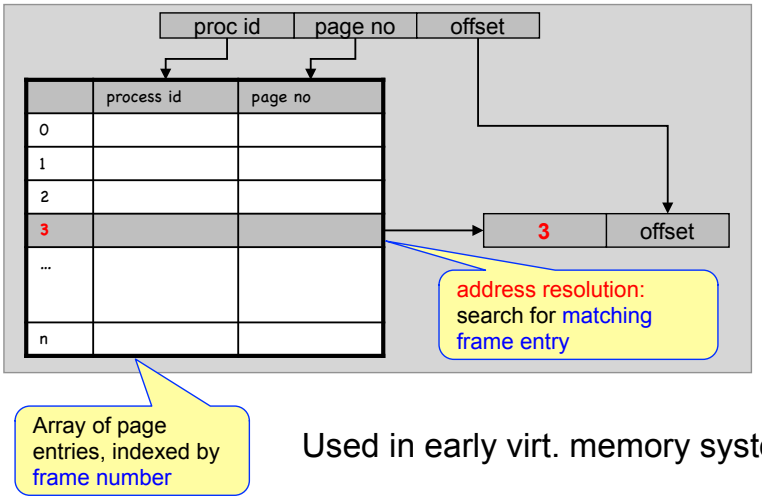


Hierarchical (Multilevel) Paging

Example: Four-level Paging (68030)



Variations: Inverted Page Table

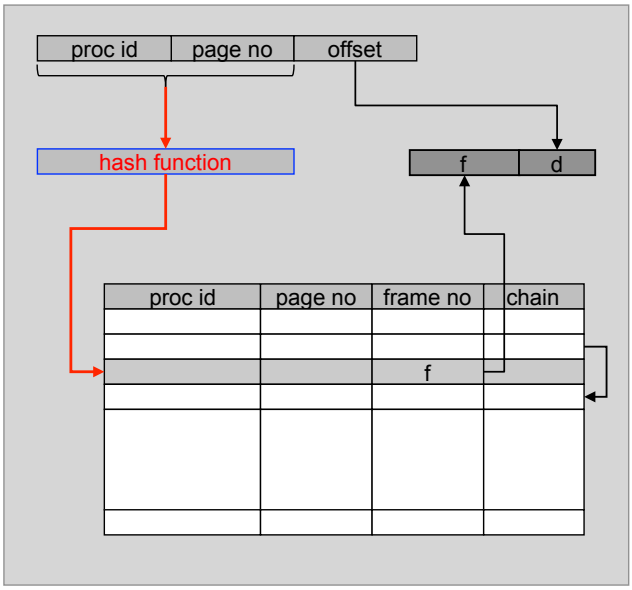


Pros:
- Scales with physical memory
- One table for whole system

Cons:
- Long search times for large physical memories

Used in early virt. memory systems, such as the Atlas computer.

Variations: Hashed Page Table



Used by many 64bit architectures:
IBM POWER, HP PA-RISC, Itanium

Pros:
- Scales with physical memory
- One table for whole system

Cons:
- How about collisions?

Summary

- Page table size is an issue for large address spaces.
- Multi-level page table
- Inverted page table
- Hashed page table