Virtual Memory

Sandhya Dwarkadas

Virtual Memory

Provide programs with the illusion of larger memory than available

Allow multiple programs to share memory

- protection
- dynamic loading (relocatable code/data)

Memory Organization

Block size - 4-16 KBytes

Placement - where is a block placed

Location - how do you locate a block

Re-placement - which block do you replace

Write policy - write back with dirty bit

Mapping or Location

- Paging - fixed-size blocks, uniform address space
- Segmentation - variable-size blocks, two-part address - segment number and segment offset
- Hierarchical division of fixed-size blocks

Page Replacement

Least recently used

- use bit or reference bit
- set when a page is accessed, periodically cleared
- select a page that is among the least recently referenced

Page Tables

Use bounds register to limit the size of a page table

Multi-level (hierarchical) page tables

Inverted page tables

- size proportional to number of physical pages in main memory
- use a hashing function
- look-up process more complex - possibly associative

Making Address Translation Fast: TLB

Translation Lookaside Buffer - Cache of page table to exploit locality of access

TLB size - 32-1024 entries

Block size - 1-2 page table entries (typically 4-8 bytes each)

n-way, or fully associative to improve hit rate

Hit time - 1/2-1 cycle

Miss penalty - 10-30 cycles

Miss rate - .01%-1%

Bits in page tables -

1. valid bit
2. reference bit
3. dirty bit
4. access permission bits
5. process id number
6. virtual page number tag

Cache Addressing

Virtually addressed caches

- aliasing problem

Physically addressed caches

- increased latency due to translation

Virtually indexed, physically tagged

Implementing Protection With Virtual Memory

Avoid one process reading/writing other process's data

- Each process with its own virtual address space mapped to physically disjoint pages
- Support two modes - user and kernel
- Provide portion of CPU state that can be read but not written by user (includes page table pointer and user/supervisor bits)
- Use `syscalls` to perform protected operations
- Read/write protection capabilities
- Process id included in TLB
- Context switch actions

– change page table register
– change register containing process identifier
– eliminates need to clear TLB

Handling Page Faults

Instructions must be restartable (side-effect free)

TLB entry filled in hardware or software

Control transferred to OS (software) to bring page in if not present

EPC and Cause Register used to interrupt process

Entire state of the process saved - general purpose registers, floating point registers, page table address registers

Must mask interrupts while performing critical operations

Look up page table entry and find location on disk

Choose page to replace and write back to disk if dirty

Start a read to bring referenced page in from disk

Context switch back to user mode and allow other processes to continue

Restore process on completing read

Memory Management in the Alpha Processor

Alpha AXP21064 in a DEC3000/Model 800 Workstation

- 8-Kbyte direct-mapped, 32-byte block, write-through no write allocate, split instruction and data cache, with 4-block write buffer (on-chip)
- 1 MByte board-level cache
- 256-bit wide bus - 4 64-bit modules
- Paged segmentation - 3 segments, 3 levels of page tables
- Fully-associative, 12-entry instruction, 32-entry data TLB

Memory Management in the Pentium Pro

8-KByte, 4-way set associative, 32-byte block, write-back, LRU replacement, split instruction and data cache

256-512KByte secondary cache as a separate die integrated into the same package

Segmented-paged virtual memory organization - 4KByte pages, 16K segments

Multiple levels of page tables, each 4MBytes (of data) each

32-entry instruction, 64-entry data TLB, 4-way set associativity, Pseud0-LRU replacement, misses handled in hardware