Operating Systems & Memory Systems: Address Translation

CPS 220
Professor Alvin R. Lebeck

Outline

• Address Translation
  – basics
  – 64-bit Address Space
• Managing memory
• OS Performance

Throughout
• Review Computer Architecture
• Interaction with Architectural Decisions
Computer Architecture

- **Interface Between Hardware and Software**

  - Applications
    - Operating System
    - Compiler

  - Hardware
    - CPU
    - Memory
    - I/O
    - Multiprocessor
    - Networks

  - Software

  This is IT
### Memory Hierarchy 101

- **Very fast 1ns clock**
- **Multiple Instructions per cycle**
- **SRAM, Fast, Small**
  - Expensive
- **DRAM, Slow, Big, Cheap**
  - (called physical or main)
- **Magnetic, Really Slow, Really Big, Really Cheap**

=> Cost Effective Memory System (Price/Performance)

---

### Virtual Memory: Motivation

- **Virtual**
- **Physical**

- **Process = Address Space + thread(s) of control**
- **Address space = PA**
  - programmer controls movement from disk
  - protection?
  - relocation?
- **Linear Address space**
  - larger than physical address space
    - 32, 64 bits v.s. 28-bit physical (256MB)
- **Automatic management**
Virtual Memory

• **Process** = virtual address space + thread(s) of control

• **Translation**
  – VA -> PA
  – What physical address does virtual address A map to
  – Is VA in physical memory?

• **Protection (access control)**
  – Do you have permission to access it?

Virtual Memory: Questions

• How is data found if it is in physical memory?

• **Where can data be placed in physical memory?**
  Fully Associative, Set Associative, Direct Mapped

• What data should be replaced on a miss?
  (Take CPS210 …)
Segmented Virtual Memory

- Virtual address \(2^{32}, 2^{64}\) to Physical Address mapping \(2^{30}\)
- Variable size, base + offset, contiguous in both VA and PA

Intel Pentium Segmentation

Logical Address

- Seg Selector
- Offset
- Global Descriptor Table (GDT)
- Segment Descriptor
- Segment Base Address

Physical Address Space
Pentium Segmentation (Continued)

- **Segment Descriptors**
  - Local and Global
  - base, limit, access rights
  - Can define many
- **Segment Registers**
  - contain segment descriptors (faster than load from mem)
  - Only 6
- **Must load segment register with a valid entry before segment can be accessed**
  - generally managed by compiler, linker, not programmer

Paged Virtual Memory

- **Virtual address \(2^{32}, 2^{64}\) to Physical Address mapping \(2^{28}\)**
  - virtual page to physical page frame
  - Fixed Size units for access control & translation
Page Table

- Kernel data structure (per process)
- **Page Table Entry (PTE)**
  - VA -> PA translations (if none page fault)
  - access rights (Read, Write, Execute, User/Kernel, cached/uncached)
  - reference, dirty bits
- **Many designs**
  - Linear, Forward mapped, Inverted, Hashed, Clustered
- **Design Issues**
  - support for aliasing (multiple VA to single PA)
  - large virtual address space
  - time to obtain translation

Alpha VM Mapping (Forward Mapped)

- "64-bit" address divided into 3 segments
  - seg0 (bit 63=0) user code/heap
  - seg1 (bit 63 = 1, 62 = 1) user stack
  - kseg (bit 63 = 1, 62 = 0) kernel segment for OS
- Three level page table, each one page
  - Alpha 21064 only 43 unique bits of VA
  - (future min page size up to 64KB => 55 bits of VA)
- **PTE bits; valid, kernel & user read & write enable (No reference, use, or dirty bit)**
  - What do you do for replacement?
Inverted Page Table (HP, IBM)

- One PTE per page frame
  - only one VA per physical frame
- Must search for virtual address
- More difficult to support aliasing
- Force all sharing to use the same VA

Intel Pentium Segmentation + Paging

Logical Address

Segment Base Address

Global Descriptor Table (GDT)

Segment Descriptor

Linear Address Space

Page Table

Physical Address Space
The Memory Management Unit (MMU)

- **Input**
  - virtual address

- **Output**
  - physical address
  - access violation (exception, interrupts the processor)

- **Access Violations**
  - not present
  - user v.s. kernel
  - write
  - read
  - execute

Translation Lookaside Buffers (TLB)

- **Need to perform address translation on every memory reference**
  - 30% of instructions are memory references
  - 4-way superscalar processor
  - at least one memory reference per cycle

- **Make Common Case Fast, others correct**

- **Throw HW at the problem**

- **Cache PTEs**
Fast Translation: Translation Buffer

- Cache of translated addresses
- Alpha 21164 TLB: 48 entry fully associative

TLB Design

- Must be fast, not increase critical path
- Must achieve high hit ratio
- Generally small highly associative
- Mapping change
  - page removed from physical memory
  - processor must invalidate the TLB entry
- PTE is per process entity
  - Multiple processes with same virtual addresses
  - Context Switches?
- Flush TLB
- Add ASID (PID)
  - part of processor state, must be set on context switch
Hardware Managed TLBs

- Hardware Handles TLB miss
- Dictates page table organization
- Complicated state machine to “walk page table”
  - Multiple levels for forward mapped
  - Linked list for inverted
- Exception only if access violation

Software Managed TLBs

- Software Handles TLB miss
- Flexible page table organization
- Simple Hardware to detect Hit or Miss
- Exception if TLB miss or access violation
- Should you check for access violation on TLB miss?
Mapping the Kernel

- **Digital Unix Kseg**
  - kseg (bit 63 = 1, 62 = 0)
- Kernel has direct access to physical memory
- One VA->PA mapping for entire Kernel
- **Lock (pin) TLB entry**
  - or special HW detection

Considerations for Address Translation

- **Large virtual address space**
  - Can map more things
    - files
    - frame buffers
    - network interfaces
    - memory from another workstation
- **Sparse use of address space**
- **Page Table Design**
  - space
  - less locality => TLB misses
- **OS structure**
  - microkernel => more TLB misses
Address Translation for Large Address Spaces

- **Forward Mapped Page Table**
  - grows with virtual address space
    - worst case 100% overhead not likely
  - TLB miss time: memory reference for each level

- **Inverted Page Table**
  - grows with physical address space
    - independent of virtual address space usage
  - TLB miss time: memory reference to HAT, IPT, list search

Hashed Page Table (HP)

- **Combine Hash Table and IPT** [Huck96]
  - can have more entries than physical page frames
- **Must search for virtual address**
- **Easier to support aliasing than IPT**
- **Space**
  - grows with physical space
- **TLB miss**
  - one less memory ref than IPT
Clustered Page Table (SUN)

- Combine benefits of HPT and Linear [Talluri95]
- Store one base VPN (TAG) and several PPN values
  - virtual page block number (VPBN)
  - block offset

Reducing TLB Miss Handling Time

- Problem
  - must walk Page Table on TLB miss
  - usually incur cache misses
  - big problem for IPC in microkernels

- Solution
  - build a small second-level cache in SW
  - on TLB miss, first check SW cache
    » use simple shift and mask index to hash table
Next Time

- More TLB issues
- Virtual Memory & Caches
- Multiprocessor Issues