ELEC2041
Microprocessors and Interfacing
Lectures 37: Cache & Virtual Memory Review
http://webct.edtec.unsw.edu.au/
June 2006
Saeid Nooshabadi
saeid@unsw.edu.au

Survey Result

<table>
<thead>
<tr>
<th>Answer</th>
<th>Value</th>
<th>Cumulative Frequency Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0%</td>
<td>19 Interrupts &amp; Exceptions</td>
</tr>
<tr>
<td>b</td>
<td>0%</td>
<td>29 VM &amp; Cache</td>
</tr>
<tr>
<td>c</td>
<td>0%</td>
<td>16 Function</td>
</tr>
<tr>
<td>d</td>
<td>0%</td>
<td>23 Float</td>
</tr>
<tr>
<td>e</td>
<td>0%</td>
<td>11 Take Questions from Students</td>
</tr>
<tr>
<td>f</td>
<td>0%</td>
<td>4 Hard Disk Operation</td>
</tr>
<tr>
<td>g</td>
<td>0%</td>
<td>1 Link list &amp; Circular Buffer</td>
</tr>
<tr>
<td>h</td>
<td>0%</td>
<td>3 Concepts in Embedded Systems</td>
</tr>
<tr>
<td>i</td>
<td>0%</td>
<td>1 SDRAM</td>
</tr>
<tr>
<td>j</td>
<td>0%</td>
<td>1 Do Nothing (Ignorance is Blissful)</td>
</tr>
</tbody>
</table>

Review (#1/3)
° Apply Principle of Locality Recursively
° Reduce Miss Penalty? add a (L2) cache
° Manage memory to disk? Treat as cache
  • Included protection as bonus, now critical
  • Use Page Table of mappings vs. tag/data in cache
° Virtual memory to Physical Memory Translation too slow?
  • Add a cache of Virtual to Physical Address Translations, called a TLB

Review (#2/3)
° Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always swap or base/bound via segmentation
° Spatial Locality means Working Set of Pages is all that must be in memory for process to run fairly well
° TLB to reduce performance cost of VM
° Need more compact representation to reduce memory size cost of simple 1-level page table (especially 32 – 64-bit addresses)
**Why Caches?**

- "Moore's Law": CPU 60%/yr.
- DRAM 7%/yr.

---

**Processor-Memory Performance Gap:**
- Processor-Memory Performance Gap: (grows 50% / year)

---

- 1989 first Intel CPU with cache on chip;
- 1999 gap “Tax”; 37% area of Alpha 21164, 61% StrongArm SA110, 64% Pentium Pro

---

**Memory Hierarchy Pyramid**

- Central Processor Unit (CPU)
- Levels in memory hierarchy
- Increasing Distance from CPU, Decreasing Cost / MB

---

- Size of memory at each level
- Principle of Locality (in time, in space) +
- Hierarchy of Memories of different speed, cost; exploit to improve cost-performance

---

**Why virtual memory? (#1/2)**

- Protection
  - regions of the address space can be read only, execute only, . . .
- Flexibility
  - portions of a program can be placed anywhere, without relocation (changing addresses)
- Expandability
  - can leave room in virtual address space for objects to grow
- Storage management
  - allocation/deallocation of variable sized blocks is costly and leads to (external) fragmentation; paging solves this

---

**Why virtual memory? (#2/2)**

- Generality
  - ability to run programs larger than size of physical memory
- Storage efficiency
  - retain only most important portions of the program in memory
- Concurrent I/O
  - execute other processes while loading/dumping page
Virtual Memory Review (#1/4)
° User program view of memory:
  • Contiguous
  • Start from some set address
  • Infinitely large
  • Is the only running program

° Reality:
  • Non-contiguous
  • Start wherever available memory is
  • Finite size
  • Many programs running at a time

Virtual Memory Review (#2/4)
° Virtual memory provides:
  • illusion of contiguous memory
  • all programs starting at same set address
  • illusion of infinite memory
  • protection

Virtual Memory Review (#3/4)
° Implementation:
  • Divide memory into “chunks” (pages)
  • Operating system controls pagetable that maps virtual addresses into physical addresses
  • Think of memory as a cache for disk
  • TLB is a cache for the pagetable

Why Translation Lookaside Buffer (TLB)?
° Paging is most popular implementation of virtual memory (vs. base/bounds in segmentation)
° Every paged virtual memory access must be checked against Entry of Page Table in memory to provide protection
° Cache of Page Table Entries makes address translation possible without memory access (in common case) to make translation fast
Virtual Memory Review (#4/4)

° Let's say we're fetching some data:
  • Check TLB (input: VPN, output: PPN)
    - hit: fetch translation
    - miss: check pagetable (in memory)
      ➢ pagetable hit: fetch translation, return translation to TLB
      ➢ pagetable miss: page fault, fetch page from disk to memory, return translation to TLB
  • Check cache (input: PPN, output: data)
    - hit: return value
    - miss: fetch value from memory

Three Advantages of Virtual Memory

1) Translation:
   • Program can be given consistent view of memory, even though physical memory is scrambled
   • Makes multiple processes reasonable
   • Only the most important part of program ("Working Set") must be in physical memory
   • Contiguous structures (like stacks) use only as much physical memory as necessary yet still grow later

2) Protection:
   • Different processes protected from each other
   • Different pages can be given special behavior
     - (Read Only, Invisible to user programs, etc).
   • Privileged data protected from User programs
   • Very important for protection from malicious programs ⇒ Far more “viruses” under Microsoft Windows

3) Sharing:
   • Can map same physical page to multiple users ("Shared memory")
4 Questions for Memory Hierarchy

Q1: Where can a block be placed in the upper level? (Block placement)

Q2: How is a block found if it is in the upper level? (Block identification)

Q3: Which block should be replaced on a miss? (Block replacement)

Q4: What happens on a write? (Write strategy)

Q1: Where block placed in upper level?

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Mod Number of Sets

Q2: How is a block found in upper level?

- Direct indexing (using index and block offset), and tag comparing
- Increasing associativity shrinks index, expands tag

Q3: Which block replaced on a miss?

- Easy for Direct Mapped
- Set Associative or Fully Associative:
  - Random
  - LRU (Least Recently Used)

Miss Rates
Associativity:

<table>
<thead>
<tr>
<th></th>
<th>2-way</th>
<th>4-way</th>
<th>8-way</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LRU</td>
<td>Ran</td>
<td>LRU</td>
</tr>
<tr>
<td>16 KB</td>
<td>5.2%</td>
<td>5.7%</td>
<td>4.7%</td>
</tr>
<tr>
<td>64 KB</td>
<td>1.9%</td>
<td>2.0%</td>
<td>1.5%</td>
</tr>
<tr>
<td>256 KB</td>
<td>1.15%</td>
<td>1.17%</td>
<td>1.13%</td>
</tr>
</tbody>
</table>
Q4: What happens on a write?

° **Write through**—The information is written to both the block in the cache and to the block in the lower-level memory.

° **Write back**—The information is written only to the block in the cache. The modified cache block is written to main memory only when it is replaced.

• **is block clean or dirty?**

° **Pros and Cons of each?**

• **WT:** read misses cannot result in writes

• **WB:** no writes of repeated writes

---

Address Translation & 3 Exercises

Virtual Address

<table>
<thead>
<tr>
<th>VPN-tag</th>
<th>INDEX</th>
<th>Offset</th>
</tr>
</thead>
</table>

**TLB**

- TLB-tag: PPN
- TLB-tag: Page Number

<table>
<thead>
<tr>
<th>PPN</th>
<th>Offset</th>
</tr>
</thead>
</table>

**Physical Address**

**Address Translation Exercise 1 (#1/2)**

° **Exercise:**

- 40-bit VA, 16 KB pages, 36-bit PA

° **Number of bits in Virtual Page Number?**

- a) 18; b) 20; c) 22; d) 24; e) 26; f) 28

° **Number of bits in Page Offset?**

- a) 8; b) 10; c) 12; d) 14; e) 16; f) 18

° **Number of bits in Physical Page Number?**

- a) 18; b) 20; c) 22; d) 24; e) 26; f) 28

---

Who is He?

° **HE HAS PLAYED GOLF AT A PRO TOURNAMENT IN HAWAII,** acted in the Japanese TV show “Astro Boy,” danced and sung on stages from Las Vegas to Hong Kong, and even conducted the Tokyo Philharmonic Orchestra in a rousing rendition of Beethoven’s Fifth Symphony.

° And he’s barely a year old and not quite 60 cm tall.

° Meet Qrio, pronounced “curio,” the biped humanoid robot from Sony Corp., Tokyo. The dream child of Yoshihiro Kuroki, general manager of Sony Entertainment Robot Co. in Shinbashi, Japan

° Qrio is a remarkable assemblage:

  • of three powerful microprocessors, 38 motor actuators, three accelerometers, two charge coupled device (CCD) cameras, and seven microphones.

  • Qrio can hear, speak, sing, recognize objects and faces, walk, run, dance, and grasp objects. It can even pick itself up if it falls.

  • At the moment, there are dozens of Qrios in existence. Will sell for $12,000 when hits the market
Address Translation Exercise 1 (#2/2)

- 40-bit virtual address, 16 KB (2^{14} B)
  - Virtual Page Number (26 bits) Page Offset (14 bits)
- 36-bit virtual address, 16 KB (2^{14} B)
  - Physical Page Number (22 bits) Page Offset (14 bits)

Address Translation Exercise 2 (#1/2)

- Exercise:
  - 40-bit VA, 16 KB pages, 36-bit PA
  - 2-way set-assoc TLB: 256 "slots", 2 per slot
- Number of bits in TLB Index?
  - a) 8; b) 10; c) 12; d) 14; e) 16; f) 18
- Number of bits in TLB Tag?
  - a) 18; b) 20; c) 22; d) 24; e) 26; f) 28
- Approximate Number of bits in TLB Entry?
  - a) 32; b) 36; c) 40; d) 42; e) 44; f) 46

Address Translation 2 (#2/2)

- 2-way set-assoc data cache, 256 (2^8) "slots", 2 TLB entries per slot => 8 bit index
  - TLB Tag (18 bits) TLB Index (8 bits) Page Offset (14 bits)
  - Virtual Page Number (26 bits)
- Data Cache Entry: Valid bit, Dirty bit, Access Control (2-3 bits?), Virtual Page Number, Physical Page Number
  - V D Access (3 bits) TLB Tag (18 bits) Physical Page No. (22 bits)

Address Translation Exercise 3 (#1/2)

- Exercise:
  - 40-bit VA, 16 KB pages, 36-bit PA
  - 2-way set-assoc TLB: 256 "slots", 2 per slot
  - 64 KB data cache, 64 Byte blocks, 2 way S.A.
- Number of bits in Cache Offset?
  - a) 6; b) 8; c) 10; d) 12; e) 14; f) 16
- Number of bits in Cache Index?
  - a) 6; b) 9; c) 10; d) 12; e) 14; f) 16
- Number of bits in Cache Tag?
  - a) 18; b) 20; c) 21; d) 24; e) 26; f) 28
- Approximate No. of bits in Cache Entry?
Address Translation 3 (#2/2)

- 2-way set-associative data cache, 64K/64 = 1K \(2^{10}\) blocks, 2 entries per slot => 512 slots => 9 bit index

Data Cache Entry: Valid bit, Dirty bit, Cache tag + 64 Bytes of Data

Cache/VM/TLB Summary: (#1/3)

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
    - Temporal Locality: Locality in Time
    - Spatial Locality: Locality in Space

- Caches, TLBs, Virtual Memory all understood by examining how they deal with 4 questions:
  1) Where can block be placed?
  2) How is block found?
  3) What block is replaced on miss?
  4) How are writes handled?

Cache/VM/TLB Summary: (#2/3)

- Virtual Memory allows protected sharing of memory between processes with less swapping to disk, less fragmentation than always swap or base/bound in segmentation

- 3 Problems:
  1) Not enough memory: Spatial Locality means small Working Set of pages OK
  2) TLB to reduce performance cost of VM
  3) Need more compact representation to reduce memory size cost of simple 1-level page table, especially for 64-bit address (See COMP3231)

Cache/VM/TLB Summary: (#3/3)

- Virtual memory was controversial at the time: can SW automatically manage 64KB across many programs?
  - 1000X DRAM growth removed controversy

- Today VM allows many processes to share single memory without having to swap all processes to disk; VM protection today is more important than memory hierarchy

- Today CPU time is a function of \(\text{ops, cache misses}\) vs. just \(\text{f(ops)}\):
  What does this mean to Compilers, Data structures, Algorithms?