Virtual Memory: Systems
CSci 2021: Machine Architecture and Organization
Lecture #28, April 1st, 2016
Your instructor: Stephen McCamant
Based on slides originally by:
Randy Bryant, Dave O’Hallaron

Today
- Simple memory system example
- Case study: Core i7/Linux memory system
- Memory mapping

Review of Symbols
- Basic Parameters
  - \( N = 2^n \): Number of addresses in virtual address space
  - \( M = 2^m \): Number of addresses in physical address space
  - \( P = 2^p \): Page size (bytes)
- Components of the virtual address (VA)
  - TLBI: TLB index
  - TLBT: TLB tag
  - VPO: Virtual page offset
  - VPN: Virtual page number
- Components of the physical address (PA)
  - PPO: Physical page offset (same as VPO)
  - PPN: Physical page number
  - CO: Byte offset within cache line
  - CI: Cache index
  - CT: Cache tag

1. Simple Memory System TLB
- 16 entries
- 4-way associative

2. Simple Memory System Page Table
Only show first 16 entries (out of 256)
3. Simple Memory System Cache

* 16 lines, 4-byte block size
* Physically addressed
* Direct mapped

![CT Co PPN PPO]

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>VPN</th>
<th>TLB</th>
<th>Valid</th>
<th>CT</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>00</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>02</td>
<td>04</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>03</td>
<td>07</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>06</td>
<td>02</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>1</td>
<td>09</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>1</td>
<td>05</td>
<td>01</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Address Translation Example #2

**Virtual Address:** 0x0020

![TLBT TLBI]

**Physical Address:**

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>VPN</th>
<th>TLB</th>
<th>Valid</th>
<th>CT</th>
<th>CO</th>
<th>PPN</th>
<th>PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>00</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>02</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>02</td>
<td>04</td>
<td>0</td>
<td>0</td>
<td>03</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>03</td>
<td>07</td>
<td>0</td>
<td>0</td>
<td>04</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>06</td>
<td>02</td>
<td>0</td>
<td>0</td>
<td>05</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>1</td>
<td>09</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>06</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>1</td>
<td>05</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>07</td>
<td>0</td>
</tr>
</tbody>
</table>

Today

- Simple memory system example
- Case study: Core i7/Linux memory system
- Memory mapping

Address Translation Example #3

**Virtual Address:** 0x0020

![TLBT TLBI]

**Physical Address:**

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>VPN</th>
<th>TLB</th>
<th>Valid</th>
<th>CT</th>
<th>CO</th>
<th>PPN</th>
<th>PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>00</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>02</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>02</td>
<td>04</td>
<td>0</td>
<td>0</td>
<td>03</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>03</td>
<td>07</td>
<td>0</td>
<td>0</td>
<td>04</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>06</td>
<td>02</td>
<td>0</td>
<td>0</td>
<td>05</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>1</td>
<td>09</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>06</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>1</td>
<td>05</td>
<td>01</td>
<td>0</td>
<td>0</td>
<td>07</td>
<td>0</td>
</tr>
</tbody>
</table>

Intel Core i7 Memory System

Processor package

- L1 d-cache 32 KB, 8-way
- L1 i-cache 32 KB, 8-way
- L1 unified TLB, 512 entries, 4-way
- L1 unified cache, 256 KB, 8-way
- L2 unified TLB, 4 entries, 4-way
- L3 unified cache 8 MB, 16-way (shared by all cores)

QuickPath interconnect

- 4 links @ 25.6 GB/s each

ODIR3 Memory controller

- 3 x 64 bit @ 10.66 GB/s

- 32 GB/s total (shared by all cores)

To other cores

To I/O bridge

Main memory
**Review of Symbols**

- **Basic Parameters**
  - \( N = 2^L \): Number of addresses in virtual address space
  - \( M = 2^L \): Number of addresses in physical address space
  - \( P = 2^L \): Page size (bytes)

- **Components of the virtual address (VA)**
  - \( \text{TBLI} \): TLB index
  - \( \text{TLB} \): TLB tag
  - \( \text{VPN} \): Virtual page number

- **Components of the physical address (PA)**
  - \( \text{PPO} \): Physical page number
  - \( \text{CT} \): Cache tag

**Core i7 Level 1-3 Page Table Entries**

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>44</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>Unused</td>
<td>Page table physical base address</td>
<td>Unused</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
</tr>
</tbody>
</table>

Each entry references a 4K child page table. Significant fields:

- \( \text{P} \): Child page table present in physical memory (1) or not (0)
- \( \text{R/W} \): Read-only or read-write access permission for all reachable pages
- \( \text{U/S} \): User or supervisor (kernel) mode access permission for all reachable pages
- \( \text{CR} \): Page size either 4 KB or 4 MB (defined for Level 1 PTEs only)
- \( \text{WT} \): Write-through or write-back cache policy for the child page table

**End-to-end Core i7 Address Translation**

**Core i7 Level 4 Page Table Entries**

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>44</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>Unused</td>
<td>Page physical base address</td>
<td>Unused</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
</tr>
</tbody>
</table>

Each entry references a 4K child page. Significant fields:

- \( \text{P} \): Child page is present in memory (1) or not (0)
- \( \text{R/W} \): Read-only or read-write access permission for child page
- \( \text{U/S} \): User or supervisor mode access
- \( \text{CR} \): Page size (KB aligned)
- \( \text{WT} \): Write-through or write-back cache policy for this page

**Cute Trick for Speeding Up L1 Access**

- **Observation**
  - Bits that determine \( \text{CI} \) identical in virtual and physical address
  - Can index into cache while address translation taking place
  - Generally we hit in TLB, so \( \text{PPN} \) bits (CT bits) available next
  - "Virtually indexed, physically tagged"
  - Cache carefully sized to make this possible

---

*Beyond and O’Hallaron, Computer Systems: A Programmer’s Perspective, Third Edition*
Virtual Address Space of a Linux Process

- Different for each process
  - Kernel virtual memory
  - User stack
  - Memory mapped region for shared libraries
  - Runtime heap (malloc)
  - Uninitialized data (.bss)
  - Program text (.text)

- Identical for each process
  - Process virtual memory
  - Stack
  - brk
  - xuser
  - scratch

Virtual Address Space of a Linux Process

Linux Page Fault Handling

- VM areas initialized by associating them with disk objects.
  - Process is known as memory mapping.

- Area can be backed by (i.e., get its initial values from):
  - Regular file on disk (e.g., an executable object file)
    - Initial page bytes come from a section of a file
  - Anonymous file (e.g., nothing)
    - First fault will allocate a physical page full of 0's (demand-zero page)
    - Once the page is written to (dirtied), it is like any other page

- Dirty pages are copied back and forth between memory and a special swap file.

Linux Organizes VM as Collection of “Areas”

- pgd:
  - Page global directory address
  - Points to L1 page table

- vm_prot:
  - Read/write permissions for this area

- vm_flags:
  - Pages shared with other processes or private to this process

Memory Mapping

- Process virtual memory

Memory Mapping

- Process 1 virtual memory
- Physical memory
- Process 2 virtual memory

Sharing Revisited: Shared Objects

- Process 1 maps the shared object.

Today

- Simple memory system example
- Case study: Core i7/Linux memory system
- Memory mapping
Sharing Revisited: Shared Objects

- Process 2 maps the shared object.
- Notice how the virtual addresses can be different.

Sharing Revisited: Private Copy-on-write (COW) Objects

- Instruction writing to private page triggers protection fault.
- Handler creates new R/W page.
- Instruction restarts upon handler return.
- Copying deferred as long as possible!