Virtual Memory Implementation

Brad Karp
UCL Computer Science

CS 0019
26th February 2019

(lecture notes derived from material from Phil Gibbons, Dave O’Hallaron, and Randy Bryant)
A page table contains page table entries (PTEs) that map virtual pages to physical pages.
Translating with a k-level Page Table

- Having multiple levels greatly reduces page table size

Page table base register (part of the process’ context)

```
VIRTUAL ADDRESS

VPN 1 | VPN 2 | ... | VPN k

the Level 1 page table | a Level 2 page table | a Level k page table

m-1  | p-1  | 0

PPN | PPO

PHYSICAL ADDRESS
```
Translation Lookaside Buffer (TLB)

- A small cache of page table entries with fast access by MMU

Typically, a TLB hit eliminates the k memory accesses required to do a page table lookup.
Set Associative Cache: Read

- Locate set
- Check if any line in set has matching tag
- Yes + line valid: hit
- Locate data starting at offset

Address of word:
- t bits
- s bits
- b bits

CT = tag
CI = index
CO = offset

data begins at this offset

\[ E = 2^e \text{ lines per set} \]

\[ S = 2^s \text{ sets} \]

\[ B = 2^b \text{ bytes per cache block (the data)} \]
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
Today

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

- **Addressing**
  - 14-bit virtual addresses
  - 12-bit physical address
  - Page size = 64 bytes
Simple Memory System TLB

- 16 entries
- 4-way associative

Translation Lookaside Buffer (TLB)

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>00</td>
<td>-</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>04</td>
<td>-</td>
<td>0</td>
<td>0A</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>08</td>
<td>-</td>
<td>0</td>
<td>06</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
Simple Memory System TLB

- 16 entries
- 4-way associative

Translation Lookaside Buffer (TLB)

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>–</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>00</td>
<td>–</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>02</td>
<td>–</td>
<td>0</td>
<td>04</td>
<td>–</td>
<td>0</td>
<td>0A</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>–</td>
<td>0</td>
<td>08</td>
<td>–</td>
<td>0</td>
<td>06</td>
<td>–</td>
<td>0</td>
<td>03</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>–</td>
<td>0</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>02</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>
Simple Memory System TLB

- 16 entries
- 4-way associative

![Diagram of TLB entries]

**Translation Lookaside Buffer (TLB)**

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>00</td>
<td>-</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>04</td>
<td>-</td>
<td>0</td>
<td>0A</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>08</td>
<td>-</td>
<td>0</td>
<td>06</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN = 0b1101 = 0x0D
Simple Memory System TLB

- 16 entries
- 4-way associative

Translation Lookaside Buffer (TLB)

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>00</td>
<td>-</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>04</td>
<td>-</td>
<td>0</td>
<td>0A</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>08</td>
<td>-</td>
<td>0</td>
<td>06</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>-</td>
<td>0</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN = 0b1101 = 0x0D
## Simple Memory System Page Table

Only showing the first 16 entries (out of 256)

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>09</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>0A</td>
<td>09</td>
<td>1</td>
</tr>
<tr>
<td>0B</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>0C</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>0D</td>
<td>2D</td>
<td>1</td>
</tr>
<tr>
<td>0E</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>0F</td>
<td>0D</td>
<td>1</td>
</tr>
</tbody>
</table>

0x0D → 0x2D
### Simple Memory System Page Table

Only showing the first 16 entries (out of 256)

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>09</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>0A</td>
<td>09</td>
<td>1</td>
</tr>
<tr>
<td>0B</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>0C</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>0D</td>
<td>2D</td>
<td>1</td>
</tr>
<tr>
<td>0E</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>0F</td>
<td>0D</td>
<td>1</td>
</tr>
</tbody>
</table>

0x0D → 0x2D

![Diagram](image-url)
Simple Memory System Cache

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

V[0b00001101101001] = V[0x369]
P[0b101101101001] = P[0xB69] = 0x15
Simple Memory System Cache

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

V[0b00001101101001] = V[0x369]
P[0b101101101001] = P[0xB69] = 0x15

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1B</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Address Translation Example

Virtual Address: 0x03D4

<table>
<thead>
<tr>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN ___  TLBI ___  TLBT ____  TLB Hit? ___  Page Fault? ___  PPN: ____

Physical Address

CO ___  CI___  CT ____  Hit? ___  Byte: ____
Address Translation Example

Virtual Address: 0x03D4

<table>
<thead>
<tr>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN 0xOF   TLBI 0x3   TLBT 0x03   TLB Hit? ___   Page Fault? ___   PPN: ____________

Physical Address

CO ___   CI ___   CT ___   Hit? ___   Byte: ______
Address Translation Example

Virtual Address: 0x03D4

![Diagram of TLB and VPN]

Physical Address

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>–</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>00</td>
<td>–</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>02</td>
<td>–</td>
<td>0</td>
<td>04</td>
<td>–</td>
<td>0</td>
<td>0A</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>–</td>
<td>0</td>
<td>08</td>
<td>–</td>
<td>0</td>
<td>06</td>
<td>–</td>
<td>0</td>
<td>03</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>–</td>
<td>0</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>02</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>
Address Translation Example

Virtual Address: 0x03D4

VPN: 0xF  TLBI: 0x3  TLBT: 0x03  TLB Hit? Y  Page Fault? N  PPN: 0xD

Physical Address

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>09</td>
<td>0D</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>07</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03</td>
<td>2D</td>
<td>1</td>
<td>1</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>04</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>04</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>02</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>08</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>06</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>06</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>03</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>07</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>03</td>
<td>0D</td>
<td>1</td>
<td>3</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>4</td>
<td>0A</td>
<td>34</td>
<td>1</td>
<td>4</td>
<td>02</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
Address Translation Example

Virtual Address: \(0\times03D4\)

<table>
<thead>
<tr>
<th>(13)</th>
<th>(12)</th>
<th>(11)</th>
<th>(10)</th>
<th>(9)</th>
<th>(8)</th>
<th>(7)</th>
<th>(6)</th>
<th>(5)</th>
<th>(4)</th>
<th>(3)</th>
<th>(2)</th>
<th>(1)</th>
<th>(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN \(0\times0F\) \(\quad\) TLBI \(0\times3\) \(\quad\) TLBT \(0\times03\) \(\quad\) TLB Hit? \(\_\) \(\quad\) Page Fault? \(\_\) \(\quad\) PPN: \(0\times0D\)

Physical Address

CO \(\_\) \(\quad\) CI \(\_\) \(\quad\) CT \(\_\) \(\quad\) Hit? \(\_\) \(\quad\) Byte: \(\_\)
Address Translation Example

Virtual Address: \texttt{0x03D4}

<table>
<thead>
<tr>
<th>TLBT</th>
<th>TLBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\begin{center}
\begin{tabular}{cccccccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{cccccccccccc}
\text{VPN} & \text{VPO} & \text{VPN} & \text{TLBI} & \text{TLBT} & \text{TLB Hit?} & \text{Page Fault?} & \text{PPN: } \texttt{0x0D} \\
\end{tabular}
\end{center}

VPN \texttt{0xF} \quad TLBI \texttt{0x3} \quad TLBT \texttt{0x03} \quad TLB Hit? \texttt{Y} \quad Page Fault? \texttt{N} \quad PPN: \texttt{0x0D}

Physical Address

<table>
<thead>
<tr>
<th>CT</th>
<th>Cl</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\begin{center}
\begin{tabular}{cccccccccccc}
0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{cccccccccccc}
\text{PPN} & \text{PPO} & \text{CO} & \text{Cl} & \text{CT} & \text{Hit?} & \text{Byte:} \\
\end{tabular}
\end{center}

CO ___ \quad Cl___ \quad CT ____ \quad Hit? ___ \quad Byte: _____
Address Translation Example

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Physical Address

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CT  CO  CI  PPO  PPN

11 10  9  8  7  6  5  4  3  2  1  0

0 0 1 1 0 1 0 1 0 1 0 0

CO ___  CI ___  CT ___  Hit? ___  Byte: _____
Address Translation Example

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Physical Address

<table>
<thead>
<tr>
<th>CT</th>
<th>Cl</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PPN</th>
<th>PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

CO __0__  Cl __0x5__  CT __0x0D__  Hit? __  Byte: ____
Address Translation Example

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>1B</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Physical Address

CT 11 10 9 8 7 6 5 4 3 2 1 0
Cl
CO

PPN

PPO

CO 0  Cl 0x5  CT 0x0D  Hit? Y  Byte: ___
# Address Translation Example

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>2</td>
<td>1B</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Idx</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
</tbody>
</table>

## Physical Address

```
<table>
<thead>
<tr>
<th>CO</th>
<th>Cl</th>
<th>CT</th>
<th>Hit?</th>
<th>Byte:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x5</td>
<td>0x0D</td>
<td>Y</td>
<td>0x36</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>PPN</th>
<th>PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Address Translation Example: **TLB/Cache Miss**

Virtual Address: 0x0020

![Address Translation Diagram]

Physical Address

![Physical Address Diagram]
Address Translation Example: TLB/Cache Miss

Virtual Address: 0x0020

<table>
<thead>
<tr>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VPN 0x00  TLBI 0  TLBT 0x00  TLB Hit? N  Page Fault?  PPN:  

Physical Address

<table>
<thead>
<tr>
<th>CO</th>
<th>CI</th>
<th>CT</th>
<th>Hit?</th>
<th>Byte:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Address Translation Example: TLB/Cache Miss

Virtual Address: 0x0020

Physical Address

<table>
<thead>
<tr>
<th>Page table</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPN</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>
Address Translation Example: **TLB/Cache Miss**

**Virtual Address:** 0x0020

<table>
<thead>
<tr>
<th>TLBT</th>
<th>TLBI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

**Physical Address**

<table>
<thead>
<tr>
<th>CO</th>
<th>CI</th>
<th>CT</th>
<th>Hit?</th>
<th>Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Page table**

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
Address Translation Example: **TLB/Cache Miss**

Virtual Address: \(0x0020\)

Virtual Page Number (VPN): __
Translation Lookaside Buffer (TLBI): __
Translation Lookaside Buffer (TLBT): __
TLB Hit?: __
Page Fault?: __
Page Frame Number (PPN): __

Physical Address

Logical Page Number (VPN): __
Translation Lookaside Buffer (TLBI): __
Translation Lookaside Buffer (TLBT): __
TLB Hit?: __
Page Fault?: __
Page Frame Number (PPN): __

Page table

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td>–</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>–</td>
<td>0</td>
</tr>
</tbody>
</table>
Address Translation Example: TLB/Cache Miss

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>1</td>
<td>99</td>
<td>11</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>1</td>
<td>00</td>
<td>02</td>
<td>04</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>1</td>
<td>43</td>
<td>6D</td>
<td>8F</td>
<td>09</td>
</tr>
<tr>
<td>5</td>
<td>0D</td>
<td>1</td>
<td>36</td>
<td>72</td>
<td>F0</td>
<td>1D</td>
</tr>
<tr>
<td>6</td>
<td>31</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1</td>
<td>11</td>
<td>C2</td>
<td>DF</td>
<td>03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>Tag</th>
<th>Valid</th>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
<td>1</td>
<td>3A</td>
<td>00</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>9</td>
<td>2D</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2D</td>
<td>1</td>
<td>93</td>
<td>15</td>
<td>DA</td>
<td>3B</td>
</tr>
<tr>
<td>B</td>
<td>0B</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>1</td>
<td>04</td>
<td>96</td>
<td>34</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
<td>1</td>
<td>83</td>
<td>77</td>
<td>1B</td>
<td>D3</td>
</tr>
<tr>
<td>F</td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Physical Address

CT 1 11 9 8 7 6 5 4 3 2 1 0
CI 1 0 1 0 0 0 1 0 0 0 0 0
PPN 1 0 1 0 0 0 1 0 0 0 0 0

Page table

<table>
<thead>
<tr>
<th>VPN</th>
<th>PPN</th>
<th>Valid</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>02</td>
<td>33</td>
<td>1</td>
</tr>
<tr>
<td>03</td>
<td>02</td>
<td>1</td>
</tr>
<tr>
<td>04</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>05</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>06</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Address Translation Example: **TLB/Cache Miss**

**Physical Address**

![Physical Address Diagram]

- **CT**
- **CI**
- **CO**
- **VPN**
- **PPN**
- **PPO**

- **CO** 0
- **CI** 0x8
- **CT** 0x28
- **Hit?** N
- **Byte:** Mem

<table>
<thead>
<tr>
<th>Page table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VPN</strong></td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
</tbody>
</table>
Today

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

Processor package

Core x4

- Registers
- L1 d-cache: 32 KB, 8-way
- L1 i-cache: 32 KB, 8-way
- L2 unified cache: 256 KB, 8-way
- L3 unified cache: 8 MB, 16-way (shared by all cores)
- Instruction fetch
- MMU (addr translation)
- L1 d-TLB: 64 entries, 4-way
- L1 i-TLB: 128 entries, 4-way
- L2 unified TLB: 512 entries, 4-way
- QuickPath interconnect: 4 links @ 25.6 GB/s each
- DDR3 Memory controller: 3 x 64 bit @ 10.66 GB/s, 32 GB/s total (shared by all cores)
- Main memory

To other cores
To I/O bridge
End-to-end Core i7 Address Translation

CPU

Virtual address (VA)

VPN
VPO

32/64

Result

L2, L3, and main memory

L1 hit

L1 miss

L1 d-cache
(64 sets, 8 lines/set)

L1 TLB (16 sets, 4 entries/set)

TLB hit

TLB miss

VPN1 VPN2 VPN3 VPN4

9 9 9 9

CR3

PTE

Page tables

VPN
VPO

32 4

TLBT TLBI

L1, L3, and main memory

Physical address (PA)

PPN PPO

VPN3 VPN4

9 9

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2

CR3

PTE

Page tables

VPN1 VPN2

VPN1 VPN2
Each entry references a 4K child page table. Significant fields:

- **P**: Child page table present in physical memory (1) or not (0).
- **R/W**: Read-only or read-write access access permission for all reachable pages.
- **SUP**: user or supervisor (kernel) mode access permission for all reachable pages.
- **WT**: Write-through or write-back cache policy for the child page table.
- **A**: Reference bit (set by MMU on reads and writes, cleared by software).
- **PS**: Page size either 4 KB or 4 MB (defined for Level 1 PTEs only).

Page table physical base address: 40 most significant bits of physical page table address (forces page tables to be 4KB aligned)

XD: Disable or enable instruction fetches from all pages reachable from this PTE.
Core i7 Level 4 Page Table Entries

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>52</th>
<th>51</th>
<th>12</th>
<th>11</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD</td>
<td>Unused</td>
<td>Page physical base address</td>
<td>Unused</td>
<td>G</td>
<td>D</td>
<td>A</td>
<td>CD</td>
<td>WT</td>
<td>SUP</td>
<td>R/W</td>
<td>P=1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Available for OS (page location on disk)  
P=0

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

- **P**: Child page is present in memory (1) or not (0)
- **R/W**: Read-only or read-write access permission for child page
- **SUP**: User or supervisor mode access
- **WT**: Write-through or write-back cache policy for this page
- **A**: Reference bit (set by MMU on reads and writes, cleared by software)
- **D**: Dirty bit (set by MMU on writes, cleared by software)

Page physical base address: 40 most significant bits of physical page address (forces pages to be 4KB aligned)

**XD**: Disable or enable instruction fetches from this page.
Core i7 Page Table Translation

- **CR3**: Physical address of L1 PT
- **L1 PT**: Page global directory
- **L2 PT**: Page upper directory
- **L3 PT**: Page middle directory
- **L4 PT**: Page table
- **VPN**: Virtual page number
- **VPN 1**: 512 GB region per entry
- **VPN 2**: 1 GB region per entry
- **VPN 3**: 2 MB region per entry
- **VPN 4**: 4 KB region per entry
- **VPN 4**: Virtual address
- **VPO**: Offset into physical and virtual page
- **PPN**: Physical address of page
- **PPO**: Physical address
Cute Trick for Speeding Up L1 Access

**Observation**

- Bits that determine CI identical in virtual and physical address
- Can index into cache while address translation taking place
- Generally we hit in TLB, so PPN bits (CT bits) available next
- "Virtually indexed, physically tagged"
- Cache carefully sized to make this possible
Virtual Address Space of a Linux Process

- **Process-specific data structs** (ptables, task and mm structs, kernel stack)
- **Physical memory**
- **Kernel code and data**
- **User stack**
- **Memory mapped region for shared libraries**
- **Runtime heap (malloc)**
- **Uninitialized data (.bss)**
- **Initialized data (.data)**
- **Program text (.text)**

**Different for each process**

**Identical for each process**

**Kernel virtual memory**

**Process virtual memory**
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

- Each process has own `task_struct`, etc
Linux Page Fault Handling

vm_area_struct
- vm_end
- vm_start
- vm_prot
- vm_flags

Process virtual memory
- shared libraries
- data
- text

- vm_end
- vm_start
- vm_prot
- vm_flags
- vm_next
Linux Page Fault Handling

Process virtual memory

```
vm_area_struct
+-- vm_end
+-- vm_start
+-- vm_prot
+-- vm_flags
```

```
shared libraries
```

```
data
```

```
text
```

read

vm_next
Linux Page Fault Handling

Process virtual memory

shared libraries

data

text

Segmentation fault: accessing a non-existing page
Linux Page Fault Handling

Process virtual memory

- shared libraries
- data
- text

vm_area_struct

- vm_end
- vm_start
- vm_prot
- vm_flags

Segmentation fault:
accessing a non-existing page

1. read
2. write
Linux Page Fault Handling

Process virtual memory

1. Segmentation fault: accessing a non-existing page

2. Protection exception: e.g., violating permission by writing to a read-only page (Linux reports as Segmentation fault)
Linux Page Fault Handling

Process virtual memory

1. read

Segmentation fault: accessing a non-existing page

2. write

Protection exception: e.g., violating permission by writing to a read-only page (Linux reports as Segmentation fault)

3. read

VM area structure:
- vm_end
- vm_start
- vm_prot
- vm_flags
- vm_next

Shared libraries

Data

Text

VM flags
Linux Page Fault Handling

Process virtual memory

- shared libraries
- data
- text

vm_area_struct
  - vm_end
  - vm_start
  - vm_prot
  - vm_flags

vm_next

1. read
   Segmentation fault: accessing a non-existing page

2. write
   Protection exception: e.g., violating permission by writing to a read-only page (Linux reports as Segmentation fault)

3. read
   Normal page fault
Today

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

- VM areas initialized by associating them with disk objects.
  - Called *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*. 
Review: Memory Management & Protection

- Code and data can be isolated or shared among processes

Virtual Address Space for Process 1:

Virtual Address Space for Process 2:

Address translation

Physical Address Space (DRAM)

(e.g., read-only library code)
Sharing Revisited: Shared Objects

- Process 1 maps the shared object (on disk).
Sharing Revisited: Shared Objects

- Process 2 maps the same shared object.
- Notice how the virtual addresses can be different.
- But, difference must be multiple of page size.
Sharing Revisited: Private Copy-on-write (COW) Objects

- Two processes mapping a private copy-on-write (COW) object
- Area flagged as private copy-on-write
- PTEs in private areas are flagged as read-only
Sharing Revisited: Private Copy-on-write (COW) Objects

Process 1 virtual memory

Physical memory

Process 2 virtual memory

Private copy-on-write object

Write to private copy-on-write page
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!

![Diagram showing memory connections and copying deferred]
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!

Diagram:
- Process 1 virtual memory
- Physical memory
- Process 2 virtual memory
- Private copy-on-write object

Copy-on-write

Write to private copy-on-write page
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!
User-Level Memory Mapping

```c
void *mmap(void *start, int len,
            int prot, int flags, int fd, int offset)
```

- Map `len` bytes starting at offset `offset` of the file specified by file description `fd`, preferably at address `start`
  - `start`: may be 0 for “pick an address”
  - `prot`: PROT_READ, PROT_WRITE, PROT_EXEC, ...
  - `flags`: MAP_ANON, MAP_PRIVATE, MAP_SHARED, ...

- Return a pointer to start of mapped area (may not be `start`)
User-Level Memory Mapping

```c
void *mmap(void *start, int len,
            int prot, int flags, int fd, int offset)
```

- `len` bytes
- `offset` (bytes)
- Disk file specified by file descriptor `fd`
- Process virtual memory
- `start` (or address chosen by kernel)
Example: Using `mmap` to Copy Files

- Copying a file to `stdout` without transferring data to user space

```c
#include "csapp.h"

void mmapcopy(int fd, int size)
{
    /* Ptr to memory mapped area */
    char *bufp;

    bufp = mmap(NULL, size,
                PROT_READ,
                MAP_PRIVATE,
                fd, 0);
    write(STDOUT_FILENO, 
       bufp, size);
    return;
}

/* mmapcopy driver */
int main(int argc, char **argv)
{
    struct stat stat;
    int fd;

    /* Check for required cmd line arg */
    if (argc != 2) {
        printf("usage: %s <filename>
            \n", argv[0]);
        exit(0);
    }

    /* Copy input file to stdout */
    fd = Open(argv[1], O_RDONLY, 0);
    fstat(fd, &stat);
    mmapcopy(fd, stat.st_size);
    exit(0);
}
```
Some Uses of mmap

- **Reading big files**
  - Uses paging mechanism to bring files into memory

- **Shared data structures**
  - When call with `MAP_SHARED` flag
    - Multiple processes have access to same region of memory
    - Risky!

- **File-based data structures**
  - E.g., database
  - Give `prot` argument `PROT_READ | PROT_WRITE`
  - When unmap region, file will be updated via write-back
  - Can implement load from file / update / write back to file
Summary

- VM requires hardware support
  - Exception handling mechanism
  - TLB
  - Various control registers

- VM requires OS support
  - Managing page tables
  - Implementing page replacement policies
  - Managing file system

- VM enables many capabilities
  - Loading programs from memory
  - Forking processes
  - Providing memory protection