ECE 2300
Digital Logic & Computer Organization
Spring 2017

Virtual Memory
Announcements

• Lab 6 prelab due tomorrow

• Final Exam: Monday May 15, 9am, Hollister B14
  – Covers the entire course
  – Inform instructor about schedule conflict ASAP
Example: Impact of L1 Caches

• **With L1 caches**
  – L1 instruction cache miss rate = 2%
  – L1 data cache miss rate = 5%
  – Miss penalty = 200 cycles
  – 20% of all instructions are loads/stores

• $\text{CPI}_{\text{memhier}} = 0.02 \times 200 + 0.2 \times 0.05 \times 200 = 6$
Extending and Sharing Main Memory?

• What if one program needs more than the amount of installed main memory (i.e., physical memory)?

• How do multiple programs share the same main memory address space?
Main memory (MM) is managed similar to a cache

- Data are brought into MM as requested
- If MM is full, older data get swapped out to disk
Sharing Main Memory

- **Multiple programs reside in MM at once** (multiprogramming / multitasking)
  - Transparency: a program should not be aware of the fact that other programs are sharing the same MM
  - Protection: a program must not be able to corrupt other programs

- **How do multiple programs share the same MM address space?**
  - Each program operates in its own virtual address space
  - The set of physical MM addresses for each program is dynamically allocated and managed
Virtual Memory

• The *hardware and software* mechanisms that dynamically manage the memory hierarchy

• Extend memory hierarchy to incorporate large permanent storage
  – Hide physical size of MM from the program
  – Moves large blocks (in unit of pages) between MM and permanent storage as needed

• Allow multiple programs to share the main memory and provides protection among programs
Physical and Virtual Addresses

- **Physical addresses** refer to the real addresses used by hardware to access the physical MM
  - With a $M$-bit physical address, the size of the **physical address space** is $2^M$ bytes

- **Virtual addresses** refer to the addresses used by the programs
  - When a program is compiled, the instructions and data addresses are **virtual**
    - Do not correspond to where they will be placed in MM
  - With a $N$-bit virtual address, the size of the **virtual address space** is $2^N$ bytes (typically, $N > M$)
Paging

- Virtual/physical address space is divided into equal sized pages
  - A page contains N bytes where N is a power of 2
    - N = 4096 in this lecture
  - A whole page is read or written during data transfer between MM and disk
  - Each page in virtual memory space has a unique index called virtual page number (VPN)
  - Similarly, each page in physical memory space has a unique physical page number (PPN)
Paging

• When requested, a page is brought into a physical MM location

• The correspondence (mapping) between virtual to physical addresses is saved
  – When the same virtual address is encountered, it is translated using this saved information
Virtual and Physical Addresses

Virtual address

31 30 29 28 27 ........................ 15 14 13 12 11 10 9 8  ............ 3 2 1 0

Virtual page number

Page offset

Translation

Physical address

29 28 27 ........................ 15 14 13 12 11 10 9 8  ............ 3 2 1 0

Physical page number

Page offset

analogous to the byte offset for a cache

program address

MM address
Address Translation Using a Page Table

Virtual page number (VPN): analogous to the index for a cache

Valid
Physical page number

If 0 then page is not present in memory

Page table entries (PTEs)

Page table

Indicates that a page contains $2^{12} = 4$ KB

Virtual address

31 30 29 28 27 15 14 13 12 11 10 9 8 3 2 1 0

Virtual page number
Page offset

Physical page number
Page offset

Physical address

Stored in MM

Lecture 23: 12
Page Table Operation

The page table register (PTR), a special register in CPU, holds the base address of the page table in MM.
Page Table Operation

PTR+VPN form the (physical) address of the PTE in MM

Virtual address

31 30 29 28 27 15 14 13 12 11 10 9 8 3 2 1 0

Virtual page number  Page offset

Valid  

Physical page number

Page table

Lecture 23: 14
Page Table Operation

If Valid = 1, then the PPN and Page Offset are concatenated to form the physical address

Assuming 1GB physical memory here
Page Table Operation

If Valid = 0, a miss (page fault) has occurred, and the page is read from disk into MM (replacing another page if the MM is full).
Example: Page Table Access

- Given the following virtual address stream (in decimal), identify the potential page faults
  - 128, 2048, 4096, 8192

<table>
<thead>
<tr>
<th>Valid</th>
<th>Physical Page#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example: Page Table Access

• Given the following virtual address stream (in decimal), identify the potential page faults
  – 128 (VPN=0), 2048 (VPN=0), 4096 (VPN=1), 8192 (VPN=2)

<table>
<thead>
<tr>
<th>Valid</th>
<th>Physical Page#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Page Faults and Page Replacement

• Miss penalty on a page fault is significant
  – Up to ~100M cycles

• Low miss (page fault) rates are essential
  – Fully associative page placement (put anywhere in MM)
  – LRU replacement of a page when MM is full

• The Operating System (OS) handles page placement
Page Faults and Page Replacement

• Too expensive to do true LRU (100K-1M pages)

• LRU approximation
  – Each PTE has a $Reference$ bit (ref)
  – Reference bit is set when a page is accessed
  – OS periodically clears all Reference bits
  – OS chooses a page with a Reference bit of 0
What About Writes to MM?

• Also write to disk (write through)?

• Disk writes can take millions of cycles

• Write back policy is used
  – *Dirty* bit in PTE is set on a write to MM
  – Page with set *Dirty* bit is written to disk if replaced
Faster Address Translation

- Have to access the page table before an instruction can be fetched and before data memory can be accessed.

- Page table accesses have good locality.

- ⇒ Cache the most recent PTEs within the CPU.
Translation Lookaside Buffer (TLB)

- Small cache of recently accessed PTE (typically 16-512 entries, fully associative)
Before Next Class

• H&H 6.7.2, 7.7

Next Time

Exceptions
Inputs/Outputs