Virtual Memory

Motivation

1. Give applications the illusion that there’s more memory than there really is
2. Give every application the illusion that it has memory to itself
3. Protect the system and applications from each other

Organization of Memory as a Cache

- Line size?
  - large
- Associativity?
  - high
- Write-through or write-back?
  - write-back

Locating an Object in a “Cache”

- SRAM Cache
  - Tag stored with cache line
  - Maps from cache block to memory blocks
    - From cached to uncached form
    - Save a few bits by only storing tag
  - No tag for block not in cache
  - Hardware retrieves information
    - can quickly match against multiple tags

```
<table>
<thead>
<tr>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D 243</td>
</tr>
<tr>
<td>1</td>
<td>X 17</td>
</tr>
<tr>
<td></td>
<td>X 1</td>
</tr>
<tr>
<td>N-1</td>
<td>J 105</td>
</tr>
</tbody>
</table>
```

Object Name

\[ = X? \]
Locating an Object in “Cache” (cont.)

- DRAM Cache
  - Each allocated page of virtual memory has entry in page table
  - Mapping from virtual pages to physical pages
    - From uncached form to cached form
    - Page table entry even if page not in memory
    - Specifies disk address
    - Only way to indicate where to find page
    - OS retrieves information

A System with Physical Memory Only

- Examples:
  - most Cray machines, early PCs, nearly all embedded systems, etc.

  Addresses generated by the CPU correspond directly to bytes in physical memory

A System with Virtual Memory

- Examples:
  - workstations, servers, modern PCs, etc.

  Address Translation: Hardware converts virtual addresses to physical addresses via OS-managed lookup table (page table)

Servicing a Page Fault

- Processor Signals Controller
  - Read block of length P starting at disk address X and store starting at memory address Y
- Read Occurs
  - Direct Memory Access (DMA)
  - Under control of I/O controller
- I/O Controller Signals Completion
  - Interrupt processor
  - OS resumes suspended process
Motivation #2: Memory Management

- Multiple processes can reside in physical memory.
- How do we resolve address conflicts?
  - what if two processes access something at the same address?

- Memory invisible to user code
- The "brk" ptr
- Linux/x86 process memory image

Solution: Separate Virt. Addr. Spaces

- Virtual and physical address spaces divided into equal-sized blocks
  - blocks are called "pages" (both virtual and physical)
- Each process has its own virtual address space
  - operating system controls how virtual pages are assigned to physical memory

Motivation #3: Protection

- Page table entry contains access rights information
  - hardware enforces this protection (trap into OS if violation occurs)

Tools/Techniques

- Per-process page tables
- Page table base register
- Translation look-aside buffer (TLB)
  - Cache for page table entries
  - Organization parameters: size, associativity
Integrating VM and Cache

- Most Caches “Physically Addressed”
  - Accessed by physical addresses
  - Allows multiple processes to have blocks in cache at same time
  - Allows multiple processes to share pages
  - Cache doesn’t need to be concerned with protection issues
  - Access rights checked as part of address translation
- Perform Address Translation Before Cache Lookup
  - But this could involve a memory access itself (of the PTE)
  - Of course, page table entries can also become cached

Speeding up Translation with a TLB

- “Translation Lookaside Buffer” (TLB)
  - Small hardware cache in MMU
  - Maps virtual page numbers to physical page numbers
  - Contains complete page table entries for small number of pages

Address Translation with a TLB

Simple Memory System TLB

- TLB
  - 16 entries
  - 4-way associative

<table>
<thead>
<tr>
<th>Set</th>
<th>Tag</th>
<th>PPN</th>
<th>Vali</th>
<th>Tag</th>
<th>PPN</th>
<th>Vali</th>
<th>Tag</th>
<th>PPN</th>
<th>Vali</th>
<th>Tag</th>
<th>PPN</th>
<th>Vali</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03</td>
<td>-</td>
<td>8</td>
<td>09</td>
<td>0D</td>
<td>$</td>
<td>00</td>
<td>-</td>
<td>8</td>
<td>07</td>
<td>02</td>
<td>9</td>
</tr>
</tbody>
</table>
Simple Memory System Cache

- Cache
  - 16 lines
  - 4-byte line size
  - Direct mapped

Main Themes

- Programmer’s View
  - Large “flat” address space
    - Can allocate large blocks of contiguous addresses
  - Processor “owns” machine
    - Has private address space
    - Unaffected by behavior of other processes

- System View
  - User virtual address space created by mapping to set of pages
    - Need not be contiguous
    - Allocated dynamically
    - Enforce protection during address translation
  - OS manages many processes simultaneously
    - Continually switching among processes
    - Especially when one must wait for resource
      - E.g., disk I/O to handle page fault