VIRTUAL MEMORY

Reading:
Chapter 6, except cache implementation details (6.4.1-6.4.6) and segmentation (6.5.5)
Wikipedia – Virtual Machine

Objectives

- Understand the concepts behind
  - Virtual memory,
  - Memory segmentation,
  - Paging
  - Address translation
  - Virtualization
Programmer Managed Memory

- Only executing programs and recently accessed data need be in memory
- Non-active programs and dormant data can be in secondary memory, waiting to be called into main memory
- Programs can manage memory
  - Older systems
  - Some game programming systems
  - Some real-time systems

Remember the memory hierarchy

What is a Virtual Address?

- A memory address that does not correspond directly to an address in physical memory
- The virtual address needs to be mapped (translated) into a physical address

© Robert F. Kelly, 2010-2016
Virtual Memory

- Provides greater memory capacity
- Allows the OS/HW to manage migration of data
- Entire address space required by a process need not be in memory at once
- Main memory page frames or pages are written (paged) to disk when they are not immediately needed

Virtual Memory Terms

- **Virtual address** – Address that the process uses. It is mapped to a physical address by the memory manager
- **Physical address** - actual memory address in physical memory
- **Page frames** – the equal-size blocks into which physical memory is divided
- **Pages** – Equal-size blocks into which the address space (virtual memory) is divided. Pages are the same size as page frames
- **Page fault** - occurs when a logical address requires that a page be brought in from disk (similar to a cache miss)
- **Memory fragmentation** - occurs when the paging process results in the creation of small, unusable clusters of memory addresses
Virtual Memory

- Main memory and virtual memory are divided into equal sized pages
  - Page frame – physical memory
  - Page – virtual memory
- Unnecessary pages are in slower secondary storage
- Can be implemented with paging or segmentation (paging most common)

Similar to a block in caching systems

Virtual Memory

- Pages allocated to a process do not need to be stored contiguously-- either on disk or in memory
- Makes it easier for the Operating System to distribute memory to many processes
Page Table

- Information concerning the location of each page is maintained in a data structure called a **page table**
- There is one page table for each active process.

Note this prevents a process from overwriting memory of other processes.

Valid bit is set to 1 if the page is in memory

Memory Access …

- The virtual address is divided into two fields: A **page** field, and an **offset** field
- The page field determines the page location of the address, and the offset indicates the location of the address within the page
- The logical page number is translated into a physical page frame through a lookup in the page table
- **000 0001111000** is translated into **10 0001111000**

Translation is done immediately since the valid bit of page 000 is 1
... Memory Access

- If the valid bit is zero in the page table entry for the virtual address, the page is not in memory and must be fetched from disk
  - This is a page fault
  - If necessary, a page is evicted from memory and is replaced by the page retrieved from disk, and the valid bit is set to 1
  - Virtual address to physical address mapping is performed
  - Physical address is used to satisfy the memory request

Page can be overwritten (not evicted) if it has not been written

Example ...

- A byte-addressable system has an 8K virtual address space ($2^{13}$), a 4K ($2^{12}$) physical address space, and a 1024 page size ($2^{10}$)
  - We have $2^{13}/2^{10} = 2^3$ virtual pages.
  - A virtual address has 13 bits with 3 bits for the page field and 10 for the offset
  - A physical memory address requires 12 bits, the first two bits for the page frame and the trailing 10 bits the offset.
... Example ...

• What happens when CPU generates address $10101010011_2$ = $1553_{16}$?

<table>
<thead>
<tr>
<th>Page #</th>
<th>Frame #</th>
<th>Valid Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

Note the 13-bit addresses

The high-order 3 bits of the virtual address, 101 ($5_{10}$), provide the page number in the page table

... Example

• The address $1 0101 0101 0011_2$ is converted to physical address $0101 0101 0011_2$ because the page field 101 is replaced by frame number 01 through a lookup in the page table.

<table>
<thead>
<tr>
<th>Page #</th>
<th>Frame #</th>
<th>Valid Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>
Virtual Memory

- Effective access time (EAT) takes all levels of memory into consideration.
- Virtual memory is also a factor in the calculation, and we also have to consider page table access time.
- Suppose a main memory access takes 200ns, the page fault rate is 1%, and it takes 10ms to load a page from disk. We have:
  \[\text{EAT} = 0.99(200\text{ns} + 200\text{ns}) + 0.01(10\text{ms}) = 100.396\text{ns}\]

One reference to the page table and one reference to memory

Notice the importance of page hit rate

1ms = 1,000,000ns

TLB

- Because page tables are read constantly, it makes sense to keep them in a special cache called a translation look-aside buffer (TLB).
- TLBs are a special associative cache that stores the mapping of virtual pages to physical pages.
Virtual Memory Advantages ...

- Translation:
  - Program can be given consistent view of memory, even though physical memory is scrambled
  - Makes multithreading 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.

... Virtual Memory Advantages

- Protection:
  - Different threads (or processes) protected from each other.
  - Different pages can be given special behavior
    - (Read Only, Invisible to user programs, etc.).
  - Kernel data protected from User programs
  - Very important for protection from malicious programs

- Sharing:
  - Can map same physical page to multiple users ("Shared memory")
Pentium

- The Pentium architecture supports both paging and segmentation, and they can be used in various combinations including unpaged/unsegmented, segmented/unpaged, and unsegmented/paged.
- The processor supports two levels of cache (L1 and L2), both having a block size of 32 bytes.
- The L1 cache is next to the processor, and the L2 cache sits between the processor and memory.
- The L1 cache is in two parts: an instruction cache (I-cache) and a data cache (D-cache).
Virtual Machines

• Software implementation of a computer that executes programs like the corresponding physical computer

• Types
  • System Virtual Machine – e.g., IBM VM
  • Process Virtual Machine – e.g., Java Virtual Machine

Virtualized OS, ISAs, I/O, etc. by mapping the virtual request to a physical request

Enables multiple servers to be combined onto one or more physical server

Possibly more on virtualization later in the semester

Have You Met The Objectives?

• Understand the concepts behind
  • Virtual memory,
  • Memory segmentation,
  • Paging
  • Address translation