CPE300: Digital System Architecture and Design

Fall 2011
MW 17:30-18:45 CBC C316

Virtual Memory
11282011

http://www.egr.unlv.edu/~b1morris/cpe300/
Outline

- Review Cache
- Virtual Memory
- Projects
Memory Hierarchy, Cost, Performance

1. Registers – internal to CPU
2. Cache levels
3. Main memory

<table>
<thead>
<tr>
<th>Component</th>
<th>CPU</th>
<th>1-3 Cache memories</th>
<th>Main memory</th>
<th>Disk memory</th>
<th>Tape memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Access type</strong></td>
<td>Random access</td>
<td>Random access</td>
<td>Random access</td>
<td>Direct access</td>
<td>Sequential access</td>
</tr>
<tr>
<td><strong>Capacity, bytes</strong></td>
<td>64–1024</td>
<td>8 KB–4 MB</td>
<td>64 MB–2 GB</td>
<td>10–200 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>.4–10 ns</td>
<td>0.4–20 ns</td>
<td>10–50 ns</td>
<td>10 ms</td>
<td>10 ms–10 s</td>
</tr>
<tr>
<td><strong>Block size</strong></td>
<td>1 word</td>
<td>16 words</td>
<td>16 words</td>
<td>4 KB</td>
<td>4 KB</td>
</tr>
<tr>
<td><strong>Bandwidth</strong></td>
<td>System clock rate</td>
<td>system clock rate - 80 MB/s</td>
<td>10–4000 MB/s</td>
<td>50 MB/s</td>
<td>1 MB/s</td>
</tr>
<tr>
<td><strong>Cost/MB</strong></td>
<td>High</td>
<td>$10</td>
<td>$0.25</td>
<td>$0.002</td>
<td>$0.01</td>
</tr>
</tbody>
</table>
Memory Hierarchy

• Combine smaller, faster memory with slower, larger memory
  ▫ Primary and secondary levels (e.g. cache and main memory)
• Move data efficiently from slow to fast memory using principle of locality
  ▫ Programs tend to reference a confined area of memory repeatedly
  ▫ Spatial locality – if a given memory location is referenced, addresses near it will likely be referenced soon
  ▫ Temporal locality – if a given memory location is referenced, it is likely to be referenced again soon
  ▫ Working set – set of memory locations referenced over a fixed time window
Hits and Misses

• **Hit** – word is found at level requested
  ▫ Hit ratio (hit rate) - \( h = \frac{\# \text{ hits}}{\text{total } \# \text{ references}} \)

• **Miss** – word not found at level requested
  ▫ Must request for containing block in the next higher level in memory hierarchy
  ▫ Miss ratio = \( 1 - h \)

• **Access time**
  ▫ \( t_a = h t_p + (1 - h) t_s \)
  ▫ \( t_p \) - primary memory access time
  ▫ \( t_s \) - secondary memory access time
Cache

- Insertion of high speed memory between CPU and main memory
  - May have more than one cache level
- Caching is usually transparent to programmer
- Caching operations must be handled in hardware
- Cache blocks are item of commerce
  - Block sizes in range of 16-256 bytes
Cache Mapping Function

- Responsible for all cache operations
  - Placement strategy – where to place an incoming block in cache
  - Replacement strategy – which block to replace upon miss
  - Read/write policy – how to handle reads and writes upon cache hits and misses
- Three common mapping functions
  - Associative
  - Direct-mapped
  - Block-set-associative – combination of associative and direct-mapped
Associative Mapped Cache

- Any block from main memory can be put anywhere in cache

- Example: 16-bit address

- Cache structure:
  - One set of 256 lines – 256 block capacity
    - \(2^8 = 256\) 8-byte blocks
  - 3-bits for byte sized word
    - Main memory has \(2^{13} = 8192\) 8-byte blocks
  - \(256 \times 13\)-bit tag memory
    - Indicates block number in cache position
  - \(256 \times 1\)-bit valid memory
    - Indicates if cache location has a value
Properties of Associative Cache

• Advantage
  ▫ Most flexible mapping because a main memory block can go anywhere in the cache

• Disadvantage
  ▫ Large tag memory required
  ▫ Must search entire tag memory simultaneously → lots of hardware required
  ▫ Replacement policy when cache is full causes issue
Direct Mapped Cache

- Divide main memory into sets
  - All blocks in a set (group) can go into only one cache location
- Example: 16-bit main memory address
  - 256 x 8-byte cache
  - The number of cache lines determines the number of sets
  - Cache only examines single group
Properties of Direct Mapped Cache

- **Advantage**
  - Requires less hardware than associative
  - Simple (trivial) replacement policy

- **Disadvantage**
  - Simple replacement policy
    - Restrictive – poor use of cache space
    - Thrashing – two blocks from the same group that are frequently referenced will compete for the same cache location
      - Cause frequent switching of cache data and performance degradation
Block-Set-Associative Cache

- Compromise between associative and direct-mapped to allow several cache blocks for each memory group
- Example: 2-way set associative cache
  - A set of 2 cache values per group
    - 256 x 2 x 8-byte cache
    - 256 sets of 2 lines each
  - Operation is same as direct-mapped
    - Must do associative comparison between tag and cache memory
    - Copy of direct mapped hardware for each set
Cache Read/Write Policies

• Hit policies
  ▫ Write-through – updates both cache and main memory upon each write
  ▫ Write-back – updates only cache
    • Update main memory only upon removal of block
    • Dirty bit is set upon first write to indicate block must be written back

• Miss Policies
  ▫ Read miss – bring block in from main memory
    • Forward word as brought into cache
    • Wait until entire line is filled then repeat cache request
  ▫ Write miss
    • Write allocate – bring block into cache, then update
    • Write-no allocate – write word to main memory without bringing block into cache
Block Replacement Strategies

• Not needed with direct-mapped cache
• Least recently used (LRU)
  ▫ Track cache usage with counter
  ▫ Each block access causes
    • Clear counter of accessed block
    • Increment counters with values less than block being accessed
    • All others remain unchanged
  ▫ When set is full, remove line with highest count
• Random replacement – replace block at random
  ▫ Actually effective strategy in practice
Virtual Memory Hierarchy

- Memory hierarchy usually of main memory and disk
- Enormous speed difference between main memory and disk
  - Order of $10^6$ factor
  - Processor should not be kept waiting for transfer into memory upon miss
- Multiprogramming shares the processor among independent programs stored in memory
  - On miss switch to another program
- Miss response can be assisted by processor
  - I/O, placement/replacement decisions, computations of disk addresses
Virtual Memory

- Technique to use secondary storage (disks) to extend the apparent size of physical memory
  - Each process views memory as if it were its own
  - Not restricted to physical size of memory
  - Logical address space now usually larger than physical memory
- Memory management unit (MMU)
  - Responsible for mapping logical addresses issued by CPU into physical addresses that are presented to cache and main memory
    - Mapping tables are used
  - OS assists with selecting data appropriately for working set
Paging and Block Placement

- Page – commonly used name for a disk block
- Page fault – synonymous with a miss
- Demand paging – pages moved from disk to main memory only when a word in the page is requested by the processor
- Block placement/replacement decisions must be made each time a block is moved
  - Placement – where a block should go
  - Replacement – what blocks can be removed to make room for new block
MMU and Address Translation

- **Effective address** – address computed by processor while executing a program
- **Logical address** – synonymous with EA but generally used to refer to address when viewed from outside the CPU
- **Virtual address** – address generated from logical address by MMU
- **Physical address** – address presented to the memory unit
- **Note**: every address reference must be translated
Virtual Addresses

- Virtual address space is larger than logical address → programs appear to get more memory
- Example: PowerPC 601
  - 32-bit logical addresses
    - Maximum space allowed for a process
  - 52-bit virtual addresses from MMU translation
    - Process limited to 32-bits but main memory could hold many processes
Virtual Addressing Advantages

- **Simplified addressing** – each program can be compiled into its own memory space (starting at address 0) and could extend beyond physical memory present in system
  - No address relocation required at load time
  - No need to fragment program to accommodate memory limits

- **Cost effective use of physical memory** – less expensive secondary storage (disk) can replace primary storage (RAM)
  - MMU brings in portions of program to physical memory as required

- **Access control** – each memory reference is translated so it can be simultaneously checked for read/write/execute privileges
  - Hardware-level access control
Memory Management by Segmentation

- Allows memory to be divided into segments of varying sizes
  - Less common than paged virtual memory
- Each segment begins at virtual memory address 0
- Segments loaded into/out memory as needed
- Gaps between segments are called external fragmentation
  - Gaps could result in unusable gaps of memory
Segmentation Mechanism

- Physical addresses are computed in the MMU
  - Virtual address is added (integer addition) to segment base register
  - Segment limits may optionally be maintained for error checks
- MMU can switch between separate segments
  - Adjust segment registers
  - Use segment tables
    - One segment per program unit
Memory Management by Paging

- Memory divided into fixed-size pages
  - 512-8K bytes
- Virtual memory is ordered into linear ascending order
  - Necessary for simple page number concatenation for addressing
- MMU maps logical address to physical address
  - May be out of order in memory
  - Page n-1 is not in physical memory by secondary (disk)
- Demand paging only brings pages into memory when needed
Paging Address Translation

- More complex mapping process than segmentation
  - More pages than segments
  - How many pages per program?
- Page table maps virtual pages to physical pages (or secondary memory)
  - Typically 1 page table per user per program unit
- Control fields
  - Access bits – read/write/execute permissions
  - Usage bits – for replacement
- Physical address is from concatenation of page number and word offset
  - Page fault causes exception and the slow update of memory from disk
- Internal fragmentation because a program will most likely not end at page
  - Trade-off between page size and secondary update rate
Page Placement and Replacement

- Page tables are direct mapped
  - Physical page computed directly from virtual page number
  - But, physical pages can reside anywhere in physical memory (like associative cache)
- Page tables as shown (slide 24) result in large page tables because each program requires a page table entry for every page
  - Can instead use multilevel page tables – root page table pointing to other page tables in hierarchy
  - Hash tables used to index only entries present in physical memory
- Replacement strategies are generally LRU
  - Use bit necessary to determine “stale” pages
Fast Address Translation

- Virtual memory is attractive but has considerable overhead
  - 2 memory references required for each actual memory reference
    - First to retrieve page table entry
    - Second to access physical memory location
  - Translation from virtual to physical address translation must happen before cache access
    - Caches designed for physical addresses
- Translation lookaside buffer (TLB) is a small cache for virtual/physical address translations
  - Allows processor to access physical memory directly
    - TLB maintains valid, dirty, protection bits
  - Usually implemented as fully associative cache for flexibility
Operation of Memory Hierarchy
Memory System Design Summary

- Multilevel storage system of modern computer
  - Cache, main memory, disk
- Semiconductor RAM – static and dynamic
  - Used at fastest levels of memory hierarchy
  - RAM speeds has greatest influence on memory system performance
  - Chips arranged into boards and modules
- 2-level memory hierarchy used to connected small and fast with larger, slower memory
  - Cache and main memory is the fastest/closest pair and requires hardware operation
  - Virtual memory is name of main memory/disk pair and because of slower access can be software managed