Computer Systems Organization

CSE 521/560M
Lecture 17
Prof. Patrick Crowley

Plan for Today

- Questions
- Today’s discussion
Main Memory

- Recall that on a cache miss, need to access memory. This process has three components
  - Send the address
  - Get the contents of the memory location
  - Transfer contents
- Parameters for dynamic random access memory (DRAM)
  - *Access time*: time between the read is requested and the desired word arrives
  - *Cycle time*: minimum time between requests to memory
    - cycle time > access time to allow address lines to stabilize and to write back data (i.e., reads are destructive)
    - In 2002, cycle times were 80 ns (compared to 250 ns in 1980)

DRAM

- Bits stored in single transistors
  - Reads are destructive
  - Transistors “leak” (i.e., Dynamic) so must be refreshed
- Address lines split into row and column addresses.
  A read consists of
  - RAS – row access strobe
  - CAS – column access strobe
- Access time is typically RAS + CAS, but periodic refreshing can stall requests
DRAM Structure

DRAM and SRAM

- Both are a memory organization using transistors
- S stands for static
  - Traditionally uses 6 Ts/bit (some use 4 or less)
  - No refresh
  - No need to write after read
- Main memory uses DRAM; on-chip caches use SRAM; on-board caches have been built with both
- DRAM semiconductor fabs use specific processes to create chips
  - Different layers, fewer layers
  - Different physical properties
Improving Main Memory Bandwidth

- Sending address
  - Can’t really be improved
  - **Can** use *split-transaction* bus to allow overlap
- Make memory wider
  - Send one address, return more than one word (if bus width allows it)
  - Wider main memory increases memory increment size
  - Complicates error correcting codes when writing one word in a multi-word memory

Interleaving

- Memory is organized in banks
  - Bank *i* stores all words at address *j mod l*
  - All banks can read a word in parallel
  - Number of banks should match L2 block size
  - Bus does not need to be wider
  - Writes to individual banks for different addresses can proceed without waiting for the preceding write to finish
  - Number of banks limited by increasing chip capacity
    - With 1M*1 bit chips, it takes 64*8=512 chips to get 64MB (easy to put 16 banks of 32 chips)
    - With 64M*1 bit chips, it takes only 8 chips (only one bank)
Memory Width Illustration

Page-mode, Synchronous and Double Data Rate DRAMs

- Page mode: Add a page buffer to DRAM
  - Keep most recent page (e.g., set of addresses) in a buffer
  - Hit: read out of buffer
  - Miss: cost of miss + RAS + CAS
- Synchronous: Add a clock
  - Traditional DRAMs were asynchronous, this eliminates sync overhead at each request
  - Also adds a “width” register to control number of words to be returned
- DDR:
  - Transfer data on the rising and falling edge of clock
  - Doubles data rate
Virtual Memory

- Hardware assists for address translation and memory protection
  - Largely invisible to programmer
- Prior to virtual memory
  - Overlays
  - Relocation registers
  - Paging
  - Segmentation
- Paging systems predate caches
  - But some questions arise (mapping, replacement and write policies)
- An enormous difference: miss penalty

Paged Virtual Memory Illustrated
Two Extremes in the Memory Hierarchy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>L1 Cache</th>
<th>Paging System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block (page) size</td>
<td>16-64 bytes</td>
<td>4K-8K bytes</td>
</tr>
<tr>
<td>Miss (fault) time</td>
<td>10-100 cycles</td>
<td>Millions</td>
</tr>
<tr>
<td></td>
<td>(20-1000 ns)</td>
<td>(3-20 ms)</td>
</tr>
<tr>
<td>Miss (fault) rate</td>
<td>1-10%</td>
<td>0.00001-0.001%</td>
</tr>
<tr>
<td>Memory size</td>
<td>4K-64K bytes</td>
<td>Gigabytes</td>
</tr>
</tbody>
</table>

Other Extreme Differences

- Mapping: Restricted (L1) vs. general (Paging)
  - Hardware assist for virtual address translation (TLB)
  - Miss handler
    - Hardware only for caches
    - Software only for paging system (context-switch)
    - Hardware/Software for TLB
  - Replacement algorithm
    - Not important for caches
    - Critical for paging systems
  - Write policy
    - Always write back for paging systems
Paging and Segmentation

- Paging: fixed block sizes
  - Easy to map, translate, replace, transfer to/from disk
  - Does not correspond to semantic objects, hence internal fragmentation
- Segmentation: variable block sizes
  - Requires two addresses per word: segment and offset
  - “Difficult” to translate (need to check segment length), place and replace (external fragmentation), transfer to/from disk
  - Corresponds to semantic objects
- Segmentation with paging
  - Good for large objects
  - Breaks up page tables

Page Tables

- Page tables contain page table entries (PTEs):
  - Virtual page number (implicit/explicit), physical page number, as well as valid, protection, dirty, and use bits
- Hardware register points to the page table of the running process
- Page table structures
  - early systems had contiguous (in virtual space) page tables
  - Current systems have multi-level page tables
  - Some systems have inverted page tables (with a hash table)
  - In all modern systems, page table entries are cached in a TLB
TLBs

- Cache for page table entries (PTEs)
- TLB miss handled by hardware or by software
  - TLB miss 10-100 cycles, means you don’t need to context switch
- Addressed in parallel with access to cache
- TLB is smaller than page table, admits faster access
  - It’s on the processor’s critical path
- For a given TLB size (number of entries)
  - Larger page size implies larger mapping range
  - Modern O.S.s have a page size parameter
Caches and Address Translation

![Diagram of address translation process]

Virtually Indexed, Physically Tagged Caches

- Does the cache use virtual or physical addresses?
- Why not all physical?
  - Need to translate every time
- Why not all virtual?
  - Protection (page level protection is checked at translation time?)
  - Context switches (each switch changes the meaning of virtual addresses)
    - Can add PID to cache address tag
  - Aliasing (OS and apps can use synonyms to refer to same locations)
  - I/O uses physical addresses
- Idea: use page offset, which is the same for virtual and physical addresses, to index a small cache
  - Limits the size of direct-mapped cache to a page!
I/O and Caches (a first look at cache coherence)

- I/O data passes through the cache on its way from/to memory
  - No coherency problem but contention for cache access
- I/O interacts directly with main memory
- Software solution
  - Output (1) Write through cache, no problem. (2) Write back cache, purge the cache via O.S. interaction.
  - Input (both WT and WB). Use a non-cacheable buffer space. Then after input, flush the cache of these addresses and make the buffer cacheable.

I/O Consistency Illustrated

(a) Cache and memory coherent: A = A and B = B
(b) Cache and memory incoherent: A = A (A state)
(c) Cache and memory incoherent: B = B (B state)
I/O Consistency (Hardware Approach)

- Subset of the shared-bus multiprocessor cache coherence protocol
- Cache controller snoops on the bus
  - On input, if there is a match in tags, store in both the cache and main memory
  - On output, if there is a match in tags: if WT invalidate the entry; if WB take the cache entry instead

Assignment

- Readings
  - For Monday
    • H&P: 5.6-5.7
  - For Wednesday
    • H&P: 5.8-5.10, Skim 5.11
    • Commentary: Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers