Outline of Today’s Lecture

- Cache Replacement Policy
- Cache Write Policy
- Example
- Summary
An Expanded View of the Memory System

- Processor
- Control
- Datapath
- Memory
- Memory
- Memory
- Memory

- Speed: Fastest
- Size: Smallest
- Cost: Highest
- Slowest
- Biggest
- Lowest

The Need to Make a Decision!

- Direct Mapped Cache:
  - Each memory location can only mapped to 1 cache location
  - No need to make any decision :-)
    - Current item replaced the previous item in that cache location

- N-way Set Associative Cache:
  - Each memory location have a choice of N cache locations

- Fully Associative Cache:
  - Each memory location can be placed in ANY cache location

- Cache miss in a N-way Set Associative or Fully Associative Cache:
  - Bring in new block from memory
  - Throw out a cache block to make room for the new block
  - We need to make a decision on which block to throw out!
Cache Block Replacement Policy

° Random Replacement:
  • Hardware randomly selects a cache item and throw it out

What is the problem with this? Can we do better?

Cache Block Replacement Policy

° Least Recently Used:
  • Hardware keeps track of the access history
  • Replace the entry that has not been used for the longest time

What about Cost/Performance?
Cache Block Replacement Policy – A compromise

- Example of a Simple “Pseudo” Least Recently Used Implementation:
  - Assume 64 Fully Associative Entries
  - Hardware replacement pointer points to one cache entry
  - Whenever an access is made to the entry the pointer points to:
    - Move the pointer to the next entry
  - Otherwise: do not move the pointer

Cache Write Policy: Write Through versus Write Back

- Cache read is much easier to handle than cache write:
  - Instruction cache is much easier to design than data cache

- Cache write:
  - How do we keep data in the cache and memory consistent?

- Two options (decision time again :-)
  - Write Back: write to cache only. Write the cache block to memory when that cache block is being replaced on a cache miss.
    - Need a “dirty” bit for each cache block
    - Greatly reduce the memory bandwidth requirement
    - Control can be complex
  - Write Through: write to cache and memory at the same time.
    - What!!! How can this be? Isn’t memory too slow for this?
Write Buffer for Write Through

- A Write Buffer is needed between the Cache and Memory
  - Processor: writes data into the cache and the write buffer
  - Memory controller: write contents of the buffer to memory

- Write buffer is just a FIFO:
  - Typical number of entries: 4
  - Works fine if: Store frequency (w.r.t. time) << 1 / DRAM write cycle

- Memory system designer’s nightmare:
  - Store frequency (w.r.t. time) -> 1 / DRAM write cycle
  - Write buffer saturation

Write Buffer Saturation

- Store frequency (w.r.t. time) -> 1 / DRAM write cycle
  - If this condition exist for a long period of time (CPU cycle time too quick and/or too many store instructions in a row):
    - Store buffer will overflow no matter how big you make it
    - The CPU Cycle Time <= DRAM Write Cycle Time

- Solution for write buffer saturation:
  - Use a write back cache
  - Install a second level (L2) cache:
Write Allocate versus Not Allocate

- Assume: a 16-bit write to memory location 0x0 and causes a miss
  - Do we read in the rest of the block (Byte 2, 3, ... 31)?
    - Yes: Write Allocate
    - No: Write Not Allocate

```
  | Cache Tag | Example: 0x00 | Cache Index | Byte Select |
  +-----------+--------------+-------------+-------------
  |           |             |             |             |
  | 0x00      |             |             |             |
  +-----------+--------------+-------------+-------------
```

What is a Sub-block?

- Sub-block:
  - A unit within a block that has its own valid bit
  - Example: 1 KB Direct Mapped Cache, 32-B Block, 8-B Sub-block
    - Each cache entry will have: 32/8 = 4 valid bits
- Write miss: only the bytes in that sub-block is brought in.
SPARCstation 20's Memory System

- Memory Bus (SIMM Bus): 128-bit wide datapath
- Memory Controller
- Memory Module 0
- Memory Module 1
- Memory Module 2
- Memory Module 3
- Memory Module 4
- Memory Module 5
- Memory Module 6
- Memory Module 7

SPARCstation 20's External Cache:

- Size and organization: 1 MB, direct mapped
- Block size: 128 B
- Sub-block size: 32 B
- Write Policy: Write back, write allocate
SPARCstation 20's Internal Instruction Cache:

- Size and organization: 20 KB, 5-way Set Associative
- Block size: 64 B
- Sub-block size: 32 B
- Write Policy: Does not apply

Note: Sub-block size the same as the External (L2) Cache

SPARCstation 20's Internal Data Cache:

- Size and organization: 16 KB, 4-way Set Associative
- Block size: 64 B
- Sub-block size: 32 B
- Write Policy: Write through, write not allocate

Sub-block size the same as the External (L2) Cache
Two Interesting Questions?

- Why did they use N-way set associative cache internally?
  - Answer: A N-way set associative cache is like having N direct mapped caches in parallel. They want each of those N direct mapped cache to be 4 KB. Same as the "virtual page size."
  - Virtual Page Size: cover in next week's virtual memory lecture

- How many levels of cache does SPARCstation 20 has?
  - Answer: Three levels.
    (1) Internal I & D caches, (2) External cache and (3) ...

SPARCstation 20's Memory Module

- Supports a wide range of sizes:
  - Smallest 4 MB: 16 2Mb DRAM chips, 8 KB of Page Mode SRAM
  - Biggest: 64 MB: 32 16Mb chips, 16 KB of Page Mode SRAM
Summary:

• Replacement Policy
  • Exploit principle of locality

• Write Policy:
  • Write Through: need a write buffer. Nightmare: WB saturation
  • Write Back: control can be complex

• Getting data into the processor from Cache and into the cache from slower memory are one of the most important R&D topics in industry.