Caches

Memory location

CMPE01 Fall 2002
Cache

- Direct mapped cache
- Reads and writes
- Cache associativity
- Cache and performance

Textbook: 7.1 to 7.3

Caches

Basic concepts

data locality:
  - temporal locality
  - spatial locality
block: amount of information transferred (in bytes or words)

hit: the block is present
  - hit rate: fraction of times a requested block is found
  - hit time: time to fetch a block that is present

miss: the block is not present
  - miss rate: fraction of times a requested block is not present
    (miss rate = 100% hit rate)
  - miss penalty: time (in clock cycles) to fetch a block from the lower level
Direct-mapped caches

The cache index

Many different memory blocks map to a single cache block which block? Use the memory address' lower bits to index the cache.

cache index = (memory block address) % (cache size in blocks)

Example 1: 32-block main memory, 8-block cache (we consider block addresses).
The memory block address is ... bits. To index the cache we need ... bits of the memory block address.
The memory block 01001 maps to the cache location ....
The memory block 10110 maps to the cache location ....

Cache mappings

Size of cache < size of main memory

- direct-mapped cache
- set-associative cache
- fully-associative cache

Example 2: 128 byte main memory, 8 block cache, 4 byte (= 1 word) cache block size (we consider byte addresses).

- ... bit memory byte address
- ... bit cache (block) index
- ... bits to address the byte within the block

The memory addresses 0100100, 0100101, 0100110, and 0100111 all map to the same cache block ....

Direct-mapped cache

Direct mapped cache

Each memory block is mapped to exactly one block in the cache.

cache block (4 bytes)
Direct-mapped caches

1-word block, direct-mapped cache

- Memory byte address
- Index
- CPU
- Tag
- Data
- Hit

- Cache line size [b]
- Bits for index
- Bits for tag
- Total cache size [b]

Direct-mapped caches

Cache trace with block address

32 block memory, 8 block cache, the memory address is a block address

- Address
  - Dec
  - Bin
  - Hit/Miss
- INDEX
  - V
  - TAG
  - DATA
  - 000
  - 001
  - 010
  - 011
  - 100
  - 101
  - 110
  - 111

Direct-mapped caches

The Tag field

Many different memory blocks map to a single cache block. How do we know which memory block is in the cache block?

To each cache line we add a tag that contains the remaining part (upper bits) of the address.

Example 3: 32-block main memory, 8-block cache. Memory blocks 00001, 01001, 10001, and 11001 all map to the same cache block.

- Memory block address
  - 00001
  - 00101
  - 01001
  - 01101
  - 10001
  - 10101
  - 11001
  - 11101

- Tag
  - 000
  - 001
  - 010
  - 011
  - 100
  - 101
  - 110
  - 111

- Cache index
  - 0
  - 1
  - 2
  - 3

Direct-mapped caches

The Valid bit

The CPU performs many different tasks, and the memory contents change. How do we know if a cache block is “good”?

To each cache line we add a valid bit to indicate whether the content of the block corresponds to what the CPU is actually looking for.

For instance, after a reset, all valid bits are reset. No block contains useful information.
Cache reads and writes

Read misses

For instructions:
• stall the CPU
• send the original PC to memory (current PC 4) and wait
• write the cache entry (including tag and valid bit)
• restart the instruction

For data:
• stall the CPU
• send the address to memory and wait
• write the cache entry (including tag and valid bit)
• restart the instruction

Cache trace with byte address

256 byte memory, 32 byte cache, 4 byte cache block, memory byte addressing

<table>
<thead>
<tr>
<th>Address</th>
<th>Bin</th>
<th>Hit/Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>001</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>011</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>

Cache reads and writes

Write misses

What is a write miss?

In a 1 word block cache, writes always hit. We do not need to know what was in the memory location, since the CPU is overwriting it anyway.

Problem: inconsistency

Solutions:
• write-through
• write-back

Cache reads and writes

In our CPU, Instruction Memory and Data Memory are actually cache memories,

On a memory access, hits are straightforward to handle.

Misses are more complex:
• read misses
• write misses
Multi-word caches

Using cache blocks larger than one word takes advantage of spatial locality. 4GB memory, 64KB direct-mapped cache with 4-word data blocks (16 bytes)

Exercise

What is the total size in bits of the cache in the previous slide?
Miss penalty (= additional clock cycles)

Has three components:
- a) sending the address to memory
- b) latency to initiate the memory transfer
- c) time for transferring each word

Example:
a) = 1 clock cycle, b) = 15 clock cycles, c) = 1 clock cycle

With a 4 word block cache and a 1 word memory bus, the miss penalty on a standard DRAM is:

On an SDRAM or with an interleaved memory organization is:

Hits/misses in a multi-word cache

Read: Just like the read misses on a single-word cache, except that the entire block is fetched.

Write: We can not just write the word, tag, and valid bit without verifying whether the block is the actual block we want to write to, since more than one memory block maps to the same cache block.

We need to compare the tag for writes too,
- the tags match: we can write the word
- the tags do not match: we need to read the block from memory and then write the word

Cache block size and miss rate

- up to a certain point, cache miss rate decreases with increasing block size
- after a certain point, cache miss rate increases with increasing block size
  - spatial locality decreases with block size
  - the miss penalty increases with block size
Cache associativity

Four-way set associative cache

![Diagram of a four-way set associative cache]

Cache associativity

What if the CPU keeps accessing two (or more) variables that map to the same location in a direct-mapped cache?

More sophisticated strategy: n-way set associative caches,

- direct-mapped ("1-way set associative")
- n-way set associative
- fully associative

Fully associative cache ≡ 8-way set associative cache

any block can go anywhere

![Diagram of a fully associative cache]

Two-way set associative cache

![Diagram of a two-way set associative cache]
Exercise 1

For an 8-line, write-through, 2-way set-associative cache with LRU replacement and 1-word data block, trace the following sequence of addresses:

<table>
<thead>
<tr>
<th>dec</th>
<th>binary</th>
<th>H/M</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>00010111</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>00010010</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>11000100</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>01111111</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>01000101</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>00010010</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td>11000100</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>10100101</td>
<td></td>
</tr>
</tbody>
</table>

Exercise 2

A computer system has 32-bit addresses and a 64-KB direct-mapped, write-back cache with 8-byte data block lines.

a) how many lines are in the cache?

b) how many bits total (including cache management bits) are in each line, minimum?

c) what is the total cache size in bits?

d) diagram a cache lookup

Pros and cons of increasing cache associativity

Advantages:
- reduces the miss rate

Disadvantages:
- requires more hardware
- requires a replacement policy

Block replacement policy:
- Least Recently Used (LRU) or random
- implemented in hardware
Solution:

a) # of sets

b) # of lines

c) # of bits per line

d) total cache size

e) cache lookup

Exercise 4: A computer has a CPI of 1.0 when there are no cache misses, and a 100 MHz clock. Each instruction fetch gets exactly one instruction. Each instruction has on average 0.4 data memory references. For each cache miss the instruction takes an additional 9 clock cycles to complete.

- what are the CPI_{100\%} and the MIPS_{100\%} rating with a cache and an unrealistic 100\% hit rate?
- what are the CPI_{NOCACHE} and the MIPS_{NOCACHE} rating without a cache?
- what are the CPI_{85\%} and the MIPS_{85\%} rating with a cache and a 90\% hit rate on instructions and an 85\% hit rate on data?

Cache associativity

Exercise 3

Suppose the 64 KB cache in Exercise 2 was instead 2 way set associative with 16-byte lines,

a) how many sets are in the cache?

b) how many lines are in the cache?

c) how many bits total (including cache management bits) are in each line, minimum?

d) what is the total cache size in bits?

e) diagram a cache lookup
Caches and performance

Solution:

- CPI\_100\% =
- MIPS\_100\% =

- CPI\_NOCACHE =
- MIPS\_NOCACHE =

- CPI\_80 \_85 =
- MIPS\_80 \_85 =

Homework

Recommended exercises

- Ex 7.1 to 7.9, 7.13 to 7.17
- Ex 7.20, 7.21

- Ex 7.27  Hint: no need to know the CPI. Remember: the miss penalty is in number of clock cycles, Note: the three machines are identical except for the cache,

- Ex 7.28  Hints: i) no need to know the CPI, ii) since the three machines only differ in the cache system (and in the clock cycle time, of course) we only need to consider the total miss cycles per instruction,

- Ex 7.29 (especially interesting for CS)  Hint: find in what cache blocks array[0], array[131], and array[132] are stored,