Cs 355 Computer Architecture

Cache & Memory

Text:
Computer Organization & Design, Patterson & Hennessy
Chapter 5-5.3, 5.8-5.9

Objectives: The Student shall be able to:
Define memory hierarchy, cache, cache line, cache hit rate, cache miss rate, hit time, miss penalty, write-through, write-buffer, write-back, multilevel caching, l1/primary cache, l2/secondary cache.
Describe how the principle of locality drives cache design.
Describe the difference between direct mapping, fully associative, and N-way set associative cache organizations.
Calculate the execution time of direct mapping, fully associative, and N-way set associative cache organizations, given a set of instructions.
Define wide memory organization, one-word wide memory organization, interleaved memory organization.

- Calculate how long it would take to do reads assuming these different memory organizations.

Class Time:

Lecture	1.5 hours
Exercises	1.5 hours
Total	3 hours

Memory

Memory Hierarchy:
Faster memories are volatile, more expensive per bit, and thus smaller in quantity
Slower memories are cheaper, often non-volatile, and often large in quantity
Goal: Provide most memory possible at cheapest price but provide access time at fastest speed.

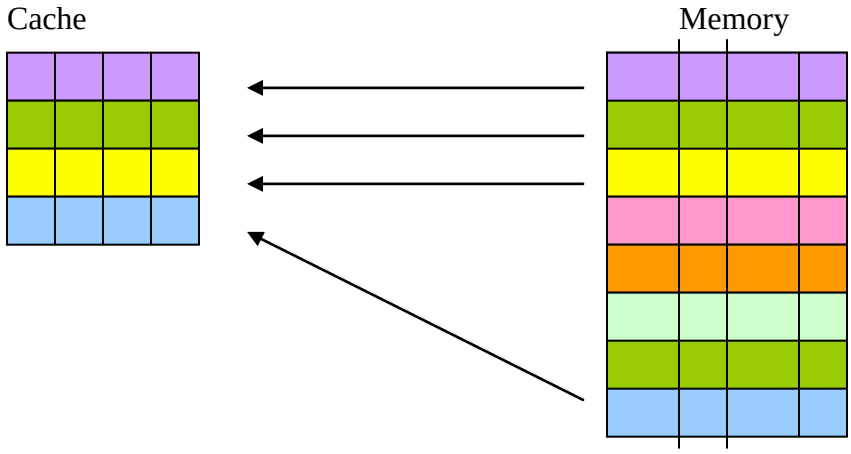
Principle of Locality: programs access memory in a relatively small space at any particular point in time
Temporal Locality: An accessed item is likely to be accessed again soon.
Spatial Locality: Items near an accessed item are likely to be accessed soon.
Is this true for both code and data? Discuss examples

DRAM (Dynamic Random-Access Memory): Used for main memory – larger capacity for silicon use
SRAM (Static Random-Access Memory): Used for cache – faster
Flash Memory: Used as disk in embedded systems

Memory speeds do not keep pace with processor speeds
Memory is a bottleneck with respect to processor speed

Cache

Cache: The level of memory between the processor and main memory
Any memory storage that takes advantage of locality of access



Split Cache: Code and data have separate caches
Unified Cache: One cache for code and data
Cache Size: Number of bytes in cache.
Cache Line or Block: A read/write to/from cache transfers N bytes.
Hit Rate: The percentage of memory accesses that can be found in cache
Miss Rate: The percentage of memory accesses NOT found in cache

Hit Time: The time to retrieve a word from cache, when it is in cache
Cache Miss Penalty: Time to retrieve a word from next lower level of memory when the word is not in cache; includes time to store block into cache.

How do we decide what to put in cache?

Not all of memory fits in cache
Principle of Locality says that most recently accessed memory goes in cache

How much memory do we read into cache at a time?

Principle of Locality says that access should be more than one word large: A block or line is (e.g.) 4 words long

#CacheBlocks = CacheSize / BlockSize

How big should the cache block be?

Little Cache Blocks: Increase the miss rate
Big cache blocks:
Reduces number of total cache blocks; eventually increases the miss rate
Increase time to load cache block: increases the miss penalty

What happens when all of cache is filled up?

Overlay the Least Recently Used cache block
Writes are a concern: **Inconsistency**: When cache != memory
Write-Through: Each write updates both the cache and memory, ensuring consistency
But this slows writes down to memory's time durations
Write Buffer: Holds data when a write is occurring to memory.
Allows for processor to continue processing even when a write is occurring
Write buffer is cleared when memory returns successful status
Larger write buffers prevent stalls due to write buffer full
If memory store time exceeds the average write generate time, buffering cannot help
Write-Back: Update memory only when cache is to be replaced
Solves the short write time problem
Sets Dirty Bit=1 when write is required
Often used in combination with Write Buffer

Example: Intrinsity FastMATH Processor
Embedded MIPS processor
Has an instruction cache and data cache

- Can simultaneously access an instruction and data word every cycle

Cache size = 4K words with 16-word blocks
Has write-through and write-back and 1-entry write buffer

Multilevel Caching: Two levels of cache:
Primary or L1 Cache: Small, expensive, very fast cache: 1 cycle access time
Secondary or L2 Cache: Large, cheaper, slower cache:

- Often < 10 processor cycles;
- Often 10 times size of L1 Cache

Memory: Often exceeds 100 processor cycles

How is memory stored in cache?

- Only a small portion of memory fits into cache.
- Most cache blocks are a fixed size with consecutive data
- Cache line(s) are accessed using an index calculated from the memory address
- Valid bit is set if data is present
- 'Tag' contains the upper part of the memory address; if match, desired data present.

Where do we put the line of memory in cache?

Multiple possible answers exist. Possibilities include:

Direct Mapped Caches

A memory address maps to one and only one cache block:

BlockIndex = HighOrderMemoryAddress % NumberOfCacheBlocks

Any memory address can be divided into:

Tag	Block index	Byte # in cache block

The size of each field is:

Block index size = Size of # blocks in cache

Byte in block size = Size of number of bytes in block

Tag Size = Remainder of address bits

Example: Assume memory is initialized to contain its address as its value.

A very small cache looks like:

Tag 0x	Valid Flag	Word 1	Word 2	Word 3	Word 4
00	1	0, 1, 2, 3	4, 5, 6, 7	8, 9, A, B	C, D, E, F
01	1	50, 51, 52, 53	54, 55, 56, 57	58, 59, 5A, 5B	5C, 5D, 5E, 5F
00	1	20, 21, 22, 23	24, 25, 26, 27	28, 29, 2A, 2B	2C, 2D, 2E, 2F
02	1	B0, B1, B2, B3	B4, B5, B6, B7	B8, B9, BA, BB	BC, BD, BE, BF

How big is the block? 16 bytes, requires lowest 4 bits of address: 0..F

How many blocks in the cache? 4 blocks, requires 2 middle bits of address: 00..11 or 0..3

How many bits in the tag? Remainder of address (upper part)

Calculate Addresses:

B1: 1011 0001 → Lowest 4 bits defines byte in block = 0001

→ Middle 2 bits defines block index into cache = 11

→ Upper 2 bits defines tag = 10

Above, the Valid Flag indicates whether the block contains data or not.

Valid flags are not shown below but exist.

Example: What data is at 0x005a or binary address 0000 0001 01 1010?

Byte = 1010 Index = 01 Tag = 0000 0001 = 0x01

This address is in memory, since at index 1 there is tag 0x01. Byte addr[0xa]=5A

(Assumes big-endian addressing)

Fully Associative Cache

A block can be stored anywhere in cache (No index)

All cache block tags are searched simultaneously for the desired cache block
Each block must have a comparator to compare address: This is an expensive solution
Cache replaced is Least Recently Used (LRU)

Address is:

Tag	Byte # in Cache Block

Example: Address 11110020
Number of bytes in cache row = 4 words = 16 bytes = 0..f. In this case 0000
Tag = remainder of address. In this case 1111.

Set Associative Cache

A memory address maps to N possible cache blocks:

SetIndex = HighOrderMemoryAddress % NumberOfSetsInCache

N-Way Set Associative Cache: Any address can be stored in N blocks
Combines direct mapped placement with fully associative placement
The N blocks are search simultaneously for the desired tag.

A 4-Way Set Associative cache looks like:

SetIndex	**Tag**	**Data**	**Tag**	**Data**	**Tag**	**Data**	**Tag**	**Data**
0								
1								

Address is:

Tag	Set #	Byte # in Cache Block

Example: 10011110
Byte# = 4 words/cache row = 16 bytes = 0..f. In this example: 1110
Set# = 2 possible sets = 0..1. In this example: 1
Tag# = remainder bits in address. In this example: 100

Example 2: Is this address in cache? 100111101
A 4-Way Set Associative cache looks like:

Set	**Tag**	**Data**	**Tag**	**Data**	**Tag**	**Data**	**Tag**	**Data**
0	1100		1011		1000		1111	
1	0000		0011		1001		1010	

Answer: Separate address by field: tag=1001 set = 1 byte = 1101
Yes! Tag 1001 does exist in Set 1, and all bytes 0000..1111 are in the Data.

What happens when a cache miss occurs?

The pipeline stalls.
PC = original PC value
Request Read to main memory: 1 cycle
Read memory: 15 cycles
Transfer read to cache: 1 cycle
Restart the instruction to request Read from cache: 1 cycle
If each word needs to be requested from memory into cache, and cache block = 4 words, then access time = (4*1) + (4*15) + 1 = 65 cycles
One-Word Wide: Each word is requested from memory individually – 4 accesses
Wide Memory Organization: Cache/Bus/Memory inerface is 2-word-size wide
Interleaved Memory Organization: Four memories exist and can do read simultaneously – only one can transmit at a time

One-Word Wide Memory Organization		**Wide Memory Organization**		**Interleaved Memory Organization**	
Cache	Memory	Cache	Memory	Cache	Memory
Addr1---------->		Addr1---------->		Addr1---------->	
	(15 cycles)		(15 cycles)		(15 cycles)
<----------Write1		<----------Write1		<----------Write(1)	
	(15 cycles)		(15 cycles)	<----------Write(2)	
<----------Write2		<----------Write2		<----------Write(3)	
	(15 cycles)			<----------Write(4)	
<----------Write3					
	(15 cycles)				
<----------Write4					
Time = 1+(4*15)+(4*1)=65		1+(2*15)+(2*1)=33		1+(1*15)+(4*1)=20	

Pipelined memory has 4 stages:
Address Translation: Reading the address into the memory for decode
Row decoding: Selecting the requested row in memory
Column decoding: Reading the row (all columns)
Tag validation: Does the memory tag read match the tag desired?

Multiprocessors & Caches

Caches are associated with one processor.

Is it possible for different processors' caches to hold different data for an address?

Snoopy Cache: Caches share a bus, and they listen for other cache accesses.

- Multiple caches may contain the same data for READ purposes.
- Write Invalidate protocol ensures that newly WRITTEN data clears other caches for that data.
- Write serialization: All writes must occur sequentially.

Cache Exercise

Assume the following scenario:
A cache has 4 blocks, where each block is 4 words.
Two caches exist, one each for code and data
Cache hit time is 1 cycle. Cache miss time is 20 cycles, using interleaved memory.
Cache is currently empty.
Assume Least Recently Used replacement algorithm is used for Writes.
Write-back occurs. However assume all reads below.

Part 1: Using a Direct Mapped Cache: Each address maps to one block via indexing

1A) Show how the bits are allocated in an address:

Tag: bits	Index: bits	Byte Number bits

1B) The instructions and data that are executed access the following addresses:

Code Address	Read from C or M?	Data Address	Read from C or M?	Delay
4001000		10020000		
4001004		10020024		
4001008				
400100C				
4001010				
4001000		10020004		
4001004		10020028		
4001008				
400100C				
4001010				
4001014				
4001040				
4001044		10020100		
4001048		10020101		
400104C		10020102		
4001050				
4001054				
4001058		10020100		
4001018				

1C) Show what is in the tags for each cache block at the end of the program:

Instruction Cache		Data Cache	
Tag Value	Address Range	Tag Value	Address Range

1D) What is the cache hit ratio?

Part 2: Using a 2-Way Set Associative Cache

In this configuration there are two sets, each with two possible blocks.
Use an index (modulo) to find the correct set, then select either block in the set.

2A) Show how the bits are allocated in an address:

Tag: bits	Set: bits	Byte Number: bits

2B) The instructions and data that are executed access the following addresses:

Code Address	Read from C or M?	Data Address	Read from C or M?	Delay
4001000		10020000		
4001004		10020024		
4001008				
400100C				
4001010				
4001000		10020004		
4001004		10020028		
4001008				
400100C				
4001010				
4001014				
4001040				
4001044		10020100		
4001048		10020101		
400104C		10020102		
4001050				
4001054				
4001058		10020100		
4001018				

2C) Show what is in the tags for each cache block at the end of the program:

Instruction Cache			Data Cache		
Set Number	Tag Value	Address Range	Set Number	Tag Value	Address Range
Set 0			Set 0		
Set 1			Set 1		

2D) What is the cache hit ratio?

Part 3: Use a Multilevel Cache

The primary cache access time is 1 cycle. The secondary cache access time is 5 cycles.
Both caches use Fully Associative organization: (i.e.) can place anywhere in cache.
The primary cache block size is 4 words. The secondary cache block size is 16 words.

3A) Show how the bits are allocated for the Primary & Secondary Cache address:

Primary Cache (C1)

Tag: bits	Byte Number: bits

Secondary Cache (C2)

Tag: bits	Byte Number: bits

3B) The instructions and data that are executed access the following addresses:

Code Address	C1, C2, or M?	Data Address	C1, C2, or M?	Delay
4001000		10020000		
4001004		10020024		
4001008				
400100C				
4001010				
4001000		10020004		
4001004		10020028		
4001008				
400100C				
4001010				
4001014				
4001040				
4001044		10020100		
4001048		10020101		
400104C		10020102		
4001050				
4001054				
4001058		10020100		
4001018				

3C) Show what is in the tags for each cache block at the end of the program:

Instruction Primary Cache		Data Primary Cache	
Tag Value	Address Range	Tag Value	Address Range

3D) Show what is in the tags for each cache block at the end of the program:

Secondary Instruction Cache		Secondary Data Cache	
Tag Value	Address Range	Tag Value	Address Range

3E) Assume Interleaved Memory Organization is used. The secondary cache loading time (for 16 words per cache line) would be: (See the appropriate notes.)