EE 156
Advanced Topics in Computer Architecture

Unit1 Memory Systems

Sample Questions: SOLUTIONS

1 (Virtual Memory)	/15
2 (Cache)	/15
3 (Coherence)	/15
4 (Cache)	/15
5 (Essay)	/10
6 (True / False)	/10
Total	/80

Name:

Prof. Mark Hempstead, Tufts University

1.) **Virtual Memory:** A virtual memory system has the following parameters.

(1) Virtual address length: 36-bit
(2) Physical Memory Size: 1GB
(3) Page Size: 4MB
(4) Additional valid, sharing and protection bits: 3
(5) 64-entry Data TLB, 16-way set-associative

Calculate the following:

Length of offset bits in virtual address (bits)	22
Length of offset bits in the physical address (bits)	22
Length of the Virtual Page Number (bits)	14
Length of the Physical Page Number (bits)	8
Total number of Virtual Pages	16384
Size of a page table entry (bits)	11
Page Table Size (bits)	180224
Size of a single TLB entry – Tag only (bits)	12
Size of a single TLB entry – Data only (bits)	11
Total memory for the entire TLB (bits)	1472

Offset: log2(4*1024*1024) = 22 bits (same for virtual and physical)
VPN: 36 – 22 = 14
PPN: 30 – 22 = 8
Vpages: 2^14 = 16384
Page table entry: 8 + 3 = 11
Page table size (bits): 2^14 * 11 = 180224
TLB index: 64-entry/16-way set-associative = 4 sets => 2 index bits
TLB tag is VPN size – index bits = 14 – 2 = 12 bits
Data in TLB is just a page table entry
Total TLB size is 64 entries * (11+12) = 1472

b.) Describe one of the benefits of virtual memory. Which of the above fields and supporting mechanisms are used to provide this feature?

Many possible answers:
Increased addressable memory capacity and scalability: VPN with more bits than PPN and mapping to physical disk (swap)
Portability: The VPN and PPN fields and the translation mechanism
Protection (security): protection bits
Sharing between processes: sharing bits

2) **Caches and Cache Access**
A test application accesses the following memory addresses (8-bit addresses), shown in hex:

0x02, 0x06, 0x0E, 0x03, 0x12, 0x06, 0x03, 0x0E

The L1 cache has the following parameters:

- Word size one byte
- Block size 2 bytes
- Cache capacity: 12 bytes
- Associativity: 3-way set associative
- Physical address size: 8-bits
- Cache Eviction Policy: LRU

Address	Tag	Index	Offset	Hit/Miss
0x02	000000	1	0	M
0x06	000001	1	0	M
0x0E	000011	1	0	M
0x03	000000	1	1	H
0x12	000100	1	0	M
0x06	000001	1	0	M
0x03	000000	1	1	H
0x0E	000011	1	0	M

a.) Calculate the size of the Tag, index and offset bits and populate in the table above.

Index = 12 bytes / (2 bytes * 3 ways) = 2 sets or 1 index bit
Offset = 2 bytes / blocK => 1 bit
tag = 8 – 1 – 1 = 6 bits

b) Record each access as a hit or miss in the table above. Draw a diagram showing the final contents of the data and tag portion of the cache below. The cache is initially empty.

Final Contents of the Cache: *(hits and misses in table above)*

	Way0		**Way1**		**Way2**	
	Tag	**Data**	**Tag**	**Data**	**Tag**	**Data**
0	invalid		invalid		invalid	
1	000000	Mem[0x02,0x03]	~~000001~~ ~~000100~~ 000011	~~Mem[0x06,0x07]~~ ~~Mem[0x12, 0x13]~~ Mem[0x0E, 0x0F]	~~000011~~ 000001	~~Mem[0x0E,0x0F]~~ Mem[0x06, 0x07]

(*Keeping track of LRU, this is not required but helps get the cache replacements right)*

LRU (left most is LRU, right most is MRU)

1 (02)	set0 = invalid=0,1,2;	set1=1,2,0
2 (06)	set0=0,1,2	set1=2,0,1
3 (0E)	set0=0,1,2	set1=0,1,2
4 (03)	set0=0,1,2	set1=1,2,0
5 (12)	set0=0,1,2	set1=2,0,1
6 (06)	set0=0,1,2	set1=0,1,2
7 (03)	set0=0,1,2	set1=1,2,0
8 (0E)	set0=0,1,2	set1=2,0,1

d) The pattern above has a significant number of misses. Why and what type of misses?

This pattern first accesses bytes in multiples of 2 so that it has all of the same index avoids one of the sets entirely thus creates more conflict misses.

There are also compulsory misses because 4 unique blocks are access for the first time.

There are NO capacity misses. If the cache was fully associative it would be big enough.

e) How many bits would you need for the tag, index and offset if you switched to a direct-mapped cache with the same block size and cache size (in words) as the first cache configuration.

The fact that the cache has a capacity of 12 and therefore 6 sets, which is not a power of 2, makes a direct map cache a bit potentially lower performing or wasteful.

You have two options either have only 4 sets and leave the other 2 always empty and not addressable:

> *Offset = 1 bit; index = 2 bits; tag = 5 bits*

Or use three bits for indexing 6 physical sets. This means that sets 110 and 111 will always miss; they will be hardcoded invalid because you don't have space for them.

Offset = 1 bit; index = 3 bits; tag = 4 bits

3.) **Cache Coherence**

Assume a processor with two cores (Core A and Core B). Each core has a private L1 **byte addressable data cache** that is kept consistent using the MSI cache coherence protocol. The L1 caches each have the following parameters:

- Block size 64 Bytes
- Cache capacity: 1 K bytes
- Associativity: 4-way set associative
- Physical address size: 16-bits
- Cache Eviction Policy: LRU

a.) Calculate the size of the tag, index and offset bits.

Offset = log2 (64) = 6 bits
Index = 1K / (4 * 64) = 4 sets => 2 bits
Tag = 16 - 6 -2 = 8 bits

b.) Show how the following set of memory accesses changes state bits. Assume the bus is fast enough that there is no latency in the transfer. Report if state changed for the two shared addresses xF000, xD000. The cache is initially empty.

	Commands			**State of Cache Line xF000 (I/S/M)**			**State of Cache Line xD000 (I/S/M)**	
Cycle	**Core A**	**Core B**		**Core A**	**Core B**		**Core A**	**Core B**
10	Write xF000	Read xD000		*Modified*	*Invalid*		*Invalid*	*Shared*
20	Write xD000	Read xF000		*Shared*	*Shared*		*Modified*	*Invalid*
30	Read xF000	Read xD000		*Shared*	*Shared*		*Shared*	*Shared*
40	Write xF000	Read xD000		*Modified*	*Invalid*		*Shared*	*Shared*
50	Read xF000	Write xD000		*Modified*	*Invalid*		*Invalid*	*Modified*
60	Read xD000	Write xF000		*Invalid*	*Modified*		*Shared*	*Shared*

4) Calculate the following cache features based on the Intel Sandybridge memory hiearchy:

- 32 kB L1 Data Cache Capacity, byte addressable
- 64 B blocks
- 8-way set associative
- Memory addresses of 48-bits

Calculate the following:

- Bits for the **tag** of the L1 data cache?
- Bits for **index** of the L1 data cache?
- Bits for the **offset** of the L1 data cache

(32KB) * (1 block/64B) = 512 blocks

(512 blocks)*(1 set/8 blocks) = 64 sets

So there are log2(# sets)=log2(64)=6 index bits.

There are log2(block size) = log2(64) = 6 offset bits.

That leaves 48-6-6=36 tag bits.

5.) **Essay Questions**

a.) Define *the Memory Wall* and explain why it has threatened to hinder the progress promised by Moore's Law. Second, describe a technique computer architects have used to mostly keep the effects of the Memory Wall from reducing application performance.

*"The memory wall" is a term to describe the fact that **memory latency** has improved at a much slower rate than logic transistor speed. Thus while core performance has improved, the memory system lags in providing the data needed by the cores quickly. Consequently, system designers have focused their energy developing a **memory hierarchy** with fast L1 caches and L2 caches with a high hit rate that hide the full effects of the memory wall. Additional cache enhancements such as associatively and way-prediction, pre-fetching all increase hit rate.*

E. What are the 3 Cs of cache misses? Give one example of how a using technique to address one type of miss could actually increase the number of misses of a different type.

Compulsory (a miss on the first access to any data in a block), Capacity (a miss on a block that was previously evicted because the entire cache was full), and Conflict (a miss on a block that was previously evicted because its set was full). You can reduce conflict misses by using smaller blocks and more ways (without increasing the cache size); however, this would increase compulsory misses.

8) **True and False Questions** (2 points each)

(Just circle true/false no explanation required)

a) In *The Performance Equation*, each term is independent; improving one term such as CPI with a hardware software technique never impacts any of the other terms (true/false)

   In practice it is difficult to design a hardware or software technique that does not impact another term such as instruction count or cycle time.

b) The TLB is managed by software through the operating system's scheduler (true/false)

The TLB is a cache of the page table and is managed in hardware. If it works at all with the OS it is with the memory manager that controls the virtual memory system.