CS105 – Computer Systems Spring 2023

Week 6: Caches

SOLUTION

February 27 - March 1, 2023

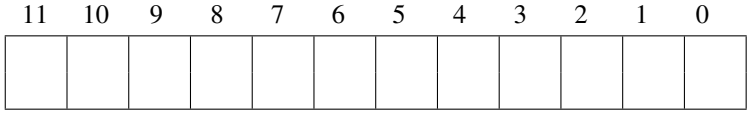
1. **Direct-Mapped Caches.** The following table depicts a direct-mapped cache, with an 8 byte block size and 4 cache lines:

Direct-Mapped Cache										
Index	Tag	Valid	Data							
0	29	0	34	29	8E	00	39	AE	AB	07
1	73	1	0D	8F	AA	E9	0C	3C	EA	01
2	A7	1	88	4B	E2	04	D2	13	B0	05
3	3B	1	AC	99	FF	1F	B5	47	0D	00

You should assume:

- Memory is byte addressable. All memory accesses read/write 1-byte.
- Memory addresses are 12 bits.

(a) The box below depicts a 12-bit memory address. Indicate (by labeling the diagram) the fields that would be used to determine (1) the tag, (2) the index, and (3) the offset.



CT: [11-5] CI: [4-3] CO: [2-0]

(b) Consider the following sequence of accesses (yes, they occur sequentially). For each access, determine the tag, index, and offset. Then indicate whether that access would correspond to a cache hit or a cache miss, and what byte is read (if the exact value is unknown because it is not shown in the initial cache diagram, use the notation `MEM[addr]` instead of giving the byte).

Operation	Tag	Index	Offset	Hit?	Byte read
i. `Read 0xAB8`	`0x55`	3	0	Miss	`Mem[0xAB8]`
ii. `Read 0xE68`	`0x73`	1	0	Hit	`0x0D`
iii. `Read 0x524`	`0x29`	0	4	Miss	Mem[0x524]
iv. `Read 0xE6C`	`0x73`	1	4	Hit	`0x0C`
v. `Read 0x526`	`0x29`	0	4	Hit	Mem[0x526]
vi. `Read 0x528`	`0x29`	1	0	Miss	Mem[0x528]

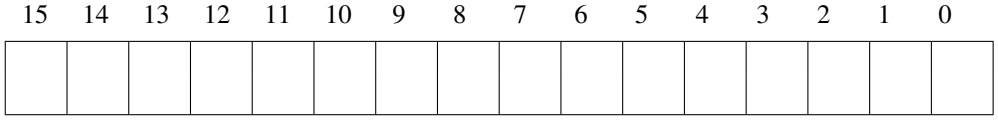
2. **Set-Associative Caches.** The following table depicts a 4-way set associate cache, with a 2 byte block size and 32 total lines:

4-way Set Associative Cache																
Index	Tag	Valid	Data		Tag	Valid	Data		Tag	Valid	Data		Tag	Valid	Data	
0	029	0	34	29	787	0	39	AE	C7D	1	68	F2	88B	1	64	38
1	AF3	1	0D	8F	C3D	1	0C	3A	D4A	1	A4	DB	6D9	1	A5	3C
2	2A7	1	E2	04	FAB	1	D2	04	BE3	0	3C	A4	D01	1	EE	05
3	23B	0	AC	1F	1E0	0	B5	70	B3B	1	66	95	E37	1	49	F3
4	780	1	60	35	02B	0	19	57	549	1	8D	0E	100	0	70	AB
5	EEA	1	B4	17	0CC	1	67	DB	08A	0	DE	AA	118	1	2C	D3
6	11C	0	3F	A4	D01	0	3A	C1	9F0	0	20	13	E7F	1	DF	05
7	D0F	0	00	FF	2AF	1	B1	5F	099	0	AC	96	C3A	1	22	79

You should assume:

- Memory is byte addressable, and all memory accesses read/write 1 byte.
- Memory addresses are 16 bits.
- The cache uses a least-recently used (LRU) eviction policy.
- The cache is write-back, write-allocate.
- No writes occur prior to the beginning of this problem.

(a) The box below depicts a 16-bit memory address. Indicate (by labeling the diagram) the fields that would be used to determine (1) the tag, (2) the index, and (3) the offset.



CT: [15-4] CI: [3-1] CO: [0]

(b) Consider the following sequence of accesses (yes, they occur sequentially). For each access, indicate whether that access would correspond to a cache hit or a cache miss, which byte is read (for reads), and whether or not a memory write will occur. Use the notation `MEM[addr]` for reads are cache misses.

Operation	Tag	Index	Offset	Hit?	Byte read	Mem write?
i. `Write $0x01, 0x0CCB`	`0x0CC`	5	1	Hit	n/a	N
ii. `Read 0xEEAA`	`0xEEA`	5	0	Hit	B4	N
iii. `Read 0x118B`	`0x118`	5	1	H	D3	N
iv. `Write $0x02, 0x047B`	`0x047`	5	1	Miss	n/a	N
v. `Read 0x119B`	`0x119`	5	1	Miss	Mem[0x119B]	Y
vi. `Read 0x047A`	`0x047`	5	0	Hit	Mem[0x047A]	N