DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Computer Organization and Architecture (CS2600)

Total Time : 50 minutes | **Class Test 6 (Cache Memories)**
Max Marks : 32 | **23-04-2024**

---

1. Consider a program which has the following behavior: load instructions is 25%; store instructions is 10%; instruction cache miss rate 0.3%; data cache miss rate 2%; CPI is 2; cache line size 64.

   What are the read and write bandwidths (measured by bytes per cycle) between RAM and the cache for (A) Instructions (B) Data Loads (C) Data Stores (D) Total Bandwidth for Reads and Writes? *Assume 1.* Each miss generates a request for one block. *Assume 2.* There is a single level, unified, write-through, write-allocate cache. *Assume 3.* Loads and Stores are always in words (4-bytes). 12 marks (3x4)

   Bandwidth due to Instructions.

   - CPI 2, means 2 cycles per instruction. Or, 0.5 instructions complete per clock cycle.
   - Instruction cache miss rate is 0.3%. Each cache miss fetches 64 bytes.
   - Cache-Memory bandwidth due to instructions is $0.5 \times 0.003 \times 64 = 0.0960$bytes/cycle

   Bandwidth due to Data Loads

   - 0.5 instructions complete per clock cycle.
   - 25% of these are load instructions, of which 2% result in cache misses.
   - Cache-Memory bandwidth due to memory loads is $0.5 \times 0.25 \times 0.02 \times 64 = 0.16$bytes/cycle

   Bandwidth due to Data Stores

   - 0.5 instructions complete per clock cycle of which 0.1 are stores.
   - Since it is write-through, every store leads to a memory access. Only 4-bytes (1 word) is written back to memory.
   - Cache-Memory bandwidth due to write-through is $0.5 \times 0.1 \times 4 = 0.2$bytes/cycle.
   - Due to write-allocate, there is 64-bytes loaded for each store that results in a cache miss. Since data cache miss rate is 2%, Cache-Memory bandwidth due to write-allocate is $0.5 \times 0.1 \times 0.02 \times 64 = 0.064$bytes/cycle

   Total Bandwidth

   - Read: $0.0960 + 0.16 + 0.064 = 0.320$bytes/cycle
   - Write: 0.2bytes/cycle

2. Continuing question 1.... If *Assume 2* is changed to "There is a single-level, unified, `write-back` cache" and a new assumption, *Assume 4* is added, which states thatthe percentage of dirty bits set in the cache is 30%. Then, what is the bandwidth. 3 marks

   - Cache-Memory Read bandwidth does not change.
   - Writes occur when there is a cache miss and a dirty line is replaced. This can occur only occurs only for data loads and stores (and not instructions). Thus bandwidth of writes is $0.5 \times (0.25 + 0.1) \times 0.02 \times 0.3 \times 64 = 0.067$bytes/cycle

3. (a) Sketch the organization of a cache address to a 3-way set-associative cache with two-word cache lines and a total size of 48 words. The cache is word-adressable (not byte) and uses a Least Recently Used replacement policy. Clearly show the width of the tag and data fields. What is the minimum size of the tag field? 3 marks

   (b) For this sequence of *word* addresses, draw a table to show the (A) way and (B) set they can be found in, and (C) if there is a hit or a miss, and (D) contents of the tag. 0x03, 0xb4, 0x2b, 0x02, 0xbe, 0x58, 0xbf, 0x0e, 0x1f, 0xb5, 0xbf, 0xba, 0x2e, 0xce. 14 marks

   Organization of the cache.

   - bytes per cache line = 4*2 = 8

- Number of cache lines in the cache is $48/2 = 24$. (#words/#(words/line)).
- Cache is 3-way set associative therefore, number of sets is $24/3 = 8$. (#(cache lines)/(#ways))
- The address will have 8-bits, as follows: **b0**: offset within a line (1 bits because 8-words per line); **b3, b2, b1**: set address (3 bits needed because there are 8 sets); **b7, b6, b5, b4**: tag (we need to have a minimum of 2 bits to distinguish between the 3 ways).

Address to Cache mapping.
Tag has the following format: {X, Y, Z}. X is least recently used, Z is most recently used

access	Address	binary	tag	set	offset	H/M	Tags	Comment
1	0x03	(0000 001 1)	0	1	1	M	Tag[0x1]={0}	Way1 used
2	0xb4	(1011 010 0)	b	2	0	M	Tag[0x2]={b}	Way1 used
3	0x2b	(0010 101 1)	2	5	1	M	Tag[0x5]={2}	Way1 used
4	0x02	(0000 001 0)	0	1	0	H	Tag[0x1]={0}	Hit due to way1
5	0xbe	(1011 111 0)	b	7	0	M	Tag[0x7]={b}	Way1 used
6	0x58	(0101 100 0)	5	4	0	M	Tag[0x4]={5}	Way1 used
7	0xbF	(1011 111 1)	b	7	1	H	Tag[0x7]={b}	Hit due to way1
8	0x0e	(0000 111 0)	0	7	0	M	Tag[0x7]={b, 0}	Way2 used
9	0x1f	(0001 111 1)	1	7	1	M	Tag[0x7]={b, 0, 1}	Way3 used
10	0xb5	(1011 010 1)	b	2	1	H	Tag[0x2]={b}	Hit
11	0xbf	(1011 111 1)	b	7	1	H	Tag[0x7]={0, 1, b}	Hit due to way1
12	0xba	(1011 101 0)	b	5	0	M	Tag[0x5]={2, b}	Way2 used
13	0x2e	(0010 111 0)	2	7	0	M	Tag[0x7]={1, b, 2}	Replaced Least recently used (0), tag in way 2
14	0xce	(1100 111 0)	e	7	0	M	Tag[0x7]={b, 2, c}	Replaced Least recently used (1), tag in way 3