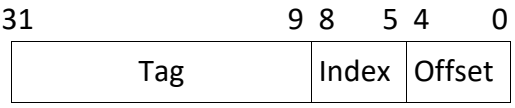
HW5 Solutions (CS552 Spring 2013)

Problem 4 – Direct Mapped Cache

1.
Number of Lines = Cache Capacity/(Block or Line Size) = 512/32 = 16 Cache lines Number of Index Bits = log2 (Number of lines)

Offset: log2 (block size) = log2 (32) = 5 bit offset
Index: log2 (number of lines) = log2 (16) = 4 bit index
Tag: 32 bit address - 4 bit index - 5 bit offset = 23 bit tag



2.

ADDRESS	TAG	INDEX	OFFSET	HIT/MISS	C TYPE
0x0000a796	0x000053	c	16	Miss	Compulsory
0x000092e8	0x000049	7	8	Miss	Compulsory
0x000092f4	0x000049	7	14	Hit	
0x00004182	0x000020	c	2	Miss	Compulsory
0x0000780a	0x00003c	0	a	Miss	Compulsory
0x0000a690	0x000053	4	10	Miss	Compulsory
0x0000408e	0x000020	4	e	Miss	Compulsory
0x0000a798	0x000053	c	18	Miss	Conflict
0x00007800	0x00003c	0	0	Hit	
0x000092fc	0x000049	7	1c	Hit	
0x00027c02	0x00013e	0	2	Miss	Compulsory
0x0000408a	0x000020	4	a	Hit	
0x00004198	0x000020	c	18	Miss	Conflict
0x00006710	0x000033	8	10	Miss	Compulsory
0x0000670c	0x000133	8	c	Hit	

0x00027c04	0x00003e	0	4	Hit	
0x0000a790	0x000053	c	10	Miss	Conflict

3.

INDEX	TAG
0	00013e
1	
2	
3	
4	000020
5	
6	
7	000049
8	000033
9	
a	
b	
c	000053
d	
e	
f	

Total Hits: 6 Total Miss: 11

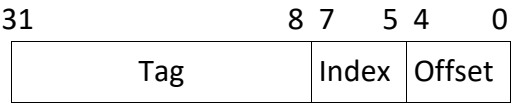
4.

(a) Compulsory Misses - Can be reduced by increasing the block size.
(b) Conflict Misses - Can be reduced by increasing the associativity.
(c) Capacity Misses - Can be reduced by increasing the cache size.

Problem 5 – Set Associative Cache

1.

Offset: same as before
Index: log2 (number of lines/num of ways) = log2(16/2) = 3 bit index Tag:
32 bit address - 3 bit index - 5 bit offset = 24 bit tag



2.

ADDRESS	TAG	INDEX	OFFSET	HIT/MISS	C TYPE	Way
0x0000a796	0x0000a7	4	16	Miss	Compulsory	1
0x000092e8	0x000092	7	8	Miss	Compulsory	1
0x000092f4	0x000092	7	14	Hit		1
0x00004182	0x000041	4	2	Miss	Compulsory	2
0x0000780a	0x000078	0	a	Miss	Compulsory	1
0x0000a690	0x0000a6	4	10	Miss	Compulsory	1
0x0000408e	0x000040	4	e	Miss	Compulsory	2
0x0000a798	0x0000a7	4	18	Miss	Conflict	1
0x00007800	0x000078	0	0	Hit		1
0x000092fc	0x000092	7	1c	Hit		1
0x00027c02	0x00027c	0	2	Miss	Compulsory	2
0x0000408a	0x000040	4	a	Hit		2
0x00004198	0x000041	4	18	Miss	Conflict	1
0x00006710	0x000041	0	10	Miss	Compulsory	1
0x0000670c	0x000067	0	c	Hit		1
0x00027c04	0x000027c	0	4	Hit		2
0x0000a790	0x0000a7	4	10	Miss	Conflict	2

3.

INDEX	TAG1	TAG2
0	000067	00027c
1		
2		

3		
4	000041	0000a7
5		
6		
7	000092	

4.

Total Hits: 6 Total Miss: 11

The speedup is 1 as the number of hits/miss are same.

Problem 6 - Cache Storage

1.

32 byte block * 1024 sets * 5-ways = 163840 bytes (160KB)

2.

Each line of each way holds a valid bit, a dirty bit, two LRU bits, a tag, and 32 bytes of data:

Tag size = 47bits - # index bits - # offset bits
= 47- log2 1024 - log2 32
= 47- 10 - 5
=32 bits

Line size = valid + dirty _ LRU + tag + data
= 1 bit + 1 bit + 2 bits + 32 bits + 32 bytes * (8 bits/byte)
= 292 bits

Cache size = 5 ways * 1024 sets * set size
= 5 * 1024 * 292
= 1495040 bits

Note: If 3 bit LRU is used, the line size would be, 293 bits and Cache Size will be 187520 bytes.

3.

Similar to the diagram in CODe4 on Page 486 but only has 5 cache tag,data arrays with changes in the positionsof tag, index and offsets in the physical address. Also, the hit logic requires only 5 bit or-gate and 5-to-1 mux for data.

Problem 7

Solution:

Data bits = 256K * 8 = 2^18 * 2^3 = 2^21 bits = 2 Mbits

To calculate the rest we need to figure out how the cache is indexed.

#blocks = 2^18 / 2^4 = 2^14
#sets = 2^14 / 2^2 = 2^12

So, the TAG field is 36-12-4 = 20 bits wide

Tag bits = #blocks * TAG field width = 2^14 * 20 bits

There is one valid bit and one dirty bit per block

Valid bits = 2^14 and Dirty bits = 2^14

The LRU bits are associated with the set. They tell us in which order the blocks within the set were accessed. As we explained we need at least log(ways!) bits to encode this information.
So,
LRU bits -= 2^12 * log(4!)

Problem 8

5.4.1
a. 8
b. 16

5.4.2
a. 32
b. 64

5.4.3
a. 1 + (22/8/32) = 1.086
b. 1 + (20/8/64) = 1.039

Problem 9

5.7.1
a.
P1 -1.52 GHz P2
- 1.11 GHz

b.
P1 - 926 MHz P2
- 495 MHz

5.7.2

a.

P1 - 6.31 ns = 9.56 cycles P2

- 5.11 ns = 5.68 cycles

b.

P1 - 3.47 ns = 3.21 cycles

P2 - 4.07 ns = 2.02 cycles

5.7.3

a.

P1- 12.64 CPI = 8.34 ns per inst

P2 - 7.36 CPI = 6.63 ns per inst. P2 is faster

b.

P1 4.01 CPI = 4.33 ns per inst

P1 2.38 CPI = 4.81 ns per inst. P1 is faster

Problem 10

Solution:

The miss penalty for L1 is 100ns/0.5ns = 200 cycles/miss. The probability of miss is 1 – 0.99 = 0.01.

Hence the averaged CPI taken into account memory access is 1 + 0.3 * 0.01 * 200 = 1.6 cycles/instr.

Now with a L2 cache with miss ratio of 20 % with an access delay of 5/0.5 = 10 cycles.
Then the averaged CPI with L2 cache may be evaluated at 1 + 0.3 * (0.01 * 10 + 0.01 * 0.2 * 200) = 1 + 0.3 * 0.01 * (10 + 40) = 1.15 cycles/instr.

The performance improvement is 1.6/1.15 – 1 = 9/23 = 39%
