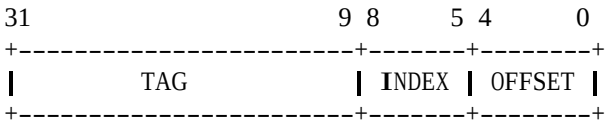
CS/ECE 552 : Introduction to Computer Architecture
Spring 2010
Prof. Wood
Problem Set #5 - Solutions

Problem 3 - Direct Mapped Cache

1. 32 byte blocks $\Rightarrow$ $\log_2 32 = 5$ bit offset
   Number of lines = Cache capacity/(Block or Line Size)
   Number of Index bits = log (Number of lines)
   512 byte capacity $\Rightarrow$ $\log_2(512/32) = 4$ bit index
   32 bit address - 4 bit index - 5 bit offset = 23 bit tag



2.

TAG	INDEX	OFFSET	HIT/MISS	C Type
0x0000DA	C	16	Miss	Compulsory
0x000007	7	8	Miss	Compulsory
0x000007	7	14	Hit	
0x000020	C	2	Miss	Compulsory
0x00003C	0	A	Miss	Compulsory
0x000053	4	10	Miss	Compulsory
0x000020	4	E	Miss	Compulsory
0x0000DA	C	18	Miss	Conflict
0x00003C	0	0	Hit	
0x000007	7	1C	Hit	
0x00013E	0	2	Miss	Compulsory
0x000020	4	A	Hit	
0x000020	C	18	Miss	Conflict
0x000033	8	10	Miss	Compulsory
0x000033	8	C	Hit	
0x00013E	0	4	Hit	
0x0000DA	C	10	Miss	Conflict

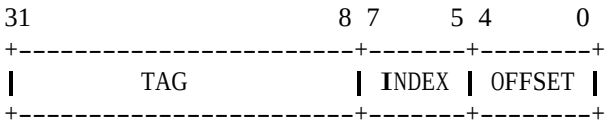
3.

INDEX	TAG
0	00013E
1	
2	
3	
4	000020
5	
6	
7	000007
8	000033
9	
a	
b	
c	0000DA
d	
e	
f	

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 4 - Set-Associative Cache

1. 32 byte blocks => log2 32 = 5 bit offset
   Number of lines = Cache capacity/(Block or Line Size)
   Number of Index bits = log (Number of lines)
   512 byte capacity, 2 ways => $\log_2(512/(32*2)) = 3$ bit index
   32 bit address - 3 bit index - 5 bit offset = 24 bit tag



2.

TAG	INDEX	OFFSET	HIT/MISS	C Type
0x0001b5	4	16	Miss	Compulsory
0x00000e	7	8	Miss	Compulsory
0x00000e	7	14	Hit	
0x000041	4	2	Miss	Compulsory
0x000078	0	A	Miss	Compulsory
0x0000a6	4	10	Miss	Compulsory
0x000040	4	E	Miss	Compulsory
0x0001b5	4	18	Miss	Conflict
0x000078	0	0	Hit	
0x00000e	7	1C	Hit	
0x00027c	0	2	Miss	Compulsory
0x000040	4	A	Hit	
0x000041	4	18	Miss	Conflict
0x000067	0	10	Miss	Compulsory
0x000067	0	C	Hit	
0x00027c	0	4	Hit	
0x0001b5	4	10	Miss	Conflict

3.

INDEX	TAG1	TAG2
0	000067*	00027c
1		
2		
3		
4	000041*	0001b5
5		
6		
7	00000e	

(lru represented by *)

4. The hit ratio remains unchanged. Hence, the speedup is 1.

Problem 5 - Cache Storage

1. 16 byte block * 512 sets * 4-ways = 32768 bytes (32 KB)
2. Each line of each way holds a valid bit, a dirty bit, 2 LRU bits, a tag, and 32 bytes of data:

   (pseudo LRU, and other implementations of LRU were allowed without penalty)

   Tag size = 42 bits - # index bits - # offset bits
   = 42 - $\log_2$ 512 - $\log_2$ 16
   = 42 - 9 - 4
   = 29 bits

   Block Size = valid + dirty + LRU + tag + data
   = 1 bit + 1 bit + 2 bits + 29 bits + 16 bytes * (8 bits/byte)
   = 161 bits

Cache size = 4 ways * (512 sets * (block size))
= 4 * 512 * 161
= 329728 bits
= 41216 bytes

3. Overhead % = (Cache Size - Cache Storage) / Cache Size
= 33 / 161= 20.497%