Cache Memories #1 — 4190.308 Computer Architecture

Name: _____

Due Date: Thursday, December 09, 2014, 23:59 **Student-Number:** _____

Submission: in paper form.
There will be a drop off box in class and in front of the CSAP Lab in building 301, room 419.

Question 1

Cache Memories

Fundamental parameters	
Parameter	Description
$S = 2^s$	Number of sets
E	Number of lines per set
$B = 2^b$	Block size (bytes)
$m = \log_2(M)$	Number of physical (main memory) address bits

Derived quantities	
Parameter	Description
$M = 2^m$	Maximum number of unique memory addresses
$s = \log_2(S)$	Number of set index bits
$b = \log_2(B)$	Number of block offset bits
$T = m - (s + b)$	Number of tag bits
$C = B * E * S$	Cache size (bytes) not including overhead as the valid and tag bits

The following table gives the parameters for a number of different caches. For each cache, determine the number of cache sets (S), tag bits (t), set index bits (s), and block offset bits (b).

Cache	m	C	B	E	S	t	s	b
1.	32	2048	4	1				
2.	32	1024	16	4				
3.	48	1024	32	32				

Question 2
Direct-Mapped Caches

In general, if the high-order *s* bits of an address are used as the set index, contiguous chunks of memory blocks are mapped to the same cache set.

A. How many blocks are in each of these contiguous array chunks?
B. Consider the following code that runs on a system with a cache of the form (S, E, B, m) = (512, 1, 32, 32):

```
int array[4096];

for (i = 0; i < 4096; i++)
    sum += array[i];
```

What is the maximum number of array blocks that are stored in the cache at any point in time?

Question 3
Set Associative Caches

The problems that follow will help reinforce your understanding of how caches work. Assume the following:

The memory is byte addressable.
Memory accesses are to 1-byte words **(not to 4-byte words)**.
Addresses are 13 bits wide.
The cache is two-way set associative ($E = 2$), with a 4-byte block size ($B = 4$) and eight sets ($S = 8$)

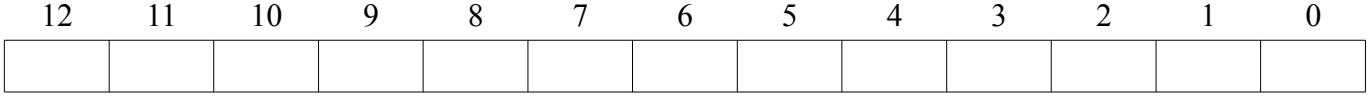
The contents of the cache are as follows, with all numbers given in hexadecimal notation.

2-way set associative cache

	Line 0						Line 1					
Set Index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	-	-	-	-
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	-	-	-	-	0B	0	-	-	-	-
3	06	0	-	-	-	-	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	1	DE	AD	BE	EF
6	91	1	A0	B7	26	2D	F0	0	-	-	-	-
7	46	0	-	-	-	-	DE	1	12	C0	88	37

The following figure shows the format of an address (one bit per box). Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO	The cache block offset
CI	The cache set index
CT	The cache tag



Question 4
Set Associative Caches

Suppose a program running on the machine in **Question 2** references the 1-byte word at address `0x0E36`. Indicate the cache entry accessed and the cache byte value returned in hex. Indicate the cache entry accessed and the cache byte value returned **in hex**. Indicate whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache byte returned."

A. Address format (one bit per box):

12	11	10	9	8	7	6	5	4	3	2	1	0

B. Memory reference:

Parameter	Value
Cache block offset (CO)	`0x`
Cache set index (CI)	`0x`
Cache tag (CT)	`0x`
Cache hit? (Y/N)	
Cache byte returned	`0x`

Question 5

Set Associative Caches

Repeate **Question 4** for memory address `0x0DD5`.

A. Address format (one bit per box):

12	11	10	9	8	7	6	5	4	3	2	1	0

B. Memory reference:

Parameter	Value
Cache block offset (CO)	`0x`
Cache set index (CI)	`0x`
Cache tag (CT)	`0x`
Cache hit? (Y/N)	
Cache byte returned	`0x`

Question 6

Set Associative Caches

Repeate **Question 4** for memory address `0x1FE4`.

A. Address format (one bit per box):

12	11	10	9	8	7	6	5	4	3	2	1	0

B. Memory reference:

Parameter	Value
Cache block offset (CO)	`0x`
Cache set index (CI)	`0x`
Cache tag (CT)	`0x`
Cache hit? (Y/N)	
Cache byte returned	`0x`