CAS CS 210 - Computer Systems

Fall 2014

PROBLEM SET 3 (PS3) (CACHING AND VIRTUAL MEMORY)
OUT: NOV 13
DUE: NOV 25, 1:30 PM

NO LATE SUBMISSIONS WILL BE ACCEPTED

Problem 1

The following problem concerns basic cache lookups.

- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not 4-byte words).
- Physical addresses are 13 bits wide.
- The cache is 2-way set associative, with a 4 byte line size and 16 total lines.

In the following tables, **all numbers are given in hexadecimal**. The contents of the cache are as follows:

2-way Set Associative Cache												
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	99	04	03	48
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	2F	81	FD	09	0B	0	8F	E2	05	BD
3	06	0	3D	94	9B	F7	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	0	B0	39	D3	F7
6	91	1	A0	B7	26	2D	F0	0	0C	71	40	10
7	46	0	B1	0A	32	0F	DE	1	12	C0	88	37

Part 1

The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

CO The block offset within the cache line
CI The cache index
CT The cache tag

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

Part 2

For the given physical address, 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”.

Physical address: `0E34`

A. Physical address format (one bit per box)

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

B. Physical memory reference

Parameter	Value
Byte offset	0x
Cache Index	0x
Cache Tag	0x
Cache Hit? (Y/N)	
Cache Byte returned	0x

Problem 3

After watching the presidential election you decide to start a business in developing software for electronic voting. The software will run on a machine with a 1024-byte direct-mapped data cache with 64 byte blocks.

You are implementing a prototype of your software that assumes that there are 7 candidates. The C-structures you are using are:

```
struct vote {
    int candidates[7];
    int valid;
};

struct vote vote_array[16][16];
register int i, j, k;
```

You have to decide between two alternative implementations of the routine that initializes the array `vote_array`. You want to choose the one with the better cache performance.

You can assume:

- `sizeof(int) = 4`
- `vote_array` begins at memory address 0
- The cache is initially empty.
- The only memory accesses are to the entries of the array `vote_array`. Variables `i`, `j` and `k` are stored in registers.

A. What percentage of the writes in the following code will miss in the cache?

```
for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        vote_array[i][j].valid=0;
    }
}

for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        for (k=0; k<7; k++) {
            vote_array[i][j].candidates[k] = 0;
        }
    }
}
```

Total number of misses in the first loop: ________

Total number of misses in the second loop: ________

Overall miss rate for writes to `vote_array`: ________

B. What percentage of the writes in the following code will miss in the cache?

```
for (i=0; i<16; i++){
    for (j=0; j<16; j++) {
        for (k=0; k<7; k++) {
            vote_array[i][j].candidates[k] = 0;
        }
        vote_array[i][j].valid=0;
    }
}
```

Miss rate for writes to `vote_array`: ________

Problem 4

Problem 6.31, on page 634, from our CS:APP2e

Problem 5

Problem 6.37 on page 636, from our CS:APP2e

1 Problem 6

Problem 9.11, on page 849, from our CS:APP2e

2 Problem 7

Problem 9.12, on page 850, from our CS:APP2e

3 Problem 8

Problem 9.13, on page 851, from our CS:APP2e

Additional Practice Problems : NOT GRADED

1. Problem 6.32, on page 634, from our CS:APP2e
2. Problem 6.33 on page 635, from our CS:APP2e
3. Problem 6.39 on page 637, from our CS:APP2e
4. Problem 9.15, on page 851, from CS:APP2e
5. Problem 9.16, on page 852, from CS:APP2e