CAS CS 210 - Computer Systems

Fall 2014

PROBLEM SET 3 (PS3) SOLUTIONS

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

Answer:

`CT: [12-5]  CI: [4-2]  CO: [1-0]`

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

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

Answer:

A. Physical address format (one bit per box)

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

B. Physical memory reference

Parameter	Value
Byte offset	0x0
Cache Index	0x5
Cache Tag	0x71
Cache Hit? (Y/N)	Y
Cache Byte returned	0x0B

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;
        }
    }
}
```

Answer:
Total number of misses in the first loop: `128 Misses / 256 Total Writes`

Total number of misses in the second loop: `128 Misses / 1792 Total Writes`

Overall miss rate for writes to `vote_array`: `256 / 2048 = 1/8 = 12.5`

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;
    }
}
```

Answer: Miss rate for writes to `vote_array`: `1/16=6.26`

Problem 4

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

A. Cache size: $C = 128$ bytes.

B. Address fields: CT: [12-5] CI: [4-2] CO: [1-0]

Problem 5

Problem 6.37 on page 636, from our CS:APP2e

A. Case 1: Assume the cache is 512-bytes, direct-mapped, with 16-byte cache blocks. What is the miss rate? In this case, each access to `x[1][i]` conflicts with previous access to `x[0][i]`, so the miss rate is 100%.

B. Case 2: What is the miss rate if we double the cache size to 1024 bytes? If we double the cache size, then the entire array fits in the cache, so the only misses are the cold (compulsary) misses for each new block. Since each block holds four array items, the miss rate is 25%.

C. Case 3: Now assume the cache is 512 bytes, 2-way set associative using an LRU replacement policy, with 16-byte cache blocks. What is the cache miss rate? Increasing the associativity removes the conflict misses that occurred in the direct mapped cache of Case 1. The only misses are the cold misses when each block is loaded, so the miss rate is 25%.

D. For Case 3, will a larger cache size help to reduce the miss rate? No. Even if the cache were infinitely large, we would still have the compulsary misses required to load each new cache block.

E. For Case 3, will a larger block size help to reduce the miss rate? Yes. A larger block size would reduce the number of compulsary misses by an amount inversely proportional to the increase. For example, if we doubled the block size, we decrease the miss rate by half.

Problem 6

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

```
A. 00 0010 0111 1100

B.      VPN:              0x9
        TLBI:             0x1
        TLBT:             0x2
        TLB hit?          N
        page fault?       N
        PPN:              0x17

C.      0101 1111 1100

D.      CO:               0x0
        CI:               0xf
        CT:               0x17
        cache hit?        N
        cache byte?       -
```

Problem 7

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

```
A. 00 0011 1010 1001

B.     VPN:            0xe
       TLBI:           0x2
       TLBT:           0x3
       TLB hit?        N
       page fault?     N
       PPN:            0x11

C.     0100 0110 1001

D.     CO:             0x1
       CI:             0xa
       CT:             0x11
       cache hit?      N
       cache byte?     -
```

Problem 8

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

```
A. 00 0000 0100 0000

B.      VPN:            0x1
        TLBI:           0x1
        TLBT:           0x0
        TLB hit?        N
        page fault?     Y
        PPN:            -

C.      n/a

D.      n/a
```

Additional Practice Problems

1. Problem 6.32, on page 634, from our CS:APP2e

Address `0x071A`

A. Address format (one bit per box):

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

B. Memory reference:

Parameter	Value
Block Offset (CO)	0x2
Index (CI)	0x6
Cache Tag (CT)	0x38
Cache Hit? (Y/N)	Y
Cache Byte returned	0xEB

2. Problem 6.33 on page 635, from our CS:APP2e

Address `0x16E8`

A. Address format (one bit per box):

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

B. Memory reference:

Parameter	Value
Block Offset (CO)	0x0
Index (CI)	0x2
Cache Tag (CT)	0xB7
Cache Hit? (Y/N)	N
Cache Byte returned	–

3. Problem 6.39 on page 637, from our CS:APP2e

In this problem, each cache line holds two 16-byte `point_color` structures. The `square` array is $256 \times 16 = 4096$ bytes and the cache is 2048 bytes, so the cache can only hold half of the array. Since the code employs a row-wise stride-1 reference pattern, the miss pattern for each cache line is a miss, followed by 7 hits.

A. What is the total number of writes? 1024 writes.

B. What is the total number of writes that miss in the cache? 128 misses.

C. What is the miss rate? $128/1024 = 12.5\%$.

4. Problem 9.15 on page 851, from our CS:APP2e

Request	Block size (decimal bytes)	Block header (hex)
`malloc(3)`	8	`0x9`
`malloc(11)`	16	`0x11`
`malloc(20)`	24	`0x19`
`malloc(21)`	32	`0x21`

5. Problem 9.16 on page 852, from our CS:APP2e

Alignment	Allocated block	Free block	Minimum block size (bytes)
Single-word	Header and footer	Header and footer	20
Single-word	Header, but no footer	Header and footer	16
Double-word	Header and footer	Header and footer	24
Double-word	Header, but no footer	Header and footer	16