CAS CS 210 - Computer Systems

Fall 2016

PROBLEM SET 3 (PS3) (CACHING AND VIRTUAL MEMORY)
OUT: NOV 15
DUE: DEC 6, 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
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: `0A35`

A. Physical address format (one bit per box)

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

B. Physical memory reference

Parameter	Value
Byte offset	0x 01
Cache Index	0x 05
Cache Tag	0x 51
Cache Hit? (Y/N)	N
Cache Byte returned	0x —

Problem 3

C = S × E × B

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.

C E B

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

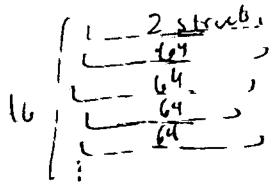
```
struct vote {
    int candidates[7];
    int valid;
};

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

8 ints
32 bytes

256

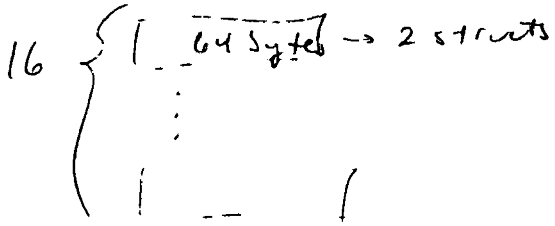
1024 = S × 1 × 64

S = 16



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.

[illegible] = S(1)16



2 structs are pulled
into cache at a time

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?

256

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

every other = hit 50%

1792

```
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;
        }
    }
}
```

every other = hit vote

$\frac{1}{2}(256) + \frac{1}{2}(256)$ = 2048 accesses

$\frac{256}{2048} =$

Total number of misses in the first loop: 128

Total number of misses in the second loop: 128

Overall miss rate for writes to `vote_array`: 12.5%

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

1792

```
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;
    }
}
```

256

miss 50% of 256

Miss rate for writes to vote_array: 6.25%

2048 reads

pull in 2 @ a time

$$\frac{\frac{1}{2}(256)}{2048} = .0625$$

PS 3 - Book Problems

Vidya Akavoor
11/21/16

4) 6.30 p.652

A. The size of the cache is $\boxed{128 \text{ bytes}}$ → 8 sets · 4 lines · 4 bytes

B.

CT	CT	CT	CT	CT	CT	CT	CT	CI	CI	CI	CO	CO

5) 6.37 p. 655

function	N=64	N=60
sumA	25%	25%
sumB	100%	25%
sumC	50%	25%

6) 9.11 p. 877

VA → 0x027c

A.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	1	1	1	1	1	0	0

B.

Parameter	Value
VPN	0x9
TLB index	0x1
TLB tag	0x2
TLB hit?	N
Page fault?	N
PPN	0x17

C.

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

D.

Parameter	Value
Byte offset	0x0
Cache index	0xF
Cache tag	0x17
Cache hit?	N
Byte returned	—

7) 9.12 p. 877
VA → 0x03a9

A.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	1	0	1	0	0	1

B.

Parameter	Value
VPN	0xE
TLB index	0x2
TLB tag	0x3
TLB hit?	N
Page fault?	N
PPN	0x11

C.

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

D.

Parameter	Value
Byte Offset	0x1
Cache index	0xa
Cache tag	0x11
Cache hit?	N
Byte returned	—

8) 9.13 p. 877
VA → 0x0040

A.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0

B.

Parameter	Value
VPN	0x1
TLB index	0x1
TLB tag	0x0
TLB hit?	N
Page fault?	Y
PPN	—

C and D do not need to be done; page fault!