Lab 14

CS 224

Practice Problem 6.12 (solution page 663)

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

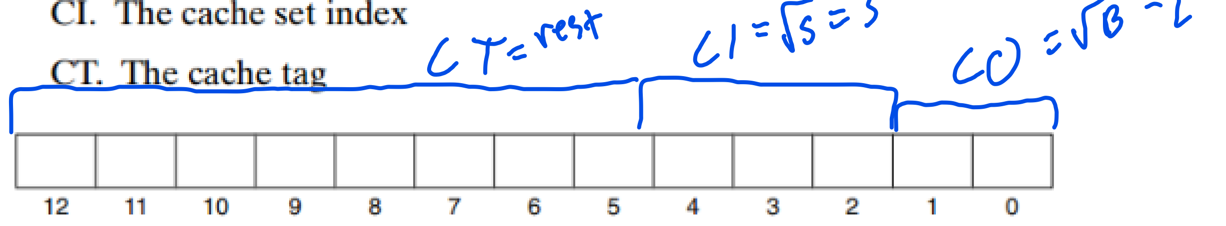
	Line 0						Line 1					
Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	09	1	86	30	3F	10	00	0	—	—	—	—
1	45	1	60	4F	E0	23	38	1	00	BC	0B	37
2	EB	0	—	—	—	—	0B	0	—	—	—	—
3	06	0	—	—	—	—	32	1	12	08	7B	AD
4	C7	1	06	78	07	C5	05	1	40	67	C2	3B
5	71	1	0B	DE	18	4B	6E	0	—	—	—	—
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 (1 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



Practice Problem 6.13 (solution page 664)

Suppose a program running on the machine in Problem 6.12 references the 1-byte word at address 0x0E34. Indicate the cache entry accessed and the cache byte value returned in hexadecimal notation. Indicate whether a cache miss occurs. If there is a cache miss, enter "—" for "Cache byte returned."

A. Address format (1 bit per box):

$$\underbrace{0\ 1\ 1\ 1\ 0\ 0\ 0\ 1}_{CT}\ \underbrace{1\ 0\ 1}_{CI}\ \underbrace{0\ 0}_{CO}$$

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

B. Memory reference:

Parameter	Value
Cache block offset (CO)	0x 0
Cache set index (CI)	0x 5
Cache tag (CT)	0x 71
Cache hit? (Y/N)	Y
Cache byte returned	0x B

0	E	3	4
0000	1110	0011	0100

000 01110001 101 00

CT
0x 71

CI
0x 5

CO
0x 0

0 D D 5

0000 1101 1101 0101

0110 1110 101 01

Practice Problem 6.14 (solution page 664)

Repeat Problem 6.13 for memory address 0x0DD5.

A. Address format (1 bit per box):

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

B. Memory reference:

Parameter	Value
Cache block offset (CO)	0x 1
Cache set index (CI)	0x 5
Cache tag (CT)	0x 6E
Cache hit? (Y/N)	N
Cache byte returned	0x —

Practice Problem 6.18 (solution page 666)

The heart of the recent hit game *SimAquarium* is a tight loop that calculates the average position of 256 algae. You are evaluating its cache performance on a machine with a 1,024-byte direct-mapped data cache with 16-byte blocks ($B = 16$). You are given the following definitions:

```
struct algae_position {
    int x;
    int y;
};

struct algae_position grid[16][16];
int total_x = 0, total_y = 0;
int i, j;
```

You should also assume the following:

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

Determine the cache performance for the following code:

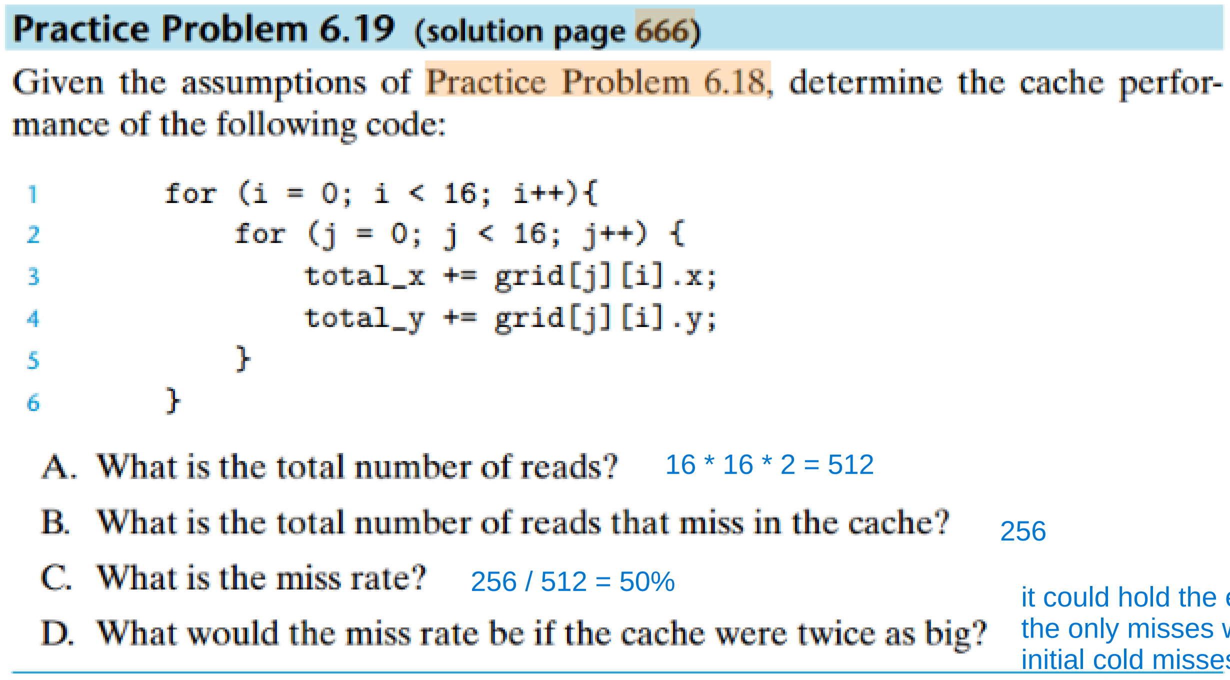
```
for (i = 0; i < 16; i++) {
    for (j = 0; j < 16; j++) {
        total_x += grid[i][j].x;
    }
}

for (i = 0; i < 16; i++) {
    for (j = 0; j < 16; j++) {
        total_y += grid[i][j].y;
    }
}
```

A. What is the total number of reads? 16 * 16 * 2 = 512

B. What is the total number of reads that miss in the cache? every other read hits, so 256

C. What is the miss rate? 256 / 512 = 50%



Practice Problem 6.19 (solution page 666)

Given the assumptions of Practice Problem 6.18, determine the cache performance of the following code:

```
for (i = 0; i < 16; i++){
    for (j = 0; j < 16; j++) {
        total_x += grid[j][i].x;
        total_y += grid[j][i].y;
    }
}
```

A. What is the total number of reads? 16 * 16 * 2 = 512

B. What is the total number of reads that miss in the cache? 256

C. What is the miss rate? 256 / 512 = 50%

D. What would the miss rate be if the cache were twice as big? it could hold the entire grid, so the only misses would be the initial cold misses, which would be 1/4 = 25%

6.26 ◆

The following table gives the parameters for a number of different caches. Your task is to fill in the missing fields in the table. Recall that m is the number of physical address bits, C is the cache size (number of data bytes), B is the block size in bytes, E is the associativity, S is the number of cache sets, t is the number of tag bits, s is the number of set index bits, and b is the number of block offset bits.

Cache	m	C	B	E	S	t	s	b
1.	32	2048	8	1	256	21	8	3
2.	32	2,048	4	4	128	23	7	2
3.	32	1,024	2	8	64	25	6	1
4.	32	1,024	32	2	16	23	4	5

6.29 ◆◆

Suppose we have a system with the following properties:

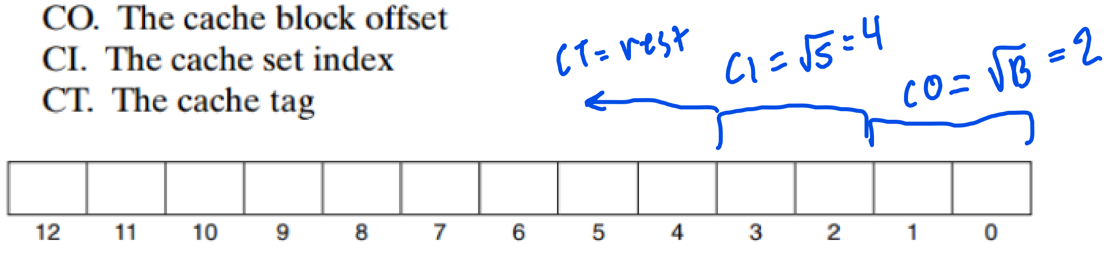
- The memory is byte addressable.
- Memory accesses are to 1-byte words (not to 4-byte words).
- Addresses are 12 bits wide.
- The cache is two-way set associative ($E = 2$), with a 4-byte block size ($B = 4$) and four sets ($S = 4$).

The contents of the cache are as follows, with all addresses, tags, and values given in hexadecimal notation:

Set index	Tag	Valid	Byte 0	Byte 1	Byte 2	Byte 3
0	00	1	40	41	42	43
	83	1	FE	97	CC	D0
1	00	1	44	45	46	47
	83	0	—	—	—	—
2	00	1	48	49	4A	4B
	40	0	—	—	—	—
3	FF	1	9A	C0	03	FF
	00	0	—	—	—	—

A. The following diagram shows the format of an address (1 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



B. For each of the following memory accesses, indicate if it will be a cache hit or miss when *carried out in sequence* as listed. Also give the value of a read if it can be inferred from the information in the cache.

Operation	Address	Hit?	Read value (or unknown)
Read	0x834	______	________
Write	0x836	______	________
Read	0xFFD	______	________

6.36 ◆◆

This problem tests your ability to predict the cache behavior of C code. You are given the following code to analyze:

```
int x[2][128];
int i;
int sum = 0;

for (i = 0; i < 128; i++) {
    sum += x[0][i] * x[1][i];
}
```

Assume we execute this under the following conditions:

- `sizeof(int)` = 4.
- Array `x` begins at memory address `0x0` and is stored in row-major order.
- In each case below, the cache is initially empty.
- The only memory accesses are to the entries of the array `x`. All other variables are stored in registers.

Given these assumptions, estimate the miss rates for the following cases:

A. Case 1: Assume the cache is 512 bytes, direct-mapped, with 16-byte cache blocks. What is the miss rate?

B. Case 2: What is the miss rate if we double the cache size to 1,024 bytes?

C. Case 3: Now assume the cache is 512 bytes, two-way set associative using an LRU replacement policy, with 16-byte cache blocks. What is the cache miss rate?

D. For case 3, will a larger cache size help to reduce the miss rate? Why or why not?

E. For case 3, will a larger block size help to reduce the miss rate? Why or why not?