CSED211 컴퓨터SW시스템개론

Homwork #3

20180551
컴퓨터공학과
이준석

1. Exercise 6.24 on page 685

6.24 ◆◆

Suppose that a 2 MB file consisting of 512-byte logical blocks is stored on a disk drive with the following characteristics:

Parameter	Value
Rotational rate	18,000 RPM
$T_{avg\ seek}$	8 ms
Average number of sectors/track	2,000
Surfaces	4
Sector size	512 bytes

For each case below, suppose that a program reads the logical blocks of the file sequentially, one after the other, and that the time to position the head over the first block is $T_{avg\ seek} + T_{avg\ rotation}$.

A. *Best case:* Estimate the optimal time (in ms) required to read the file given the best possible mapping of logical blocks to disk sectors (i.e., sequential).

B. *Random case:* Estimate the time (in ms) required to read the file if blocks are mapped randomly to disk sectors.

[A] 2MB의 파일이므로 512-byte인 block이 4000개 필요하다. average number of sectors/track이 2000이므로 총 2번의 full rotation이 필요하다. 그리고 best case이므로 일단 head의 위치가 결정되면 seek을 위해 다시 head를 움직일 필요가 없다.

즉, 최종적으로 소모되는 시간은 $T_{avg\ seek} + T_{avg\ rotation} + T_{one\ rotation} \times 2$ 이다.

$T_{avg\ seek} = 8$ ms

$T_{full\ rotation} = 1(min)/18000 \times 60(sec)/1(min) \times 1000(ms)/1(sec) = 3.33$ ms

$T_{avg\ rotation} = T_{full\ rotation} / 2 = 1.67$ ms

즉, 8 + 1.67 + 3.33 x 2 = **16.33 ms**

[B] block들이 disk sector에 대해 무작위로 배치되어 있다면 모든 sector에 대한 탐색과 head의 이동이 필요하다. 즉, 소요되는 총 시간은 다음과 같다.

$4000 \times (T_{avg\ seek} + T_{avg\ rotation} + T_{full\ rotation} / 2000) =$ **38.673333 ms** ≈ 39 sec

2. Exercise 6.29 on page 687

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:

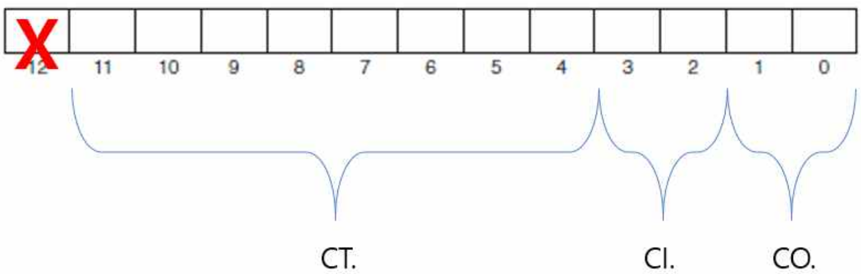
CO. The cache block offset
CI. The cache set index
CT. The cache tag

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

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	____	____

[A]



[B]

read / 0x834 -> miss / unknown

write / 0x836 -> hit / write operation이다. ‘read’ value와 무관하다.

read / 0xFFD -> hit / c0

3. Exercise 6.30 on page 688

6.30 ◆

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 13 bits wide.
- The cache is 4-way set associative ($E = 4$), with a 4-byte block size ($B = 4$) and eight sets ($S = 8$).

Consider the following cache state. All addresses, tags, and values are given in hexadecimal format. The Index column contains the set index for each set of four lines. The Tag columns contain the tag value for each line. The V columns contain the valid bit for each line. The Bytes 0–3 columns contain the data for each line, numbered left to right starting with byte 0 on the left.

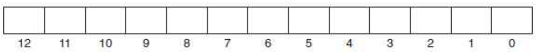
4-way set associative cache

Index	Tag	V	Bytes 0–3	Tag	V	Bytes 0–3	Tag	V	Bytes 0–3	Tag	V	Bytes 0–3
0	F0	1	ED 32 0A A2	8A	1	BF 80 1D FC	14	1	EF 09 86 2A	BC	0	25 44 6F 1A
1	BC	0	03 3E CD 38	A0	0	16 7B ED 5A	BC	1	8E 4C DF 18	E4	1	FB B7 12 02
2	BC	1	54 9E 1E FA	B6	1	DC 81 B2 14	00	0	B6 1F 7B 44	74	0	10 F5 B8 2E
3	BE	0	2F 7E 3D A8	C0	1	27 95 A4 74	C4	0	07 11 6B D8	BC	0	C7 B7 AF C2
4	7E	1	32 21 1C 2C	8A	1	22 C2 DC 34	BC	1	BA DD 37 D8	DC	0	E7 A2 39 BA
5	98	0	A9 76 2B EE	54	0	BC 91 D5 92	98	1	80 BA 9B F6	BC	1	48 16 81 0A
6	38	0	5D 4D F7 DA	BC	1	69 C2 8C 74	8A	1	A8 CE 7F DA	38	1	FA 93 EB 48
7	8A	1	04 2A 32 6A	9E	0	B1 86 56 0E	CC	1	96 30 47 F2	BC	1	F8 1D 42 30

A. What is the size (C) of this cache in bytes?

B. The box that follows 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:

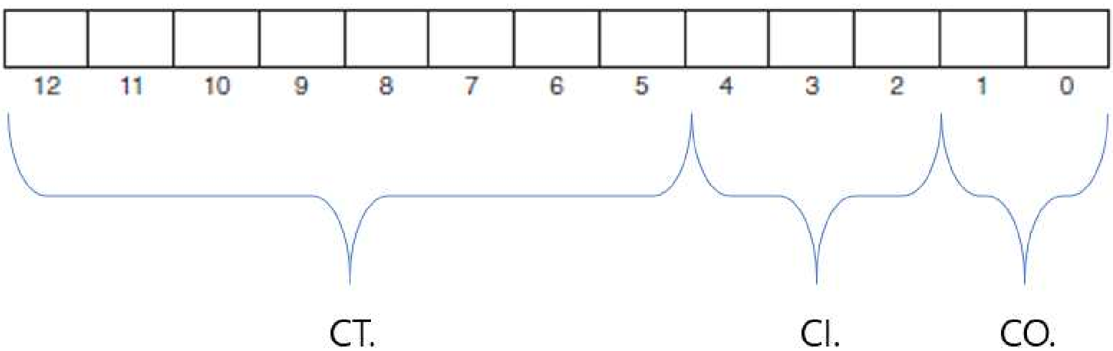
CO. The cache block offset
CI. The cache set index
CT. The cache tag



[A]

size (C) = S x E x B = 8 x 4 x 4(bytes) = 128(bytes)

[B]



4. Exercise 6.36 on page 690

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.

miss rate = misses의 수 / references의 수

[A]

miss rate = 256/256 = 1

512bytes인 direct-mapped cache이므로, x[0][i]와 x[1][i]는 set이 같다.

그래서 x[0][i]와 x[1][i]를 번갈아가면서 참조하면 계속 cold miss와 conflict miss가 발생한다. 따라서 전부 miss이다.

[B]

miss rate = 64/256 = 1/4

1024byte이므로 A번 문항에서 발생한, set이 겹치는 문제가 발생하지 않는다. 4개 중의 1개 꼴로 miss

[C]

miss rate = 1/4

set 하나에 line이 2개가 있어서 A번 문항에서 발생한 문제가 발생하지 않는다. 그래서 4개 중의 하나 꼴로 miss가 발생한다.

[D]

miss rate를 줄이지 못한다. block size가 그대로여서 한 번의 miss가 발생했을 때 cache에 미리 저장되는 것이 그대로이기 때문이다. 그래서 cache size가 늘어나도 miss rate는 그대로 1/4일 것이다.

[E]

miss rate를 줄일 수 있다. 예를 들어 block size가 2배가 된다면 miss rate는 1/8로 줄어들 것이다. 이렇게 miss rate가 감소하는 이유는 한 번의 miss가 발생했을 때 cache에 미리 저장되는 것들이 늘어나기 때문이다.

5. Exercise 7.6 on page750

Symbol	swap.o .symtab entry?	Symbol type	Module where defined	Section
buf	o	extern	m.o	.data
bufp0	o	global	swap.o	.data
bufp1	o	local	swap.o	.bss
swap	o	global	swap.o	.text
temp	x			
incr	o	local	swap.o	.text
count	o	local	swap.o	.bss

6. Exercise 7.8 on page751

7.8 ◆

In this problem, let REF(x.i) → DEF(x.k) denote that the linker will associate an arbitrary reference to symbol x in module i to the definition of x in module k. For each example below, use this notation to indicate how the linker would resolve references to the multiply-defined symbol in each module. If there is a link-time error (rule 1), write "ERROR". If the linker arbitrarily chooses one of the definitions (rule 3), write "UNKNOWN".

A.

```
/* Module 1 */              /* Module 2 */
int main()                  static int main=1[
{                           int p2()
}                           {
                            }
```

(a) REF(main.1) → DEF(_____._____)

(b) REF(main.2) → DEF(_____._____)

B.

```
/* Module 1 */              /* Module 2 */
int x;                      double x;
void main()                 int p2()
{                           {
}                           }
```

(a) REF(x.1) → DEF(_____._____)

(b) REF(x.2) → DEF(_____._____)

C.

```
/* Module 1 */              /* Module 2 */
int x=1;                    double x=1.0;
void main()                 int p2()
{                           {
}                           }
```

(a) REF(x.1) → DEF(_____._____)

(b) REF(x.2) → DEF(_____._____)

[A]

(a) REF(main.1) -> DEF(main.1)

(b) REF(main.2) -> DEF(main2.)

[B]

(a) unknown

(b) unknown

[C]

(a) error

(b) error

7. Exercise 7.12 on page 753

7.12 ◆◆

Consider the call to function swap in object file m.o (Problem 7.6).

```
9:   e8 00 00 00 00          callq  e <main+0xe>      swap()
```

with the following relocation entry:

```
r.offset = 0xa
r.symbol = swap
r.type   = R_X86_64_PC32
r.addend = -4
```

A. Suppose that the linker relocates .text in m.o to address 0x4004e0 and swap to address 0x4004f8. Then what is the value of the relocated reference to swap in the callq instruction?

B. Suppose that the linker relocates .text in m.o to address 0x4004d0 and swap to address 0x400500. Then what is the value of the relocated reference to swap in the callq instruction?

```
foreach section s {
    foreach relocation entry r {
        refptr = s + r.offset;  /* ptr to reference to be relocated */

        /* Relocate a PC-relative reference */
        if (r.type == R_X86_64_PC32) {
            refaddr = ADDR(s) + r.offset; /* ref's run-time address */
            *refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr);
        }

        /* Relocate an absolute reference */
        if (r.type == R_X86_64_32)
            *refptr = (unsigned) (ADDR(r.symbol) + r.addend);
    }
}
```

Figure 7.10 **Relocation algorithm.**

[A]

ADDr(s) = ADDr(.text) = 0x4004e0

ADDr(r.symbol) = ADDr(swap) = 0x4004f8

refaddr = ADDr(s) + r.offset = 0x4004e0 + 0xa = 0x4004ea

*refptr = (unsigned) (ADDr(r.symbol) + r.addend – refaddr)

= (unsigned) (0x4004f8 + (-4) - 0x4004ea) = (unsigned) 0xa

4004e9: e8 0a 00 00 00   callq 4004f8 <swap>

[B]

ADDr(s) = ADDr(.text) = 0x4004d0

ADDr(r.symbol) = ADDr(swap) = 0x400500

refaddr = ADDr(s) + r.offset = 0x4004d0 + 0xa = 0x4004da

*refptr = (unsigned) (ADDr(r.symbol) + r.addend – refaddr)

= (unsigned) (0x400500 + (-4) - 0x4004da) = (unsigned) 0x22

```
4004d9: e8 22 00 00 00     callq 400500 <swap>
```

8. Exercise 8.13 on page 825.

8.13 ◆
What is one possible output of the following program?

code/ecf/global-forkprob3.c

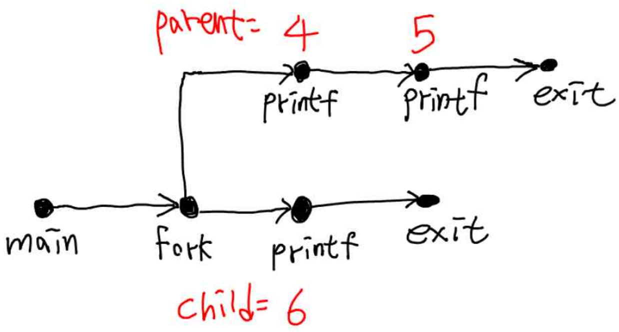
```
#include "csapp.h"

int main()
{
    int a = 5;

    if (Fork() != 0)
        printf("a=%d\n", --a);

    printf("a=%d\n", ++a);
    exit(0);
}
```

code/ecf/global-forkprob3.c



6

4

5

혹은

4

5

6

등등 여러 가지가 가능하다.

하나의 가능한 output을 고르라고 하였으므로

[정답]

6

4

5

9. Exercise 8.18 on page 827

8.18 ◆◆

Consider the following program:

code/ecf/forkprob2.c

```
#include "csapp.h"

void end(void)
{
    printf("2"); fflush(stdout);
}

int main()
{
    if (Fork() == 0)
        atexit(end);
    if (Fork() == 0) {
        printf("0"); fflush(stdout);
    }
    else {
        printf("1"); fflush(stdout);
    }
    exit(0);
}
```

code/ecf/forkprob2.c

Determine which of the following outputs are possible. *Note:* The `atexit` function takes a pointer to a function and adds it to a list of functions (initially empty) that will be called when the `exit` function is called.

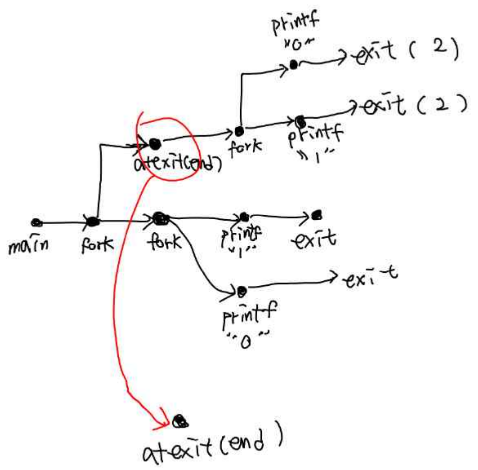
A. 112002

B. 211020

C. 102120

D. 122001

E. 100212



output으로 여러 가지가 가능하다.

A, C, E는 가능하지만 B와 C는 불가능하다. B는 2가 제일 먼저 나올 수 없어서 불가능하고 D는 12까지는 가능하지만 바로 2가 나오는 것이 불가능하므로 나올 수 없는 output이다.

[정답]

A, C, E

10. Exercise 8.21 on page 828

8.21 ◆◆
What are the possible output sequences from the following program?

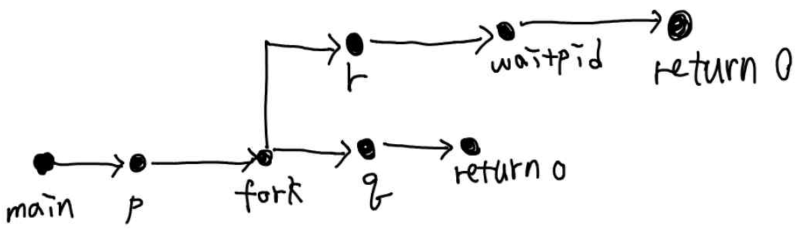
code/ecf/global-waitprob3.c

```
int main()
{
    printf("p"); fflush(stdout);
```



```
    if (fork() != 0) {
        printf("q"); fflush(stdout);
        return 0;
    }
    else {
        printf("r"); fflush(stdout);
        waitpid(-1, NULL, 0);
    }
    return 0;
}
```

code/ecf/global-waitprob3.c



pqr

prq

총 2가지가 가능하다.