**CSED211 Homework #4**

20210643 Hyun June Kim

**1. Exercise 6.26 on page 686.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cache | m | C | B | E | S | t | s | b |
| 1. | 32 | **2048** | 8 | 1 | **256** | 21 | 8 | 3 |
| 2. | 32 | 2048 | **4** | **4** | 128 | 23 | 7 | 2 |
| 3. | 32 | 1024 | 2 | 8 | 64 | **25** | **6** | 1 |
| 4. | 32 | 1024 | **32** | 2 | 16 | 23 | 4 | **5** |

1행: 이므로 이다.

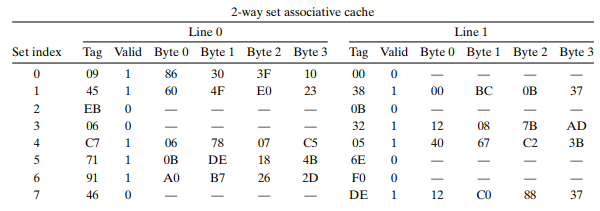
2행: 에서 , 에서 ,

3행: 에서 , 이므로

4행: 이므로 , 에서

**2. Exercise 6.28 on page 686.**

Problem 6.12.



Block offset: 2, Set index: 3, Tag: 8

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CI | CO | CO |

**A. List all of the hex memory addresses that will hit in set 2.**

Set index 2에서는 line0과 1 모두 valid bit이 0이다. 따라서 Hit인 memory addresses가 없다.

**B. List all of the hex memory addresses that will hit in set 4.**

Set4이므로 set bit가 100이다.

Tag가 C7(11000111), 05(00000101)이므로 그리고 line0, 1 모두 valid가 1이다.

따라서 11000111100xx, 00000101100xx 꼴은 모두 HIT이다.

답: 0x18F0, 0x18F1, 0x18F2, 0x18F3, 0x00B0, 0x00B1, 0x00B2, 0x00B3

**C. List all of the hex memory addresses that will hit in set 5.**

Set5이므로 set bit가 101이고, line 0만 valid이다.

Tag가 71(01110001)이므로, 결론적으로 01110001101xx꼴은 HIT이다.

답: 0x0E34, 0x0E35, 0x0E36, 0x0E37

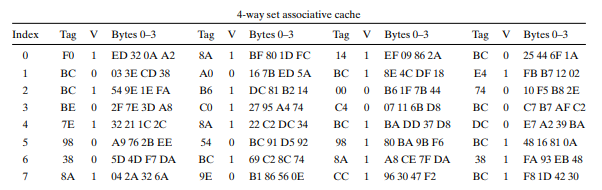
**D. List all of the hex memory addresses that will hit in set 7**

Set7이므로 set bit는 111이고, line1만 valid이다.

Tag가 DE(11011110)이므로, 11011110111xx꼴은 모두 HIT이다.

답: 0x1BDC, 0x1BDD, 0x1BDE, 0x1BDF

**3. Exercise 6.31 ~ 6.32 on pages 688 ~ 689.**



8개 set이므로 CI는 3bit, 1byte Tag이므로 CT 8bit, Block size 4이므로 CO 2bit이다.

**6.31)** Address 0x071A는 11100011010

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Parameter | | | | Value | | | |
| Block offset(CO) | | | | 0x**02** | | | |
| Index(CI) | | | | 0x**06** | | | |
| Cache tag(CT) | | | | 0x**38** | | | |
| Cache hit?(Y/N) | | | | **Y** | | | |
| Cache byte returned | | | | 0x**EB** | | | |

6번째 index에서 1번째 블록은 Tag가 일치하나 valid bit가 0이다. 하지만 4번째 블록에서 Tag가 같고 valid가 1이므로 HIT가 발생한다.

**6.32)** Address 0x16E8=1011011101000(2)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| Parameter | | | | Value | | | |
| Block offset(CO) | | | | 0x**00** | | | |
| Index(CI) | | | | 0x**02** | | | |
| Cache tag(CT) | | | | 0x**B7** | | | |
| Cache hit?(Y/N) | | | | N | | | |
| Cache byte returned | | | | **—** | | | |

Index 2에 0xB7 tag가 없으므로 miss

**4. Exercise 6.36 on page 690**

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

C=512, B=16, E=1, S=32인 경우이다. 총 256번 read할 텐데,

x[0][i]의 address: 4\*I, x[1][i]의 address: 4\*(128+i)-512+i4이다.

따라서 x[0][i]와 x[1][i]를 읽을 때 똑 같은 block으로 캐싱된다. No spatial locality.

즉, miss rate: 1.00=100%이다.

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

C=1024, B=16, E=1, S=64로, index가 두 배로 늘어나 cache size가 1024가 된 경우이다.

1024=4\*256이므로 전체 array가 캐시에 들어간다. 그리고 sizeof(int)=4 이다.

따라서 4/B=1/4의 miss rate을 가지게 될 것이다.

Miss rate: 1/4=0.25=25%이다.

**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?**

C=512, B=16, E=2, S=16으로 two way가 되고 index가 절반이 된 상황이다.

Total read count는 256번이고, x[0][i]의 address: 4\*I, x[1][i]의 address: 4\*(128+i)-512+i4이다.

그런데 two way이므로 x[0][i]와 x[1][i]는 다른 블록에 들어가게 된다.

따라서 i=0부터 i=63까지는 4/B=25%의 miss rate을 보일 것이다.

그런데 LRU replacement policy에 의해 가장 오래된 블록이 대체되면서

x[0][i]가 x[0][i-64], x[1][i]가 x[1][i-64]와 same block으로 대체되게 되므로,

그 이후에는 다시 25T%의 miss rate이 나올 것이다.

따라서 전체 miss rate: 0.25=25%

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

전체 Cache size를 올리더라도 여전히 B=16이라면 miss rate은 그대로일 것이다.

sizeof(int)=4이기 때문에 block이 16이면 계속해서 처음 miss와 세 번 hit이 반복되면서 4/16=25%의 miss rate이 나오기 때문이다.

따라서 larger cache size는 miss rate를 reduce해주지 않을 것이다.

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

Block size B를 올려준다면 miss rate를 낮출 수 있을 것이다. B=32라고 생각해보면, 8개의 int가 한 블록에 들어가게 되므로, 8번 중 1번만 miss가 나올 것이므로 12.5%의 miss rate이 나타난다.

따라서 block size를 늘리는 것은 miss rate를 낮추는 데 도움이 될 것이다.

**5. Exercise 6.41 on page 694**

Direct-mapped cache이므로 C=32\*1024, B=8, E=1, S=4096인 경우일 것이고,

sizeof(char)=1 이므로 pixel 구조체의 size는 4일 것이다. 따라서 한 struct가 한 block에 다 들어갈 수가 있다. 따라서, following code에서 buffer[i][j].r=0에서는 miss가 일어나고, buffer[i][j].g=0, buffer[i][j].b=0, buffer[i][j].a=0에서는 hit이 일어난다. 그러므로 miss rate =1/4=25%가 될 것이다.

따라서 hit rate는 0.75=75%이다.

**6. Practice Problem 7.5 on page 731 when the linker relocates .text in m.o to address 0x4004d0 and swap to address 0x4003e8.**

Address 0x4004d0 and swap to address 0x4003e8

r.offest을 더하면, 0x4004d0+0xa=0x4004da이다. 그리고 r.addend = -4이므로,

0x4003e8-4-0x4004da=0xffffff0a가 된다.

따라서 value of the relocated reference to swap in the callq instruction: 0xffffff0a

**7. Exercise 7.8 on page 751**

**A.** module2에서 main은 static int로 선언하였다. 따라서 main을 local variable로 생각할 수가 있고, 따라서 충돌이 발생하지 않는다. 즉, 각 module마다 main이 유효하게 존재하므로, 답은

(a) REF(main.1) → DEF(main.1)

(b) REF(main.2) → DEF(main.2)

**B.** 이번에는 module들의 x에 두 개의 weak definition이 존재한다. (두 개의 weak symbol인 경우) 따라서 rule3의 경우이므로, UNKOWN이 된다.

(a) REF(x.1) → DEF(unknown)

(b) REF(x.2) → DEF(unknown)

**C.** C의 경우에는 두 module에서 두 개의 strong definition이 존재하는 경우이다. (두 개의 strong symbol인 경우) 따라서 rule1의 경우이므로 error가 답이 된다.

(a) REF(x.1) → DEF(error)

(b) REF(x.2) → DEF(error)

**8. Exercise 7.10 on page 752**

Symbol resolution을 하려면 먼저 참조된 값이 나중에 정의가 되어야만 한다. 따라서 아래와 같이 화살표 뒤에 위치한 것들을 뒤에 위치하도록 하여야 한다.

**A. p.o → libx.a → p.o**

답: gcc p.o libx.a

**B. p.o → libx.a → liby.a and liby.a → libx.a**

답: gcc p.o libx.a liby.a libx.a

**C. p.o → libx.a → liby.a → libz.a and liby.a → libx.a → libz.a**

답: gcc p.o libx.a liby.a libx.a libz.a

**9. Exercise 7.12 on page 753**

**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?**

swap+r.addend-(.text+r.offset)으로 계산하면,

0x4004f8-4-(0x4004e0+0xa)=0xa가 나온다. 따라서 답: 0xa

**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?**

A번과 같이 계산하면, 0x400500-4-(0x4004d0+0xa)=0x22이다. 따라서 답: 0x22