Homework3

20173155

김진권

1. a.

the latency of cache hit access time = 1 cycle,

the latency of cache miss access time = 105 cycles;

average memory access time = hit rate \* hit access time + (1-hit rate) \* miss access time

= 0.95\*1 + 0.05\*105 = 6.2

b. 문제를 간단하게 하기 위해서 data size와 cache block size가 동일하다고 가정한다.

따라서 cache block은 오직 1개의 data만 가질 수 있다.

시간이 어느 정도 흐른 후 average hit rate는 steady state에 도달하게 되고, cache는 64kb/data size만큼의 data을 가진다. 전체 data의 개수는 256MB/data size이기 때문에,

Average hit rate = (64kb/data size) / (256MB/data size) = 64KB/256MB = 1/2^12 = 0.0002

따라서 average memory access time = 0.0002 \* 1 + 0.9998 \* 105 = 104.9792이다.

c. b문제에서 average memory access time (104.9792)는 cache를 끈 상태의 main memory acces time (100)보다 큰 값이기 때문에, locality가 존재하지 않는 경우에는 cache가 없는 경우가 더 효율이 좋기 때문에, locality는 cache 존재 이유에 필수적이다.

d. miss rate의 bound을 구하기 위해서는 gain과 loss가 0이 되는 시점을 알면되고, 다음과 같은 식으로 구할 수 있다.

(1-miss rate)\*gain = miss rate \* loss

Miss rate = x

(1-x)\*99 = x \* 5

X = 99/104 = 0.95

따라서 miss rate는 0.95이다.

2. a 64 byte loop는 128byte fully associative cache에 완전하게 들어갈 수 있기 때문에 miss rate는 0이다.

b. LRU의 경우 loop의 경우 asymptotic하게 FIFO처럼 행동하기 때문에, 192 byte, 320 byte 경우 모두 miss rate가 100%가 될 것이다.

간단하게 예를 들어 다음과 같은 상황이 발생할 것이다.

메모리 사이즈가 4이고, cache 사이즈가 2라고 가정하면,

파랑색 영역은 cache안에 포함되는 것이다.

Address 3을 접근할 때 address 1,2영역이 cache에 포함되기 때문에 miss가 발생하고, address 1이 cache에서 spill되고 address 3이 cache에 insert된다. (첫번째 그림 ~ 두번째 그림)

Address 4을 접근할 때 address 2,3영역이 cache에 포함되기 때문에 miss가 발생하고 address 2이 cache에서 spill되고 address 4이 cache에 insert된다. (두번째 그림 ~ 세번째 그림)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |

c. MRU의 경우 64byte loop의 경우, LRU와 마찬가지로 cache에 모두 포함될 수 있기 때문에 miss rate는 0이 되고, 192byte. 320byte의 경우 asymptotic하게 LIFO처럼 동작하기 때문에, 각각 miss rate 가 1 - 31%48 = 0.35 , 1 – 31%80 = 0.61이 된다. 따라서 LRU보다 두 경우 모두 이득을 얻게 된다.

간단하게 예를 들어 다음과 같은 상황이 발생할 것이다.

메모리 사이즈가 4이고, cache 사이즈가 2라고 가정하면,

파랑색 영역은 cache안에 포함되는 것이다.

Address 3을 접근할 때 address 1,2영역이 cache에 포함되기 때문에 miss가 발생하고, address 2이 cache에서 spill되고 address 3이 cache에 insert된다. (첫번째 그림 ~ 두번째 그림)

Address 4을 접근할 때 address 1,3영역이 cache에 포함되기 때문에 miss가 발생하고 address 3이 cache에서 spill되고 address 4이 cache에 insert된다. (두번째 그림 ~ 세번째 그림)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |

따라서 첫번째 data는 cache에 계속적으로 머물게 되기 때문에 miss rate가 1 – 1/4가 된다.

d. random replacement가 문제에서 주어진 상황에서 LRU보다 성능이 좋을 것이다.

1) 64byte의 경우 : 기존 LRU도 miss rate가 0이기 때문에, 더 좋아질 수는 없지만 random replacement cache의 경우에도 data가 cache에 모두 포함되기 때문에 miss rate는 0이된다.

2) 192byte의 경우: asymptotic하게 miss rate가 1 – 32/48 = 0.33이기 때문에 LRU보다 성능이 좋다.

3) 320byte이 경우: asymptotic하게 miss rate가 1 – 32/80 = 0.6 이기 때문에 LRU보다 성능이 좋다.

3. a. cache inclusive 경우 : L1에 있는 data가 L2에 존재하는 cache 구조이다.

Case1.

L1 miss 1. Victim algorithm – if the block is dirty (Write back to L2)

↓2.request ↑3.reply .

L2 hit

Case2.

L1 miss 1. Victim algorithm – if the block is dirty (write back to L2)

↓2.request ↑6.reply (write to L1)

L2 miss 3. Victim algorithm – if the block is dirty (write back to main memory )

& victim block is in the L1 cache (invalidate that block in L1)

↓4.request ↑5.reply (write to L2)

Main memory (이 아래로는 고려 x)

b. cache exclusive 경우 : L1에 있는 data가 L2에 존재하지 않는 cache 구조이다.

case1.

L1 miss 1. Victim algorithm choose the victim block

↓2.send (victim block)&request ↑3.reply .

L2 hit (같은 공간에 mapping 되는 설계 구조만 고려. Victim block이 hit로 인해 선택된 block 자리에 insert된다.)

Case2.

L1 miss 1. Victim algorithm choose the victim block

↓2.send (victim block) & request ↑6.reply .

L2 miss 3. Victim algorithm choose the victim block (L1의 victim block

이 L2의 victim block자리에 insert된다.) if the victim block is dirty (write back to main memory )

↓4.send(victim block) & request ↑5.reply (pass through the L2 )

Main memory (이 아래로는 고려 x)

c.

L1 cache의 경우

inclusive cache의 경우, L1 cache에 insert 된 순간 clean 상태이다. (L1 cache에서의 dirty bit는 L2에 write 할지 안할지 결정하기 때문에) 따라서 , victim으로 선발되기 전까지 write가 일어날 확률에 대해서면 dirty probability가 결정되는 반면,

exclusive cache의 경우 L1 cache에 insert 된 순간에 clean일지, dirty일지는 그 block 상태에 따라 다르고, exclusive의 경우 victim block이 dirty에 상관없이 L2 block과 교환이 된다. 따라서, exclusive cache에서 단순히 dirty block이 victim으로 선발될 확률이 더 높다.