컴퓨터구조 HW 4

201932045 지동환

1

```
int sum
for (i = 0; i < 1024; i++)
{
  sum = 0;
  for (j = 0; j < 1024; j++)
    sum += a[i][j];
  b[i] = sum;
}
```

(a)

`i, j, sum`

고정된 값이 Loop 속에서 반복되면서 참조된다. Temporal Locality.

(b)

`a, b`

같은 row에 있는 Data들이 한번에 stored된다. a는 안쪽 for loop에서, b는 바깥 for loop에서 Spatial locality를 가진다.

(c)

Array a : 2^{20} ld instructions

Array b : 2^{10} sd instructions

(d)

Cold miss rate of A :

Array 각 element의 크기는 64-bit = 8 bytes.

한 Block size는 32 bytes, 즉 한 번에 4개의 element를 가져올 수 있다.

같은 row의 데이터가 연속되어 저장되어 있다 => `a[0][0]` 을 가져오면, `a[0][1], a[0][2], a[0][3]` 을 함께 가져온다.

그러므로, Cold miss rate는 $1/4 = 25\%$

Cold miss rate of B :

Array b도 마찬가지로, 4번에 한 번 Cold miss가 발생한다. 25%

(e)

Capacity miss rate of A :

32 bytes block이 128개가 있다. 한 block은 Array의 element 4개를 저장한다. 즉, Cache에 저장할 수 있는 공간의 최대는 512개의 element이다.

512번 `sum += a[i][j]` 가 실행되면 Cache의 용량은 가득찬다.

그렇지만, Capacity miss는 이미 Cache에서 삭제된 데이터에 접근해야 할 때 발생한다. 이런 경우가 발생하지 않는다.

그러므로, Capacity miss rate는 0% 이다.

Capacity miss rate of B :

`sum += a[i][j]` 가 돌고 나면, Cache에 저장되어 있던 b의 데이터는 삭제되어 있다.

그렇기 때문에 Capacity miss가 발생한다. 75%

2

(a)

한 Cache block에 2 word를 저장. 그렇게 16개의 block이 있다. => 총 32 word = $2^5 * 2^6$ bits = 2^8 bytes의 크기를 가진다.

(b)

Cache block은 16개이다. => Index가 4자리 필요.

한 block에 두 개의 word를 저장한다. => Offset 1자리 필요.

Word address	Binary address	Tag	Index	Hit/Miss
0xFD	1111 1101	111	1110	M
0xBA	1011 1010	101	1101	M
0x2C	0010 1100	001	0110	M
0xB5	1011 0101	101	1010	M
0x0E	0000 1110	000	0111	M
0xBE	1011 1110	101	1111	M
0x58	0101 1000	010	1100	M
0xBF	1011 1111	101	1111	H
0x02	0000 0010	000	0001	M
0x2B	0010 1011	001	0101	M
0xB4	1011 0100	101	1010	H
0x03	0000 0011	000	0001	H

(c)

$9/12 = 75\%$

(d)

AMAT = Hit time + Miss rate * Miss penalty

$= 1 + 0.75 \times (8 \times 2)$

$= 13$

13 cycles이 AMAT가 된다.

(e)

같은 Cache size를 가지고 있으므로, 2 word / block * 16 blocks = 4 word / block * 8 blocks이 된다.

즉, Cache block은 8개가 있다. => Index로 3 bits만 필요하다.

block size가 4 words가 되었다. => offset이 2 bits가 필요해진다.

Word address	Binary address	Tag	Index	Hit/Miss
0xFD	1111 1101	111	111	M
0xBA	1011 1010	101	110	M
0x2C	0010 1100	001	011	M
0xB5	1011 0101	101	101	M
0x0E	0000 1110	000	011	M
0xBE	1011 1110	101	111	M
0x58	0101 1000	010	110	M
0xBF	1011 1111	101	111	H
0x02	0000 0010	000	000	M
0x2B	0010 1011	001	010	M
0xB4	1011 0100	101	101	H
0x03	0000 0011	000	000	H

(f)

12번 중 3번 hit 했다.

Miss rate = $9/12 = 0.75$

(g)

AMAT = $1 + 0.75 \times 8 \times 4$

$= 25$

25 cycles이 된다.

3

Tag	Index	Offset
54	5	5

(a)

Offset이 5이므로, 한 block에는 2^5 bytes가 저장되어 있다.

2^5 bytes = 2^8 bits = 4 words가 된다.

(b)

Index가 5 bits이다.

그러므로, $2^5 = 32$ 개의 Block이 있다.

(c)

Cache 전체의 bits / Data storage bits 를 계산하자.

Cache 전체의 bits :

Cache Block 하나는,

Valid	Tag	Data
1 bit	54 bits	4×64 bits = 2^8 bits

로 구성되어 있다. 총 32 blocks이 존재한다.

그러므로, 전체 Cache의 bits 수는 $(1 + 54 + 2^8) \times 32 = 9952$ 개이다.

Data storage bits :

각 block마다 2^8 bits가 사용된다. $2^8 \times 32 = 8192$

둘을 나누면,

$$\frac{(1 + 54 + 2^8) \times 32}{2^8 \times 32} \simeq 1.21$$

이 된다.

(d)

먼저 byte-address를 Binary address로 옮기면,

Address	Binary Address
0x000	0000 0000 0000
0x004	0000 0000 0100
0x010	0000 0001 0000
0x084	0000 1000 0100
0x0E8	0000 1110 1000
0x0A0	0000 1010 0000
0x400	0100 0000 0000
0x01E	0000 0001 1110
0x08C	000 1000 1100
0xC1C	1100 0001 1100
0x0B4	0000 1011 0100
0x884	1000 1000 0100

이다.

Address	Tag	Index	Offset	Hit/Miss	Replace
0x000	0	0x00	0x00	M	
0x004	0	0x00	0x04	H	
0x010	0	0x00	0x10	H	
0x084	0	0x04	0x04	M	
0x0E8	0	0x07	0x08	M	
0x0A0	0	0x05	0x00	M	
0x400	1	0x00	0x00	M	0x000 ~ 0x01F
0x01E	0	0x00	0x1E	M	0x400 ~ 0x41F
0x08C	0	0x04	0x0C	H	
0xC1C	3	0x00	0x1C	M	0x000 ~ 0x01F
0x0B4	0	0x05	0x14	H	
0x884	2	0x04	0x04	M	0x080 ~ 0x09F

(e)

$4/12 \simeq 33.33\%$ 이다.

(f)

<4, 2, Mem[0x880] - Mem[0x89F]>

<5, 0, Mem[0x0A0] - Mem[0x0BF]>

<0, 3, Mem[0xC00] - Mem[0xC1F]>

<7, 0, Mem[0x0E0] - Mem[0x0FF]>

4

Processor	L1 Size	L1 miss rate	L1 hit time
P1	2 KB	8%	0.5ns
P2	4 KB	4%	0.8ns

(a)

L1 hit time은 한 Clock Cycle을 의미한다. Clock Rates는 이것의 역수.

P1 : 2GHz

P2 : 1.25GHz

(b)

AMAT = Hit time + Miss rate * Miss Penalty

= L1 Hit time + L1 Miss rate * L1 Miss Penalty

P1 : L1 Miss Penalty = $50ns/0.5ns = 100$ clocks

$1 + 0.08 \times 100 = 9$ clocks.

P2 : L1 Miss Penalty = $50ns/0.8ns = 62.5$ clocks

$1 + 0.04 \times 62.5 = 3.5$ clocks.

(c)

Instruction Cache와 Data Cache의 load/save에서 발생하는 모든 경우를 고려해야 한다.

CPI = Base CPI + L1 Miss rate * L1 Miss Penalty + DMEM ld/sd rate * L1 Miss rate * L1 Miss Penalty

P1 CPI :

50ns = 100 clock cycles

$1.0 + 0.08 \times 100 + 0.36(0.08 \times 100) = 11.88$

P2 CPI :

50ns = 62.5 clock cycles

$1.0 + 0.04 \times 62.5 + 0.36(0.04 \times 62.5) = 4.4$

P2가 더 빠르다.

(d)

AMAT = Hit time + Miss rate * Miss Penalty

= L1 Hit time + L1 Miss rate * (L2 Hit time + L2 Miss rate * L2 Miss Penalty)

L2 Hit time = 5.0ns = 10 clocks

P1 : $1 + 0.08[10 + 0.8 \times 100] = 8.2$ clocks.

성능이 증가했다.

(e)

CPI =

Base CPI + L1 Miss rate * (L2 Hit time + L2 Miss rate * L2 Miss Penalty) + DMEM ld/sd rate * L1 Miss rate * (L2 Hit time + L2 Miss rate * L2 Miss Penalty)

$1.0 + 0.08 \times (10 + 0.8 \times 100) + 0.36 \times 0.08 \times (10 + 0.8 \times 100) = 10.792$

5

Data reads per 1000 instructions	Data writes per 1000 instructions	Instruction cache miss rate	Data cache miss rate	Block size (bytes)
250	100	0.3%	2.0%	64

(a)

Instruction Cache :

CPI가 2이면 평균적으로 cycle 당 0.5개의 instruction이 실행된다.

0.3%의 Instruction이 Miss한다. 각각의 Miss마다 한 block을 가져온다.

0.5 instruction/cycle * 0.003 * 64 bytes = 0.096 bytes/cycle

Data Cache read :

250/1000 = 25%의 instruction이 read를 요청한다.

그 중, 2.0%가 Miss한다.

한번 Miss하면 64 bytes인 1개의 block을 가져온다.

0.5 instruction/cycle * 0.25 * 0.02 * 64 bytes = 0.16 bytes/cycle

Data Cache write :

100/1000 = 10%의 instruction이 write를 요청한다.

Write-through를 사용하기 때문에, **모든** write마다 8 bytes의 write가 발생한다.

0.5 instruction/cycle * 0.10 * 8 bytes = 0.4 bytes/cycle

Data Cache write miss :

write 중 2.0%에서 Miss가 발생한다.

Write miss가 발생하면, Write-allocation을 사용하기 때문에 DRAM에서 Read를 해 와야 한다.

0.5 instruction/cycle * 0.10 * 0.02 * 64 bytes = 0.064 bytes/cycle

RAM read bandwidth = 0.096 + 0.16 + 0.064 = 0.32 bytes/cycle

RAM write bandwidth = 0.4 bytes/cycle

(b)

Data Cache read :

Read 정책에는 변화가 없다. 0.32 bytes/cycle

Data Cache :

Write-back의 경우, 모든 write 마다 RAM write를 해야 하는 Write-through와 달리, miss case에만 Write를 해 준다.

그렇지만, write, read instructions 두 경우 모두에서 Cache Miss가 발생할 수 있다.

0.5 instruction/cycle * (0.25 + 0.10) * 0.02 * 0.30 * 64bytes = 0.0672 bytes/cycle

RAM read bandwidth = 0.32 bytes/cycle

RAM write bandwidth = 0.0672 bytes/cycle

6

(a)

Page size가 4KB이다. => Virtual Address의 Page offset이 12자리이다.

Virtual Page Number	Page Offset
36-bits [47:12]	12-bits [11:0]

48-bits address로 표현할 수 있는 최대의 메모리 크기는, 2^{48} bytes = 2^{8} TiB = 256 TiB 이다.

(b)

Physical memory는 4GB의 크기를 가지고 있다. 4GB = $2^2 \times 2^{30}$ bytes

그러므로, Physical address는 32 자리이다.

Physical Page Number	Page Offset
20-bits [31:12]	12-bits [11:0]

(c)

TLB는 two-way set associative를 사용한다.

한 TLB에는 512/2 = 2^8개의 entries가 있다. => TLB의 index는 8자리이다.

나머지는 Tag로 사용된다.

Virtual Page Number	Page Offset
28-bits [47:20] + 8-bits [19:12]	12-bits [11:0]

으로 사용된다.
