컴퓨터 구조 3번째 과제

2019040164 정지오

[Review Questions]

<4.5>

List: tag, line, word

1. word: identify a unique word or byte within a block of main memory.

2. line: identify one of the lines of the cache.

3. tag: identify one of the blocks that can fit into that line.

Line field and tag field specify one of the blocks of main memory.

<4.6>

List: tag, word

1. tag: uniquely identify a block of main memory.

2. word: identify a unique word or byte within a block of main memory.

<4.7>

List: tag, set, word

1. word: identify a unique word or byte within block of main memory.

2. set: identify one of the sets of the cache.

3. tag: identify one of the blocks that can fit into that set.

[Problems]

<4.1>

Cache consists of 64 lines, divided into 4-line sets -> number of sets = 64/4 = 16

→ cache is divided into 16(2^4) sets.

Therefore, 4 bits are needed to identify set number.

Main memory consists of 4K blocks. (=2^12 blocks) (1K = 2^10)

Therefore, tag + set lengths must be 12 bits. Set length is 4. So, tag length is 12.

Each block contains 128(2^7) words.

Therefore, 7 bits are needed to specify the word field.

→ tag length = 8, set length = 4, word length = 7

Tag	Set	Word
8	4	7

<4.2>

Cache- lines of 16 bytes, total size of 8 kbytes

1K = 2^10 → 1 kbyte = 2^10 bytes = 1024 bytes → 8 kbytes = 8192 bytes

There are total of 8 kbyte / 16 bytes = 8192 bytes / 16 bytes = 512 lines in the cache.

Because of it is two-way set, the cache consists of 256 sets of 2 lines each.

256 sets = 2^8 sets → therefore 8 bits are needed to identify the set number.

For the 64-Mbyte main memory, a 26-bit address is needed.

Main memory consists of 64-Mbyte/16 bytes = 2^22 blocks.

22-8 = 14 = tag length

Tag	Set	Word
14	8	4

<4.3>

1. address = 111111 (H), 0001 0001 0001 0001 0001 0001 (B)

 a) tag = 11 (H), Line = 0444 (H), Word = 1 (H)

 b) tag = 044444 (H), Word = 1 (H)

 c) tag = 022 (H), Set = 0444 (H), Word = 1 (H)

2. address = 666666 (H), 0110 0110 0110 0110 0110 0110 (B)

 a) tag = 66 (H), line = 1999 (H), word = 2 (H)

 b) tag = 199999 (H), word = 2 (H)

 c) tag = 0CC (H), set = 1999 (H), word = 2 (H)

3. address = BBBBBB (H), 1011 1011 1011 1011 1011 1011 (B)

 a) tag = BB (H), line = 2EEE (H), word = 3 (H)

 b) tag = 2EEEEE (H), word = 3 (H)

 c) tag = 177 (H), set = 0EEE (H), word = 3 (H)

<4.4>

a) Address length: 24, number of addressable units: 2^24, block size: 4,
 number of blocks in main memory: 2^22, number of lines in cache: 2^14, size of tag: 8.

b) Address length: 24, number of addressable units: 2^24, block size: 4,
 number of blocks in main memory: 2^22, number of lines in cache: ?, size of tag: 22.

c) Address length: 24, number of addressable units: 2^24, block size: 4,
 number of blocks in main memory: 2^22, number of lines in set: 2, number of sets: 2^13,
 number of lines in cache: 2^14, size of tag: 9.

<4.5>

16Kb cache size, 4 way set associative

Line size = 4 x 32 bit = 16 bytes/

Line # = 2^10, set # = 2^8 → set = 8 bits, word = 2 bits, tag = 22 bits

Address ABCDE8F8 = 1010 1011 1100 1101 1110 1000 1111 1000 (B)

→ <1010 1011 1100 1101 1110 10><00 1111 11><00>

→ mapped to set number 62 in cache.

<4.12>

Main memory: 1MB, Word: 1 byte, Block: 16 bytes, Cache = 64KB

Cache line = memory block, cache slot = 16 bytes

Cache size 64 kB → 64 kB / 16 = 4096 cache slot → need 12 bits

a. 16 word per block -> 4 bits needed(2^4)

MM size = 1MB = 2^10KB = 2^20

Tag: 4 bit, Cache slot = 12 bit, word = 4 bit

a. 1) F0010 – tag: 1111, cache = 0000 0000 0001, word = 0000

2) 01234 – tag: 0000, cache = 0001 0010 0011, word = 0100

3) CABBE – tag: 1100, cache = 1010 1011 1011, word = 1110

b. different tag, same cache(line) → 1111 1111 1111

1) word = 1111, line = 1111 1111 1111, tag = 0000, address = 0FFFF

2) word = 0001, line = 1111 1111 1111, tag = 0011, address = 3FFF1

c. F0010 – word = 0h, tag = F001h

CABBE – word = Eh, tag = CABBh

d. 1) F0010 – word = 0000 (0), cache = 000 0000 0001 (001), tag = 11110 -> 1 1110 (1E)

2) CABBE – word = 1110 (E), cache = 010 1011 1011 (2BB), tag = 11001 -> 1 1001 (19)

<4.13>

LRU – least recently used

So, Choose the one you haven't used for the longest time.

It is easy to use software, but hard to use hardware.

Pointer points one of the cache, if accessed, move next and delete pointer's point.

<4.21>

a. 2.5 + 50 + 15*5 + 2.5 = 130 ns

b. line size 128 bytes -> num of word = 32

before: (0.95)(2.5) + (0.05)(130) = 8.875

after: (0.97)(2.5) + ((0.05)(210) = 8.725

➔ Reduce average memory access time

<4.22>

1) word in cache = 20 ns
2) word in MM = 60 + 20 = 80 ns
3) word in Disk = 12 ms + 80 ns = 12,000,080 ns

→ average time = (0.9)(20) + (0.1){(0.6)(80) + (0.4)(12,000,080)} = 480026 ns.