存储

优化

计算位数

1. By convention, a cache is named according to the amount of data it contains (i.e., a 4 KB cache can hold 4 KB of data); however, caches also require SRAM to store metadata such as tags, valid bits, dirty bits. Assume that the caches are byte addressable, and that addresses and words are 32 bits. Calculate the total number of bits required to implement a 16 KB direct-mapped cache with 4-word blocks.

$$Offset: \log_2 16(4-word=16B)=4$$
$$index: \log_2 1K=10bits$$
$$tag: 32-4-10=18bits$$
$$total: (18(tag)+1(valid)+1(dirty))\times 1024+16KB=151552bits$$

2. A memory and cache subsystem uses byte-addressing,32-bit address. Each row of the table below indicates a kind of cache,the "Cache feature" column of table describes the total size of cache data(not containing tag and valid bit),cache kind, cache block size,A 32-bit memory address 0x22339AB contains 3 fields: tag,index,byte offset,some of these field's size(bit number of this field)or value should be filled into table blank below, You should use Hex-decimal number to fill into column "Index value (Hex)",Total cache size containing data block, tag and valid
   bit should also be filled into column "Total Size,kB"(kB means unit is KB-1024 byte,not byte),only number(not expression)is allowed to be filled into this column.

   0x22339AB=0010 0010 0011 0011 1001 1010 1011

cache feature	index value	index size,bits	tag size,bits	total size ,KB
64KB data,direct-mapped,8 bytes block	0x0735	13	16	81
512KB data,direct-mapped,64 bytes block	0xce6	13	13	526
512KB data,2-Way set associative,8 bytes block	0x6735	15	14	632
1024KB data,8-Way set associative,32 bytes block	0x9cd	12	15	1536

- 第二行，--->512K/64=8K blocks --> index 13 bits , 512 KB direct mapped ---> tag 32-19=13 bits, index value : （中间13位）011 0011 1001 10 = 0xce6 ， total size = 512KB+(tag 13+valid 1)b*8K/8 = 526 KB
- 第三行，--->512K/2/8=32K blocks --> index 15 bits , 512 KB direct mapped ---> tag 32-15-3=14 bits, index value : （中间15位）0110 0111 0011 0101= 0x6735 ， total size = 512KB+(tag 14+valid 1)b*32K/8 *2（set-associative） = 632 KB
- 第四行，1024K/8/32=4K blocks , 12 bits index,tag = 32-12-5=15, 1 0011 1001 101=0x9cd, totally 1024KB+8* 4K * (15+1)=1536KB

优化效果

1. 提高block size可以显著增加 D

   A. cache miss B. Compulsory miss rate C. hit time D. miss penalty

A 有减少有增加，因为可能总大小不变，造成如脏块等问题

B 减少了第一次找不到的概率，因为一次拿进来的多了

C 轻微增加了，但是一般考试我们不认为增加了

D 一次要换的更多了

2. 提高 cache 的组相联度可以提高(improve) B
   A. capacity miss
   B. hit time
   C. conflict miss
   D. compulsory miss

A：认为不变

B: 组相联高了自然难找到了

C：减少了

D：认为不影响

cache 访问

过程模拟

1. Assume we have a direct-mapped byte-addressed cache with capacity 32B and block size of 8B.
   1. Of the 32 bits in each address, which bits do we use to find the tag, index, and offset of the cache?

$$offet : \log_2 8 = 3$$
$$index : \log_2 \frac{32}{1 \times 8} = 2$$
$$tag : 32 - 2 - 3$$

tag	index	offset
31:5	4:3	2:0

2. Classify each of the following byte memory accesses as a cache hit, cache miss, or cache miss with replacement

> 此类题目先正确计算 tag index 和 offset 才行

address	tag	index	offset	Hit/Miss/Replace	标记
0x00000004	0	0	4	M	1
0x00000005	0	0	5	H（1取入 index=1）	2
0x00000068	3	1	0	M	3
0x000000c8	6	1	0	R（换3 index=1 tag不是3）	4
0x00000068	3	1	0	R（换4 index=1 tag 不是 6）	5
0x000000dd	6	3	5	M（index=3）	6
0x00000045	2	0	5	R（换2 index=0 tag不是0）	7
0x000000cf	6	1	7	R（换5 index=1 tag不是3）	8
0x000000f3	7	2	3	M	9
0x000000db	6	3	3	H （6 时进去 index=3 tag=6）	10
0x0000009c	4	3	4	R （换10 index=3 tag不是6）	11
0x00000157	10	3	7	R（换11 index=3 tag不是4）	12

address	tag	index	offset	Hit/Miss/Replace	标记
0x00000fe9	127	1	1	R（换8 index=1 tag不是6）	13

-

综合应用

1. It assumes that the address of main memory is 32-bit and is addressed by byte. 8- way set-associative mapping is used between instruction cache and data cache and main memory, and write through strategy is used. The data capacity of the cache is 32KB and the block size is 64B

    1. How many bits is the tag of each cache line? Does it contain dirty bits?

        - Offset: 6bits index: 32KB÷64B÷8 = 2^6 -> 6bits
        - Tag: 32–6–6 = 20bits
        - 因为采用 write-through 策略，不需要在 cache 中设置脏位

    2. Here is a C code:

        ```
        for (k = 0; k < 1024; k++)
            s[k] = 2 * s[k];
        ```

        If both array s and variable k are int, int accounts for 4B, variable k is allocated in registers, and the start address of array s in main memory is 0x008000C0, what are the number of data cache misses accessing array s during the execution of this program segment?

        一个 block 能取入 $64B/4B = 16$ 个int，且 0x008000C0 最后 6 位都是0，所以刚好从一个块的开始开始读取，所以轮回的出现

        - 读取一个，miss，取 16 个块
        - 15 个 hit

        因此一共miss $\frac{1024}{16} = 64$ 次

    3. If the first access operation of the CPU is to read the instruction in the main memory unit 0x0001003, briefly explain the process of accessing the instruction from the cache, including the cache miss handling process.

        根据地址算出对应的 index，再由 index 访问对应的 cache line，若 valid 位无效或 valid 有效但 tag 与 cache 中的不相等则发生 read miss，需要到内存中取出对应地址所在 block 的所有数据并写回内 存