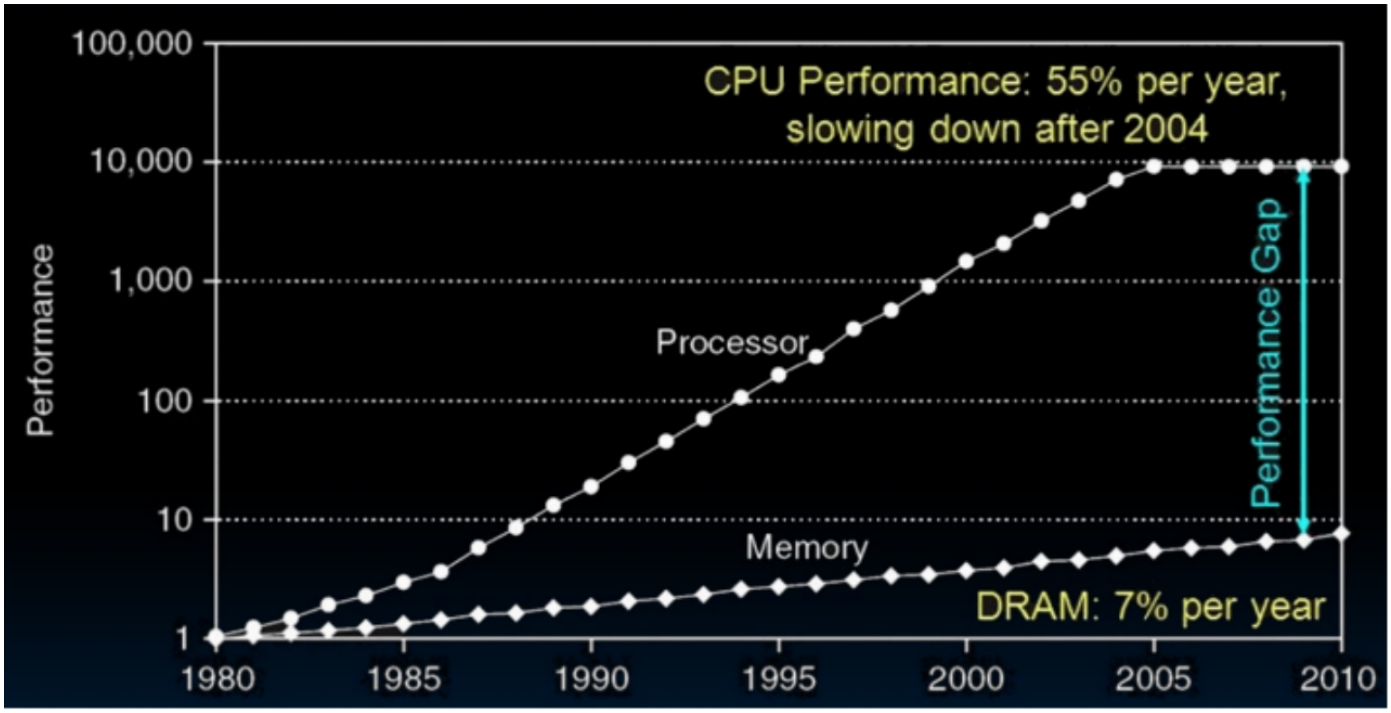
Memory Hierarchy and Direct-mapped Cache

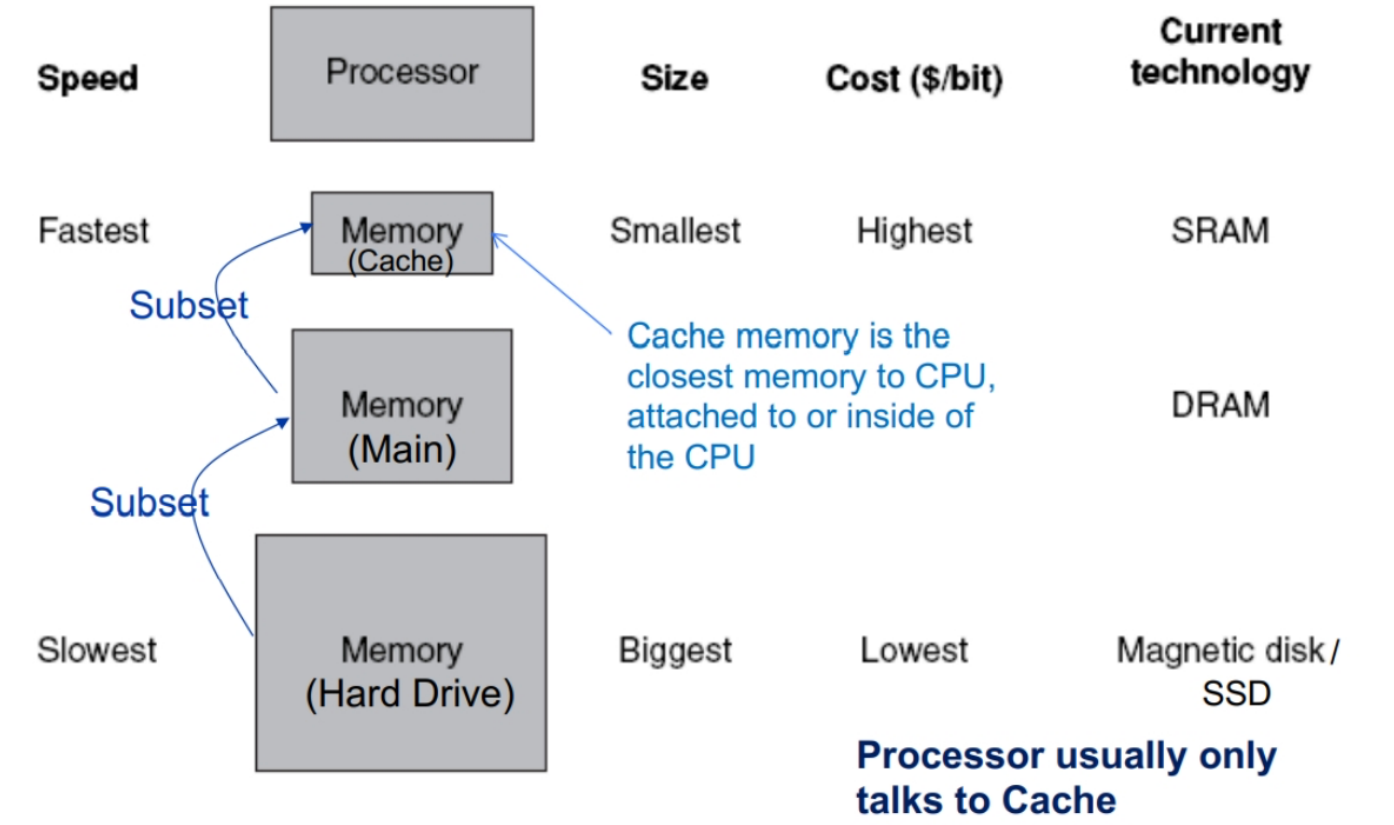
Liyang Xu Final RC Part 2

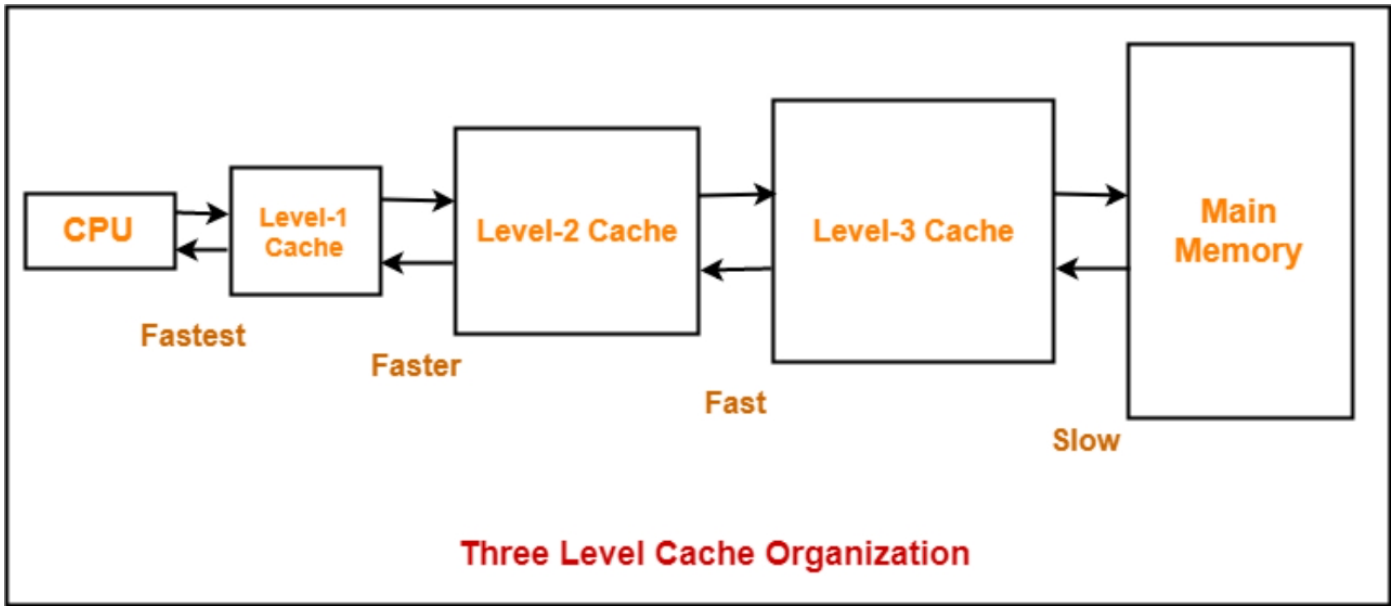
Memory Hierarchy

Why do we need cache?



Can't find memory technology that is affordable in GByte and also cycle in GHz(time and space trade-off). We do not need as much memory as the whole address space at a time when computing.





Direct-mapped Cache

Principles of Locality

Temporal locality: If at one point a particular memory location is referenced, then it is likely that *the same location* will be referenced again *in the near future*. e.g., instructions in a loop.

Spatial locality: If a particular storage location is referenced at a particular time, then it is likely that *nearby memory* locations will be referenced in the near future. e.g., sequential instruction access, array data.

In this case it is common to attempt to guess the size and shape of the area around the current reference for which it is worthwhile to prepare faster access for subsequent reference.

Memory locality (or data locality): Spatial locality explicitly relating to memory.

1. (10 points) The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 32-bit integer.

```
for (I=0; I<8; I++)
    for (J=0; J<8000; J++)
      A[I][J]=B[I][0]+A[J][I];
```

   (1) Which variable references exhibit temporal locality? (5 points)

   (2) Which variable references exhibit spatial locality? (5 points)

4 Concerns for cache design:

Q1: Where can a block be placed in cache? (Block placement)

Q2: How is a block found if it is in cache? (Block identification)

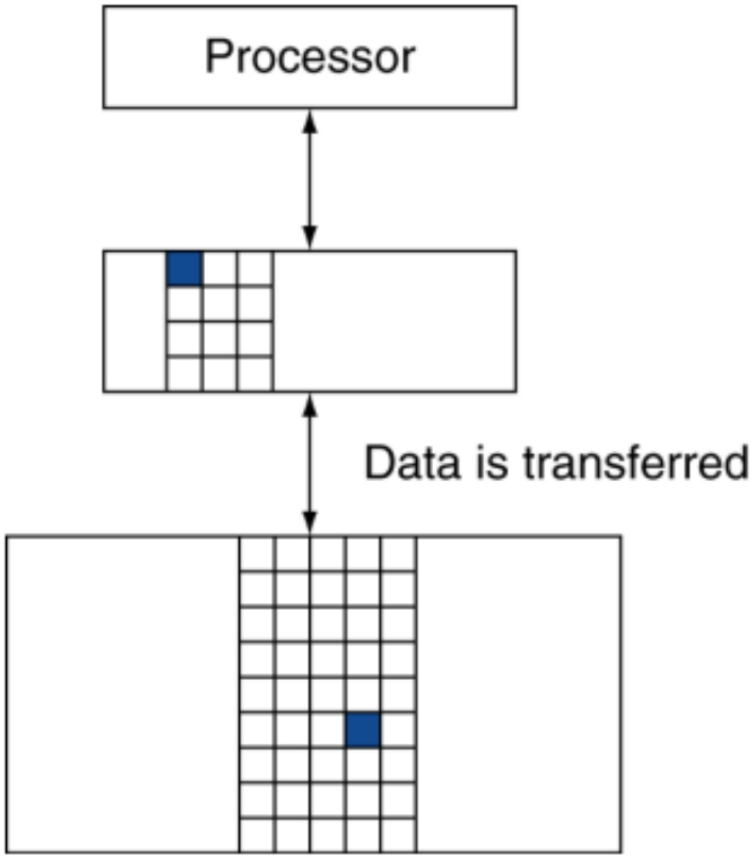
Q3: Which block should be replaced on a miss? (Block replacement)

Q4: What happens on a write? (Write strategy)

Concepts in Cache

- Block
  - Block address
  - Byte address
  - Word address
- Hit / Miss
  - Hit rate
  - Hit time
  - Miss rate
  - Miss penalty

Block (aka line): Unit of data referencing to take advantage of temporal and spatial locality. May be one or multiple words Block also has an address.



- Byte address vs. word address (assume 16-bit address)
  - Example: one word has 4 bytes

Byte offset

Word number (word address)

Byte Address		Contents
1111_1111_0000_00	00	Byte 1
1111_1111_0000_00	01	Byte 2
1111_1111_0000_00	10	Byte 3
1111_1111_0000_00	11	Byte 4

14 bits 2 bits

- Word address vs. block address (block number)
  - Example: one block has 2 words (8 bytes)

Word offset

Block number (Block address)

Word Number		Contents
1111_1111_0000_0	0	Word 1
1111_1111_0000_0	1	Word 2

13 bits 1 bit

Tag	Index	Offset
31 - 10	9 - 5	4 - 0

(1) What is the cache block size (in words)? (5 points)

(2) How many blocks does the cache have? (5 points)

(3) What is the ratio between total bits required for such a cache implementation over the data storage bits? (5 points)

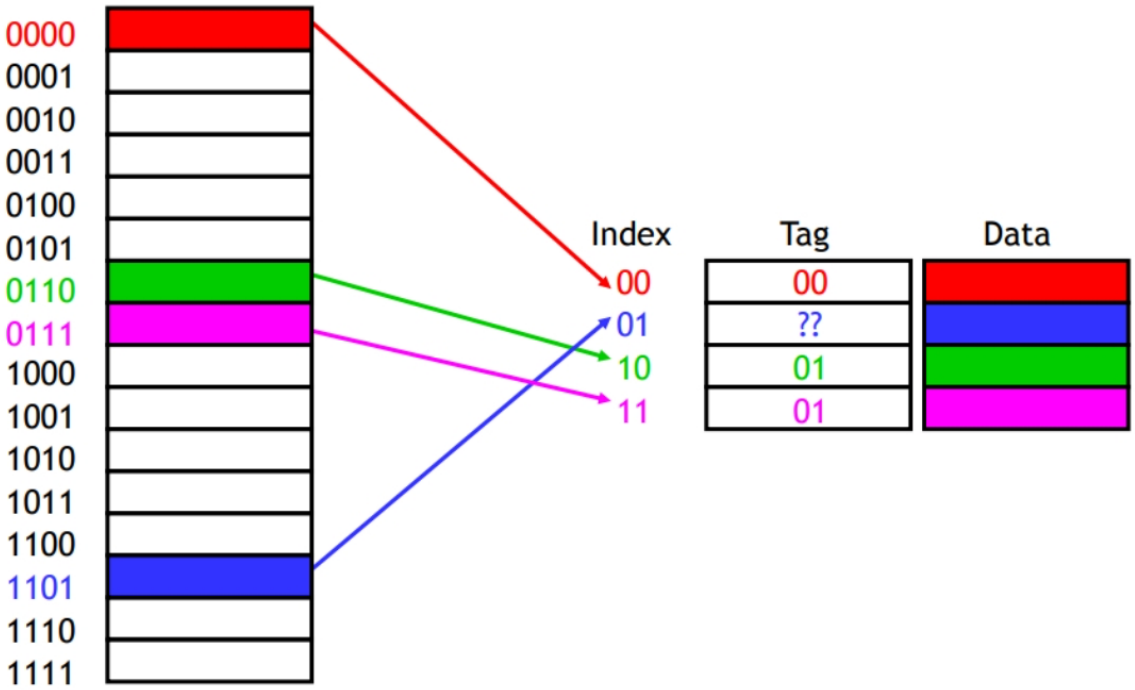
- **Hit**: If accessed data is present in upper level, access satisfied by upper level
- Hit rate: hits/accesses
- Hit time: time to access a memory including:
    - Time to determine whether a hit or miss
    - Time to pass block to requestor
- **Miss**: If accessed data is absent n Block copied from lower to higher level. Then accessed data supplied from upper level.
- Time taken: miss penalty
- Miss rate: misses/accesses = 1 – hit rate
- Miss (time) penalty Time to fetch a block from lower level upon a miss, including
    - Time to access the block
    - Time to transfer it between levels
    - Time to overwrite the higher level block
    - Time to pass block to requestor

Direct mapped cache: each (main) memory location corresponds to one choice in cache

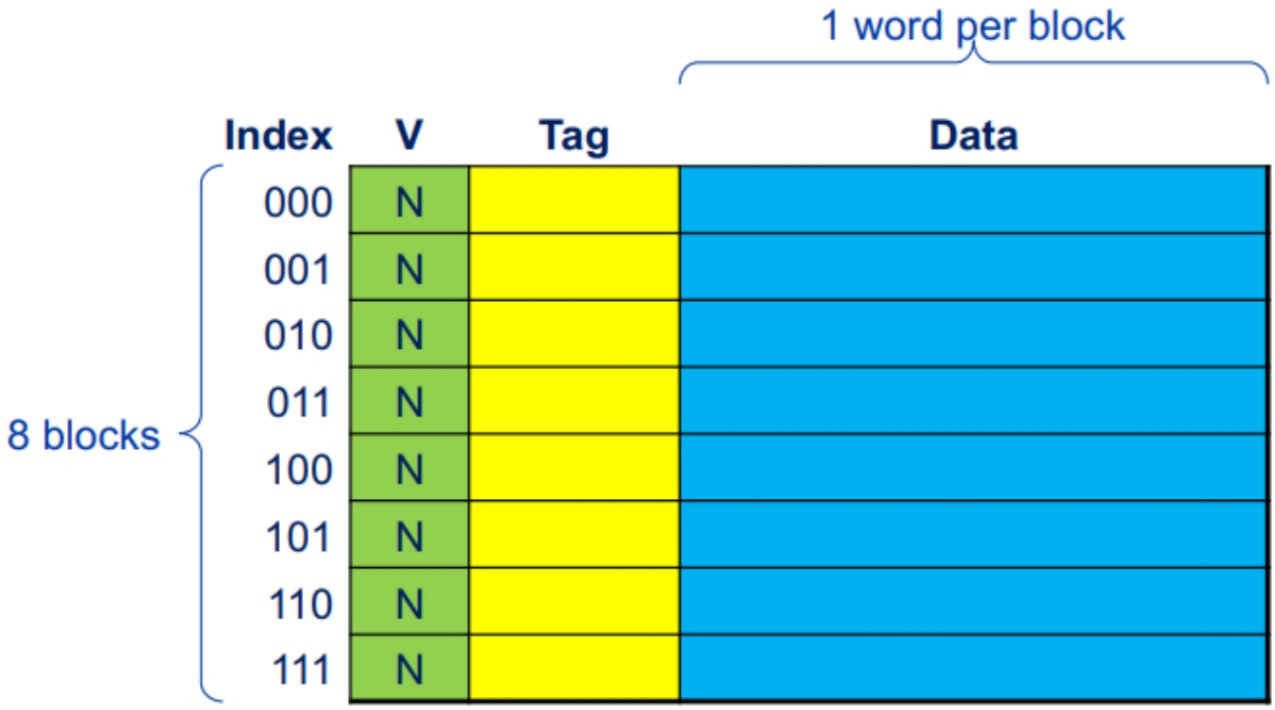
How to map?

Location of a block in cache is determined by address of the requested word

- Given word address, we can get block address (block number)
- Cache location (or cache block index) = (Block address in memory) % (Number of blocks in cache) = lower log2(Number of blocks in cache) bits of block address
- Multiple memory locations correspond to one block in cache: n-to-1 mapping



We need to add tags to the cache, which supply the rest of the address bits to let us distinguish between different memory locations that map to the same cache block.

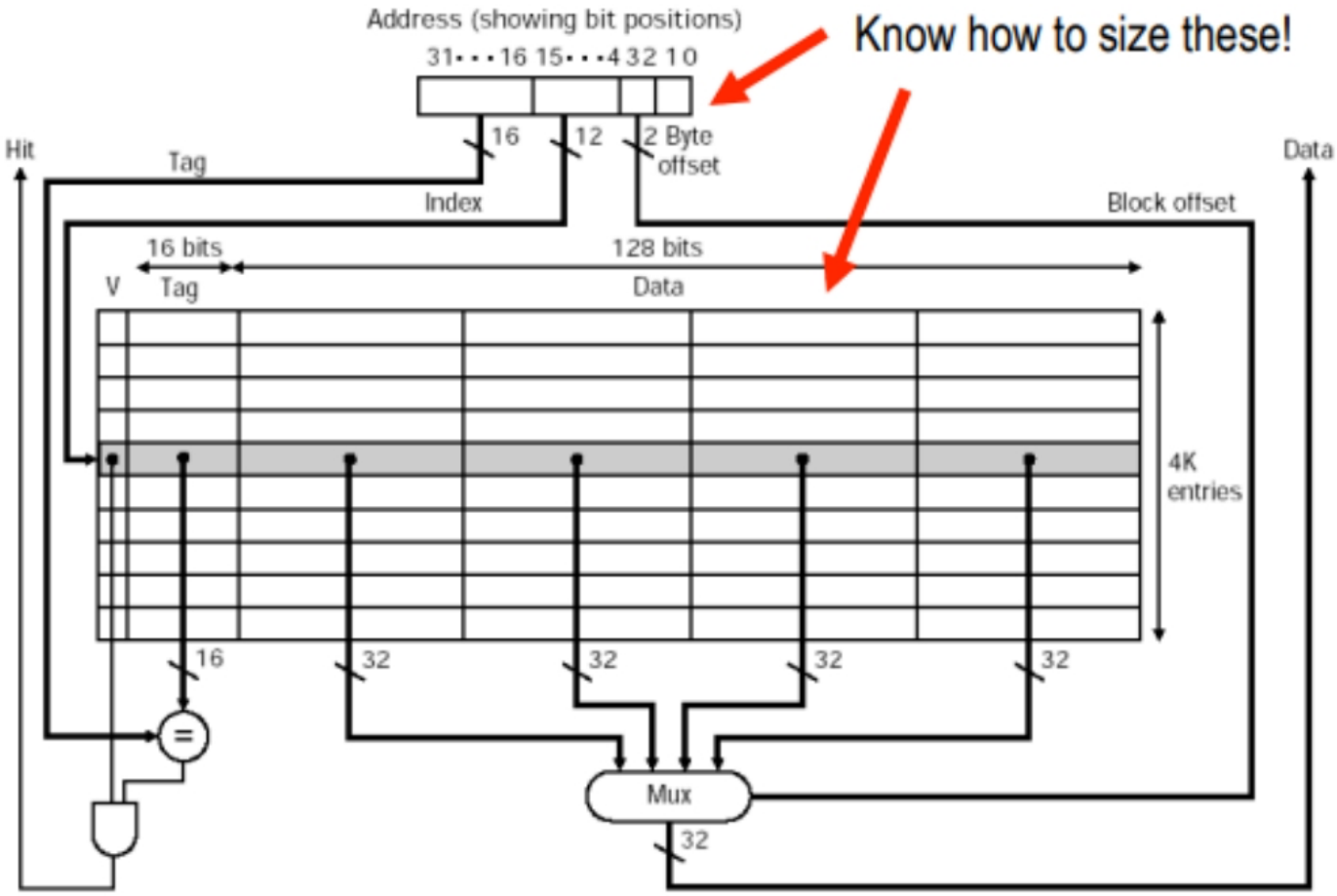


8-block cache

1 word per block (word address = block address)

direct mapped

Assuming 7-bit byte addresses sent by CPU



Write Policy

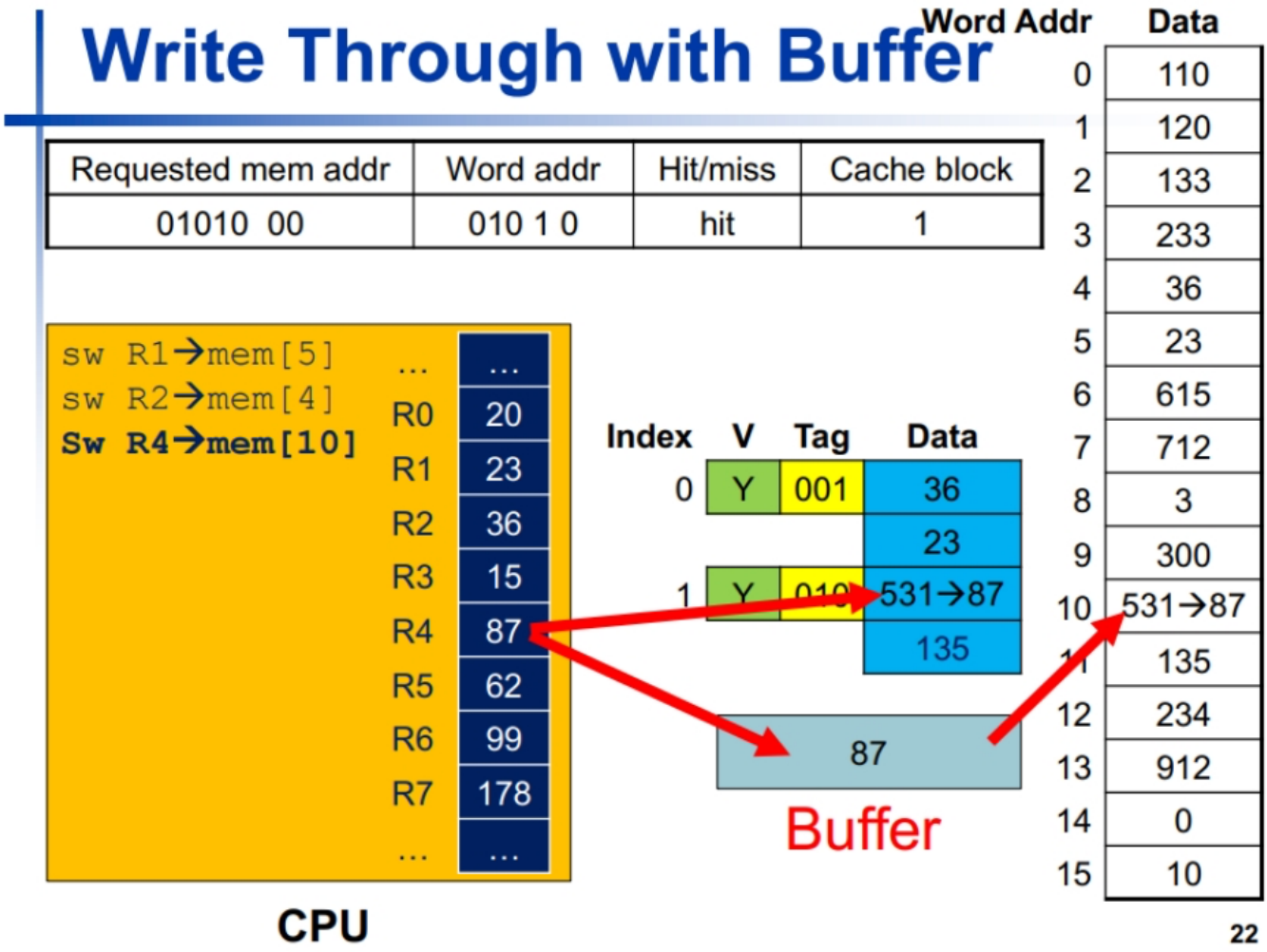
Write Through: On data-write (e.g. sw) hit, could just update the block in cache, but then cache and memory would be inconsistent

Write through: also update the word in memory, but makes writes take longer time

Effective CPI = base CPI + write time (cycles) per instruction

Even worse for write miss

Write Buffer: Buffer stores data to be written to memory. CPU proceeds to next step, while letting buffer complete write through. Frees buffer when completing write to memory CPU stalls if buffer is full.



Write Back: Alternative of write through: On data-write hit, just update the block in cache. CPU keeps track of whether each block is dirty (updated with new values). Write a block back to memory only when a dirty block has to be replaced (on miss). More complex than write through.

What will happen when block size increases?

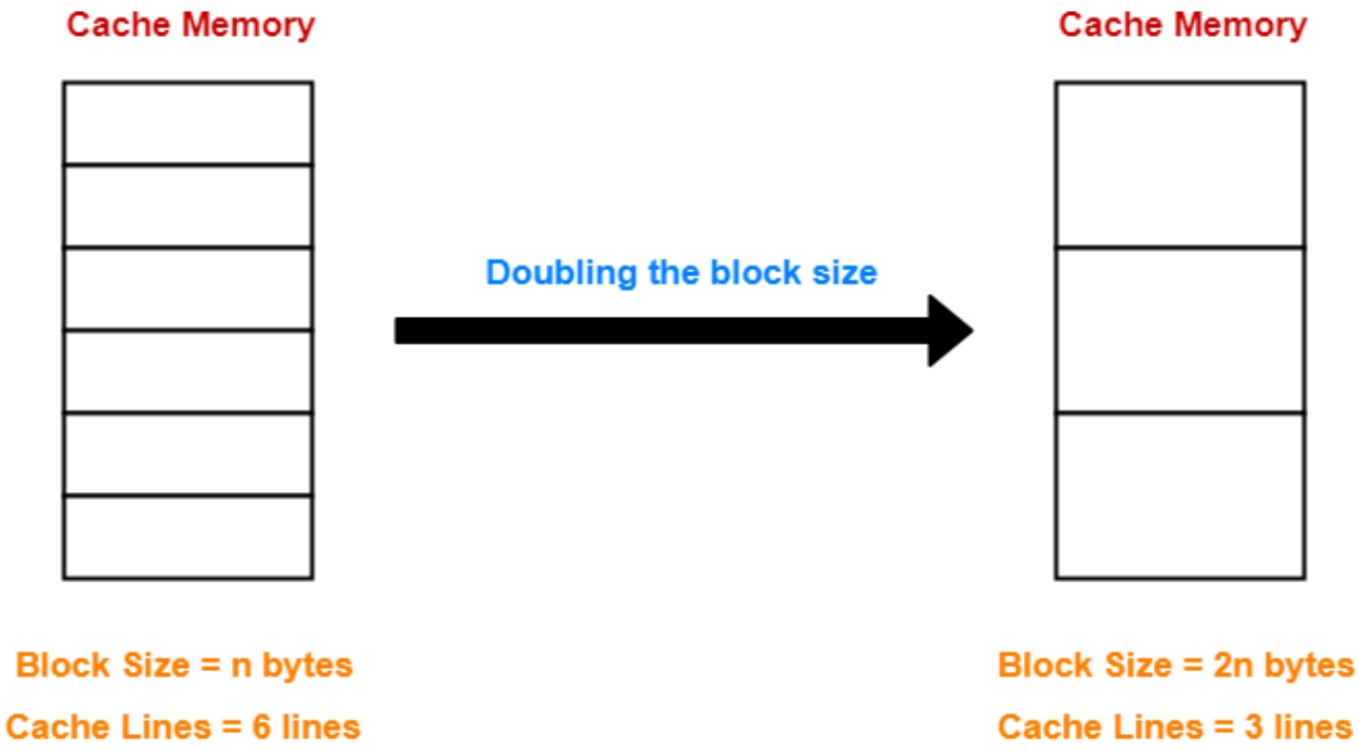
Increasing the block size decreases the number of lines in cache.

With the increase in block size, the number of bits in block offset increases.

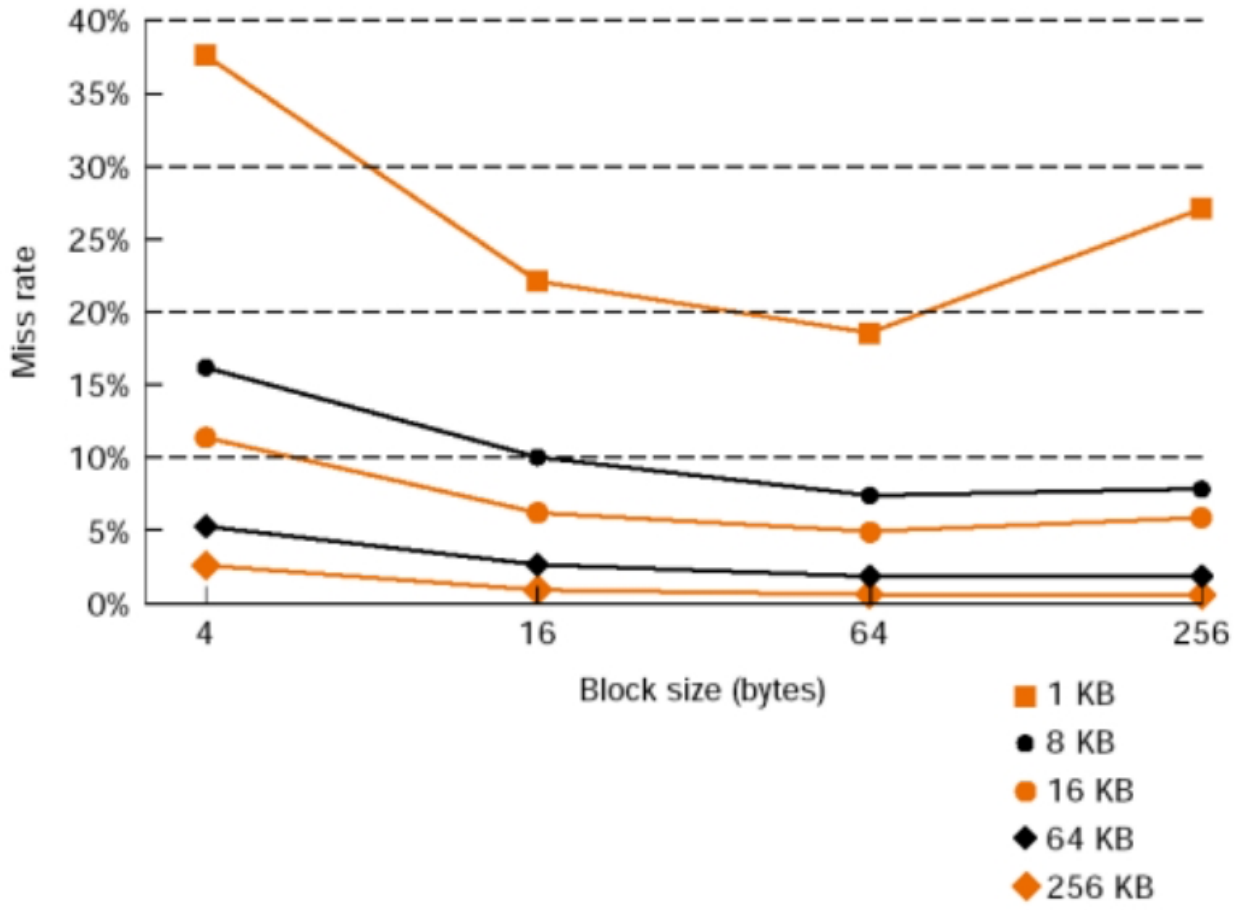
However, with the decrease in the number of cache lines, number of bits in line number decreases.

Thus, number of bits in line number + number of bits in block offset = remains constant.

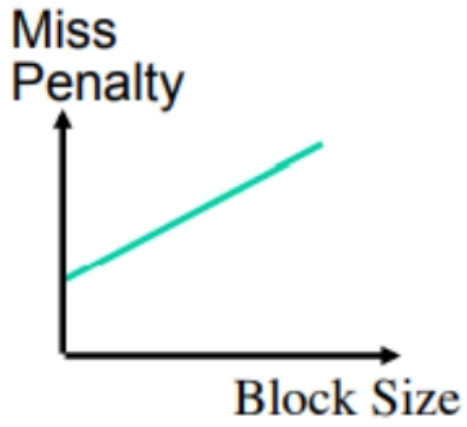
Thus, there is no effect on the cache tag.

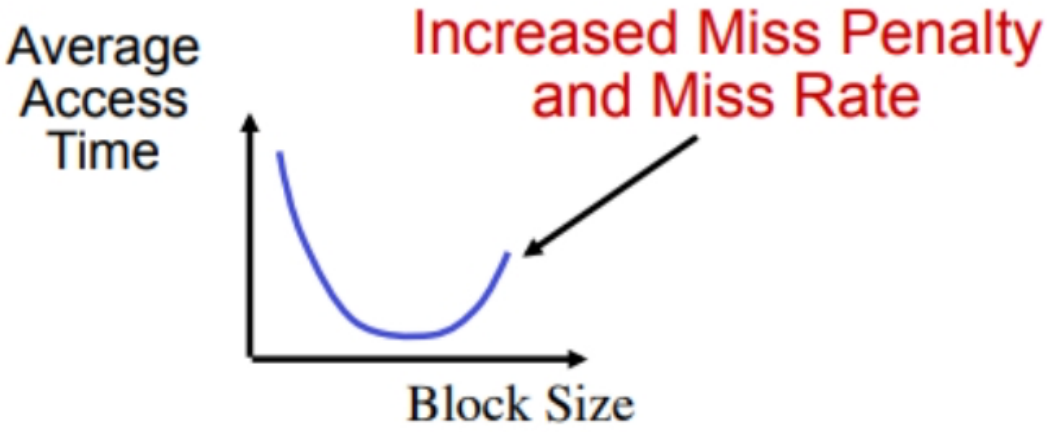


Block Size Increase: Miss Rate



Block Size Increase: Overall Performance





Block Size Increase: Fill Time

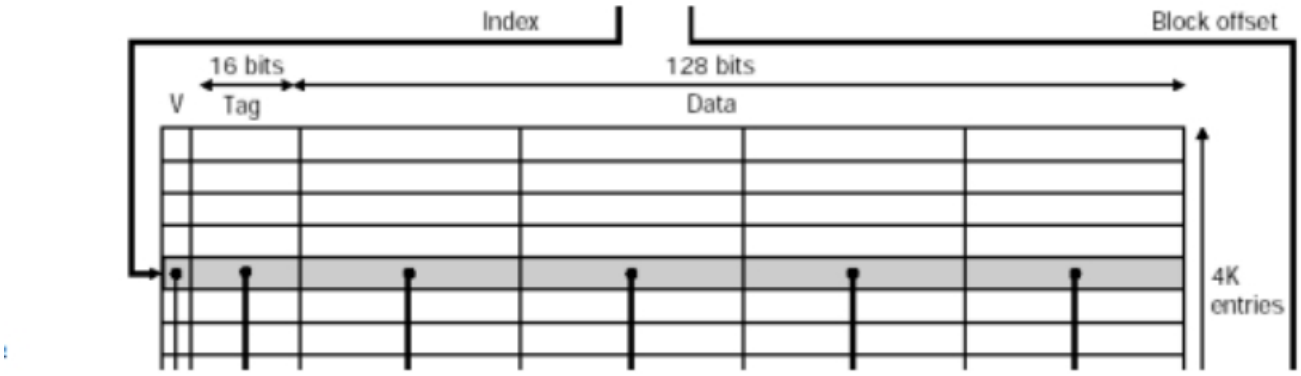
Larger Block Size → Must Wait for Block to Fill

Early Restart

- Deliver word to process/continue execution when word requested is delivered.

Critical Word First

- Early Restart and Fetch the requested word first.



Practice

1. In a direct mapped cache, the number of blocks in the cache is always the same as: (more than one may be correct)

A) The number of bytes in the cache

B) The number of offset bits

C) The number of sets

D) The number of rows

E) The number of valid bits

F) 2^(The number of index bits)

G) The number of index bits

2. Multiple Choice: In designing a computer's cache system, the cache block or cache line size is an important parameter. Which of the following statements is correct in this context?

A. A smaller block size implies better spatial locality

B. A smaller block size implies a smaller cache tag and hence lower cache tag overhead

C. A smaller block size implies a larger cache tag and hence lower cache hit time

D. A smaller bock size incurs a lower cache miss penalty

3. Consider a direct mapped cache with 8 cache blocks (0-7). If the memory block requests are in the order: **3, 5, 2, 8, 0, 6, 3, 9, 16, 20, 17, 25, 18, 30, 24, 2, 63, 5, 82, 17, 24**

   Which of the following memory blocks will not be in the cache at the end of the sequence?

   3 18 20 30

   Also, calculate the hit ratio and miss ratio.