Computer Organization & Architecture

3-6 Cache Memories

Wang Guohua

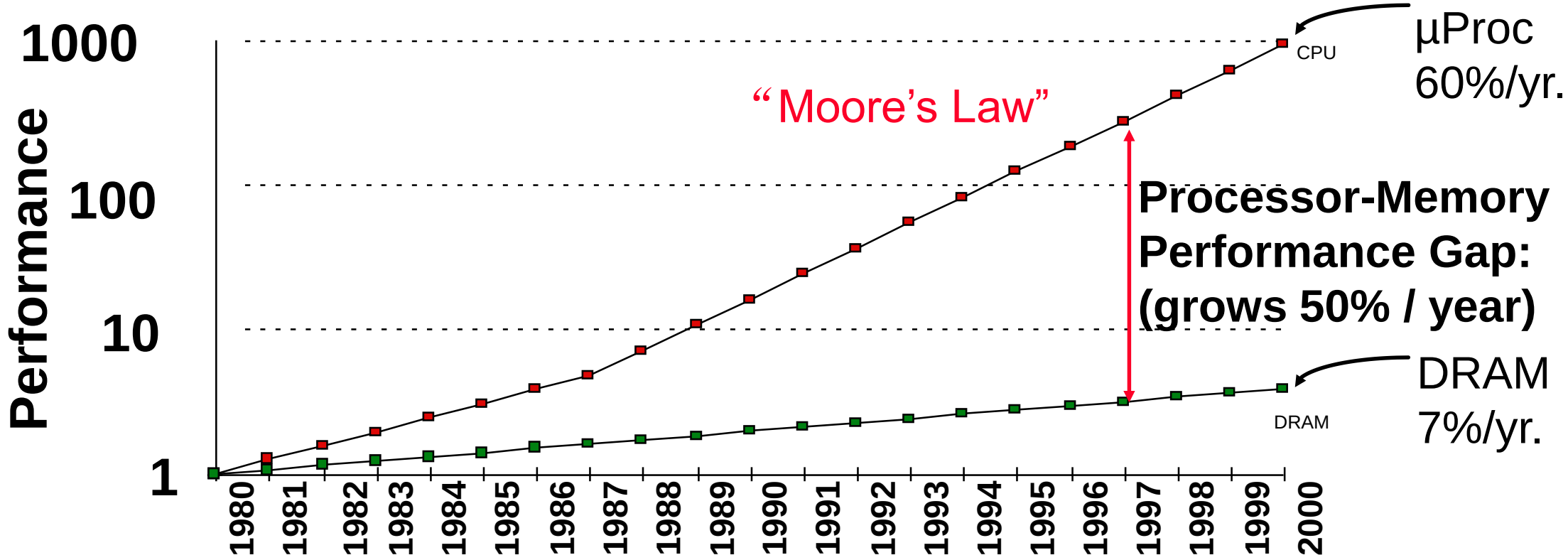
School of Software Engineering

Contents of this lecture

- Purpose of Cache Memory
- Organization of Cache/Main-Memory System
- Hit and Miss
- Cache Design Issues
  - Mapping Schemes
  - Block Identification
  - Replacement Algorithms
  - Write Policies

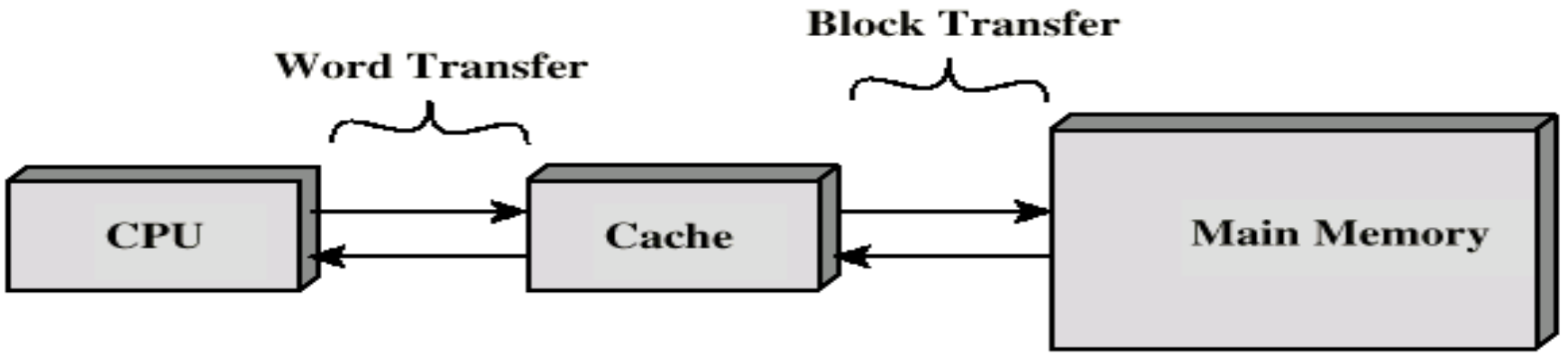
Purpose of Cache Memory (1)

- Problem
  - Processors get faster more quickly than DRAM



Purpose of Cache Memory (2)

- Cache memories are small, fast SRAM-based memories managed automatically in hardware.
- Serves as a buffer between CPU and main memory



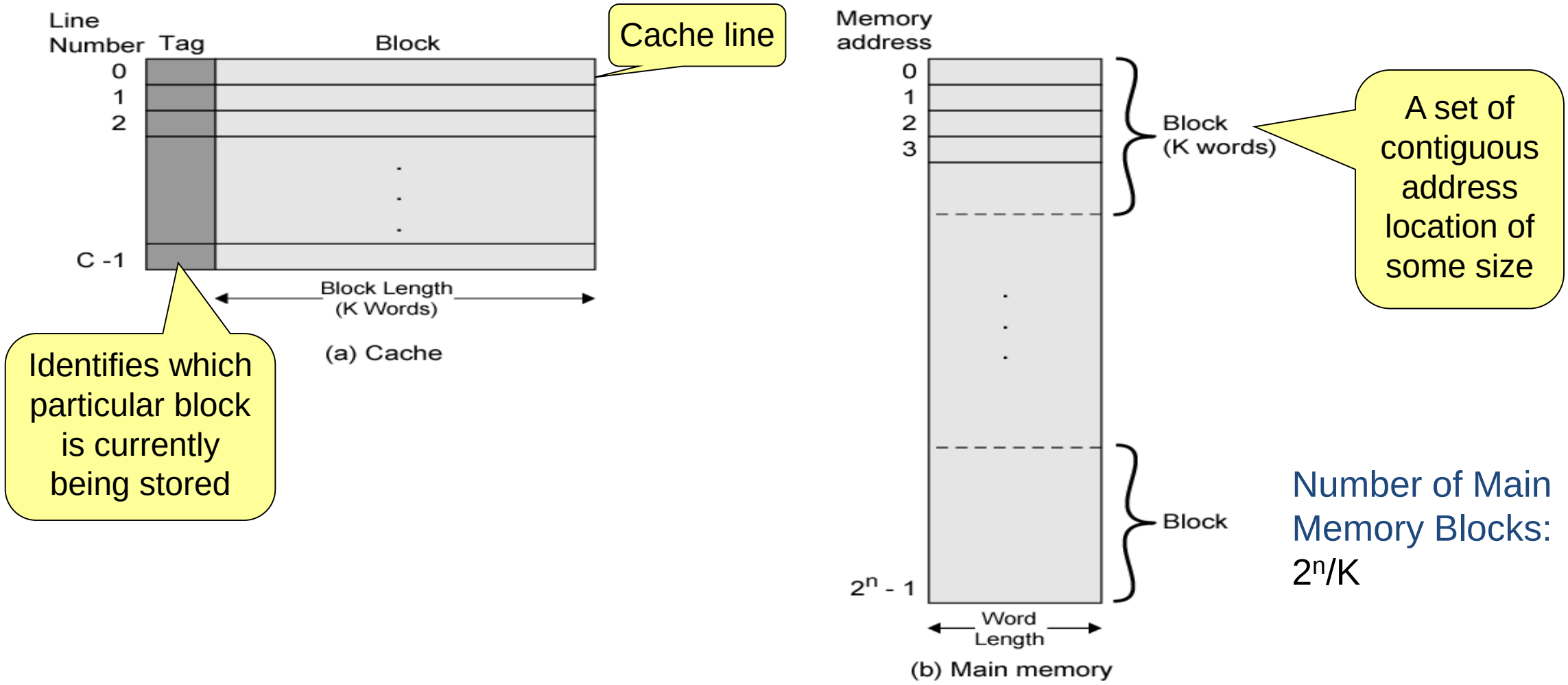
- Hold frequently accessed blocks of main memory

Purpose of Cache Memory (3)

- Purpose
  - Give main memory speed approaching that of the fastest memories available.
  - At the same time, provide a large memory size at the price of less expensive types of semiconductor memories.

Organization of Cache/Main-Memory System

- Cache and Main Memory Organization Figure



Number of Main Memory Blocks: $2^n/K$

Hit and Miss

- Hit
  - A cache access finds data resident in the cache memory.
- Miss
  - A cache access does not find data resident, forcing access to main memory.
- Hit Rate
  - Percentage of memory accesses which are satisfied by cache.
  - High hit rates, well over 0.9, are essential for high-performance computers.
- Miss Rate = 1- (Hit Rate)

Cache Read Operation (1)

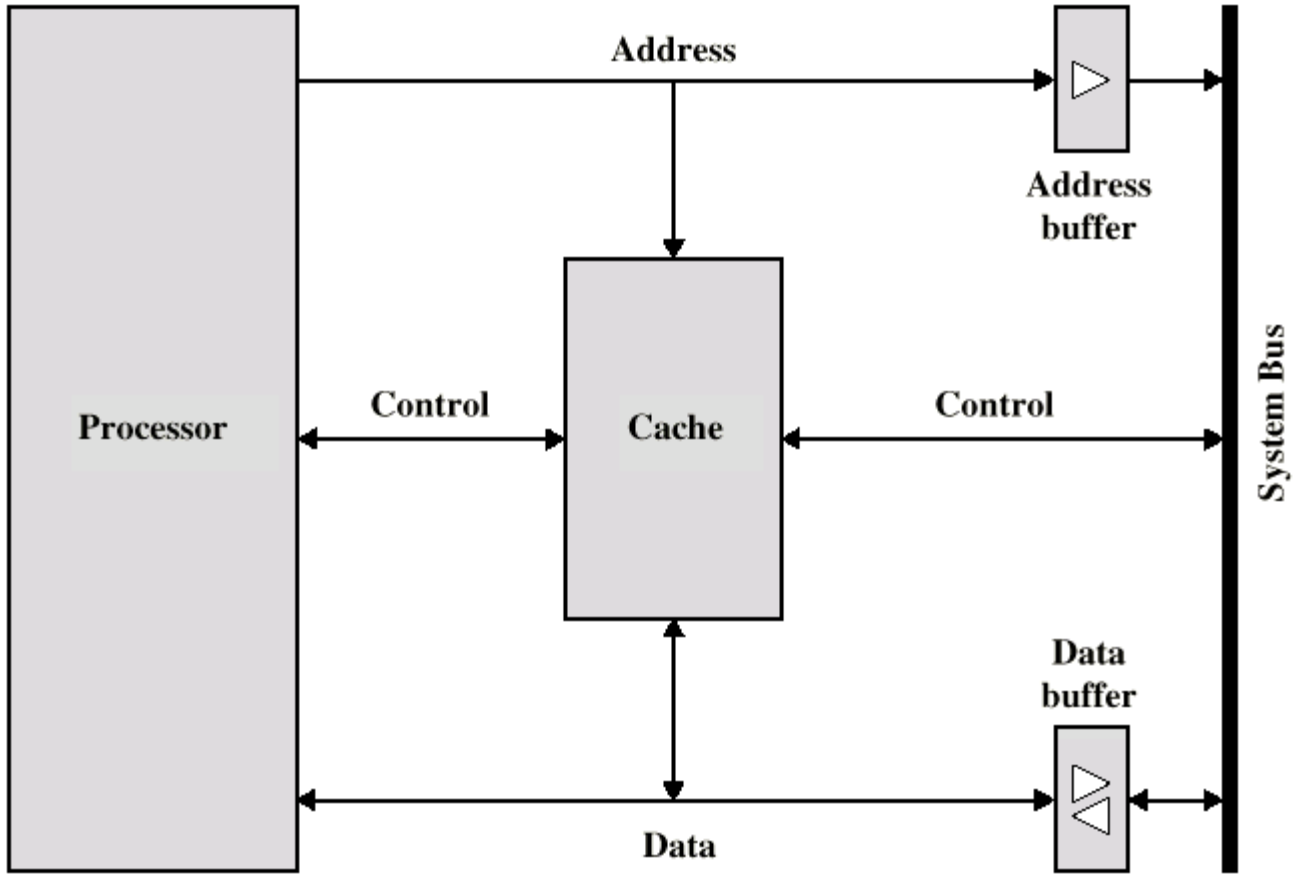
- The processor does not need to know explicitly about the existence of the cache.
- It simply issues Read and Write requests using addresses that refer to locations in the memory.
- The cache control circuitry determines whether the requested word currently exists in the cache.

Cache Read Operation (2)

- Processor: Read Request
  - Cache Read Hit
    - Forward the requested word to the processor
  - Cache Read Miss, Two Approaches
    - Copy the block of words containing the requested word from the main memory into the cache. Then forward the particular word requested to the processor.
    - Load Through/Early Restart
      - Forward the requested word at the same time copy the block of words containing the requested word from the main memory into the cache.

Cache Read Operation (3)

- Cache Read Miss
  - Contemporary Cache Organization



Cache Design Issues

- Cache Mapping Schemes
  - Where can a block be placed in a cache?
- Block Identification
  - How is a block found if it is in a cache?
- Replacement Algorithms
  - Which block should be evicted on a cache miss?
- Write Policy
  - What happens on write hit? What happens on write miss?

Mapping Scheme (1)

- Mapping scheme determines where the block will be placed when it originally copied into the cache.
- Mapping scheme is also required to convert the generated main memory address into a cache location.
  - If the CPU generates an address for a particular word in main memory, and that data happens to be in the cache, the same main memory address cannot be used to access the cache.

Mapping Scheme (2)

- Three Schemes
  - Direct Mapping
  - Associative Mapping
  - Set Associative Mapping
- Assume that
  - Cache line L_i i = 0, 1, …, m – 1 $m = 2^r$
  - Memory block B_j j = 0, 1, …, n – 1 $n = 2^s$
  - Each line or block consists of $k = 2^w$ consecutive words
  - Main memory address: $s+w$ bits

Direct Mapping (1)

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Mapping Function
  - $i = j$ modulo m, where
    - i = cache line number
    - j = main memory block number
    - m = number of lines in the cache

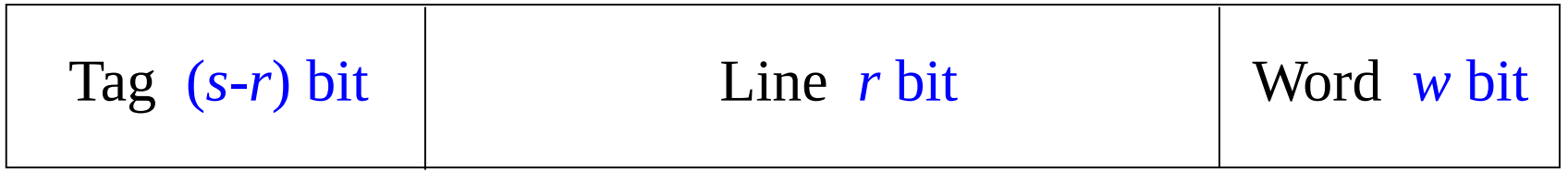
Direct Mapping (2)

- Cache Line Table

Cache Line	Main Memory Blocks Assigned
0	0, m, $2m$, …, $2^s - m$
1	1, $m + 1$, $2m + 1$, …, $2^s - m + 1$
…	…
m-1	m-1, $2m$-1, $3m$-1, …, 2^s-1

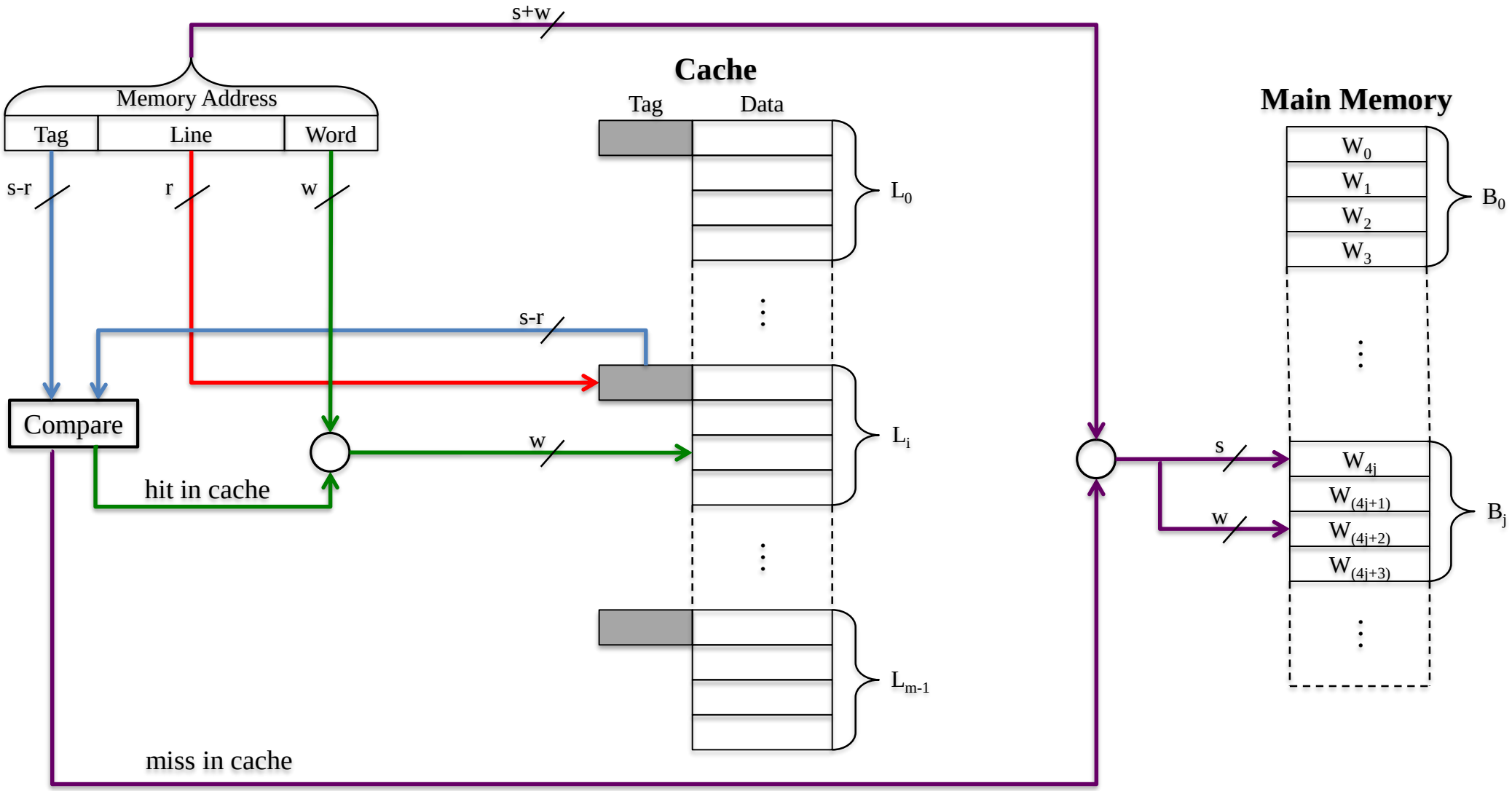
Direct Mapping (3)

- Main Memory Address Structure
  - Least significant *w* bits identify unique word
  - Most significant *s* bits specify one memory block
    - The MSBs are split into a cache line field *r* and a tag of *s-r* (most significant)



Direct Mapping (4)

- Access Cache

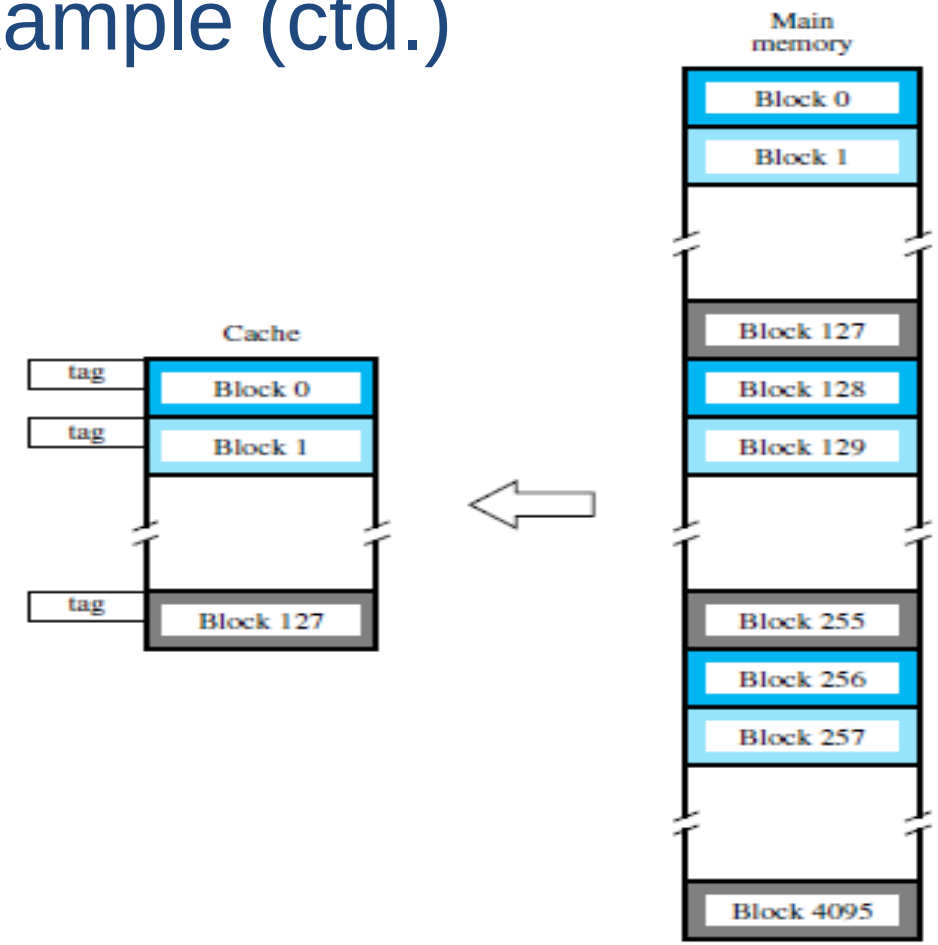


Direct Mapping (5)

- Example: Consider a cache consisting of 128 lines of 16 words each. The main memory has 64K words. Assume that :
  - The main memory is addressable by a 16-bit address.
  - Consecutive addresses refer to consecutive words. Thus,
    - Total Main Memory Blocks =64K/16= 4K=4096
    - $i = j$ modulo 128

Direct Mapping (6)

- Example (ctd.)



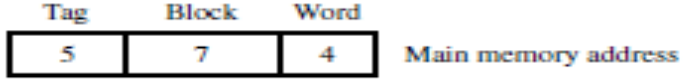


Figure 8.16 Direct-mapped cache.

Cache Line	Main Memory Blocks Assigned
0	0, 128, 256, …, 3968
1	1, 129, 257, …, 3969
…	…
127	127, 255, 383, …, 4095

Direct Mapping (7)

- Example (ctd.)
  - How to divide 16-bits main memory address into tag, line and word fields?
    - *word* field = $log_2^{16} = 4\ bits$
    - *line* field = $log_2^{128} = 7\ bits$
    - *tag* field = 16 – 4 – 7 = 5 bits

Direct Mapping (8)

- Advantages
  - Simple, easy to implement
  - Inexpensive
- Disadvantage
  - Fixed location for given block
    - If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high

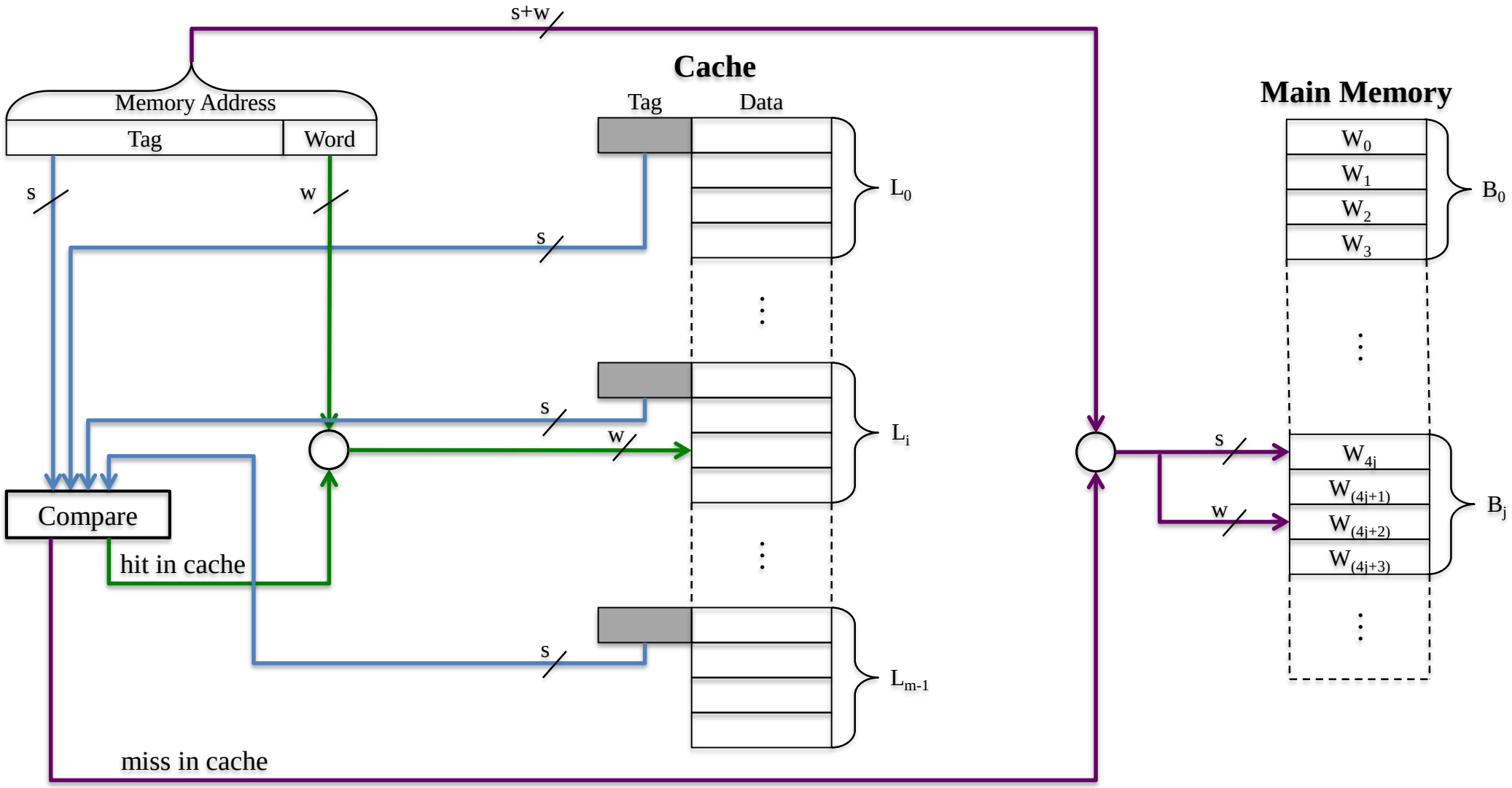
Associative Mapping (1)

- No mapping functions. A main memory block can load into any line of cache
- Main Memory Address Structure
  - Memory address is interpreted as tag and word

  - Tag uniquely identifies block of memory
    - Every line's tag is examined for a match
    - Cache searching gets expensive

Associative Mapping (2)

- Access Cache



Associative Mapping (3)

- Example: Cache 128 lines, Main memory 4096 blocks

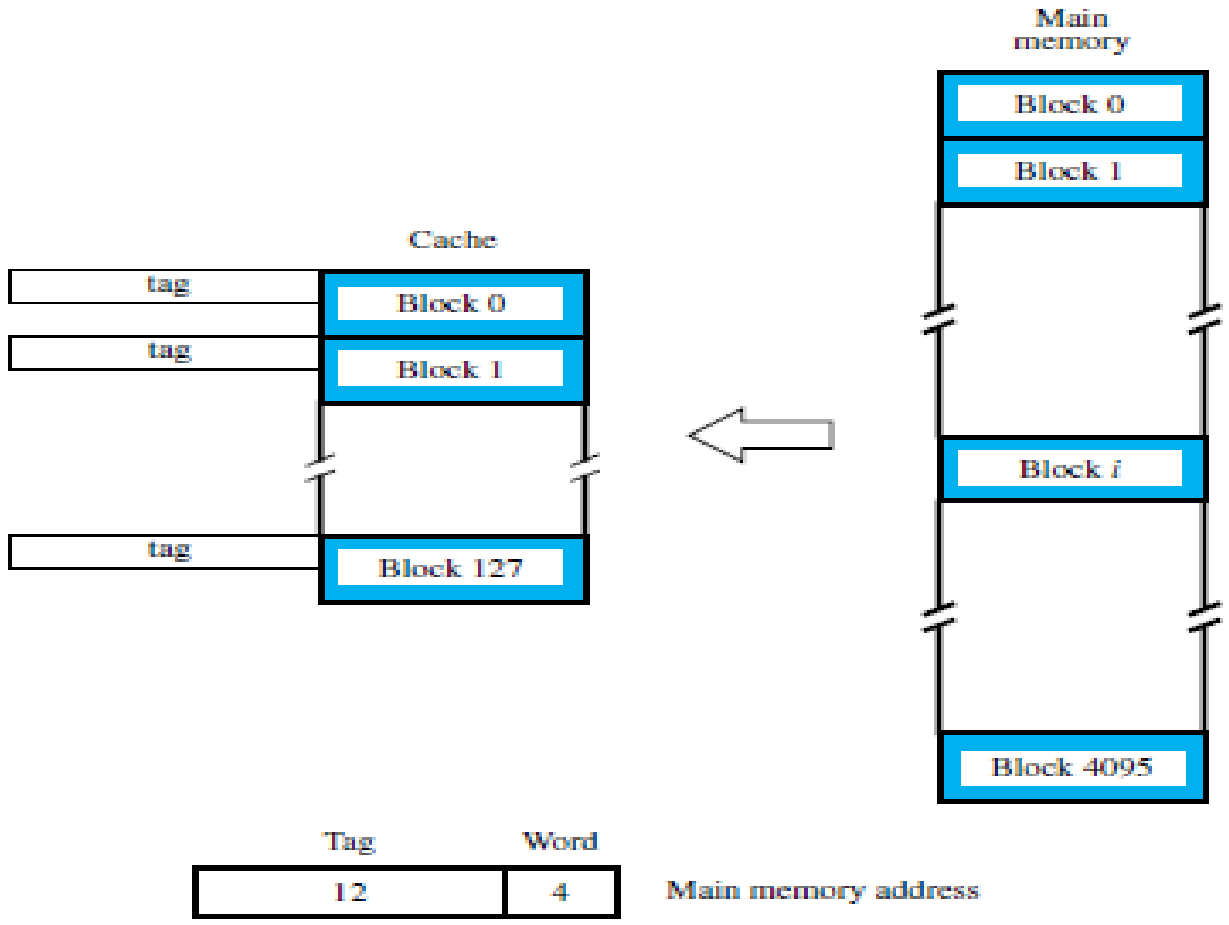


Figure 8.17 Associative-mapped cache.

Associative Mapping (4)

- Advantage
  - It gives complete freedom in choosing the cache location in which to place the memory blocks.
- Disadvantage
  - Complex circuitry required to examine the tags of all cache blocks in parallel.

Set Associative Mapping (1)

- A combination of the direct- and associative-mapping techniques.
- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B_j can be in any line of set *i*
  - e.g. 2 lines per set
    - 2-way set associative mapping
    - A given block can be in one of 2 lines in only one set

Set Associative Mapping (2)

- The cache is divided into v ($=2^d$) sets, each of which consists of k lines. (k-way set associative mapping)
  - $m = v \times k$
  - $i = j$ modulo v, where
    - i = cache set number
    - j = main memory block number
    - v = number of sets in the cache
  - Special Cases
    - $v = m$, $k = 1$ Direct mapping
    - $v = 1$, $k = m$ Associative mapping
    - $v = m/2$, $k = 2$ 2-way Set associative mapping
    - $v = m/4$, $k = 4$ 4-way Set associative mapping

Set Associative Mapping (3)

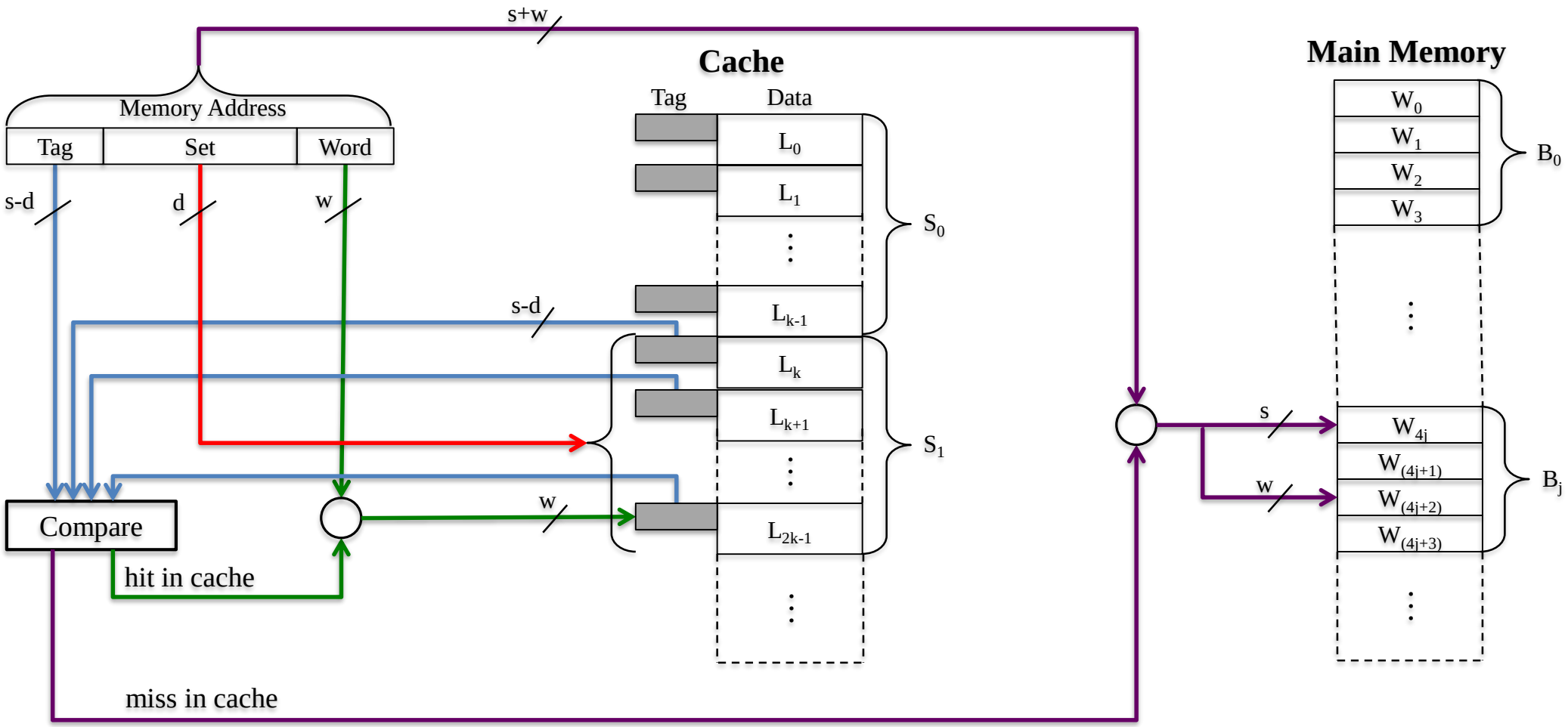
- Main Memory Address Structure

Tag (s-d) bit	Set d bit	Word w bit

– Set Field
  - Determines which set of the cache might contain the desired block.

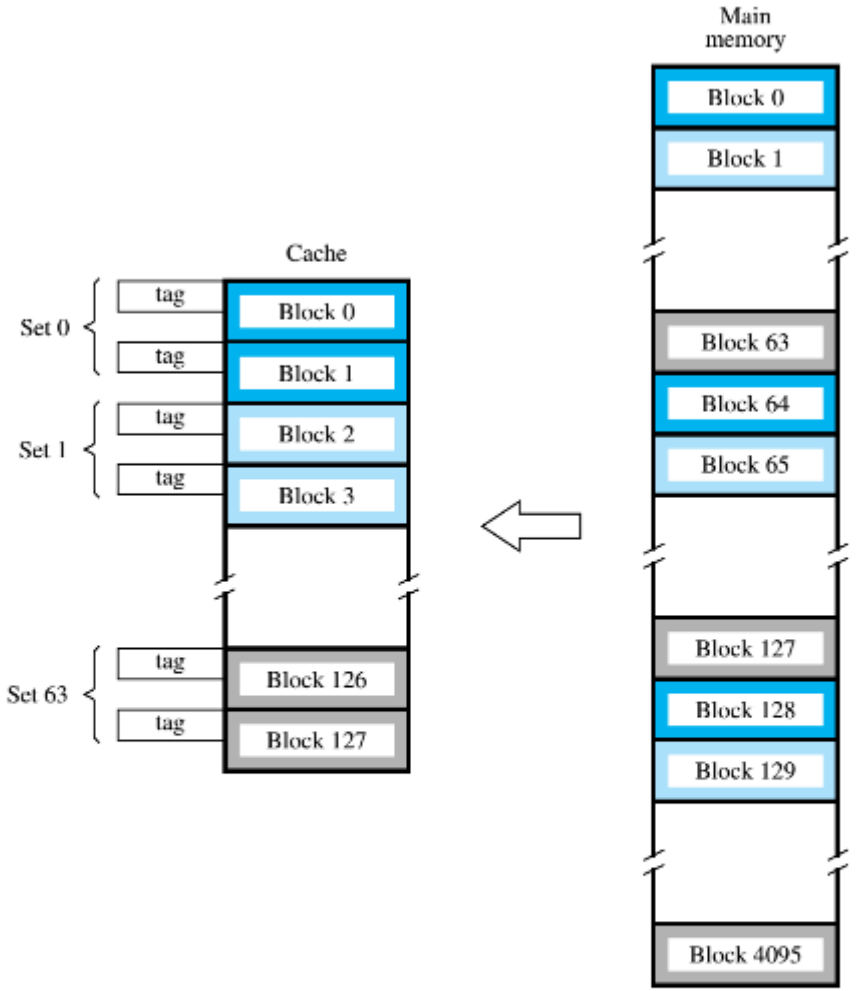
Set Associative Mapping (4)

- Access Cache



Set Associative Mapping (5)

- Example
  - 2-way set associative
  - *i* = *j* modulo 64



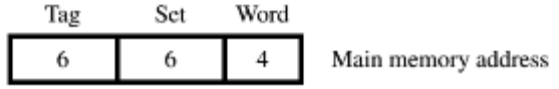


Figure 8.18 Set-associative-mapped cache with two blocks per set.

Set Associative Mapping (6)

- Example (ctd.)
  - How to divide 16-bits main memory address into tag, set and word fields?
    - *word* field = $log_2^{16} = 4\ bits$
    - *set* field = $log_2^{64} = 6\ bits$
    - *tag* field = 16 – 4 – 6 = 6 bits

Set Associative Mapping (7)

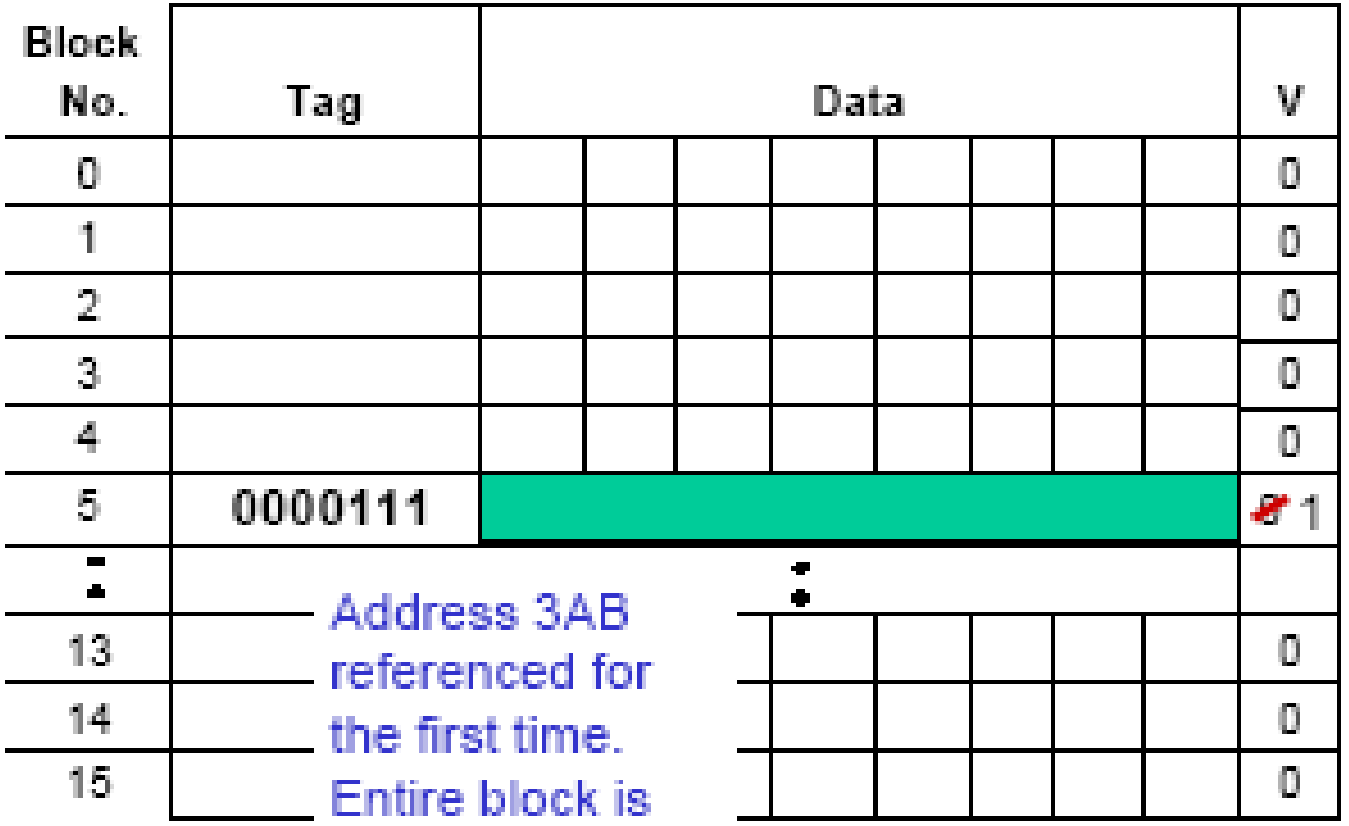
- Advantages
  - The contention problem of the direct method is eased by having a few choices for block replacement.
  - The hardware cost is reduced by decreasing the size of the associative search.
- Disadvantages
  - Tags of each block in a set need to be matched (in parallel) to figure out whether the data is present in cache. Need *k* comparators.
  - Although, the hardware cost for matching is less than fully associative (need *n* comparators, where *n* = #blocks), but it is more than direct mapped (need only one comparator)

Valid Bit (1)

- When power is first turned on, the cache contains no valid data.
- A control bit, usually called the *valid bit*, must be provided for each cache block to indicate whether the data in that block are valid.
- 0 – not valid, 1 – valid
- The valid bits of all cache blocks are set to 0 when power is initially applied to the system.
- If data from main memory is got into cache for a particular block, then valid bit for that field is set.

Valid Bit (2)

- Example



Replacement Algorithms (1)

- Direct Mapped Cache
  - No choice
  - Each main memory block only maps to one cache line
  - Replace that cache line

Replacement Algorithms (2)

- Associative and Set Associative Mapped Cache
  - Hardware implemented algorithm (speed)
  - LRU (Least Recently Used)
    - Replace block not accessed for longest time
  - FIFO (First In First Out)
    - Push block onto queue when accessed
    - Choose block to replace by popping queue
  - Random
    - Replace block chosen at random

Example of Mapping Techniques (1)

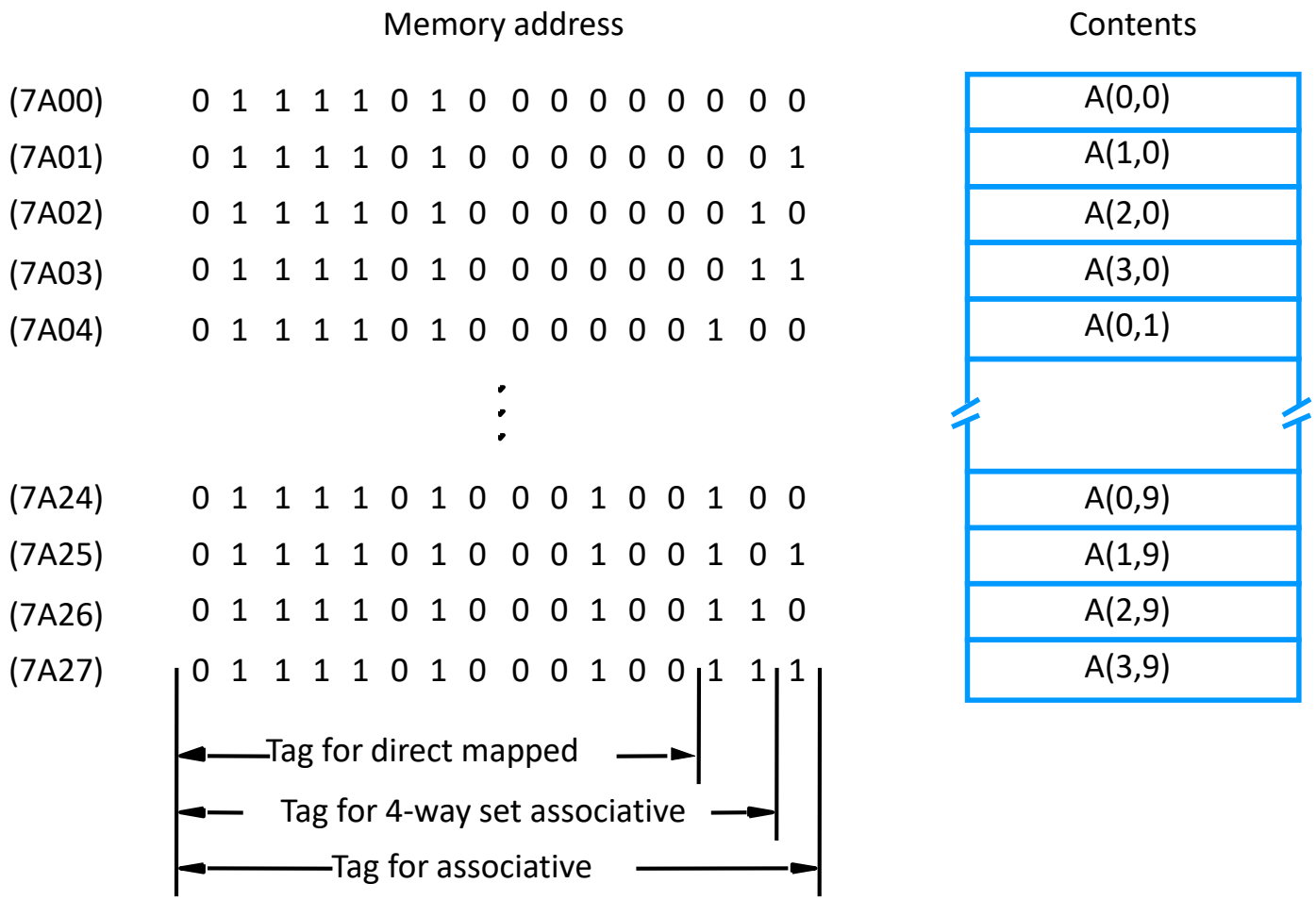
- Assume that
  - The processor has separate instruction and data caches.
  - The data cache has space for only 8 blocks of data.
  - Each block consists of only one 16-bit word of data and the memory is word-addressable with 16-bit addresses.
  - The LRU replacement algorithm is used for block replacement in the cache.

Example of Mapping Techniques (2)

- Question
  - Examine the changes in the data cache entries caused by running the following applications:
    - A 4×10 array of numbers, each occupying one word, is stored in memory locations 7A00 through 7A27.
    - The elements of this array, *A*, are stored in column order.
    - The application normalizes the elements of the first row of *A* with respect to the average value of the elements in the row. That is,

$$A(0,i) \leftarrow \frac{A(0,i)}{(\sum_{j=0}^{9} A(0,j))/10} \qquad \text{for i=0,1,…,9}$$

Example of Mapping Techniques (3)



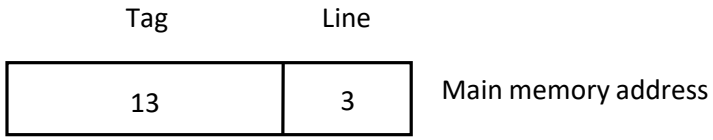
Example of Mapping Techniques (4)

- Program Structure

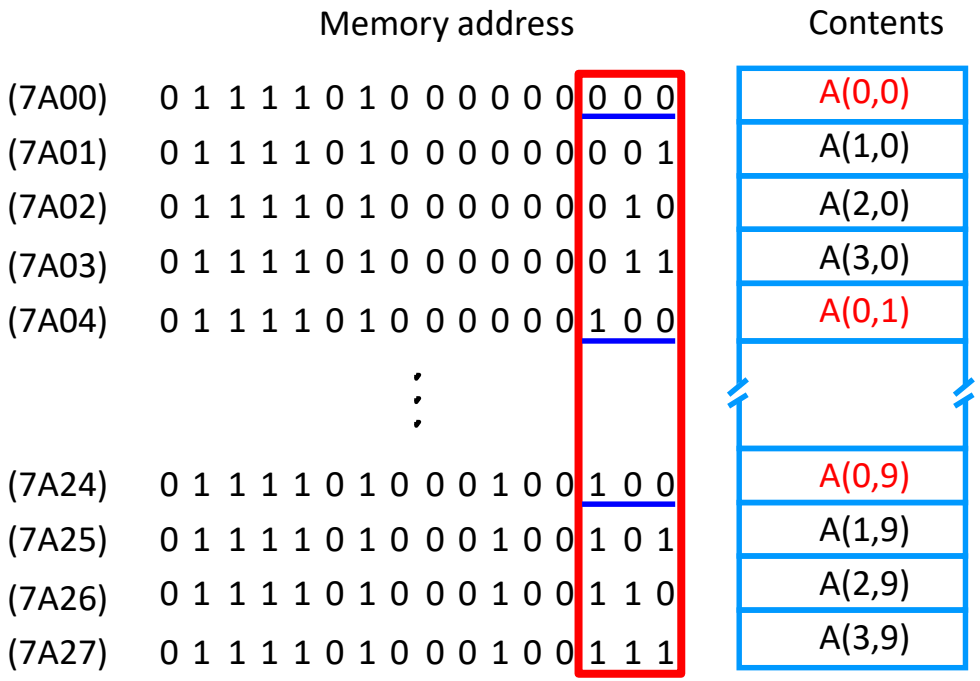
```
SUM   :=   0
for   j:=   0 to   9 do
          SUM   :=   SUM   +   A(0,j)
end

AVE   :=   SUM   /  10

for   i:=   9 down  to   0 do
          A(0,i)   :=   A(0,i)   /  AVE
end
```

Example of Mapping Techniques (5)

- Direct-Mapped Cache



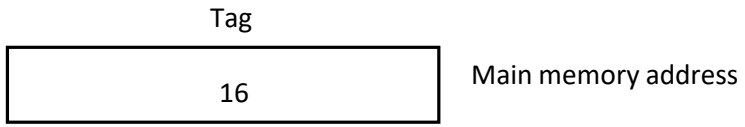
Block position	Contents of data cache after pass:								
	j = 0	*j* = 2	*j* = 4	*j* = 6	*j* = 8	*i* = 6	*i* = 4	*i* = 2	*i* = 0
0	A(0,0)	A(0,2)	A(0,4)	A(0,6)	A(0,8)	A(0,8)	A(0,6)	A(0,4)	A(0,2)
1									
2									
3									
4	A(0,1)	A(0,3)	A(0,5)	A(0,7)	A(0,9)	A(0,7)	A(0,5)	A(0,3)	A(0,1)
5									
6									
7									



Eight elements are replaced while the second loop is executed.

Example of Mapping Techniques (6)

- Associative-Mapped Cache

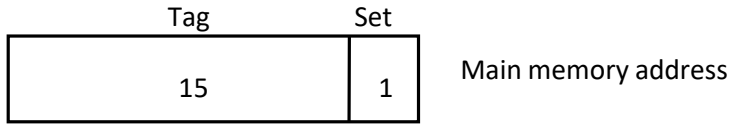


Block position	Contents of data cache after pass:				
	j = 0	*j* = 8	*j* = 9	*i* = 1	*i* = 0
0	A(0,0)	A(0,8)	A(0,8)	A(0,8)	A(0,0)
1	A(0,1)	A(0,1)	A(0,9)	A(0,1)	A(0,1)
2	A(0,2)	A(0,2)	A(0,2)	A(0,2)	A(0,2)
3	A(0,3)	A(0,3)	A(0,3)	A(0,3)	A(0,3)
4	A(0,4)	A(0,4)	A(0,4)	A(0,4)	A(0,4)
5	A(0,5)	A(0,5)	A(0,5)	A(0,5)	A(0,5)
6	A(0,6)	A(0,6)	A(0,6)	A(0,6)	A(0,6)
7	A(0,7)	A(0,7)	A(0,7)	A(0,7)	A(0,7)

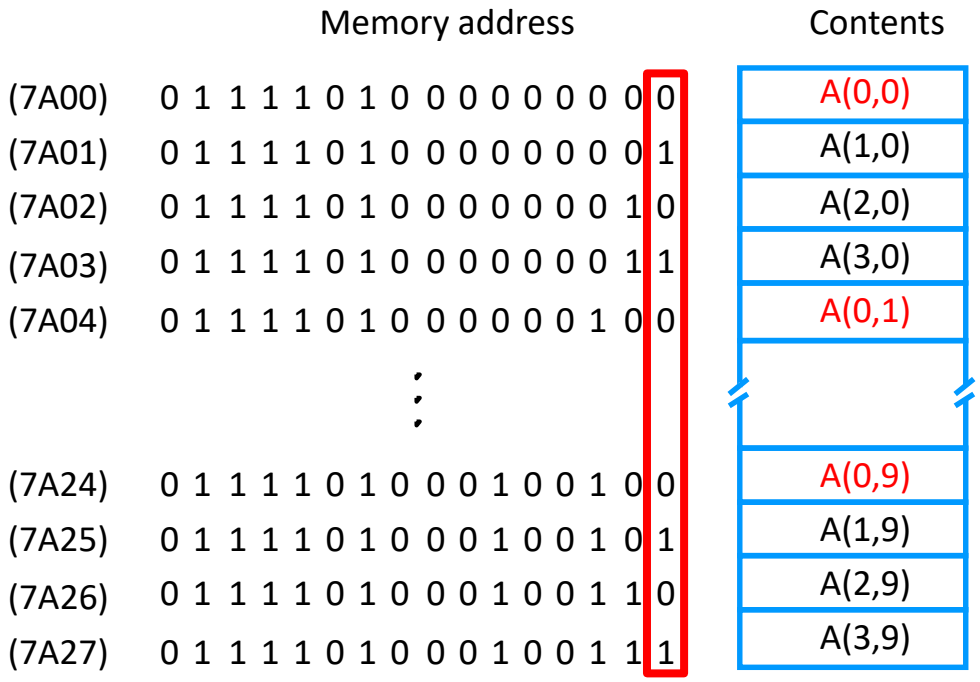
Two elements are replaced while the second loop is executed.

Example of Mapping Techniques (7)

- Set-Associative-Mapped Cache



	Contents of data cache after pass:					
	$j = 3$	$j = 7$	$j = 9$	$i = 4$	$i = 2$	$i = 0$
Set 0	A(0,0)	A(0,4)	A(0,8)	A(0,4)	A(0,4)	A(0,0)
	A(0,1)	A(0,5)	A(0,9)	A(0,5)	A(0,5)	A(0,1)
	A(0,2)	A(0,6)	A(0,6)	A(0,6)	A(0,2)	A(0,2)
	A(0,3)	A(0,7)	A(0,7)	A(0,7)	A(0,3)	A(0,3)
Set 1						



Six elements must be reloaded during execution of the second loop.

Quiz (1)

1. By ____ mapping technique, a main memory block can be placed into any cache block position.

   A. direct　　B. associative　　C. set associative　　D. sequential

2. If a cache has a capacity of 16KB and a line length of 128 bytes, how many sets does the cache have if it is 2-way, 4-way, or 8-way set associative?

   A. 64,32,16　　B. 64,16,8　　C. 32,16,8　　D. 128,64,32

   cache包含16KB/128B=128个line

Quiz (2)

3. A set-associative cache consists of a total of 64 blocks divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 128 words. How many bits are there in a main memory address?

   A. 21　　B. 24　　C. 19　　D. 32

   主存有4096个块，所以需要12位指定块地址；每个块中有128个字，所以需要7位指定字地址。

   12+7=19，所以主存地址是19位。

4. A set-associative cache consists of a total of 64 blocks divided into 4-block sets. The main memory contains 4096 blocks, each consisting of 128 words. How many bits are there in each of the TAG, SET, and WORD fields?

   A. 2,5,12　　B. 12,2,5　　C. 4,8,7　　D. 8,4,7

主存地址总共19位，word部分7位，cache划分为16个set，故set部分4位，剩下19-7-4=8位是tag部分。

Quiz (3)

5. What are advantages and disadvantages of direct mapping?

   Its advantages are: (1) Simple, easy to implement (2) Inexpensive

   Its disadvantage is: fixed location for given block

6. What are advantage and disadvantage of associative mapping?

   Its advantage is that it gives complete freedom in choosing the cache location in which to place the memory blocks.

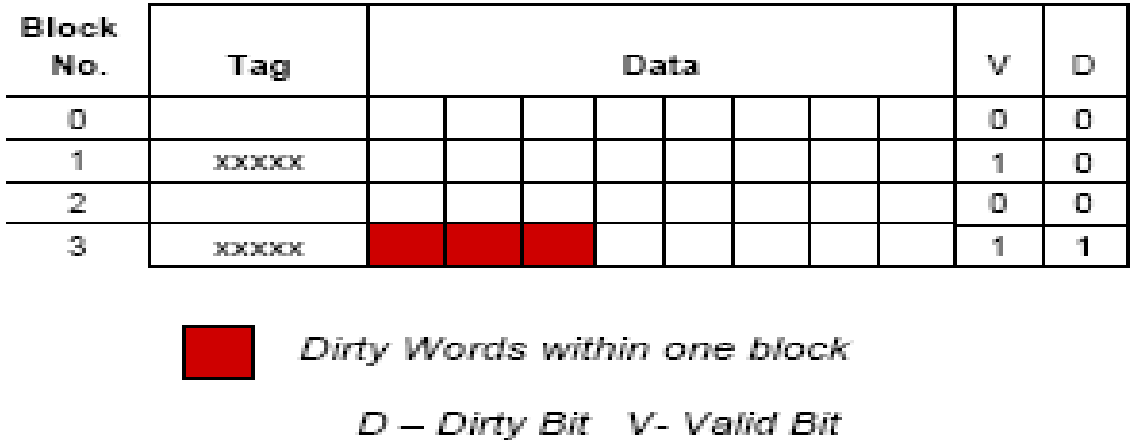
   Its disadvantage is that it requires complex circuitry to examine the tags of all cache blocks in parallel.

Write Policies (1)

- Write Hit
  - Write Through
    - The cache location and the main memory location are updated simultaneously.
    - Advantage
      - Keeps cache main memory consistent at the same time.
    - Disadvantages
      - All writes require main memory access (bus transaction).
      - Slows down the system
        - » If the there is another read request for main memory due to miss in cache, the read request has to wait until the earlier write was serviced.

Write Policies (2)

- Write Hit (ctd.)
  - Write Back
    - Update only the cache location and to mark it as updated with an associated flag bit (dirty or modified bit).
    - When the block containing the marked word is to be removed from the cache, update the main memory location of the word.



Block No.	Tag	Data								V	D
0										0	0
1	xxxxx									1	0
2										0	0
3	xxxxx									1	1

Dirty Words within one block

D – Dirty Bit V- Valid Bit

0 – Not Dirty
1 – Dirty (modified)

Write Policies (3)

- Write Hit (ctd.)
  - Write Back (ctd.)
    - Advantages
      - Faster than write-through, time is not spent accessing main memory.
      - Writes to multiple words within a block require only one write to the main-memory.
    - Disadvantages
      - Portions of main memory are invalid, and hence accesses by I/O modules can be allowed only through the cache.
      - Need extra bit in cache to indicate which block has been modified. Adds to size of the cache.

Write Policies (4)

- Write Miss
  - No-Write Allocate
    - Write-through cache: The information is written directly into the main memory and not loaded into the cache.
  - Write Allocate
    - Write-back cache: The block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

Cache Hit Rate (1)

- The number of hits stated as a fraction of all attempted accesses is called the *hit rate*.
- $H = \frac{N_c}{N_c + N_m} \times 100\%$, where
  - H: hit rate of the cache
  - N_c: number of successful access to data in the cache
  - N_m: number of access to data in the main memory
  - $N_c + N_m$: number of all attempted accesses
- Thus, cache *miss rate* is 1-H.

Cache Hit Rate (2)

- Average Access Time
  - $t_{ave} = Ht_c + (1-H)t_m$, where
    - t_{ave}: average access time experienced by the processor
    - H: hit rate of the cache
    - t_c: time to access information in the cache
    - t_m: time to access information in the memory
  - Example: Consider 70nsec memory with 10nsec cache with a 90% hit rate.
    - Average access time = 0.9*10+(1-0.9)*70=9+7=16ns
    - That's considerably better than memory.

Cache Hit Rate (3)

- Example
  - Consider a computer that has the following parameters
    - Access time to the cache and the main memory are τ and 10τ, respectively.
    - A block of 8 words is transferred to the cache when a cache miss occurs.
      - It takes 10τ to transfer the first word of the block, and τ for each of the remaining 7 words.
      - The miss penalty also includes
        - A delay of τ for the initial access to the cache (cache miss)
        - Another delay of τ to transfer the word to the processor after the block is loaded into the cache.
      - Thus, the miss penalty is $M = \tau + 10\tau + 7\tau + \tau = 19\tau$

Cache Hit Rate (4)

- Example (ctd.)
  - Assume that 30 percent of the instructions in a program perform a Read or a Write operation (i.e., 130 memory accesses for every 100 instructions), and the hit rates in the cache are 0.95 for instructions and 0.9 for data.
  - The improvement using cache:

$$\frac{time\ without\ cache}{time\ with\ cache} = \frac{130 \times 10\tau}{100(0.95\tau + 0.05 \times 19\tau) + 30(0.9\tau + 0.1 \times 19\tau)} = 4.7$$

Multi-level Cache (1)

- Most of today's systems employ multilevel cache hierarchies.
- The levels of cache form their own small memory hierarchy.
- Instruction and Data Caches
  - A unified or integrate cache is one where both instructions and data are cached.
    - Causing excessive cache misses
  - Many modern systems employ separate caches for data and instructions.
  - Advantage
    - Allows accesses to be less random and more clustered.
    - Less access time than unified cache (typically larger).

Multi-level Cache (2)

- Level 1 and 2 Caches
  - Level 1 (L1)
    - Commonly within CPU core
    - Typically split (instruction & data cache)
    - Each relative small, perhaps 8KB to 128KB
    - Access time is typically about 4ns
  - Level 2 (L2)
    - External to CPU core
    - Larger than Level 1, perhaps 256KB to several MB
    - Connected to CPU by high-speed bus, access time is usually around 15-20ns
    - Typically unified

Multi-level Cache (3)

- Level 1 and 2 Caches
  - Average access time

$$t_{ave} = h_1 C_1 + (1 - h_1)\left(h_2 C_2 + \left(1 - h_2\right) M\right)$$

    - h_1: the hit rate in the L1 caches
    - h_2 : the hit rate in the L2 cache
    - C_1: the time to access information in the L1 caches
    - C_2: the miss penalty to transfer information from the L2 cache to an L1 cache
    - M: the miss penalty to transfer information from the main memory to the L2 cache

Multi-level Cache (4)

- Examples of Caches in Commercial Processors
  - Intel Cache
    - 80386 – no on chip cache, external cache
    - 80486 – on chip L1 cache, 8k using 16-byte lines and 4-way set associative organization; external L2 cache
    - Pentium (all versions) – two on chip L1 caches
      - Data & instructions
    - Pentium II – L2 cache on chip
    - Pentium III – L3 cache added off chip

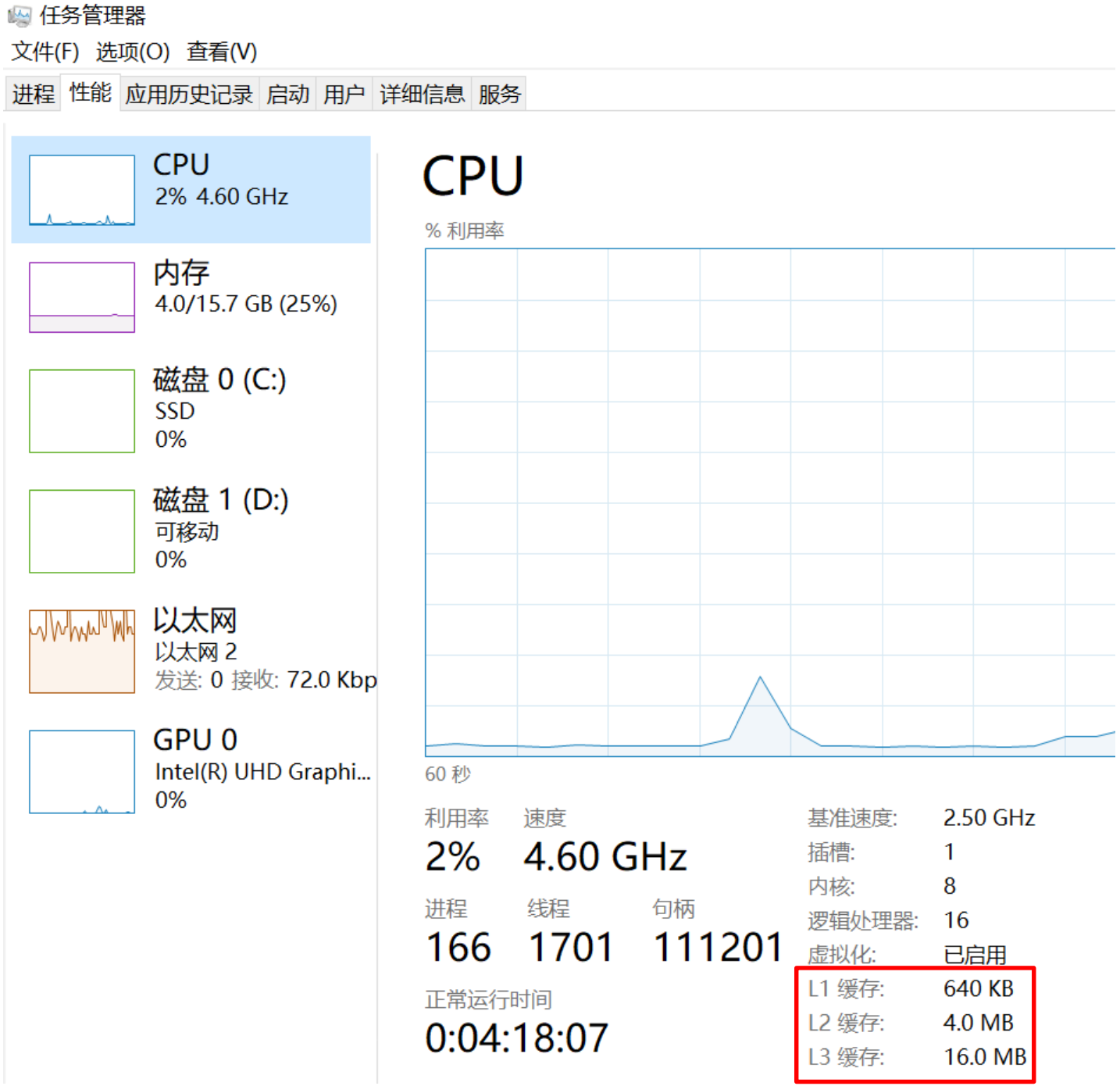
Multi-level Cache (5)

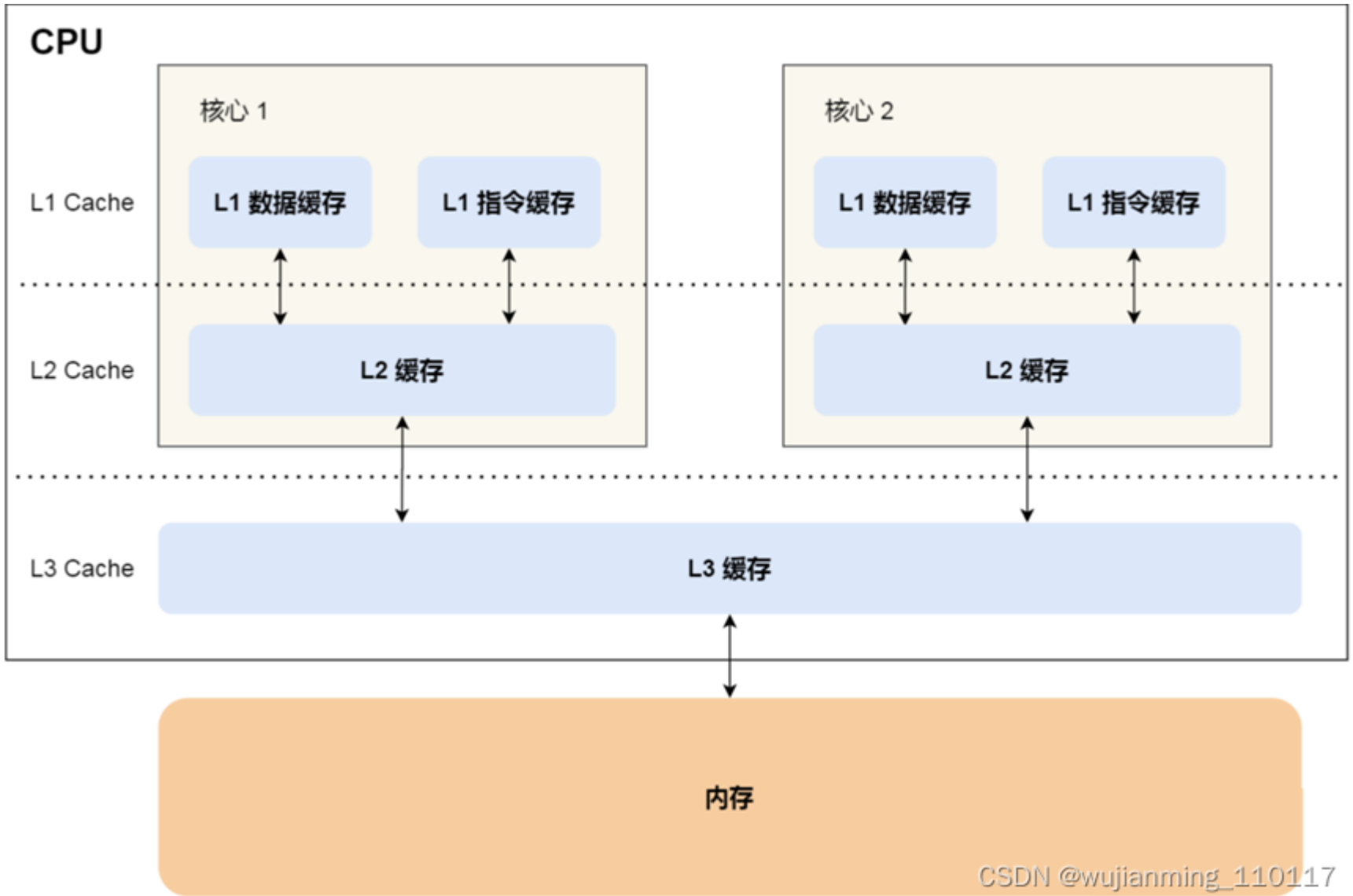
- Examples of Caches in Commercial Processors (ctd.)
  - Intel Cache (ctd.)
    - Pentium 4
      - L1 caches
        - » 8K~16K bytes
        - » 64-byte lines
        - » 4-way set associative
      - L2 cache
        - » Feeding both L1 caches
        - » 256K~512K bytes
        - » 128-byte lines
        - » 8-way set associative
      - L3 cache on chip

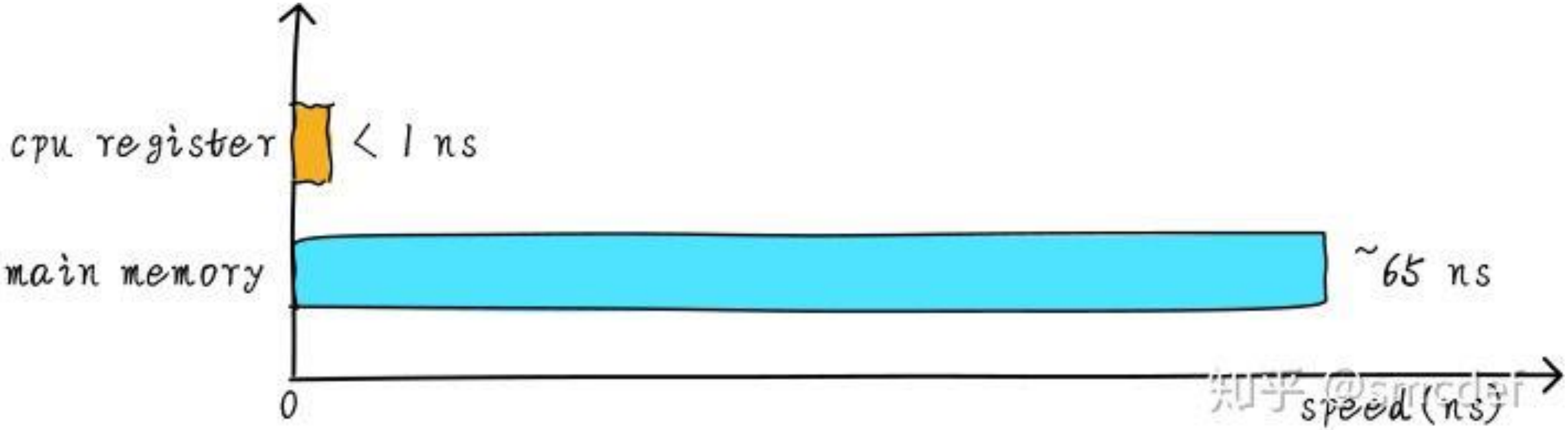
Multi-level Cache (6)

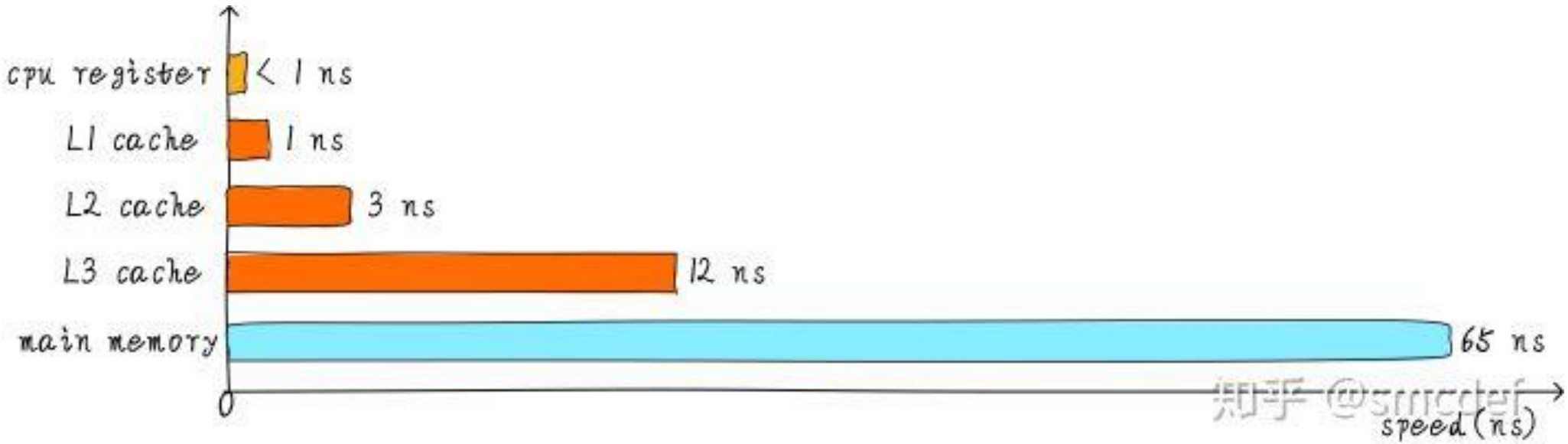
- Examples of Caches in Commercial Processors (ctd.)
  - Intel Cache Evolution

Problem	Solution	Processor on which feature first appears
External memory slower than the system bus.	Add external cache using faster memory technology.	386
Increased processor speed results in external bus becoming a bottleneck for cache access.	Move external cache on-chip, operating at the same speed as the processor.	486
Internal cache is rather small, due to limited space on chip	Add external L2 cache using faster technology than main memory	486
Contention occurs when both the Instruction Prefetcher and the Execution Unit simultaneously require access to the cache. In that case, the Prefetcher is stalled while the Execution Unit's data access takes place.	Create separate data and instruction caches.	Pentium
Increased processor speed results in external bus becoming a bottleneck for L2 cache access.	Create separate back-side bus that runs at higher speed than the main (front-side) external bus. The BSB is dedicated to the L2 cache.	Pentium Pro
	Move L2 cache on to the processor chip.	Pentium II
Some applications deal with massive databases and must have rapid access to large amounts of data. The on-chip caches are too small.	Add external L3 cache.	Pentium III
	Move L3 cache on-chip.	Pentium 4


任务管理器
文件(F) 选项(O) 查看(V)
进程 性能 应用历史记录 启动 用户 详细信息 服务
CPU
2% 4.60 GHz
内存
4.0/15.7 GB (25%)
磁盘 0 (C:)
SSD
0%
磁盘 1 (D:)
可移动
0%
以太网
以太网 2
发送: 0 接收: 72.0 Kbp
GPU 0
Intel(R) UHD Graphi...
0%
CPU
% 利用率
60 秒
利用率
2%
速度
4.60 GHz
进程
166
线程
1701
句柄
111201
正常运行时间
0:04:18:07
基准速度: 2.50 GHz
插槽: 1
内核: 8
逻辑处理器: 16
虚拟化: 已启用
L1 缓存: 640 KB
L2 缓存: 4.0 MB
L3 缓存: 16.0 MB




cpu register
< 1 ns
main memory
~65 ns
0
speed (ns)


cpu register
< 1 ns
L1 cache
1 ns
L2 cache
3 ns
L3 cache
12 ns
main memory
65 ns
0
speed (ns)

Quiz (1)

1. The effectiveness of the cache mechanism is based on a property of computer programs called _____.

   A. parallelism　　B. locality of reference

   C. make the common case fast　　D. forwarding

2. Which of the following manages the transfer of data between the cache and main memory?

   A. compiler　　B. registry　　C. operating system　　D. hardware

   Cache-主存系统完全由硬件管理

Quiz (2)

3. If a cache has 64-byte cache lines, it takes ____ cycles to fetch a cache line if the main memory takes 20 cycles to respond to each memory request and returns 2 bytes of data in response to each request.

   A. 128　　B. 320　　C. 640　　D. 256

   （64/2）*20=640

4. In cache system, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced. This technique is called the ____ replacement algorithm.

   A. FIFO　　B. Random　　C. LFU　　D. LRU

   替换掉最长时间没有被访问过的块，是最近最少使用算法（LRU）的替换原则

Quiz (3)

5. *True or False?* In a direct-mapped cache, it is sensible to use Random replacement policy when a line must be evicted from the cache to make room for incoming data.

   In direct-mapped caches, there is no choice about which line to evict, since the incoming line can only be placed in one location in the cache.

6. *True or False?* For a Write operation, the cache location and the main memory are updated simultaneously. This technique is called the write-through protocol.

Quiz (4)

7. What are advantages and disadvantages of write through policy?

Advantage: Keeps cache main memory consistent at the same time.

Disadvantages:

(1) All writes require main memory access (bus transaction).

(2) Slows down the system - If the there is another read request for main memory due to miss in cache, the read request has to wait until the earlier write was serviced.

Quiz (5)

8. What are advantages and disadvantages of write back policy?

Advantages:

(1)Faster than write-through, time is not spent accessing main memory.

(2)Writes to multiple words within a block require only one write to the main-memory.

Disadvantages:

(1)Portions of main memory are invalid, and hence accesses by I/O modules can be allowed only through the cache.

(2)Need extra bit in cache to indicate which block has been modified. Adds to size of the cache.