Cache Memory

Content:

7.1 Computer memory system overview:
Characteristics of memory system, The memory hierarchy.

7.2 Cache memory principle.

7.3 Elements of cache design:
Cache address, cache size, mapping functions, replacement algorithm, paging, page faults, page replacement policies & write policy.

6.1 Semiconductor main memory:
DRAM, SRAM, Associative memory and Interleaved memory.

Characteristics of memory system

- Location
- Capacity
- Unit of transfer
- Access method
- Performance
- Physical type
- Physical characteristics
- Organisation

Location

- Internal:
  - The internal memory is often equated with main memory, but there are other forms of internal memory.
  - Ex: Processor registers, cache, main memory.
- External:
  - The external memory consist of peripheral storage devices.
  - Ex: optical disks, magnetic disks, tapes.

Capacity

- Number of words
  - The internal memory capacity is expressed in terms of word or bytes.
  - Common word lengths are 16, 32 and 64.
  - A word is 2 bytes (16 bits), a double word is 4 bytes (32 bits), and a quad word is 8 bytes (64 bits).
- Number of bytes
  - External memory capacity is expressed in terms of bytes.
  - A byte is 8 bits.

Unit of Transfer

- Internal Memory:
  - This is the no. of bits read out or written into memory.
  - It may be equal to a word or larger bits such as, 64, 128, 256 bits.
- External Memory:
  - Data is transferred in much larger units than a word, these are referred to as blocks.
- Addressable unit
  - Smallest location which can be uniquely addressed.
  - In some systems, the addressable unit is a word.
  - Many other systems allow addressing at byte level.

Access Methods

- Sequential
  - Memory is organized into units of data called records.
  - Access of such memory has a specific sequence which starts at the beginning and read through in order.
  - A shared read-write mechanism is used to move a record from current location to desired location.
  - Time to access is highly variable.
  - e.g. Tape
- Direct
  - As sequential access, this involves a shared read-write mechanism.
  - Individual blocks have unique address.
  - Access is accomplished by direct jumping to general vicinity plus sequential search and counting to reach final location.
  - Access time is variable.
  - e.g. Disk units.

Access Methods contd…

- Random
  - Each individual addressable location in memory has unique addressing mechanism.
  - Access time of location is independent of sequence of previous access and is constant.
  - e.g. Main memory & Cache memory
- Associative
  - This makes a comparison of desired bits location within a word for a specific match.
  - Thus a word is retrieved based on a portion of its content rather than its address.
  - e.g. RAM

Performance

- Access time (latency)
  - Time from the instant an address is presented to a memory to the instant of getting the valid data for use.
- Memory Cycle time
  - This consist of latency plus any additional time required for the memory to “recover” data before next access can commence.
- Transfer Rate
  - This is the rate at which data can be moved in and out of memory.

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD
- Magneto-optical
  - Blu-ray
  - CD-RW & DVD

Physical Characteristics

- Volatile/Non-volatile
  - Information is lost when electrical power is switched off.
  - Information once recorded remains without deterioration until deliberately changed.
- Erasable/Non-erasable
  - Erasable memory in which content can be remove or replaced by another content.
  - Non-erasable memory can not be erased or altered after manufacturing.

Organisation

- Physical arrangement of bits into words
- Not always used.
- e.g. Interleaved

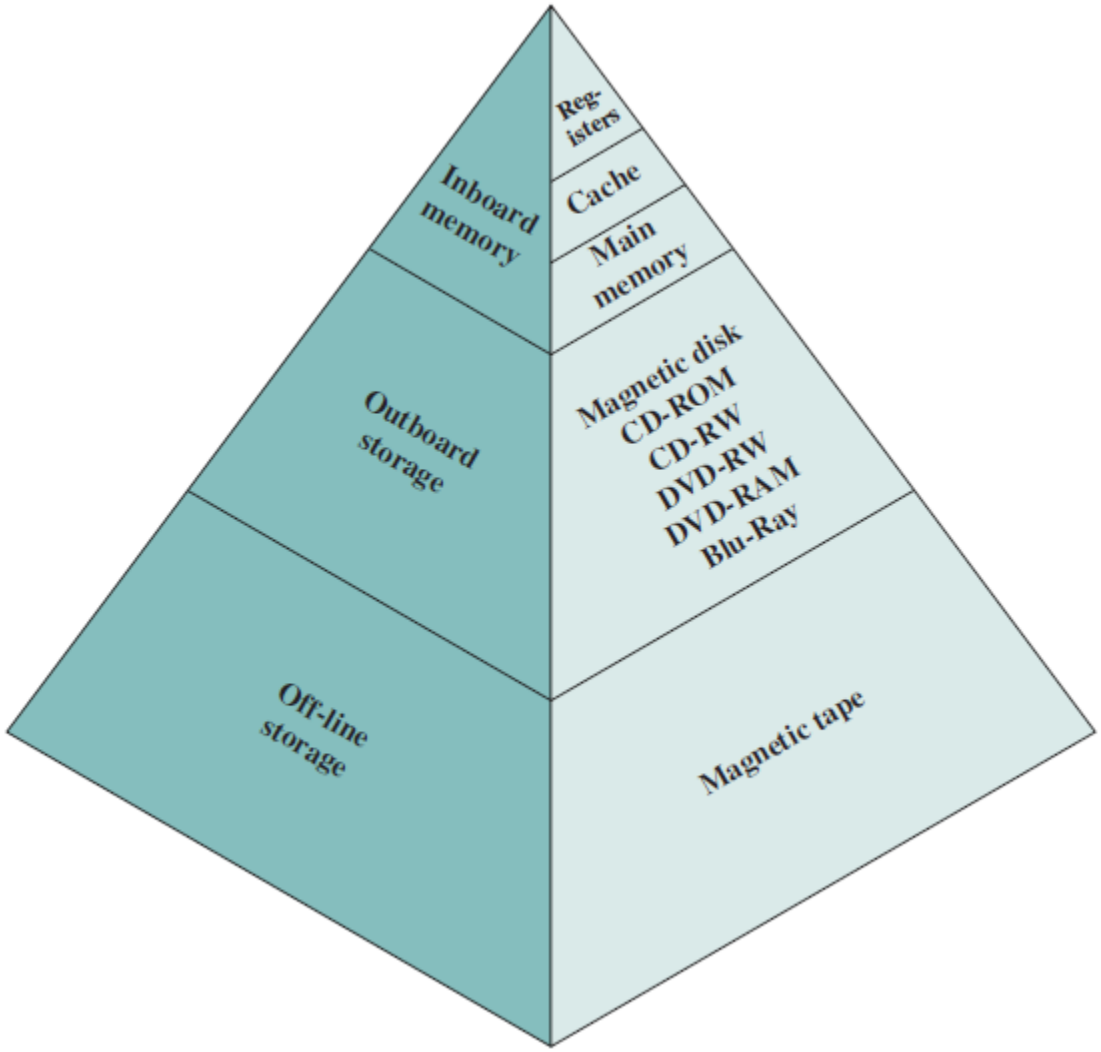
Memory Hierarchy

- Inboard memory
  - Registers
  - Cache
  - Main memory
- Outboard storage
  - Magnetic disks
  - CD/DVD
  - Blu-ray
- Offline storage
  - Magnetic tape

The three characteristics of memory

- How much?
  - Capacity
- How fast?
  - Access time
- How expensive?
  - Cost per bit

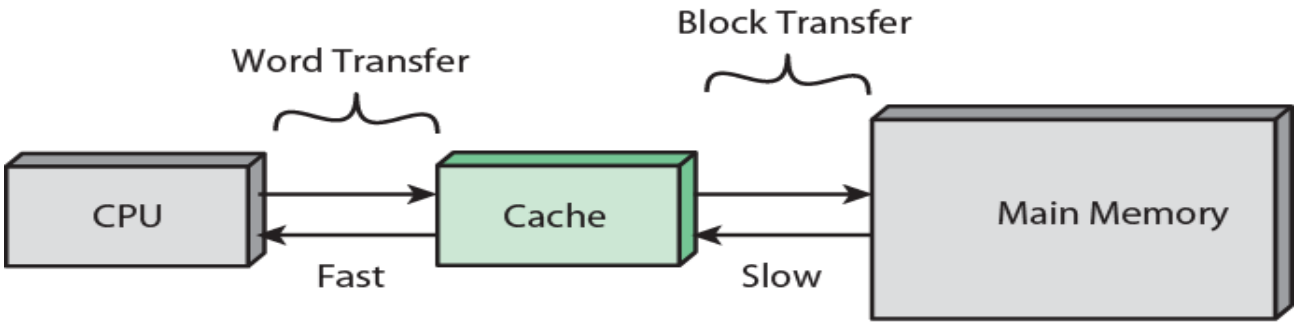
Memory Hierarchy - Diagram



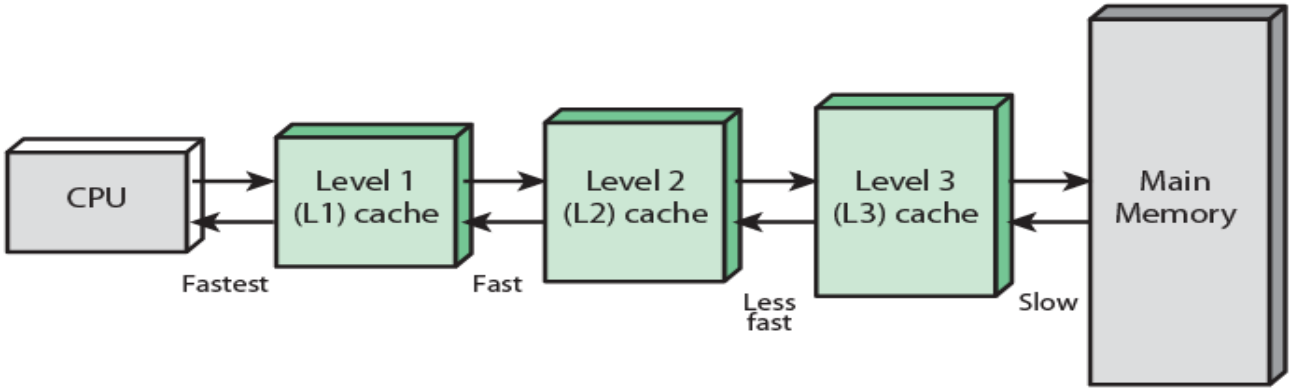
Cache Memory Principle

- Small amount of fast memory
  - Designed to combine the memory access time of processor/CPU registers with main memory.
- Sits between normal main memory and CPU
  - A single cache organization has a relatively large and slow main memory together with smaller and faster cache memory.
  - The cache contains a copy of portions of main memory.

Cache and Main Memory



(a) Single cache



(b) Three-level cache organization

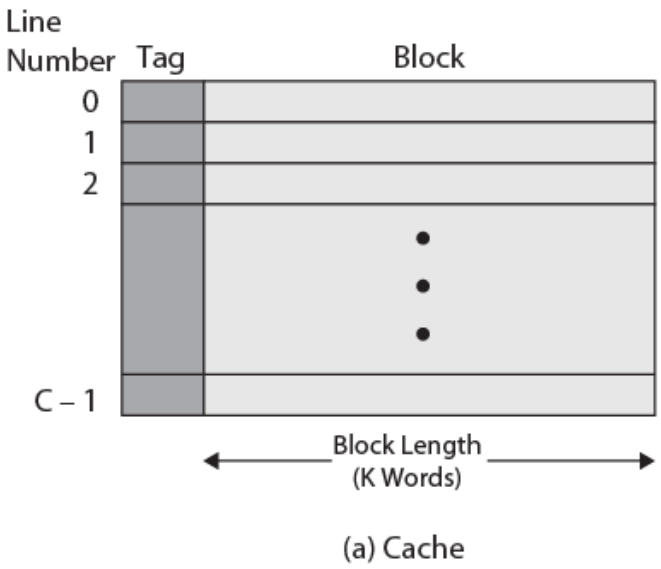
Cache operation – overview

- CPU requests contents (word) of main memory location.
- Check is made to see if word is in the cache.
- If present, word gets delivered to processor/CPU.
- If not present, a block of main memory is read into cache and then the word is delivered to the processor.
- A multi level cache organization has three levels L1, L2, L3, where L2 is slower & larger than L1 and L3 is slower & larger than L2.
- The structure of a cache/main memory system consists of 2^n addressable memory. There are (M) memory blocks in main memory.
- The cache consist of (m) blocks called lines.

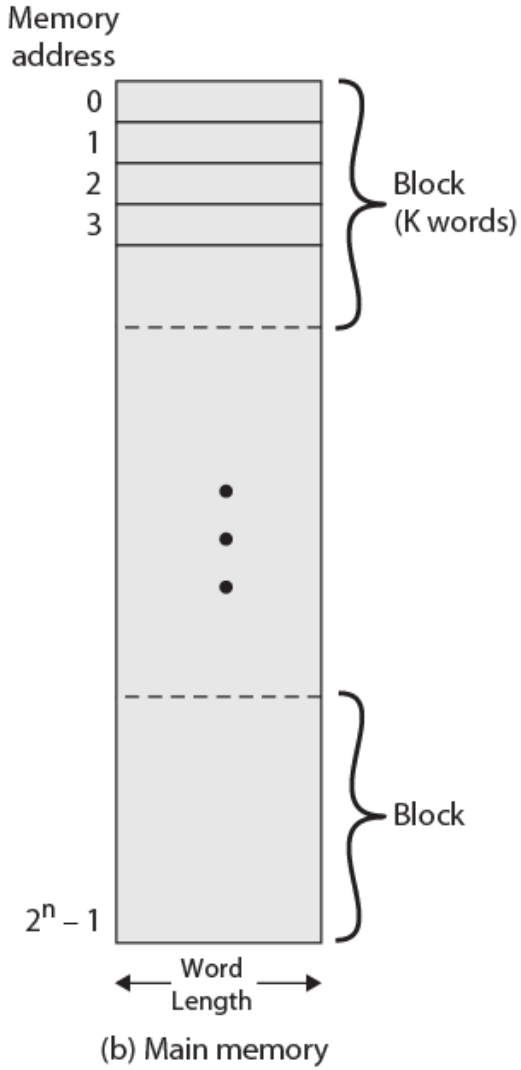
Cache operation – overview

- Each line contains k words plus a tag of few bits and control bits.
- Cache includes tags to identify which block of main memory is in each cache slot.
- The length of line excluding tag and control bits is the line size.
- The no. of lines (m) is less than main memory blocks (M).
- Therefore, an individual line cannot be permanently dedicated to a particular block.
- Thus, each line has a **tag** that identifies which particular block is being currently stored.

Cache/Main Memory Structure

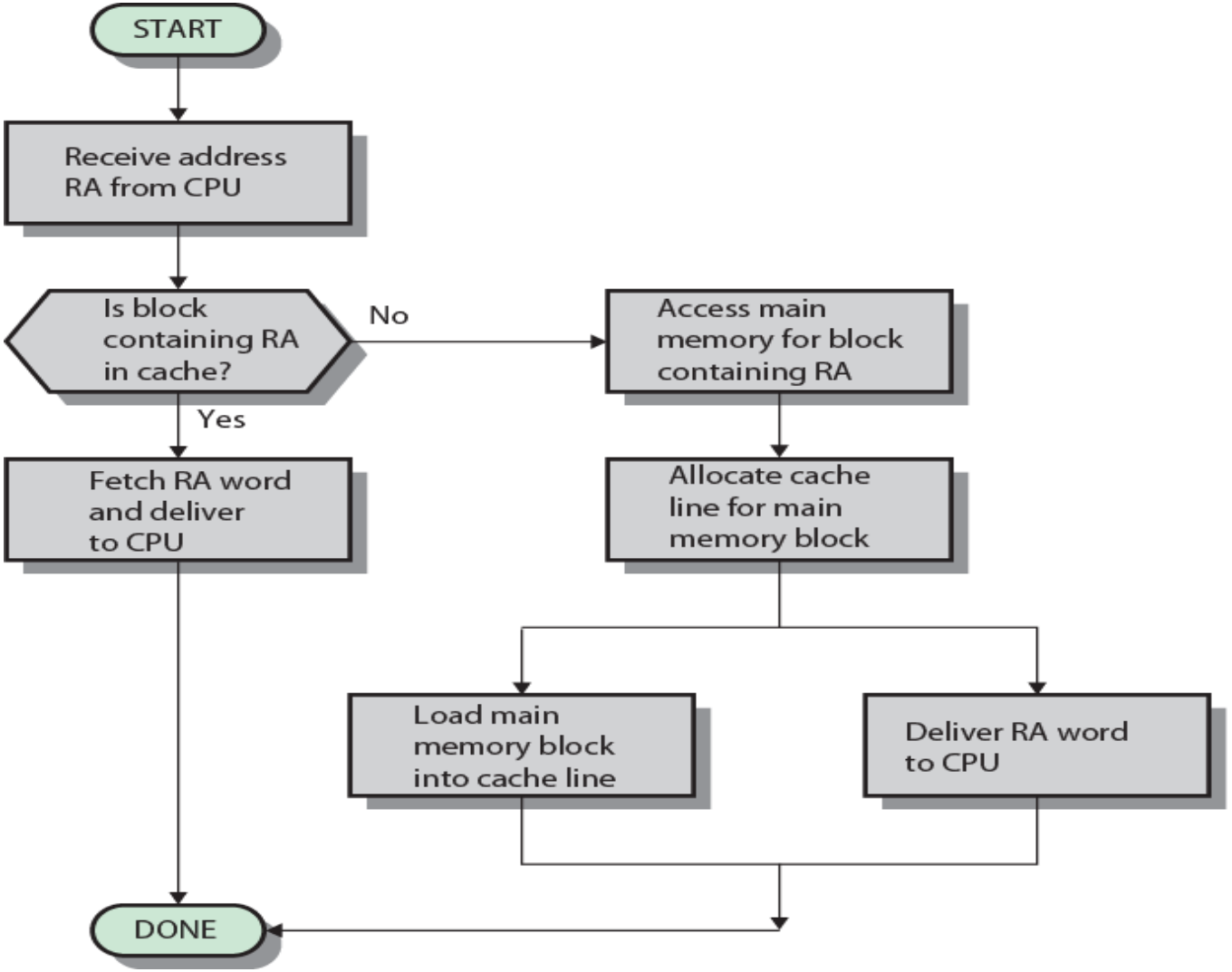


(a) Cache

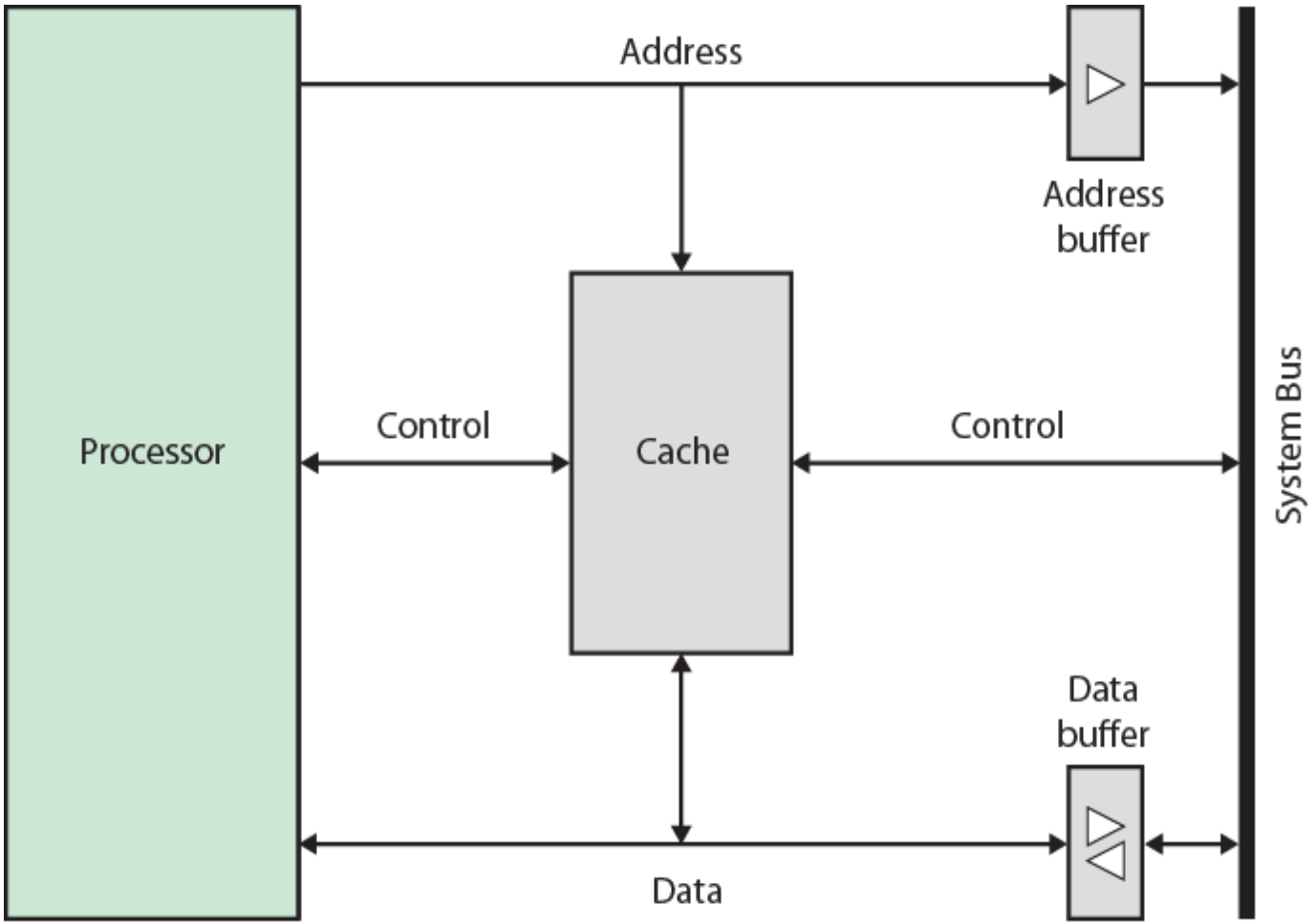


(b) Main memory

Cache Read Operation - Flowchart



Typical Cache Organization (Hit & Miss operation)



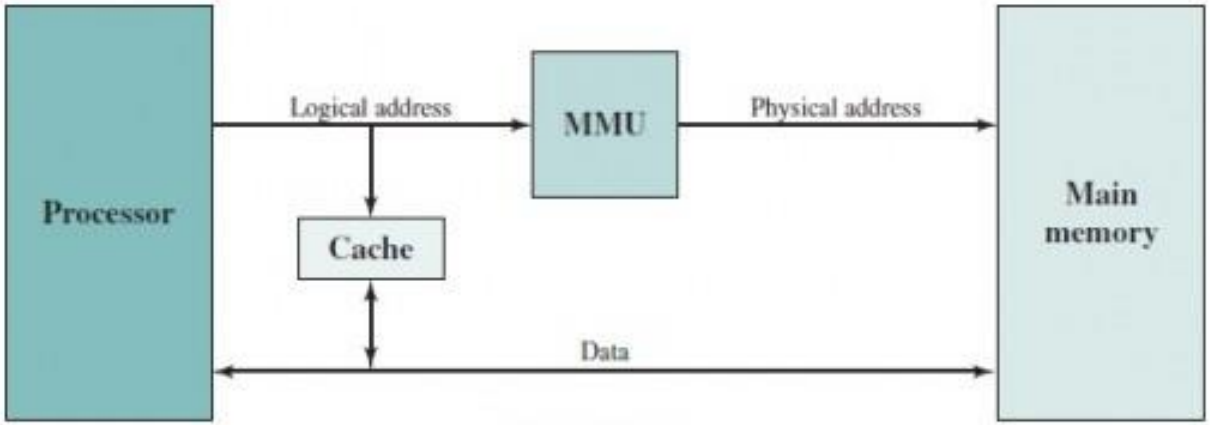
Elements of Cache Design

- Addressing
- Size
- Mapping Function
- Replacement Algorithm
- Write Policy
- Block Size
- Number of Caches

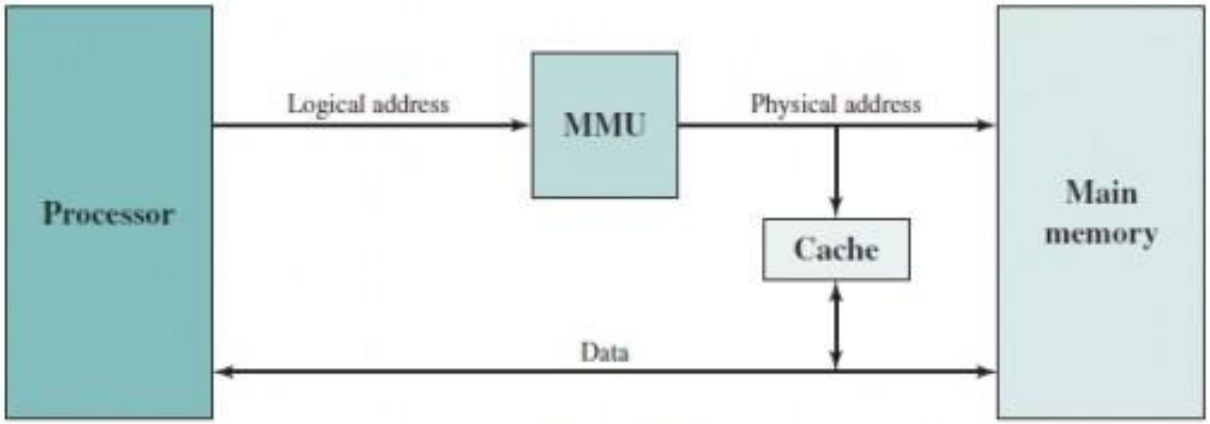
Cache Addressing

- Logical cache & Physical cache:
  - Cache is placed between processor and virtual memory management unit (MMU)
  - Cache is placed between virtual MMU and main memory
  - For data transfer to and from main memory, the MMU translates each virtual address into a physical address in main memory
  - Logical cache (virtual cache) stores data using virtual memory addresses
  - Processor accesses cache directly, not thorough MMU
  - Physical cache stores data using main memory physical addresses
  - Advantage: Logical cache access speed is faster than physical cache
  - Disadvantage: Virtual addresses use same address space for different applications

- Logical cache & Physical cache:



(a) Logical cache



(b) Physical cache

Figure 4.7 Logical and Physical Caches

Cache Size

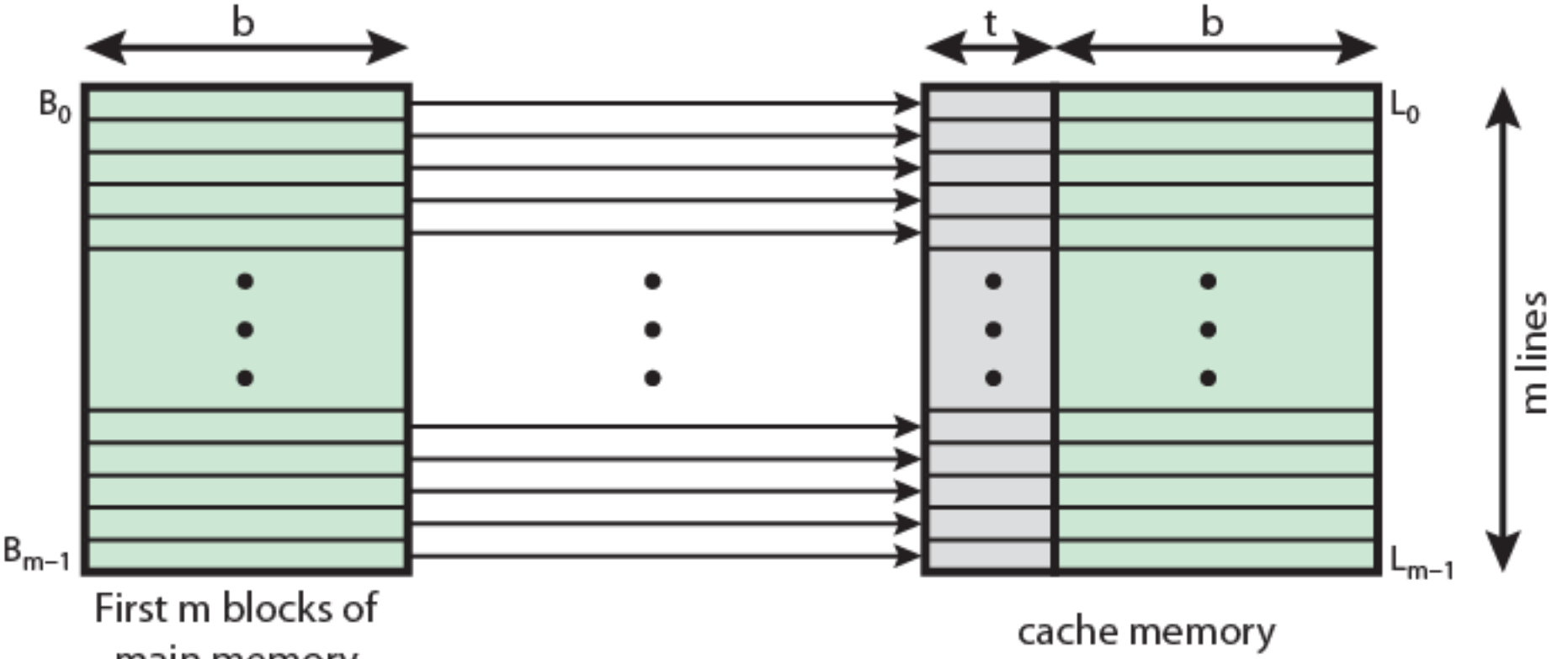
- Size
  - The size of the caches should be small enough so that overall avg. Cost/bit is close to main memory and large enough so that overall avg. access time is close to that of cache
- Cost
  - More size cache is expensive
- Speed
  - Larger cache is slightly slower that smaller cache
  - Checking cache for data takes time

Mapping Function

- Direct mapping
- Associative mapping
- Set – associative mapping

Direct Mapping

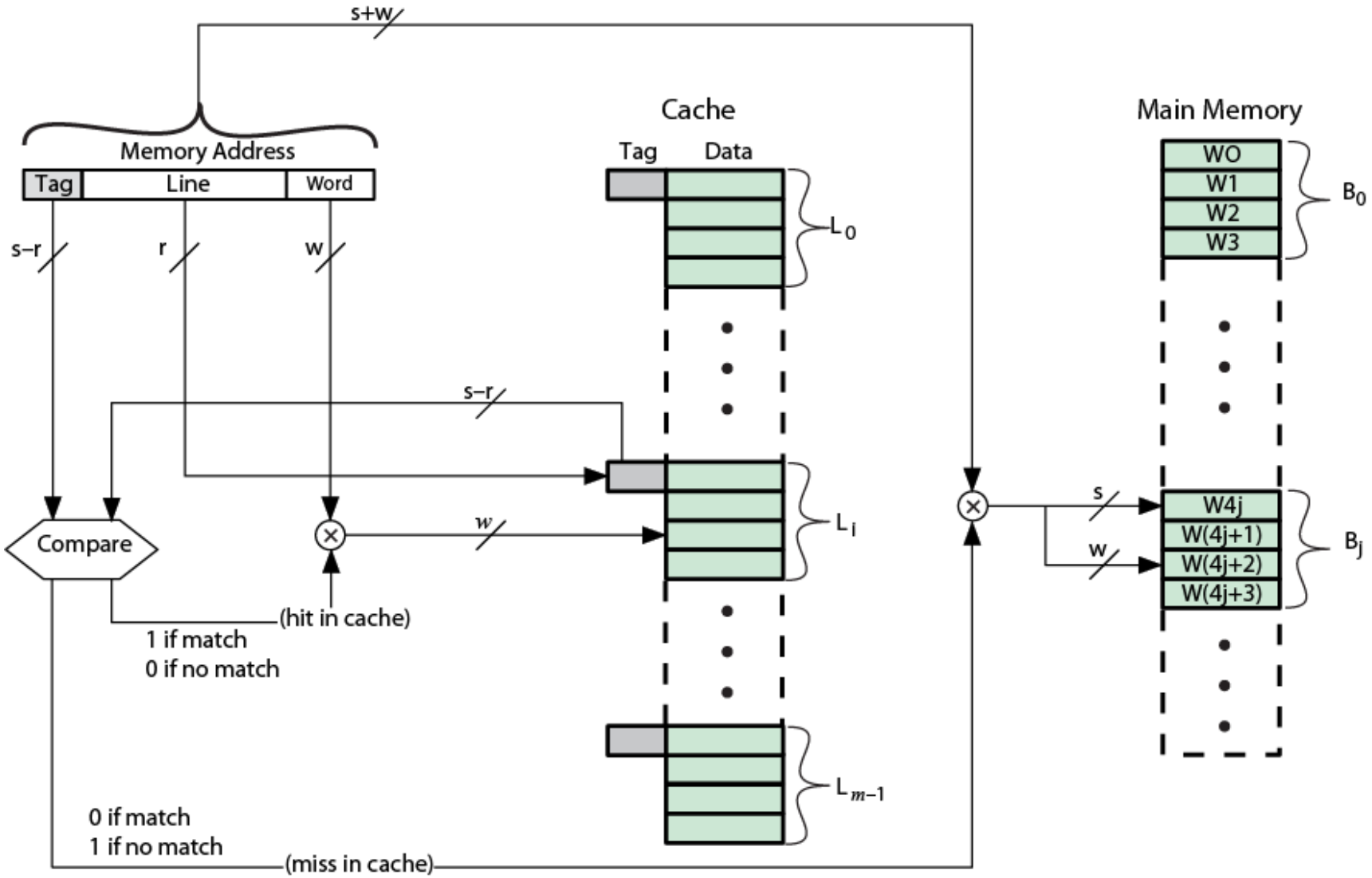
- Each block of main memory maps to only one possible cache line
  - i.e. if a block is in cache, it must be in one specific place
  - The figure shows mapping of first (m) blocks of main memory into each line of the cache
  - The next (m) blocks of main memory maps into the cache in same fashion
- The mapping function is easily implemented using main memory address in two parts
- 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)



(a) Direct mapping

Direct Mapping Cache Line Table

Cache line	Main Memory blocks held
0	0, m, 2m, 3m…2s-m
1	1,m+1, 2m+1…2s-m+1
…	
m-1	m-1, 2m-1,3m-1…2s-1


s+w
Memory Address
Tag
Line
Word
s–r
r
w
Cache
Tag
Data
L_0
L_i
L_{m-1}
s–r
w
Compare
(hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(miss in cache)
Main Memory
WO
W1
W2
W3
B_0
s
w
W4j
W(4j+1)
W(4j+2)
W(4j+3)
B_j

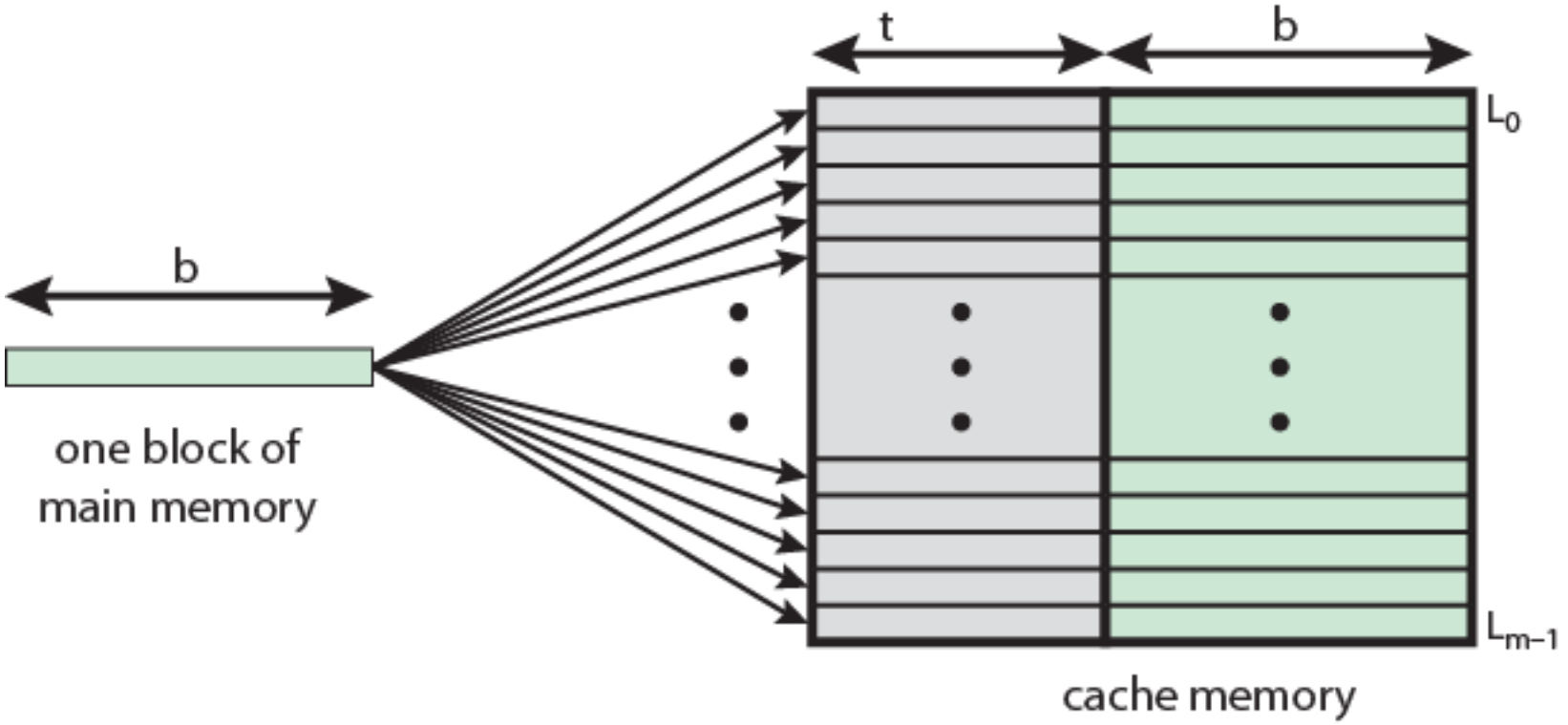
Direct Mapping Summary

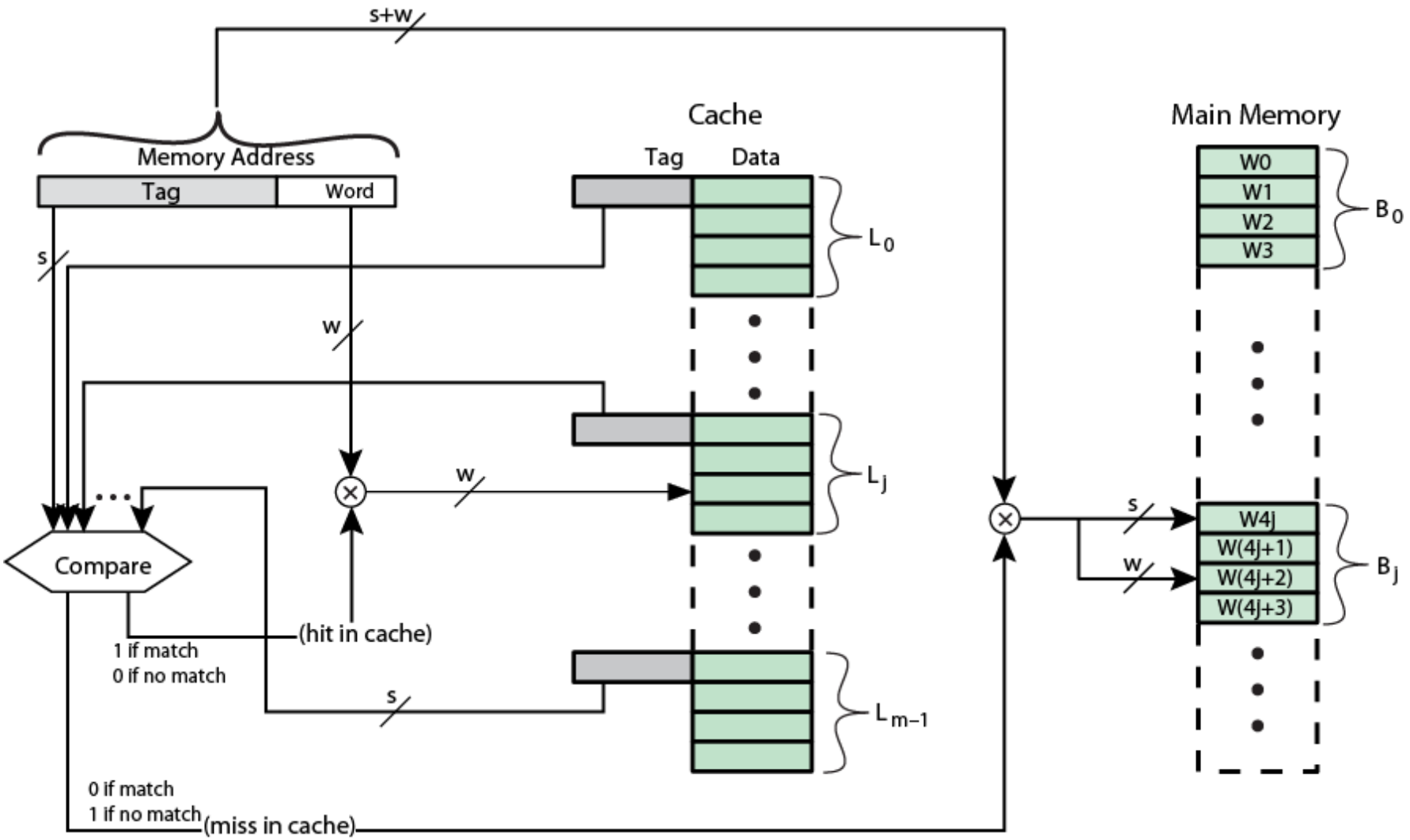
- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+ w/2w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s – r) bits

Associative Mapping

- A main memory block can load into any line of cache
- The cache control logic (CCL) memory address is interpreted simply, as tag and word
- Tag uniquely identifies a block of memory
- To determine whether a block is in cache, every line's tag is examined simultaneously by cache control logic for a match
- Cache searching gets expensive

Associative Mapping from Main Memory to Cache




s+w
Memory Address
Tag
Word
s
w
Cache
Tag
Data
L_0
L_j
L_{m-1}
Main Memory
W0
W1
W2
W3
B_0
W4J
W(4J+1)
W(4J+2)
W(4J+3)
B_J
Compare
1 if match
0 if no match
(hit in cache)
0 if match
1 if no match
(miss in cache)

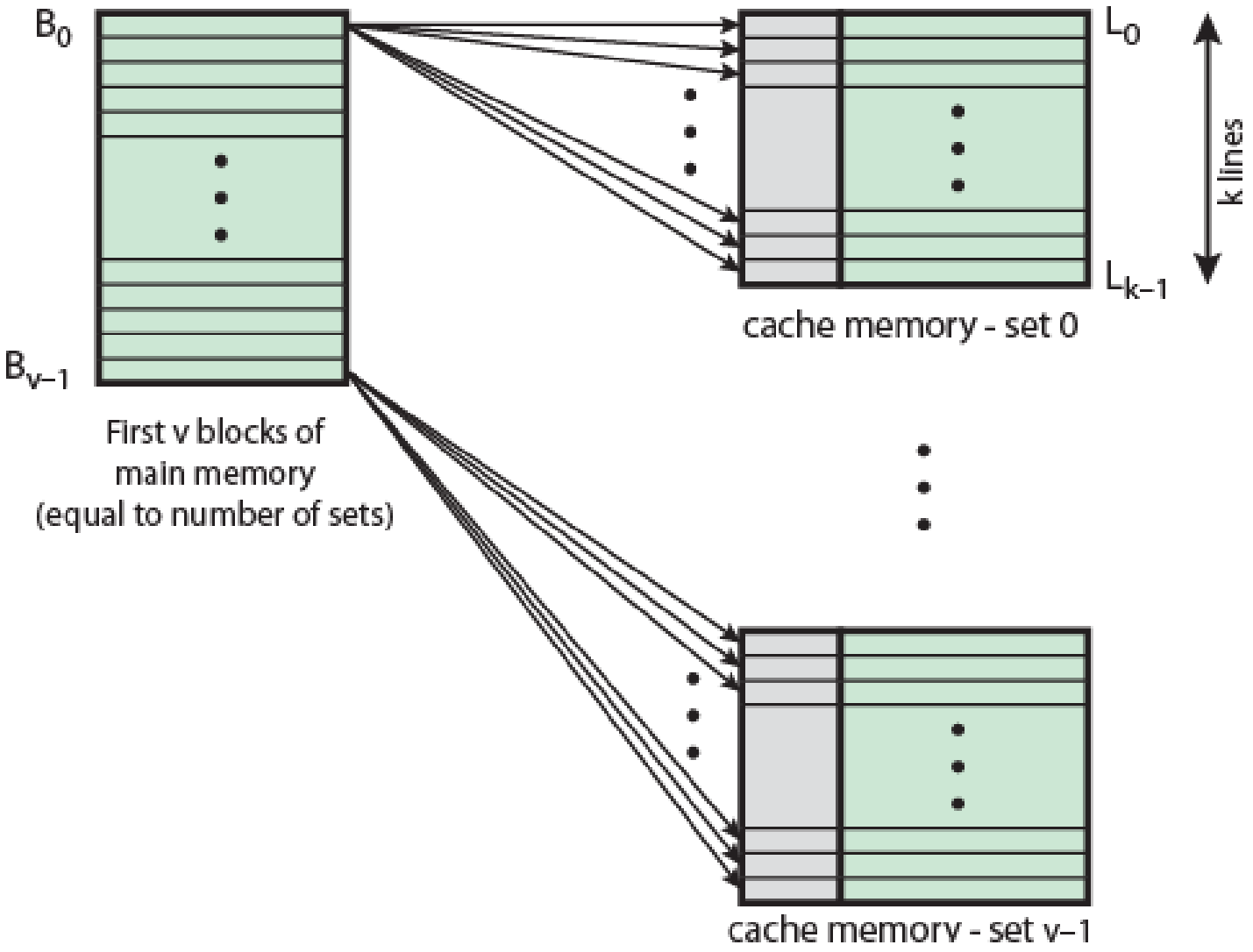
Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2s+w words or bytes
- Block size = line size = 2w words or bytes
- Number of blocks in main memory = 2s+ w/2w = 2s
- Number of lines in cache = undetermined
- Size of tag = s bits

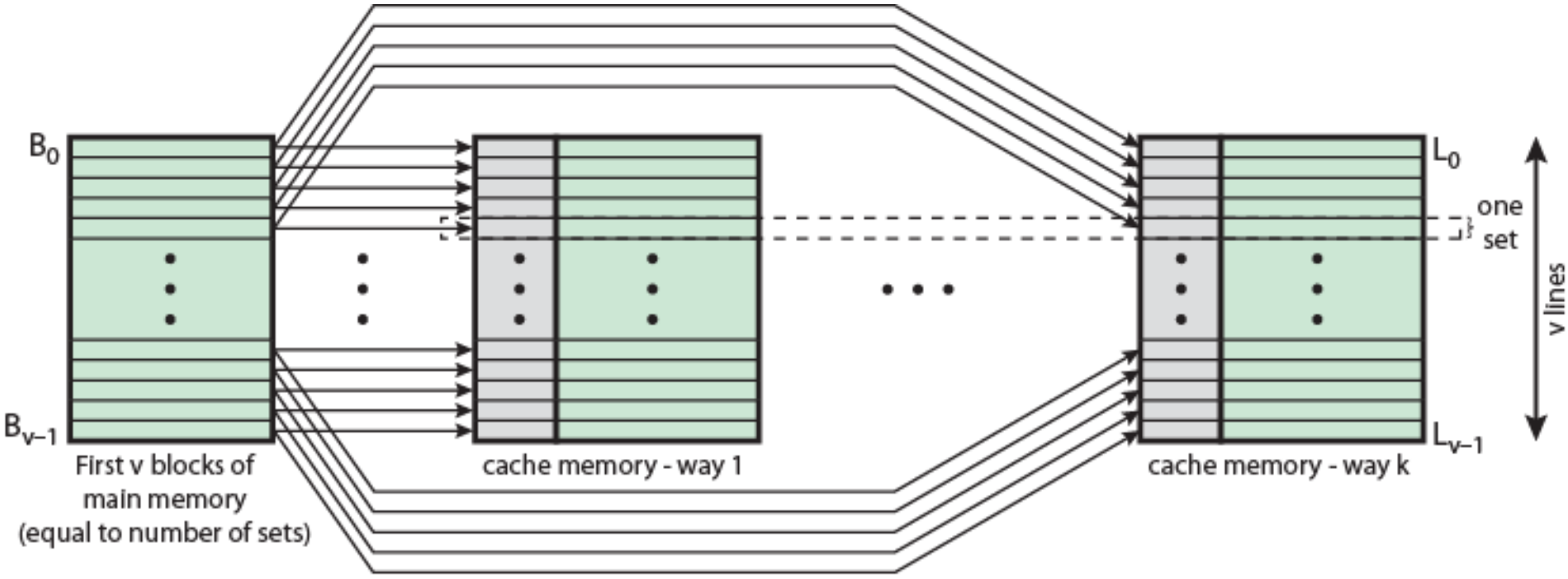
Set Associative Mapping

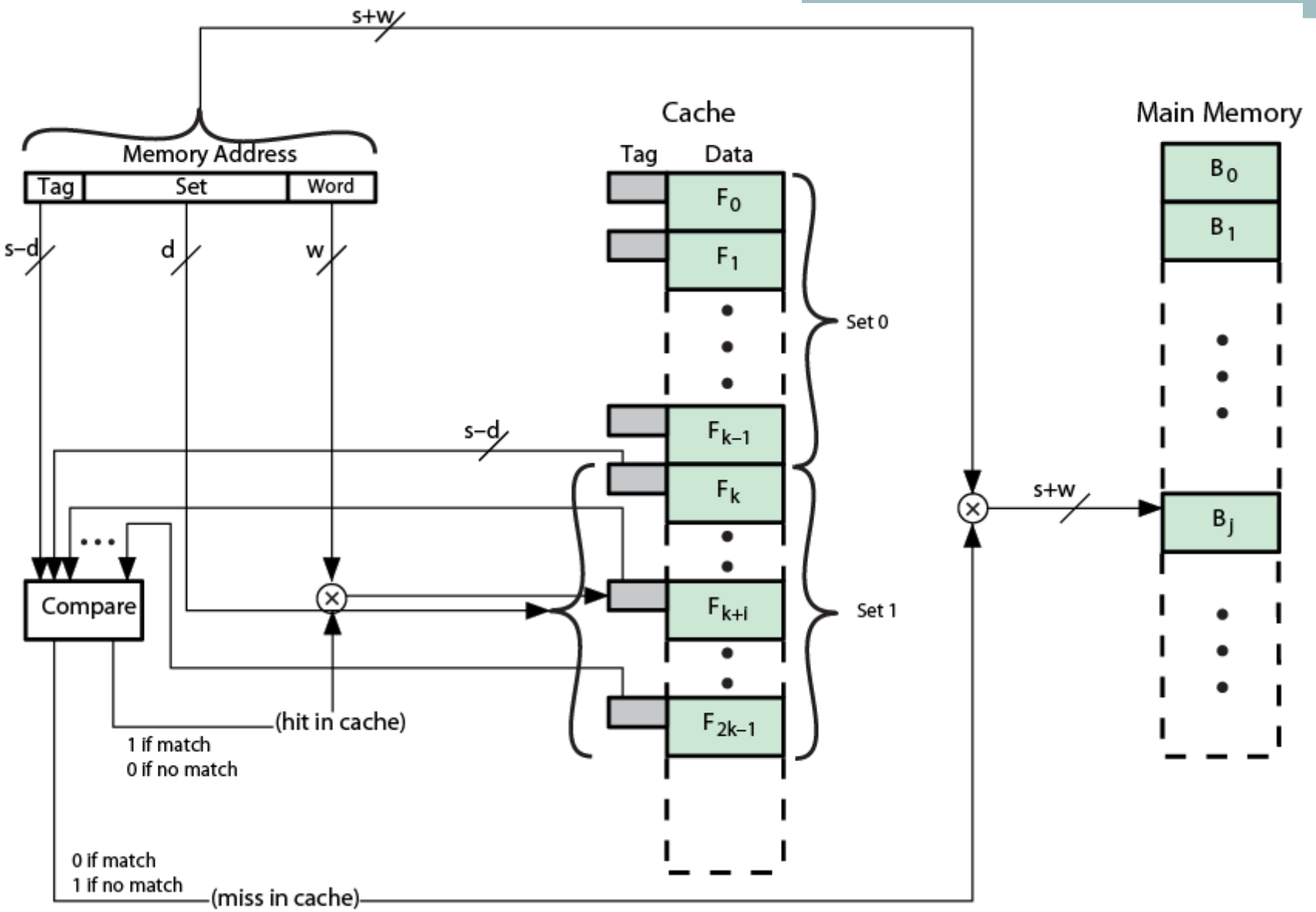
- This mapping technique exhibits strength of both direct and associative mapping techniques
- With v-associative mapping each word from first (v) block maps into multiple cache lines.
- So, each word maps into all the cache lines in a specific set i.e. block B0 maps into set 0 and so on.
- With k-direct mapping, the first (v) lines of the main memory are directly mapped into the (v) lines of each cache memory **way**

Mapping From Main Memory to Cache: v Associative



Mapping From Main Memory to Cache: k-way Associative




s+w
Memory Address
Tag
Set
Word
s–d
d
w
Cache
Tag
Data
F_0
F_1
Set 0
F_{k-1}
F_k
F_{k+1}
Set 1
F_{2k-1}
s–d
Compare
(hit in cache)
1 if match
0 if no match
0 if match
1 if no match
(miss in cache)
s+w
Main Memory
B_0
B_1
B_j

Set Associative Mapping Summary

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = 2^d
- Number of lines in set = k
- Number of sets = v = 2^d
- Number of lines in cache = kv = k * 2^d
- Size of tag = (s – d) bits

Replacement Algorithms: Direct mapping

- Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks in the cache must be replaced.
- For direct mapping, each block only maps to one line and no choice is possible.
- So in this, that line is replaced entirely.

Replacement Algorithms: Associative & Set Associative

- Unlike direct mapping, for associative and set associative mapping, replacement policy is needed.
- Hardware implemented algorithm are used to achieve high speed.
- There are four most commonly used algorithms:
  - Least recently used (LRU)
  - First -in-first-out (FIFO)
  - Least frequently used (LFU)
  - Random

- Least recently used algorithm (LRU):
  - Most effective algorithm is Least Recently used (LRU)
  - Replaces that block in the set that has been in the cache longest (not recently used) with new block which is supposed to be required by CPU.
  - To implement this algorithm, each line in the cache is added with a flag bit, this bit is called as the USE bits.
  - If any one line from a block is used by CPU, the USE bit of all the lines from that block is set to 1. By looking at value of USE bit, we can find out which block is recently used and which is not.
  - The block whose value of USE bit is O, gets replaced by new block.
  - LRU is the most popular algorithm as it is the simplest algorithm to implement.

- First in first out (FIFO):
  - This algorithm replaces the block which is there in cache memory for the longest time. This algorithm is implemented as Round Robin technique.
- Least frequently used (LFU):
  - Least frequently used replace block which has had fewest hits.
  - This algorithm is used when the program contained in the main memory contains non repetitive, sequential set of instructions. LFU replaces the block from cache which has experienced fewer references.
- Random:
  - This method, randomly replaces a line from cache memory and allocates new line from main memory.
  - There are some simulated results showing random method being slightly inferior to the other above-mentioned methods.

Paging and page re-placement policies:

- Paging & Page Replacement Policy:
  - The main memory available in the computer system is usually partitioned to contain the OS programs and the user programs.
  - Multiple processes are run simultaneously to improve the system performance. In such cases, the memory allotted for user programs needs to be divided and sub divided to ensure that simultaneous parallel tasks can be carried out. This task of division and subdivision is performed by the OS and is called as Memory Management.
  - Creating these divisions with a fixed size can lead to waste of memory for smaller tasks whereas creating unequal divisions will make it difficult to access locations since their starting addresses won't be deterministic.

contd.....

- In order to avoid these issues, not only the memory but the programs or processes are also divided I. e. Paging is performed.
- Paging involves the division of process/program or data into much smaller pieces of fixed size called as pages.
- The memory is also divided into equal sized chunks called as, Page Frames or Frames. The pages are then assigned to page frames. This ensures minimal wastage of memory, since the only frame that will have memory wasted will be the frame carrying last page of the process and that last page does not require entire frame memory locations.

- Page faults:
  - The virtual memory space is divided into fixed size Pages and the physical memory space is divided in to same fixed sized Page Frames.
  - However, the number of pages in virtual memory space are much bigger in number as compared to the number of page frames in the physical memory space and therefore, only a few pages can be accommodated in the available page frames in the physical memory space.
  - Therefore, two issues are confronted in loading and unloading the pages in page frames they are:
    1. Which pages would be loaded in the page frames in the physical memory space, I. e. Page Allocation.
    2. Which pages moved or unloaded when a new page is needed to be loaded and there is no page frame empty in the physical memory space. I. e. Page Replacement.

contd.....

- Page Faults:
  - When the Processor or CPU demands an element of program code or data and it is currently NOT available in the pages already loaded in the page frames in the physical memory space, processor or CPU generates an exception during the process of page address translation mechanism. This exception is called as Page Fault.
  - The Page Fault indicates that the data or code element needed by processor or CPU is currently not available in the physical memory space.
  - In other words, the page containing that specific code or data element is not there in any of the page frames in the physical memory space.
  - The page is lying out in the virtual memory space (which is on the secondary memory such as Hard disk).

- Page replacement policies:
  - Page replacement policy is the policy that decides on which pages would be removed or unloaded when a new page is needed to be loaded and there is no-page frame empty in the physical memory space I. e. the physical memory space is fully occupied.
  - Choosing the right page to go out has its impact on performance as the OS should select that specific page qualify to go out which is most unlikely to be required again in the physical memory space ; in the near future.
  - For this number of policies are devised:
    1. Least Recently Used (LRU)
    2. First-In-First-Out (FIFO)
    3. Least Frequently Used (LFU)
    4. Optimal (OPT)
    5. Random (RND)

- Least Recently Used (LRU):
  - In this policy, the page that is used oldest or least recently is the one which qualifies to go out of physical memory space.
  - This operates on the notion that it is highly unlikely that the page would be required in the near future if the page has not been referred to in the recent past.
  - This policy provides reasonably good hits and is associated with low hardware resource requirements to implement the policy logic.
  - This is one of the most widely used page replacement policy on the currently used paging virtual memory systems.

- First-In-First-Out (FIFO):
  - In this policy, the page that is comes in first is the page that is replaced first and the one that qualifies to go out of the physical memory space.
  - This operates on the concept that it newer and newer pages are needed to be loaded and earlier pages (older pages) are less likely to be used again as most probably usage of those pages has stopped.
  - This policy provides reasonably good hits and is associated with quite low hardware resource requirements to implement the policy logic.
  - This is the second most widely used page replacement policy after LRU, on the contemporary paging virtual memory systems.

- Least Frequently Used (LFU):
  - In this policy, a track is kept on the number of times every page currently in the physical memory space is used and the one which is used minimum number of times is the page that qualifies to go out of the physical memory space.
  - As the policy involves complex hardware such as counters for every page frame and it gives similar hits as that of LRU policy, this policy is less widely used.

- Optimal (OPT):
  - This policy is hypothetical policy as it looks in to the future and checks what pages that would be needed (i.e. upcoming page loading requirements) and accordingly replaces pages from the current situation optimally, in such a way that page hits are maximized and page faults are minimized.
  - This is the best but unrealizable policy.
  - The physical implementation of this policy is not possible as it is not feasible to estimate page loads in the future.
  - This policy is applied in the post-scenario to benchmark quality of the implemented policy.
  - The closer is the implemented replacement policy to the results obtained by OPT, better is the quality of that policy.

- Random (RND):
  - In this policy, a page currently-loaded in the page frame, is selected randomly and it qualifies to go out of the physical memory space.
  - There is no other logic or algorithm that is implemented for the page replacement. It only needs a Pseudo Random Number Generator (PRNG) to generate a random number for the page to be replaced.
  - This policy is used sometimes in the paging system as it doesn't involve any per page frame hardware and therefore works out to be very light-weight.

The links to you tube video which will help you to solve the problem based questions:

1. https://youtu.be/VePK5TNgQU8 (Link for explanation on direct mapping function)
2. https://youtu.be/Trd5XxEk2jU (Link for solved problem on direct mapping function)
3. https://youtu.be/3eriC-pIQKg (Link for explanation on associative mapping function)
4. https://youtu.be/yH_AHVAaBM4 (Link for solved problem on associative mapping function)
5. https://youtu.be/mCF5XNn_xfA (Link for explanation on set-associative mapping function)
6. https://youtu.be/j5PUJllPPVE (Link for solved problem on set-associative mapping function)
7. https://youtu.be/Ub4VVDGLJx0 (Link for solved problem on page replacement policy)

Write Policy

- CPU generates an address when it tries to write some data into memory.
- If memory block containing that address is present in cache, it modifies cache or else that memory block is first brought into cache and then it is modified into cache memory itself.
- This modified data needs to be updated back into main memory as any other I/O device might need to use the data.
- 'If at least 'one memory write operation is performed on a word line of cache, the main memory should be updated with the updated data before bringing in new block into cache.

Write Policy

- There are two main situations needed to be taken care of. First, one I/O device along with CPU has read-write access to memory.
- If a word is altered only in cache, then the corresponding memory word is invalid. Or, if I/O device has altered main memory, the word in cache is invalid.
- A more complex situation arises when one memory is shared with multiple processors attached to a same bus and each processor has its own local cache.
- There are two techniques to address the above mentioned problem.

(1) Write Through

(2) Write Back

Write through

- This is a very simple technique. In this technique a write operation is made both to cache and main memory, thus ensuring the cache and main memory are both updated and synchronized.
- There is a disadvantage of this technique. This technique creates huge amount of memory traffic resulting into a bottleneck.
- This slows down writes policy.
- Remember bogus write through caches!

Write back

- In this method, the update is made only in cache memory.
- There is an additional flag bit called as DIRTY bit or USE bit. If an update is made in cache memory, the flag bit associated with that particular line is set to 1.
- When a block is getting replaced, the changes will be written into main memory only if the USE bit of the line is set.
- Also, if I/O tries to access main memory, first the USE bit of the corresponding word line in cache memory will be checked.
- If the bit is set, the updated value from the cache memory will be used else the previous memory value will be taken.
- The main drawback is, entire memory access happens through cache memory, which increases circuit complexity and results in bottleneck.

semiconductor main memories

- Semiconductor main memory
  - Early computers used doughnut shaped ferromagnetic loops called cores for each bit
  - Main memory was often referred to as “core” memory or just “core”
  - Semiconductors are almost universal today

Semiconductor Memory Types

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Semiconductor memory types

- RAM (Random Access Memory):
- Misnamed as all semiconductor memory is “Random Access”
- Time required to access any address is constant and does not depend on previous address accessed.
- Read/Write
- Volatile
- Temporary storage
- Two technologies:
  - Dynamic RAM: Analog device, uses capacitor to store charge
  - Static RAM: Digital device, uses flip-flop logic gates to store state

Dynamic RAM (DRAM)

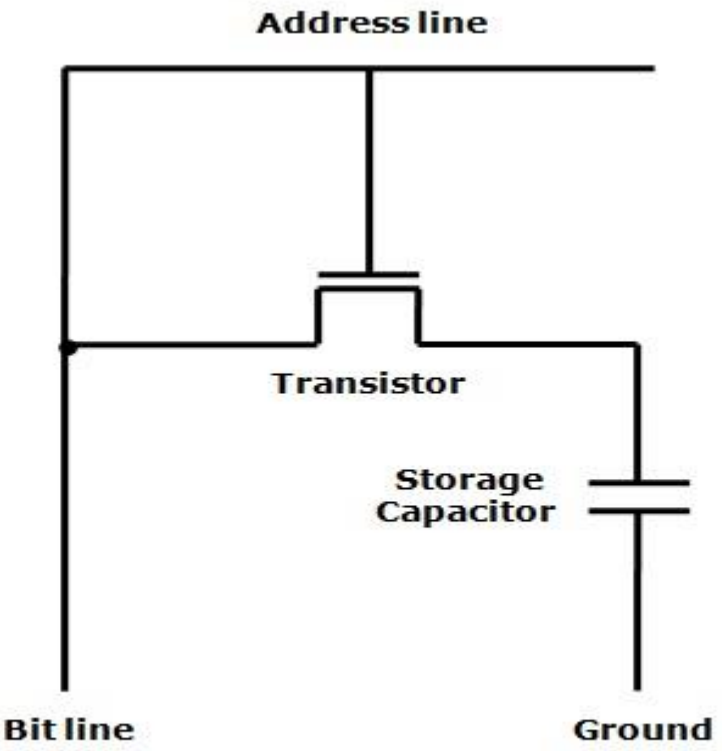
- Bits stored as charge in capacitors
- But charges leak, need refreshing even when powered
- Simpler construction than static RAM (SRAM)
- Slower, but smaller per bit and less expensive than SRAM
- Used for main memory
- Essentially analog rather than digital — Level of charge determines value

Dynamic RAM (DRAM): Operation

- Figure shows a is a typical DRAM structure for an individual cell that stores one bit. The address line is activated when the bit value from this cell is to be read or written.
- The transistor acts as a switch that is closed (allowing current to flow if a voltage is applied to the address line and open (no current flows) if no voltage is present on the address line.
- For the write operation, a voltage signal is applied to the bit line a high voltage represents 1, and a low voltage represents 0.

- A signal is then applied to the address line, allowing a charge to be transferred to the capacitor.
- For the read operation, when the address line is selected, the transistor turns on and the charge stored on the capacitor is fed out onto a bit line and to a sense a amplifier.
- The sense amplifier compares the capacitor voltage to a reference value and determines if the cell contains a logic 1 or a logic 0.
- The readout from the cell discharges the capacitor, which must be restored to complete the operation.
- Although the DRAM cell is used to store a single bit (0 or 1), it is essentially an analog device.

- The capacitor can store any charge value within a range; a thresh-old value determines whether the charge is interpreted as 1 or 0.



Dynamic RAM (DRAM) Cell

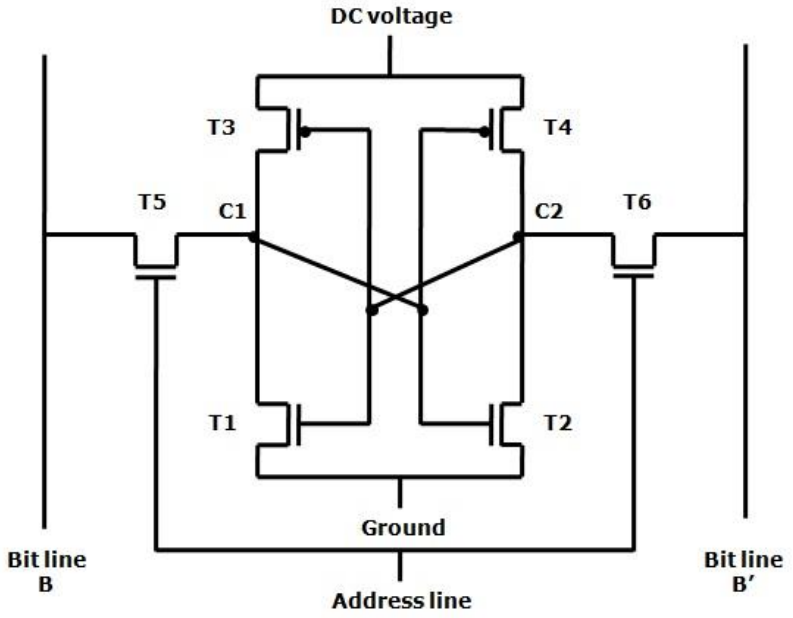
Static RAM

- Bits stored as on/off switches (flip-flops)
- No charges to leak
- No refresh needed when powered
- More complex construction – 6 transistors
- Larger and more expensive per bit, but faster than DRAM
- Used for cache
- True digital device — Uses flip-flops

Static RAM: Operation

- A Static Ram (SRAM), in contrast is a digital device that uses the same logic elements used in the processor.
- In a SRAM, binary values are stored using traditional flip-flop logic-gate configurations. A static RAM will hold its data as long as power is supplied to it.
- Figure shows, a typical SRAM structure for an individual cell. Four transistors (T1, T2, T3, T4) are cross connected in an arrangement that produces a stable logic state.
- In logic state 1, point C1 is high and point C2 is low in this state, T1 and T4 are off and T2 and T3 are on.
- In logic state 0, point C1 is low and point C2 is high, both state T1 and T4 are on and T2 and T3 are off.
- Both states are stable as long as the direct current (dc) voltage is applied. Unlike the DRAM, no refresh is needed to retain data.

- AS in the DRAM, the SRAM address line is used to open or close a switch.
- The address line controls two transistors (T5 and T6). When a signal is applied to this line, the two transistors are switched on, allowing a read or write operation.
- For a write operation, the desired bit value is applied to line B, while its complement is applied to line B'. For a read operation, the bit value is read from line B.



Static RAM (SRAM) Cell

Interleaved memory

- Main memory is composed of a collection of DRAM memory chips.
- A number of chips can be grouped together to form a memory bank.
- It is possible to organize the memory bank in a way known as interleaved memory
- Each bank is independently able to service a memory read/write request
- So that a system with (k) banks can service (k) requests simultaneously, which increases memory read/write rates by a factor of (k)
- If consecutive words of memory are stored in different banks, then transfer of a block of memory speeds up

THANK YOU