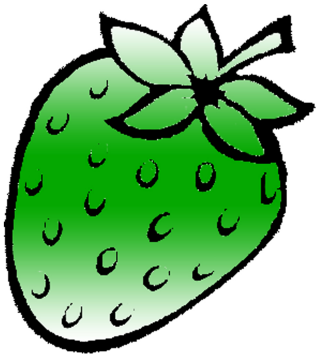
STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com

UNIT III: MEMORY ORGANIZATION

Agenda

- Characteristics
- Memory Hierarchy
- Cache Memory
  - Elements of Cache Design
  - Address Mapping
  - Translation of Cache Memory
  - Replacement Algorithm
- DRAM Organization
- Magnetic Disk: Assignment 1
- RAID
- Magnetic Tape
- Optical Memory: Assignment 1
- High Speed Memory: Cache Memory
- Associative & Interleaved Memory

Characteristics

Table Key Characteristics of Computer Memory Systems

Location
- Internal (e.g. processor registers, main memory, cache) — Main memory
- External (e.g. optical disks, magnetic disks, tapes) — Secondary memory

Capacity
- Number of words
- Number of bytes

Unit of Transfer
- Word
- Block

Access Method
- Sequential
- Direct
- Random
- Associative

Performance
- Access time
- Cycle time
- Transfer rate

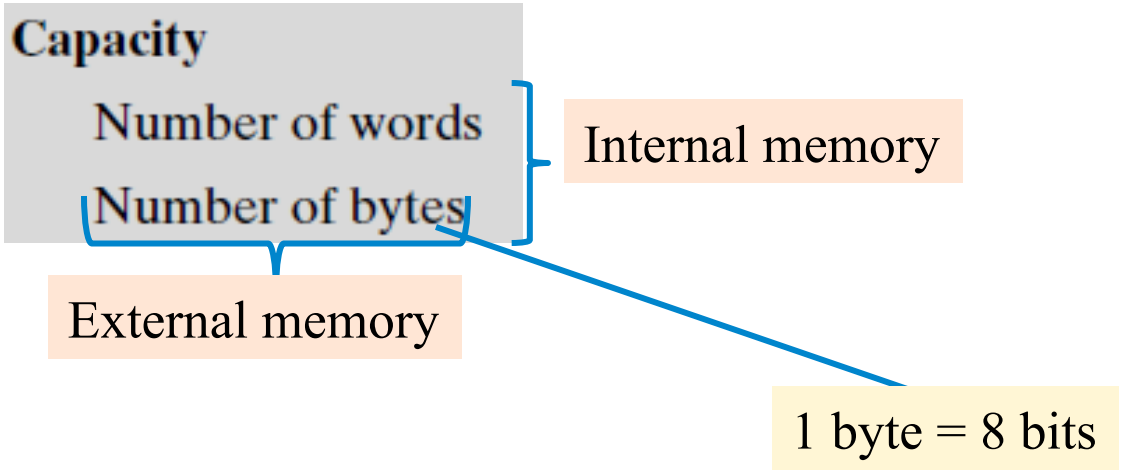
Physical Type
- Semiconductor
- Magnetic
- Optical
- Magneto-optical

Physical Characteristics
- Volatile/nonvolatile
- Erasable/nonerasable

Organization
- Memory modules

Physical arrangement of bits into words

Capacity


Capacity
Number of words
Number of bytes
Internal memory
External memory
1 byte = 8 bits

Unit of Transfer

- Internal
  - Usually governed by data bus width
- External
  - Usually a block which is much larger than a word
- Addressable unit
  - Smallest location which can be uniquely addressed
    - Byte (sometimes)
    - Word internally

Access Methods (1)

- **Sequential**
  - Start at the beginning and read through in order
  - Access time depends on location of data and previous location
  - e.g. tape
- **Direct**
  - Individual blocks have unique address
  - Access is by jumping to vicinity plus sequential search
  - Access time depends on location and previous location
  - e.g. disk

Access Methods (2)

- **Random**
  - Individual addresses identify locations exactly
  - Access time is independent of location or previous access
  - e.g. RAM
- **Associative**
  - a word is retrieved based on a portion of its contents rather than its address.
  - Access time is independent of location or previous access
  - e.g. cache

Performance

- **Access time**
  - Time between presenting the address and getting the valid data
- **Memory Cycle time**
  - Time may be required for the memory to "recover" before next access
  - Cycle time is access + recovery
- **Transfer Rate**
  - Rate at which data can be moved

Physical Types

- **Semiconductor**
  - RAM
- **Magnetic**
  - Disk & Tape
- **Optical**
  - CD & DVD
- **Others**
  - Bubble
  - Hologram

Physical Characteristics

Volatile Memory

- Information decays naturally or is lost when electrical power is switched off.

Non-Volatile Memory

- Information once recorded remains without deterioration until deliberately changed; no electrical power is needed to retain information.

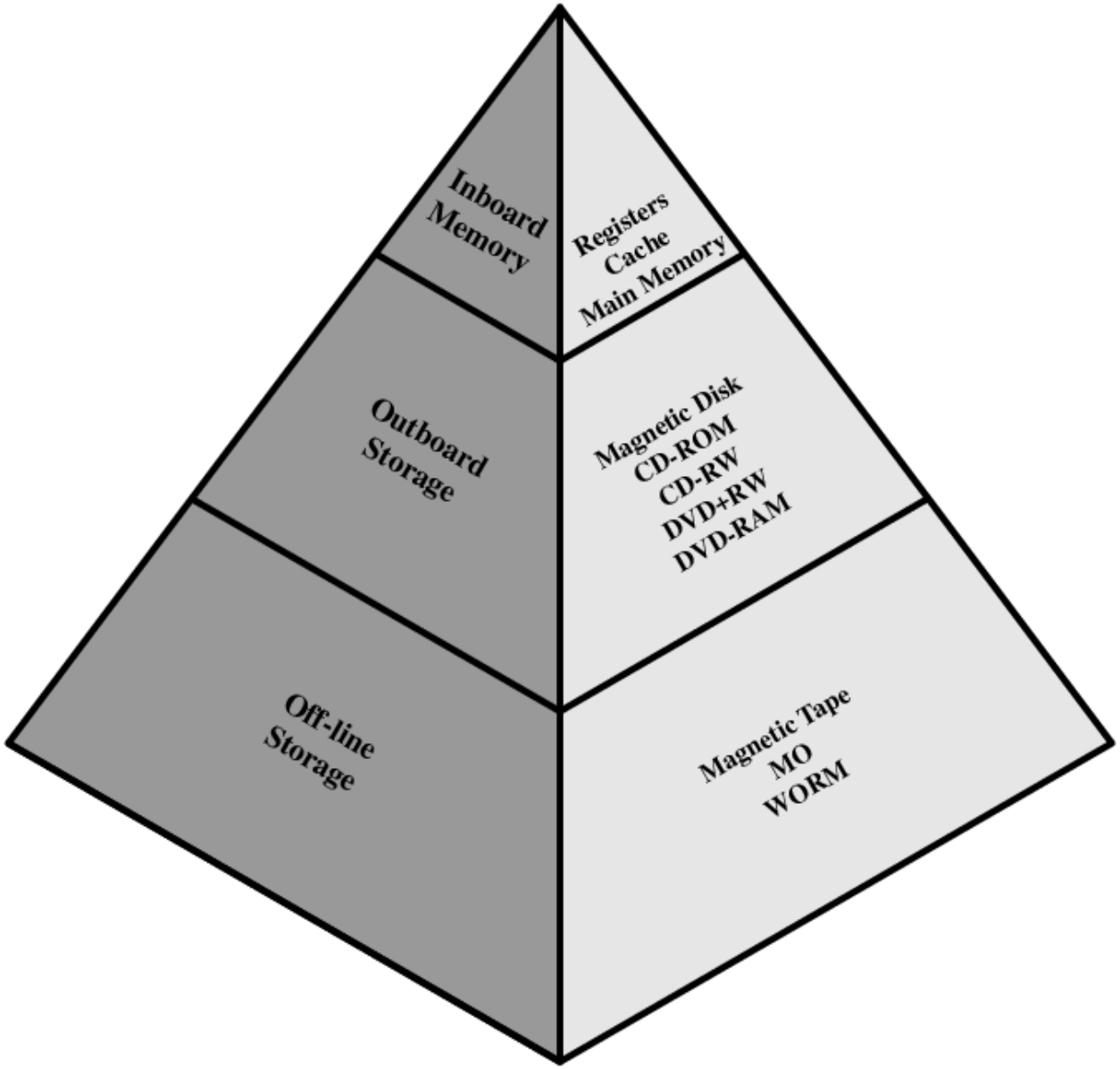
Non-Erasable Memory

- Cannot be altered, except by destroying the storage unit. E.g. *read-only memory* (ROM)

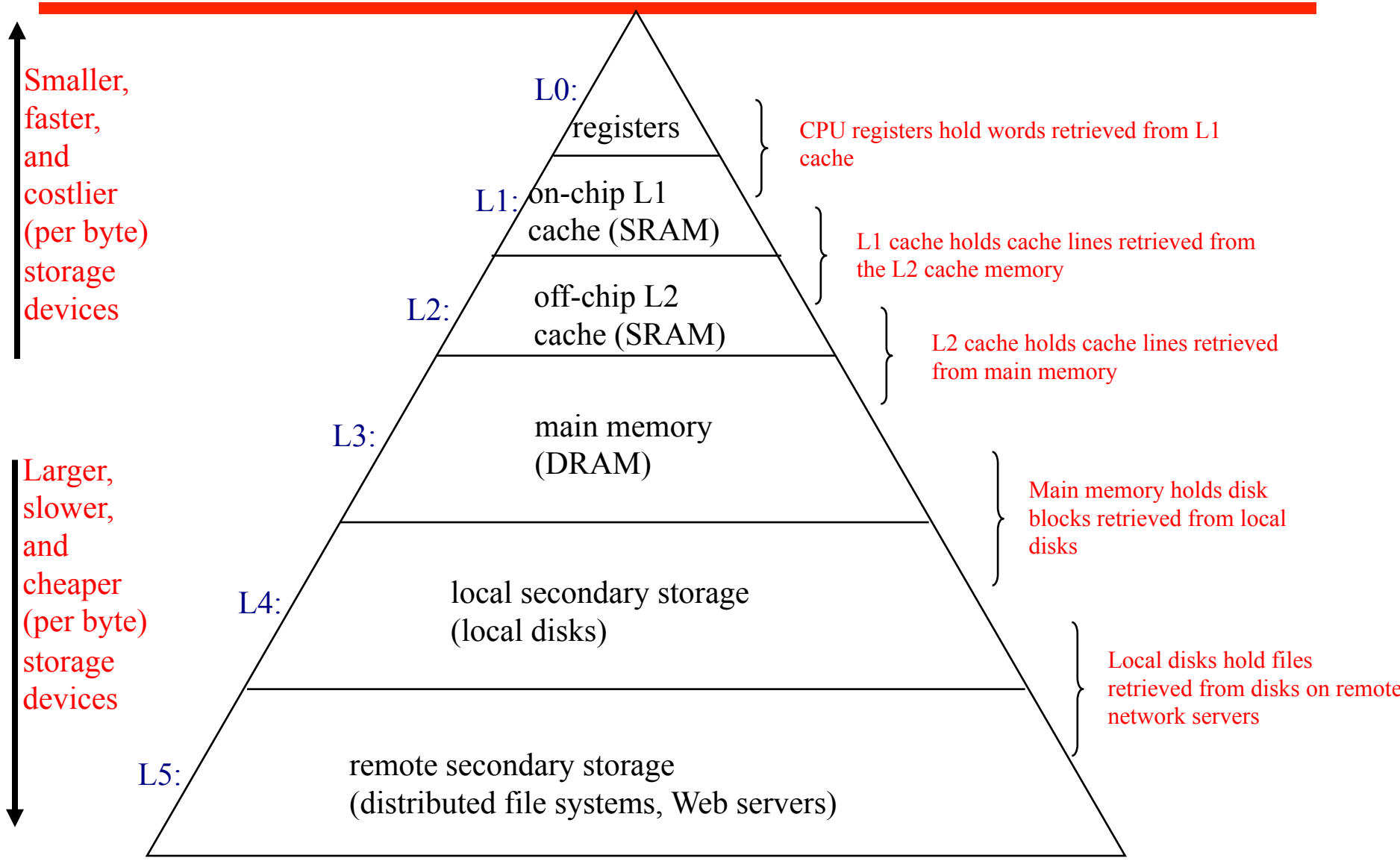
Memory Hierarchy

- How much?
  - Capacity
- How fast?
  - Time is money
- How expensive?

Memory Hierarchy - Diagram



An Example Memory Hierarchy

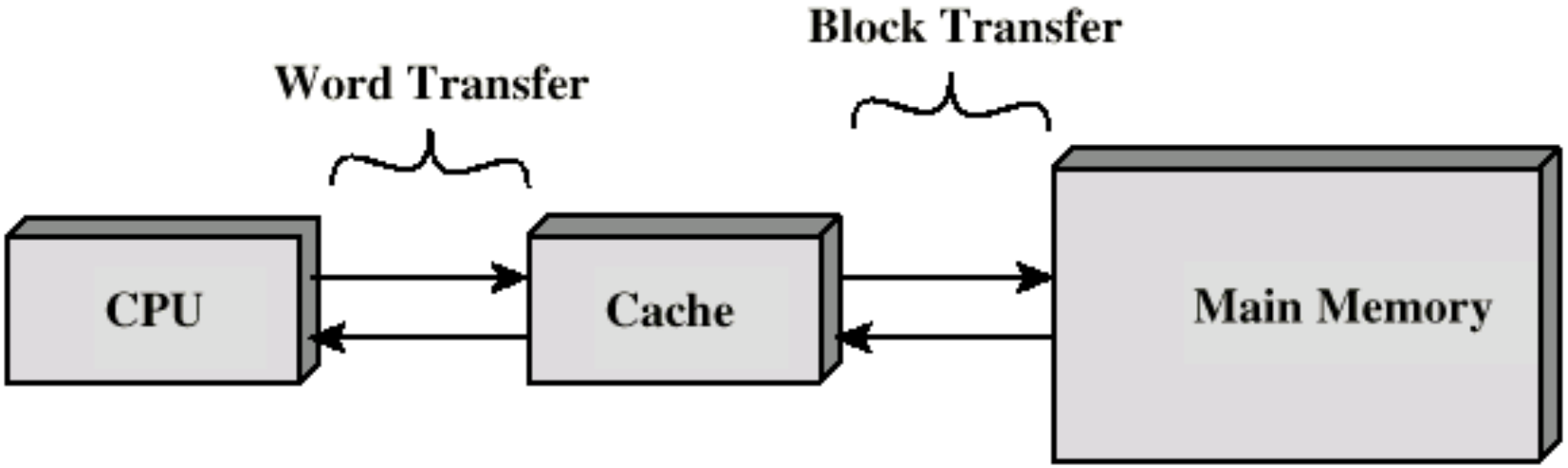


Memory Hierarchy

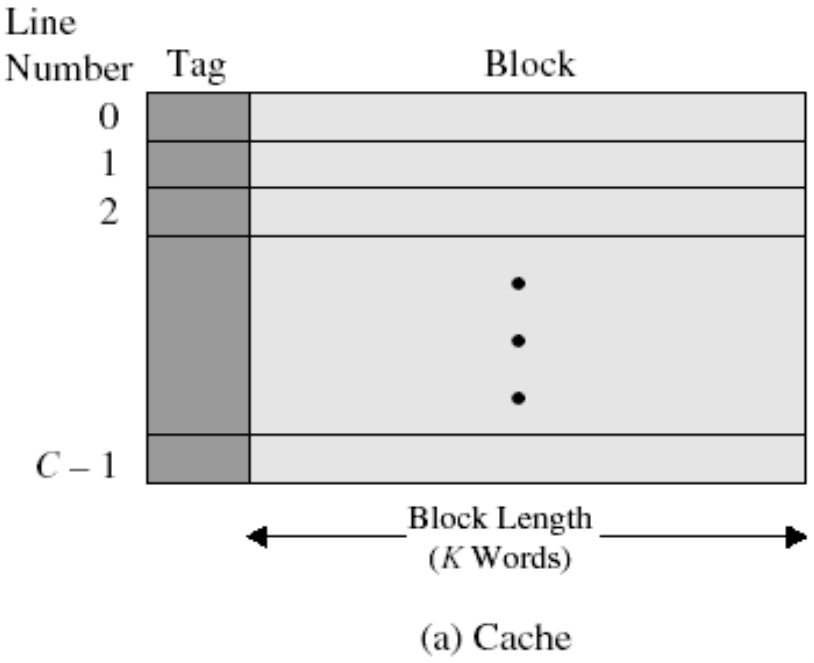
- Registers
  - In CPU
- Internal or Main memory
  - May include one or more levels of cache
  - “RAM”
- External memory
  - Backing store

Cache Memory

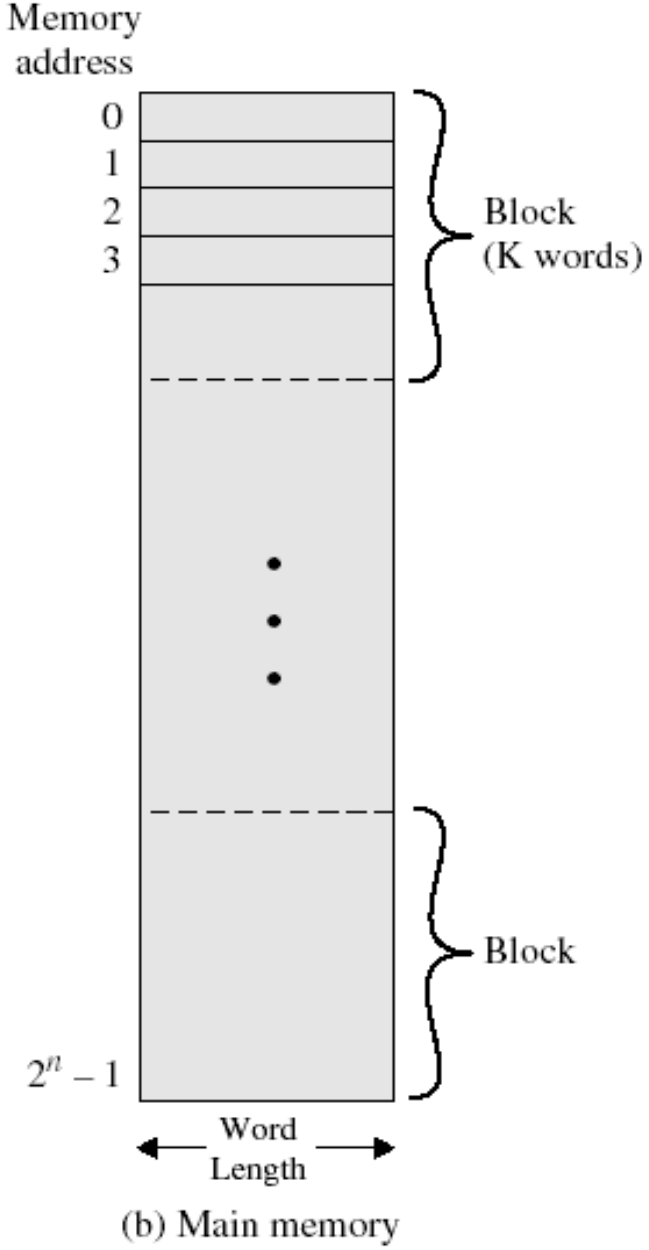
- Small amount of fast memory
- Sits between normal main memory and CPU
- May be located on CPU chip or module



Cache/Main Memory Structure



(a) Cache

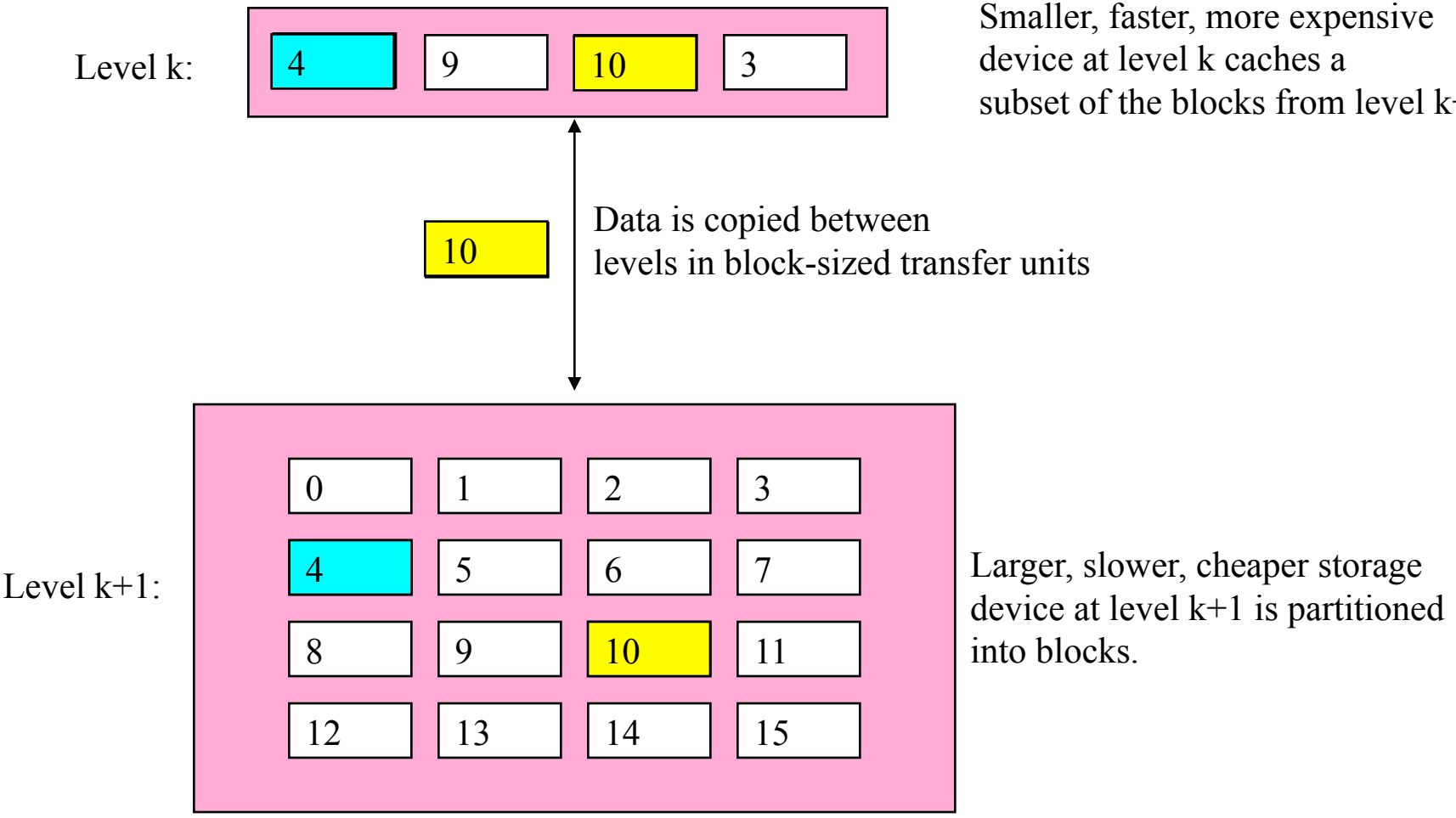


(b) Main memory

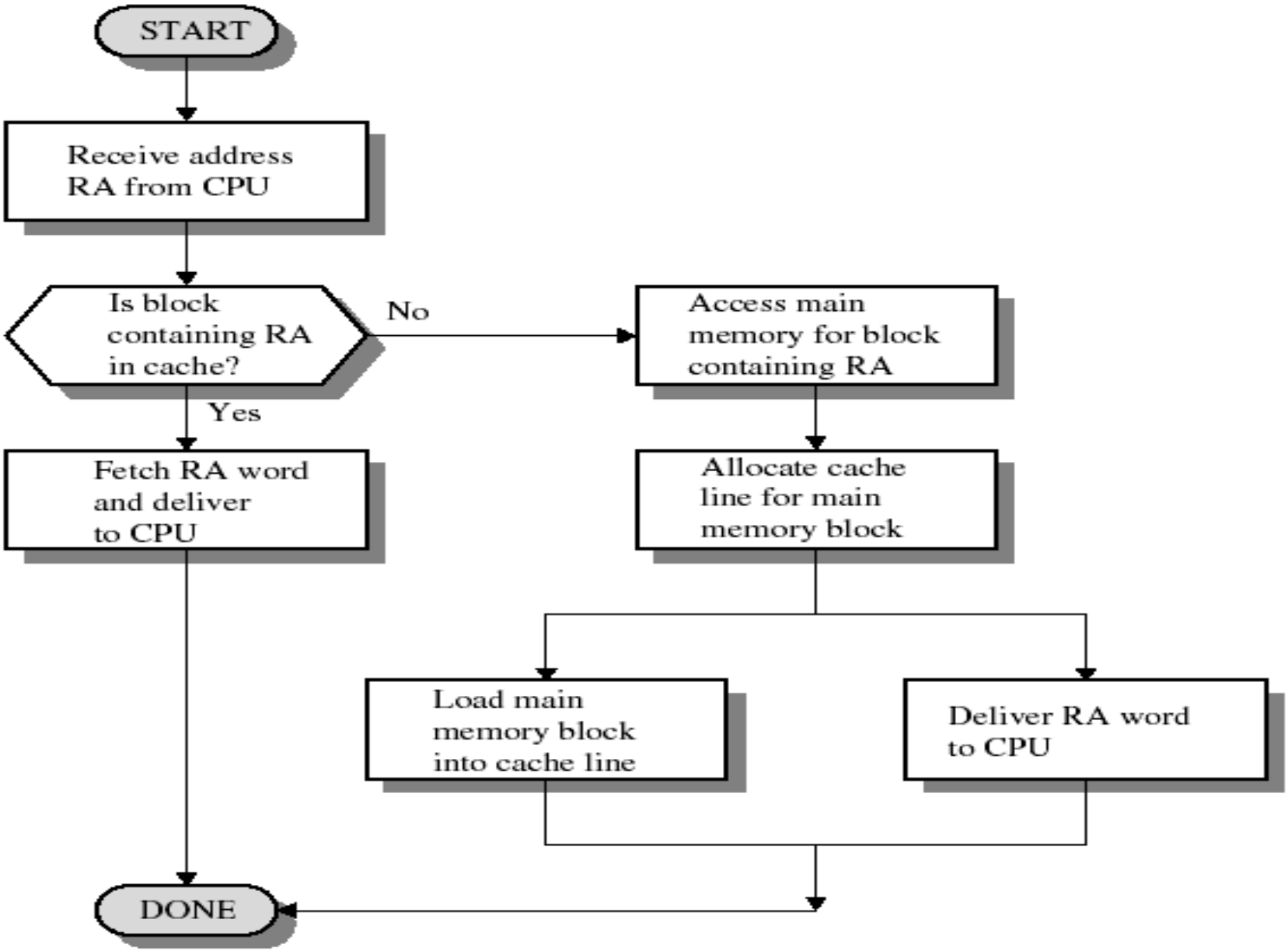
Cache operation - overview

- CPU requests contents of memory location
- Check cache for this data
- If present, get from cache (fast)
- If not present, read required block from main memory to cache
- Then deliver from cache to CPU
- Cache includes tags to identify which block of main memory is in each cache slot

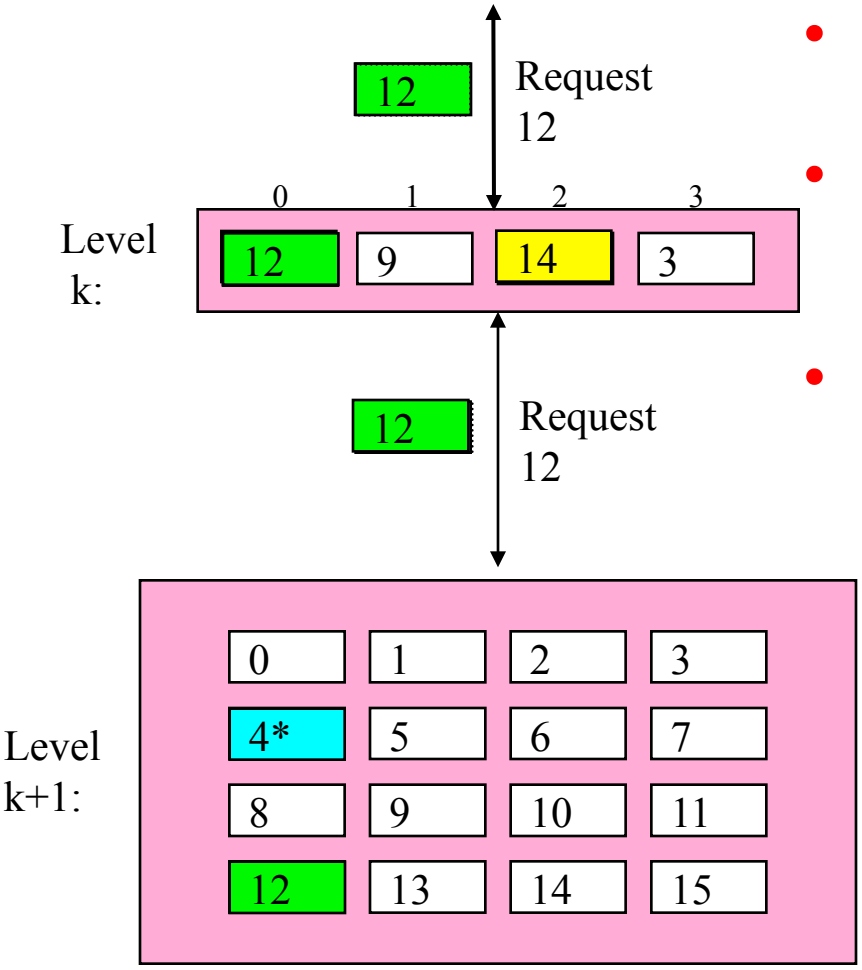
Cache Operation



Cache Read Operation



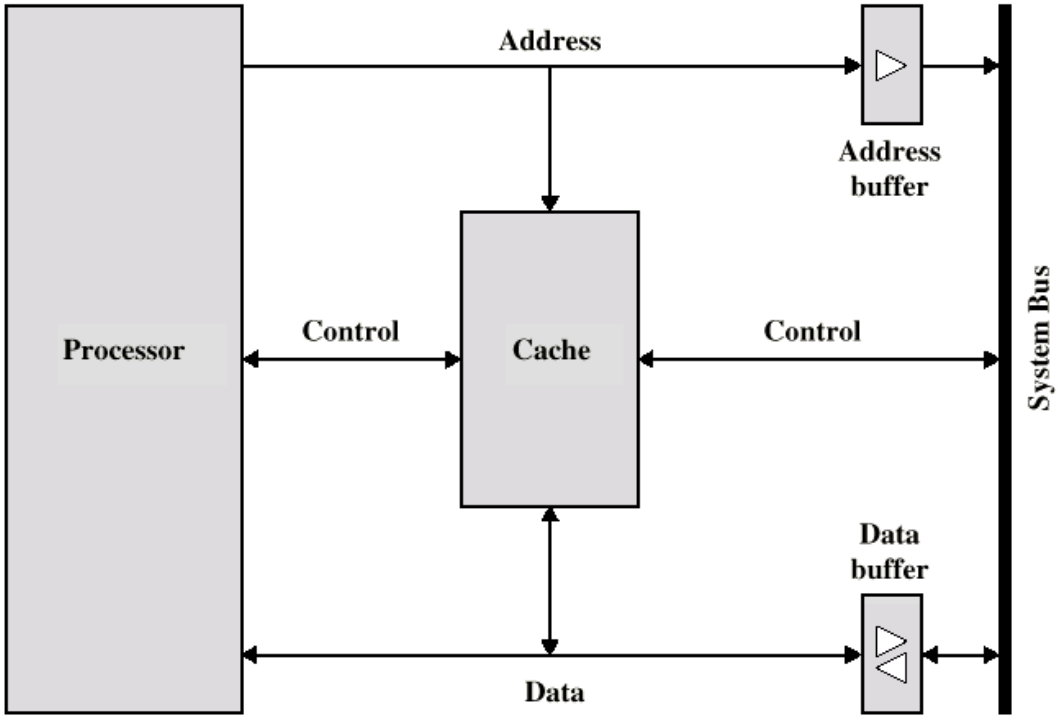
General Caching Concepts



- Program needs object d, which is stored in some block b
- Cache hit
  - Program finds b in the cache at level k. E.g., block 14
- Cache miss
  - b is not at level k, so level k cache must fetch it from level k+1. E.g., block 12
  - If level k cache is full, then some current block must be replaced (evicted). Which one is the "victim"?
    - Placement policy: where can the new block go? E.g., b mod 4
    - Replacement policy: which block should be evicted? E.g., LRU

Typical Cache Organization

- Cache connects to the processor via data, control, and address lines.
- The data and address lines also attach to data and address buffers, which attach to a system bus from which main memory is reached.
- When a cache hit occurs, the data and address buffers are disabled and communication is only between processor and cache, with no system bus traffic.
- When a cache miss occurs, the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.



Elements of Cache Design

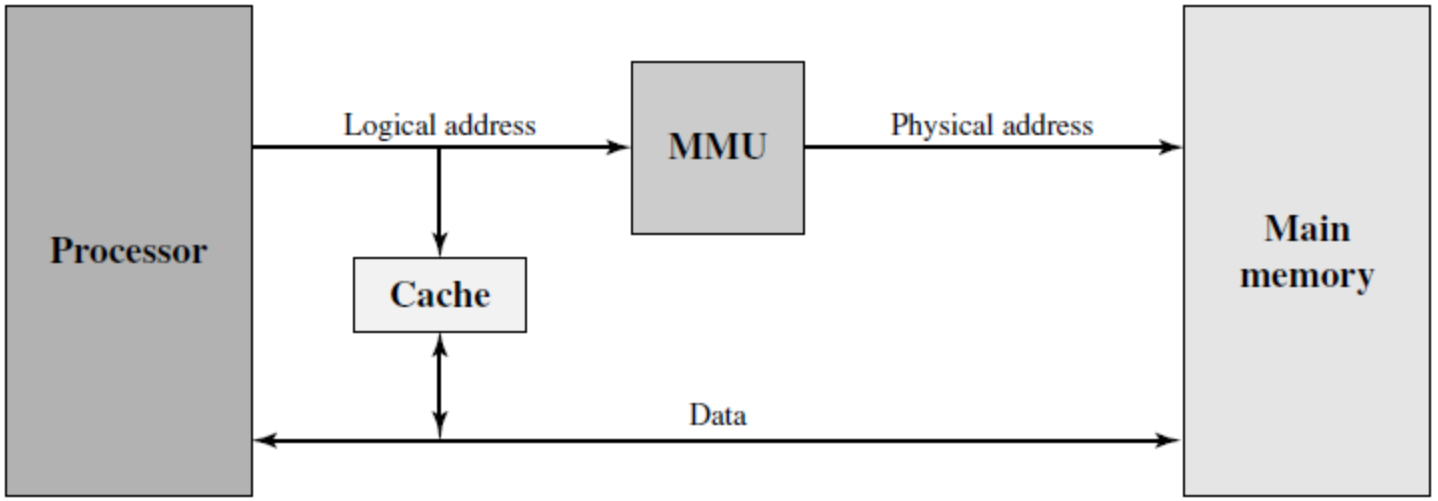
Table Elements of Cache Design

Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	Write once
Mapping Function	**Line Size**
Direct	**Number of caches**
Associative	Single or two level
Set Associative	Unified or split
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

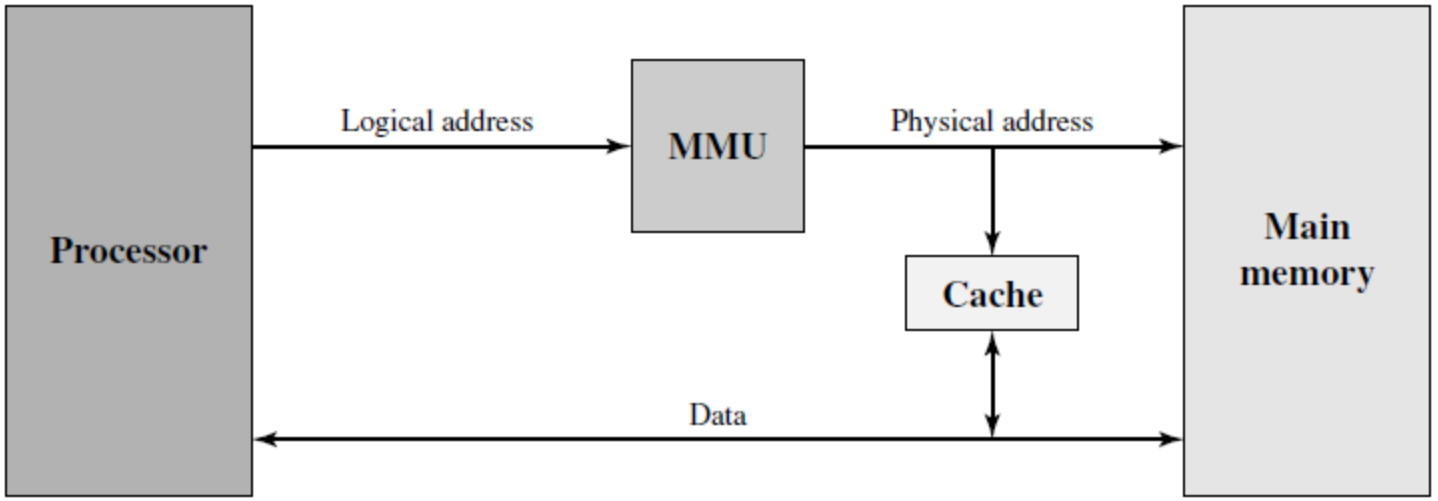
Cache Addresses

- Virtual memory is a facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available.
- When virtual memory is used, the address fields of machine instructions contain virtual addresses.
- For reads to and writes from main memory, a hardware memory management unit (MMU) translates each virtual address into a physical address in main memory.
- A **logical cache**, also known as a **virtual cache**, stores data using **virtual addresses**. The processor accesses the cache directly, without going through the MMU.
- A **physical cache** stores data using main memory **physical addresses**.

Cache Addresses



(a) Logical cache



(b) Physical cache

Cache Addresses

- **Advantage** of the logical cache is that cache access speed is faster than for a physical cache, because the cache can respond before the MMU performs an address translation.
- **Disadvantage** has to do with the fact that most virtual memory systems supply each application with the same virtual memory address space. That is, each application sees a virtual memory that starts at address 0. Thus, the same virtual address in two different applications refers to two different physical addresses. The cache memory must therefore be completely flushed with each application context switch.

Cache Size

- We would like the size of the cache to be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

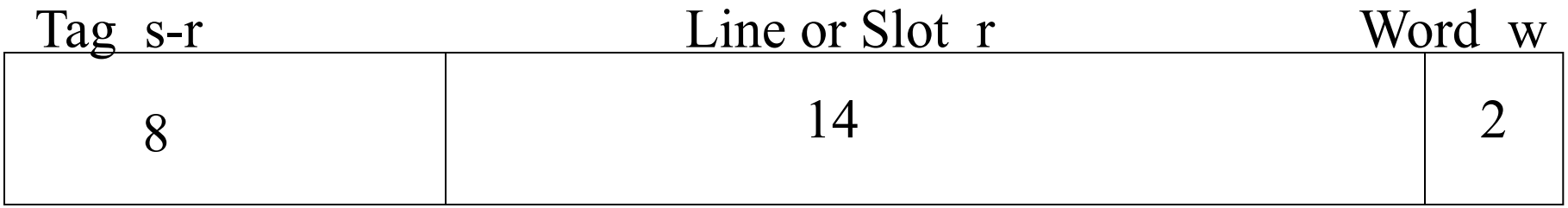
Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

Direct Mapping

- 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
- Address is 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 (most significant)

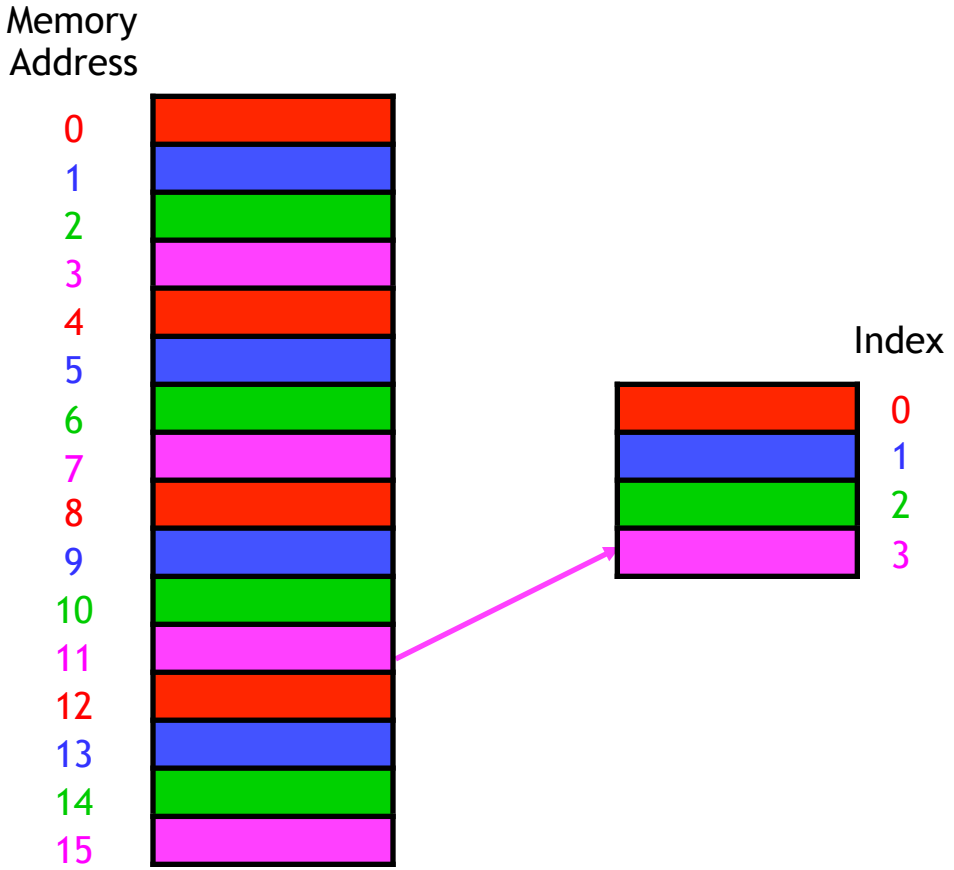
Direct Mapping: Address Structure



- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

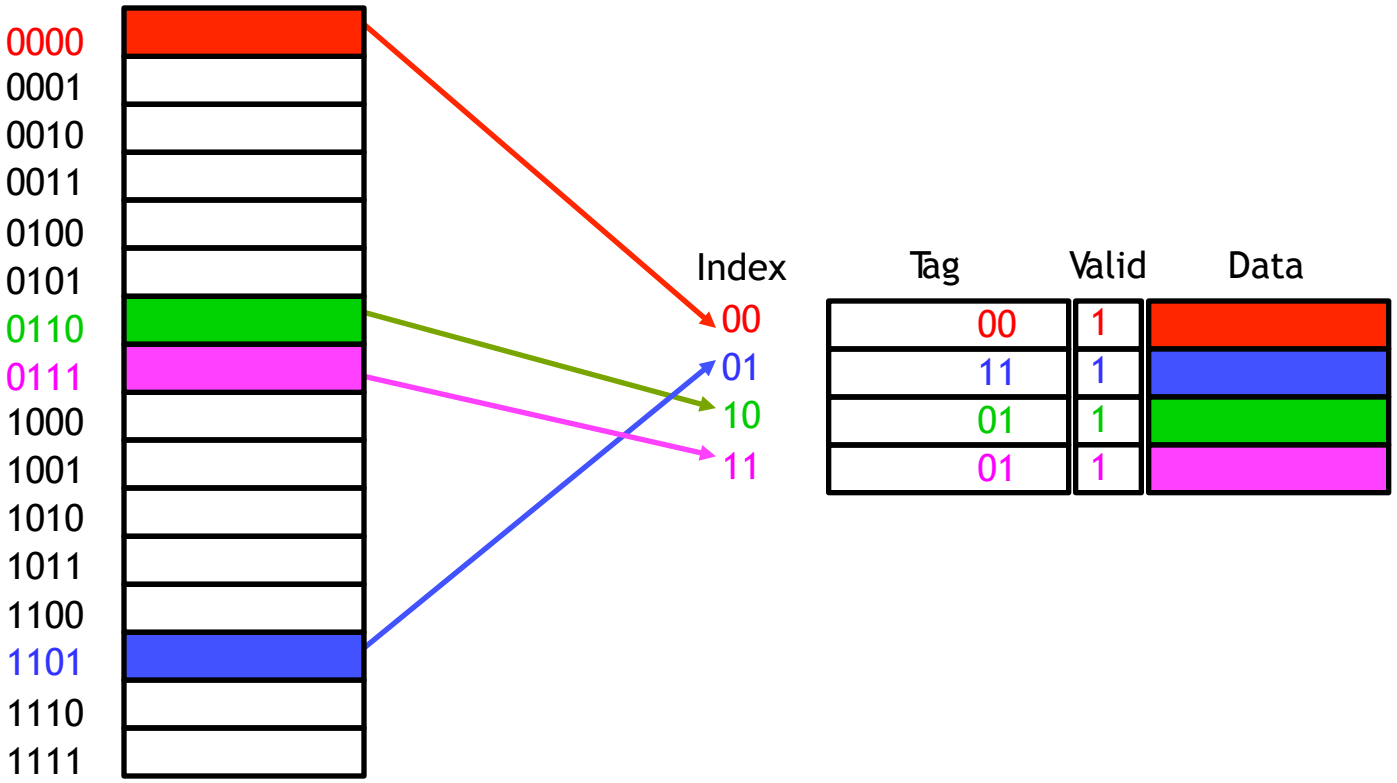
Direct Mapping: Index/Line

- **If the cache contains 2^k bytes, then the k least significant bits (LSBs) are used as the index.**
  - **data from address *i* would be stored in block $i \bmod 2^k$.**
- **For example, data from memory address 11 maps to cache block 3 on the right, since 11 mod 4 = 3 and since the lowest two bits of 1011 are 11.**

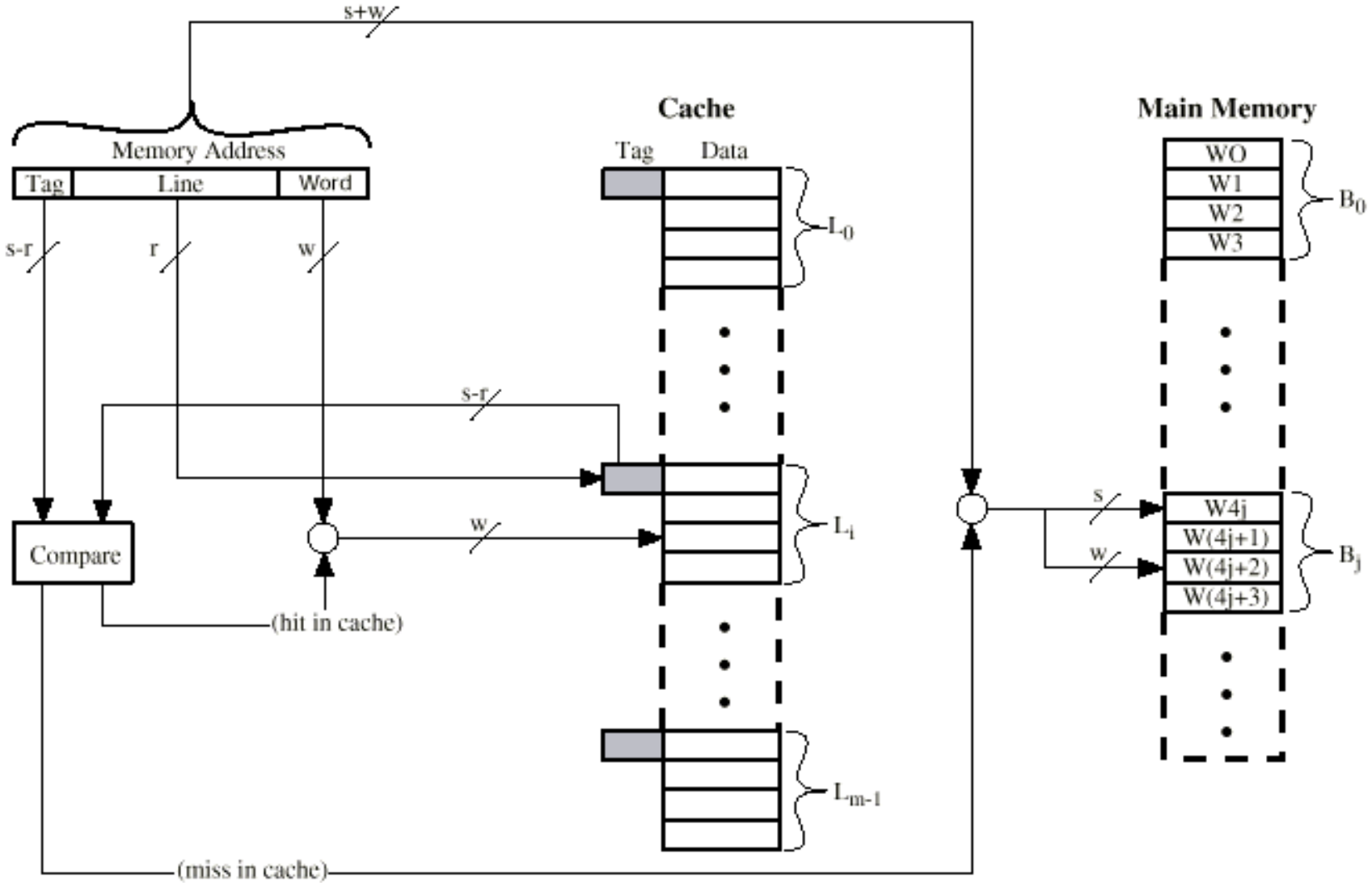


Direct Mapping: Tag

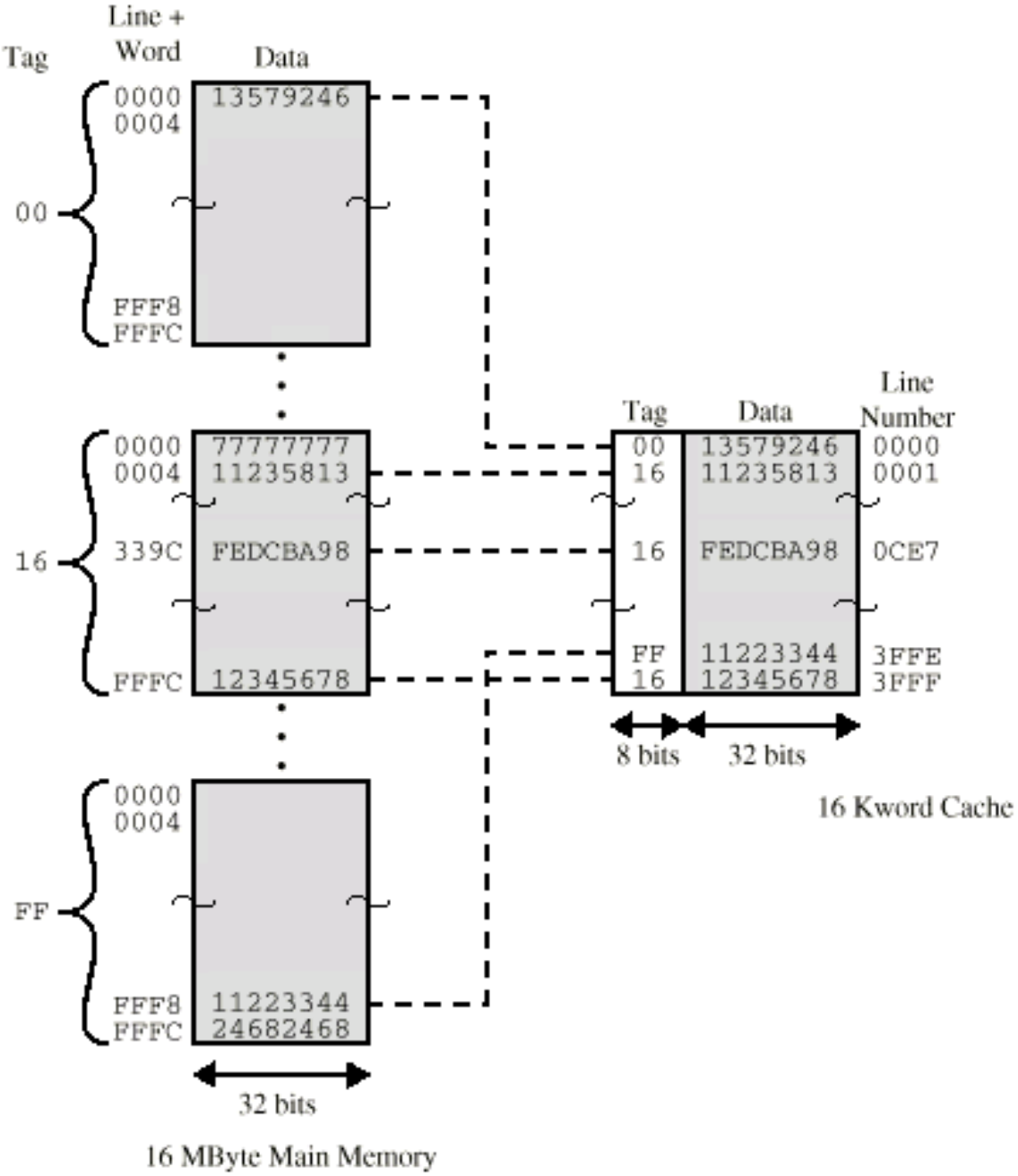
- To find data stored in the cache, we need to add tags to distinguish between different memory locations that map to the same cache block.
- We include a single valid bit per block to distinguish full and empty blocks.



Direct Mapping Cache Organization



Direct Mapping Example



Direct Mapping Summary

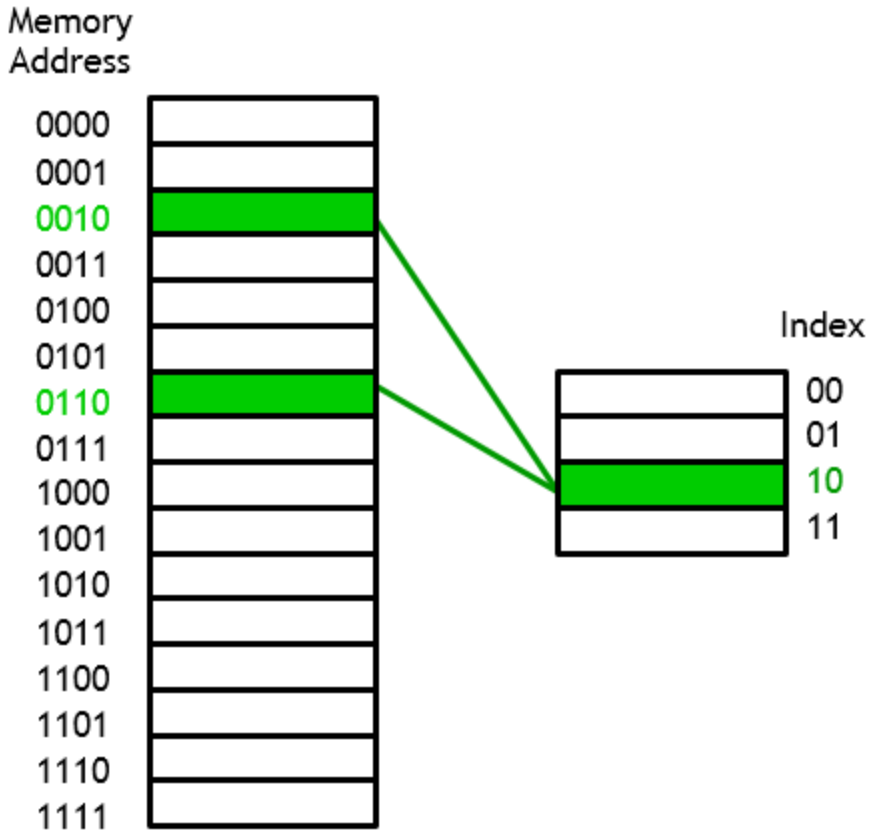
- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+ w}/2^{w} = 2^{s}$
- Number of lines in cache = m = 2^{r}
- Size of tag = (s – r) bits

Direct Mapping Advantage

- Simple
- Inexpensive

Direct Mapping Disadvantage

- **Fixed location for given block**
  - **If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high**
- E.g.: If a program uses addresses 2, 6, 2, 6, 2, ..., then each access will result in a cache miss and a load into cache block 2.
- This cache has four blocks, but direct mapping might not let us use all of them. This can result in more misses than we might like.



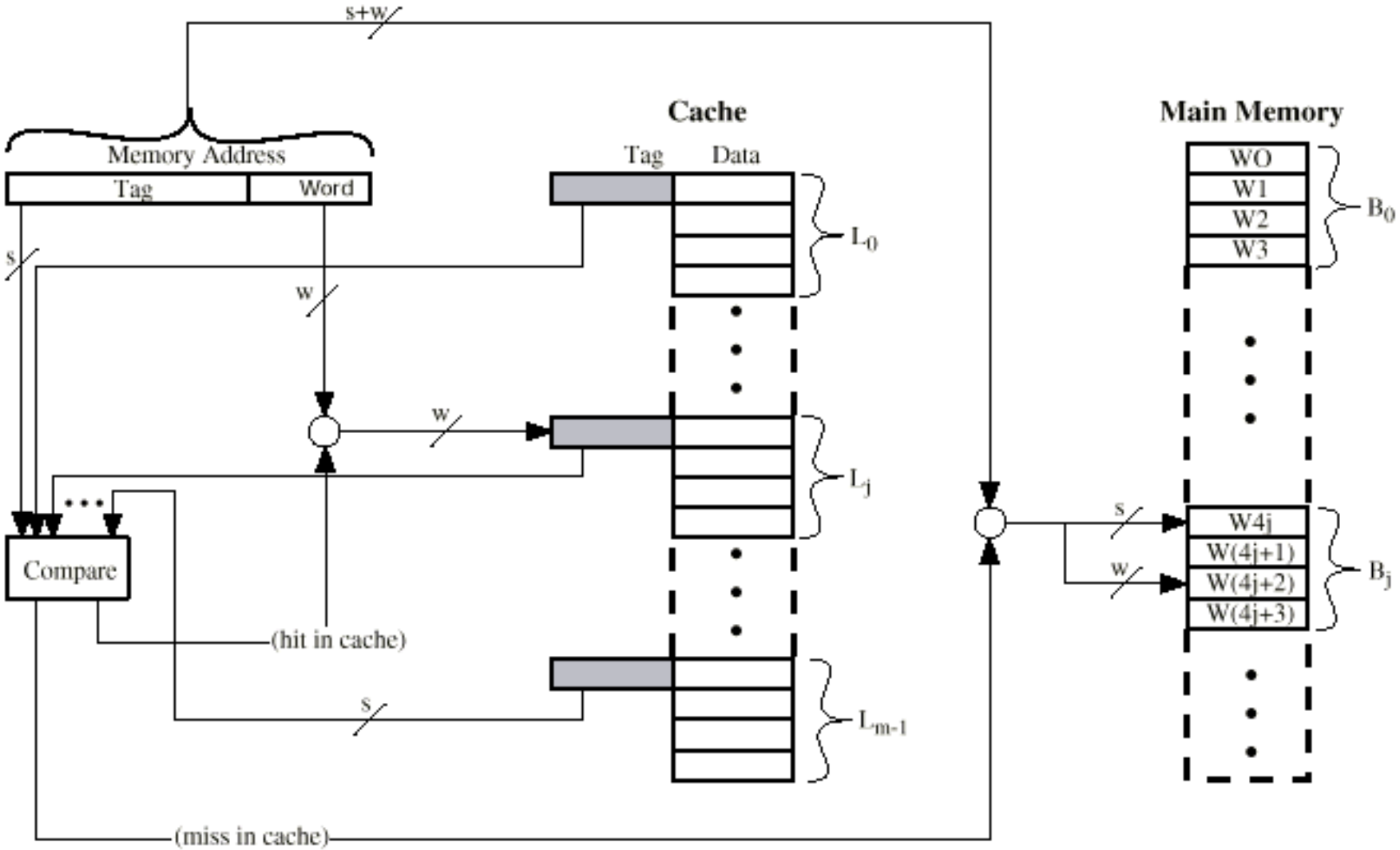
Fully Associative Mapping

- A fully associative cache permits data to be stored in any cache block, instead of forcing each memory address into one particular block.
  - When data is fetched from memory, it can be placed in any unused block of the cache.
  - This way we'll never have a conflict between two or more memory addresses which map to a single cache block.
- 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

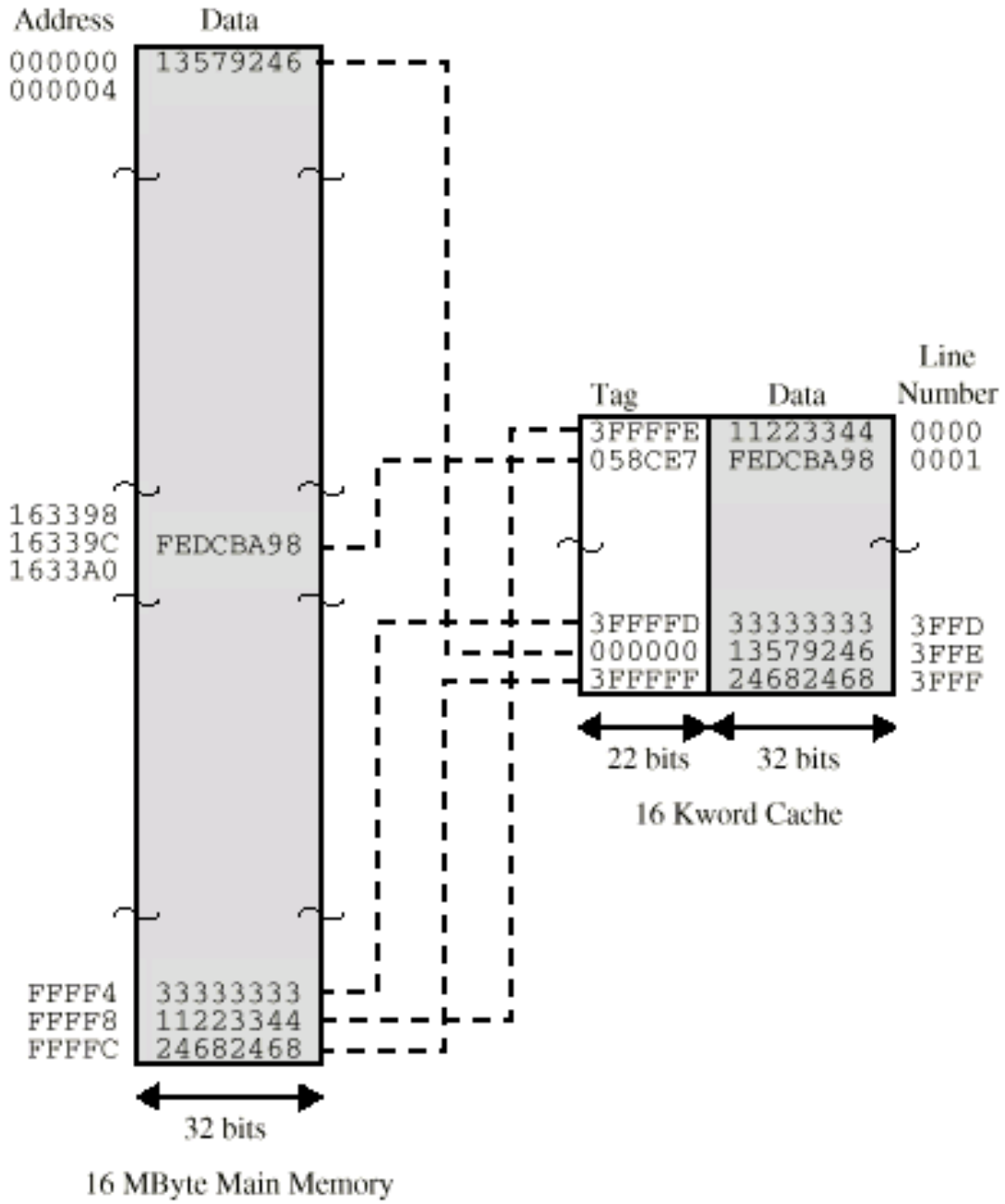
Associative Mapping Address Structure

- 22 bit tag stored with each 32 bit block of data
- Compare tag field with tag entry in cache to check for hit
- Least significant 2 bits of address identify which 16 bit word is required from 32 bit data block

Fully Associative Cache Organization

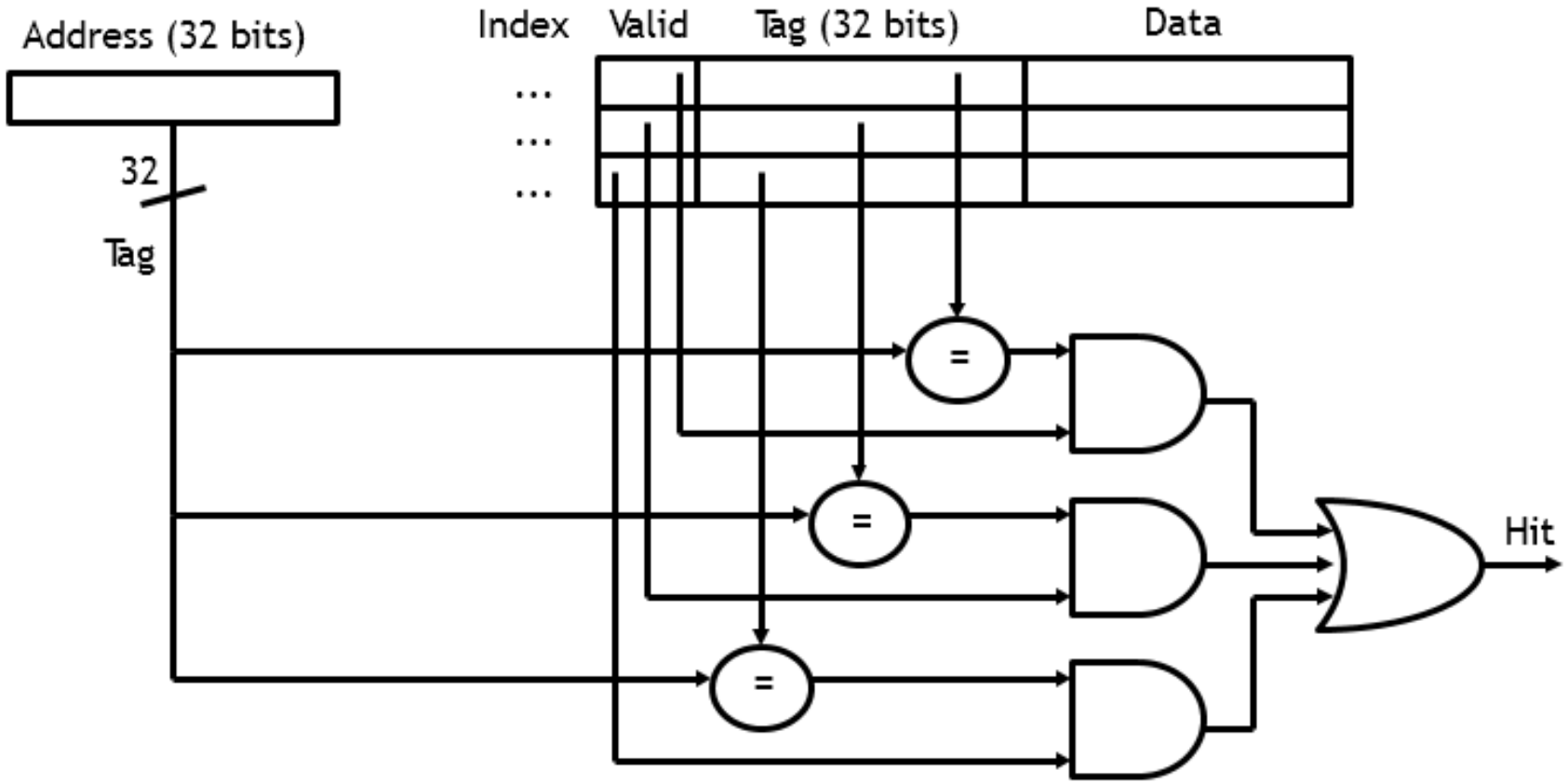


Fully Associative Mapping Example



Fully Associative Mapping - Disadvantage

- However, a fully associative cache is expensive to implement.
  - Because there is no index field in the address anymore, the *entire* block address must be used as the tag, increasing the total cache size.
  - Data could be anywhere in the cache, so we must check the tag of every block. That's a lot of comparators!



Associative Mapping Summary

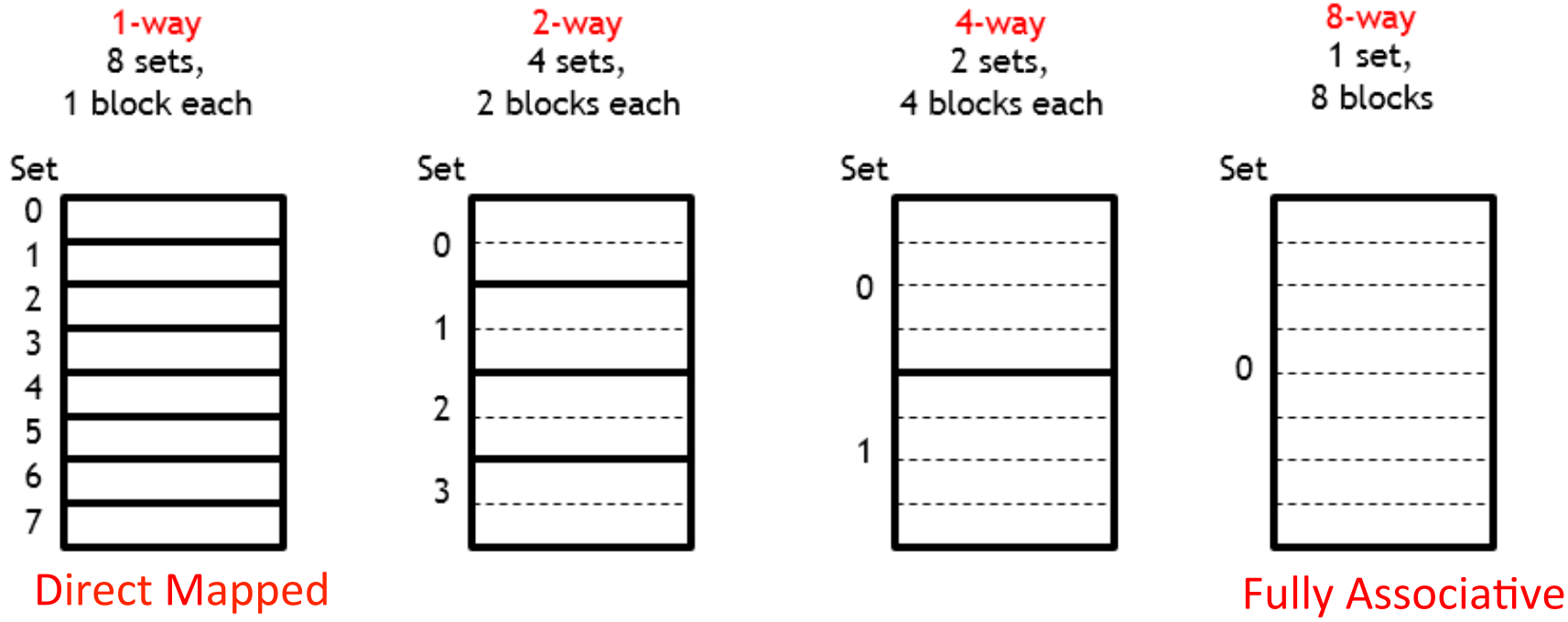
- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+ w}/2^{w} = 2^{s}$
- Number of lines in cache = undetermined
- Size of tag = s bits

Set Associative Mapping

- 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 can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

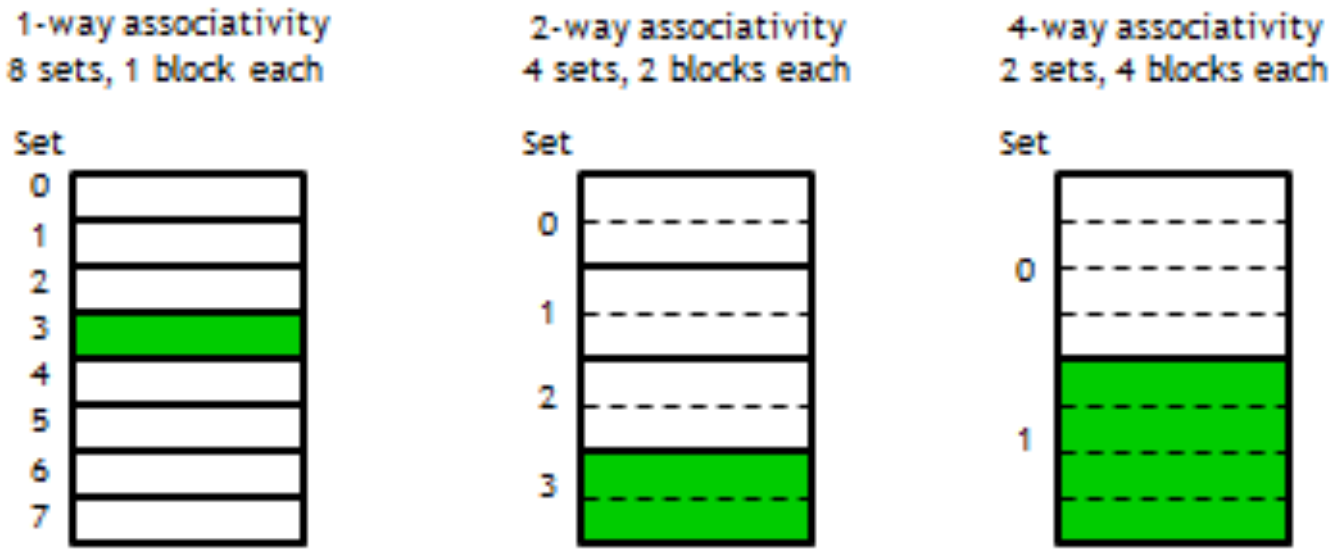
Set Associative Mapping

- An intermediate possibility is a set-associative cache.
  - The cache is divided into groups of blocks, called sets.
  - Each memory address maps to exactly one set in the cache, but data may be placed in any block within that set.

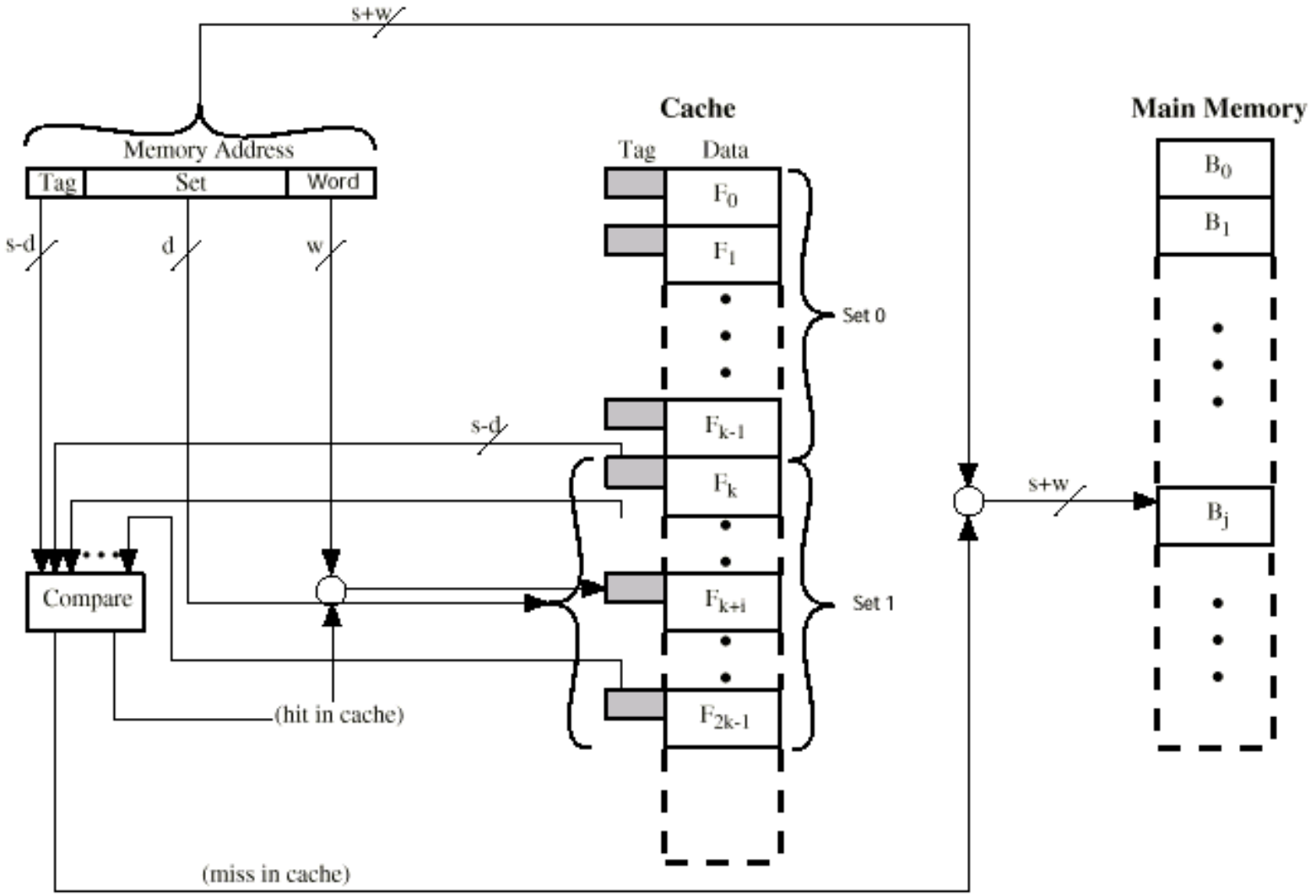


Example placement in Set-Associative Cache

- Each block has 16 bytes, so the lowest 4 bits are the block offset.
- For the 1-way cache, the next three bits (011) are the set index.
  For the 2-way cache, the next two bits (11) are the set index.
  For the 4-way cache, the next one bit (1) is the set index.
- The data may go in *any* block, shown in green, within the correct set.



Two Way Set Associative Cache Organization

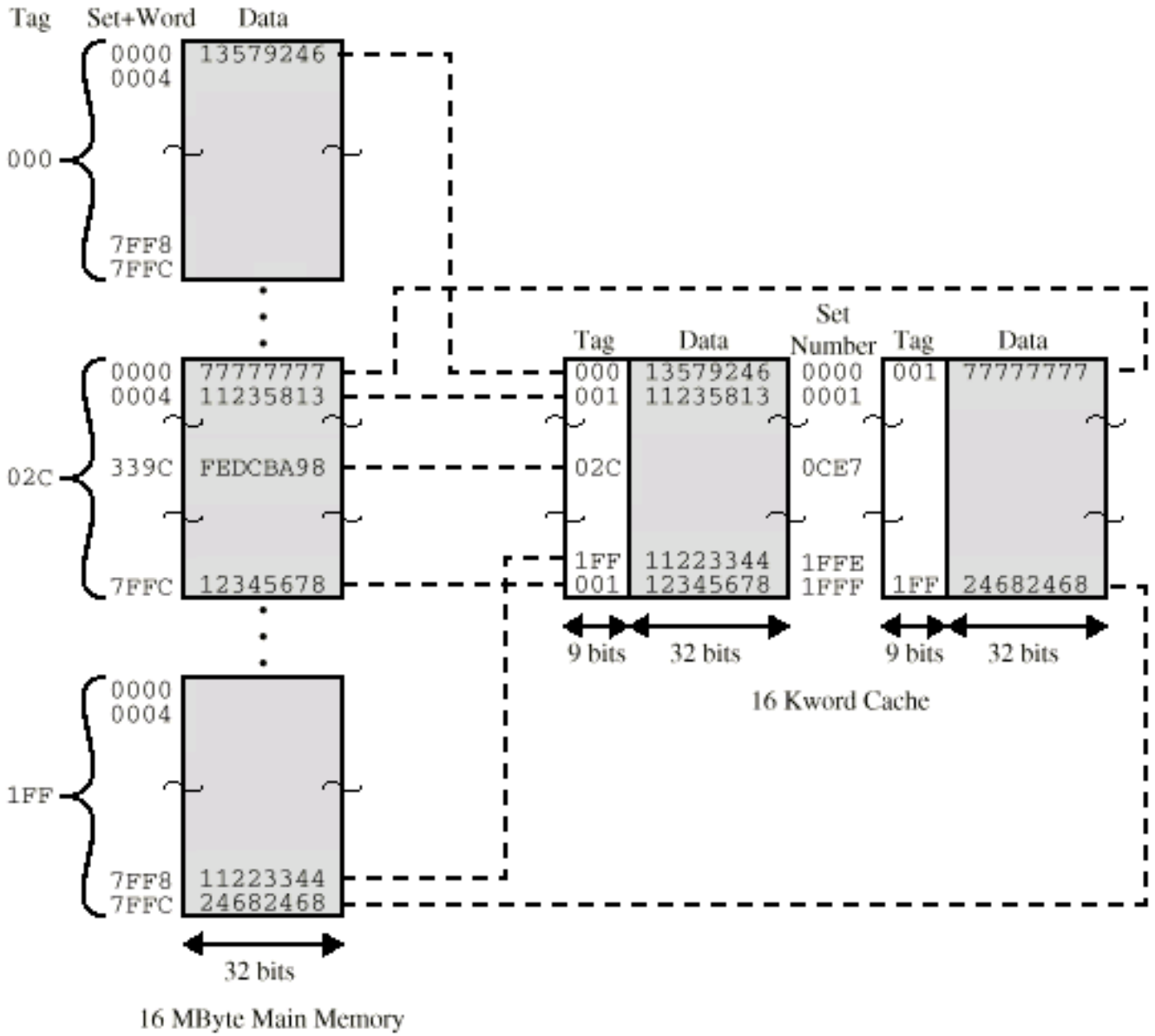


Set Associative Mapping Address Structure

Tag 9 bit	Set 13 bit	Word 2 bit

- Use set field to determine cache set to look in
- Compare tag field to see if we have a hit
- e.g
  - Address Tag Data Set number
  - 1FF 7FFC 1FF 12345678 1FFF
  - 001 7FFC 001 11223344 1FFF

Two Way Set Associative Mapping Example



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

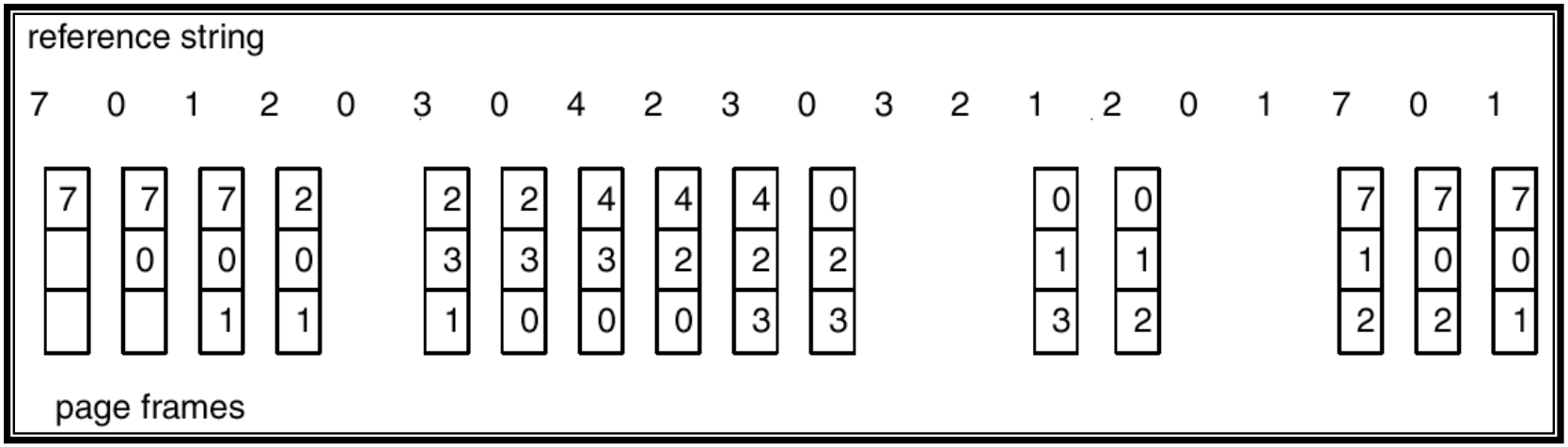
- No choice
- Each block only maps to one line
- Replace that line

Replacement Algorithms: Associative & Set Associative

- Least Recently used (LRU): 2 way set associative
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - The block that has been in the cache the longest with no reference to it is replaced
  - The cache consists of a stack of blocks
  - Most recently referenced block is on the top of the stack
  - When a block is referenced or brought into the cache, it is placed on the top of the stack

FIFO Algorithm

- **First in First Out.**
- **Replace the page that is oldest.**



15 page faults

LRU Algorithm

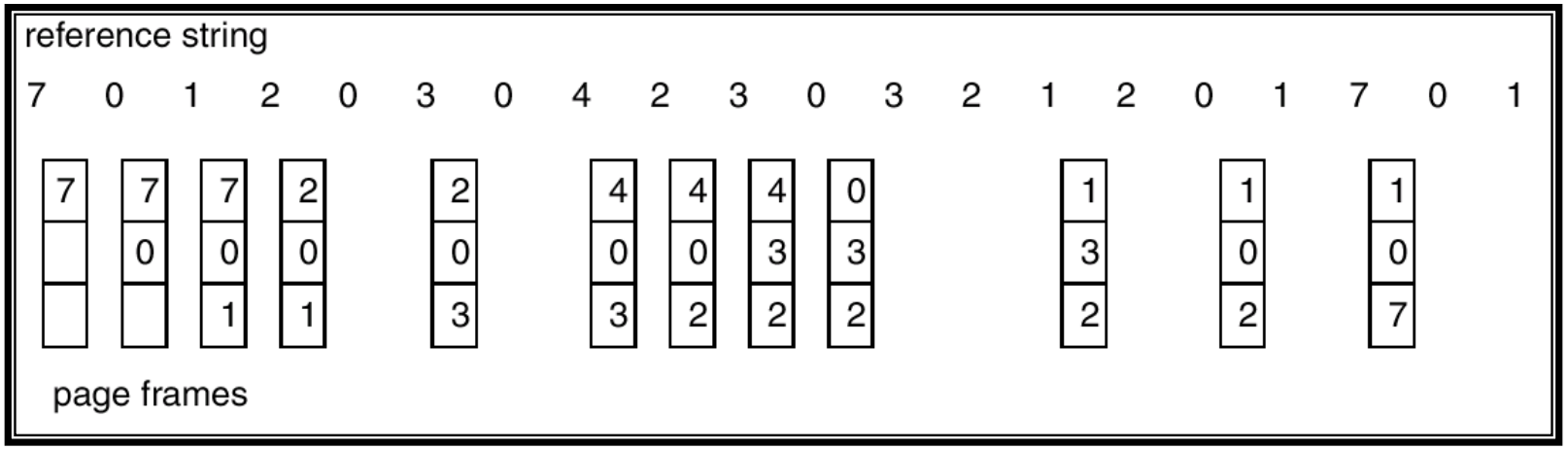
- **Replace the page that has not been used for the longest period of time.**
- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- Most recently referenced block is on the top of the stack
- When a block is referenced or brought into the cache, it is placed on the top of the stack

E.g. Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

— Cache will have 4 frames.

Sol. 8 Page Faults

LRU Page Replacement



12 page faults

Least Recently Used (LRU)

- The block that has been in the cache the longest with no reference to it is replaced
- The cache consists of a stack of blocks
- Most recently referenced block is on the top of the stack
- When a block is referenced or brought into the cache, it is placed on the top of the stack

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write policy in the cache

- Write-though: the information is written to both the block in cache and block in main memory.
- Write-back: information is only written to the block in cache. The modified block is written to main memory only when it is replaced.

Advantage of write-back

- Individual words can be written by the processor in the cache level, fast!
- Multiple writes within a block requires only one write to main memory
- When blocks are written back, the system can make effective use of a high bandwidth transfer.

disadvantage of write-back

- Interaction with other processors when RAW (Read after Write) hazard occurs, say other processor will read the incorrect data in its own cache.

Advantage of write-through

- Misses are simpler and cheaper because they never require a block in cache to be written to main memory.
- Easy to implement than write-back, a write-through cache only needs a write buffer.

disadvantage of write-through

- Cost since write to main memory is very slow

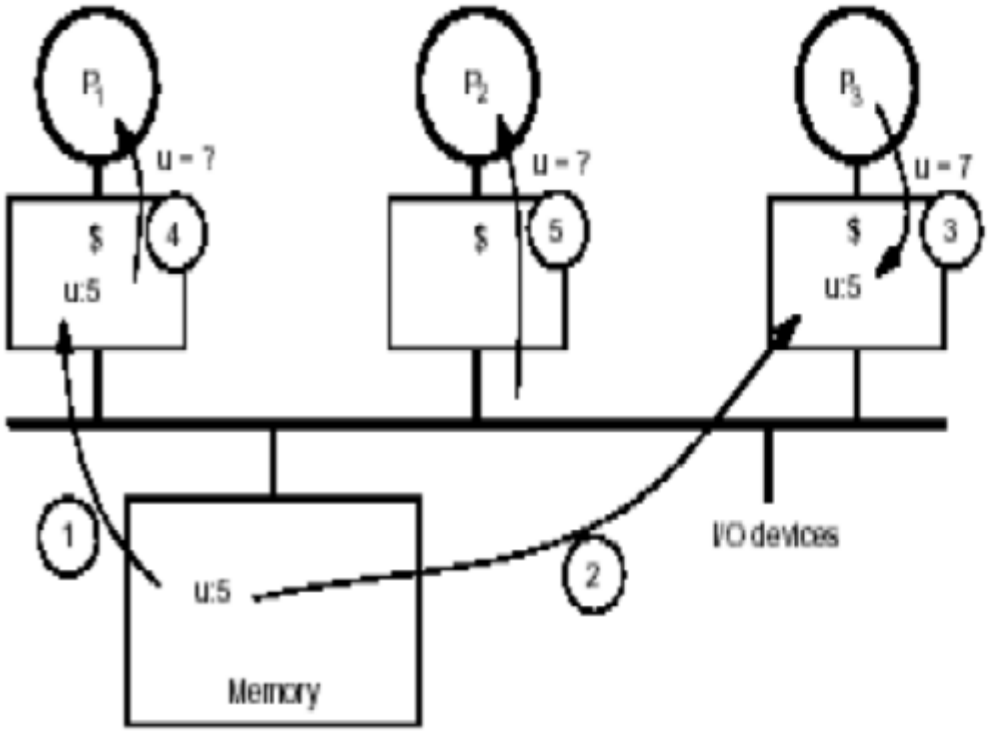
Write through

- All writes go to main memory as well as cache
- Multiple CPUs can monitor main memory traffic to keep local (to CPU) cache up to date
- Lots of traffic
- Slows down writes

Write back

- Updates initially made in cache only
- Update bit for cache slot is set when update occurs
- If block is to be replaced, write to main memory only if update bit is set
- Other caches get out of sync
- I/O must access main memory through cache
- N.B. 15% of memory references are writes

Example: Write back



1. P1 reads u=5 from memory

2. P3 reads u=5 from memory

3. P3 writes u=7 to local P3 cache

4. P1 reads old u=5 from local P1 cache

5. P2 reads old u=5 from memory

Line Size

- Larger blocks reduce the number of blocks that fit into a cache. Because each block fetch overwrites older cache contents, a small number of blocks results in data being overwritten shortly after they are fetched.

Number of caches

MULTILEVEL CACHES:

- Most contemporary designs include both on-chip and external caches. The simplest such organization is known as a two-level cache, with the internal cache designated as level 1 (L1) and the external cache designated as level 2 (L2).
- The reason for including an L2 cache is the following: If there is no L2 cache and the processor makes an access request for a memory location not in the L1 cache, then the processor must access memory across the bus.
- Due to the typically slow bus speed and slow memory access time, this results in poor performance.
- On the other hand, if an L2 SRAM (static RAM) cache is used, then frequently the missing information can be quickly retrieved.

Number of caches

UNIFIED VERSUS SPLIT CACHES:

- When the on-chip cache first made an appearance, many of the designs consisted of a single cache used to store references to both data and instructions.
- More recently, it has become common to split the cache into two: one dedicated to instructions and one dedicated to data.
- These two caches both exist at the same level, typically as two L1 caches.
- When the processor attempts to fetch an instruction from main memory, it first consults the instruction L1 cache, and when the processor attempts to fetch data from main memory, it first consults the data L1 cache.

Semiconductor Memory Types

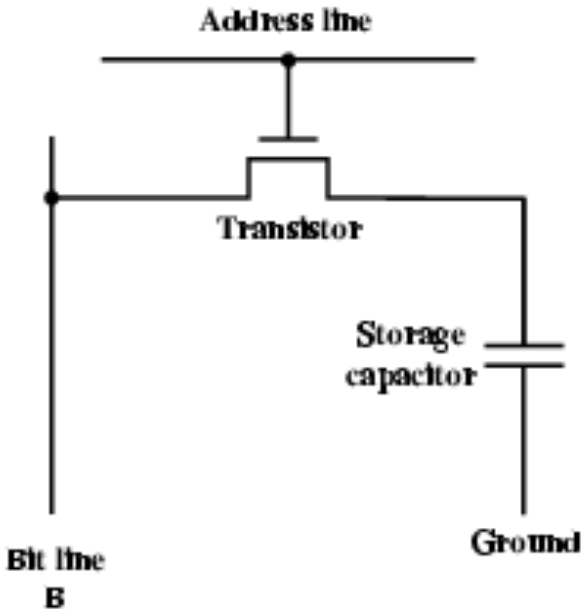
Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV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

DRAM Organization

- A dynamic RAM (DRAM) is made with cells that store data as charge on capacitors.
- The presence or absence of charge in a capacitor is interpreted as a binary 1 or 0.
- Because capacitors have a natural tendency to discharge, dynamic RAMs require periodic charge refreshing to maintain data storage.
- Simpler construction
- Less expensive
- Need refresh circuits
- Slower
- Main memory

DRAM Organization

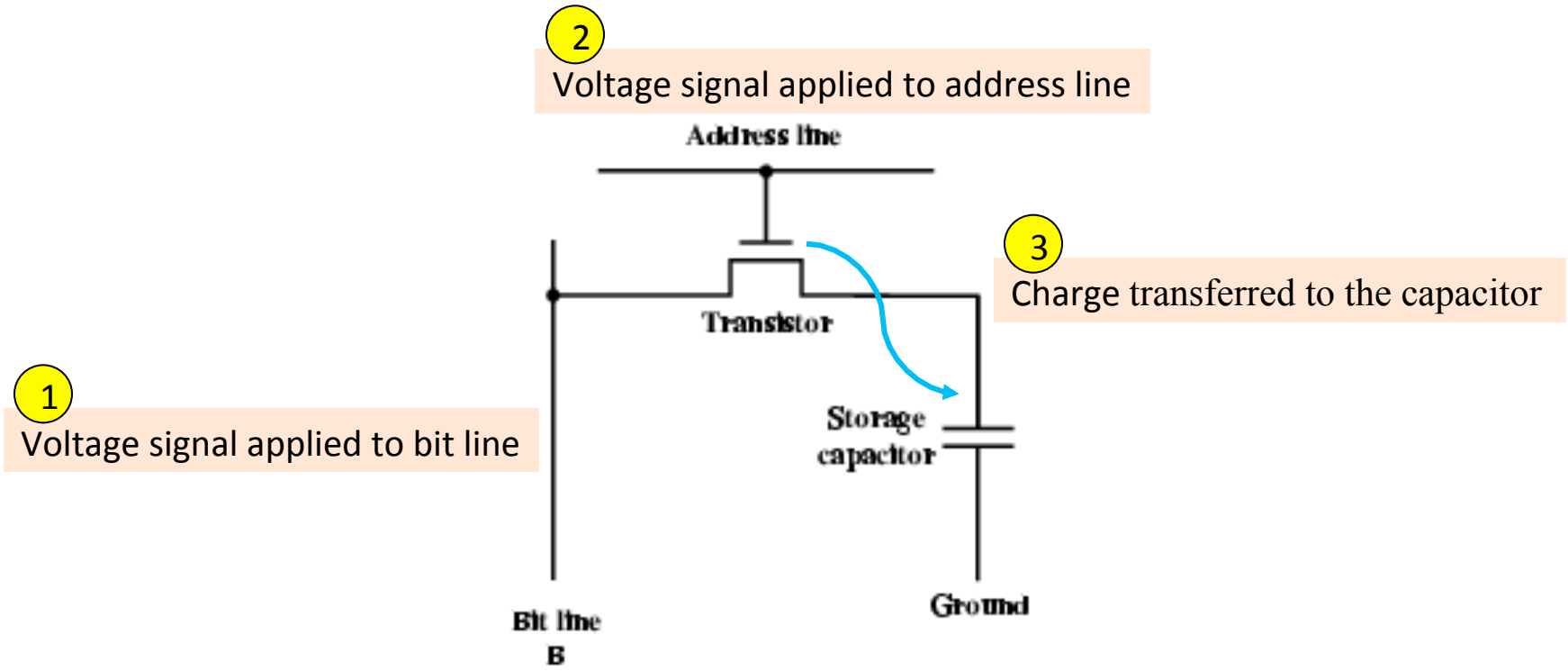
- An individual cell that stores 1 bit.
- The address line is activated when the bit value from this cell is to be read or written.
- The transistor acts as a switch that is closed (allowing current to flow) if a voltage is applied to the address line and open (no current flows) if no voltage is present on the address line.
- A high voltage represents 1, and a low voltage represents 0.



DRAM Organization: Write operation

WRITE:

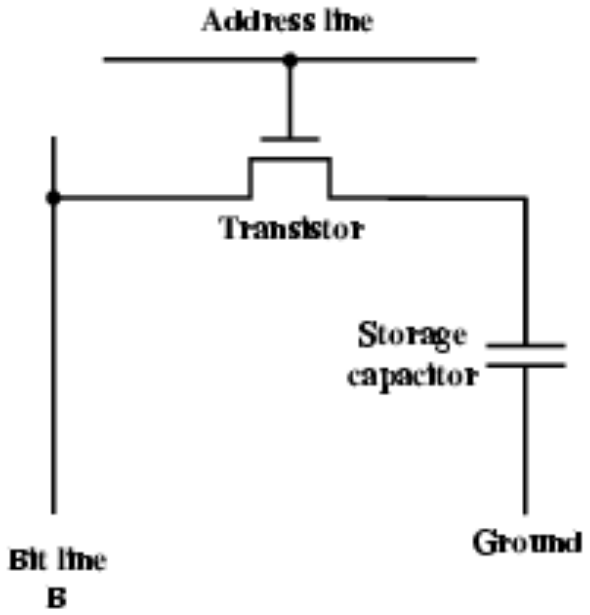
- A voltage signal is applied to the bit line
- A signal is then applied to the address line, allowing a charge to be transferred to the capacitor.



DRAM Organization: Read operation

READ:

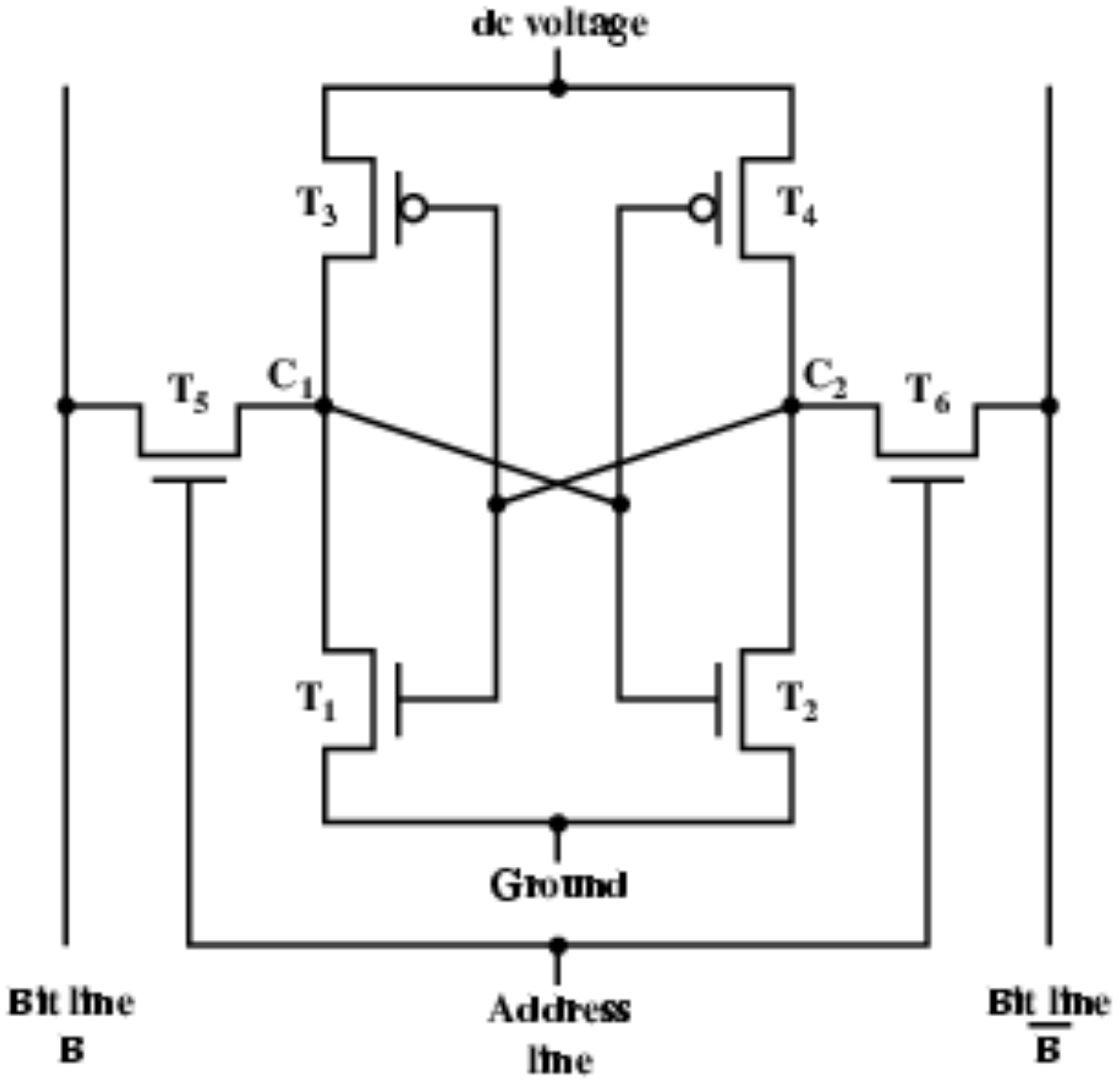
•When the address line is selected, the transistor turns on and the charge stored on the capacitor is fed out onto a bit line and to a sense amplifier. The sense amplifier compares the capacitor voltage to a reference value and determines if the cell contains a logic 1 or a logic 0. The readout from the cell discharges the capacitor, which must be restored to complete the operation.



Static RAM

- Bits stored as on/off switches
- Holds data as long as power is supplied
- No charges to leak
- No refreshing needed when powered
- More complex construction
- More expensive
- Does not need refresh circuits
- Faster
- Cache
- Digital
  - Uses flip-flops

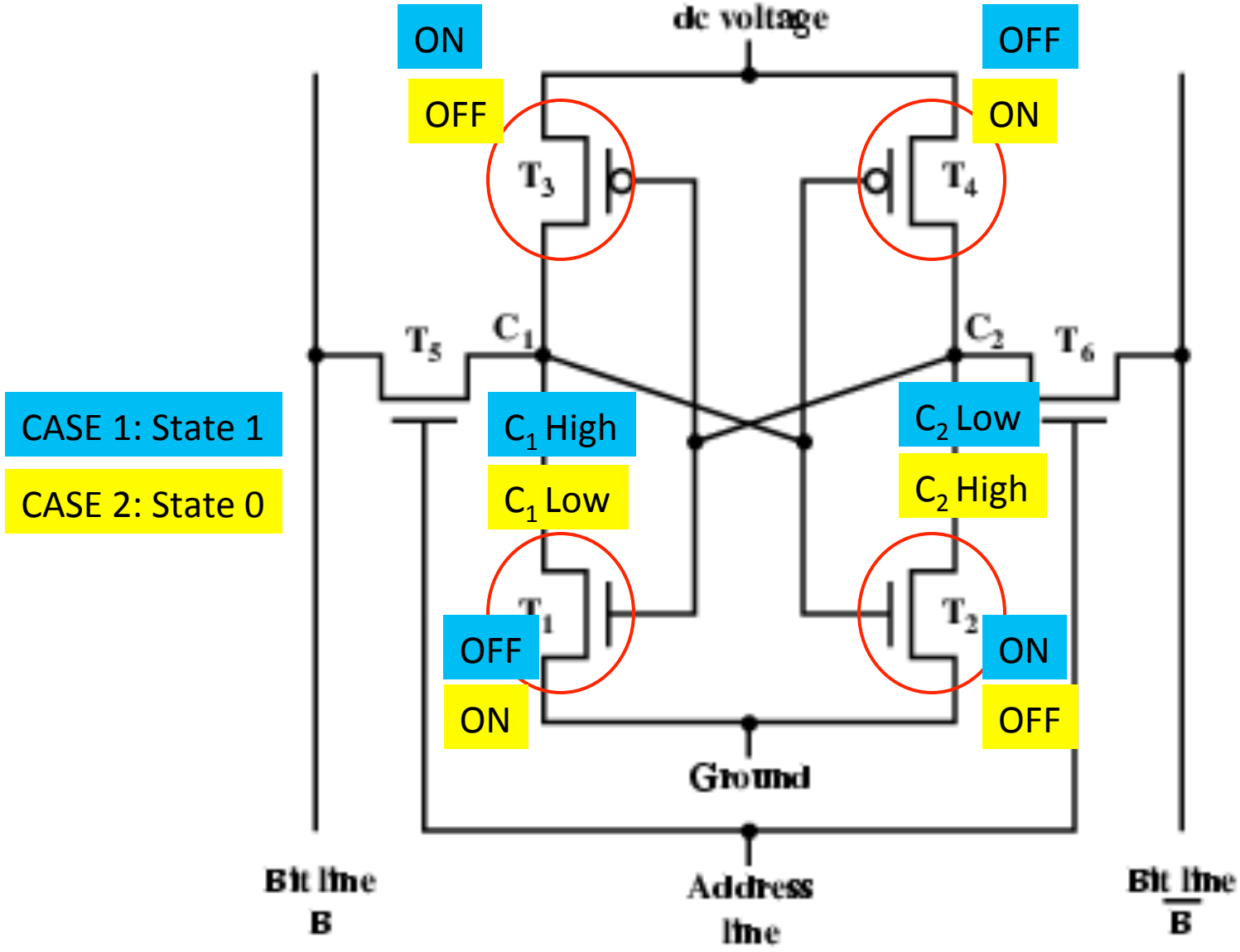
Static RAM Structure



Static RAM Operation

- Four Transistors – T_1 T_2 T_3 T_4
- Transistor arrangement gives stable logic state
- State 1
  - C_1 high, C_2 low
  - T_1 T_4 off, T_2 T_3 on
- State 0
  - C_2 high, C_1 low
  - T_2 T_3 off, T_1 T_4 on
- Both states are stable as long as the direct current (dc) voltage is applied.
- 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.

Static RAM Operation



For a write operation, the desired bit value is applied to line B, while its complement is applied to line . This forces the four transistors (T1, T2, T3, T4) into the proper state.

For a read operation, the bit value is read from line B.

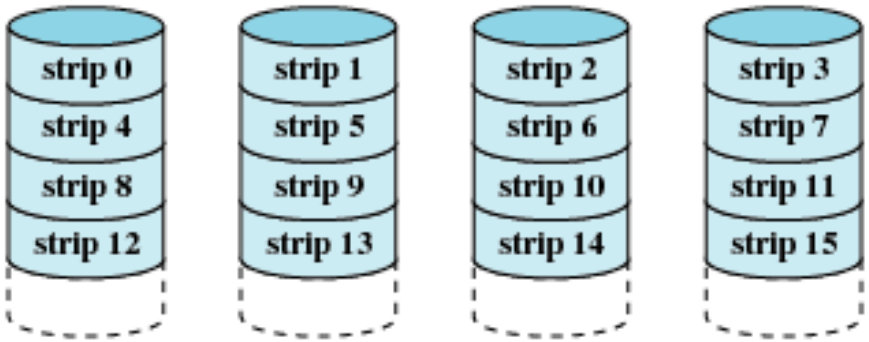
SRAM v DRAM

- Both volatile
  - Power needed to preserve data
- Dynamic cell
  - Simpler to build, smaller
  - More dense
  - Less expensive
  - Needs refresh
  - Larger memory units
- Static
  - Faster
  - Cache

RAID

- Redundant Array of Independent Disks
- Not a hierarchy—but different design architectures with 3 common characteristics:
  - Set of physical disks viewed as single logical drive by O/S
  - Data distributed across physical drives of an array
  - Can use redundant capacity to store parity information, which guarantees data recoverability in case of a disk failure (except RAID 0)
- It replaces the large-capacity disk drives with multiple smaller-capacity drives and distributes data in such a way to enable simultaneous access to data from multiple drives.
- Improves I/O performance and allows easier incremental increase in capacity.

RAID 0 (non-redundant)



(a) RAID 0 (non-redundant)

Each disk is divided into strips.
A set of consecutive strips is referred to as stripe.
Advantage: reduces I/O transfer time

RAID 0 (non-redundant)

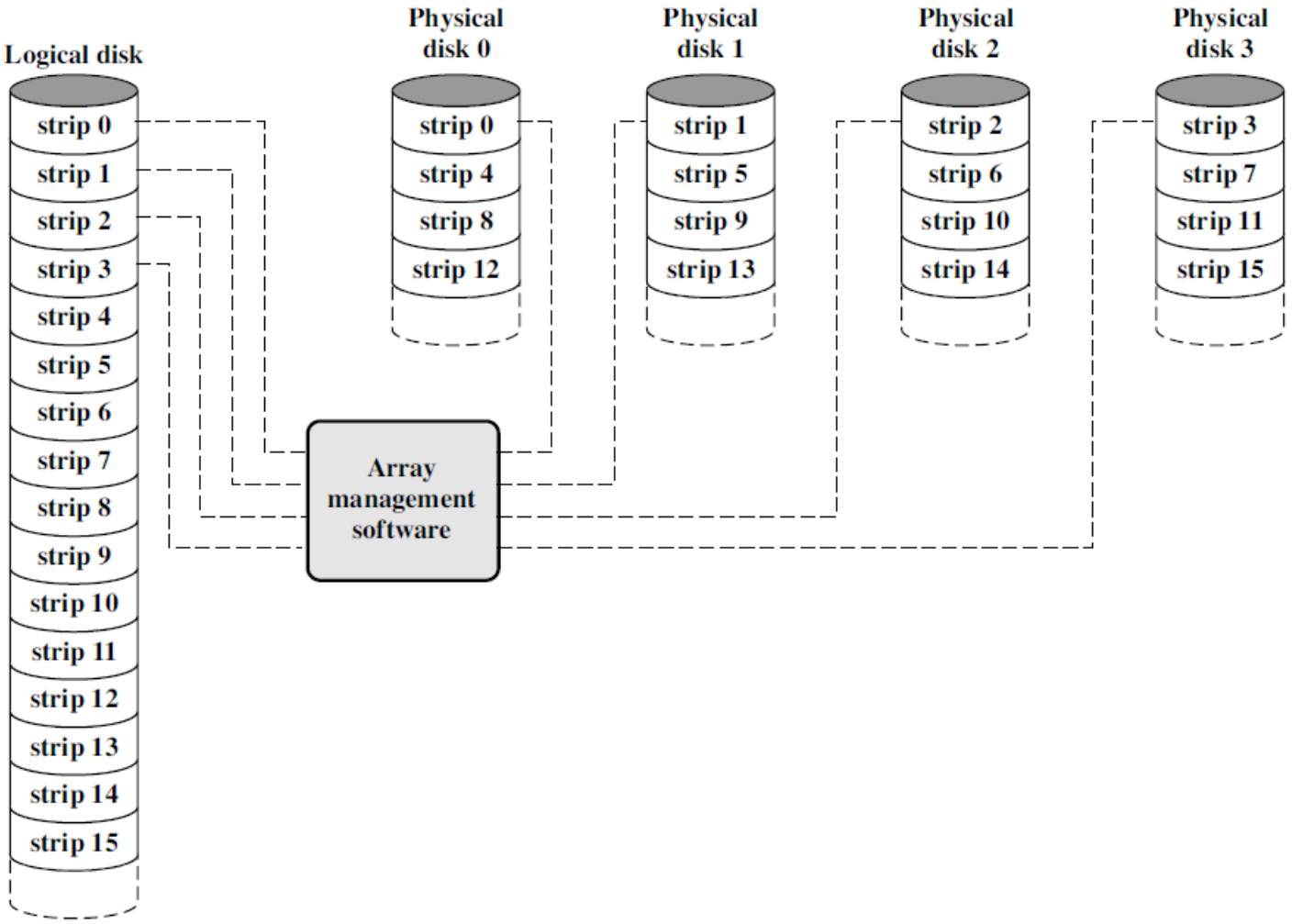
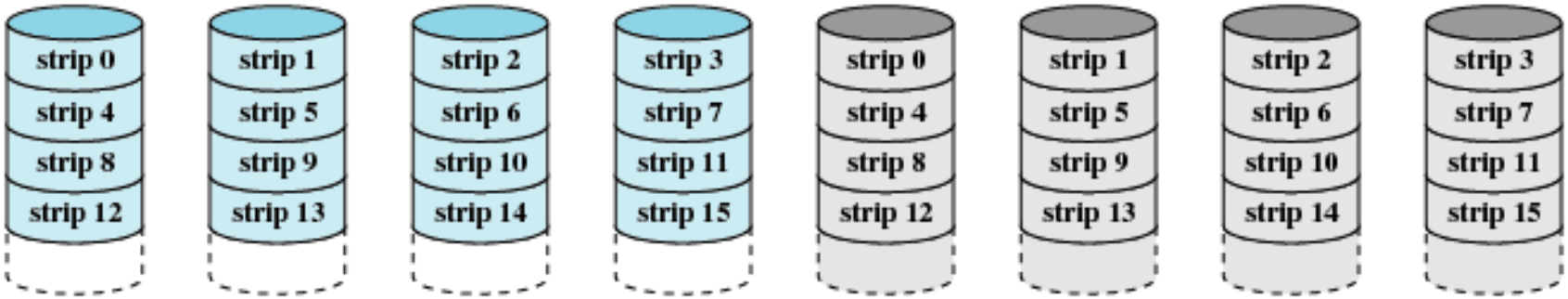


Figure Data Mapping for a RAID Level 0 Array

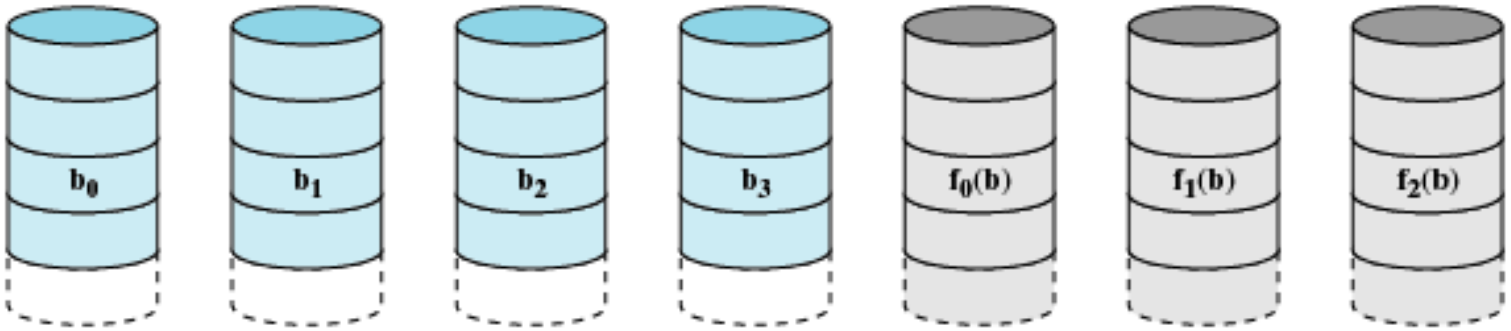
RAID 1 (mirrored)



(b) RAID 1 (mirrored)

Redundancy is achieved by duplicating all the data.
Advantages: (i) recovery from failure is simple
Disadvantage: cost

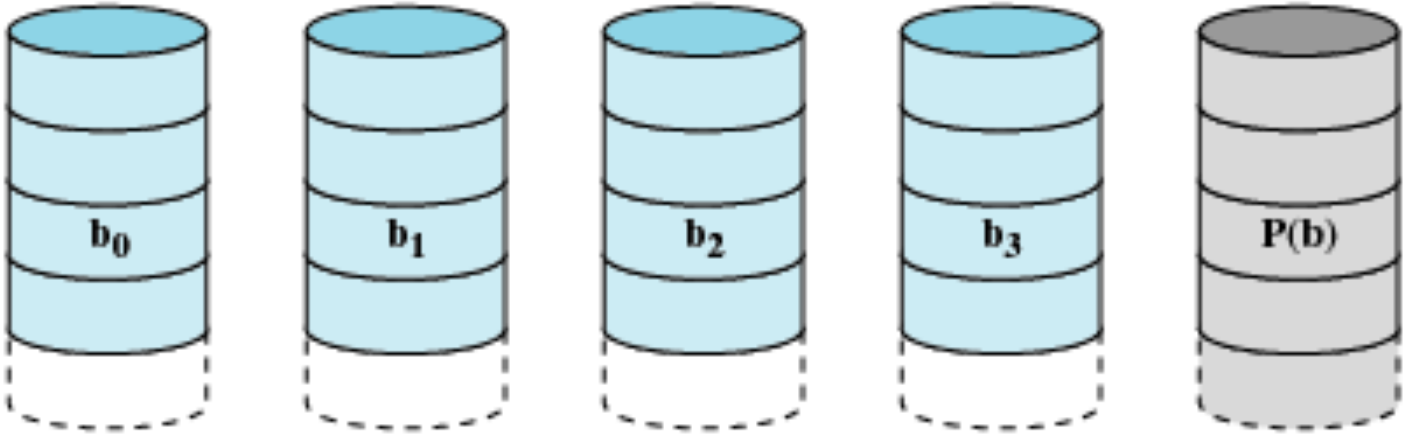
RAID 2 (redundancy through Hamming code)



(c) RAID 2 (redundancy through Hamming code)

Makes use of parallel access technique
All member disks participate in the execution of every I/O request.
Strips are small (byte or word)
An error correcting code is calculated across corresponding bits on each data disk, and the bits of code are stored in corresponding bit positions on multiple parity disks.

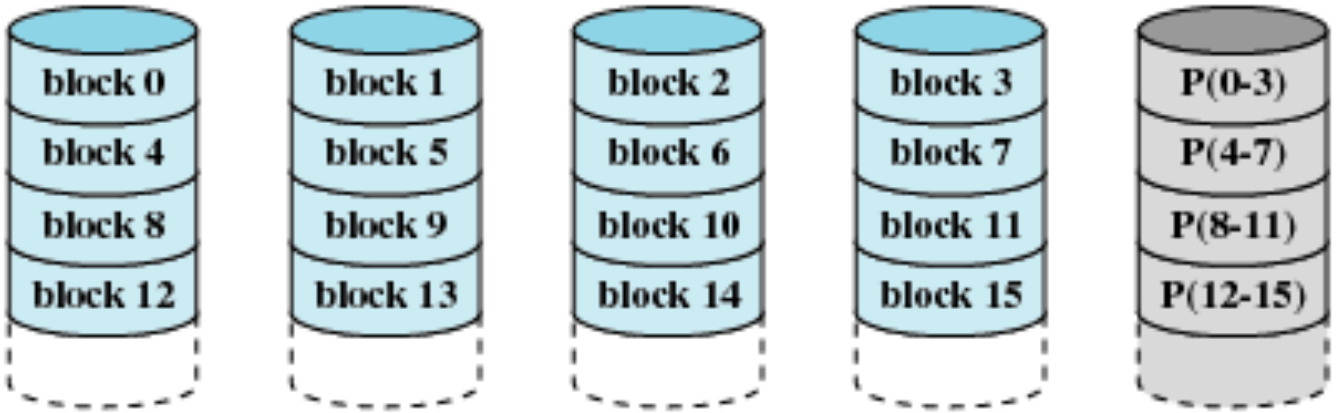
RAID 3 (bit-interleaved parity)



(d) RAID 3 (bit-interleaved parity)

Similar to RAID 2
Difference is that it requires a single redundant disk.
Instead of an error correcting code, simple parity bit is computed for the set of bits in the same position on all the data disks.

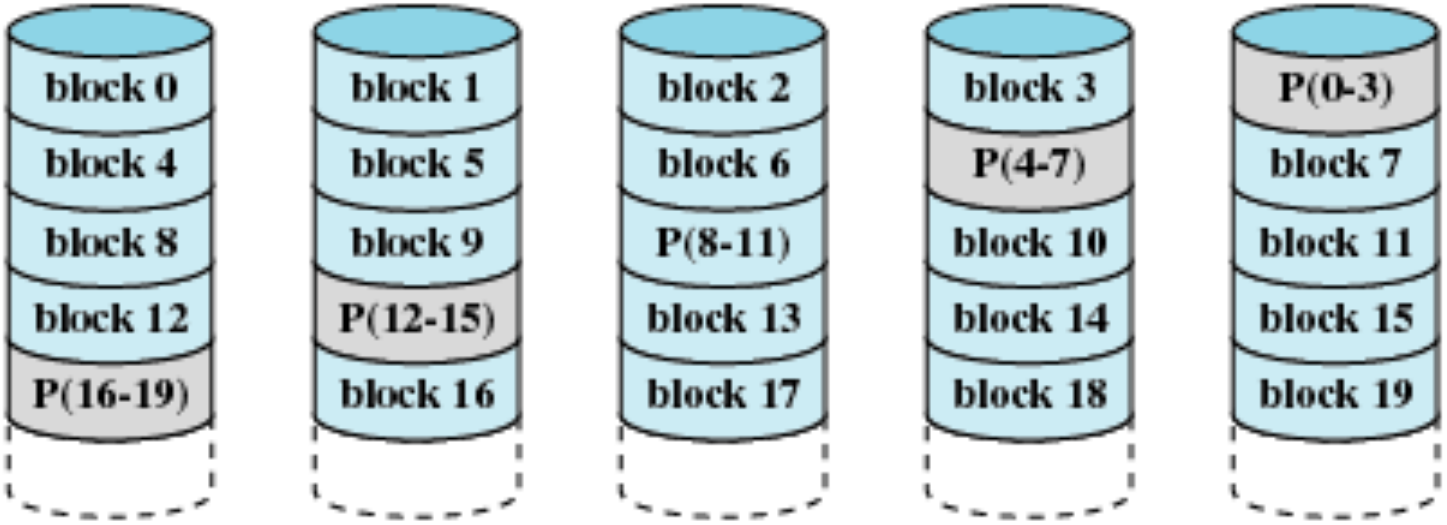
RAID 4 (block-level parity)



(e) RAID 4 (block-level parity)

Makes use of an independent access technique
Strips are relatively large.

RAID 5 (block-level distributed parity)

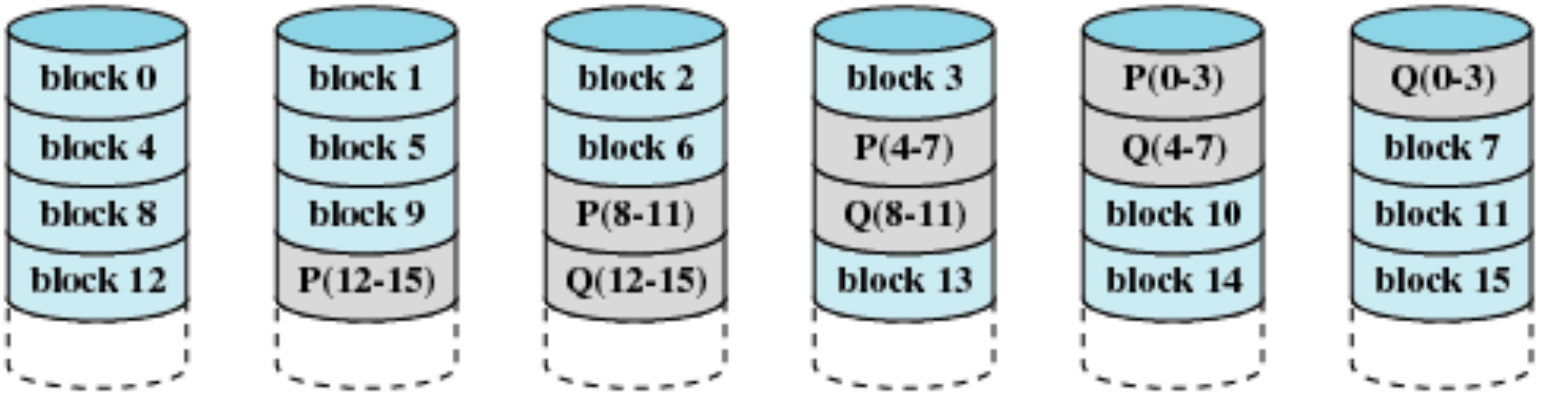


(f) RAID 5 (block-level distributed parity)

Similar to RAID 4

Difference is that it distributes parity strips across all disks.

RAID 6 (dual redundancy)



(g) RAID 6 (dual redundancy)

Two different parity calculations are carried out and stored in separate blocks on different disks.

RAID Levels

Category	Level	Description	I/O Request Rate (Read/Write)	Data Transfer Rate (Read/Write)	Typical Application
Striping	0	Non-redundant	Large strips: Excellent	Small strips: Excellent	Applications requiring high performance for non-critical data
Mirroring	1	Mirrored	Good/Fair	Fair/Fair	System drives; critical files
Parallel access	2	Redundant via Hamming code	Poor	Excellent	
	3	Bit-interleaved parity	Poor	Excellent	Large I/O request size applications, such as imaging, CAD
Independent access	4	Block-interleaved parity	Excellent/Fair	Fair/Poor	
	5	Block-interleaved distributed parity	Excellent/Fair	Fair/Poor	High request rate, read-intensive, data lookup
	6	Block-interleaved dual distributed parity	Excellent/Poor	Fair/Poor	Applications requiring extremely high availablity

RAID Comparison

Level	Advantages	Disadvantages	Applications
0	I/O performance is greatly improved by spreading the I/O load across many channels and drives No parity calculation overhead is involved Very simple design Easy to implement	The failure of just one drive will result in all data in an array being lost	Video production and editing Image Editing Pre-press applications Any application requiring high bandwidth
1	100% redundancy of data means no rebuild is necessary in case of a disk failure, just a copy to the replacement disk Under certain circumstances, RAID 1 can sustain multiple simultaneous drive failures Simplest RAID storage subsystem design	Highest disk overhead of all RAID types (100%)—inefficient	Accounting Payroll Financial Any application requiring very high availability
2	Extremely high data transfer rates possible The higher the data transfer rate required, the better the ratio of data disks to ECC disks Relatively simple controller design compared to RAID levels 3, 4 & 5	Very high ratio of ECC disks to data disks with smaller word sizes—inefficient Entry level cost very high—requires very high transfer rate requirement to justify	No commercial implementations exist/ not commercially viable

RAID Comparison contd.

3	Very high read data transfer rate Very high write data transfer rate Disk failure has an insignificant impact on throughput Low ratio of ECC (parity) disks to data disks means high efficiency	Transaction rate equal to that of a single disk drive at best (if spindles are synchronized) Controller design is fairly complex	Video production and live streaming Image editing Video editing Prepress applications Any application requiring high throughput
4	Very high Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency	Quite complex controller design Worst write transaction rate and Write aggregate transfer rate Difficult and inefficient data rebuild in the event of disk failure	No commercial implementations exist/ not commercially viable

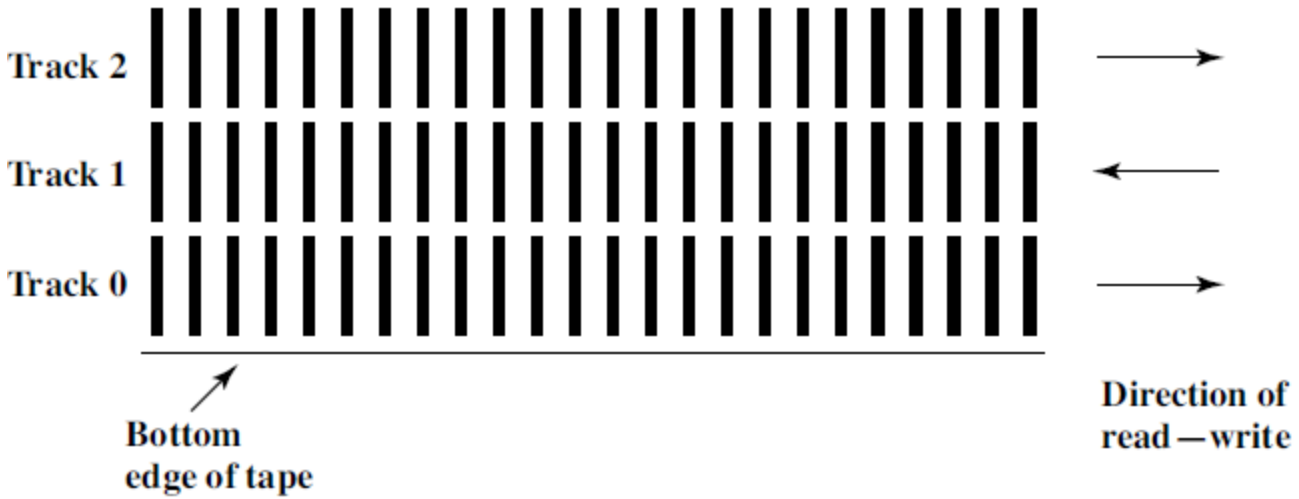
RAID Comparison contd.

Level	Advantages	Disadvantages	Applications
5	Highest Read data transaction rate Low ratio of ECC (parity) disks to data disks means high efficiency Good aggregate transfer rate	Most complex controller design Difficult to rebuild in the event of a disk failure (as compared to RAID level 1)	File and application servers Database servers Web, e-mail, and news servers Intranet servers Most versatile RAID level
6	Provides for an extremely high data fault tolerance and can sustain multiple simultaneous drive failures	More complex controller design Controller overhead to compute parity addresses is extremely high	Perfect solution for mission critical applications

Magnetic Tape

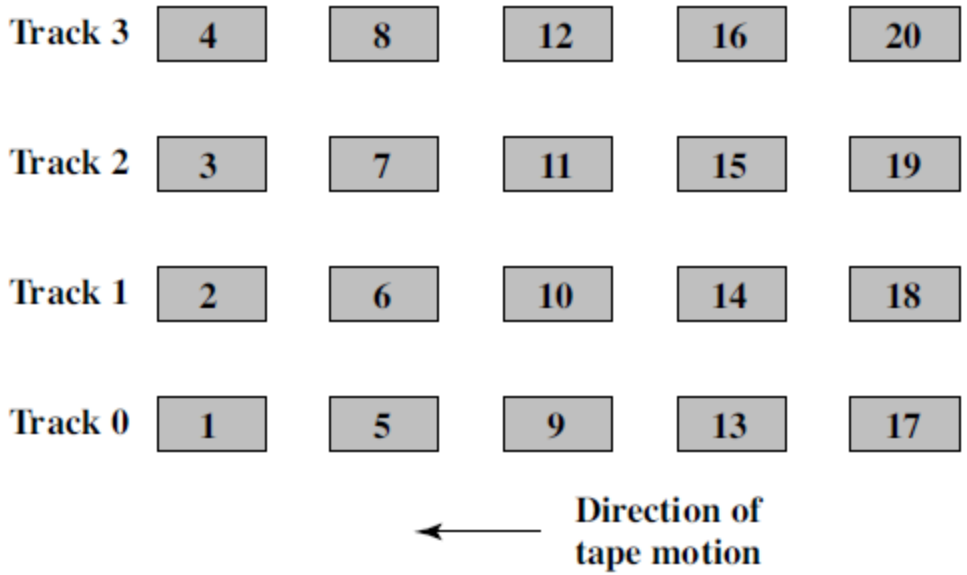
- Serial access
- Slow
- Very cheap
- Backup and archive

Magnetic Tape



(a) Serpentine reading and writing

Magnetic Tape



(b) Block layout for system that reads—writes four tracks simultaneously

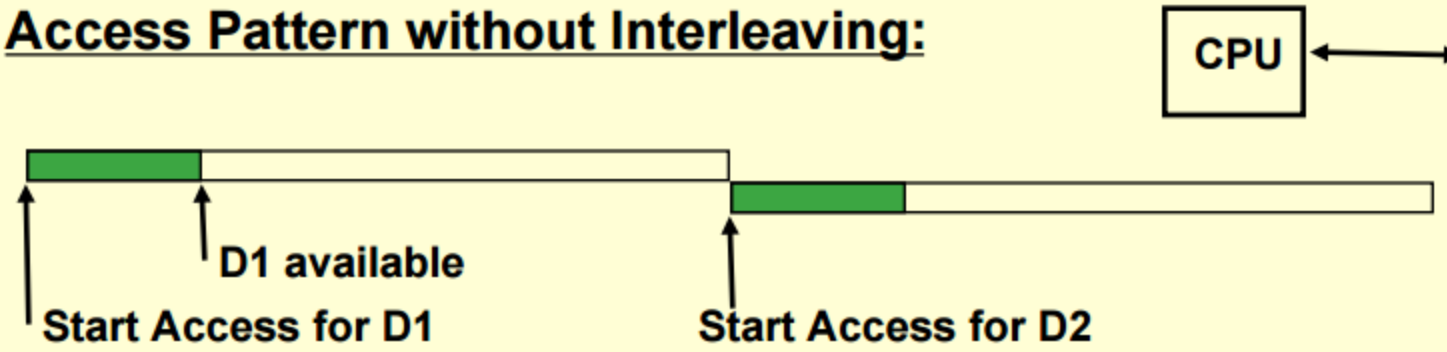
Figure Typical Magnetic Tape Features

Associative Memory

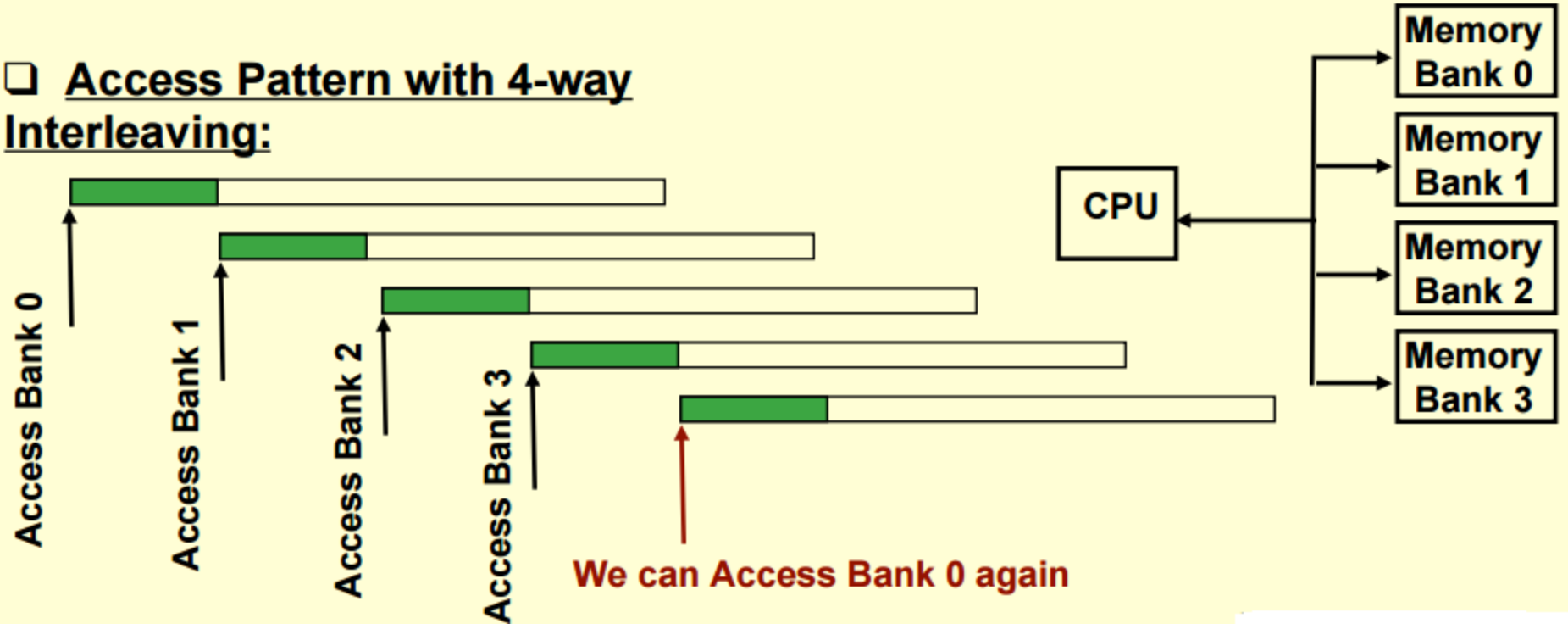
- This type of memory is accessed simultaneously and in parallel on the basis of data content rather then by specific address or location.
- When a word is written in an associative memory, no address is given. The memory is capable of finding an empty unused location to store the word.
- When a word is to be read from an associative memory, the content of the word, or part of the word, is specified. The memory locates all words which match the specified content and marks them for reading.
- An associative memory is more expensive than a random access memory.

Interleaved Memory

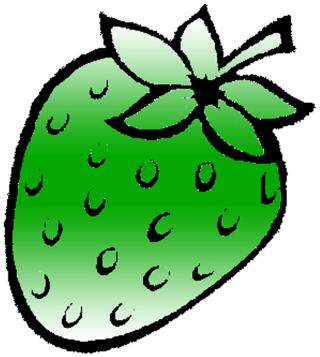
- **Access Pattern without Interleaving:**



- **Access Pattern with 4-way Interleaving:**



STRAWBERRY



/strawberrydevelopers

/strawberry_app

For more visit:

Strawberrydevelopers.weebly.com