CHAPTER

7

MEMORY HIERARCHY

TRUONG DAI HOC KHOA HOC TU NHIEN
KHTN
TP. HO CHI MINH

cdio 4.0

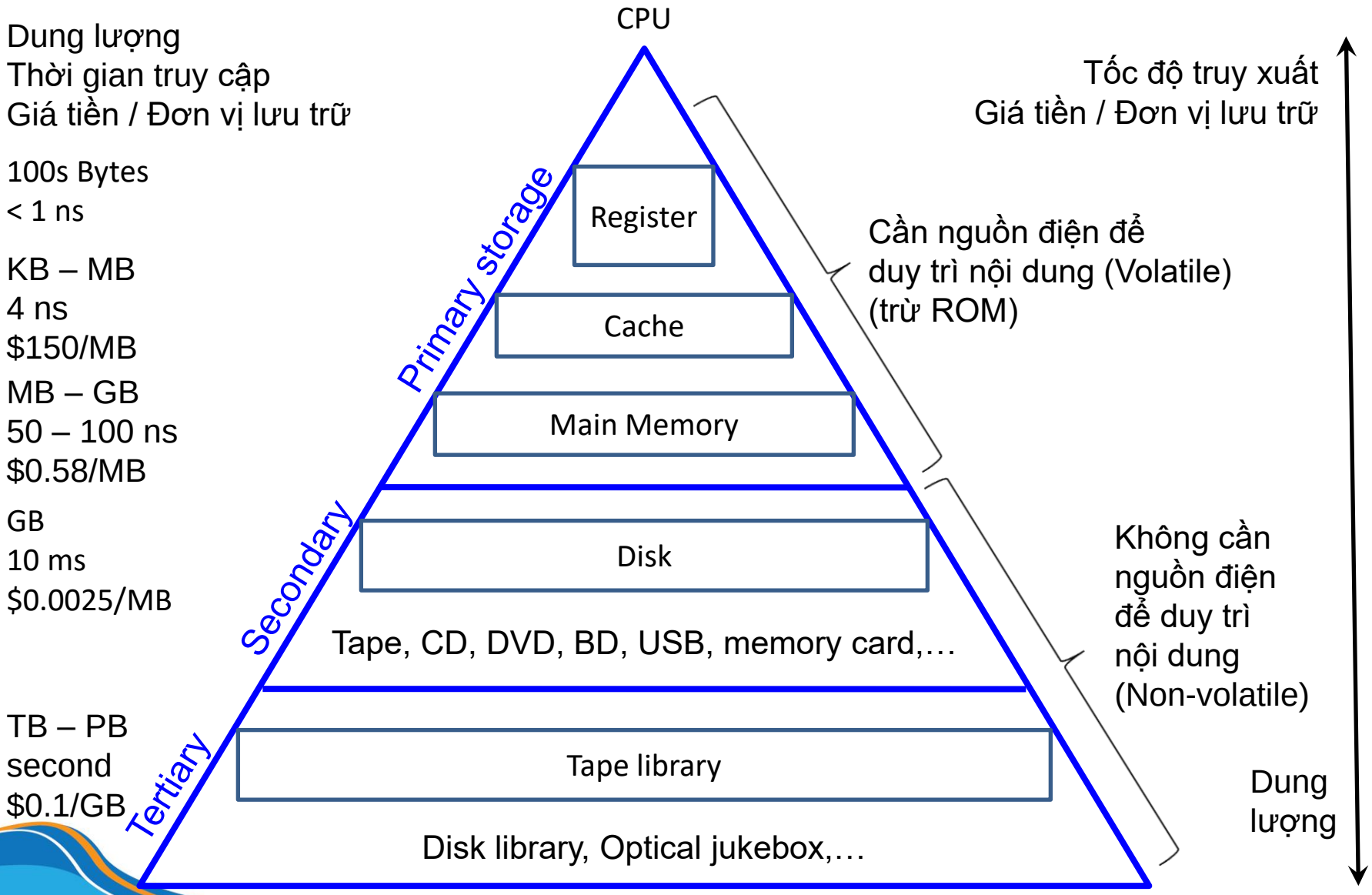
KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

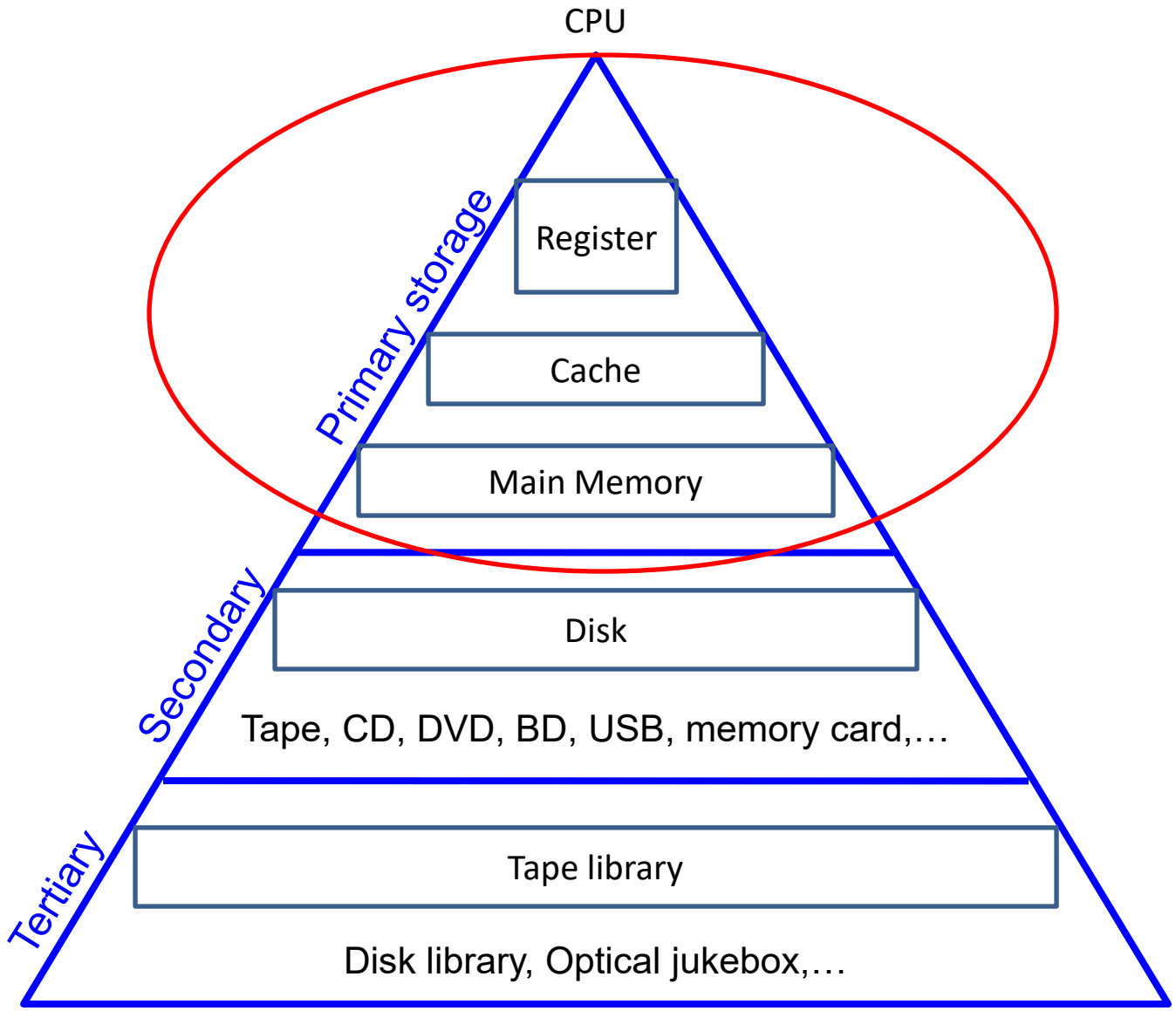
Classification

- Primary storage / internal memory
  - Registers, Cache, ROM, RAM
- Secondary storage / external memory
  - Magnetic disk (HDD, floppy disk, magnetic tape)
  - Optical disc (CD, DVD, BD)
  - Flash memory (USB, memory card, SSD)
- Tertiary storage / tertiary memory
  - Tape library, disk library, optical jukebox,…

Hierarchy of storage system



Primary storage



Register

- Storage with the smallest capacity but the fastest access speed
- Inside CPU
  - Store instructions and data (operands, calculation results, status bits) for processing
- Made by flip-flops
- Usually organized into a “Register file”

Read-Only Memory

- ROM – A type of memory that is read-only, non-writable, and does not require power to maintain its contents
- PROM - Programable ROM, can be written only once, needs special equipment to program
- EPROM – Erasable PROM, can write but all the storage cells must be erased (by UV) to the same initial state
- EEPROM – Electrically EPROM, can be written in byte-level at any time without erasing prior contents, takes much longer to write than read
- FlashROM – Cannot erase in byte-level, must erase block-level, but the writing and erasing speed is very high

EPROM

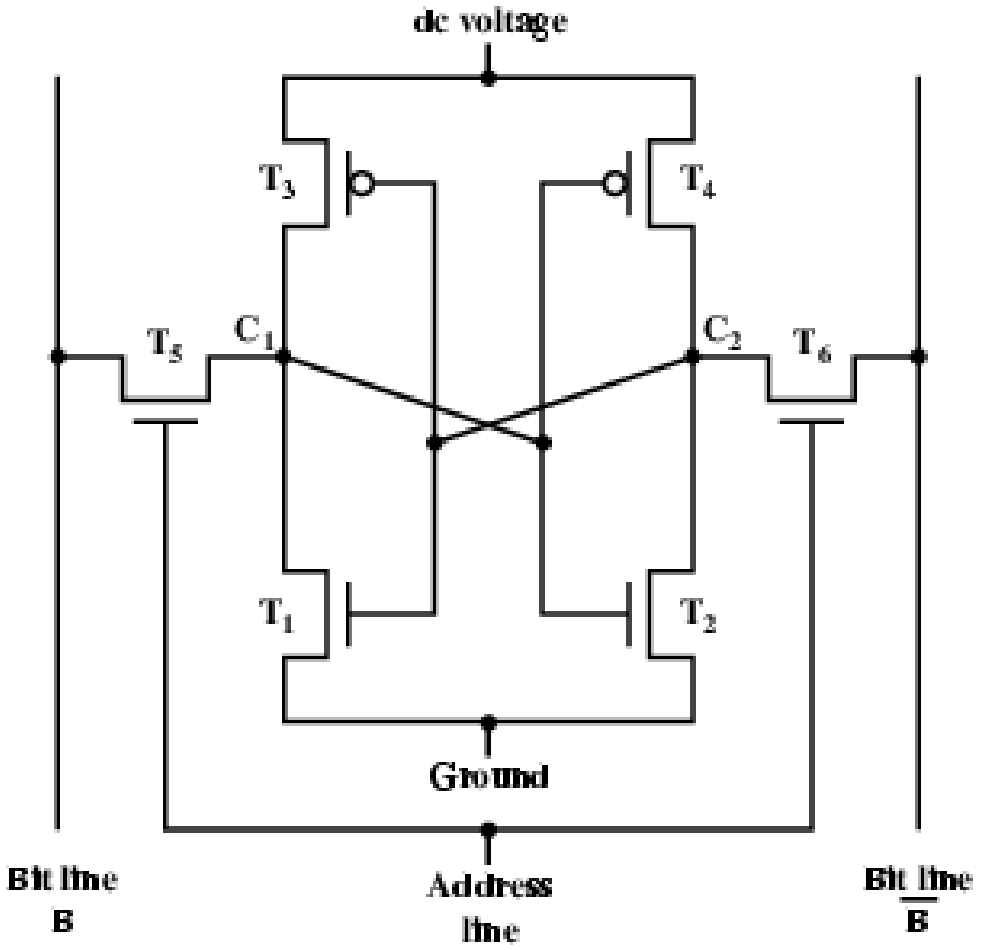
EEPROM

FlashROM

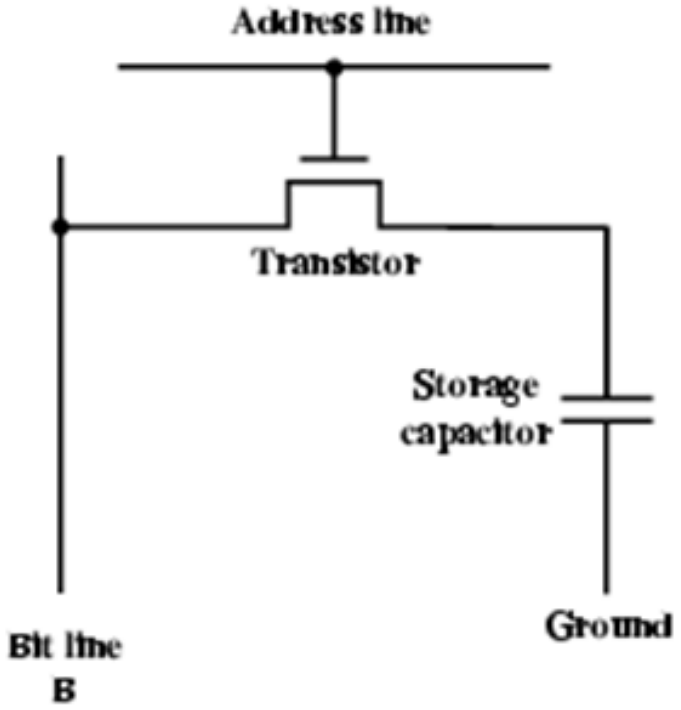
Random Access Memory

- 2 types of RAM

(Static RAM - SRAM)

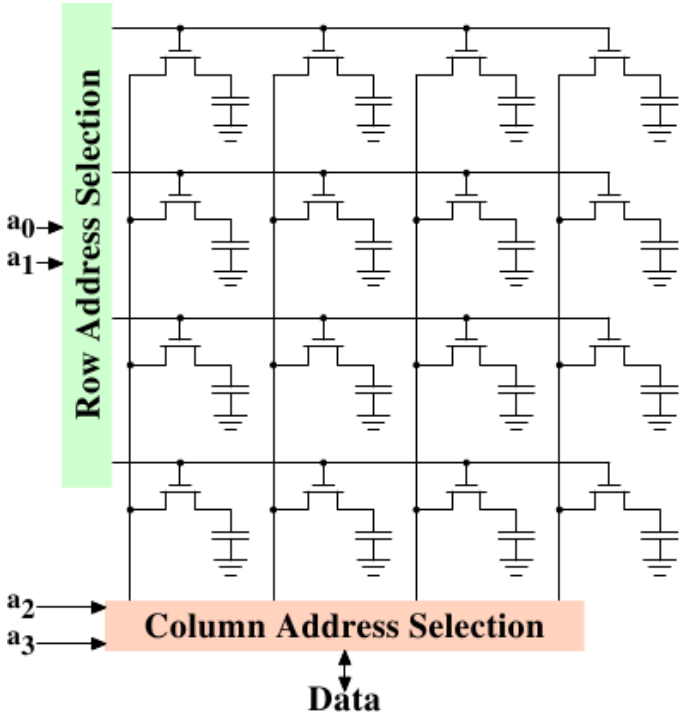


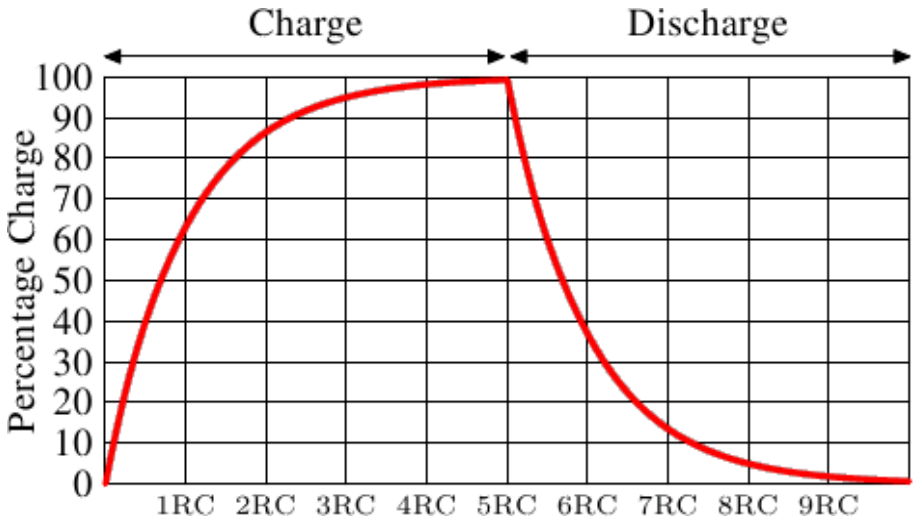
(Dynamic RAM – DRAM)



Main memory

- Use SDRAM (Synchronous DRAM) technology
  - Access is synchronized by an external clock

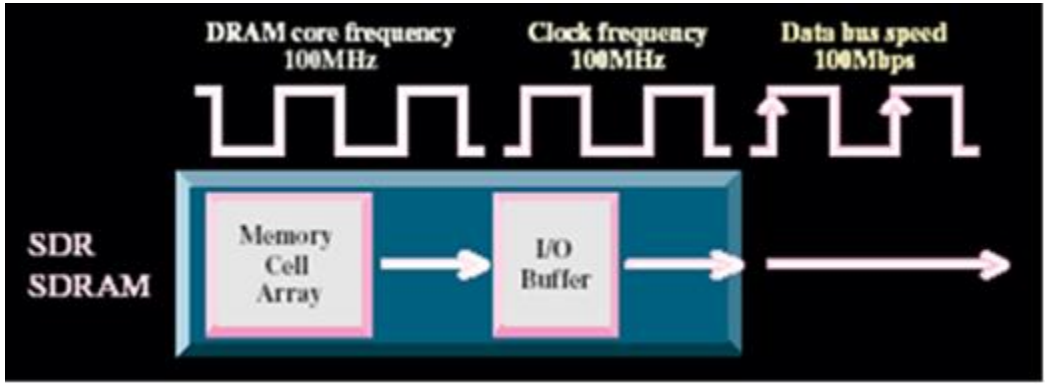




SDR-SDRAM

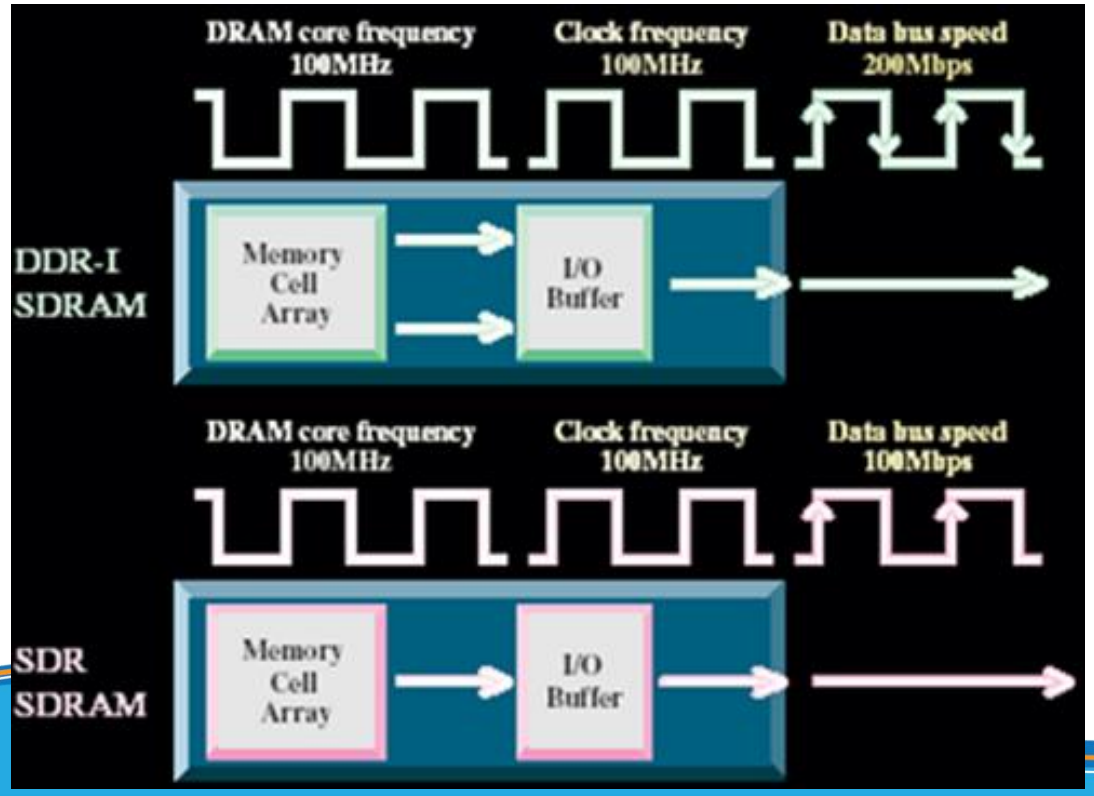
- SDR-SDRAM (Single Data Rate SDRAM)
- DIMM 168-pin (Dual In-line Memory Module
- Data bus: 64 bit

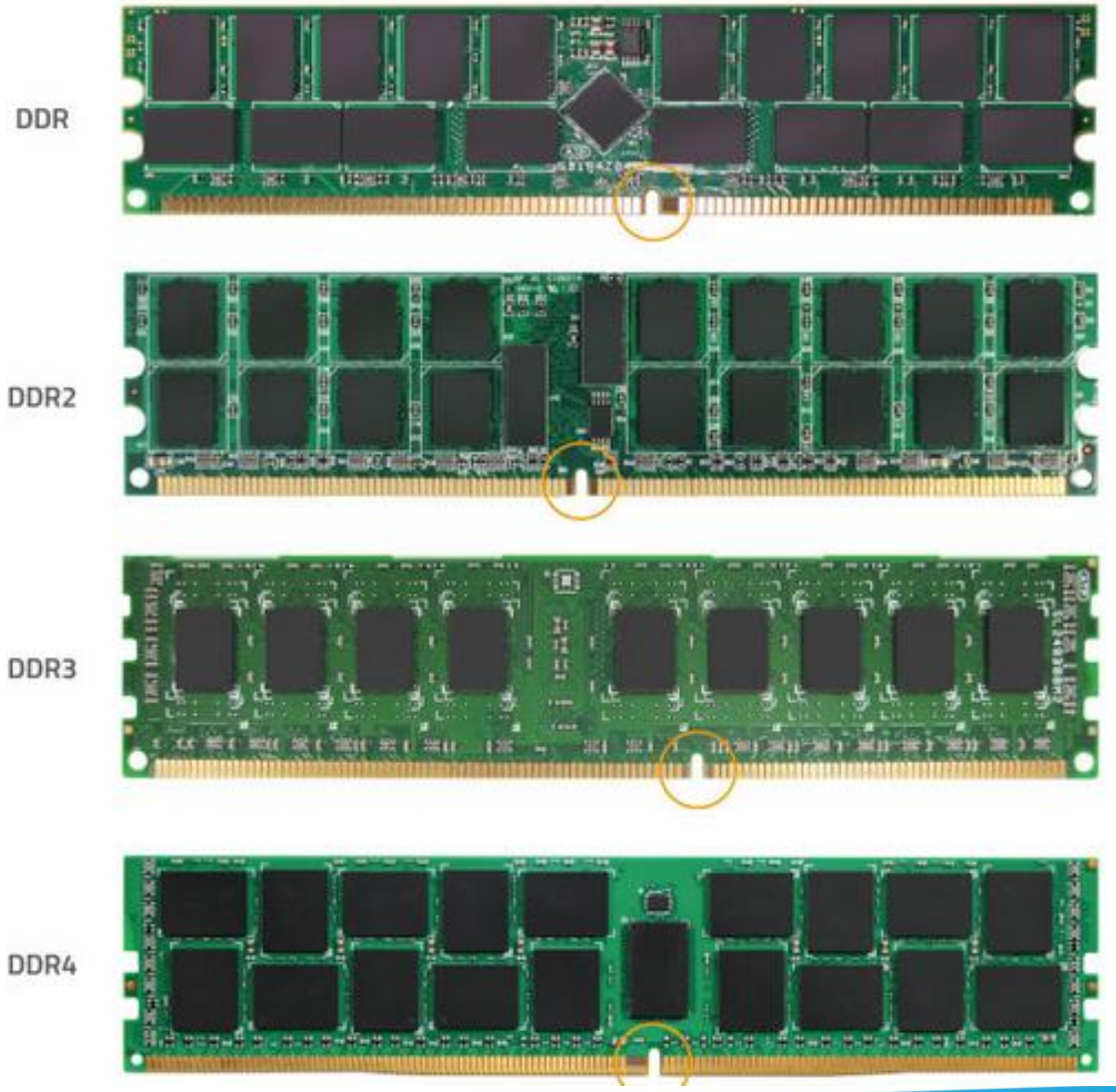
Standard name	Memory clock	Cycle time	I/O bus clock	Peak transfer rate
SDR-66	66 MHz	15 ns	66 MHz	528 MB/s
SDR-100	100 MHz	10 ns	100 MHz	800 MB/s
SDR-133	133 MHz	7.5 ns	133 MHz	1064 MB/s



DDR-SDRAM (1/2)

- DDR-SDRAM (Double Data Rate)
  - DDR1-SDRAM (184-pin DIMM)
  - DDR2-SDRAM (240-pin DIMM)
  - DDR3-SDRAM(240-pin DIMM)
  - DDR4-SDRAM(288-pin DIMM)

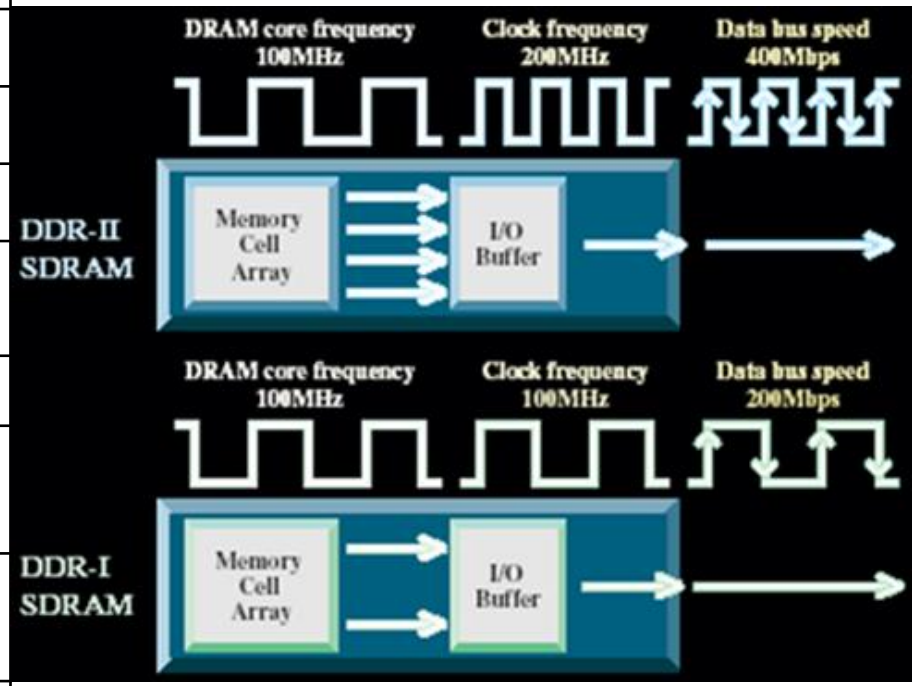




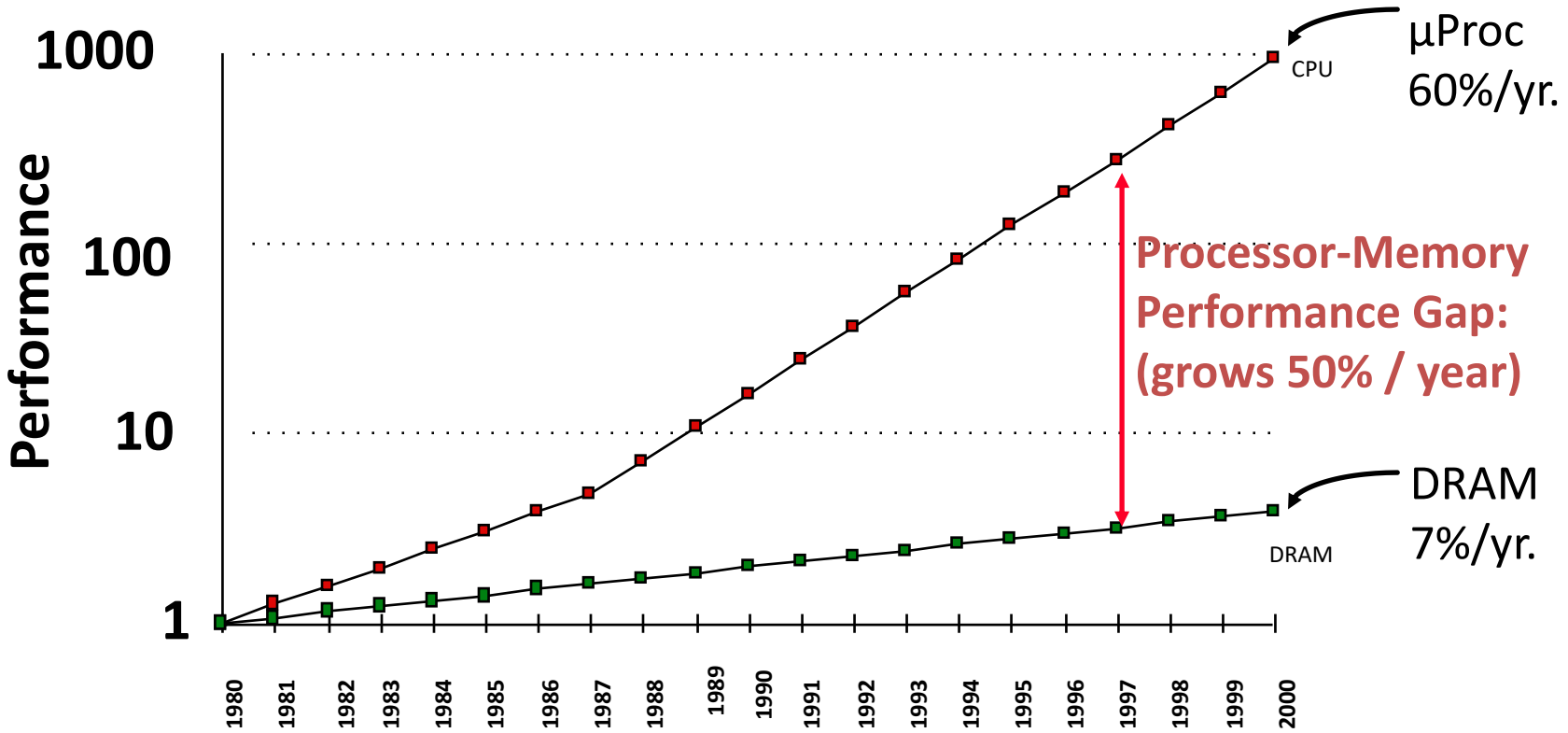
DDR-SDRAM (2/2)

- Data bus: 64 bit

Standard name	Memory clock	Cycle time	I/O Bus clock	Data transfers per second	Module name	Peak transfer rate
DDR-200	100 MHz	10 ns	100 MHz	200 Million	PC-1600	1600 MB/s
DDR-400	200 MHz	5 ns	200 MHz	400 Million	PC-3200	3200 MB/s
DDR2-400	100 MHz	10 ns	200 MHz	400 Million	PC2-3200	3200 MB/s
DDR2-1066	266 MHz	3.75 ns	533 MHz	1066 Million	PC2-8500 PC2-8600	8533 MB/s
DDR3-800	100 MHz	10 ns	400 MHz	800 Million	PC3-6400	6400 MB/s
DDR3-2133	266 MHz	3.75 ns	1066 MHz	2133 Million	PC3-17000	17066 MB/s
DDR4-1600	200 MHz	5 ns	800 MHz	1600 Million	PC4-12800	12800 MB/s
DDR4-3200	400 MHz	2.5 ns	1600 MHz	3200 Million	PC4-25600	25600 MB/s



Cache



- Using SRAM technology, faster than main memory (using DRAM technology)
- Acts as a fast access buffer (intermediate between CPU and main memory)
- Temporarily stores a partial copy of main memory's contents to reduce accessing to main memory

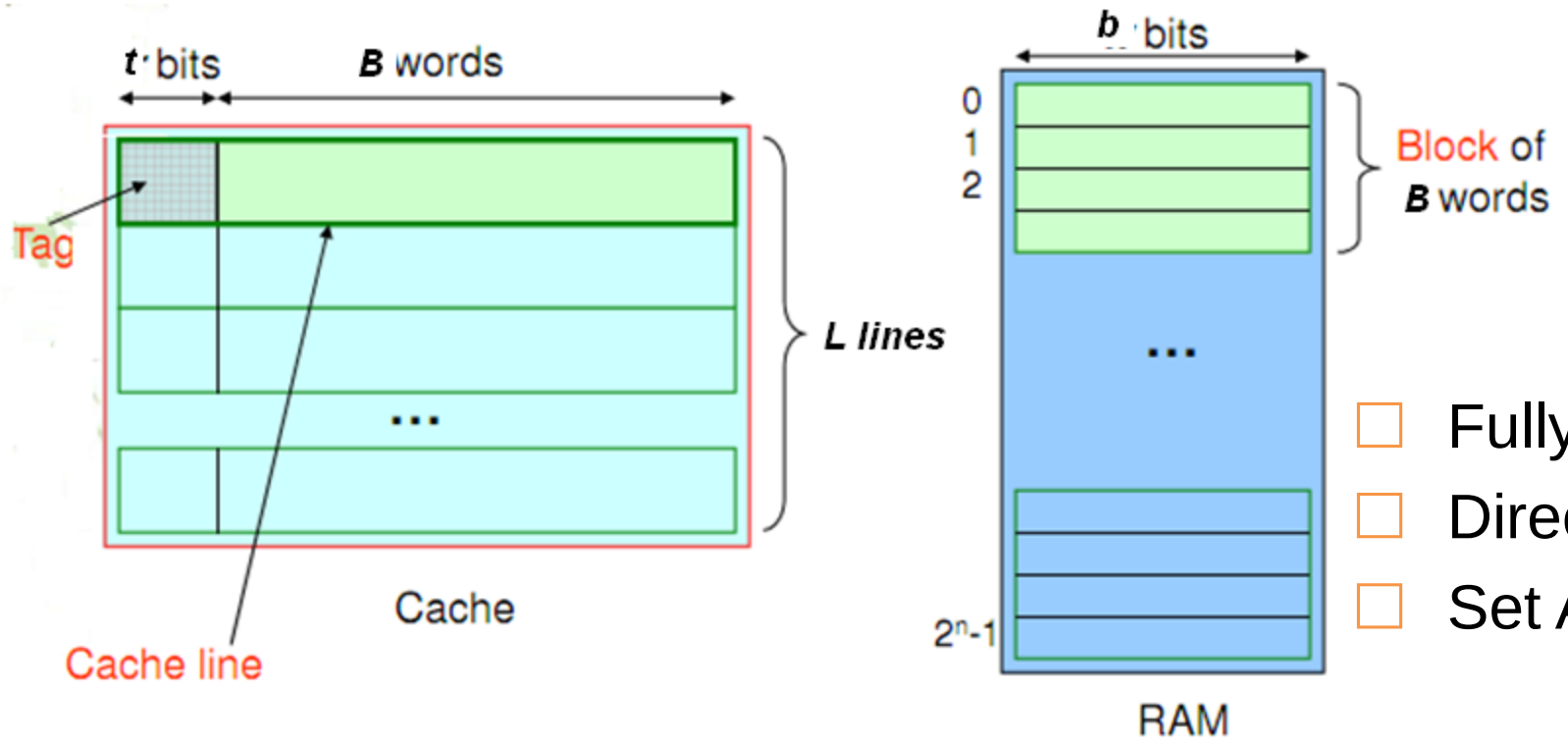
Principle of Operation

- Two principles of Locality
  - Temporal locality
    - Items accessed recently are likely to be accessed again soon
    - e.g., instructions in a loop, induction variables
  - Spatial locality
    - Items near those accessed recently are likely to be accessed soon
    - E.g., sequential instruction access, array data
- When CPU/IO needs to read data from main memory
  - Check if it's already in cache or not ?
  - If Yes (cache hit): read the contents of the cache, no need to access main memory
  - If No (cache miss): copy the memory blocks containing the data to be read from main memory to cache and then to CPU/IO. The time it takes to process a cache miss is called a miss penalty

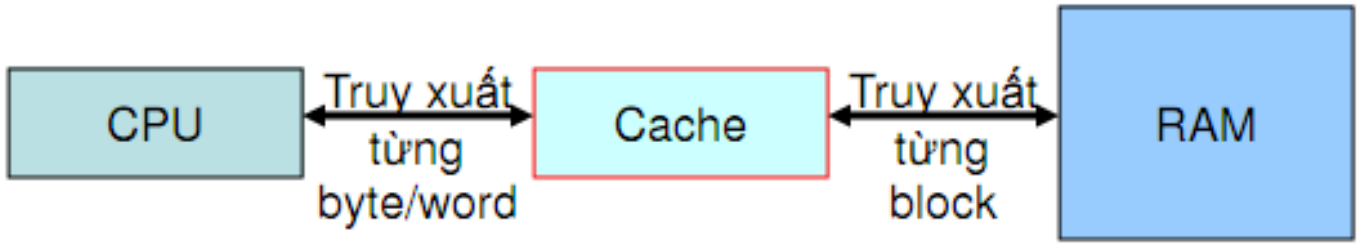
Cache Design

- Cache Organization
  - When you need to access a memory cell, how do you know if that cell is already in the cache? Then how to identify that place ?
- Replacement strategy
  - When the cache has no more space but still needs to hold another data from main memory, how to identify which data will be replaced ?
- Write Policy
  - If content of a cache element is changed/modified, how this change is reflected to main memory and vice versa ?
- Cache size
- Size of an element of cache
- Number and level of caches

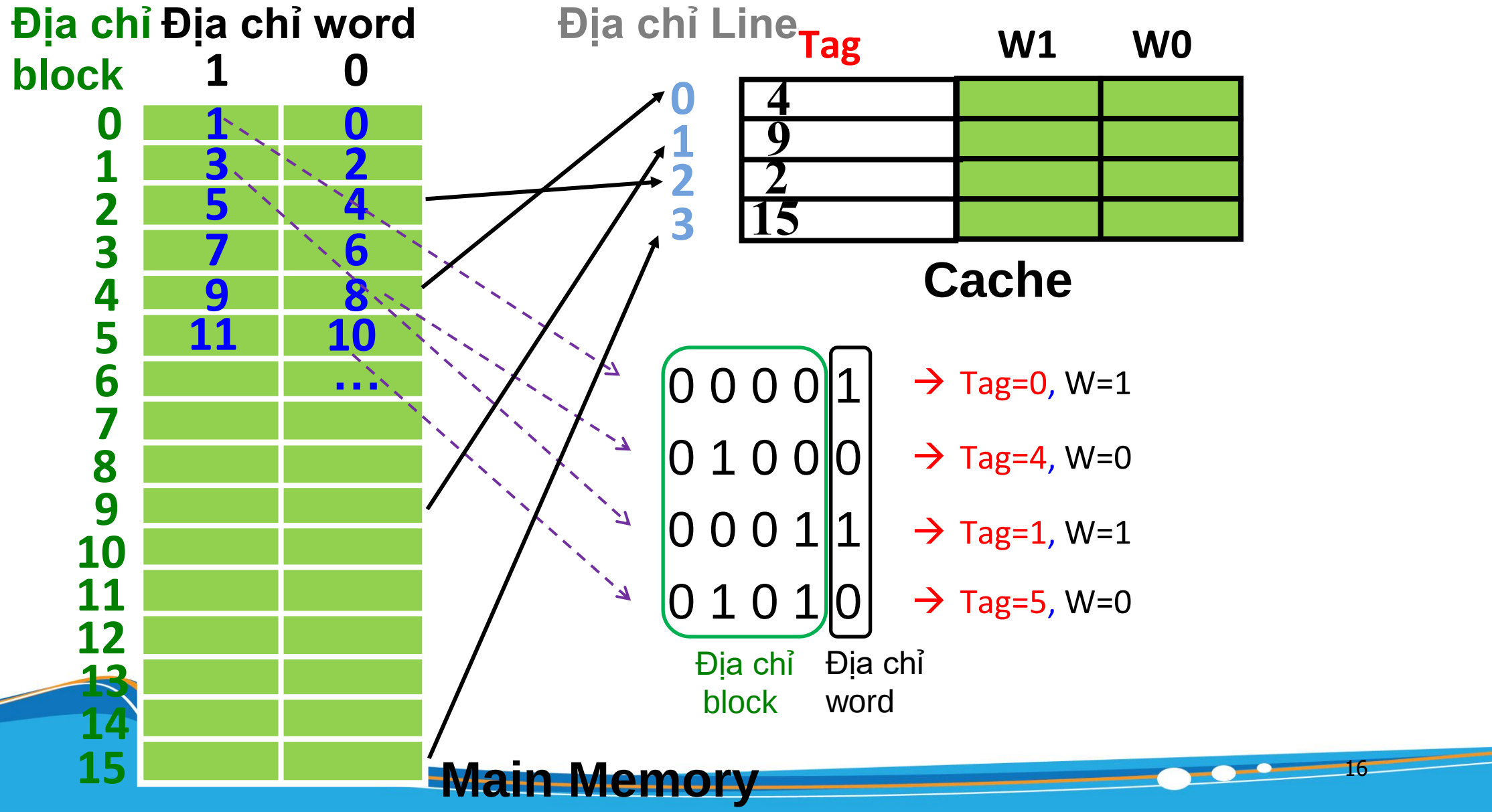
Cache Organization



- Fully Associate Mapping
- Direct Mapping
- Set Associate Mapping



Fully Associate Mapping (1/2)

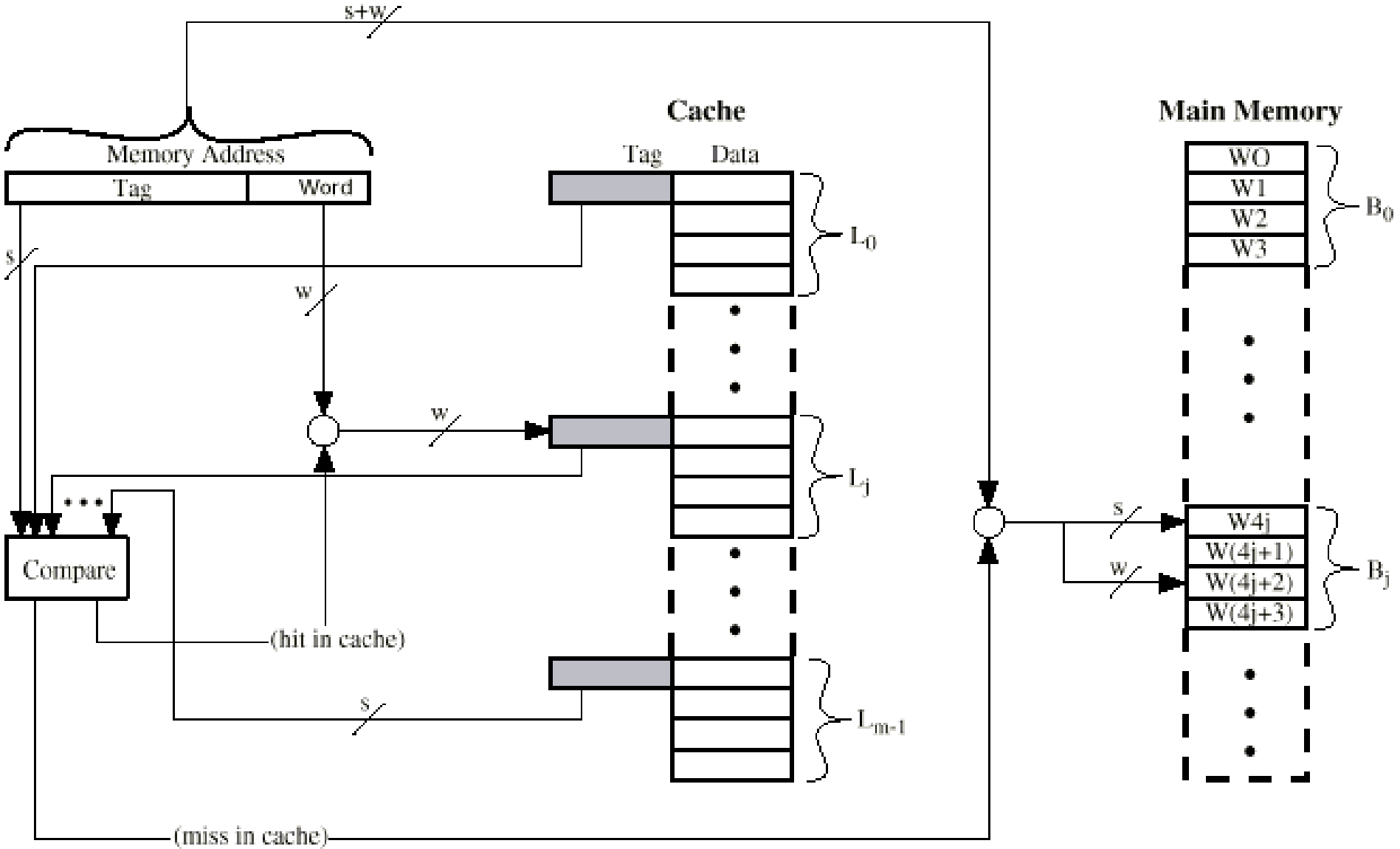


Fully Associate Mapping (2/2)

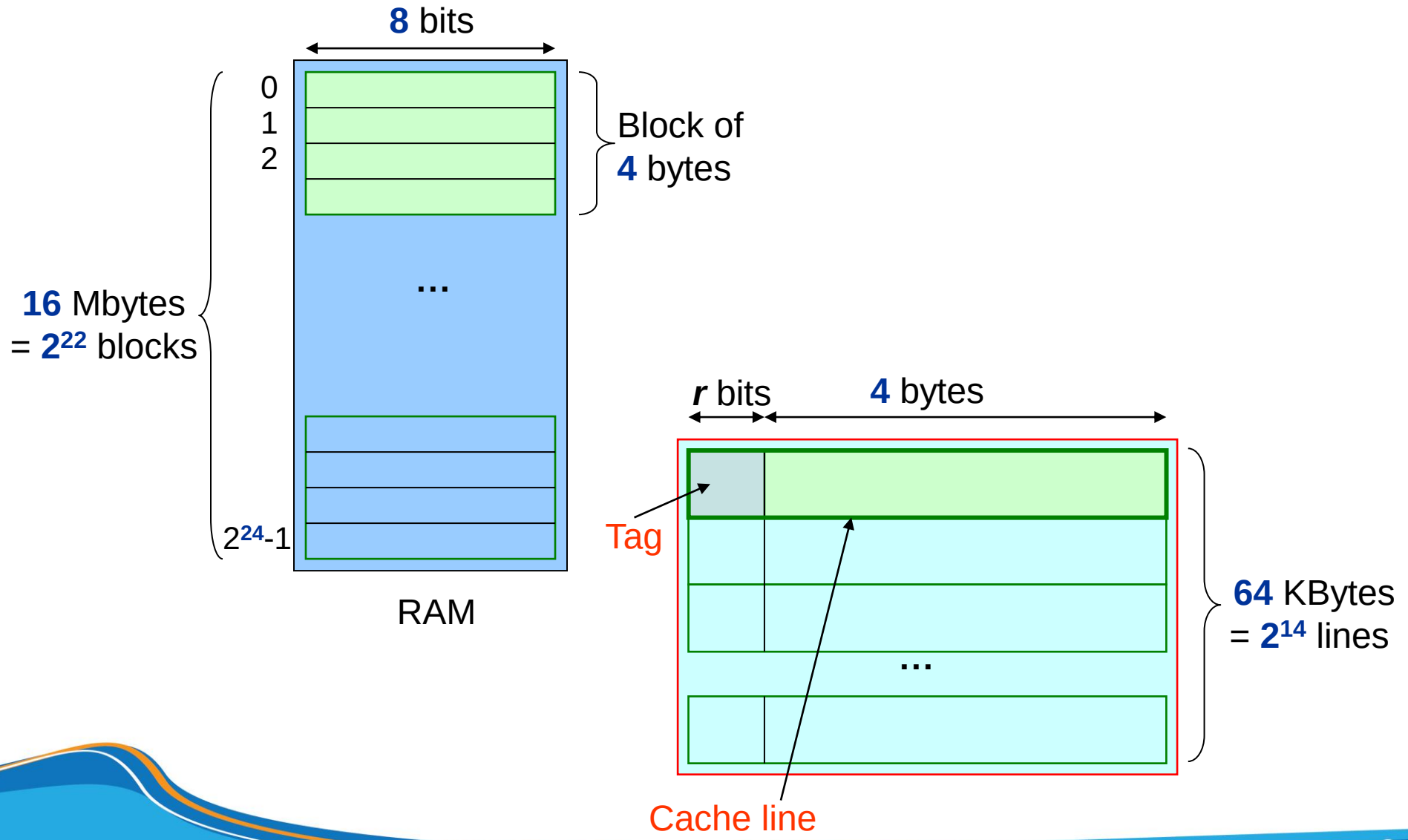
- A memory block can be stored into any cache line
- The memory address will have the following structure

  - s - number of bits identifying the block address, used to determine which block is stored in the cache line
  - w - number of bits identifying the word address in a block
- Since a block can be stored into any line, to determine whether the block is already in the line or not, all lines must be traversed.

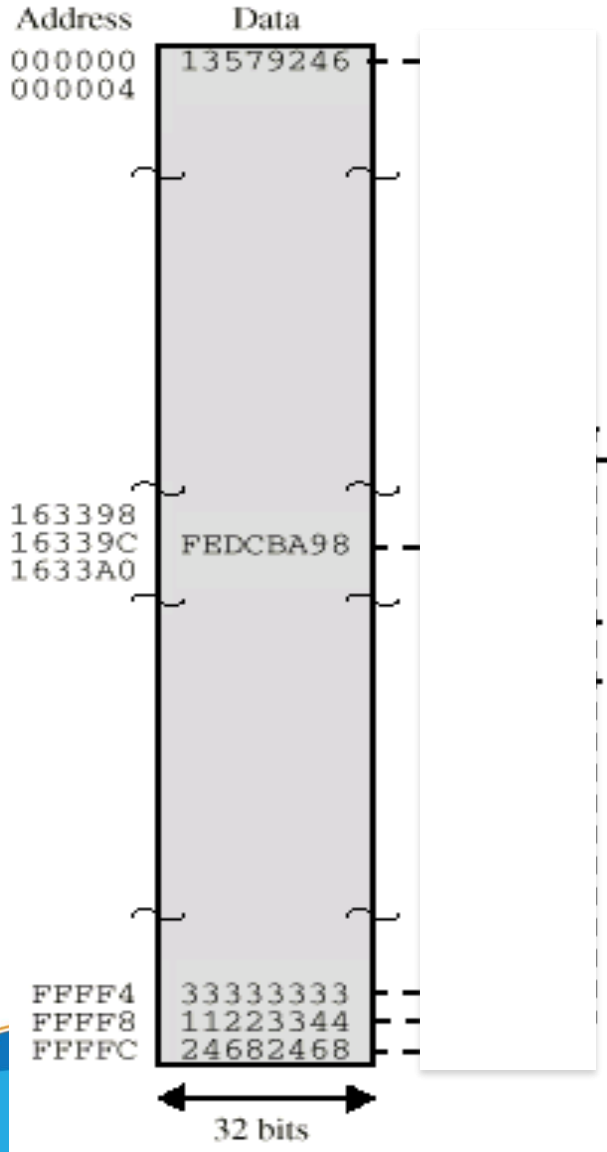
Fully Associate Mapping – Data Access

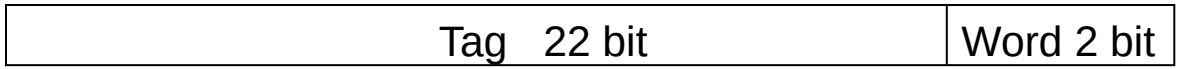


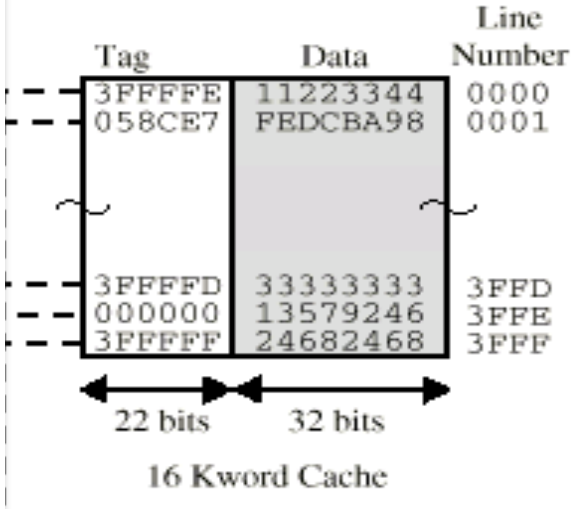
Fully Associate Mapping – Example (1/2)



Fully Associate Mapping – Example (2/2)







Memory address has 24 bit

- 2 bit for word address (4 byte / block)
- 22 bit for block address, which is Tag

Mem	**1**	**6**	**3**	**3**	**9**	**C**	**(hex)**
Addr	**0001**	**0110**	**0011**	**0011**	**1001**	**1100**	**(binary)**
Tag	**0**	**5**	**8**	**C**	**E**	**7**	**(hex)**
	00	**0101**	**1000**	**1100**	**1110**	**0111**	**(binary)**

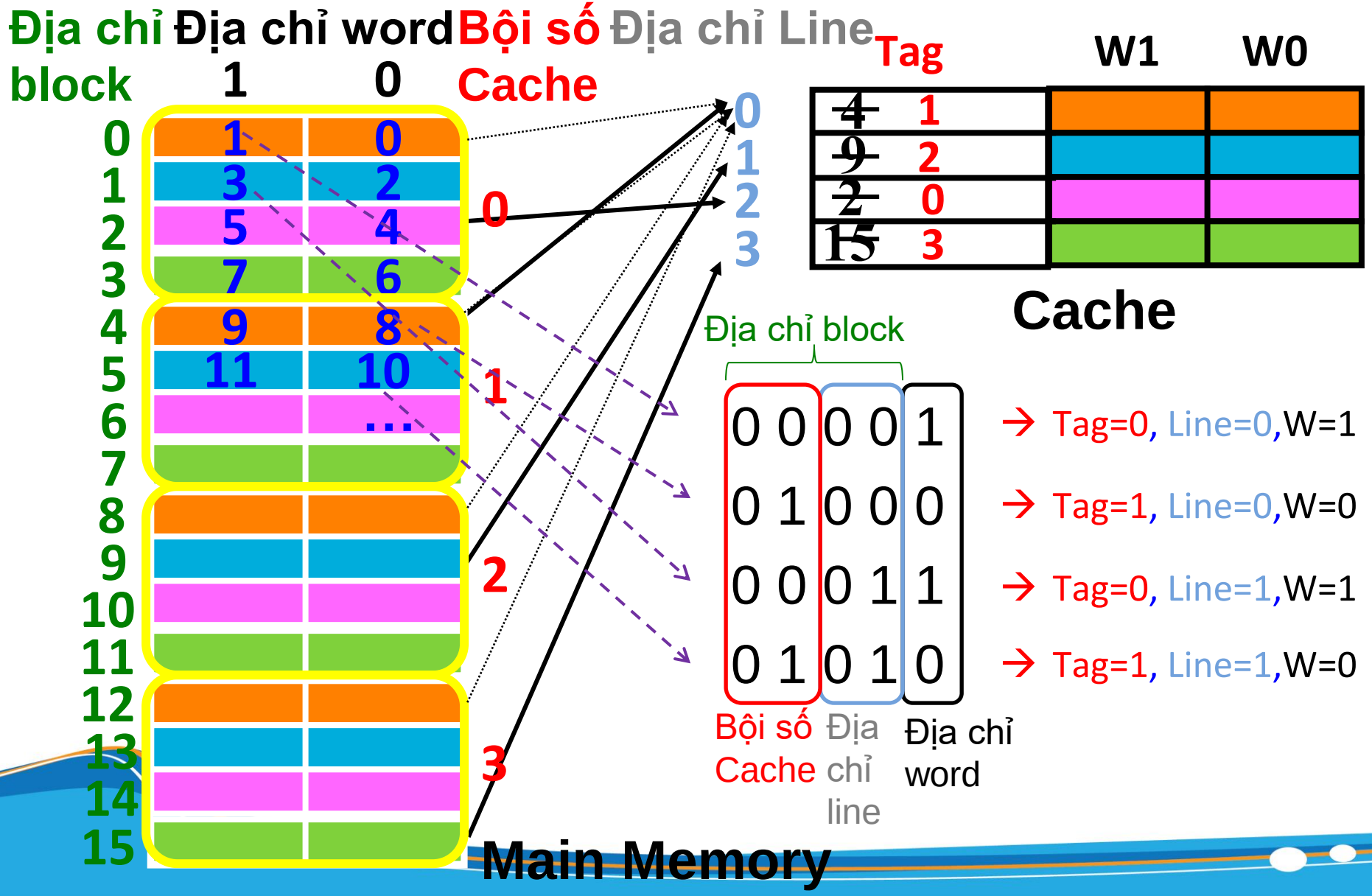
Fully Associate Mapping – Pros & Cons

(–) Store more bits for a Tag in cache

(–) High search cost

(–) What happens if a block needs to be loaded into the cache and there are no empty lines?

Direct Mapping (1/2)



Direct Mapping (2/2)

- Each block of main memory B_j is uniquely mapped into a cache element (line) L_i as following

  $$L_i = B_j \bmod L$$

  where L is the number of cache elements

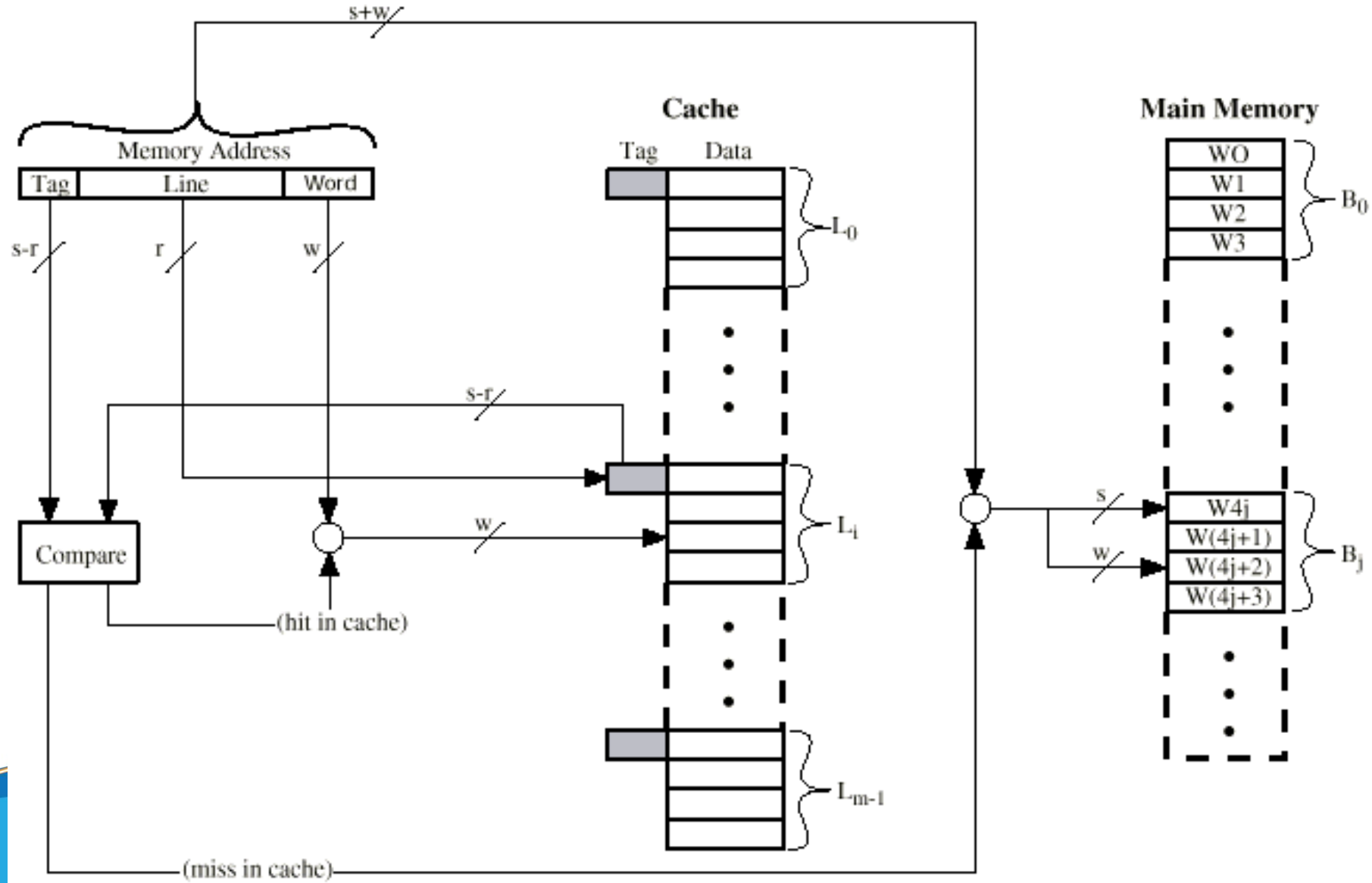
Cache line	Main Memory blocks
0	0, L, 2L, 3L, …
1	1, L+1, 2L+1, …
…	…
L-1	L-1, Lm-1, Lm-1, …

- The memory address will have the following structure

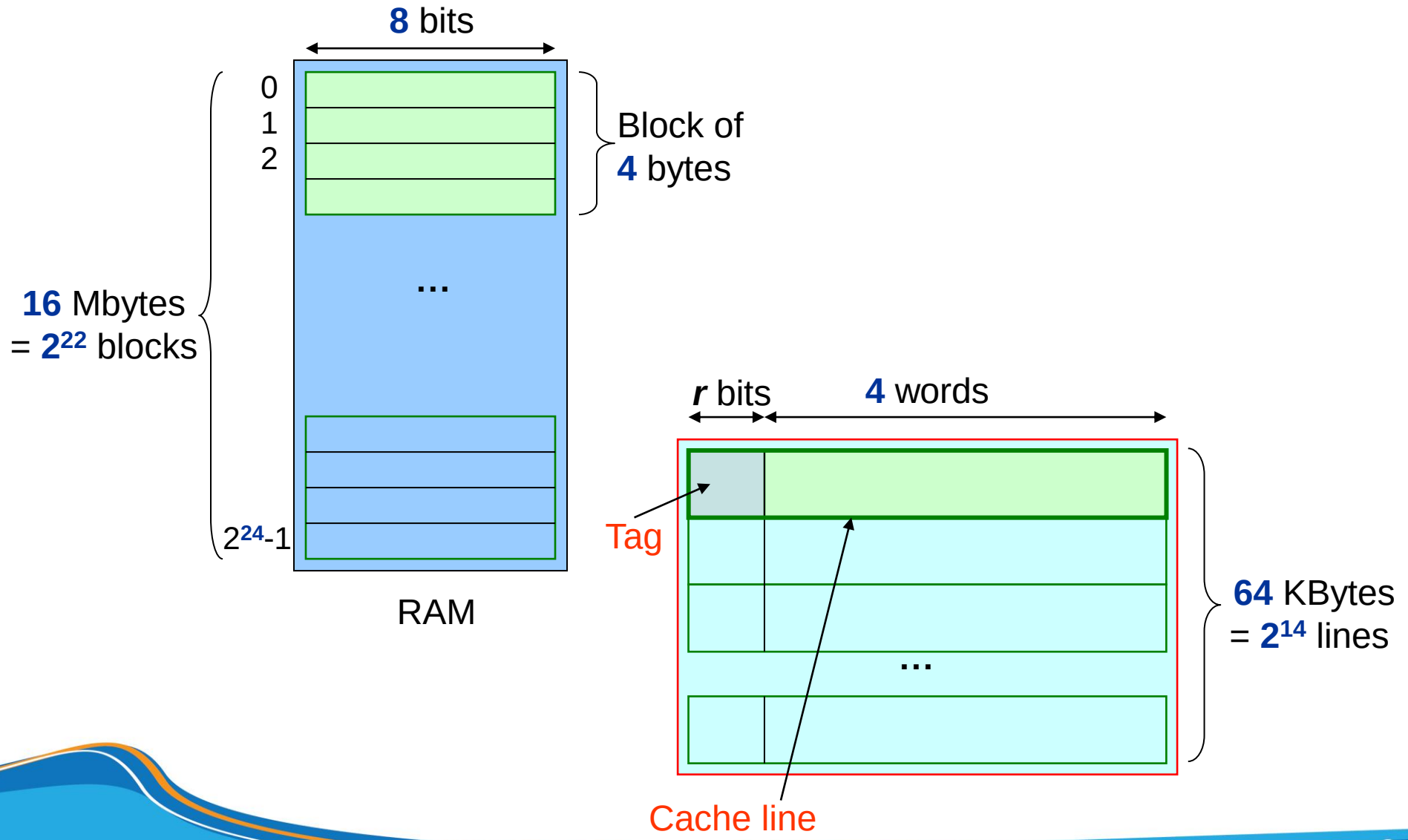
Tag ***s-r***	Cache line ***r***	Word ***w***

  - s - number of bits identifying the block address
  - w - number of bits identifying the word address in a block
  - r - number of bits identifying cache line address
  - (s – r) - the number of bits that determine which block stored in the cache line

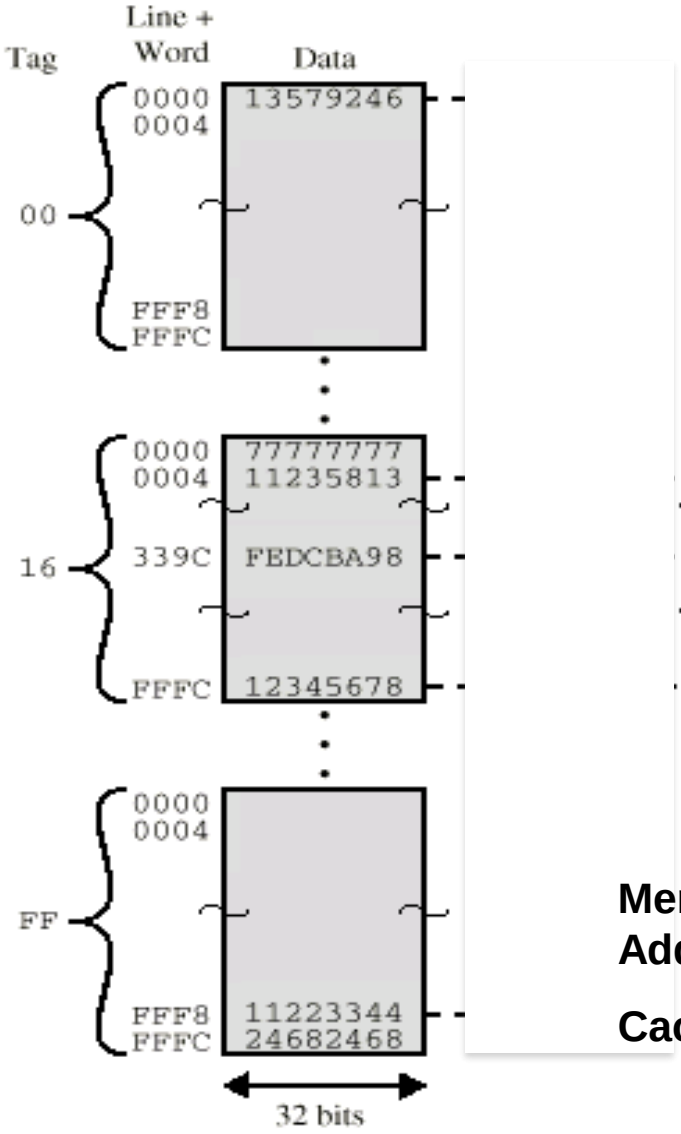
Direct Mapping – Data Access



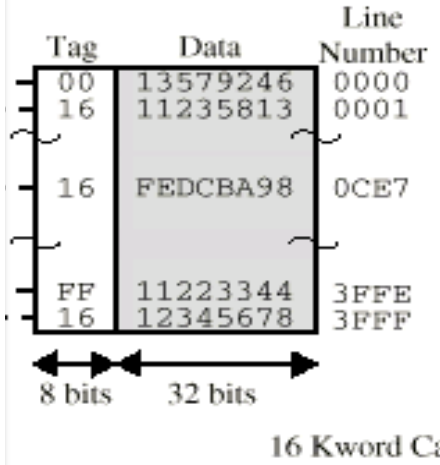
Direct Mapping – Example (1/2)



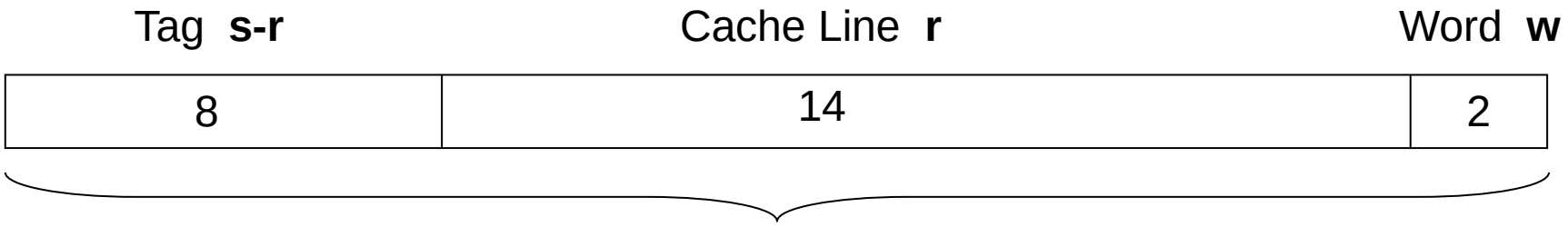
Direct Mapping – Example (2/2)



16 MByte Main Memory



16 Kword Cache



Memory address has 24 bit

- 2 bit for word address (4 byte / block)
- 22 bit for block address
  - 14 bit for cache line number
  - 8 bit for tag (= 22-14)

Mem Addr	**1** **0001**	**6** **0110**	**3** **0011**	**3** **0011**	**9** **1001**	**C** **1100**	**(hex)** **(binary)**
Cache Line			**0** **00**	**C** **1100**	**E** **1110**	**7** **0111**	**(hex)** **(binary)**

Cache line	Starting memory address of block
0	**00**0000, **01**0000, …, **FF**0000
1	**00**0004, **01**0004, …, **FF**0004
…	…
2^{14} -1	**00**FFFC, **01**FFFC, …, **FF**FFFC

Direct Mapping – Pros & Cons

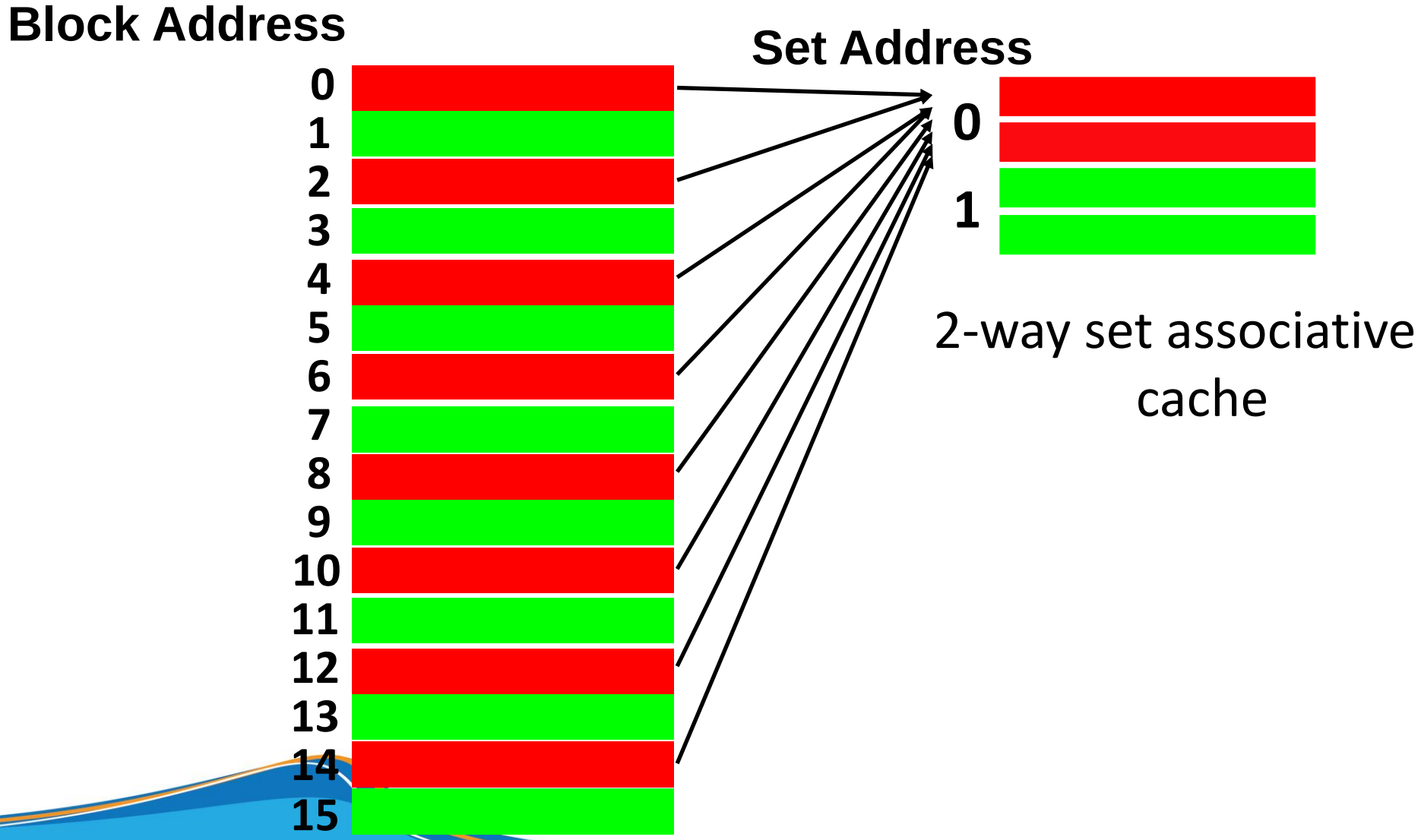
(+) Simple

(+) Fast

(–) Not flexible

- Each block must be fixed at a cache line
- If a block (e.g, block L) needs to be stored into a line (line 0) but this line is storing another block (e.g, block 2L), then block 2L must be overrriden by block L, although cache still has a lot of empty lines
- What happens if a process needs to access blocks which must be stored the same line (e.g, L, 2L, 3L, …) continuously ?

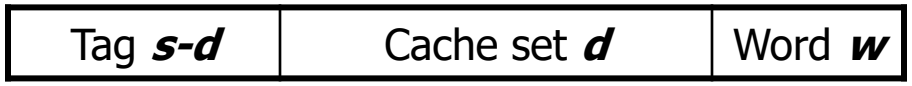
K-way Set Associate Mapping (1/2)



K-way Set Associate Mapping (2/2)

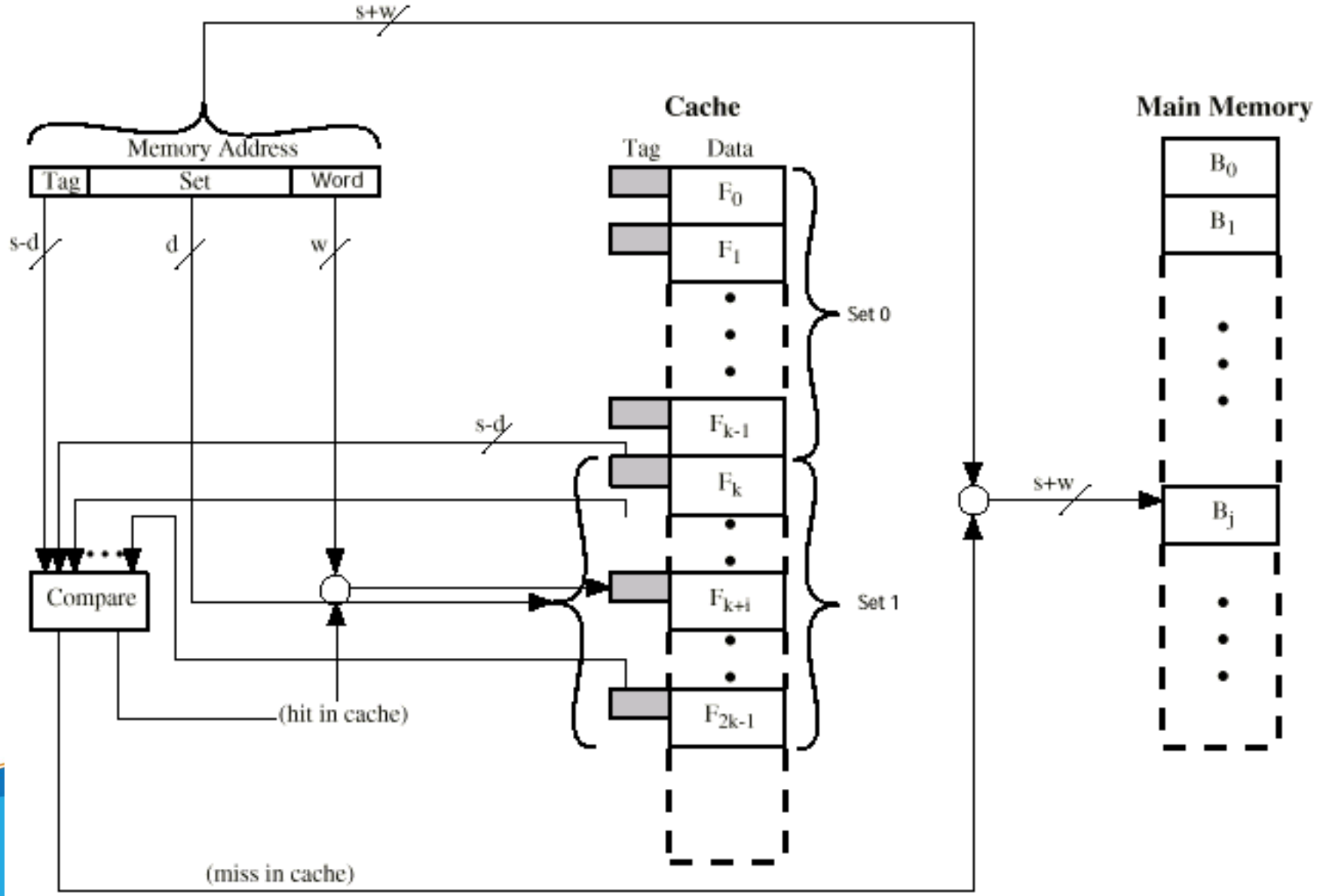
- Combine ideas between direct mapping and full associate mapping
  - The cache lines are divided into S sets, each of which has K lines
  - Each block B_j is mapped (directly) to only one set S_i as following

    $S_i = B_j \bmod S$
  - Lines in a set are managed as in full associate mapping
- The memory address will have the following structure

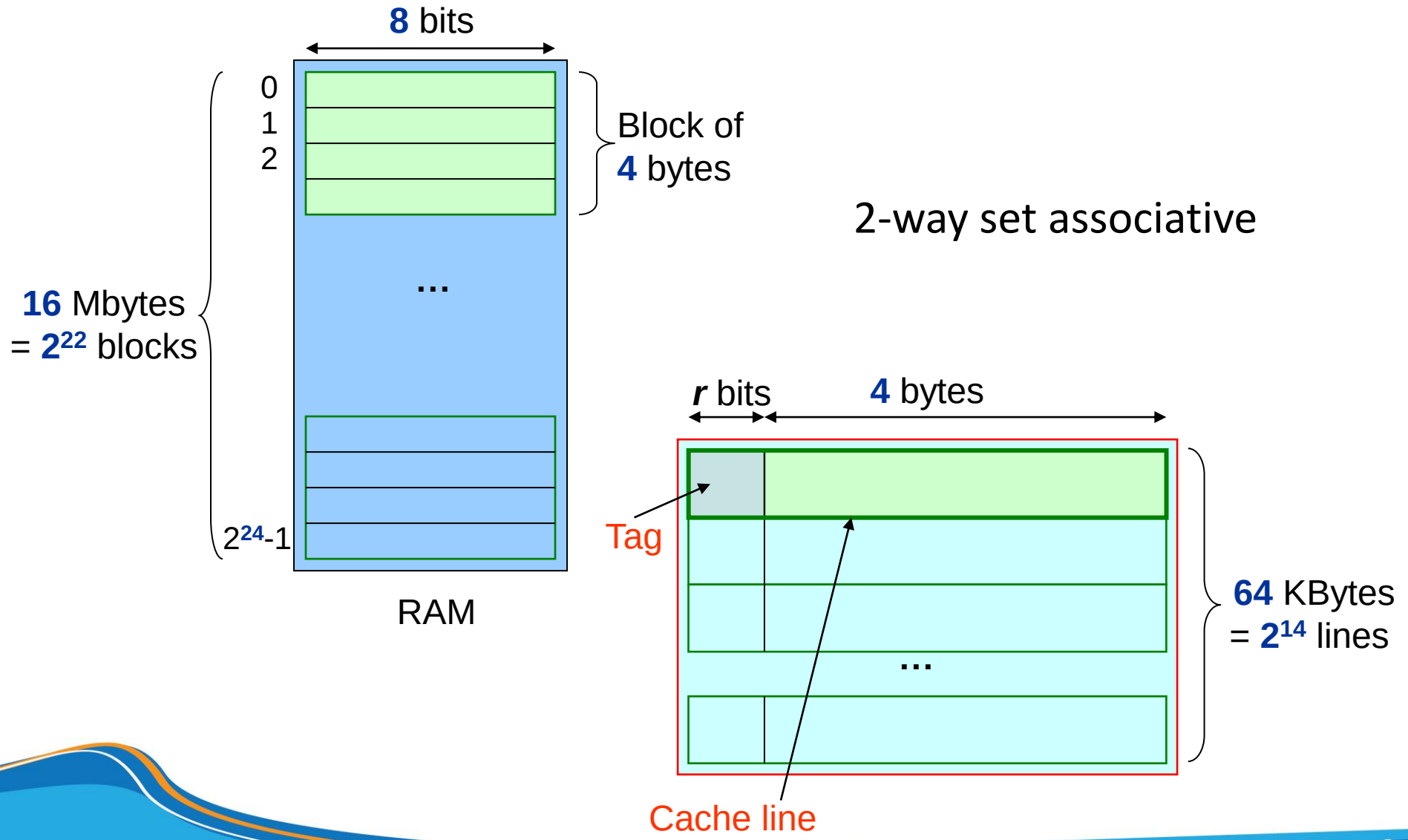


  - s - number of bits identifying the block address
  - w - number of bits identifying the word address in a block
  - d - number of bits identifying cache set address
  - (s – d) - the number of bits that determine which block stored in the cache line

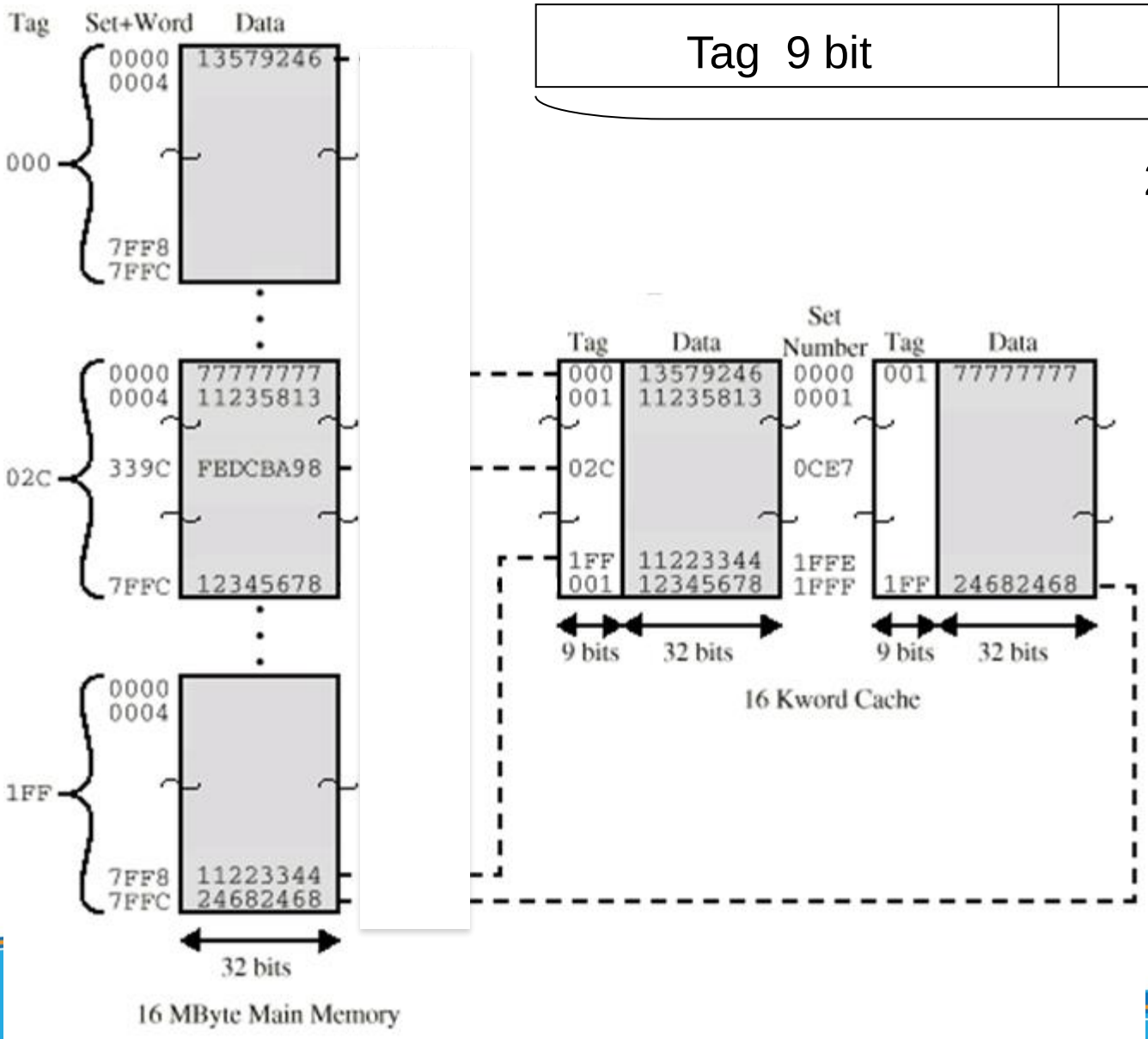
Set Associate Mapping – Data Access



Set Associate Mapping – Example (1/2)



Set Associate Mapping – Example (2/2)



- Memory address has 24 bit
  - 2 bit for word address (4 byte / block)
  - 22 bit for block address
    - 13 bit for cache set number
      - Blocks with address 000000, 008000, …, FF8000 are mapped into same set 0
    - 9 bit for tag (= 22-13)

Memory Address 1111 1111 1111 1111 1111 1100
Tag Set + Word Set Number Data
1FF 7FFC 1FFF 24682468
Memory Address 0001 0110 0111 1111 1111 1100
Tag Set + Word Set Number Data
02C 7FFC 1FFF

Set Associate Mapping – Pros & Cons

- S=L, K=1 → direct mapping
- S=1, K=L → fully associative mapping
- (–) Same as full associate mapping, what happens if a block needs to be stored into a set but there are no empty line in that set ?

Replacement Strategy

- Problem
  - When you need to store a new block into the cache but can't find an empty line as required, which line should be replaced ?
- Direct Mapping
  - No choice
- Fully Associate Mapping and Set Associate Mapping
  - Random
  - FIFO (First In First Out)
  - LRU (Least Recently Used)
  - LFU (Least Frequently Used)

Write Policy

- Problem
  - CPU can change content of cache lines. I/O may address main memory directly
  - If content of line is changed in cache, how this change is reflected to the corresponding block and vice versa ?
- Write through
  - All writes go to main memory as well as cache
  - Lots of traffic, slows down writes
- Write back
  - Updates initially made in cache only and “update bit” for cache slot is set.
  - Then if the block stored in a line is to be replaced, write to main memory only if update bit is set
  - I/O must access main memory through cache

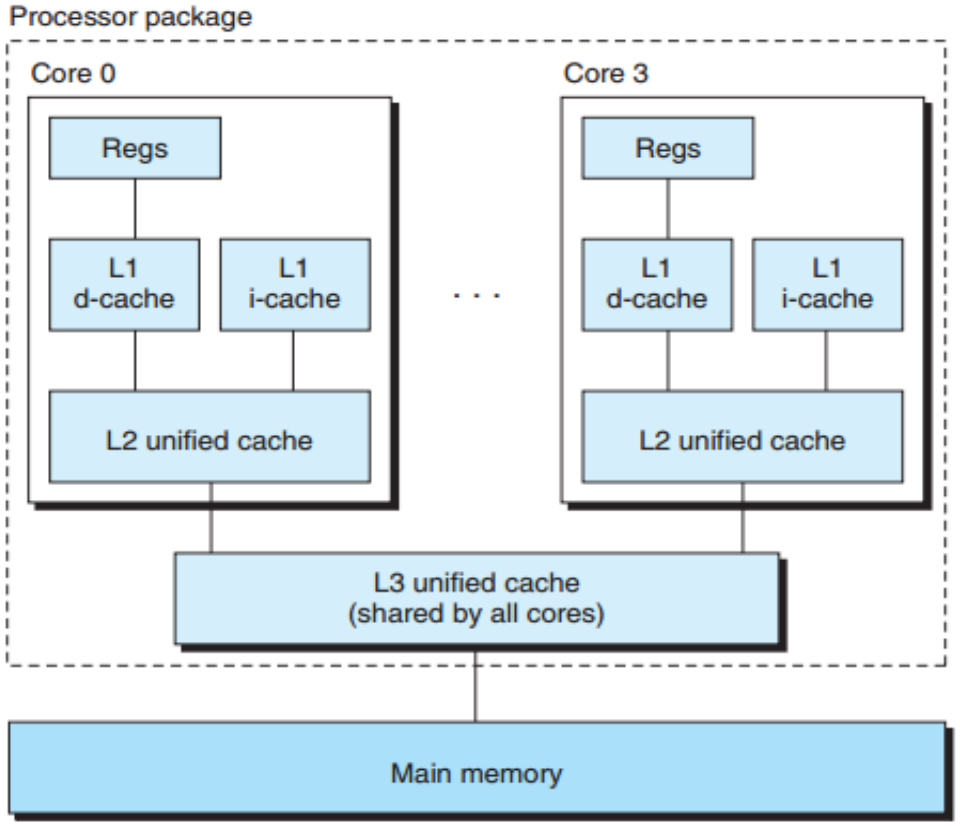
Block size and Cache size

- Block size
  - Too small: decrease spatial locality
  - Too big: take long time to transfer block to cache (miss penalty)
  - Block size is usually from 8 to 64 bytes
- Cache size
  - Cache size is proportional to cost, is it proportional to speed ?
  - Too small: the number of blocks that can be cached is small, leading to a high cache miss rate
  - Too big: lots of unnecessary cached content. It takes a long time to check if the block is in the cache or not

Type and Level of Cache

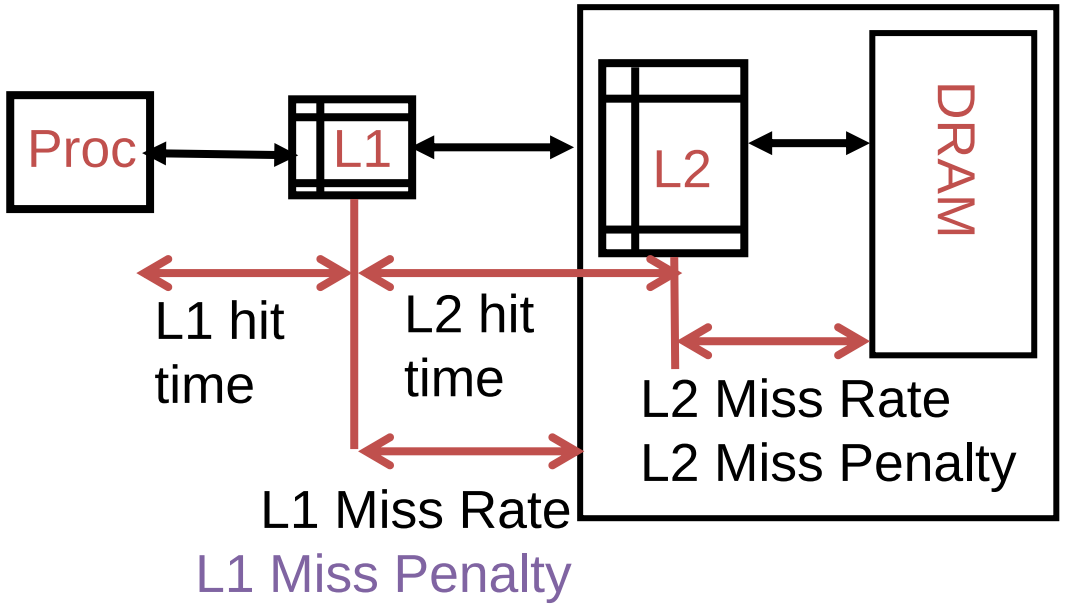
- Types of Cache
  - On-chip cache vs Off-chip cache
  - Unified cache vs Split cache
- Cache levels: L1, L2, L3, …
  - L1 cache
    - Size 10s KB
    - Hit time: 1s cycles
    - Miss rate: 1-5%
  - L2 cache
    - Size 100s KB
    - Hit time: 10s cycles
    - Miss rates: 10-20%

Characteristic of the Intel core i7 cache hierarchy



Cache type	Access time (cycles)	Cache size (C)	Assoc. (E)	Block size (B)	Sets (S)
L1 i-cache	4	32 KB	8	64 B	64
L1 d-cache	4	32 KB	8	64 B	64
L2 unified cache	10	256 KB	8	64 B	512
L3 unified cache	40–75	8 MB	16	64 B	8,192

Average Memory Access Time



AMAT (Average Memory Access Time) =
L1 Hit Time + L1 Miss Rate * L1 Miss Penalty

L1 Miss Penalty =
L2 Hit Time + L2 Miss Rate * L2 Miss Penalty

AMAT = L1 Hit Time + L1 Miss Rate*
(L2 Hit Time + L2 Miss Rate * L2 Miss Penalty)

AMAT - Example

- Assumption
  - L1 Hit Time = 1 cycle
  - L1 Miss rate = 5%
  - L2 Hit Time = 5 cycles
  - L2 Miss rate = 15%
  - L1, L2 Miss Penalty = 200 cycles
- Don’t use cache L2
  - AMAT = 1 + 0.05 x 200 = 11 cycles
- Use cache L2
  - L1 miss penalty = 5 + 0.15 * 200 = 35
  - AMAT = 1 + 0.05 x 35 = 2.75 cycles

→ Using L2 is 4 times faster than not using L2

Writing cache-friendly code

- Make the common case go fast:
  - The core functions of the program
  - Especially the loops inside functions
- Minimize the number of cache misses in each inner loop:
  - The total number of loads and stores, loops with higher hit rates will run faster

→ Maximize the temporal locality in your programs by using a data object as often as possible once it has been read from memory

→ Maximize the spatial locality in your programs by reading data objects sequentially, in the order they are stored in memory

Writing cache-friendly code

- Example: Compare the following two code snippets
  - 4-byte words, 4-word cache lines

```
int sumarr(int a[M][N])
{
        int i, j, sum =0;
        for (i=0; i<M; i++)
                for(j=0; j<N; j++)
                        sum += a[i][j];
        return sum;
}
```

a[i][j]	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	j = 7
i = 0	1 **[m]**	2 [h]	3 [h]	4 [h]	5 **[m]**	6 [h]	7 [h]	8 [h]
i = 1	9 **[m]**	10 [h]	11 [h]	12 [h]	13 **[m]**	14 [h]	15 [h]	16 [h]
i = 2	17 **[m]**	18 [h]	19 [h]	20 [h]	21 **[m]**	22 [h]	23 [h]	24 [h]
i = 3	25 **[m]**	26 [h]	27 [h]	28 [h]	29 **[m]**	30 [h]	31 [h]	32 [h]

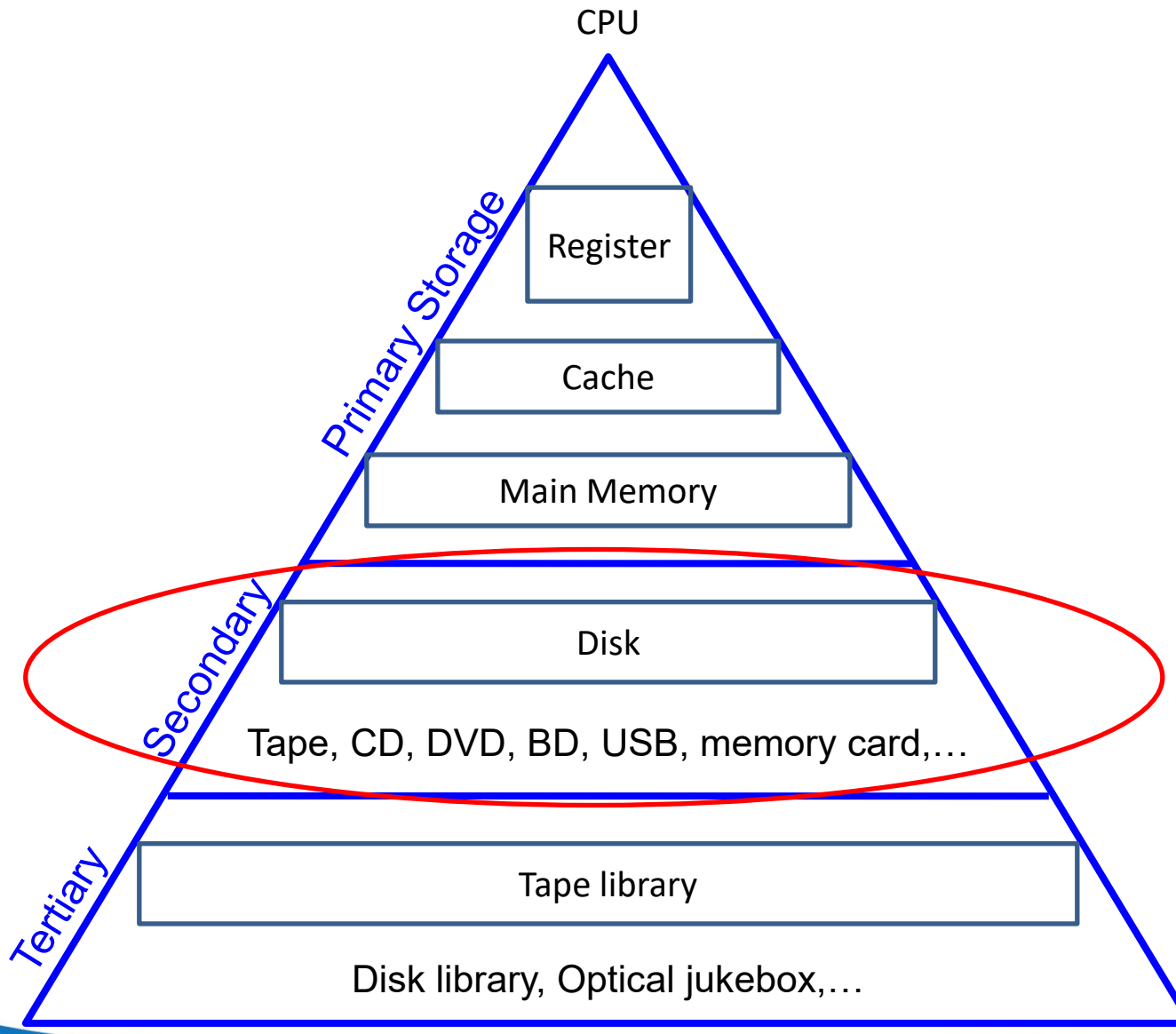
Miss rate = ¼ = 25%

```
int sumarr(int a[M][N])
{
        int i, j, sum =0;
        for (j=0; j<N; j++)
                for(i=0; i<M; i++)
                        sum += a[i][j];
        return sum;
}
```

a[i][j]	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	j = 7
i = 0	1 **[m]**	5 **[m]**	9 **[m]**	13 **[m]**	17 **[m]**	21 **[m]**	25 **[m]**	29 **[m]**
i = 1	2 **[m]**	6 **[m]**	10 **[m]**	14 **[m]**	18 **[m]**	22 **[m]**	26 **[m]**	30 **[m]**
i = 2	3 **[m]**	7 **[m]**	11 **[m]**	15 **[m]**	19 **[m]**	23 **[m]**	27 **[m]**	31 **[m]**
i = 3	4 **[m]**	8 **[m]**	12 **[m]**	16 **[m]**	20 **[m]**	24 **[m]**	28 **[m]**	32 **[m]**

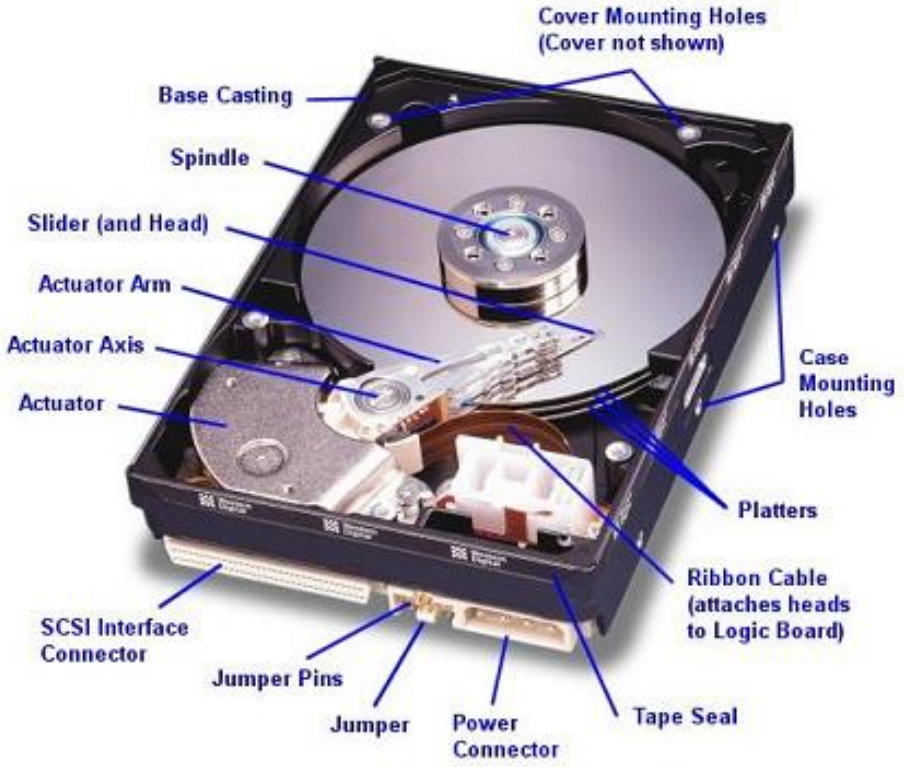
Miss rate = 100%

Secondary Storage

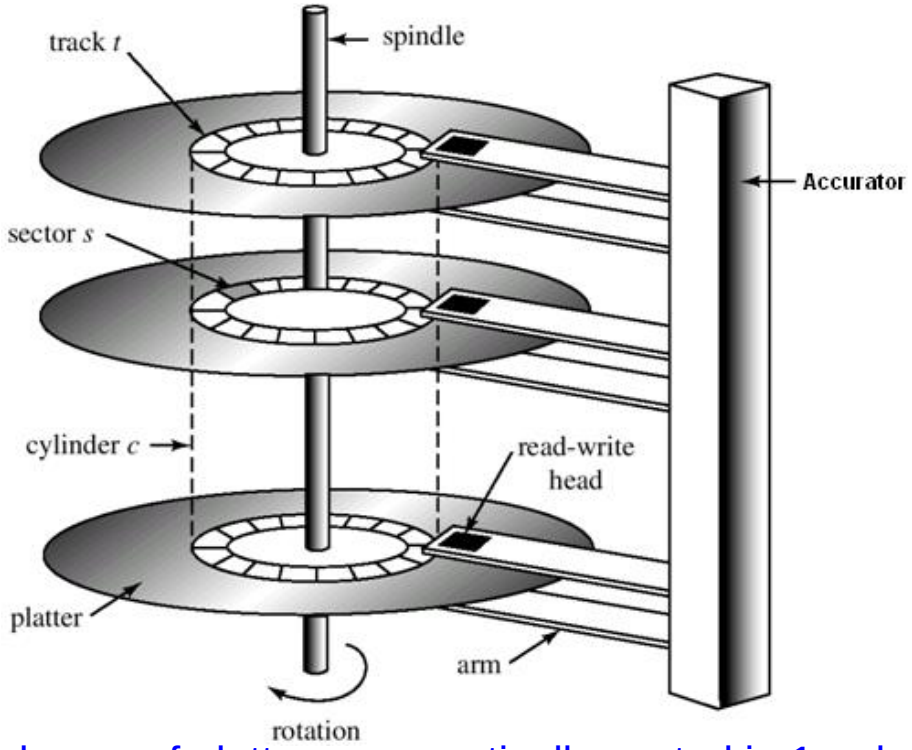


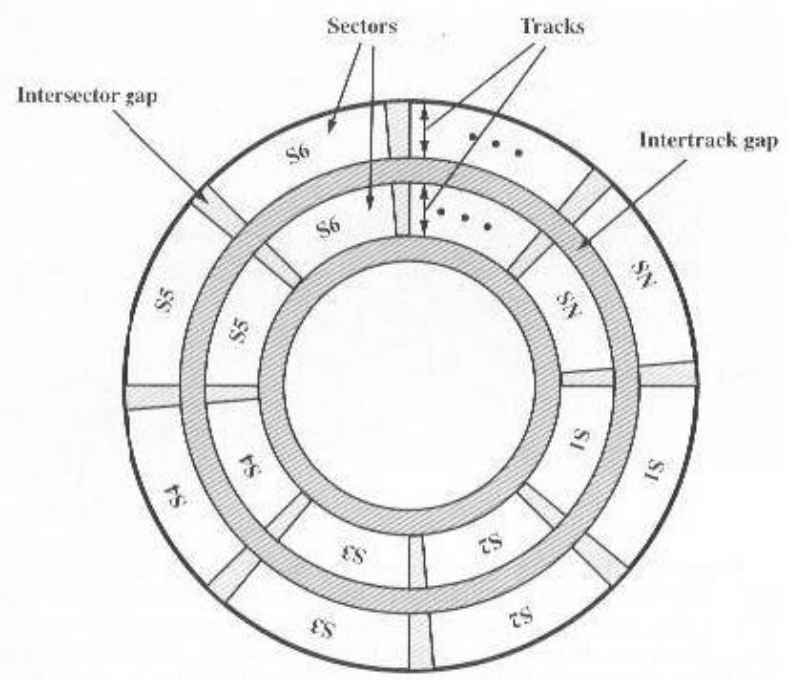
Magnetic Disk

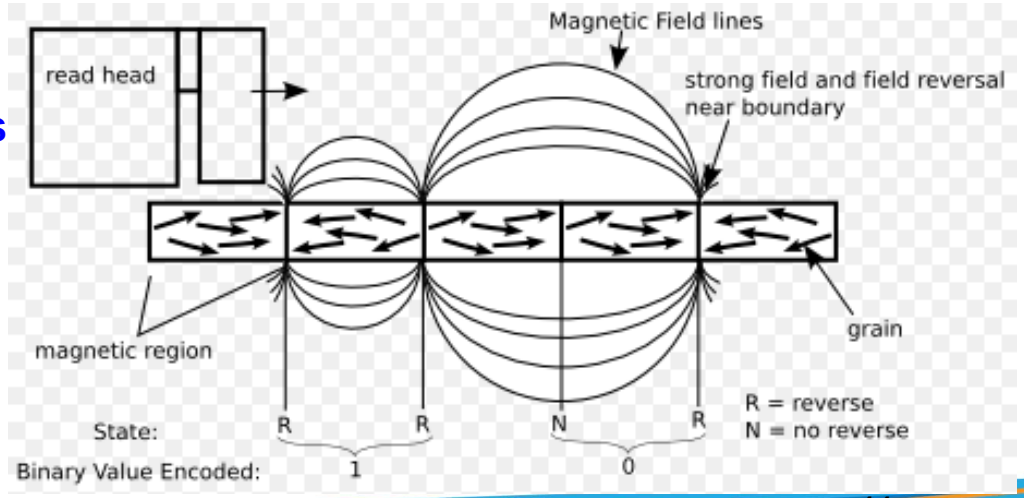
- Still be the most common permanent data storage device, consisting of one or more layers of flat magnetically coated disks to store data
- 2 types
  - Floppy disks – slow, only 1 disk layer
  - Hard Disk Drives - HDD – faster, more layers of disks
- HDD



HDD Organization

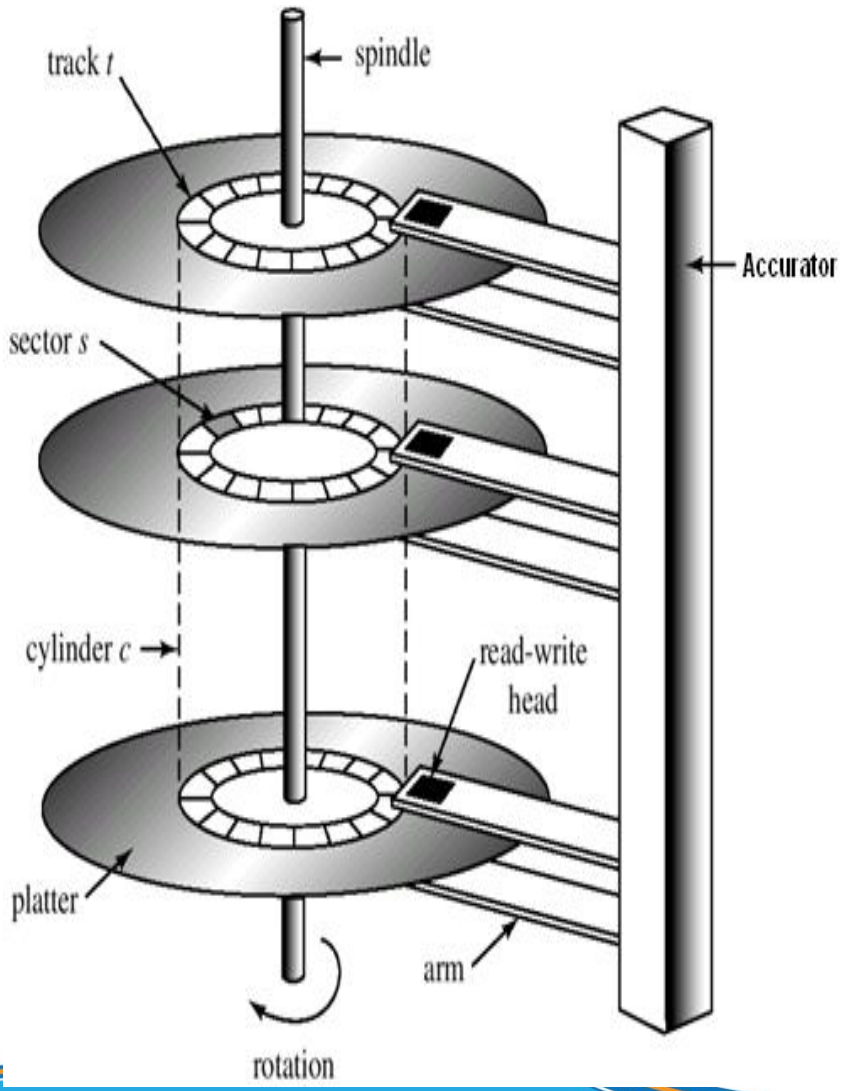






- Consists of many layers of platters, magnetically coated in 1 or both **sides**
- Each side has a corresponding **head** to read or write data
- Each side has many concentric circles (**tracks**)
- Aligned tracks on each platter form **cylinders**
- Each track is subdivided into **sectors**, typically each sector contains 4096 magnetically region (~4096 bits = 512 bytes)
- Sector is the unit of read/write operation

HDD Access Mechanism



- Disk Latency = Seek Time + Rotation Time + Transfer Time
  - Seek Time – Time to move the head to the correct track for reading/writing, depending on the number of tracks on one side and the speed of the actuator
    - Average < 10 ms
  - Rotation Time – The time to rotate the disk so that the sector to be read/written is below the head, depending on the rotation speed of the disk
    - 7200 rpm (Revolutions Per Minute) → 120 Rev/sec
    - 1 revolution = 1/120 sec ~ 8.33 milliseconds
    - Average (1/2 revolution) = 4.17 ms
  - Transfer Time – The time to read/write and transfer data, depending on the magnetic coverage density of the sector and the communication standard (ATA, SATA, SCSI, ...)

Magnetic Disk Capacity

- Floppy disks
  - 1 platter, 2 heads, 80 tracks/head, 18 sectors/track.
  - So capacity will be:

    2 head/disk * 80 track/head * 18 sector/track
    = 2880 sector/disk

    = 0.5 KB/sector * 2880 sector/disk

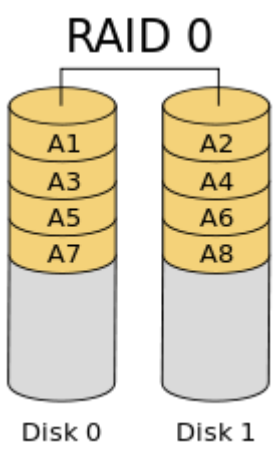
    = 1440 KB/disk (~ 1.44 MB)
- HDD
  - 16 heads, 684 cylinders, 18 sectors/track
  - So the capacity will be ?

Mass storage device: RAID

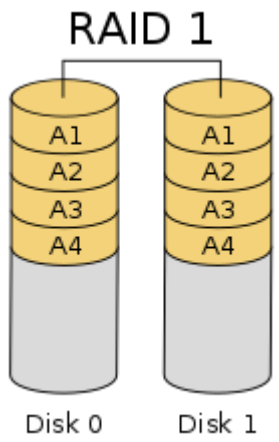
- RAID: Redundant Array of Inexpensive Disks
  - Combine multiple (physical) drives into a single (logical) disk system using hardware (RAID controller) or software
  - Data stored in distributed physical disk
  - Transparent to the user
- Purpose:
  - Ensure data redundancy
  - Performance improvement
- RAID types: 0, 1, 0+1, 1+0, 2, 3, 4, 5, 5+0, 6…

Some common RAID types

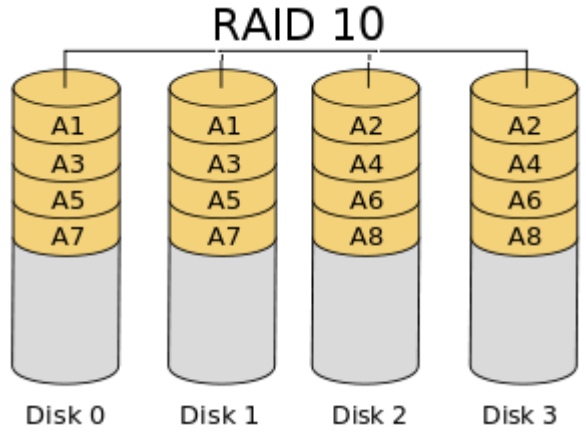
Stripping,
no redundancy



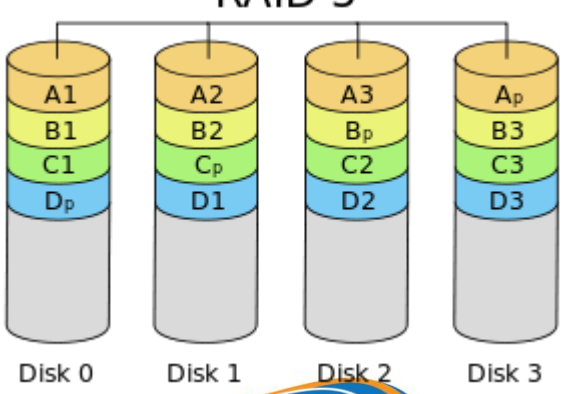
Mirror Data,
most expensive solution



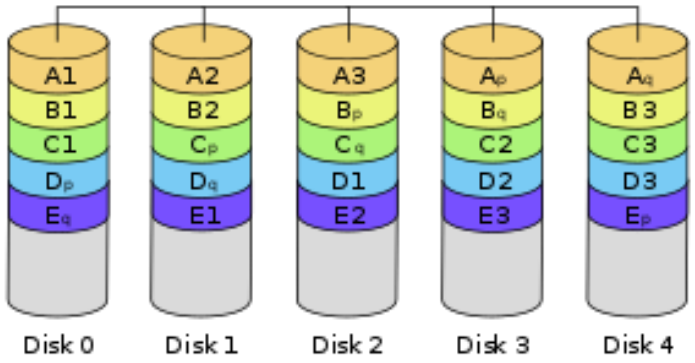
Combining mirroring and striping



Striping with parity



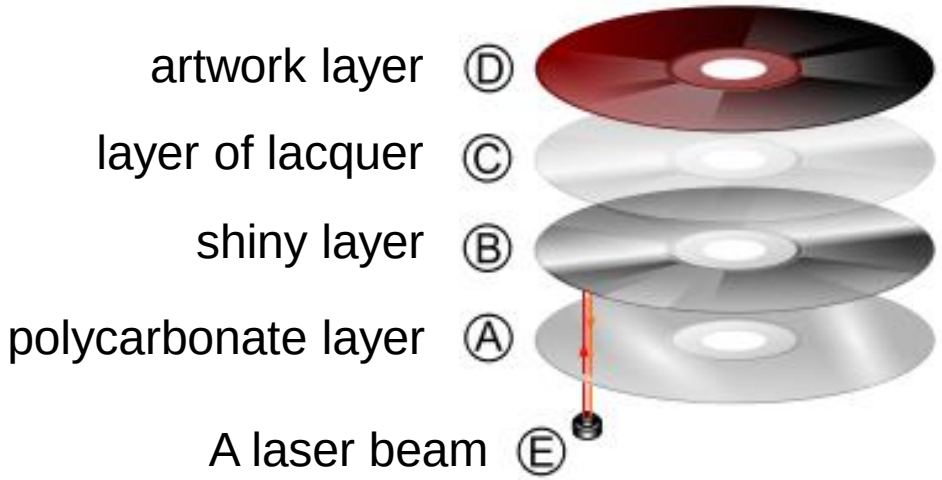
Striping with double parity



7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110

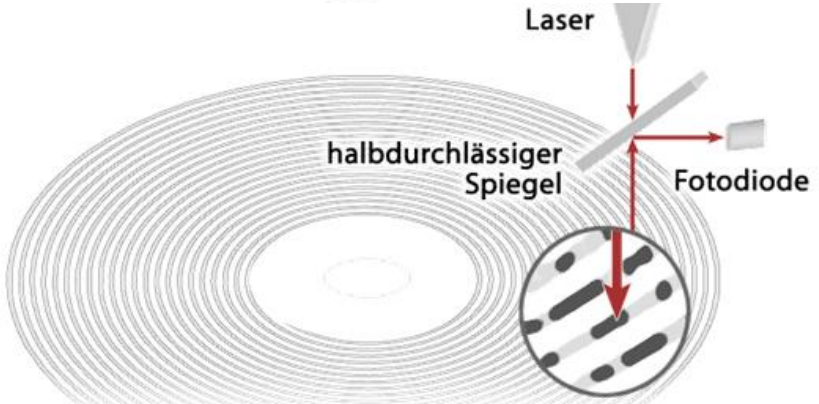
Optical Disc (1/3)

Compact Disc (CD)



52x / 32x / 52x : speed
for CD-R / CD-RW / CD

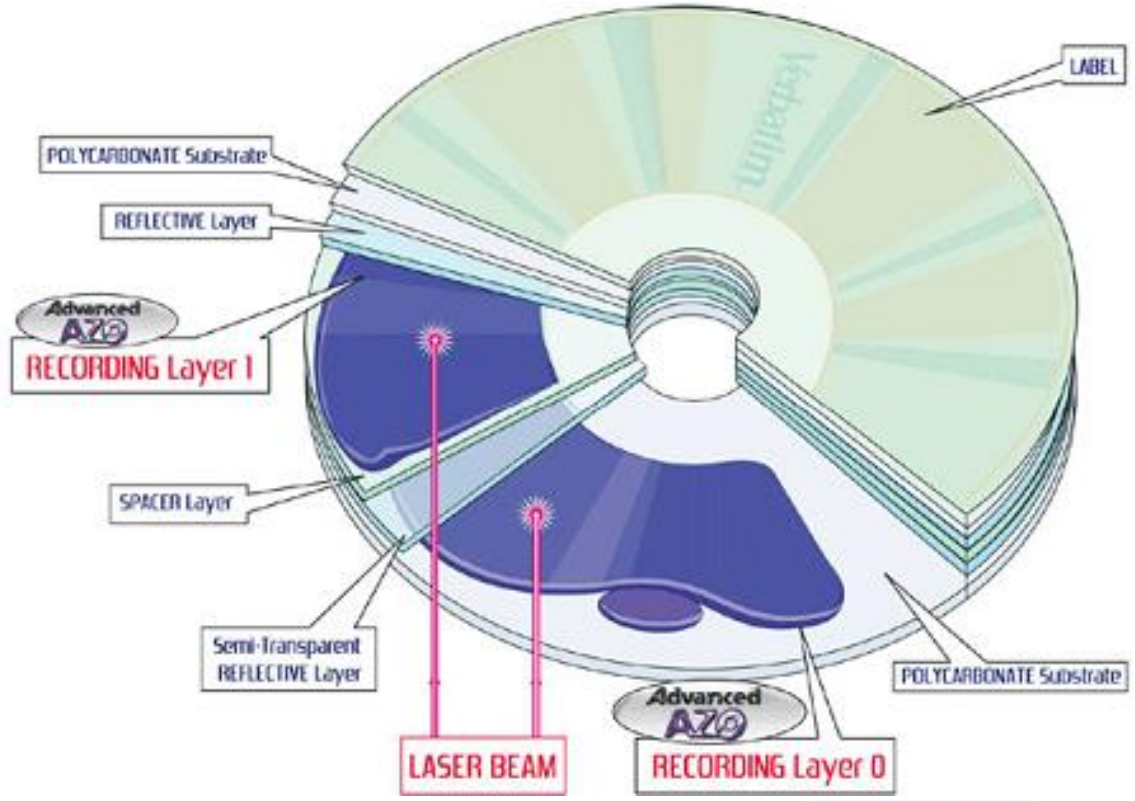
Transfer Speed	Megabytes/s	Megabits/s
1x	0.15	1.2
2x	0.3	2.4
4x	0.6	4.8
8x	1.2	9.6
10x	1.5	12
12x	1.8	14.4
20x	3	24
32x	4.8	38.4
36x	5.4	43.2
40x	6	48
48x	7.2	57.6
50x	7.5	60
52x	7.8	62.4



Type	Sectors	Data max size (MB)	Audio max size (MB)
8 cm	94,500	193.536	222.264
650 MB	333,000	681.984	783.216
700 MB	360,000	737.28	846.72
800 MB	405,000	829.44	952.56
900 MB	445,500	912.384	1,047.82

Optical Disc (2/3)

- Digital Versatile Disc (DVD)



Drive speed	Data rate		
	(Mbit/s)	(MB/s)	(MiB/s)
1×	10.80	1.35	1.29
2×	21.60	2.70	2.57
2.4×	25.92	3.24	3.09
2.6×	28.08	3.51	3.35
4×	43.20	5.40	5.15
6×	64.80	8.10	7.72
8×	86.40	10.80	10.30
10×	108.00	13.50	12.87
12×	129.60	16.20	15.45
16×	172.80	21.60	20.60
18×	194.40	24.30	23.17
20×	216.00	27.00	25.75
22×	237.60	29.70	28.32
24×	259.20	32.40	30.90

	Single layer (GB)	Double layer (GB)
12 cm, single sided	4.7	8.5
12 cm, double sided	9.4	17.1
8 cm, single sided	1.4	2.6
8 cm, double sided	2.8	5.2

Optical Disc (3/3)

- HD DVD & Blue-ray Disc

Blu-ray Disc

HD DVD

HD DVD

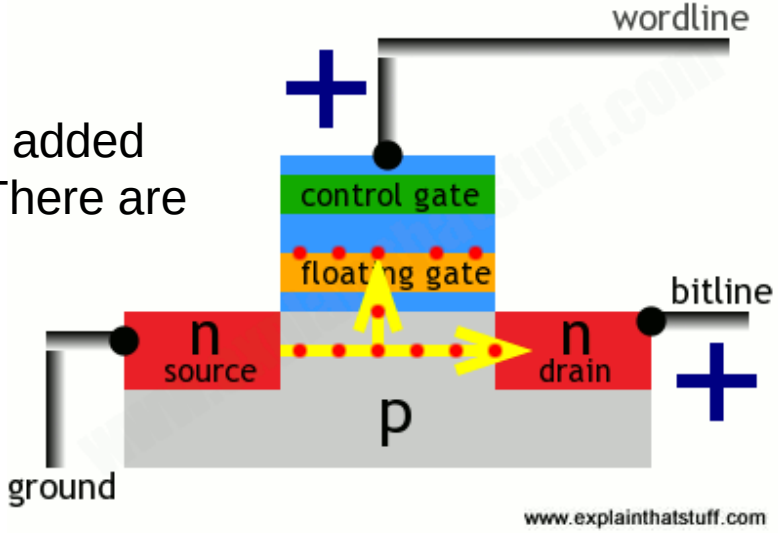
	Single layer	Dual layer
12 cm, single sided	15 GB	30 GB
12 cm, double sided	30 GB	60 GB
8 cm, single sided	4.7 GB	9.4 GB
8 cm, double sided	9.4 GB	18.8 GB

BD

	Single layer	Dual layer
12 cm, single sided	25 GB	50 GB
8 cm, single sided	7.8 GB	15.6 GB

Flash memory (1/5)

- A type of storage technology that most commonly used today
  - USB, Memory card, ROM, SSD …
- No power required to maintain data (non-volatile)
  - A data bit is created from the NMOS transistor, in which a conductor added between G(gate) and S(source)/D(drain) to maintain the electrons. There are electrons corresponding to bit 1, no electrons corresponding to bit 0
- Pros
  - Fast
  - Durability
  - Less power consumption
- Cons
  - Cost
  - Limited number of data write/erase cycles (due to wear and tear)
    - Most are over 100k, some over 1 million data write/erase cycles



Flash memory (2/5)

- USB (Universal Serial Bus)






Flash memory (3/5)

- Memory card

Name	Acronym	Form factor
PC Card	PCMCIA	85.6 × 54 × 3.3 mm
CompactFlash I	CF-I	43 × 36 × 3.3 mm
CompactFlash II	CF-II	43 × 36 × 5.5 mm
SmartMedia	SM / SMC	45 × 37 × 0.76 mm
Memory Stick	MS	50.0 × 21.5 × 2.8 mm
Memory Stick Duo	MSD	31.0 × 20.0 × 1.6 mm
Memory Stick Micro M2	M2	15.0 × 12.5 × 1.2 mm
Multimedia Card	MMC	32 × 24 × 1.5 mm
Reduced Size Multimedia Card	RS-MMC	16 × 24 × 1.5 mm
MMCmicro Card	MMCmicro	12 × 14 × 1.1 mm
Secure Digital Card	SD	32 × 24 × 2.1 mm
miniSD Card	miniSD	21.5 × 20 × 1.4 mm
microSD Card	microSD	11 × 15 × 1 mm
xD-Picture Card	xD	20 × 25 × 1.7 mm
Intelligent Stick	iStick	24 x 18 x 2.8 mm
µ card	µcard	32 x 24 x 1 mm

Card reader

CompactFlash, Memory Stick, Secure Digital, and xD

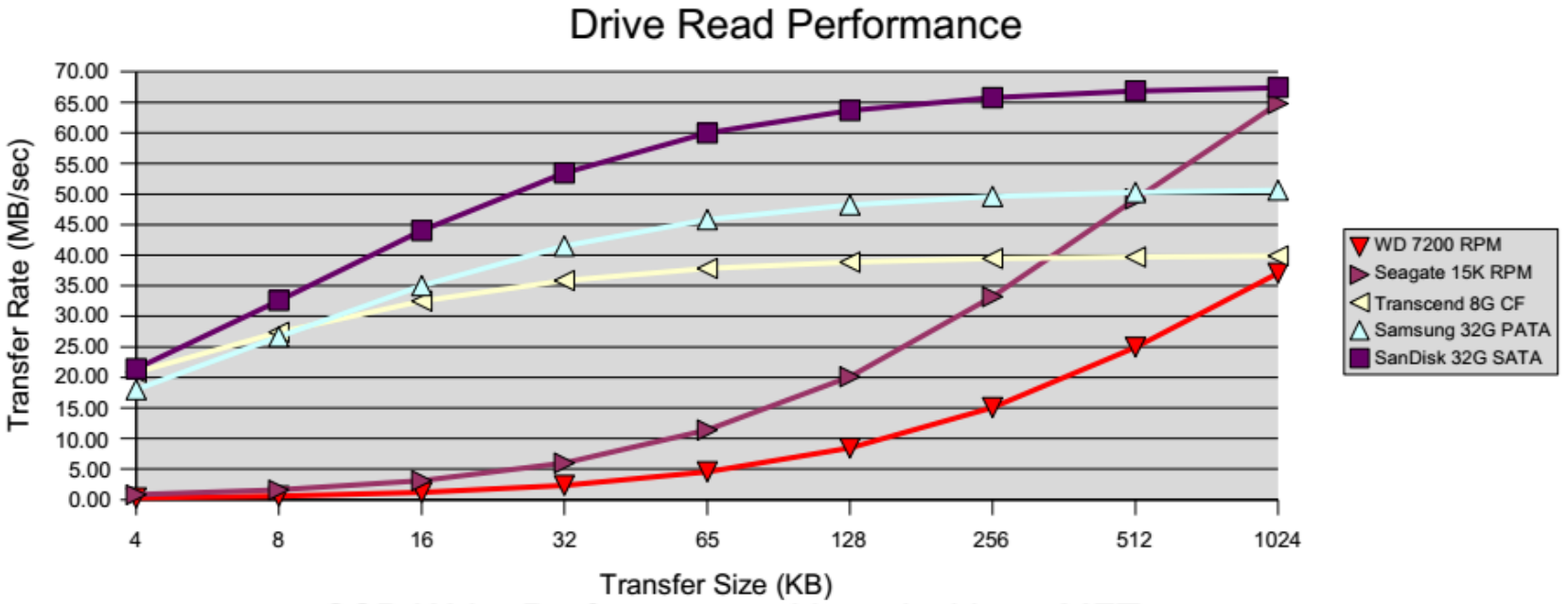
Flash memory (4/5)

- Solid State Drive (SSD)

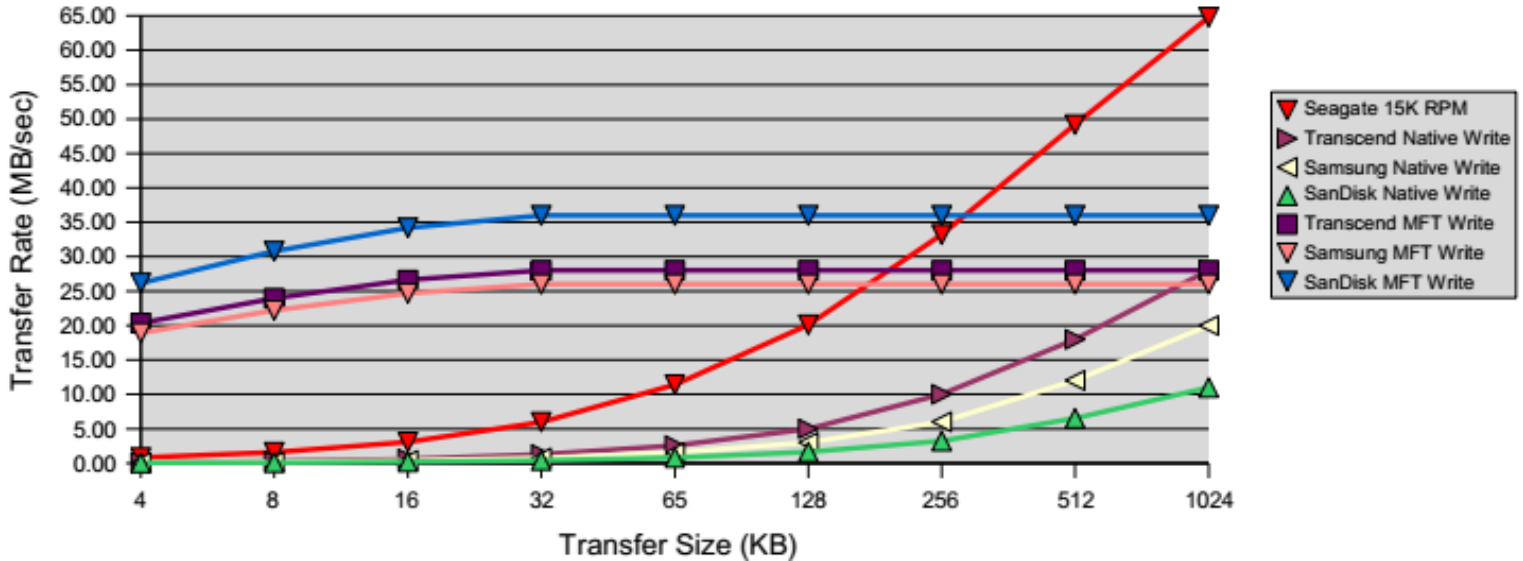
	Drive Model	Description	Seek Time			Latency	Read XFR Rate		Write XFR Rate	
			Track to Track	Average	Full Stroke		Outer Tracks	Inner Tracks	Outer Tracks	Inner Tracks
Hard Drives	Western Digital WD7500AYYS	7200 RPM 3.5" SATA	0.6 ms	8.9 ms	12.0 ms	4.2 ms	85 MB/sec	60 MB/sec*	85 MB/sec	60 MB/sec*
	Seagate ST936751SS	15K RPM 2.5" SAS	0.2 ms	2.9 ms	5.0 ms*	2.0 ms	112 MB/sec	79 MB/sec	112 MB/sec	79 MB/sec
Flash SSDs	Transcend TS8GCF266	8GB 266x CF Card	0.09ms				40 MB/sec		32 MB/sec	
	Samsung MCAQE32G5APP	32G 2.5" PATA	0.14ms				51 MB/sec		28 MB/sec	
	Sandisk SATA5000	32G 2.5" SATA	0.125ms				68 MB/sec		40 MB/sec	

Flash memory (5/5)

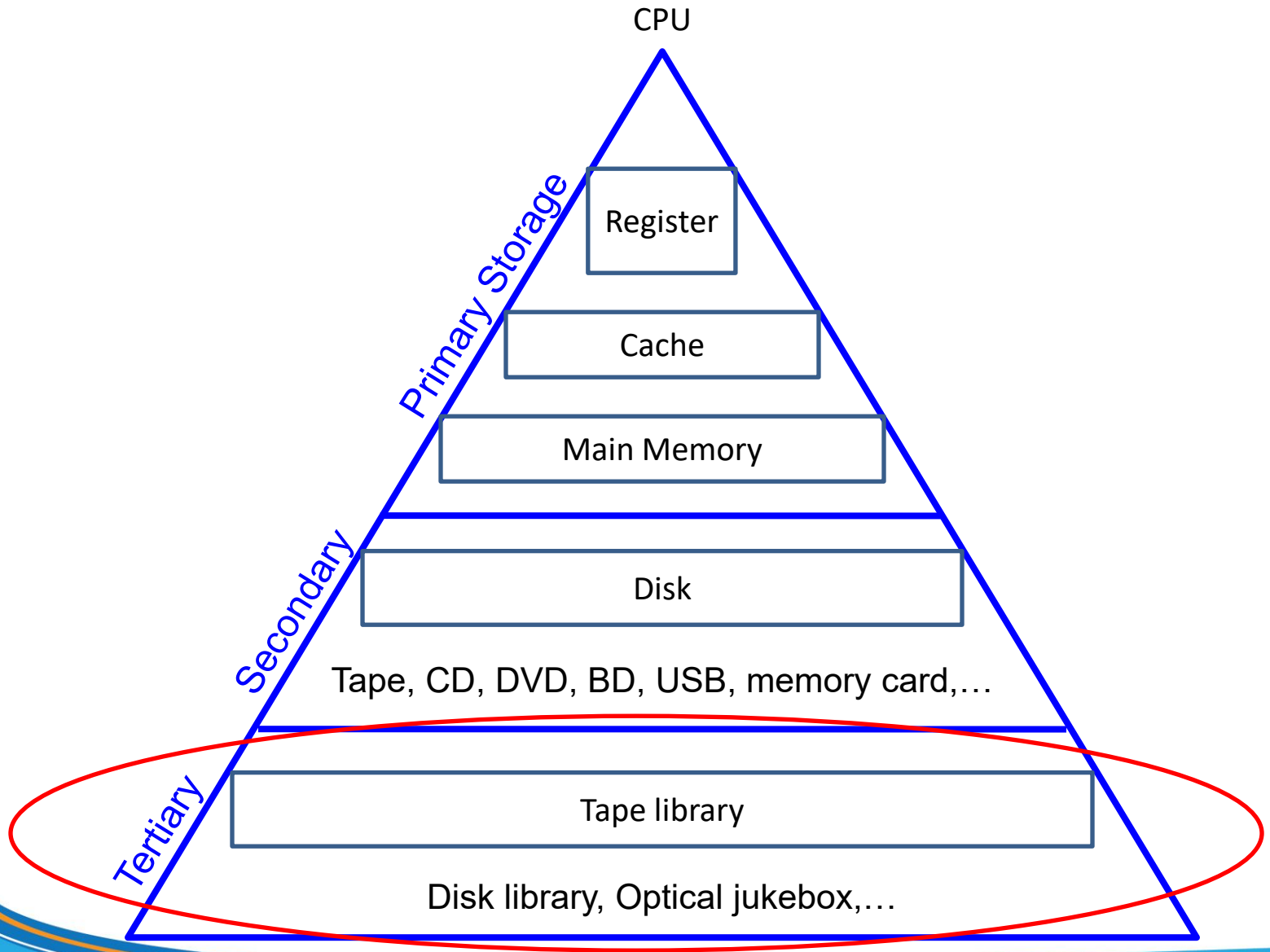
- SSD



SSD Write Performance with and without MFT (Managed Flash Technology)



Tertiary Storage

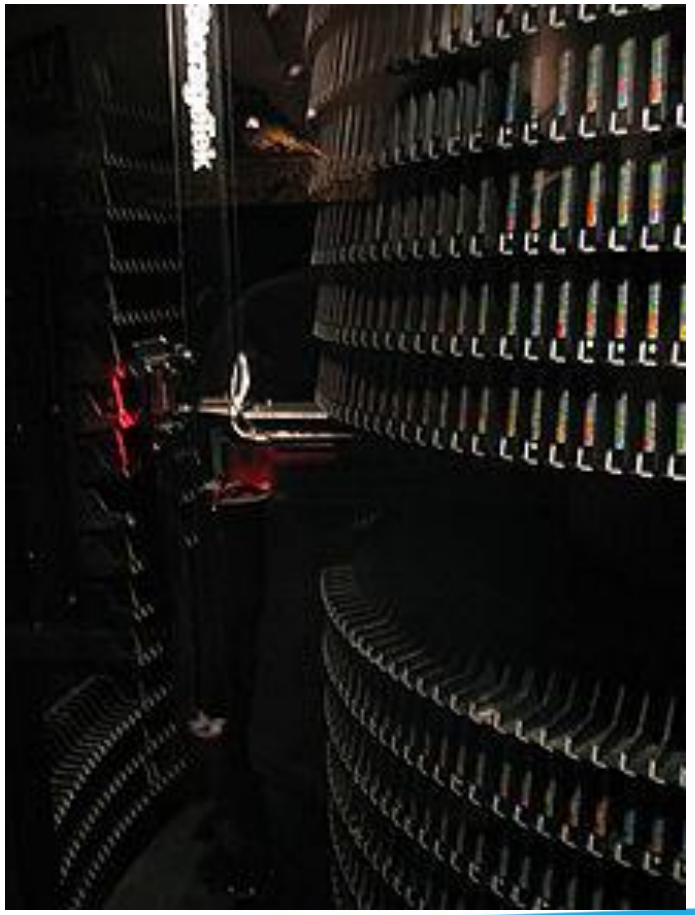


Tape library

- The device allows thousands of tapes to be combined to form a storage device with capacities up to Terabytes, Petabytes

HP StorageWorks Ultrium 960

Recording Technology	LTO Ultrium 3
Compressed Capacity *	800 GB
Sustained Transfer Rate (compressed) *	576 GB/hr
Buffer Size	128 MB
Interface	Ultra320 SCSI (LVDS)
Form Factor	5.25 inch half-height
WORM Capability	Yes



Disk library

- The device allows to combine thousands of hard disks to form a storage device with capacities up to Terabyte, Petabyte

Optical jukebox

- The device allows to combine thousands of optical discs (CD, DVD, Blue-ray disk) to form a storage device with a capacity of up to Terabyte, Petabyte