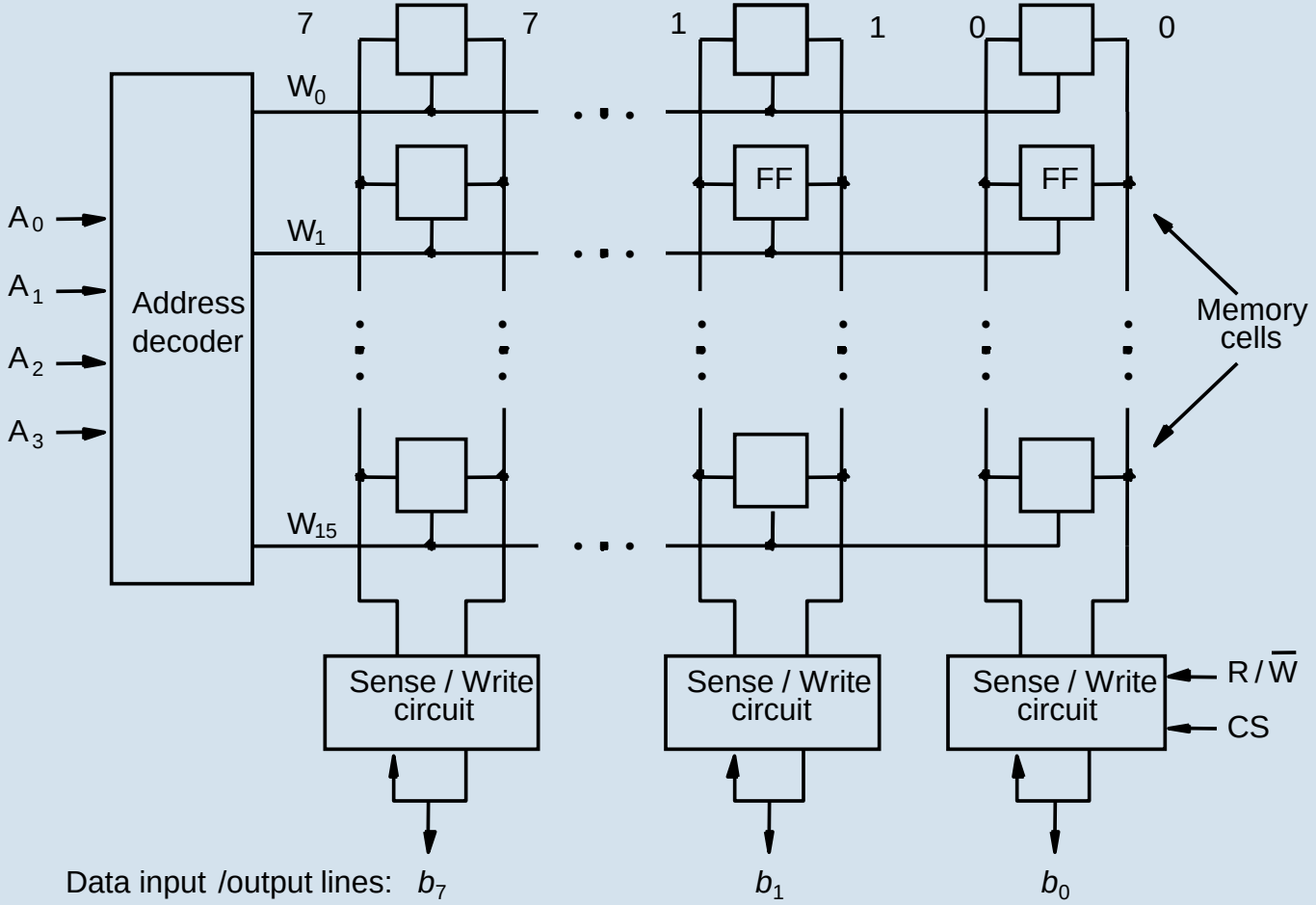
MODULE 2: MEMORY SYSTEM ORGANIZATION & ARCHITECTURE

Dr.Maheswari.R

Ideal Memory Characteristics

- CPU should have rapid, **uninterrupted** access to external memories.
- **Memory speed should match CPU speed.**
- Unfortunately, such high speed memories are very expensive.
- So the general approach is to distribute information over various memory types that have different performance and cost.

Internal organization of memory chips



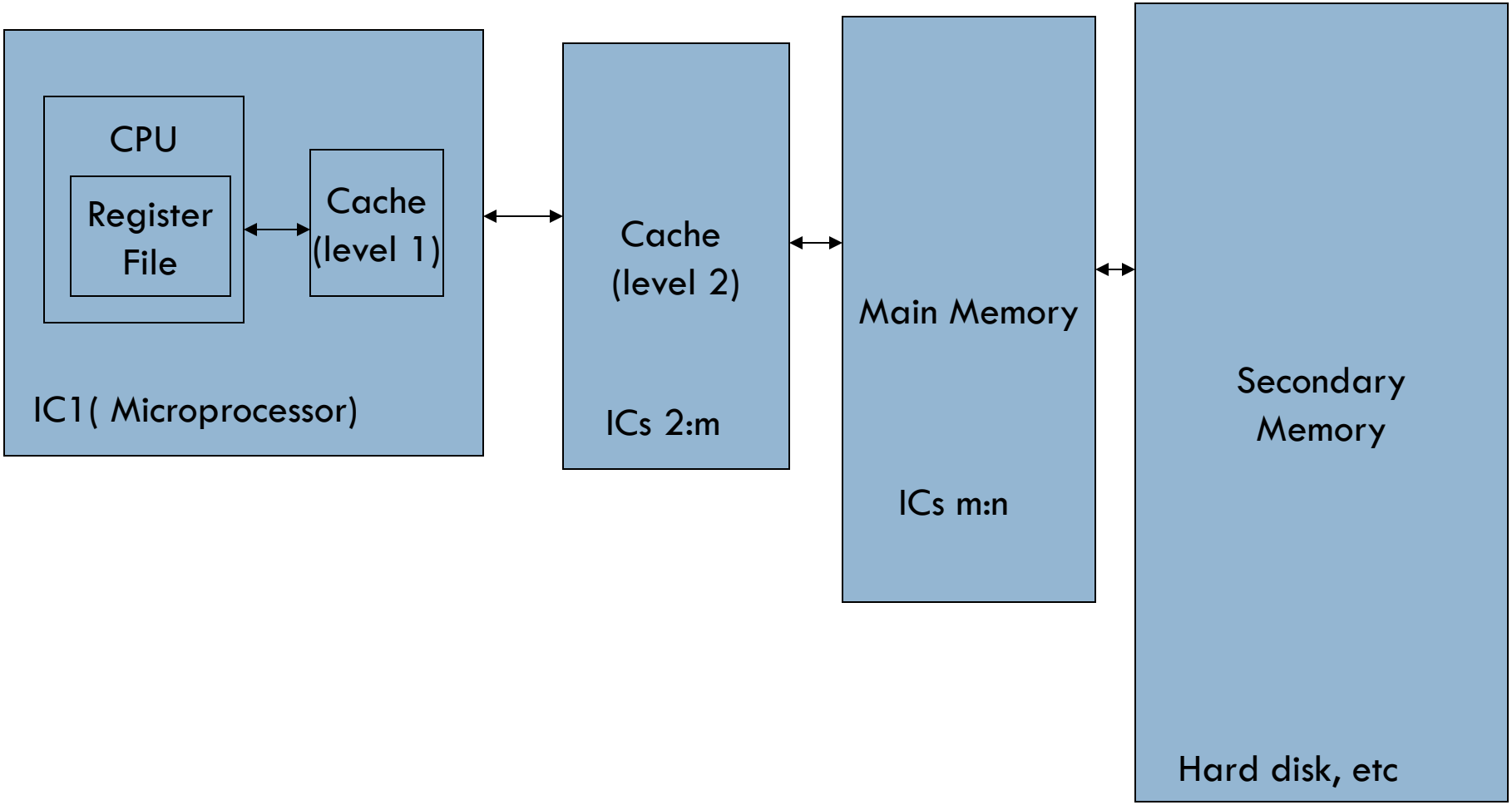
Principles of Locality

- **SPATIAL LOCALITY**
  - The locality principle stating that if a data location is referenced, data locations with **nearby addresses** will tend to be referenced soon

- **TEMPORAL LOCALITY**
  - The principle stating that if a data location is referenced then it will tend to be **referenced again soon**.

MEMORY HIERARCHY

Conceptual Organisation of Multilevel memory



Memory Types

- **CPU registers**
  - High speed -- Costly
  - Working memory for temporary storage of instruction and data.
  - Usually General Purpose Registers are used.
  - Size : comparatively small like 32 data word.
  - Within a clock cycle they can be accessed.
- **Main(primary) Memory**
  - Fairly fast external memory
  - Storage addressed by Load and Store instructions
  - Generally sizes 1MegaByte. Now a days we have GB.
  - 1 MB= 2^{20} bytes and 1 GB= 2^{10} MB.
  - Access Time five or more clock cycles.

- 4 bit of address -> 2^4
- 10 bit of address → 2^10 ->
- 0000
- 0001
- ..
- 1111

Continued…

- Secondary Memory
  - Larger capacity – Many gigabytes
  - Slower
  - Acts as overflow memory when capacity of main memory exceeded.
  - Accessed via I/O programs
  - Eg: Hard disks, CD-ROM's etc.
- Cache
  - Positioned logically between Registers and Main memory.
  - Capacity less than main memory and greater than registers.
  - Speed vice versa.
  - External Memory → Together "Main Memory and Cache memory"

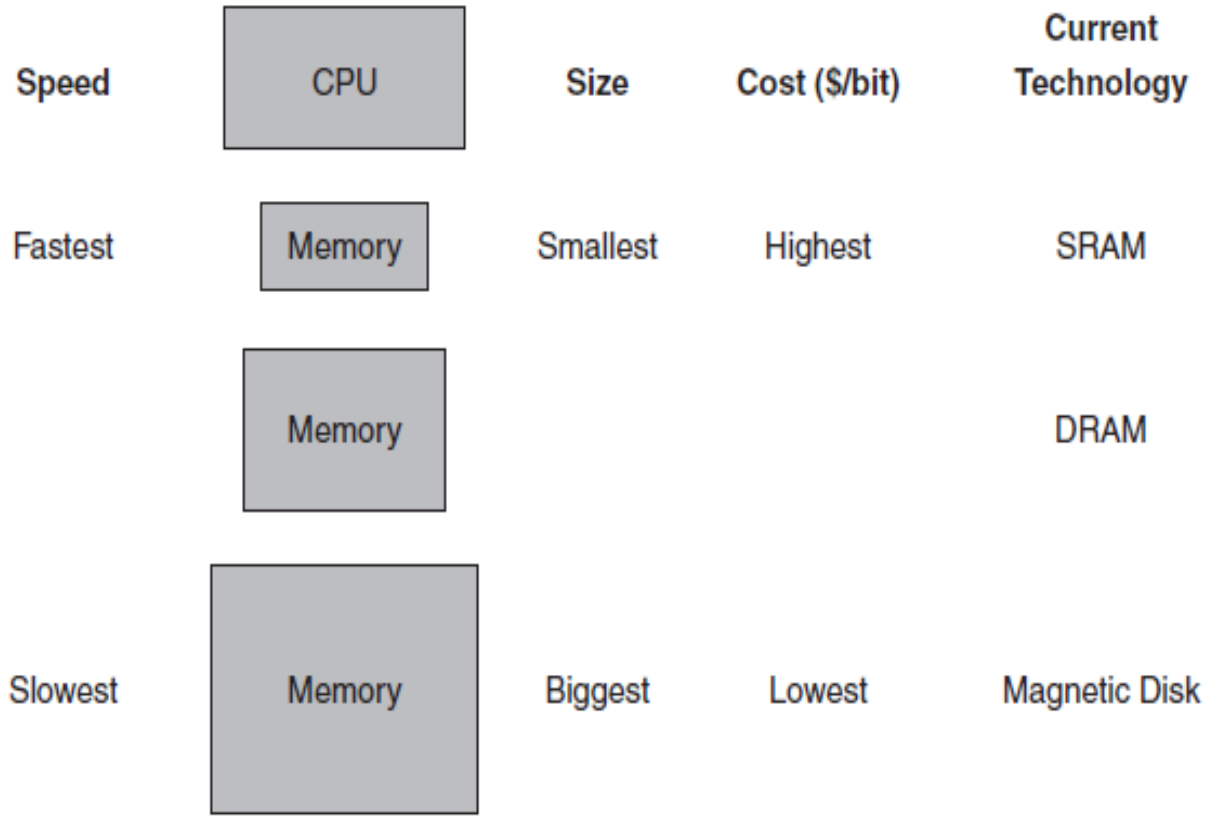
Continued…

- Random Access Memory
- Sequential Access Memory

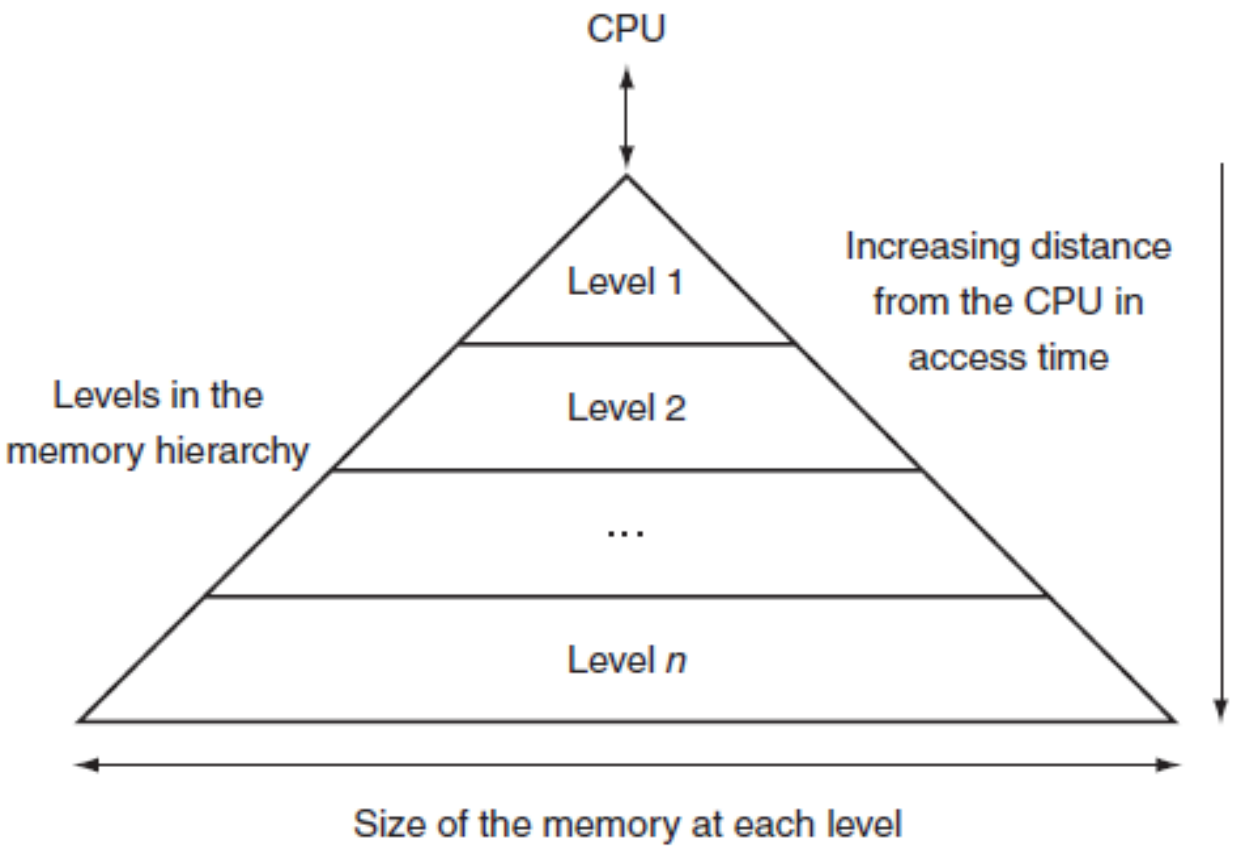
Understanding RAM and ROM (Video Available in Youtube)

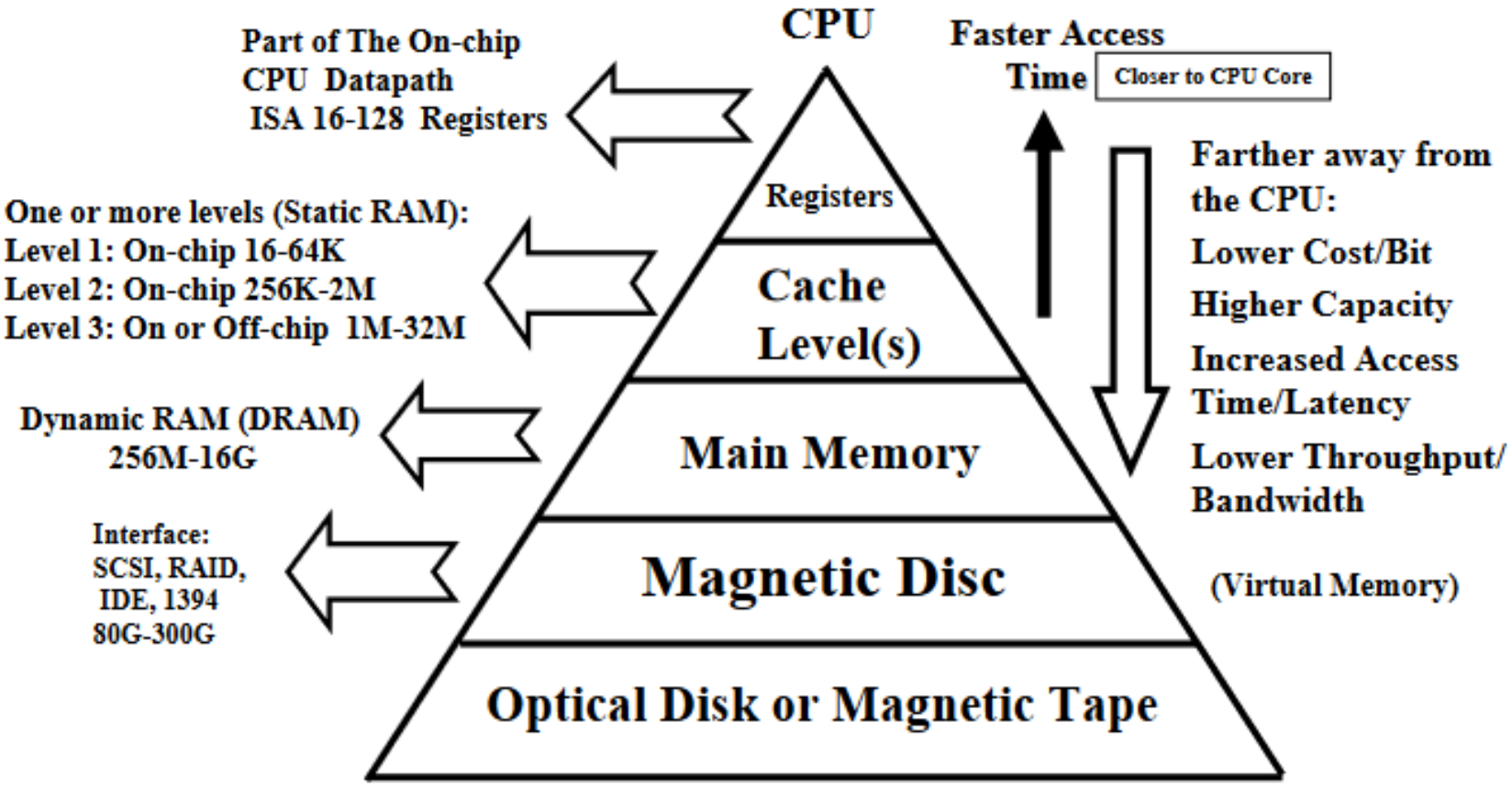
- Or refer to the link
  - https://www.youtube.com/watch?v=ufGRoLOvM9I
  - Ram
  - https://www.youtube.com/watch?v=PVad0c2cljo

Basic Structure of Memory hierarchy



Continued…


CPU
Level 1
Level 2
…
Level n
Increasing distance from the CPU in access time
Levels in the memory hierarchy
Size of the memory at each level


CPU
Part of The On-chip
CPU Datapath
ISA 16-128 Registers
Faster Access
Time
Closer to CPU Core
Farther away from
the CPU:
Lower Cost/Bit
Higher Capacity
Increased Access
Time/Latency
Lower Throughput/
Bandwidth
Registers
One or more levels (Static RAM):
Level 1: On-chip 16-64K
Level 2: On-chip 256K-2M
Level 3: On or Off-chip 1M-32M
Cache
Level(s)
Dynamic RAM (DRAM)
256M-16G
Main Memory
Interface:
SCSI, RAID,
IDE, 1394
80G-300G
Magnetic Disc
(Virtual Memory)
Optical Disk or Magnetic Tape

What is SRAM?

- SRAM is a type of RAM that holds data in a static form, that is, as long as the memory has power
- SRAM stores a bit of data on four transistors using two cross-coupled inverters
- SRAM is best suited for secondary operations like the CPU's fast cache memory and storing registers
- SRAM is most often found in hard drives as disc cache
- SRAM is about 10 nanoseconds
- SRAM's cycle time is shorter than DRAM's because it does not need to refresh. The cycle time of SRAM is shorter because it does not need to stop between accesses to refresh

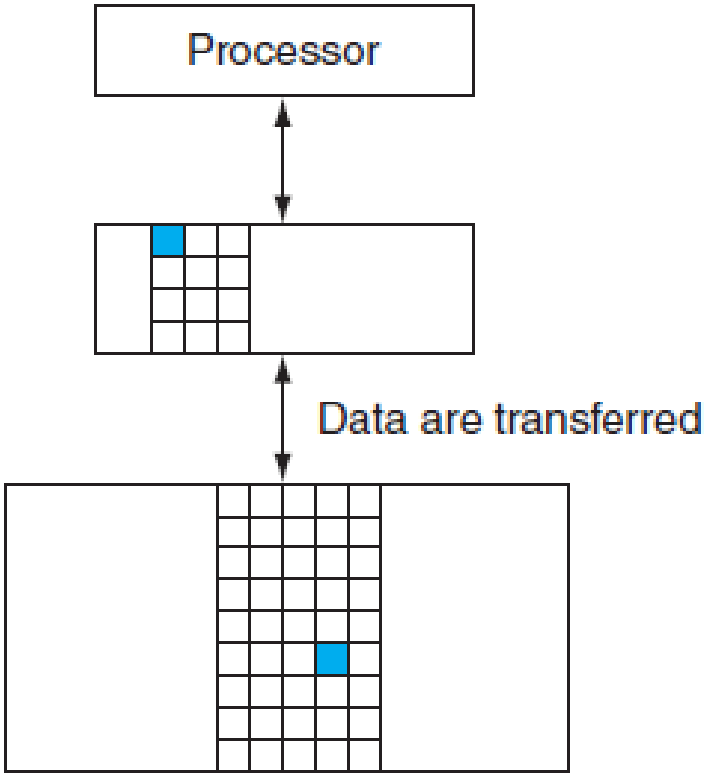
What is DRAM?

- Dynamic random-access memory (DRAM) is a type of random-access memory that stores each bit of data in a separate capacitor within an integrated circuit.
- The capacitor can be either charged or discharged; these two states are taken to represent the two values of a bit, conventionally called 0 and 1.
- The capacitors will slowly discharge, and the information eventually fades unless the capacitor charge is refreshed periodically
- The main memory (the "RAM") in personal computers is dynamic RAM (DRAM)

Memory Hierarchy and the components used

- Register Files → fast static RAMs with multiple ports. Such RAMs are distinguished by having dedicated read and write ports, whereas ordinary multi-ported SRAMs will usually read and write through the same ports
- Cache → SRAM with single read/write port
- Main Memory → DRAM
- Secondary memory → Hard disk and CD's

Upper and Lower level in Memory Hierarchy



Continued…

- **Block** : The minimum unit of information that can be either present or not present

- **Hit** : If the data requested by the processor appears in some block in the upper level
  - L1 Cache hit, L2 cache, Memory hit etc

- Miss: If the data is **not found** in the upper level
  - L1 Cache miss, L2 cache miss, Memory miss etc
  - The lower level in the hierarchy is then accessed to retrieve the block containing the requested data.

Continued…

- **Hit rate** The fraction of memory accesses found in a level of memory hierarchy

- **Miss Rate :** The fraction of memory accesses not found in a level of the memory hierarchy.

- **Hit Time :**The time required to access a level of the memory hierarchy, including the time needed to determine whether the access is a hit or a miss

- **Miss penalty :** The time required to fetch a block into a level of the memory hierarchy from the lower level, including the time to access the block, transmit it from one level to the other, and insert it in the level that experienced the miss.

- L 1 (2 ms)
- L2 (2 ms)
- M.m ((2 ms)
- S.m (2 ms)

Random Access in RAM

- Or refer to the link
  - https://www.youtube.com/watch?v=Kav6oOFDQSA
  - https://www.youtube.com/watch?v=Kav6oOFDQSA
  - **https://www.youtube.com/watch?v=UaFSsD0LPS8**

Continued…

- Random Access Memory
- Sequential Access Memory
- Semi Random Access memory

Understanding RAM and ROM (Video Available in Youtube)

- Refer to the video uploaded Memory Concepts 1

- Or refer to the link
  - https://www.youtube.com/watch?v=p3q5zWCw8J4
  - https://www.youtube.com/watch?v=sK-49uz3lGg

Random Access in RAM

- Refer to the video uploaded Memory Concepts 2

- Or refer to the link
  - https://www.youtube.com/watch?v=Kav6oOFDQSA

Addressing and Data(read/write) in RAM

- refer to the link
  - **https://www.youtube.com/watch?v=UaFSsD0LPS8**

Basics of Cache

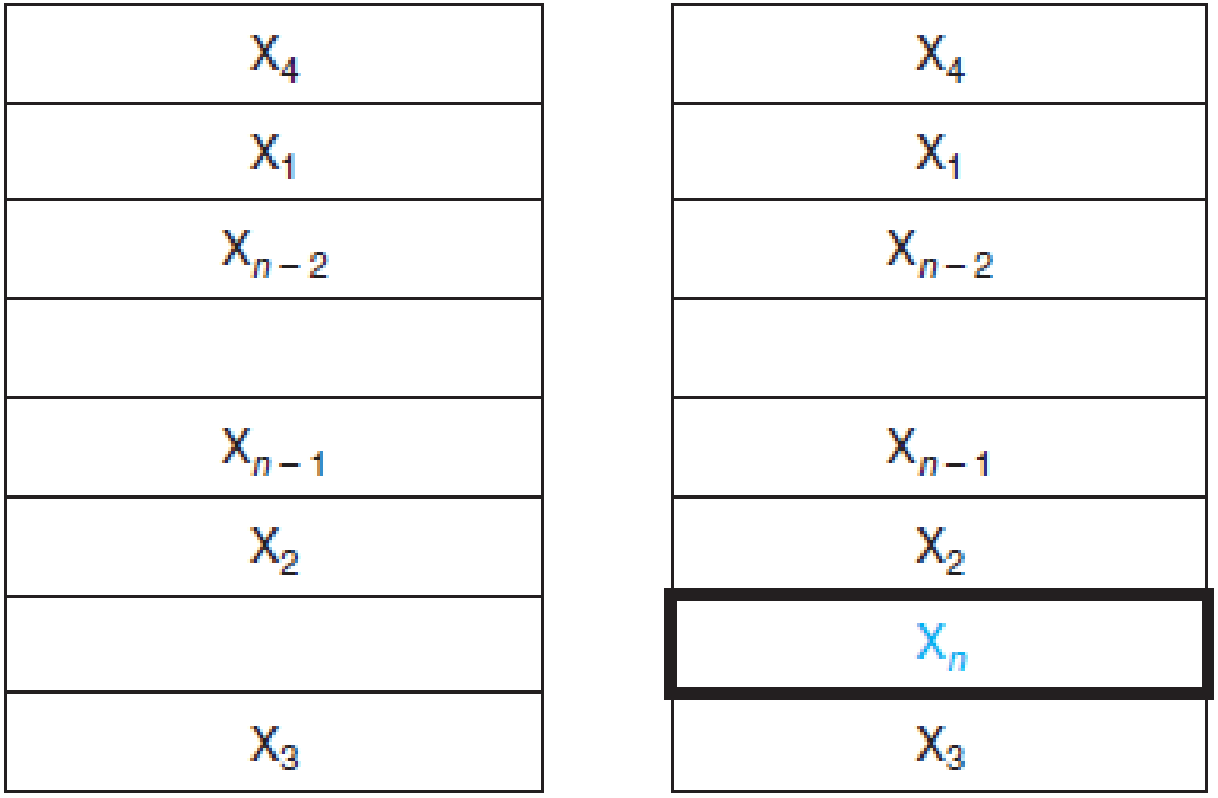
- Cache was the name chosen to represent the level of the memory hierarchy between the processor and main memory
- The term is also used to refer to any storage managed to take advantage of locality of access.
- Let's consider a cache with blocks of one word

Cache Memory

- Cache :is a small high-speed memory. Stores
- data from some frequently used addresses of main memory
- **Cache hit** :Data found in cache. Results in data transfer at maximum speed.
- **Cache miss:** Data not found in cache. Processor loads data from M and copies into cache. This results in extra delay, called miss penalty.

- Hit ratio = percentage of memory accesses satisfied by the cache.
- Miss ratio = 1-hit ratio

Cache before and after reference to word Xn



a. Before the reference to X_n

b. After the reference to X_n

Continued…

- Before the request, X*n* is not in the cache.
- This results in a cache miss
- So Xn is brought in to cache memory from main memory

Mapping between Cache & Main Memory

- Direct Mapped
- Associative Mapped
- Set Associative

Direct Mapping

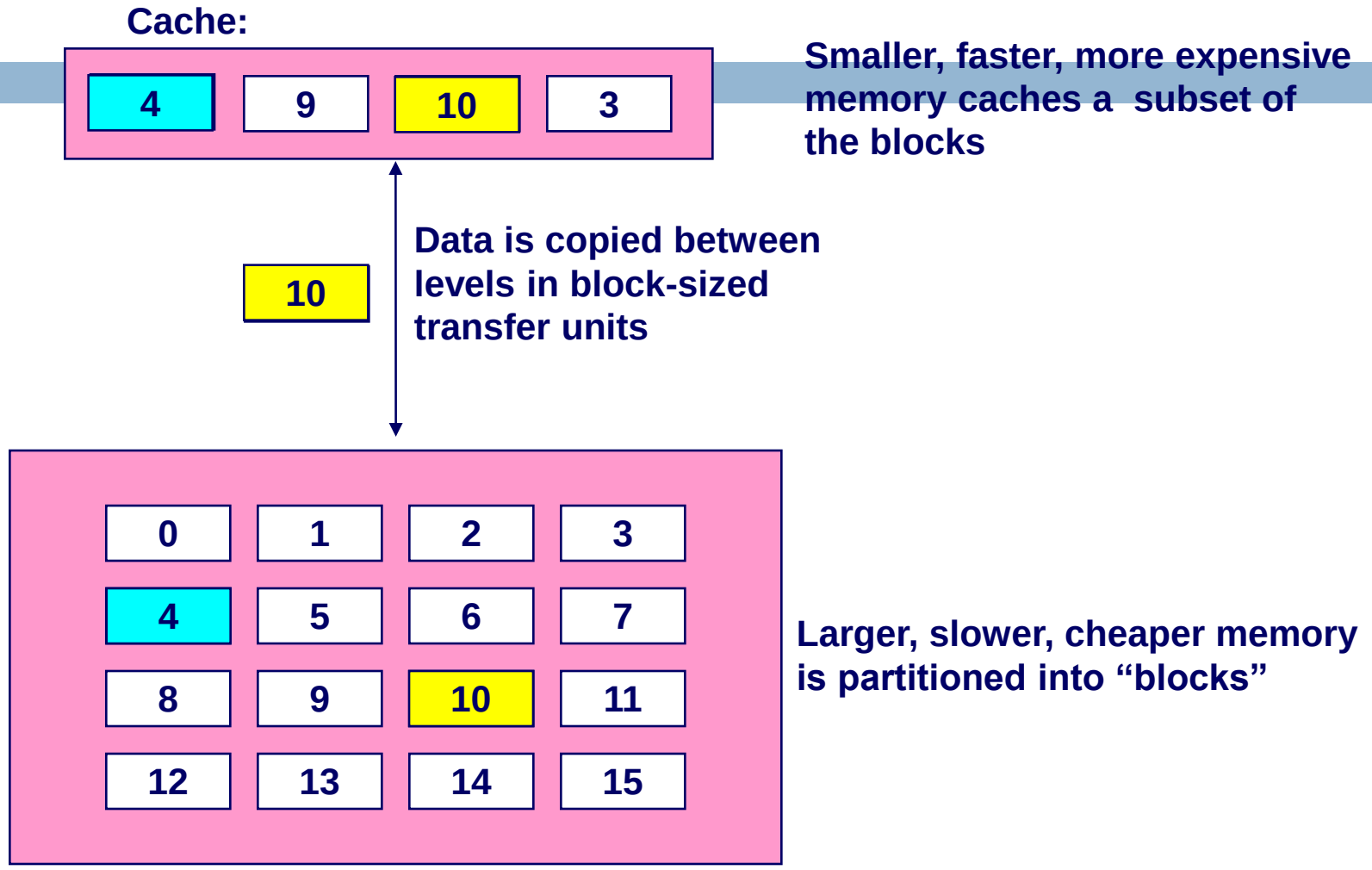
- A cache structure in which each memory location is mapped to exactly one location in the cache.
- Most popular mapping:

(Block address) modulo (Number of cache blocks in the cache)

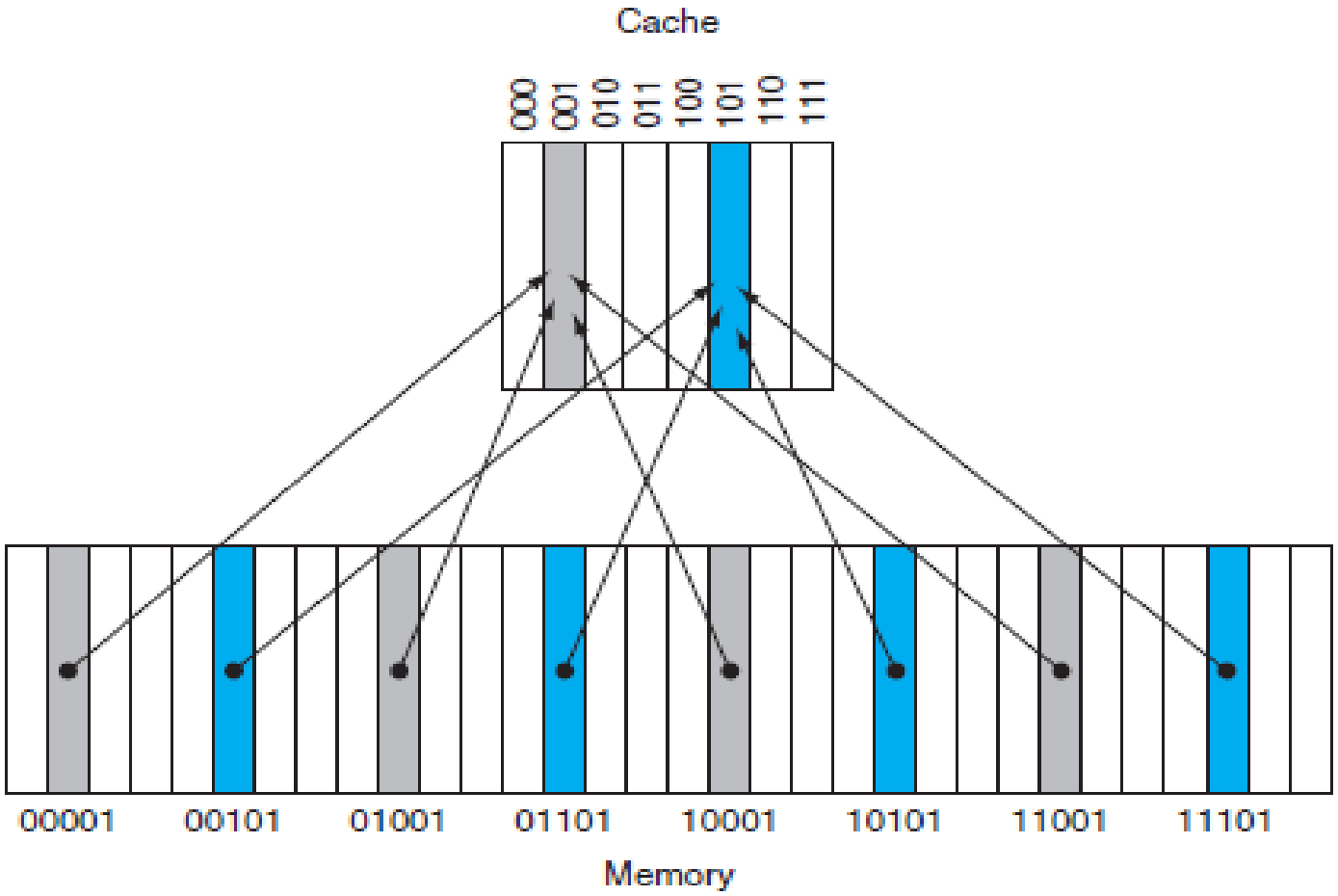
- This mapping is attractive because if the number of entries in the cache is a power of two, then modulo can be computed simply by using the low-order log2 (cache size in blocks) bits of the address

- Hence the cache may be accessed directly with the low-order bits

General cache mechanics



Example


Cache
000
001
010
011
100
101
110
111
00001
00101
01001
01101
10001
10101
11001
11101
Memory

Continued…

- Each cache location can contain the contents of a number of different memory locations
- How do we know whether the data in the cache corresponds to a requested word
  - Tags are added for this purpose
- Tags contain the address information required to identify whether a word in the cache corresponds to the requested word
- Tag is a field in a table to represent the above information

(a part of cache)

Continued….

- In the previous example, upper 2 bits of the 5 bit address is used as tag
- Now we know whether it is the requested block or not.
- But how to know whether the requested block in the cache is valid ? (not corrupt or up to date with respect to a particular program)
  - **Valid bits** are used for this purpose
  - **Field** in the tables of a memory hierarchy that

Example Continued….

1. For the above example, the block references in main memory are as follows (in the same order)

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}

1. Initial state of the cache after powered on

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Continued…

Decimal address of reference	Binary address of reference
22	10110_{two}
26	11010_{two}
22	10110_{two}
26	11010_{two}
16	10000_{two}
3	00011_{two}
16	10000_{two}
18	10010_{two}

2.First reference is for 22, so 22 mod 8➔110 (where the data from main memory goes)

$(22)_{10}$ → 10110_2 So tag =10. Valid bit is set.

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory(10110_{two})
111	N		

Continued…

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memorlly (10110_{two})
111	N		

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

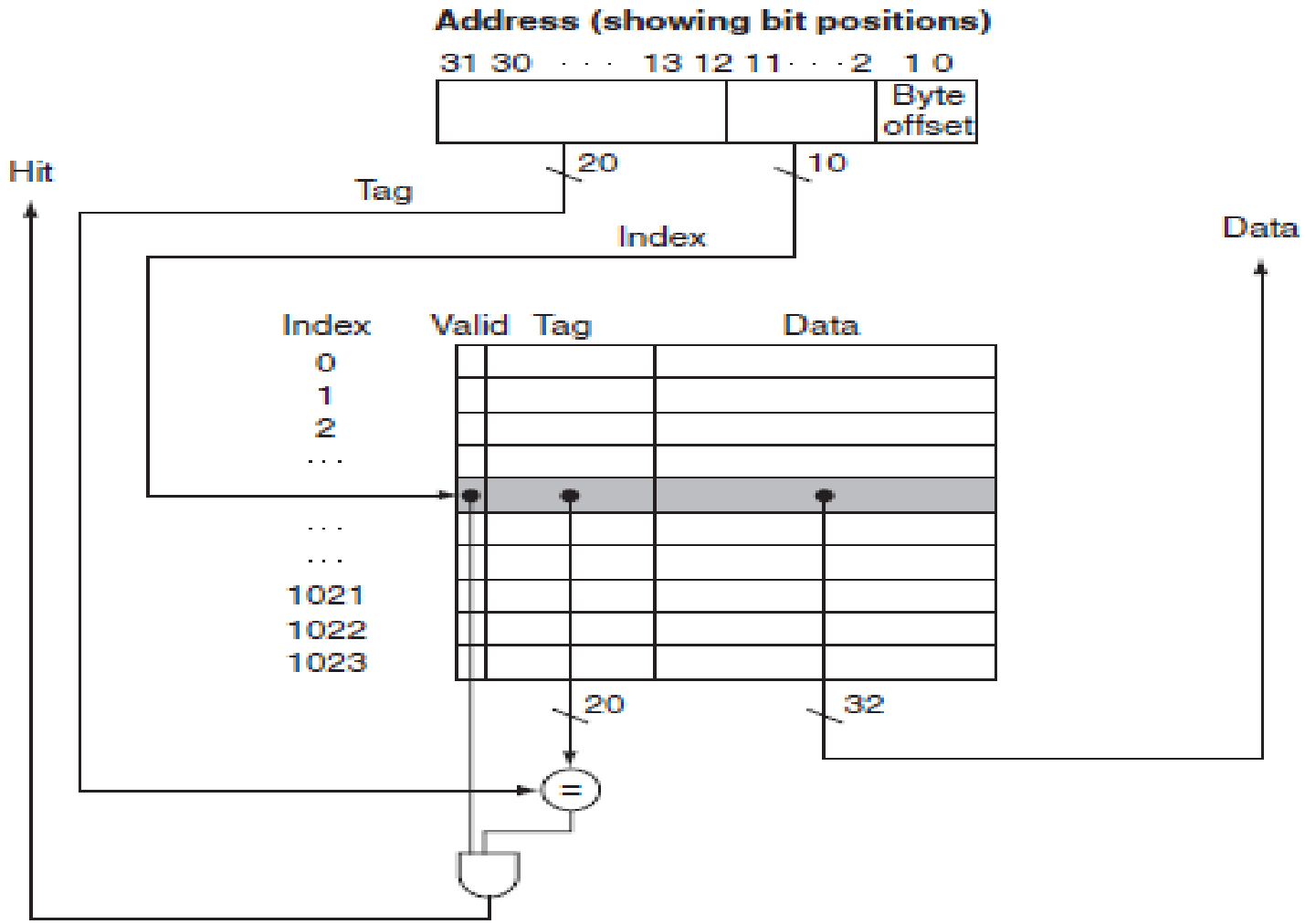
Continued…

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Reference Address (Decimal)	Reference Address (Binary)	Hit or Miss	Cache Location
22	10110	Miss	10**110** =110
26	11010	Miss	11**010** =010
22	10110	Hit	10**110** =110
26	11010	Hit	11**010** =010
16	10000	Miss	10**000** =000
3	00011	Miss	00**011** =011
16	10000	Hit	10**000** =000
18	10010	Miss	10**010** =010
16	10000	Hit	10**000** =000

Direct mapping hardware



Multi-word cache

- If a cache has 2^n blocks and each block has 2^m words, the number of bits required for representing the cache address will be "n + m"
  - **n → to address the block with in the cache**
  - **m→ to address the word within the block**

- No. of address bits required for the above cache including the bytes within words will be "n+m+2"

For byte with in word (4 bytes so 2 bits are enough)

Bits in a cache

- The total number of bits needed for a cache is a function of **the cache size and the address size** because the cache includes both the storage for the data and the tags.

- So if a cache has 2^n blocks and each block has 2^m words. If the main memory address is X bits then tag can be calculated as,

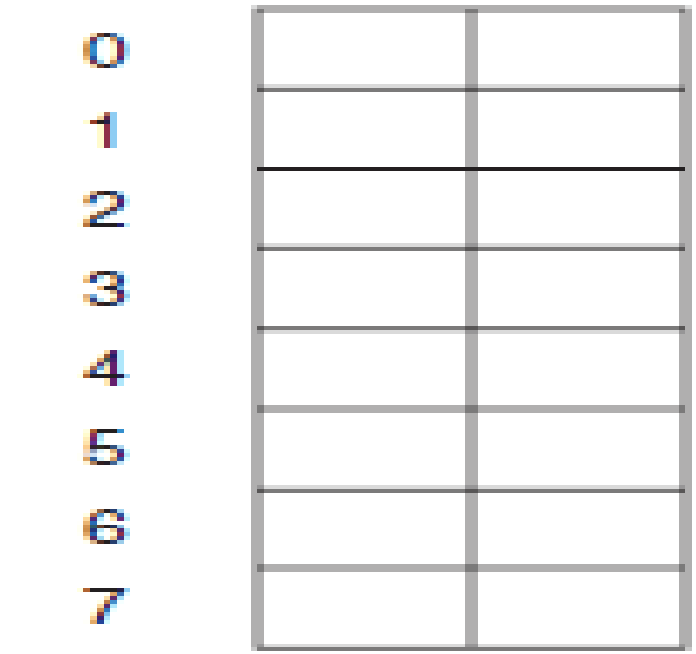
Tag size = X-(n+m+2)

SET ASSOCIATIVE

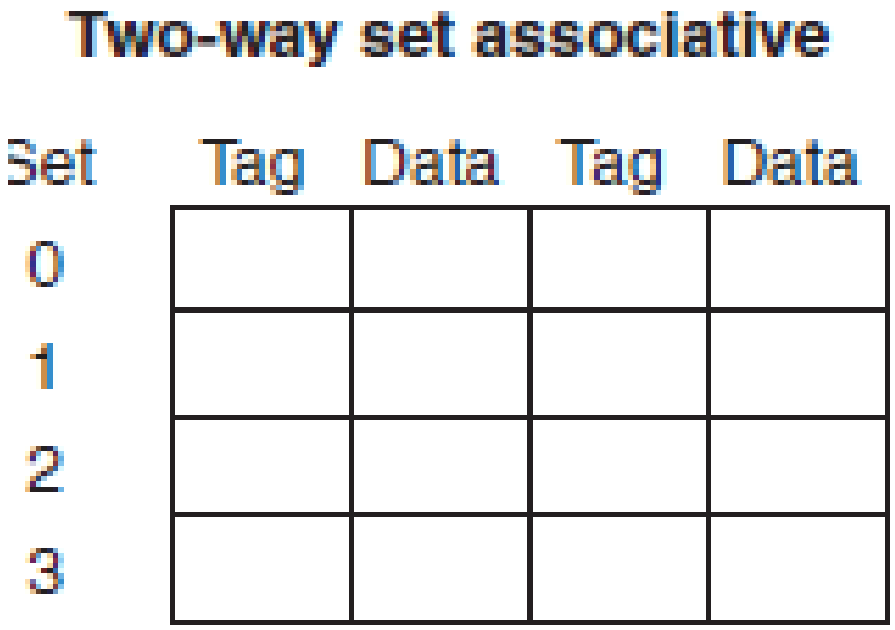
- A cache that has a fixed number of locations (at least two) where each block can be placed.
  - **(Block number) modulo (Number of sets in the cache)**

Direct Mapped or One-Way Set Associative

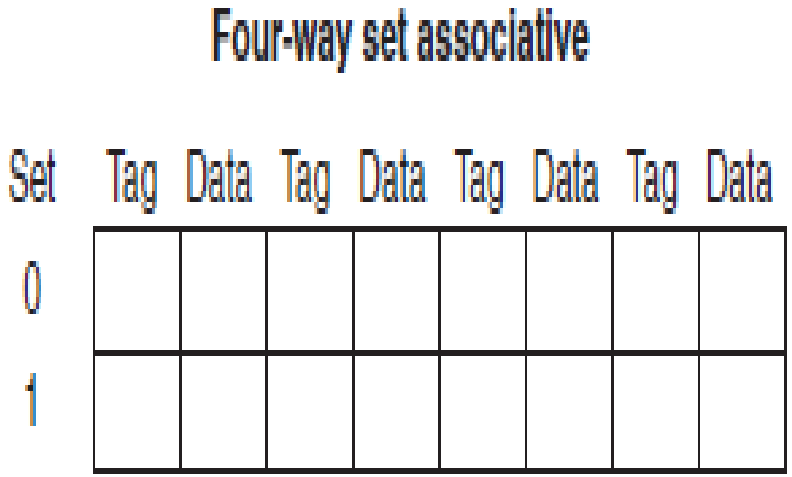
One-way set associative (direct mapped)



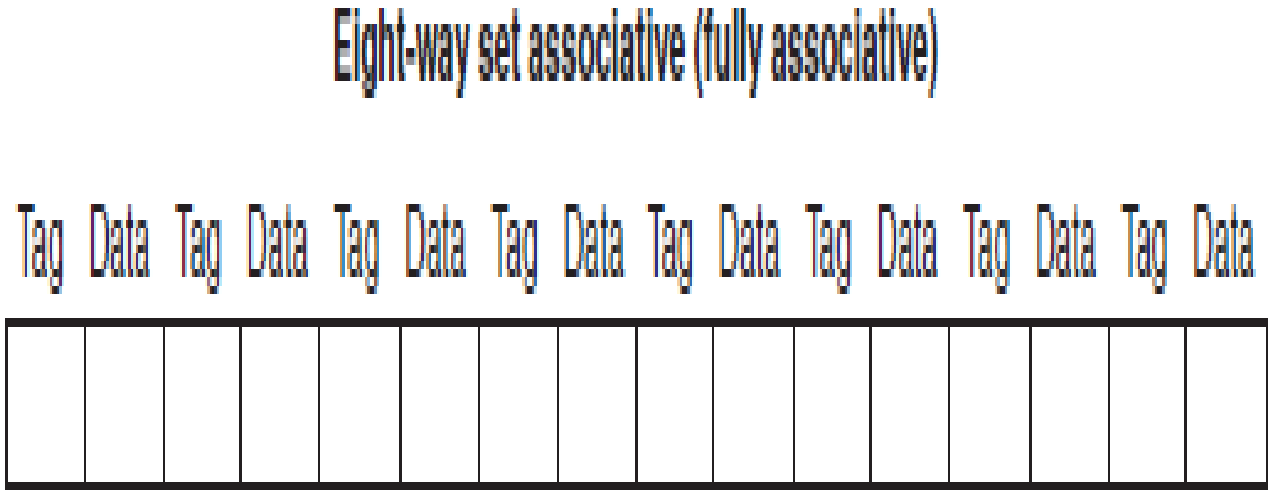
Two Way Set Associative



Four Way Set Associative



8-Way Set Associative



Set Associative Mapping

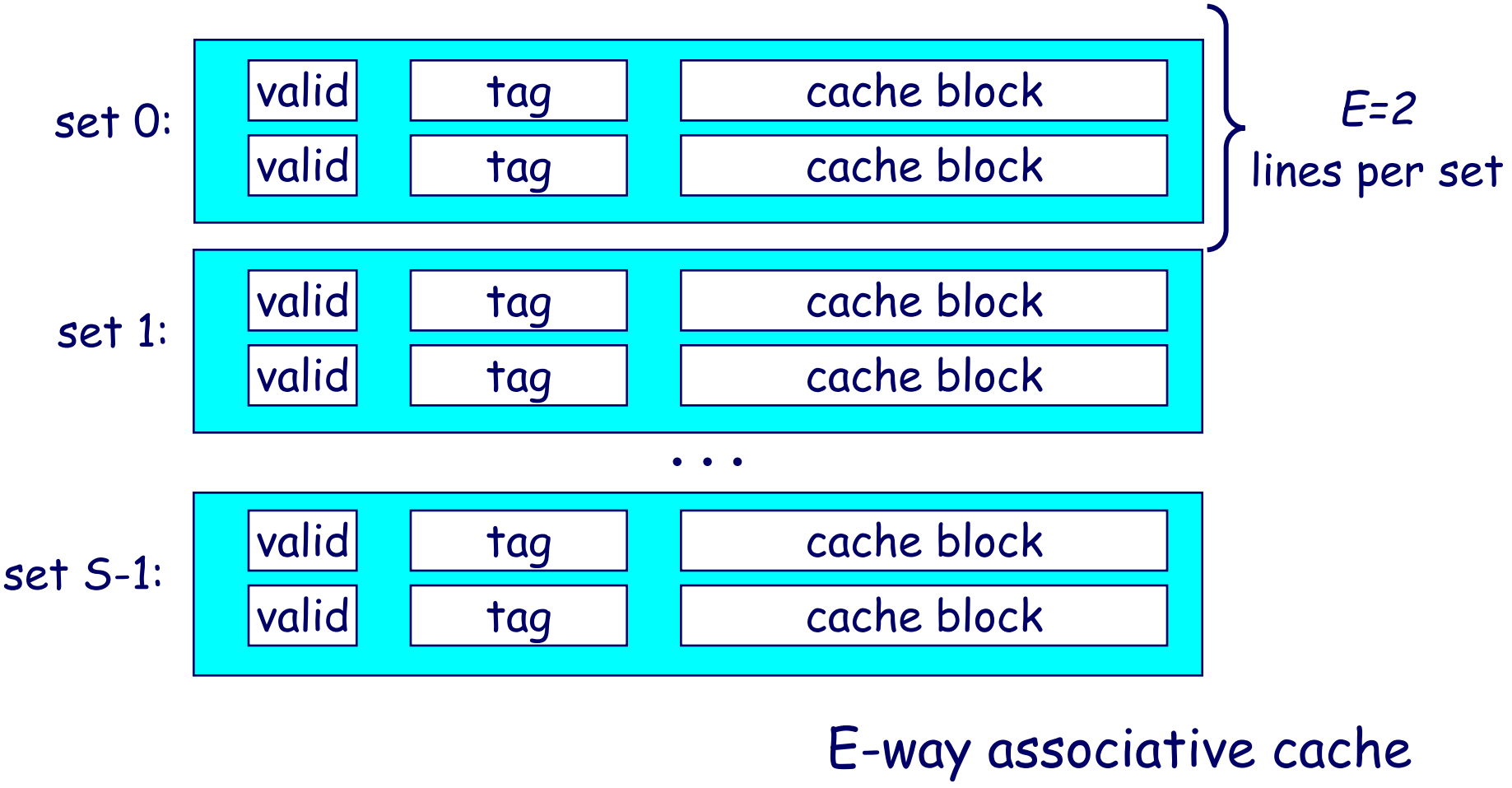
- Cache is divided into a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set i
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

Set Associative Mapping Example

- 13 bit set number
- Block number in main memory is modulo 2^{13}
- 000000, 00A000, 00B000, 00C000 … map to same set

Example: Set Associative Cache

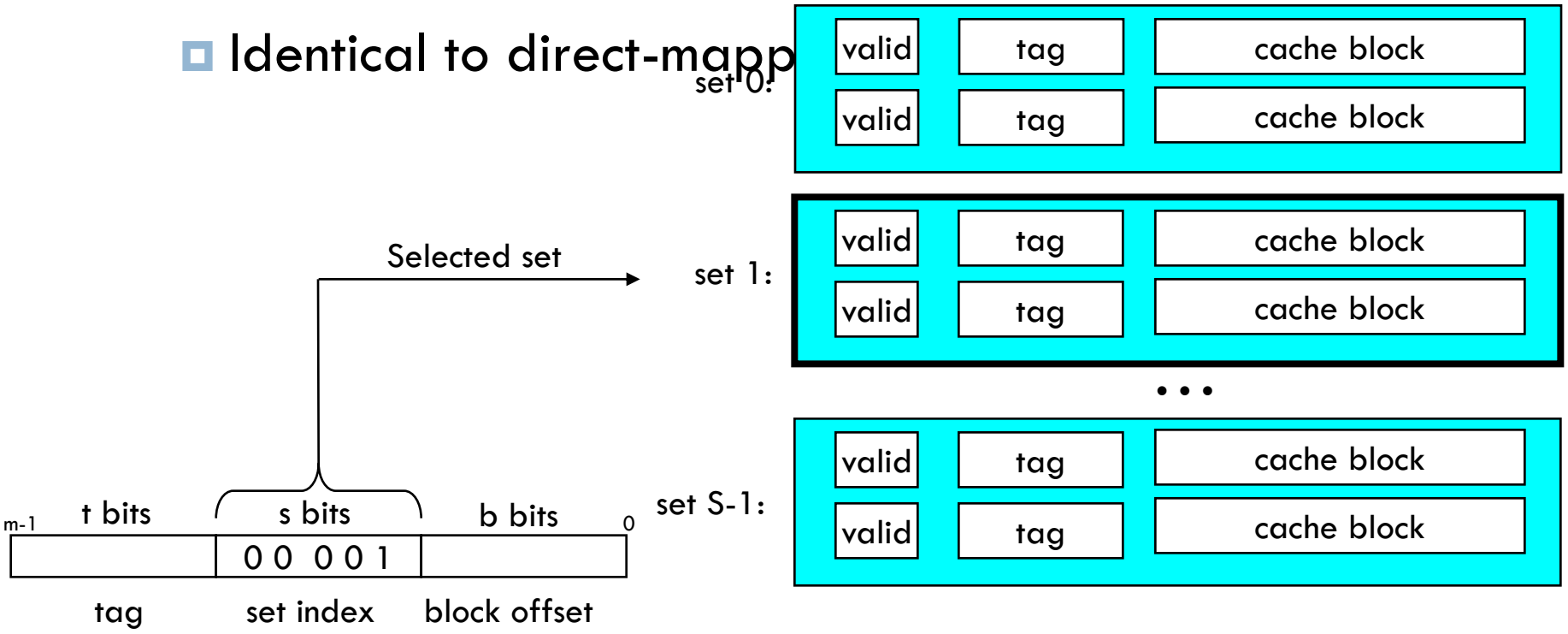
Characterized by more than one line per set



E-way associative cache

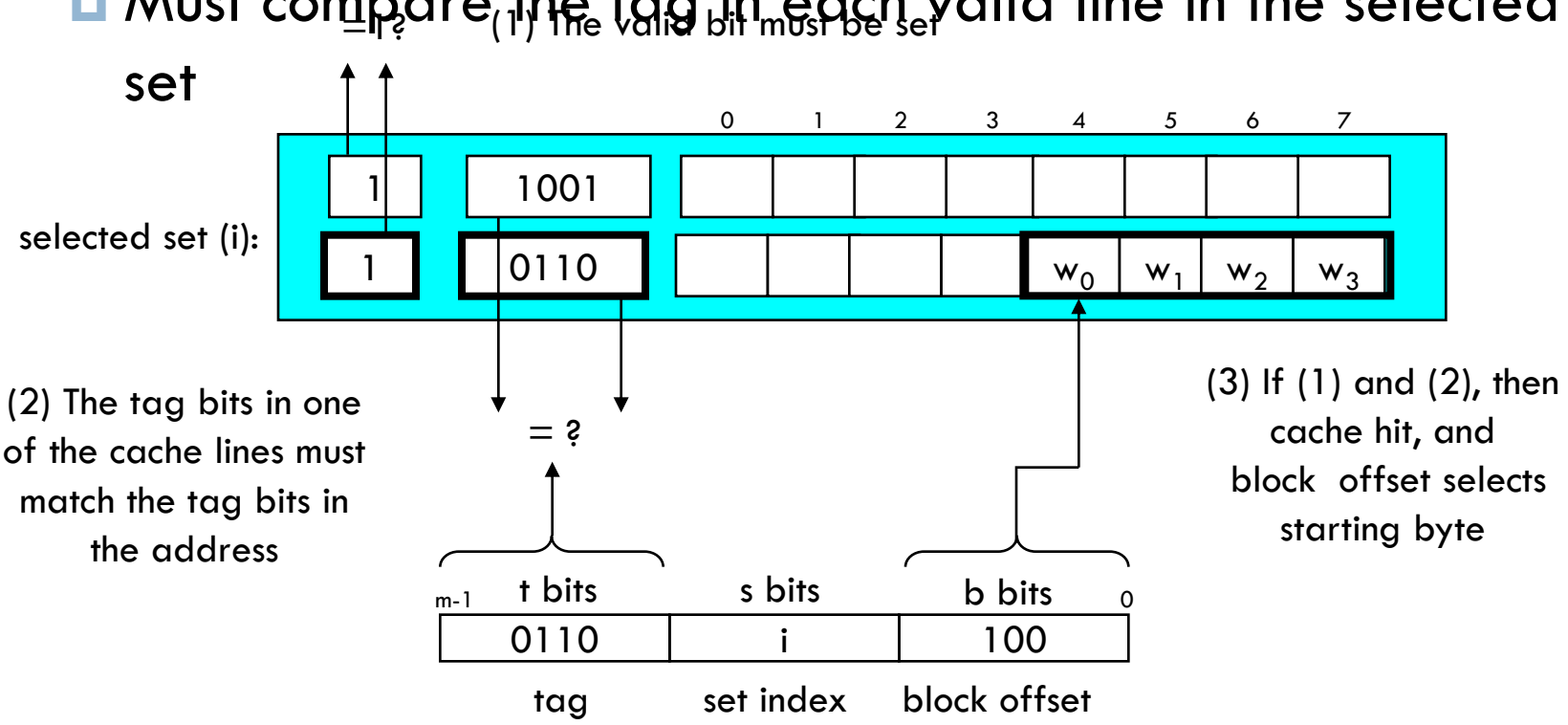
Accessing Set-Associative Caches

- Set selection
  - Identical to direct-mapp



Accessing Set Associative Caches

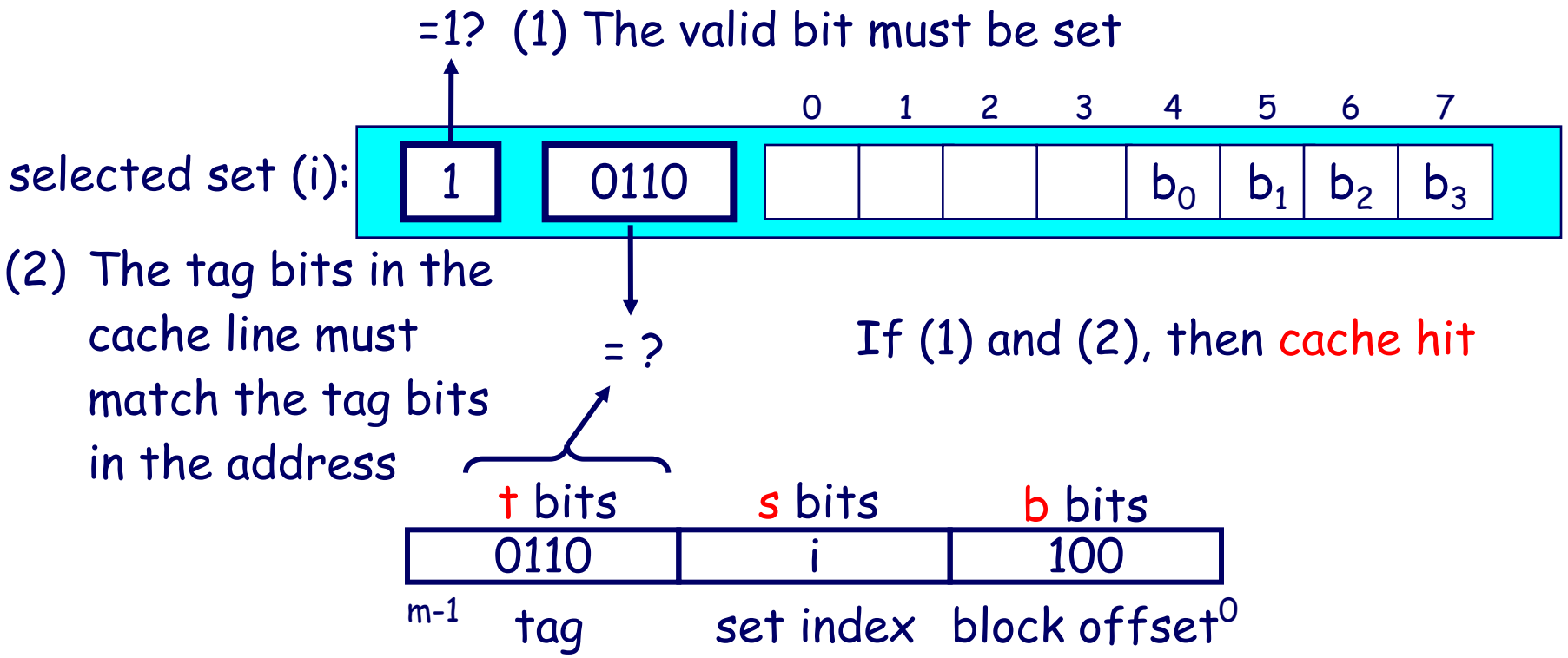
- Line matching and word selection
  - Must compare the tag in each valid line in the selected set



Accessing Caches

Line matching and word selection

- Line matching: Find a valid line in the selected set with a matching tag
- Word selection: Then extract the word



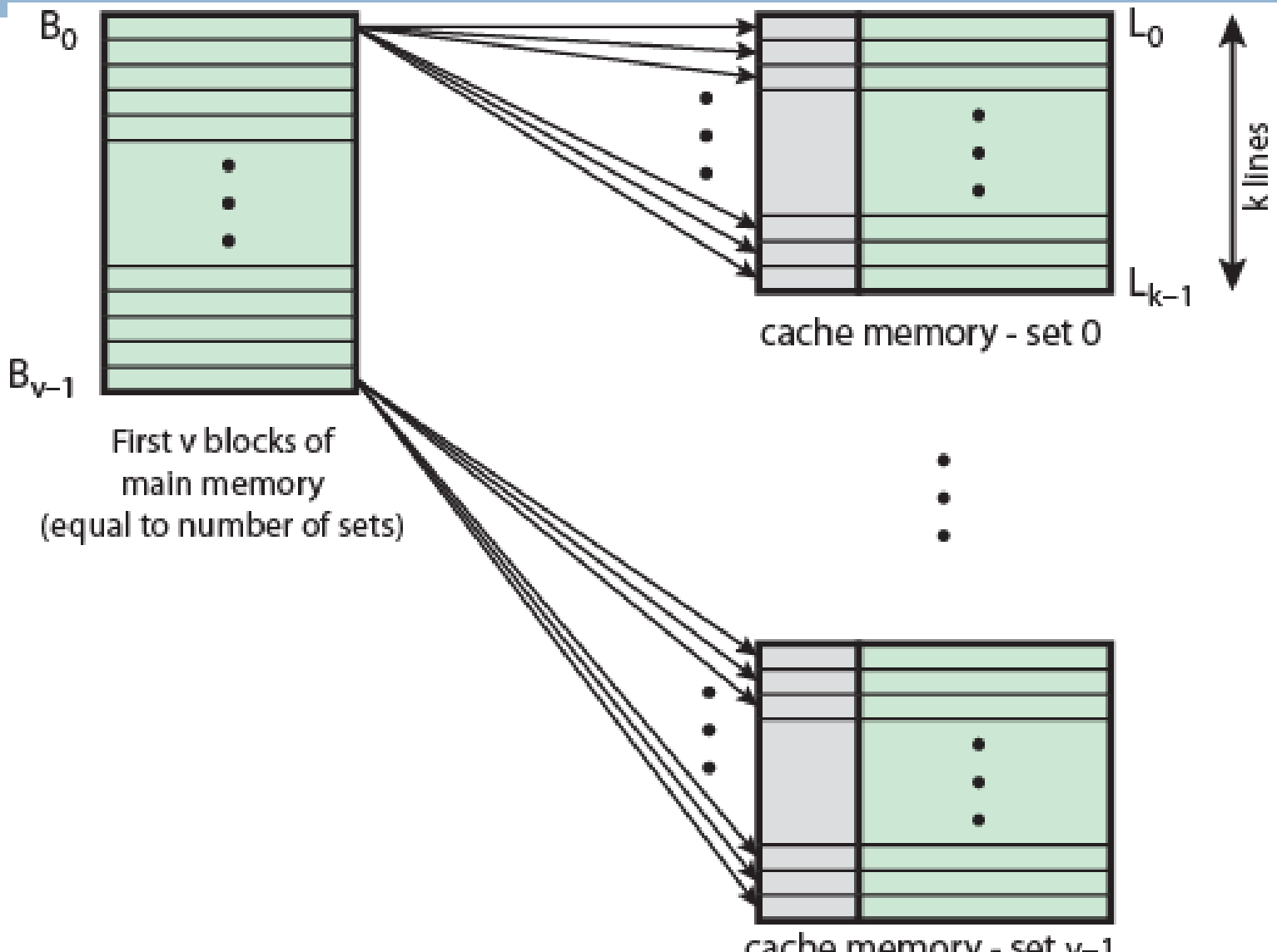
Locating a block in the Set Associative cache

- **Tag** : To see if it matches the block address from the processor
- The index value is used to select the set containing the address of interest
- Because speed is of the essence, all the tags in the selected set are searched in parallel

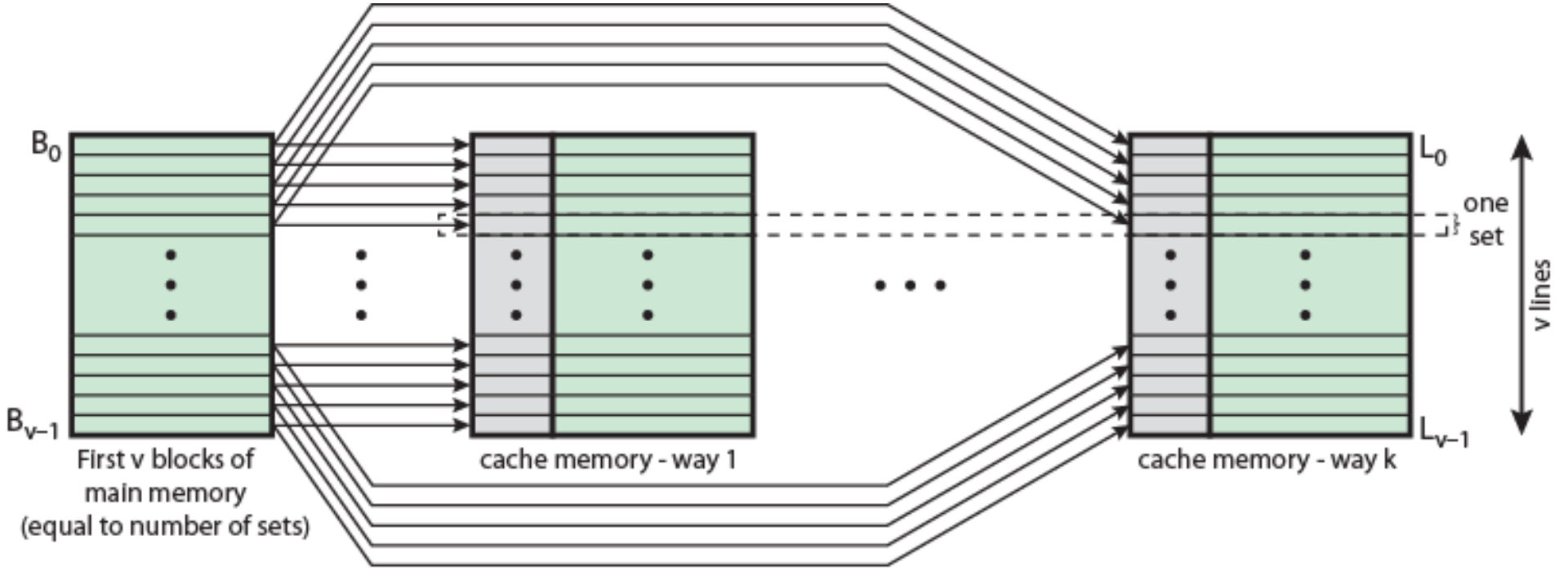
Continued…

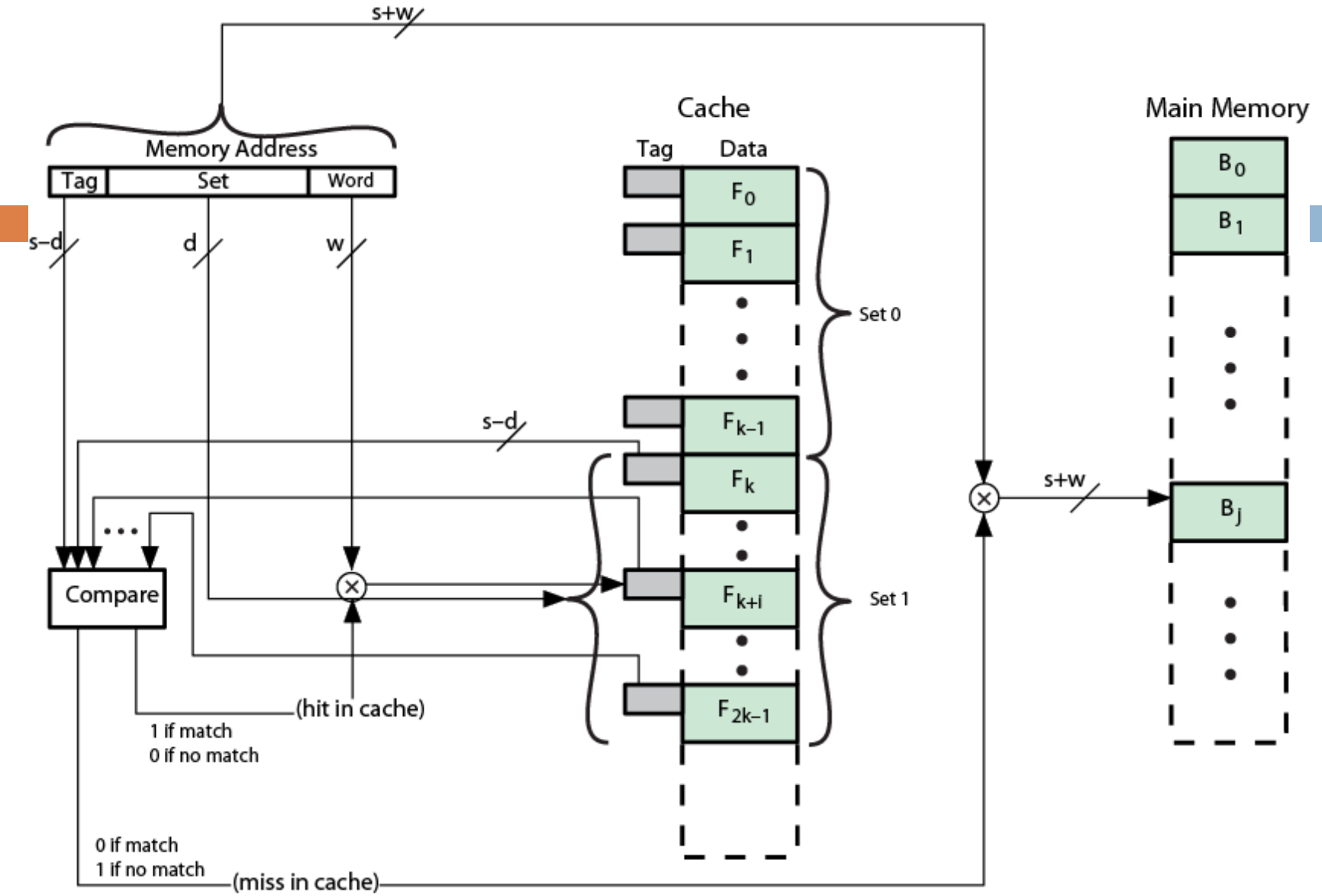
- If the total cache size is kept the same,
  - Increasing the associativity increases the number of blocks per set
  - This increases the number of simultaneous compares needed to perform the search in parallel

Mapping From Main Memory to Cache: v Associative

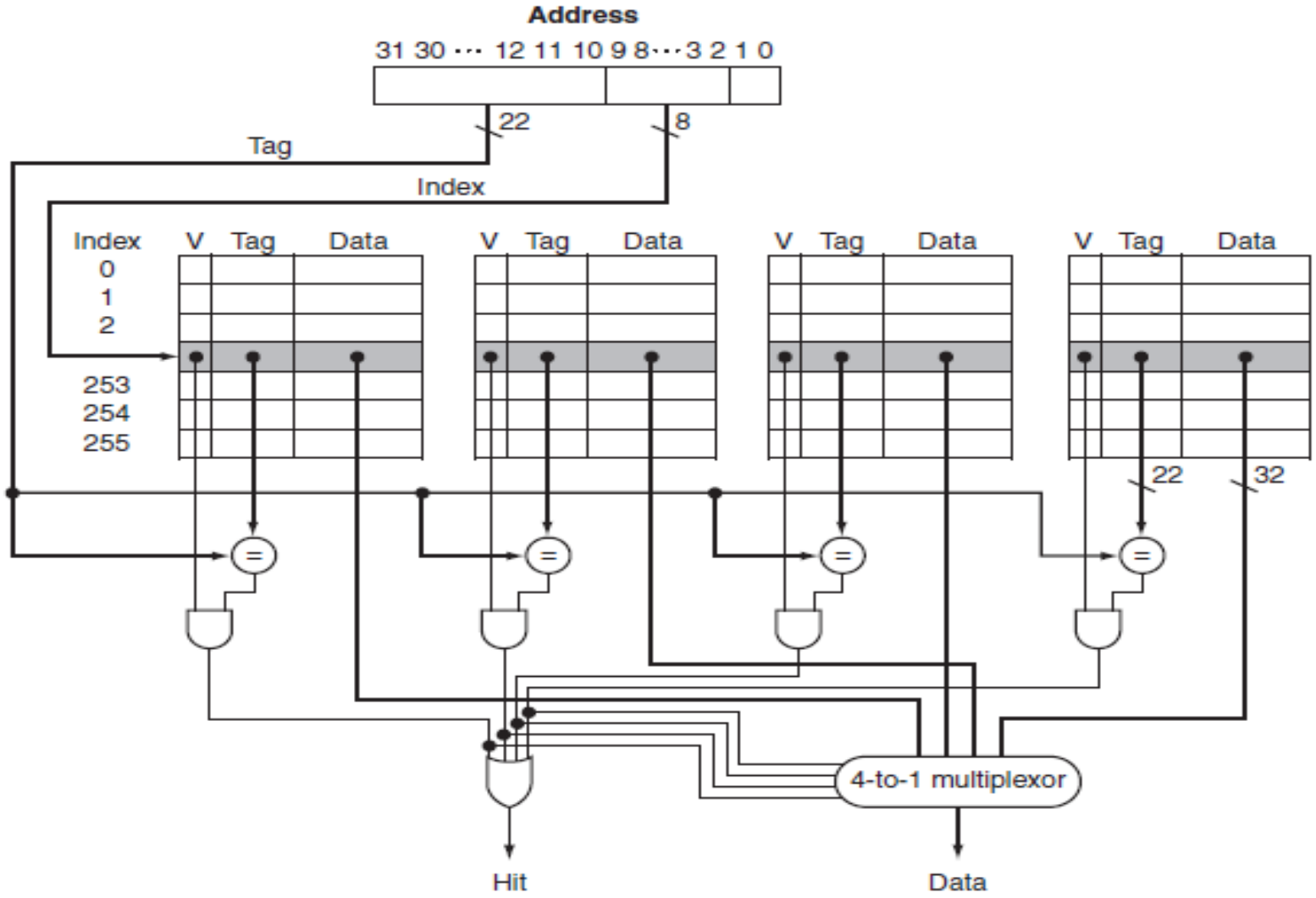


Mapping From Main Memory to Cache: k-way Associative




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+i}
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

Implementation of 4-way set associative cache



Set associative simulation

- https://www.scss.tcd.ie/Jeremy.Jones/VivioJS/caches/cache.htm

Fully Associative Mapping

- Cache structure in which a block can be placed in any location in the cache.
- To find a given block in a fully associative cache, all the entries in the cache must be searched because a block can be placed in any one.
- To make the search practical, it is done in parallel with a comparator associated with each cache entry
- These comparators significantly increase the hardware cost, effectively making fully associative placement practical only for caches with small numbers of blocks.

No. of Comparators needed

- In direct mapped cache,
  - 1 comparator needed
- In m-way set associative,
  - m comparators needed
  - m to 1 mux is also needed
- In fully associative
  - As many comparators as there are blocks

Replacement in Fully Associative and Set Associative Caches

- What to be replaced when there is a MISS in Set-Associative / Fully Associative Cache

Eg: Consider 16 blocks grouped in to 4 sets in 4-way set associative cache. Each block contains 1 byte.

4-Way Set Associative

Set 0	Block 0	Block 1	Block 2	Block 3
Set 1	Block 0	Block 1	Block 2	Block 3
Set 2	Block 0	Block 1	Block 2	Block 3
Set 3	Block 0	Block 1	Block 2	Block 3

Continued…

- If references are like, 0 ,1, 2, 3,7, 16, 20, 24

1. 0→00000 →{goes to Set 0, any block}(Block 0)
2. 1→00001→{Set 0;Any block except 0}(Block 1)
3. 2→00010→{Set 0; Any block except 0,1}(Block 2)
4. 3→00011→{Set 0;Any Block except 0,1,2}(Block 3)
5. 7→00111→{Set 0;Any Block} →Replacement Needed

What to replace?

- Many algorithms Exist → LRU →Popular
- LRU: Least Recent Used
- So in the above scenario replace Block 0

Cache Performance Metrics

- Miss Rate
  - Fraction of memory references not found in cache (misses/references)
  - Typical numbers:
    - 3-10% for L1
    - Can be quite small (e.g., < 1%) for L2, depending on size, etc.
- Hit Time
  - Time to deliver a line in the cache to the processor (includes time to determine whether the line is in the cache)
  - Typical numbers:
    - 1 clock cycle for L1
    - 3-8 clock cycles for L2
- Miss Penalty
  - Additional time required because of a miss
    - Typically 25-100 cycles for main memory
- Average Access Time = Hit Time + Miss Rate * Miss Penalty

Cache Performance

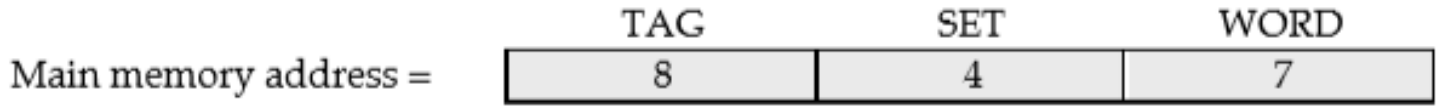
- Average Memory Access Time
- **AMAT = Time for a hit + Miss rate X Miss penalty**

Cache memory

- A set-associative cache consists of 64 lines, or slots, divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of main memory addresses.

A set-associative cache consists of 64 lines, or slots, divided into four-line sets. Main memory contains 4K blocks of 128 words each. Show the format of main memory addresses.

The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number. Main memory consists of $4K = 2^{12}$ blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits. Each block contains 128 words. Therefore, 7 bits are needed to specify the word.



(b) Given the following specification:

For every 1000 CPU-to-memory references

40 will miss in L1$;

20 will miss in L2$;

10 will miss in L3$;

L1$ hits in 1 clock cycle;

L2$ hits in 10 clock cycles;

L3$ hits in 100 clock cycles;

Main memory access is 400 clock cycles;

There are 1.25 memory references per instruction; and

The ideal CPI is 1.

Answer the following questions:

(i) What is the local miss rate in the L2$?

(ii) What is the global miss rate in the L2$?

(iii) What is the local miss rate in the L3$?

(iv) What is the global miss rate in the L3$?

(b) Given the following specification:

For every 1000 CPU-to-memory references
- 40 will miss in L1$;
- 20 will miss in L2$;
- 10 will miss in L3$;

L1$ hits in 1 clock cycle;
L2$ hits in 10 clock cycles;
L3$ hits in 100 clock cycles;
Main memory access is 400 clock cycles;
There are 1.25 memory references per instruction; and
The ideal CPI is 1.

Answer the following questions:

(i) What is the local miss rate in the L2$?

20/40 = 50%

(ii) What is the global miss rate in the L2$?

20/1000 = 2%

(iii)What is the local miss rate in the L3$?

10/20 = 50%

(iv)What is the global miss rate in the L3$?

10/1000 = 1%

(b) Given the following specification:

For every 1000 CPU-to-memory references

40 will miss in L1$;

20 will miss in L2$;

10 will miss in L3$;

L1$ hits in 1 clock cycle;

L2$ hits in 10 clock cycles;

L3$ hits in 100 clock cycles;

Main memory access is 400 clock cycles;

There are 1.25 memory references per instruction; and

The ideal CPI is 1.

Answer the following questions:

(v) What is the AMAT with all three levels of cache?

(vi) What is the AMAT for a *two-level* cache *without L3$?*

(b) Given the following specification:

For every 1000 CPU-to-memory references
- 40 will miss in L1$;
- 20 will miss in L2$;
- 10 will miss in L3$;

L1$ hits in 1 clock cycle;
L2$ hits in 10 clock cycles;
L3$ hits in 100 clock cycles;
Main memory access is 400 clock cycles;
There are 1.25 memory references per instruction; and
The ideal CPI is 1.

Answer the following questions:

(v) What is the AMAT with all three levels of cache?

(v) What is the AMAT with all three levels of cache?

1 + 4%*(10+50%*(100+50%*400)) = 1 + 0.4 + 2%*300 = 7.4

(vi) What is the AMAT for a *two-level* cache *without L3$?*

1 + 4%*10 + 2%*400 = 1 + 0.4 + 8 = 9.4

- In (vi) 2% is derived by (4% * 50% (4%*(10+50%*400))

Practice Problems

- A block-set associative cache memory consists of 128 blocks divided into four block sets . The main memory consists of 16,384 blocks and each block contains 256 eight bit words.
- How many bits are required for addressing the main memory?
- How many bits are needed to represent the TAG, SET and WORD fields?

block sets . The main memory consists of 16,384 blocks and each block contains 256 eight bit words.
How many bits are required for addressing the main memory?
How many bits are needed to represent the TAG, SET and WORD fields?

- Given-
- Number of blocks in cache memory = 128
- Number of blocks in each set of cache = 4
- Main memory size = 16384 blocks
- Block size = 256 bytes
- 1 word = 8 bits = 1 byte

Solution:

Main Memory Size-

We have-

Size of main memory

= 16384 blocks

= 16384 x 256 bytes

= 2^{22} bytes

Thus, Number of bits required to address main memory = 22 bits

Number of Bits in Block Offset-

We have-

Block size

$= 256$ bytes

$= 2^8$ bytes

Thus, Number of bits in block offset or word $= 8$ bits

Number of Bits in Set Number-

Number of sets in cache

= Number of lines in cache / Set size

= 128 blocks / 4 blocks

= 32 sets

= 2^5 sets

Thus, Number of bits in set number = 5 bits

Number of Bits in Tag Number-

Number of bits in tag

= Number of bits in physical address – (Number of bits in set number + Number of bits in word)

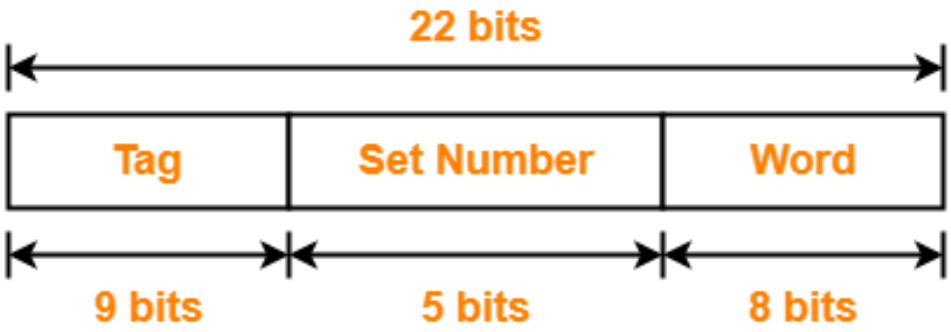
= 22 bits – (5 bits + 8 bits)

= 22 bits – 13 bits

= 9 bits

Thus, Number of bits in tag = 9 bits

Thus, physical address is-



- A 4-way set associative cache memory unit with a capacity of 16 KB is built using a block size of 8 words. The word length is 32 bits. The size of the physical address space is 4 GB. The number of bits for the TAG field is _____.

Solution-

Given-

Set size = 4 lines

Cache memory size = 16 KB

Block size = 8 words

1 word = 32 bits = 4 bytes

Main memory size = 4 GB

Number of Bits in Physical Address-

- We have,
- Main memory size
- = 4 GB
- = 2^{32} bytes
- Thus, Number of bits in physical address = 32 bits

Number of Bits in Block Offset-

- We have,
- Block size
- = 8 words
- = 8 x 4 bytes
- = 32 bytes
- = 2^5 bytes
- Thus, Number of bits in block offset = 5 bits
-

Number of Lines in Cache-

Number of lines in cache

$= \text{Cache size} / \text{Line size}$

$= 16 \text{ KB} / 32 \text{ bytes}$

$= 2^{14} \text{ bytes} / 2^{5} \text{ bytes}$

$= 2^{9} \text{ lines}$

$= 512 \text{ lines}$

Thus, Number of lines in cache $= 512$ lines

Number of Sets in Cache-

Number of sets in cache

= Number of lines in cache / Set size

= 512 lines / 4 lines

= 2^9 lines / 2^2 lines

= 2^7 sets

Thus, Number of bits in set number = 7 bits

Number of Bits in Tag-

Number of bits in tag

= Number of bits in physical address – (Number of bits in set number + Number of bits in block offset)

= 32 bits – (7 bits + 5 bits)

= 32 bits – 12 bits

= 20 bits

Thus, number of bits in tag = 20 bits

Replacement Algorithms (2) Associative & Set Associative

- Hardware implemented algorithm (speed)
- Least Recently used (LRU)
- e.g. in 2 way set associative
  - Which of the 2 block is lru?
- First in first out (FIFO)
  - replace block that has been in cache longest
- Least frequently used
  - replace block which has had fewest hits
- Random

What happens on a write?

	Write-Through	Write-Back
Policy	Data written to cache block also written to lower-level memory	Write data only to the cache Update lower level when a block falls out of the cache
Debug	Easy	Hard
Do read misses produce writes?	No	Yes
Do repeated writes make it to lower level?	Yes	No

Additional option -- let writes to an un-cached address allocate a new cache line ("write-allocate").

Where to misses come from?

- Classifying Misses: 3 Cs
  - *Compulsory*—The first access to a block is not in the cache, so the block must be brought into the cache. Also called *cold start misses* or *first reference misses*.
    (Misses in even an Infinite Cache)

Capacity

- *Capacity*—If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being discarded and later retrieved. *(Misses in Fully Associative Size X Cache)*

Conflict

- *Conflict*—If block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory & capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. Also called *collision misses* or *interference misses*.
  (Misses in N-way Associative, Size X Cache)
- 4th "C":
  - *Coherence* - Misses caused by cache coherence.