SCS 1205
Computer Systems

Kasun Gunawardana

E-mail: kgg

University of Colombo School of Computing

Memory Organization

Primary & Secondary Memory

- Primary Memory
  - Cache Memory
  - Main Memory
- Secondary Memory
  - Magnetic Disks
  - Magnetic Tapes
  - USB Sticks

Types of Memory

- Read Only Memory (ROM)
  - Non-volatile
  - Variants
    - PROM (Programmable Read Only Memory)
    - EEPROM (Electrically Erasable Programmable Read Only Memory)
- Random Access Memory (RAM)
  - Volatile
  - Variants
    - Dynamic RAM (DRAM)
    - Static RAM (SRAM)

Flash Memory

- Floating-gate metal–oxide–semiconductor field-effect transistors
- Non-volatile
- Read and Write
- Common Usages
  - Memory Cards
  - USB Flash Drives
  - Solid State Drives

Solid State Devices - SSD

- Typically based on Flash Memory
- Compared to HDD
  - More resistant to physical shock
  - Run silently
  - Faster access time
- Not appropriate for long term storage
  - Without power it depletes charge over time

3D XPoint

- Newly developed memory technology

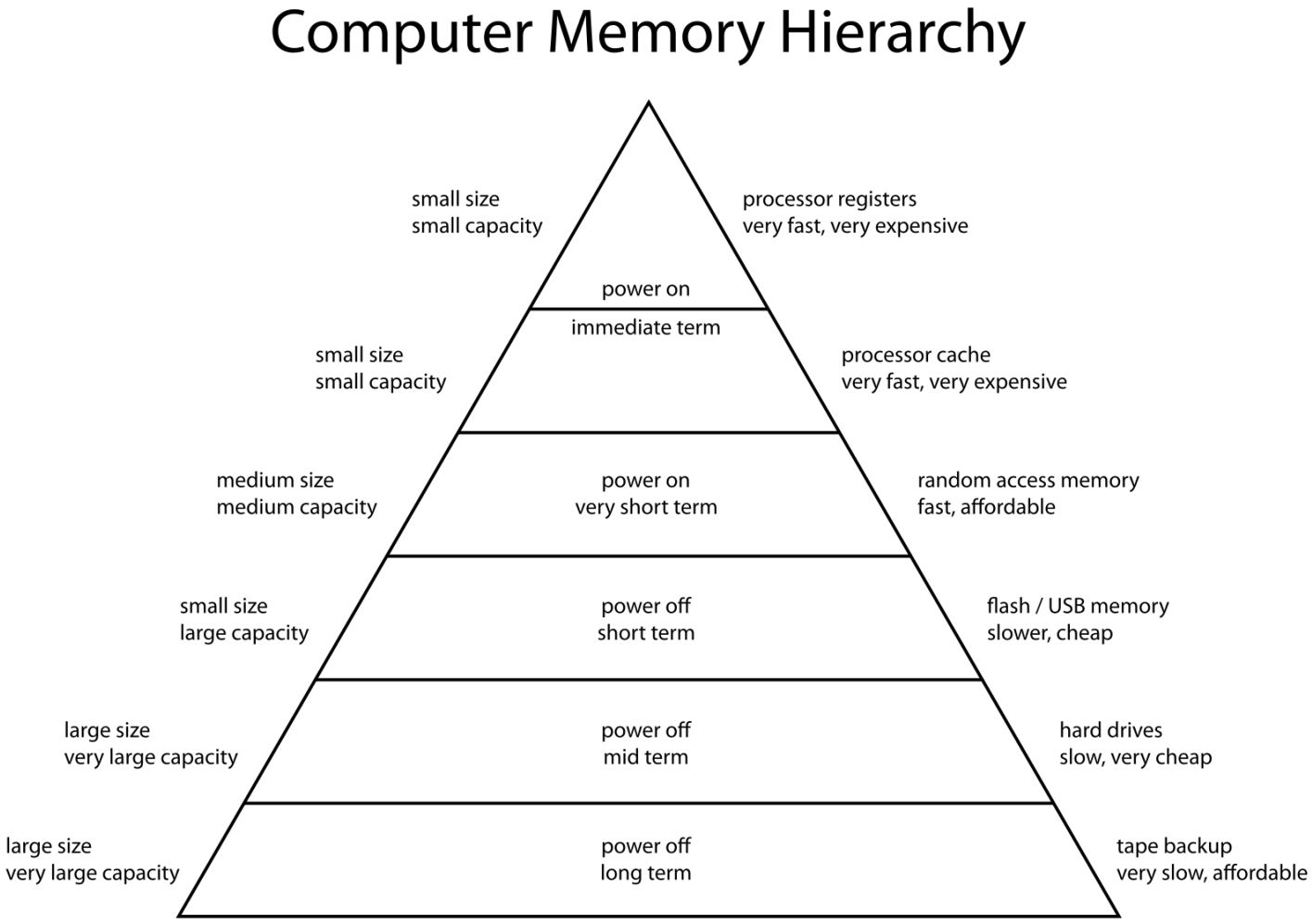
- Read more about it…
  - Start with https://en.wikipedia.org/wiki/3D_XPoint

Memory Hierachy

Organization of computer storage components based on different attributes.

- Access time/ Latency
- Cost
- Size
- Distance from CPU

Memory Hierachy



https://en.wikipedia.org/wiki/Memory_hierarchy

Typical Data Access Pattern

- CPU requests data from its nearest storage
  - Cache
- If not found, then request goes to the Main Memory
- If not found, then request goes to the HDD
- Once the data is found it will be loaded into the Main Memory and then into Cache

The CPU – Memory Gap

- CPU Cycle time decreases rapidly
- Memory access times cannot match the CPU speed
  - HDD > SSD > DRAM > SRAM >> CPU
- CPU - Memory gap affects the system performance
- "**Principle of Locality**" is used to bridge the gap

Principle of Locality

Tendency of CPU to access the same set of memory locations repeatedly over a short period of time.

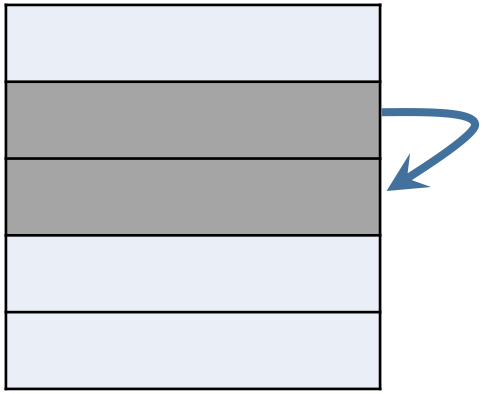
*****

Programs tend to refer data and instructions with addresses near or equal to those they have referred recently.

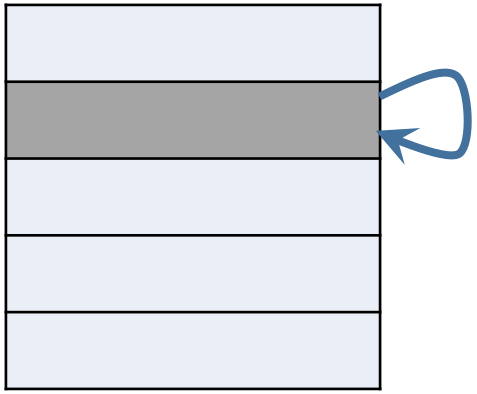
*****

Principle of Locality

- Spatial Locality: Items with nearby addresses tend to be referenced close together in time



- Temporal Locality: Recently referenced items are likely to be referenced again in the near future



Reading…

- https://en.wikipedia.org/wiki/Locality_of_reference
- https://stackoverflow.com/questions/7638932/what-is-locality-of-reference
- https://www.geeksforgeeks.org/locality-of-reference-and-cache-operation-in-cache-memory/

Cache Memory

- A smaller and faster memory space that keeps data of a larger and slower memory space.
- Reside in CPU chip
- Typical characteristics
  - Transient – Highly volatile
  - May keep frequently referred data
  - May keep data that would be referred in future

Caching…

Cache Hierarchy

- Chache memories are typically SRAMs
- Per unit cost is high
- Faster and larger cache space will be expensive
- As a solution hierarchy of caches is used
  - Different Levels of Chaches (i.e. L1, L2, L3)

Reading..

- https://stackoverflow.com/questions/3699582/what-is-the-difference-between-l1-cache-and-l2-cache

Cache Performance

- **Cache Hit**: Requested data for processing by a CPU is found in the cache memory.
- **Cache Miss**: Requested data is not found.

- **Cache Hit Ratio**: The probability of finding requested data in the cache.
- **Cache Miss Ratio** = 1 – Cache Hit Ratio

Cache Performance (Cont.)

- Effective Access Time (EAT)
  - Average time it takes for CPU to access data

- $EAT = HR \times T_C + (1 - HR) \times T_M$

$$HR - Hit\ Ratio$$
$$T_C - Cache\ Access\ Time\ in\ a\ cache\ hit$$
$$T_M - Data\ access\ time\ in\ a\ cache\ miss$$

Key Design Decisions of a Cache

- 1. Cache Volume (Capacity, Size)
- 2. Write Policy
  - Write Back
  - Write Through
- 3. Mapping Scheme
  - Direct Mapped
  - Set Associative
    - Line replacement policies

2. Cache – Write Policy

Main Memory

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
…	…

Cache Memory

2	c
3	d
10	k
11	l

What if CPU changes one of the values in cache ?

2. Cache – Write Policy (Cont.)

Main Memory

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
…	…

Cache Memory

2	c
3	d
10	k
11	l

How shall we ensure consistency and reliability of memory content?

2. Cache – Write Policy (Cont.)

- Cache is a redundant storage.
- Same data appears in two locations.
- Data maybe shared across processes.
- Maintaining consistency is a challenge when the same data appears in multiple locations.
- Consistency in Cache and Main Memory.
- Inconsistency - The data is altered (write).
- Cache needs a write policy to ensure consistency.

Write Policy – Write Through

Whenever write operation happens, it is perfromed on both cache and main memory synchronusly.

- Comparatively simple policy to be implemented.
- Bus traffic will be increased.

Write Policy – Write Back

Write operations on a data is performed only on cache. But whenever the data is evicted from the cache it is written to the main memory.

- Lazy writes.
- Complex to implement.

Reading Exercise…

Cache coherence

- https://en.wikipedia.org/wiki/Cache_coherence

3. Mapping Schemes

Main Memory

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
…	…

Cache Memory

2	c
3	d
10	k
11	l

How shall we map memory identifiers to the cache ?

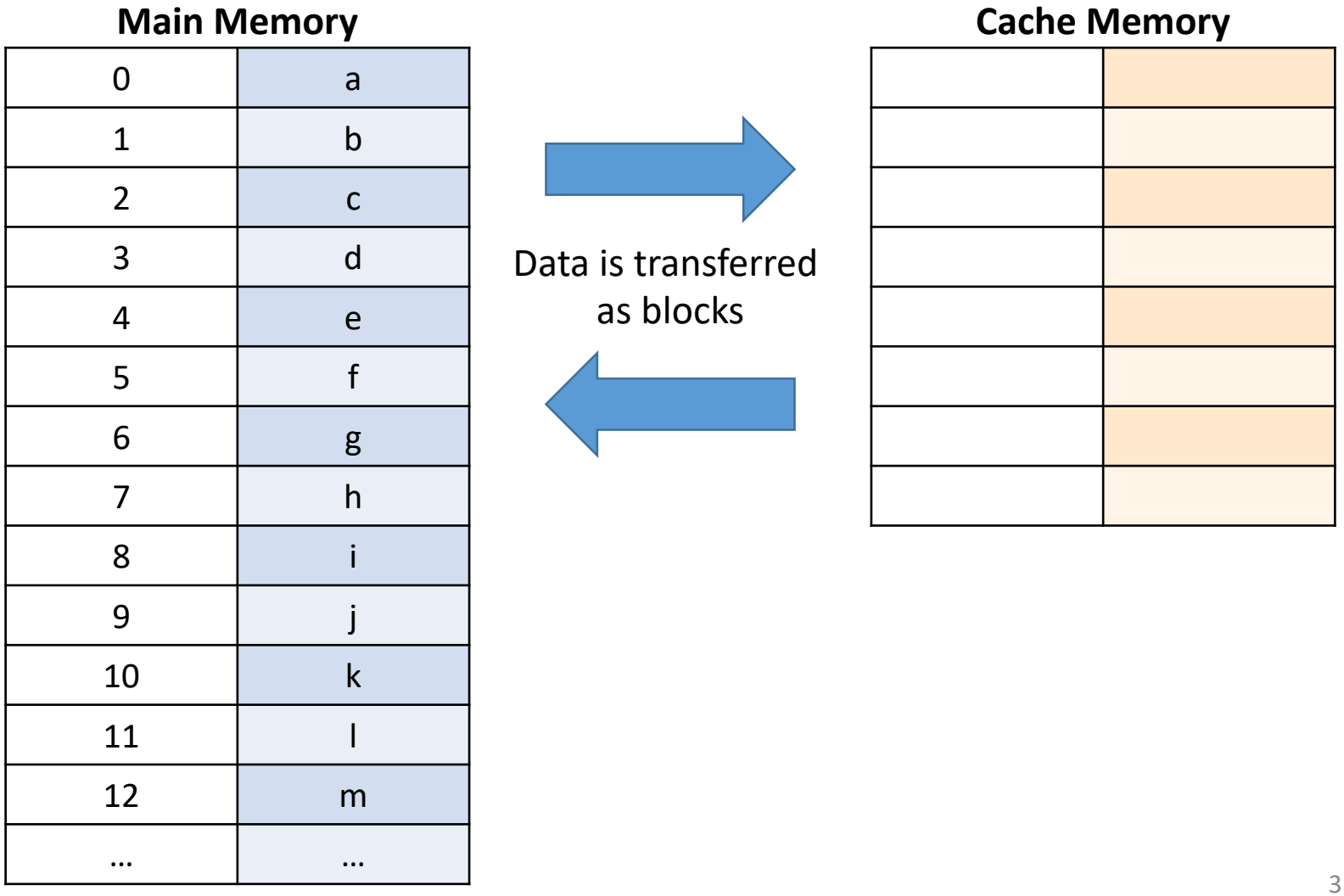
3. Mapping Schemes (Cont.)

- The CPU refers data or instruction by its addresses.
- The data is first searched in Cache.
- Thus, cached data should be able to access using addresses.
- But cache is smaller compared to the main memory.
- Cache memory locations cannot be given main memory addresses.
- A mapping scheme is required.

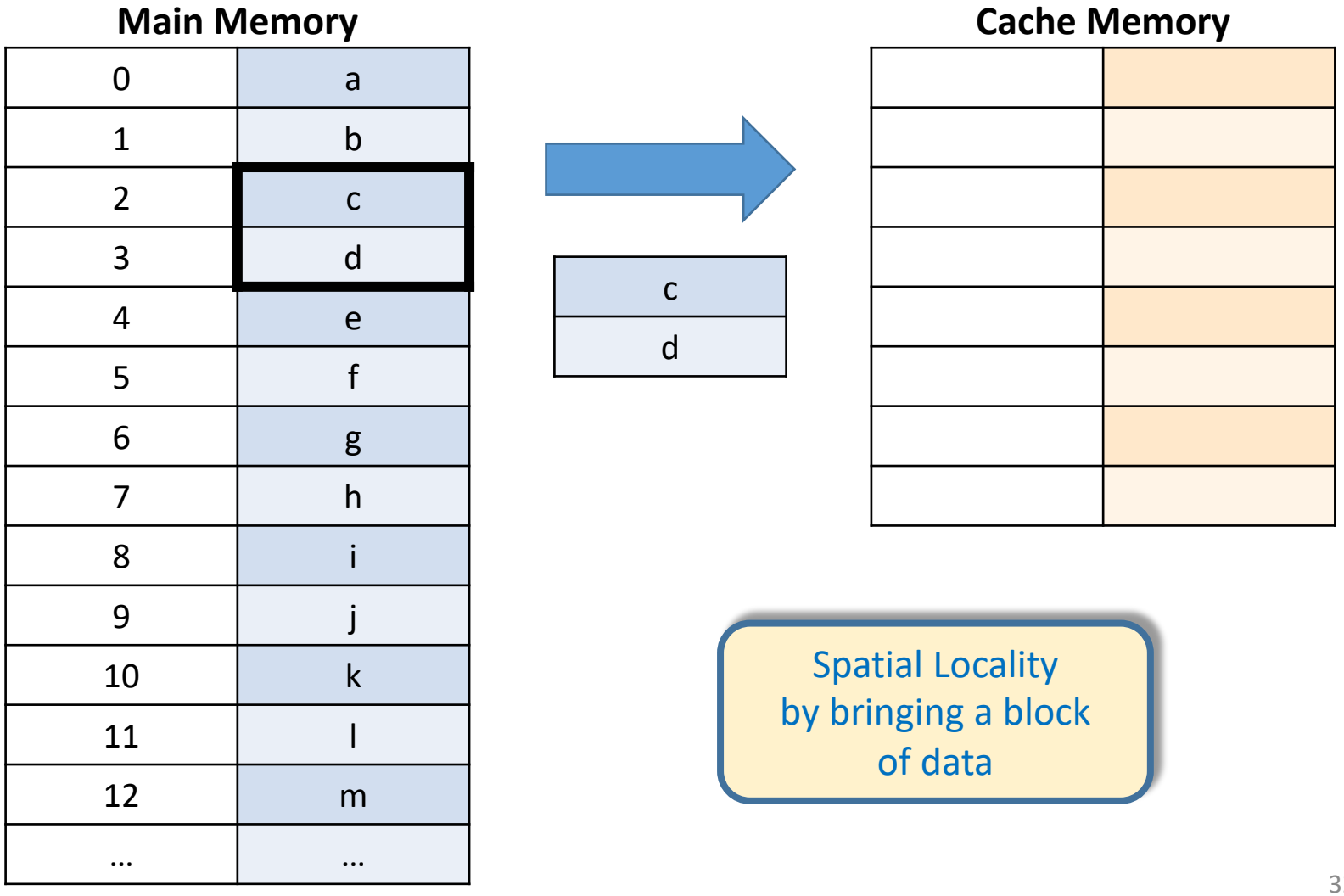
Mapping Schemes (Cont.)

- Direct Mapped
- Fully Associative
- Set Associative

Overview of a Cache



Overview of a Cache



Cache Line (Cache Block)

- Data is transferred between memory and cache in blocks of fixed size, called **Cache Lines** (cache blocks).
- When a **`Cache Line`** is copied from the memory into the cache, a **`Cache Entry`** is created.
- The **Cache Entry** will include the copied data as well as the requested memory location identifier (called a tag).
- A single cache line may contain multiple bytes/words of data.
- **Amount of data** in a **Cache Entry** and a **Cache Line** are always equivalent in size.

Next...

Cache Mapping Schemes

- Direct Mapped Cache

Computer Systems

Kasun Gunawardana

E-mail: kgg

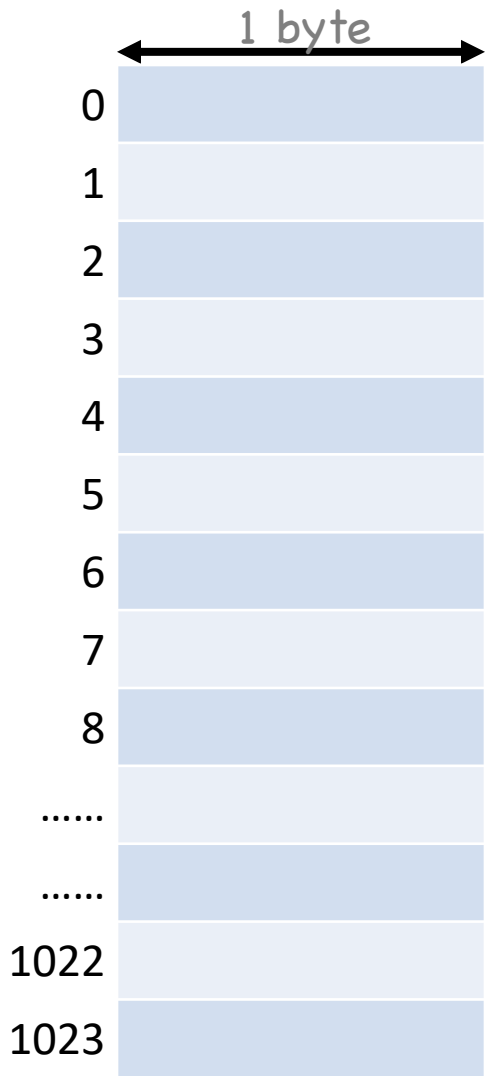
University of Colombo School of Computing

Direct Mapped Cache

Direct Mapped Cache

- A memory is identified as blocks of fixed size.
- Each block has a pre-defined cache entry position.
- A block in the memory will always map to the same cache entry position.
- The number of blocks in the main memory is larger than the maximum number of cache entries.
- Thus, several memory blocks are mapped to the same cache entry position.
- Tag is used to identify the address of a mapped data.

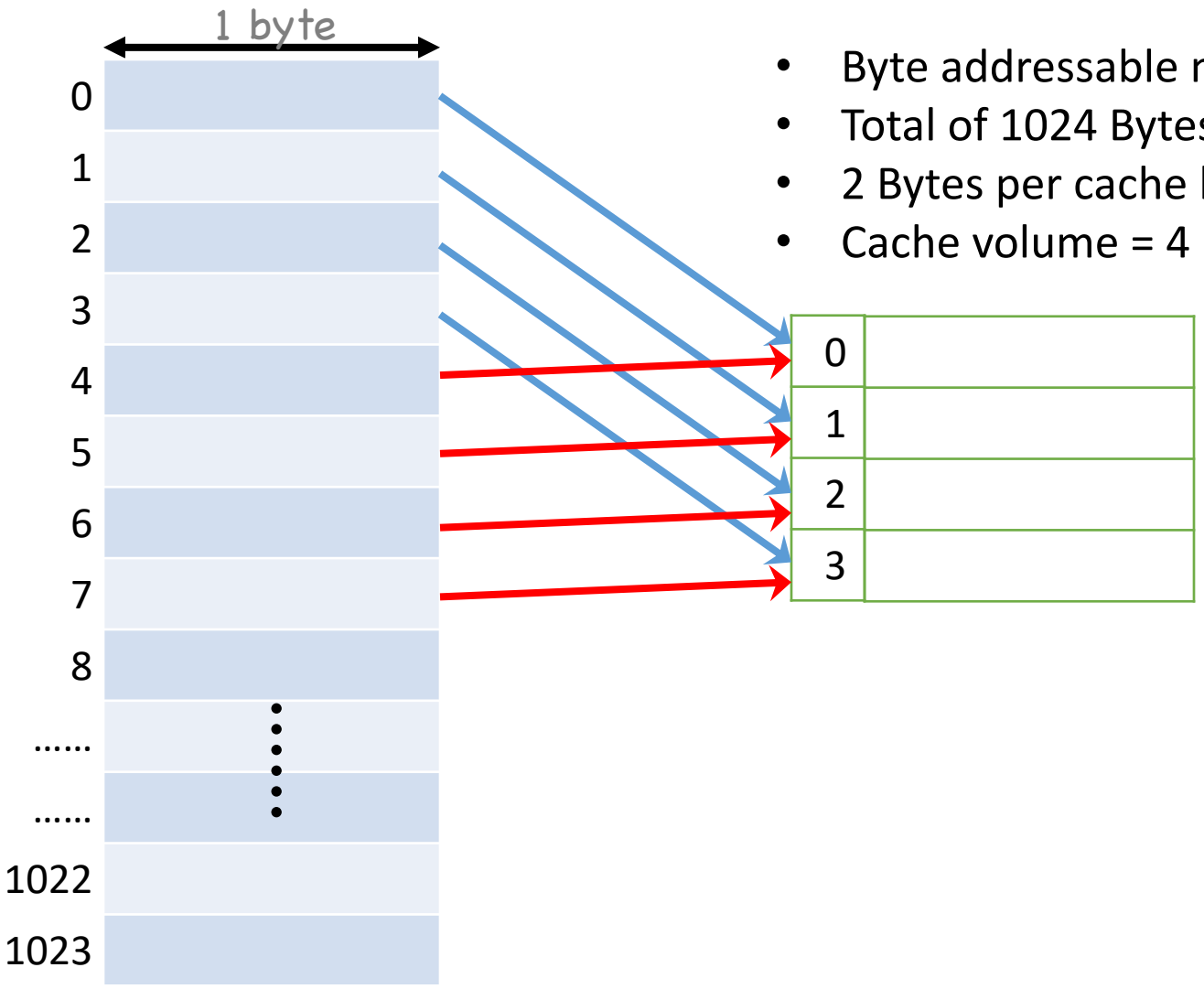
Direct Mapped Cache



- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 4 Bytes

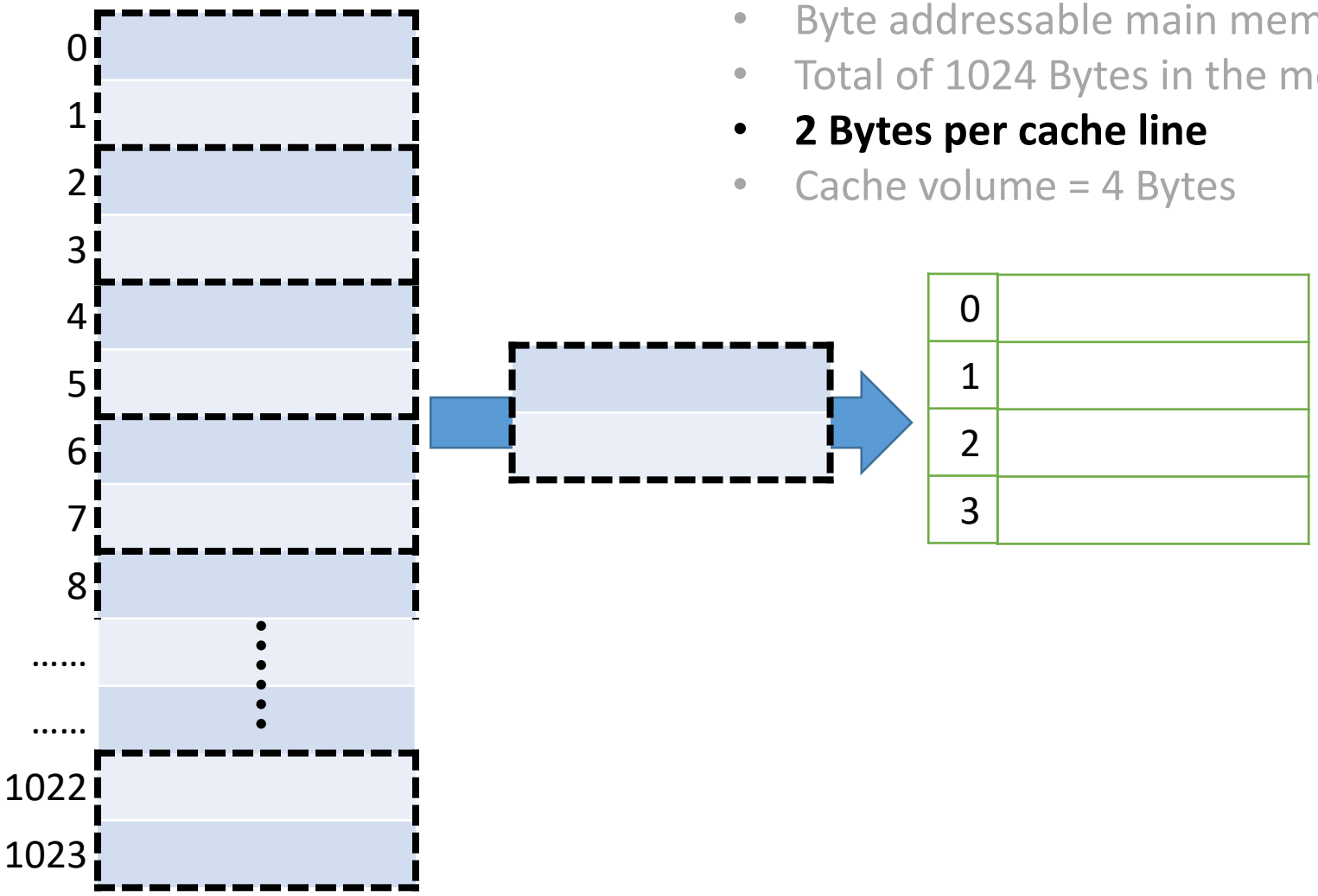
0	
1	
2	
3	

Direct Mapped Cache



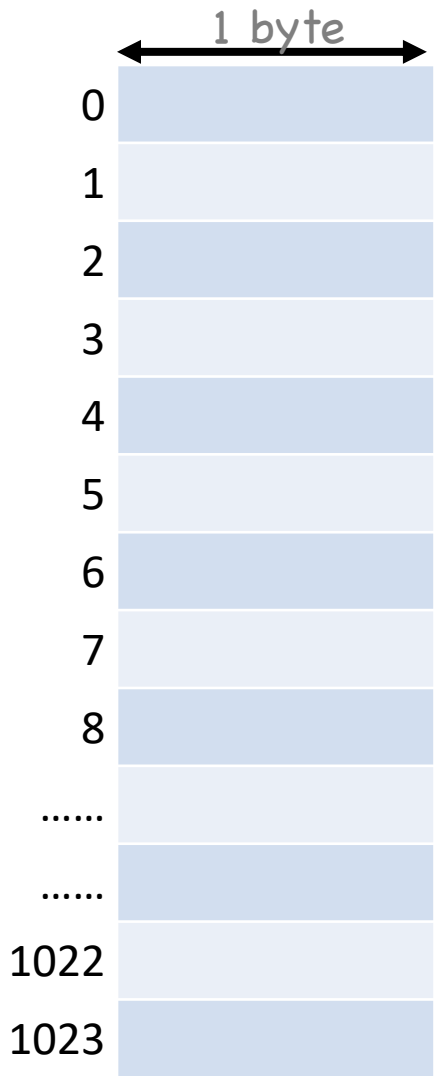
- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 4 Bytes

Direct Mapped Cache

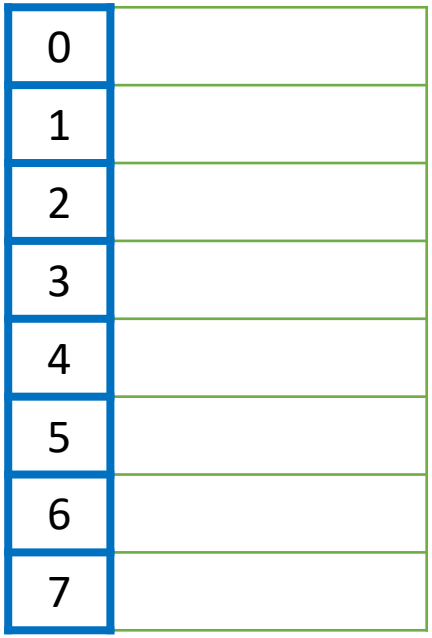


- Byte addressable main memory
- Total of 1024 Bytes in the memory
- **2 Bytes per cache line**
- Cache volume = 4 Bytes

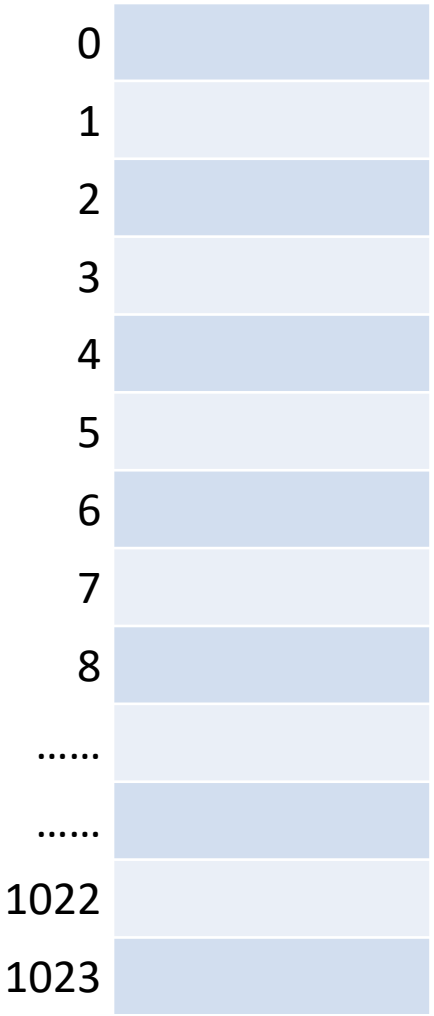
Direct Mapped Cache



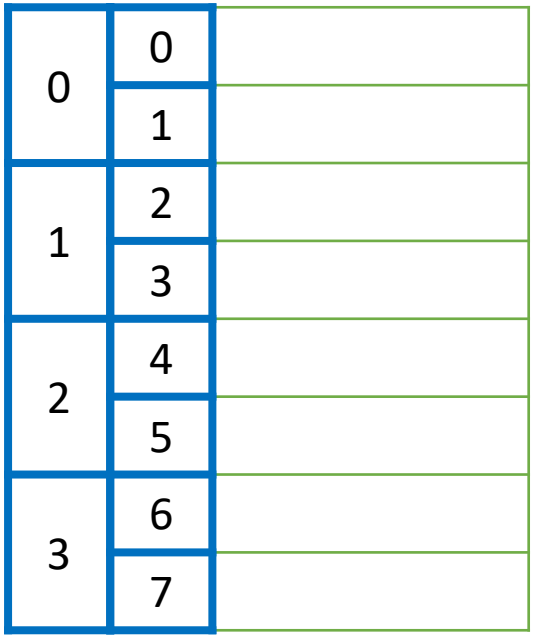
- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes



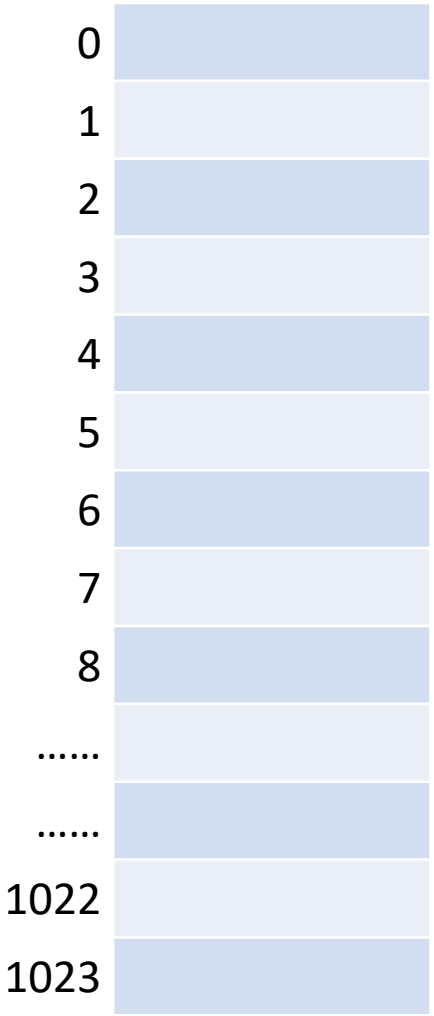
Direct Mapped Cache Index



- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes



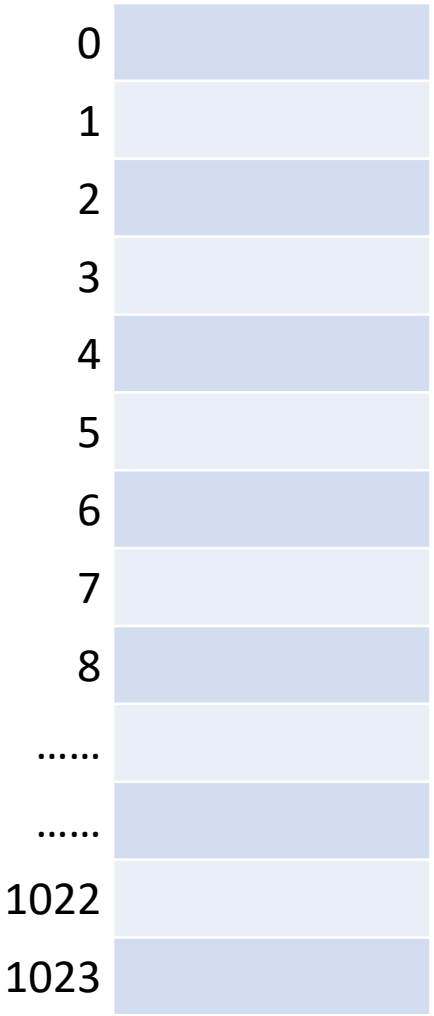
Direct Mapped Cache Offset



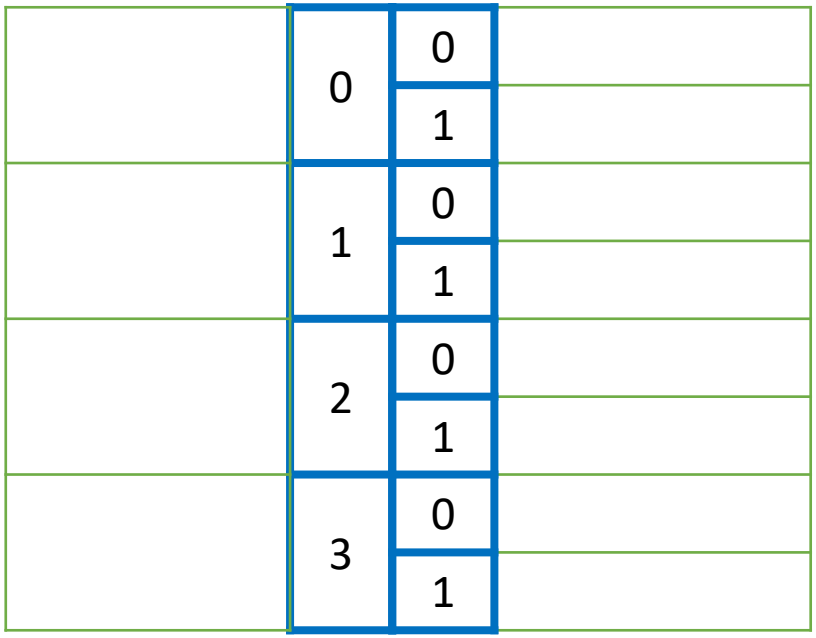
- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

0	0	
	1	
1	0	
	1	
2	0	
	1	
3	0	
	1	

Direct Mapped Cache Tag



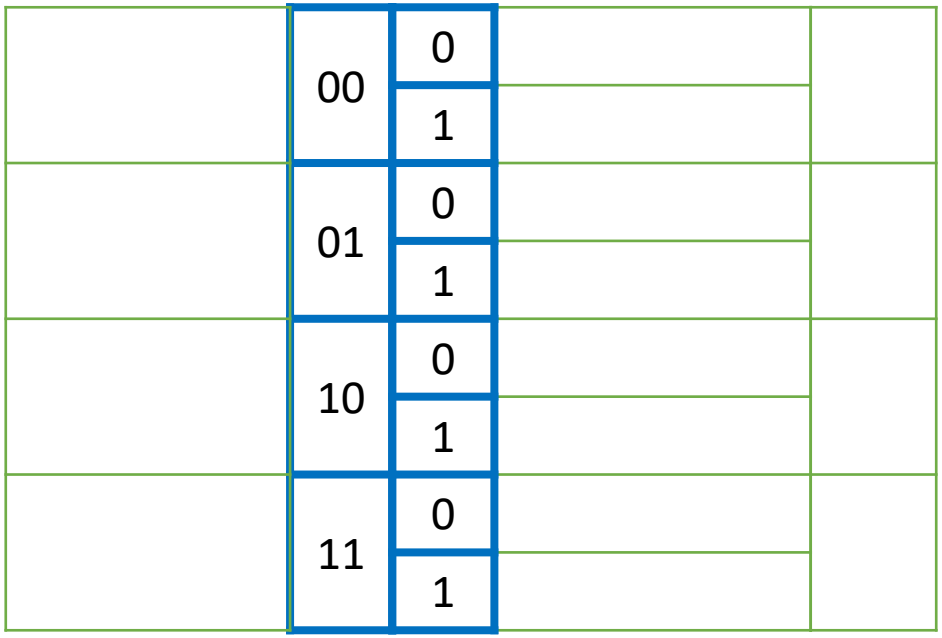
- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes



Direct Mapped Cache Tag

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

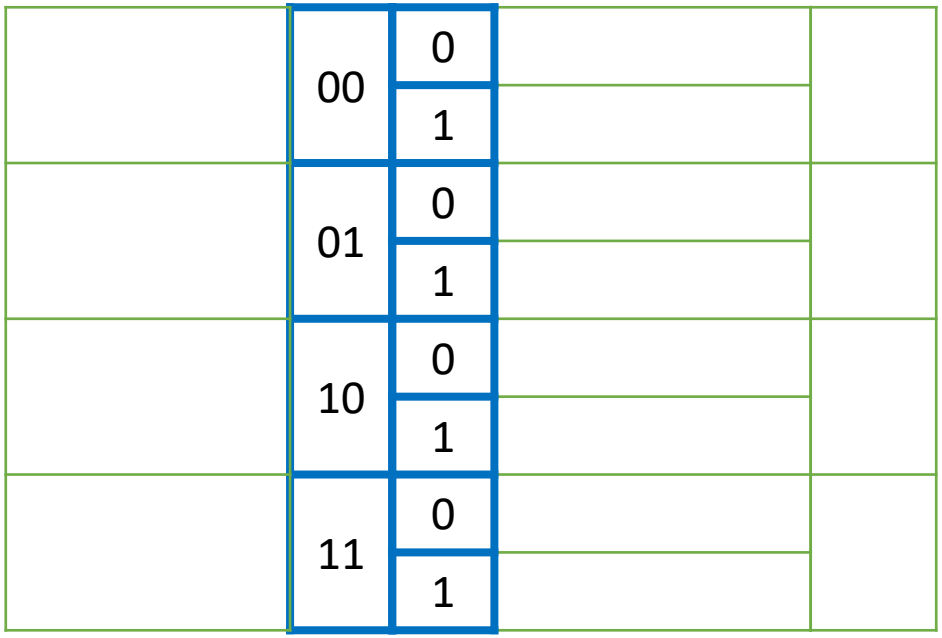
- Byte addressable main memory
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes



Direct Mapped Cache Tag

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

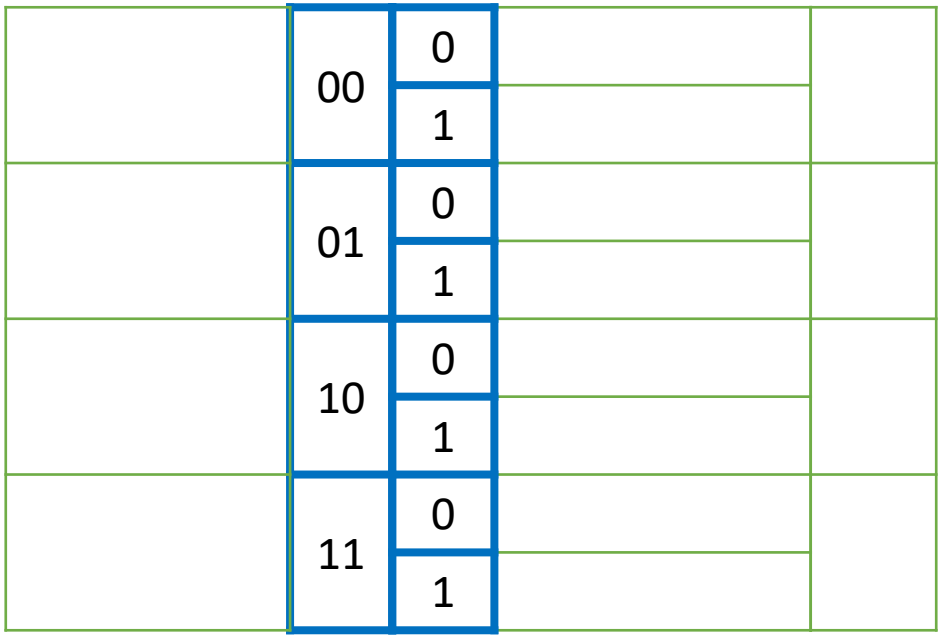
- Minimum number of bits required for
  - Offset
  - Index
  - Tag



Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

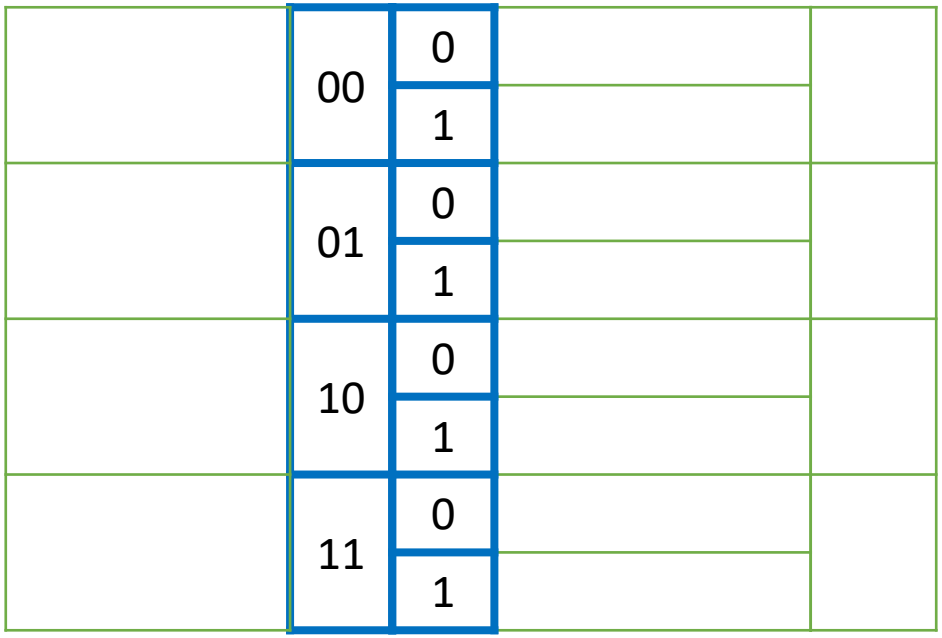
- CPU wants to read address 1
- 1 = 0000000 00 1



Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 1
- 1 = 0000000 00 1



Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 1
- 1 = 0000000 00 1

0000000	00	0	a	1
		1	b	
	01	0		
		1		
	10	0		
		1		
	11	0		
		1		

Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 4
- 4 = 0000000 10 0

0000000	00	0	a	1
		1	b	
	01	0		
		1		
0000000	10	0	e	1
		1	f	
	11	0		
		1		

Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 8
- 8 = 0000001 00 0

0000000	00	0	a	1
		1	b	
	01	0		
		1		
0000000	10	0	e	1
		1	f	
	11	0		
		1		

Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 8
- 8 = 0000001 00 0

0000000	00	0	a	1
		1	b	
	01	0		
		1		
0000000	10	0	e	1
		1	f	
	11	0		
		1		

Direct Mapped Cache – Example

a	0000000 00 0
b	0000000 00 1
c	0000000 01 0
d	0000000 01 1
e	0000000 10 0
f	0000000 10 1
g	0000000 11 0
h	0000000 11 1
i	0000001 00 0
j	0000001 00 1
…	…….
yyy	1111111 11 0
zzz	1111111 11 1

- CPU wants to read address 8
- 8 = 0000001 00 0

0000001	00	0	i	1
		1	j	
	01	0		
		1		
0000000	10	0	e	1
		1	f	
	11	0		
		1		

Direct Mapped Cache - Searching

- Use the Low Order Address Bits (LOAB) of an address to index into the Cache Tag Memory.
- Use comparator to find whether the corresponding Tag matches with Higher Order Address Bits of the address.
- If the Tag matches then check for the valid bit.
  - Valid Bit =1 then Cache Hit
  - Valid Bit =0 then Cache Miss
- If the Tag not found then it is Cache Miss

Exercise

- 2 MB Direct-Mapped cache
- Cache line size is 32 KB
- Byte addressable 1GB memory

1. What are lengths of tag, index and offset?

Exercise

- 2 MB Direct-Mapped cache
- Cache line size is 32 KB
- Word addressable 1GB memory
- Word size is 32 bits

1. What are lengths of tag, index and offset?

Exercise

- 2 MB Direct-Mapped cache
- Cache line size is 32 KB
- Word addressable 1GB memory
- Word size is 32 bits

- Cache Line = 32KB
- Word = Smallest Unit which has an address
- Offset = How many addressable units within a cache line?

$$\frac{32KB}{32\ bits} = \frac{32 \times 2^{10} Bytes}{4\ Bytes} = \frac{2^{15} Bytes}{2^2\ Bytes}$$

$$2^{13}$$

- How many addressable units within a cache line? 2^{13}
- Therfore, we need at least 13 bits for Offset

Exercise

- 2 MB Direct-Mapped cache
- Cache line size is 32 KB
- Word addressable 1GB memory
- Word size is 32 bits

- Cache Line = 32KB
- Index = How many cache entries can we create ?

$$\frac{2MB}{32\ KB} = \frac{2\times 2^{20} Bytes}{2^5 \times 2^{10}\ Bytes} = \frac{2^{21} Bytes}{2^{15}\ Bytes}$$

$$2^6$$

- How many cache entries can we create? 2^6
- Therfore, we need at least 6 bits for Index

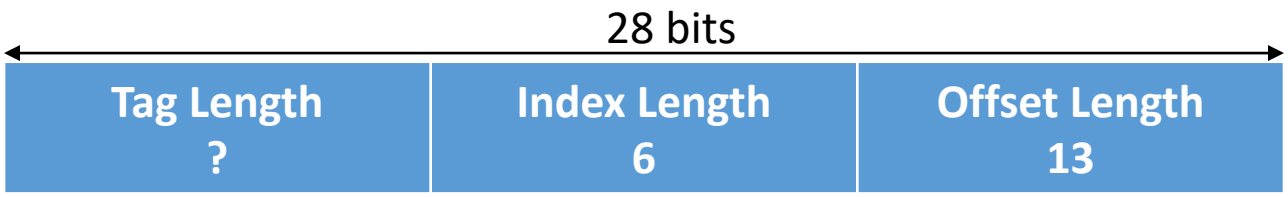
Exercise

- 2 MB Direct-Mapped cache
- Cache line size is 32 KB
- Word addressable 1GB memory
- Word size is 32 bits

- Minimum address length ??
- Main memory = 1GB
- Word size = 32 bits = 4 Bytes
- How many addresses do we have ?

$$\frac{1GB}{32\ bits} = \frac{2^{30}Bytes}{4\ Bytes} = 2^{28}$$

- Therfore, we need at least 28 bits for address



Homework..

- Find advantages and disadvantages of Direct Mapped Cache

Next...

Cache Mapping Schemes

- Set Associative Cache

Computer Systems

Kasun Gunawardana

E-mail: kgg

University of Colombo School of Computing

Set Associative Cache

Fully Associative Cache

- Direct Mapped Cache affirms strict mapping rule.
- Fully Associative Cache is the opposite.
  - No restrictions.
  - Cache lines can be placed on anywhere available.
- More flexible cache mapping scheme.
- Searching is expensive.

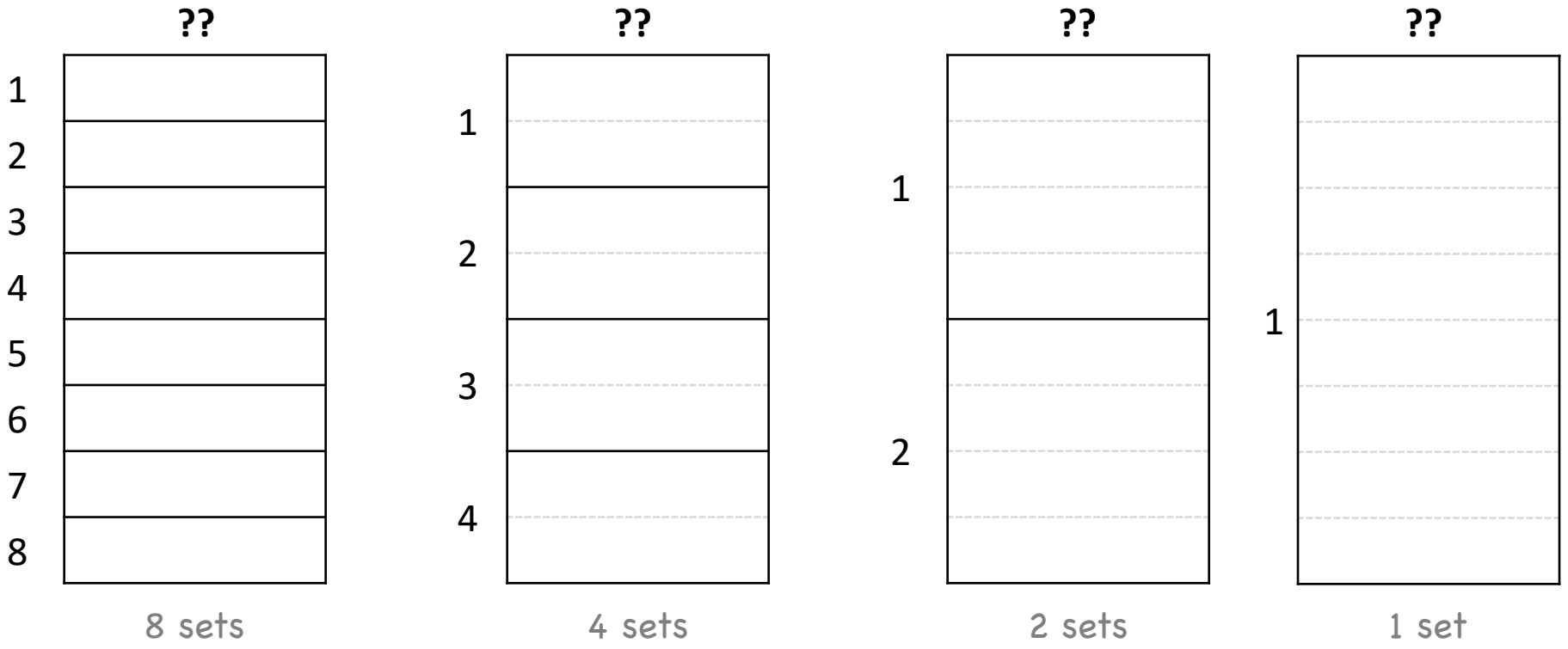
Set Associative Cache

- Hybrid model of Fully Associative and Direct Mapped caches.
- Cache is divided into equal sized sets.
- Cache line can be mapped into only one set (Directly mapped to a set).
- Within the mapped set, cache line can be placed anywhere available (Fully Associative within a set).

If a set in the cache can accommodate N number of cache lines, then it is called "N-Way Set Associative Cache".

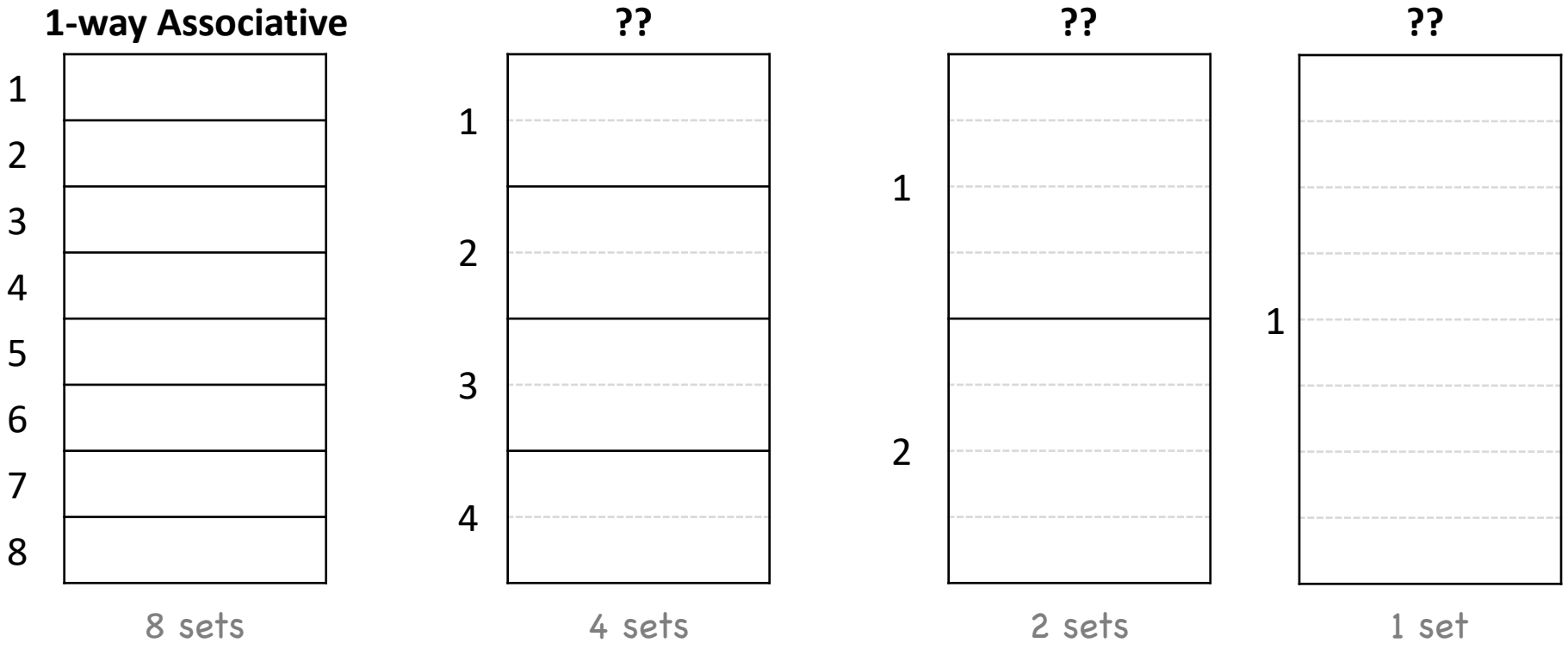
N-Way Set Associative

- If  is a cache line and segmentation shows the sets, recognize N in following cache arrangements.



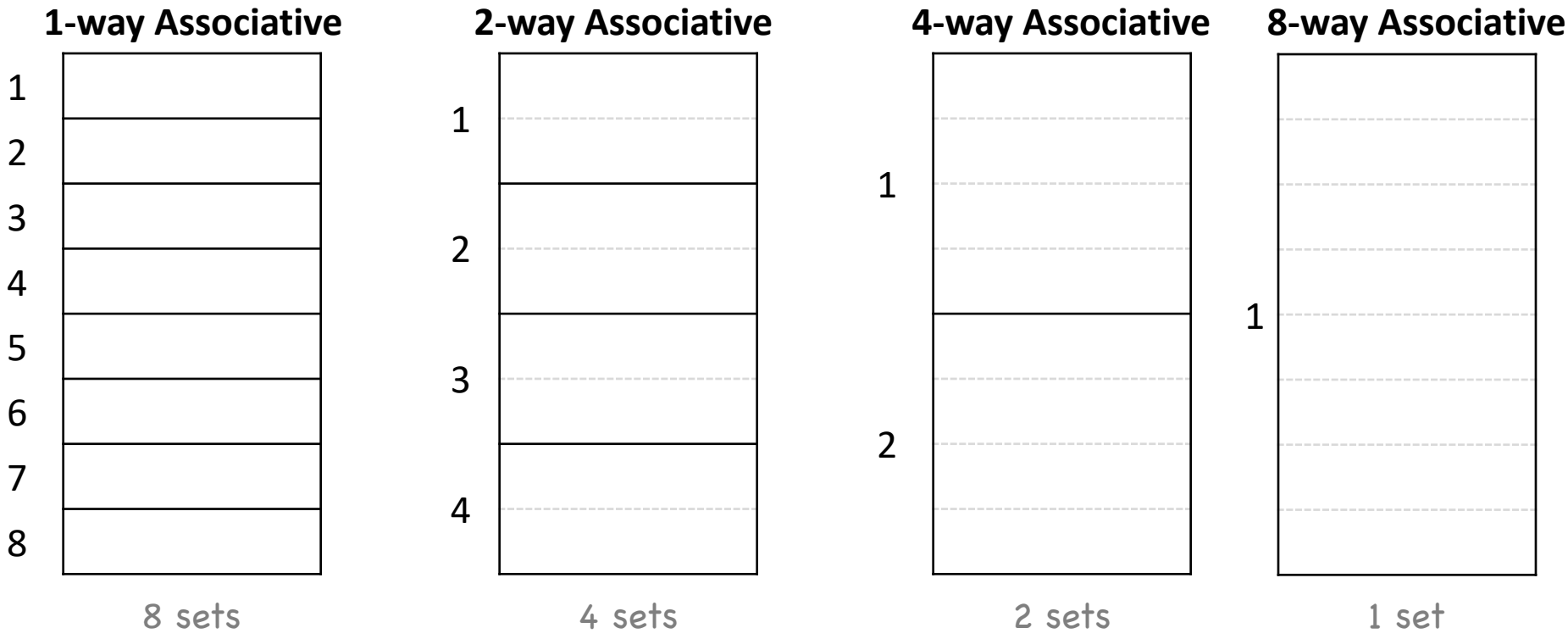
N-Way Set Associative

- If  is a cache line and segmentation shows the sets, recognize N in following cache arrangements.



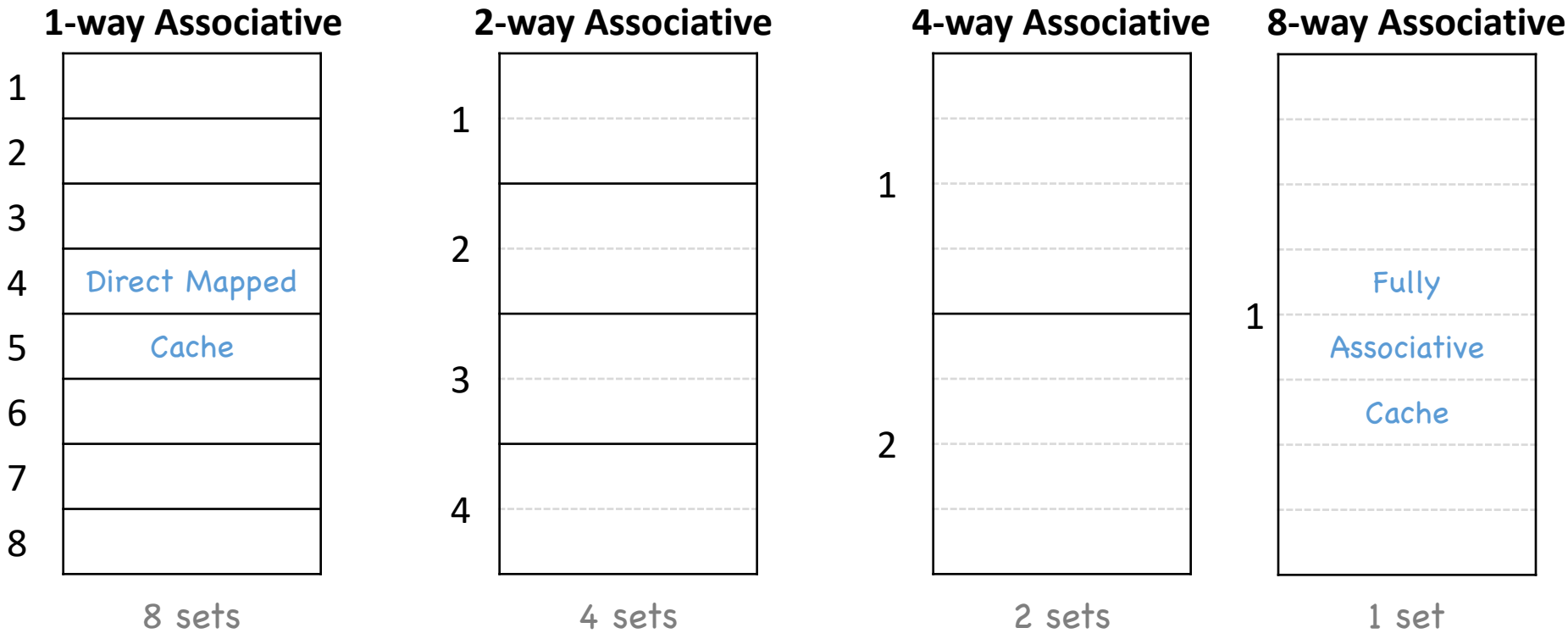
N-Way Set Associative

- If  is a cache line and segmentation shows the sets, recognize N in following cache arrangements.

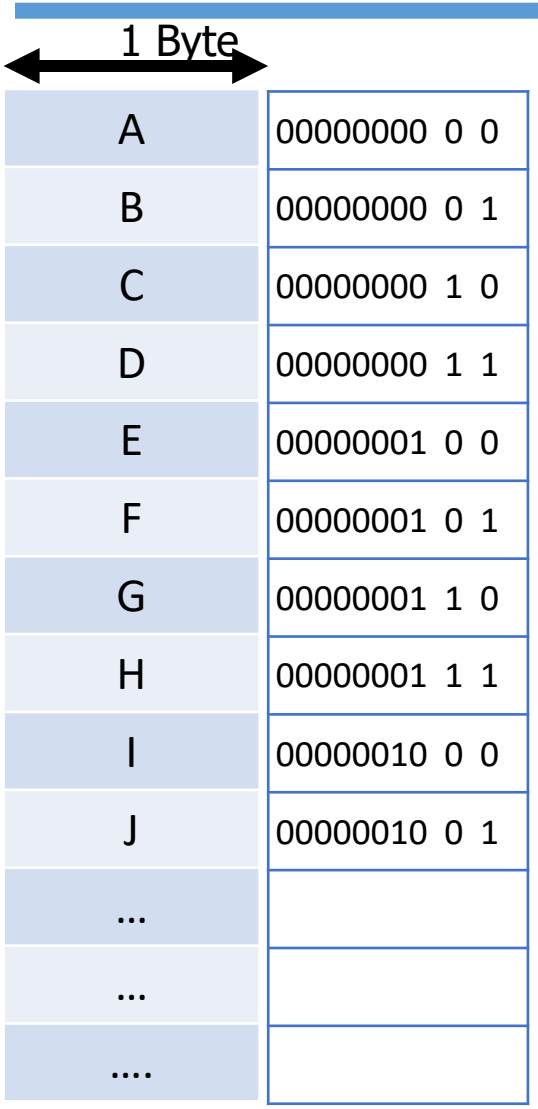


N-Way Set Associative

- If  is a cache line and segmentation shows the sets, recognize N in following cache arrangements.



Example: 2-Way Set Associative



- Byte addressable 2-way set associative cache
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

- How do we arrange the cache layout?

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

- 2-Way set associative – Within a single set there will be room for two cache lines.
- Thus the set size = 4 Bytes
- Since the cache volume is 8 Bytes, there will be only two sets.
- A cache line placed in a set is identified by a Tag
- Within a cache entry, each unit of memory is identified by the offset

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

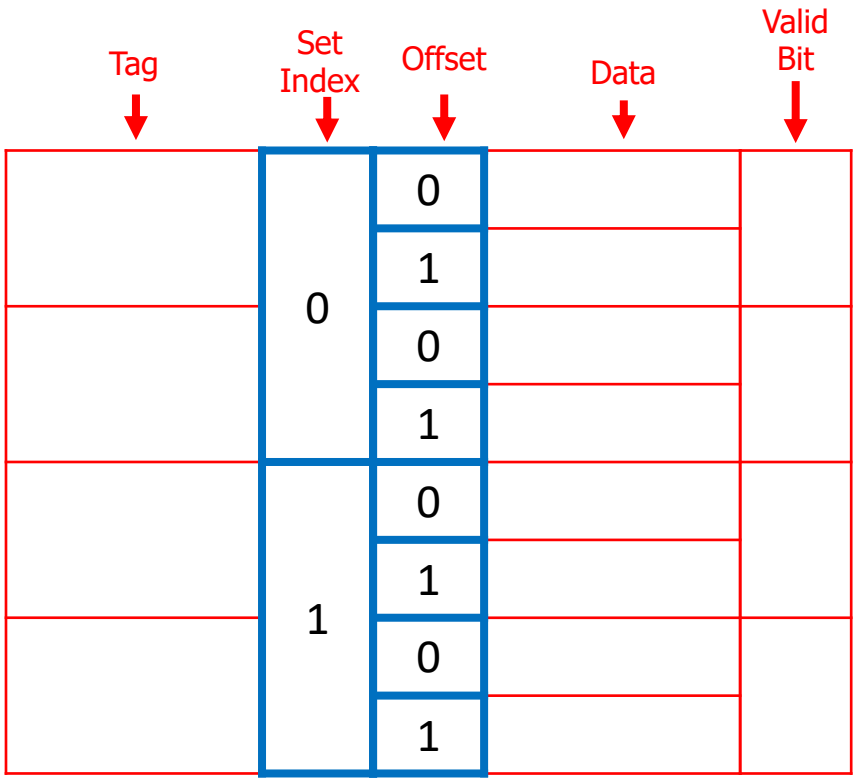
A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

- A cache line will be mapped to one of two sets.
- We need set identifier
- Within mapped set, there will be cache entries.
- Tag is required to identify the address.
- Offset will be used to refer the exact memory unit within the cache entry.

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

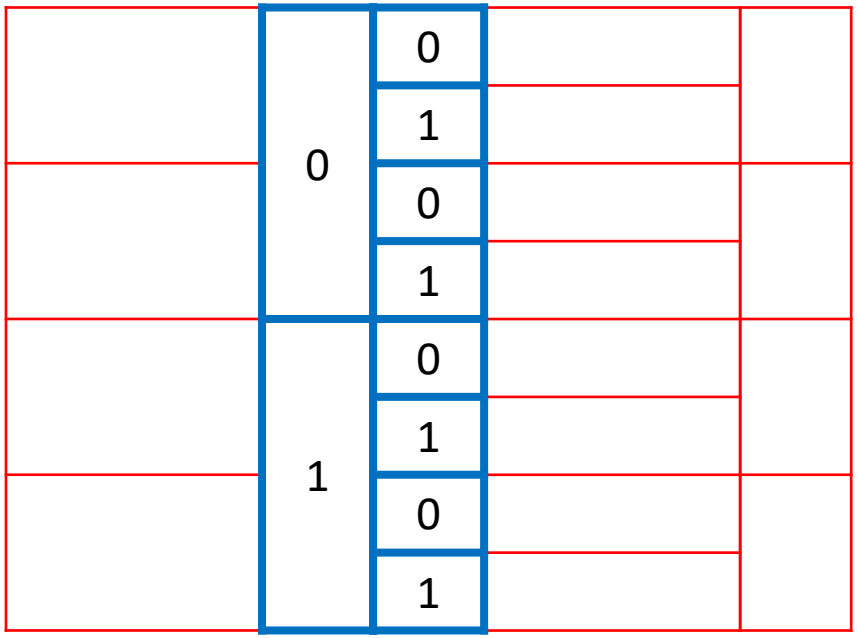


Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

You may remember, DMC made cache replacement when the CPU referred addresses 1 and 8 in order. What will happen here if the CPU refers the same?

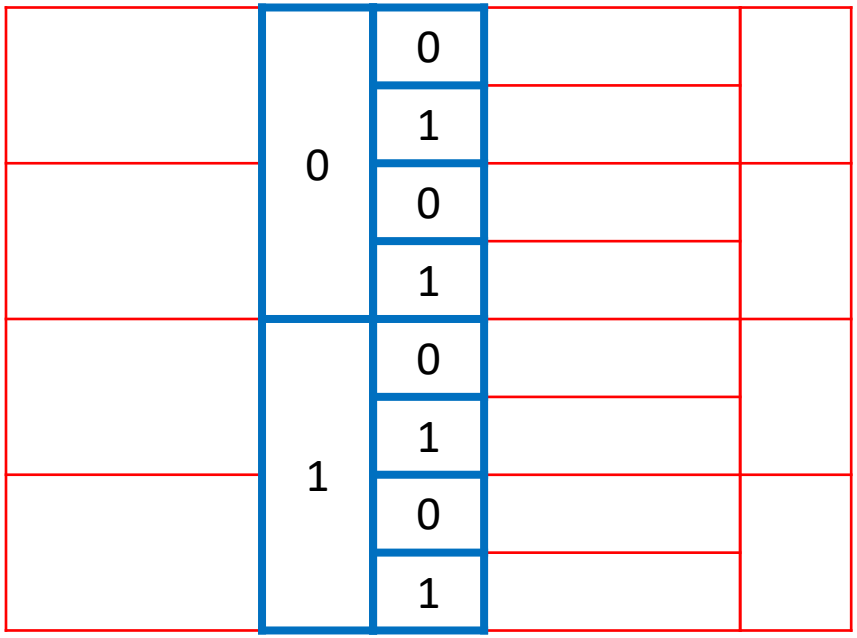


Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

CPU referrs address 1



Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

CPU referrs address 1

- Mapping set is 0
- Can place anywhere within the set 0 (Assuming the cache is empty at the beginning)

00000000	0	0	A	1
		1	B	
		0		
		1		
	1	0		
		1		
		0		
		1		

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

CPU referrs address 8

- Mapping set is 0
- Still it can be placed. Because set 0 can accommodate another cacheline.

00000000	0	0	A	1
		1	B	
		0		
		1		
	1	0		
		1		
		0		
		1		

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

CPU referrs address 8

- Mapping set is 0
- Still it can be placed. Because set 0 can accommodate another cacheline.

00000000	0	0	A	1
		1	B	
00000010		0	I	1
		1	J	
	1	0		
		1		
		0		
		1		

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

But, if CPU referrs address 4 or 5 there will be an eviction.

- Mapping set is 0
- One cache entry must be replaced with new cacheline

00000000	0	0	A	1
		1	B	
00000010		0	I	1
		1	J	
	1	0		
		1		
		0		
		1		

In this circumstance, which cache entry should be evicted ?

Example:

- Byte addressable 2-way set assoc.
- Total of 1024 Bytes in the memory
- 2 Bytes per cache line
- Cache volume = 8 Bytes

A	00000000 0 0
B	00000000 0 1
C	00000000 1 0
D	00000000 1 1
E	00000001 0 0
F	00000001 0 1
G	00000001 1 0
H	00000001 1 1
I	00000010 0 0
J	00000010 0 1
…	
…	
….	

But, if CPU referrs address 4 or 5 there will be an eviction.

- Mapping set is 0
- One cache entry must be replaced with new cacheline

00000000	0	0	A	1
		1	B	
00000010		0	I	1
		1	J	
	1	0		
		1		
		0		
		1		

In this circumstance, which cache entry should be evicted ?

Cache Replacement Policies

- When implementing set associative caches, we need to affirm a cache replacement policy.
  - Which cache entry should be evicted when the set is fully occupied?
- This decision should be taken aligned with the Locality of reference that we need to exploit.

Common Policies

- Least Recently Used
  - The cache entry which is unused for the longest time will be evicted.
- First In First Out
  - The cache entry which has the longest period of stay will be evicted.
- Least Frequently Used
  - The cache line with least access frequency will be evicted.
- Random

Homework…

- Discuss advantages and disadvantages of those cache replacement policies.

Exam

- Try to answer past paper questions.
- Discuss doubts and clarify with your colleagues.
- Exam Paper – 2 Hour paper –
  - 4 compulsory questions
  - A question from each section of the course

Thank You..!