Microprocessor & Computer Architecture (μpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

Microprocessor & Computer Architecture (μpCA)

Unit 3: Mapping Functions

Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Syllabus

~~**Unit 1: Basic Processor Architecture and Design**~~

~~**Unit 2: Pipelined Processor and Design**~~

Unit 3: Memory

- ~~Memory Hierarchy~~
- ~~Principles of Locality~~
- ~~Cache Design Principles~~
- Mapping Functions

Unit 4: Input/Output Device Design

Unit 5: Advanced Architecture

Where can a block be placed in the cache?

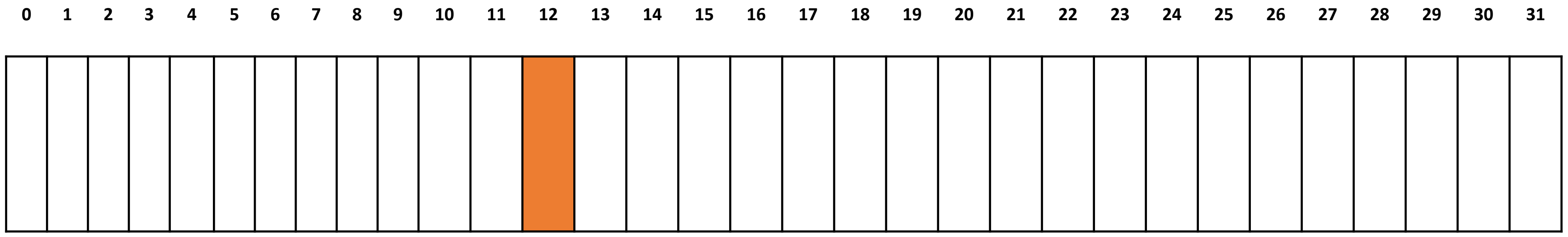
- **Three mapping functions:**
  - **Direct mapping**
  - **Fully Associative mapping**
  - **Set-associative mapping.**

- If a block of memory from the main memory can be placed in exactly one place, we have a cache which is **direct-mapped**

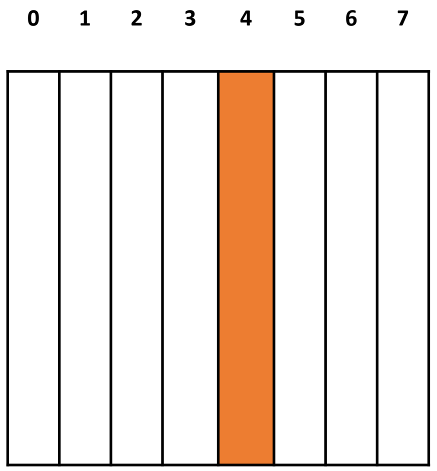
- If the block can be placed anywhere, the cache is **fully associative**

- If there are a restricted set of places that the block can be placed, the cache is **set associative**

Three mapping functions:

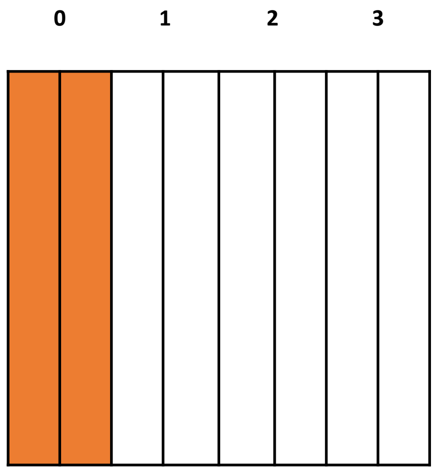


Direct mapping



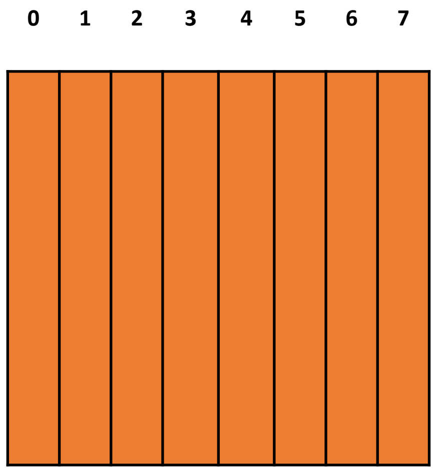
12%8=4

2-Set-associative mapping.



12%4=0

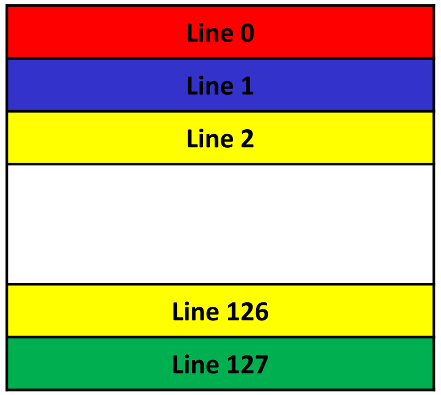
Fully Associative mapping



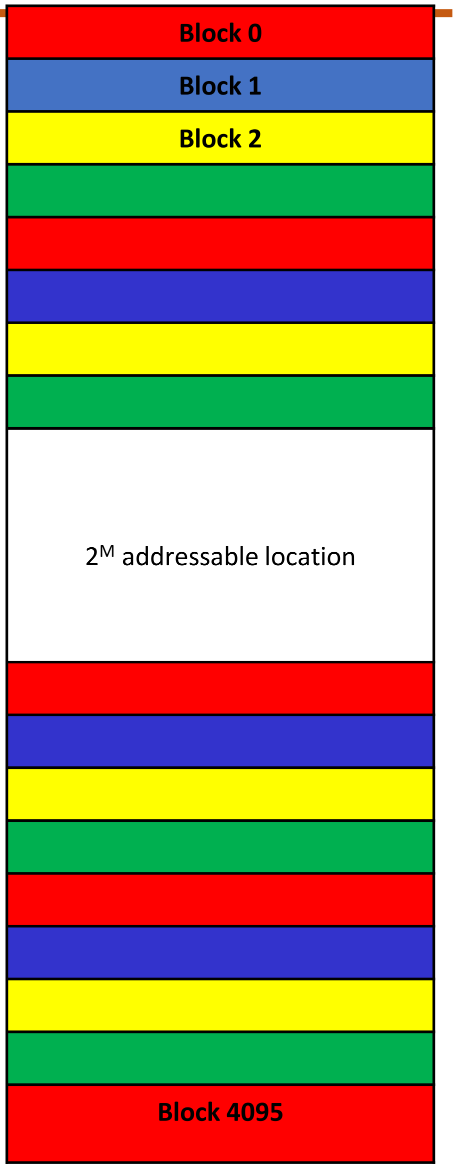
Microprocessor & Computer Architecture (μpCA)

Direct Mapping

- **A simple processor example:**
  - Main memory is addressable by a 16-bit address.
  - Main memory has 65536 (64 k) words.
  - Main memory has 4096 Blocks of 16 words each.
  - Consecutive addresses refer to consecutive words.
  - Cache consisting of 128 Lines of 16 words each.
  - Total size of cache is 2048 (4 K) words.
  - 4096/128= 32 possible Blocks for one Line



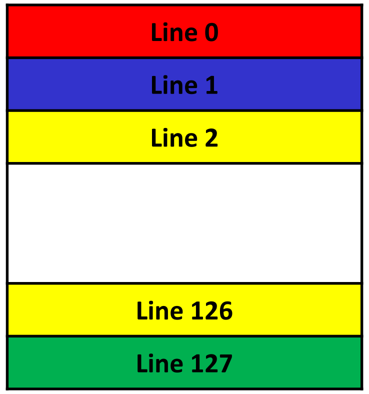
128 Block Cache



Microprocessor & Computer Architecture (μpCA)

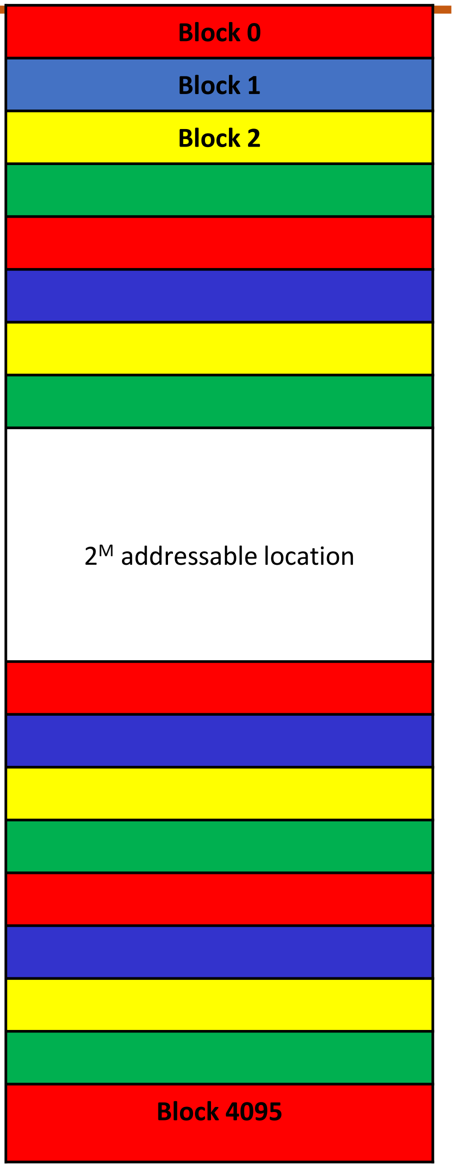
Direct Mapping: Block Placement

- Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.



15018 and 10922 will be mapped to 42nd Line

16 bit Address:
0011101010101010 = 15018
0010101010101010 = 10922



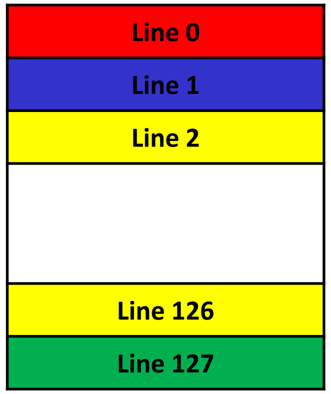
Microprocessor & Computer Architecture (μpCA)

Direct Mapping: Block Identification

Memory address is divided into three fields:

- Low order 4 bits determine one of the 16 words in a block.
- When a new block is brought into the cache, the next 7 bits determine which cache block this new block is placed in.
- High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.

• Simple to implement but not very flexible.



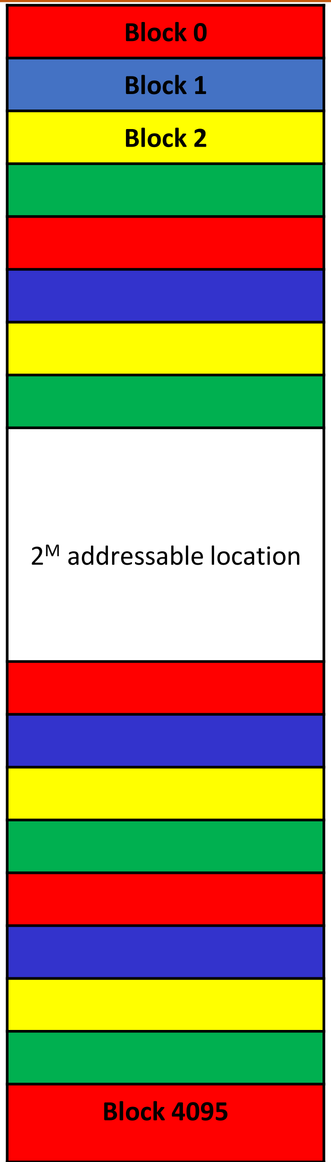
16 bit Address

Tag	Block / Line	Word
5 bits	7 bits	4 bits

2^4 =16 words

2^7 =128 Line

2^5 = Which block out of 32 Blocks



Direct Mapping: Block Identification HIT

16 bit Address: 00111 0101010 1010 = 15018
7th Block of 32 Blocks 42nd Line 10th Word

16 bit Address: 00101 0101010 1010 = 10922
5th Block of 32 Blocks 42nd Line 10th Word

Cache Hit or Miss

If Processor is looking for data in the address: 0011101010101010

Step 1: Look at Line Number **42 i.e 0101010**
Step 2: Compare the TAG field.
- **2.1** If it Matches, it is a *HIT* Fetch the 10th Word
- **2.2** If not Matched it is a *MISS.*
  Replace current Line with the block which contain the word which the Processor is looking for.

Line No	Tag	Word	Data
Line 0			
Line 1			
Line 42 0101010		0000	
		0001	
		0010	
		0011	
		0100	
		0101	
		0110	
		0111	
		1000	
		1001	
	00111	1010	
		1011	
		1100	
		1101	
		1110	
		1111	
Line 126			
Line 127			

Direct Mapping: Block Identification MISS

16 bit Address: **00111** **0101010** 1010 = 15018
7th Block of 32 Blocks **42nd Line** 10th Word

16 bit Address: **00101** **0101010** 1010 = 10922
5th Block of 32 Blocks **42nd Line** 10th Word

Cache Hit or Miss

If Processor is looking for data in the address: **0011101010101010**

Step 1: Look at Line Number **42 i.e 0101010**

Step 2: Compare the TAG field.

2.1 If it Matches, it is a *HIT* Fetch the 10th Word

2.2 If not Matched it is a *MISS.*

Replace current Line with the block which contain the word which is Processor is looking for.

Line No	Tag	Word	Data
Line 0			
Line 1			
Line 42 0101010		0000	
		0001	
		0010	
		0011	
		0100	
		0101	
		0110	
		0111	
		1000	
		1001	
	00101	1010	
		1011	
		1100	
		1101	
		1110	
		1111	
Line 126			
Line 127			

Microprocessor & Computer Architecture (μpCA)

Exercise 1

Consider a direct mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find Number of bits in tag

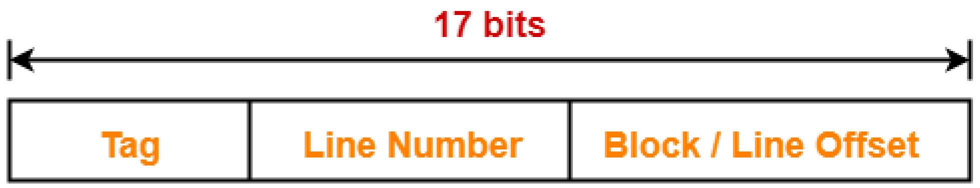
Given-

Cache memory size = 16 KB

Block size = Frame size = Line size = 256 bytes

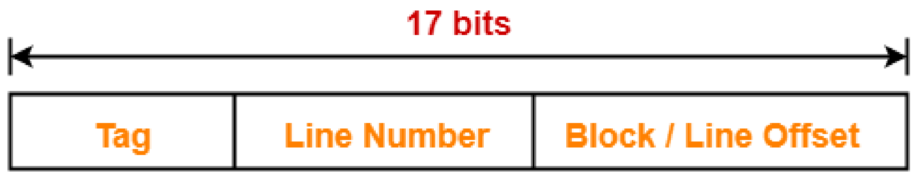
Main memory size = 128 KB = 2^17

Size of Memory Address: 17 bits

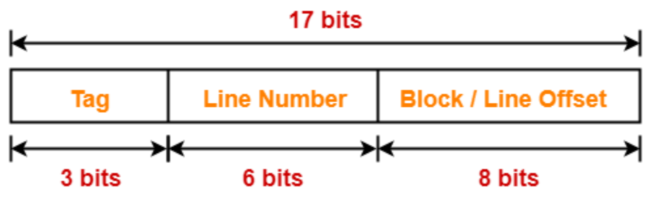


Microprocessor & Computer Architecture (μpCA)

Exercise 1



TAG	Line Number	Block / Line Offset
17-(8+6)= 3 bits	Total number of Blocks or Lines = Cache size / Block size = 16kb/256 = 2^14 bytes / 2^8 bytes = 2^6 Blocks or Lines **Line Number =6 bits**	Block Size = 256 = 2^8 **Block Offset = 8 bits**



Click here for Extra Exercise

Microprocessor & Computer Architecture (μpCA)

Exercise 2

A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a direct-mapped manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) When a program is executed, the processor reads data sequentially from the following word addresses: **128, 144, 2176, 2180, 128, 2176**

All the above addresses are shown in decimal values. Assume that the cache is initially empty.

For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

Microprocessor & Computer Architecture (μpCA)

Exercise 2

Block size = 64 bytes = 2^6 bytes

Therefore, **Number of bits in the *Word* field = 6**

Cache size = 2K-byte = 2^{11} bytes

Number of cache blocks = Cache size / Block size = $2^{11}/2^6 = 2^5$

Therefore, **Number of bits in the *Block* field = 5**

Total number of address bits = 16

Therefore, **Number of bits in the *Tag* field = 16 - 6 - 5 = 5**

For a given 16-bit address, the 5 most significant bits, represent the *Tag*, the next 5 bits represent the *Block*, and the 6 least significant bits represent the *Word*.

Microprocessor & Computer Architecture (μpCA)

Exercise 2

Address = $(128)_{10}$ = $(\mathbf{00000}00010000000)_2$

Address = $(144)_{10}$ = $(\mathbf{00000}00010010000)_2$

Address = $(2176)_{10}$ = $(\mathbf{0000100010}000000)_2$

Address = $(2180)_{10}$ = $(\mathbf{0000100010}000100)_2$

Microprocessor & Computer Architecture (μpCA)

Exercise 2

Access # 1: The cache is initially empty. Therefore, all the cache blocks are invalid

Address = $(128)_{10}$ = $(0000000010000000)_2$

For this address, *Tag* = 00000, *Block* = 00010, *Word* = 000000

Since the cache is empty before this access, this will be a cache **miss**

After this access, **Tag field for cache block 00010 is set to 00000**

Access # 2: Address = $(144)_{10}$ = $(0000000010010000)_2$

For this address, *Tag* = **00000**, *Block* = **00010**, *Word* = 010000

Since tag field for cache block **00010** is **00000** before this access, this will be a cache **hit** (because address tag = block tag)

Microprocessor & Computer Architecture (μpCA)

Exercise 2

Access # 3: Address = $(2176)_{10}$ = $(0000100010000000)_2$
For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000000
Since tag field for cache block **00010** is **00000** before this access, this will be a cache **miss** (address tag ≠ block tag)
After this access, **Tag field for cache block 00010 is set to 00001**

Access # 4:
Address = $(2180)_{10}$ = $(0000100010000100)_2$
For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000100
Since tag field for cache block **00010** is **00001** before this access, this will be a cache **hit** (address tag = block tag)

Microprocessor & Computer Architecture (μpCA)

Exercise 2

Access # 5:

Address = $(128)_{10}$ = $(0000000010000000)_2$

For this address, *Tag* = **00000**, *Block* = **00010**, *Word* = 000000

Since tag field for cache block **00010** is **00001** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 00010 is set to 00000**

Access # 6:

Address = $(2176)_{10}$ = $(0000100010000000)_2$

For this address, *Tag* = **00001**, *Block* = **00010**, *Word* = 000000

Since tag field for cache block **00010** is **00000** before this access, this will be a cache **miss** (address tag ≠ block tag)

After this access, **Tag field for cache block 00010 is set to 00001**

Microprocessor & Computer Architecture (μpCA)

Exercise 2

$$
\begin{aligned}
\text{Cache hit rate} &= \text{Number of hits / Number of accesses} \\
&= 2/6 \\
&= 0.333
\end{aligned}
$$

Microprocessor & Computer Architecture (μpCA)

Next Session

Set Associative Mapping

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135