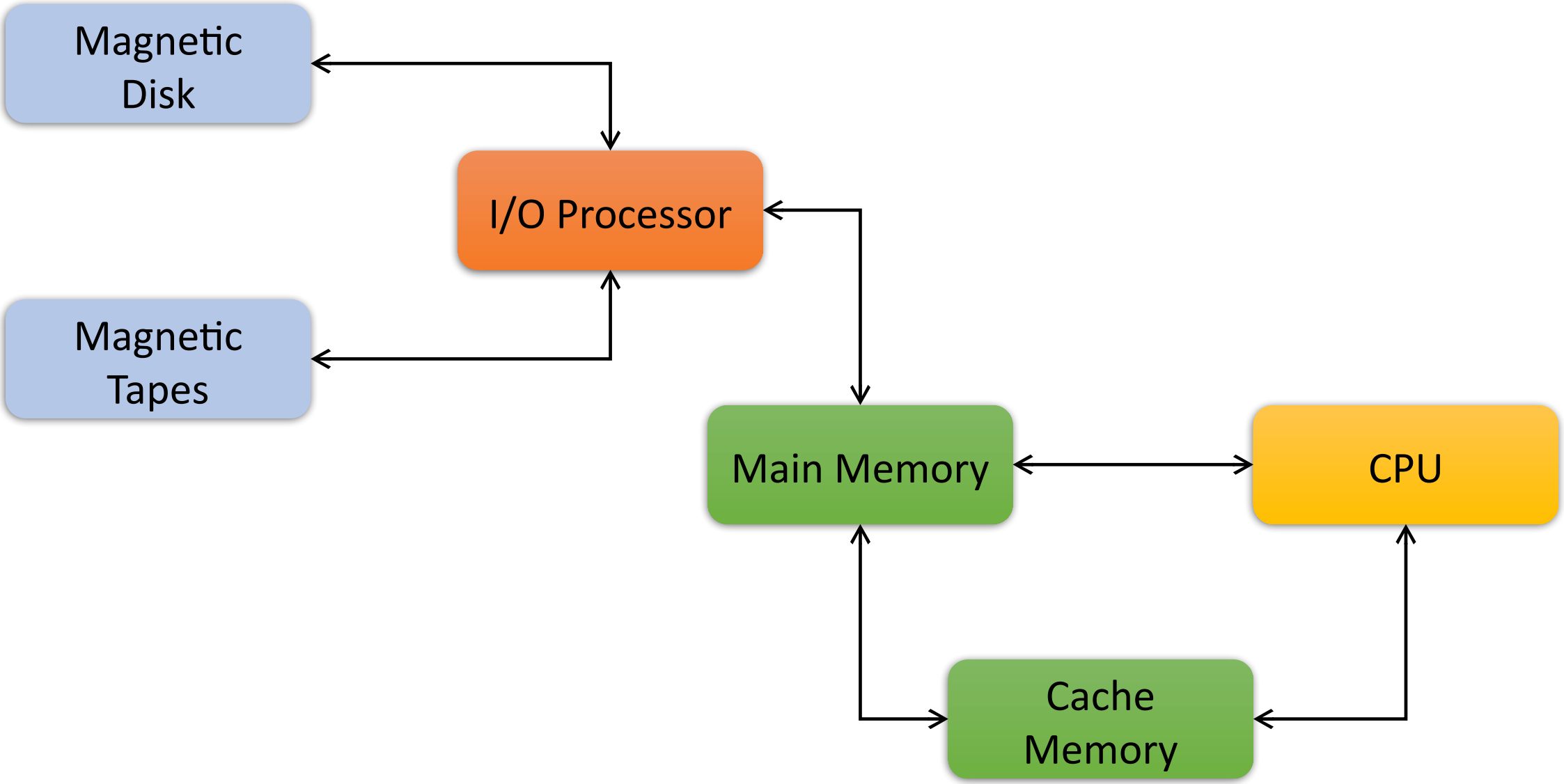
Unit 7: Memory Organization

Reference : Chapter 12 Computer System Architecture by Morris Mano

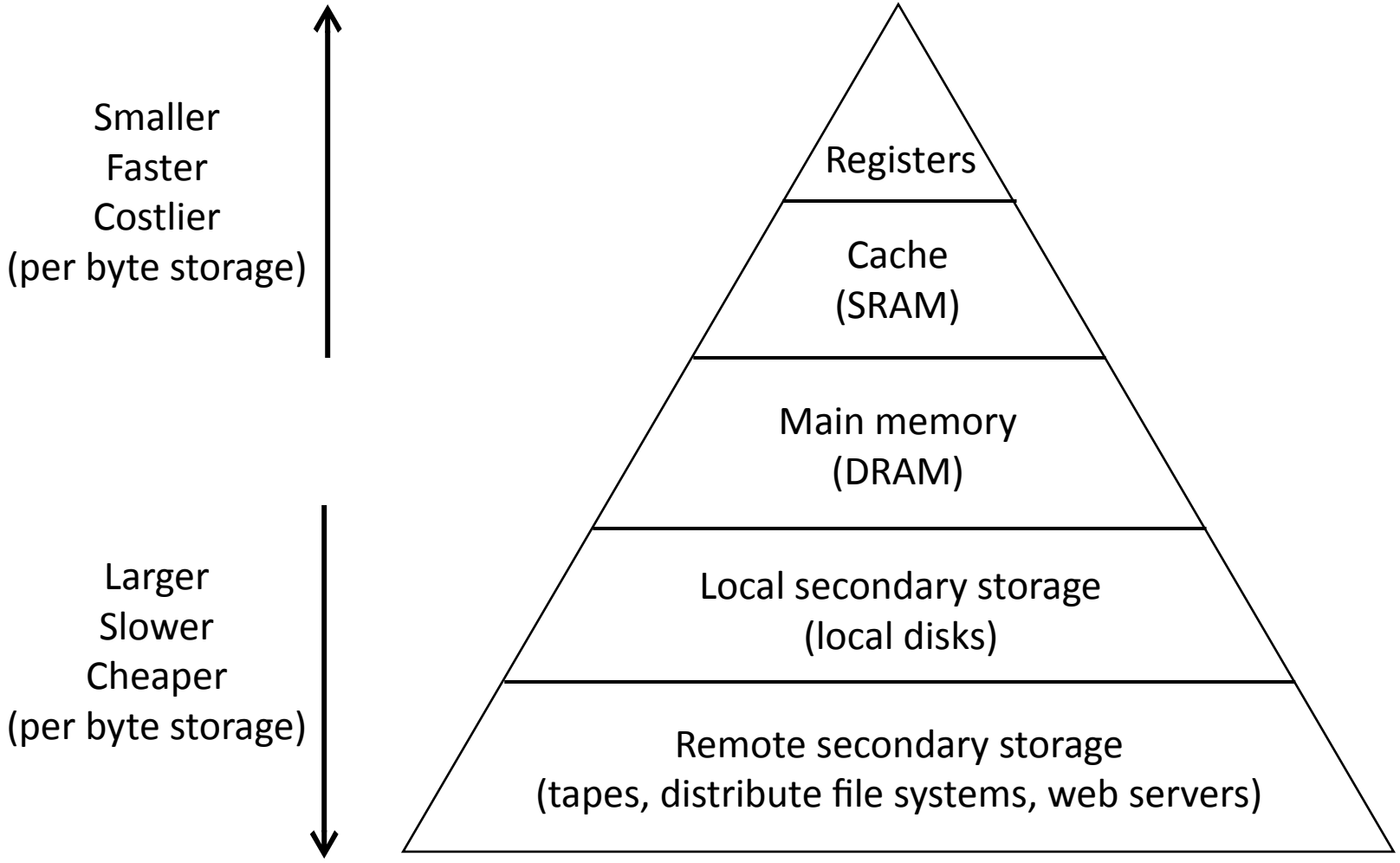
Memory Hierarchy

- What is Memory?
- Why different types of memory? (Main Memory, Auxiliary Memory)
    - Volatile, Large Storage, Speed
- Main Memory: memory unit that communicates directly with the CPU
    - RAM, ROM
- Auxiliary Memory: provide backup storage
    - Magnetic disk

Memory Organization and Relation



Memory Hierarchy



Multiprogramming

- Concept used in OS which enable the CPU to Process a number of independent programs concurrently.

- Keep CPU busy by working with several programs in sequence.

Main Memory

- Relatively large and fast memory used to store programs and data during the computer operation.

- RAM(Random Access Memory)
  - Static and Dynamic

- ROM(Read Only Memory)

RAM (Random Access Memory)

- Volatile
- Can perform Read and Write operation

1. SRAM (Static RAM)
   - consist of internal flip-flops that store the binary information.
     - ✓ Cache Memory

2. DRAM(Dynamic RAM)
   - Stores the Binary information in the form of electric charges that are applied to capacitors
   - Refreshing is required
   - Reduced power consumption and Larger storage capacity

ROM (Read Only Memory)

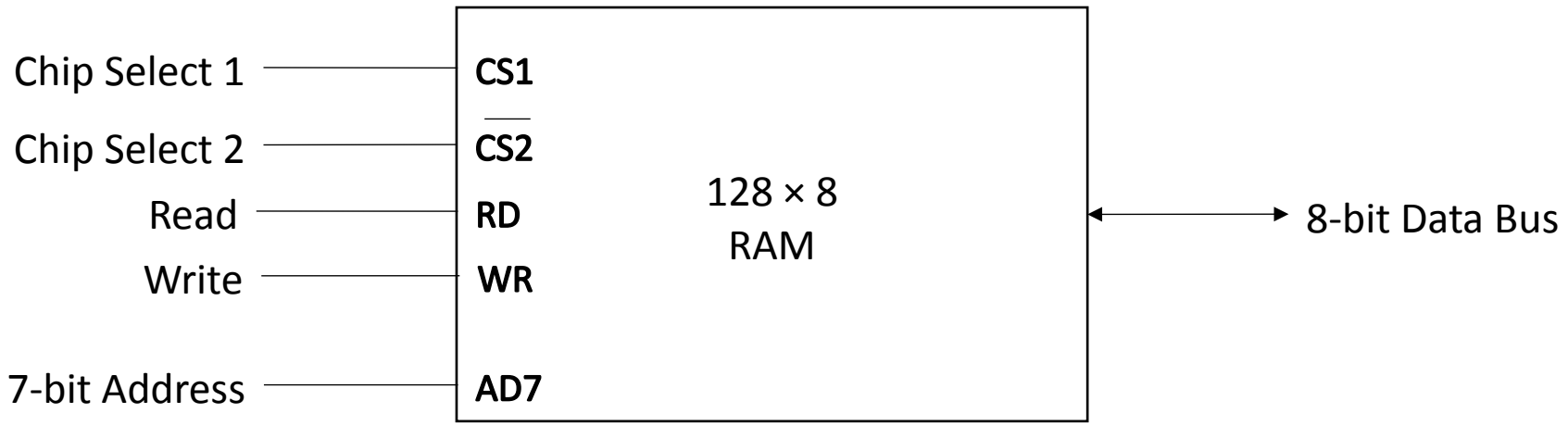
- Non volatile
- Can perform Read operation only
- Store programs and data that are permanently reside in the computer that do not change in value.
  - Store Bootstrap Loader

Bootstrap Loader

- It is a program whose function is to start the computer software operating when power is turned on.
- When power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.
- The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the OS, which prepares the computer for general use.

RAM Chip

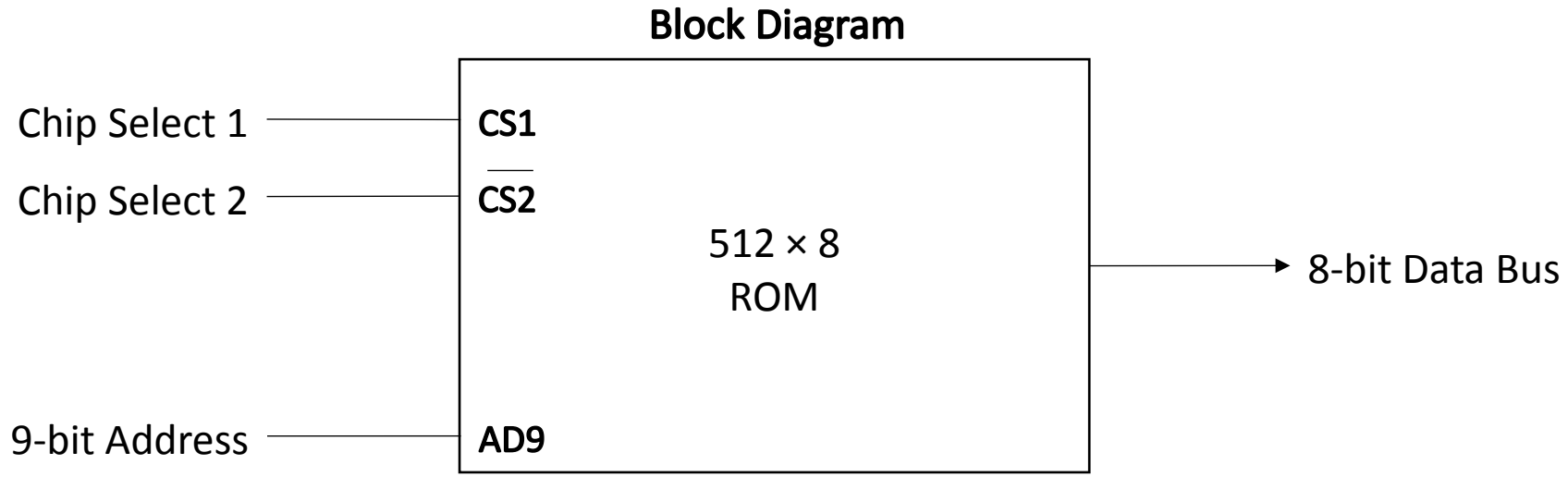
Block Diagram



Function Table

CS1	$\overline{CS2}$	RD	WR	Memory Function	State of the Data Bus
0	0	×	×	Inhibit	High Impedance
0	1	×	×	Inhibit	High Impedance
1	1	×	×	Inhibit	High Impedance
1	0	0	0	Inhibit	High Impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM

ROM Chip

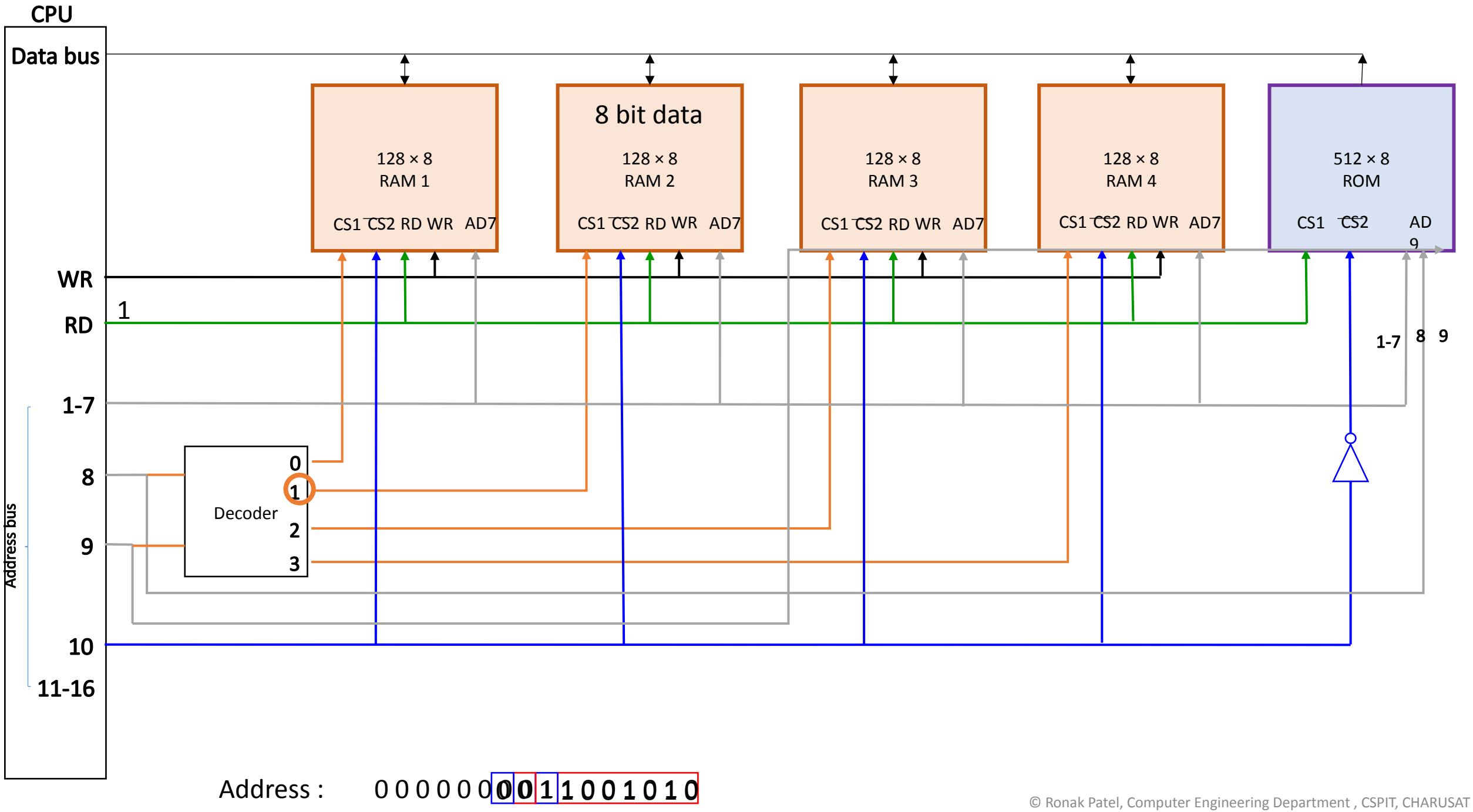


Memory Connection to the CPU

- Require Memory: 512 Bytes of RAM and 512 Bytes of ROM

- Available chip
  - RAM: 128 × 8
  - ROM: 512 × 8

- No of RAM and ROM Chip required
  - 4 RAM Chip
  - 1 ROM Chip



Memory Address Map

Note: Bits 11 to 16 are all zero

Component	Starting Address											Ending Address										
	Binary										Hexa Decimal	Binary										Hexa Decimal
	10	9	8	7	6	5	4	3	2	1		10	9	8	7	6	5	4	3	2	1	
	0	0	0	0	0	0	0	0	0	0	0000	0	0	0	1	1	1	1	1	1	1	007F
RAM-1	0	0	1	0	0	0	0	0	0	0	0080	0	0	1	1	1	1	1	1	1	1	00FF
RAM-2	0	1	0	0	0	0	0	0	0	0	0100	0	1	0	1	1	1	1	1	1	1	017F
RAM-3	0	1	1	0	0	0	0	0	0	0	0180	0	1	1	1	1	1	1	1	1	1	01FF
RAM-4	1	0	0	0	0	0	0	0	0	0	0200	1	1	1	1	1	1	1	1	1	1	03FF
ROM																						

Associative Memory

- A memory unit accessed by content is called an **associative memory** or **content addressable memory(CAM)**.
- Time required to find an item stored in memory can be reduced.
- Can be accessed simultaneously.
- The memory is capable of finding an empty unused location to store the word.
- When a word is to be read from the memory, the content of the word, or part of the word, is specified.
- The memory locates all words which match the specified content and marks them for reading.
- Expensive

Cache Memory

- **Locality of reference**: few localized areas of memory are repeatedly referred.

- If the active program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing total execution time of the program. Such a fast small memory is referred to as a **cache memory**.

- Less access time than main memory.

- What happens when CPU access memory?
  - CPU-----> Cache Memory ----->Main Memory

Performance of Cache Memory

- Hit ratio =

- T_C = Cache Memory Access Time
- T_m = Main Memory Access Time
- T_{avg}= Average Memory Access Time
- H = Hit ratio

- $T_{avg} = H \times T_C + (1 - H) \times (T_C + T_m)$

Addressing Mapping

- Decide which block of main memory has to be placed where in the cache memory.

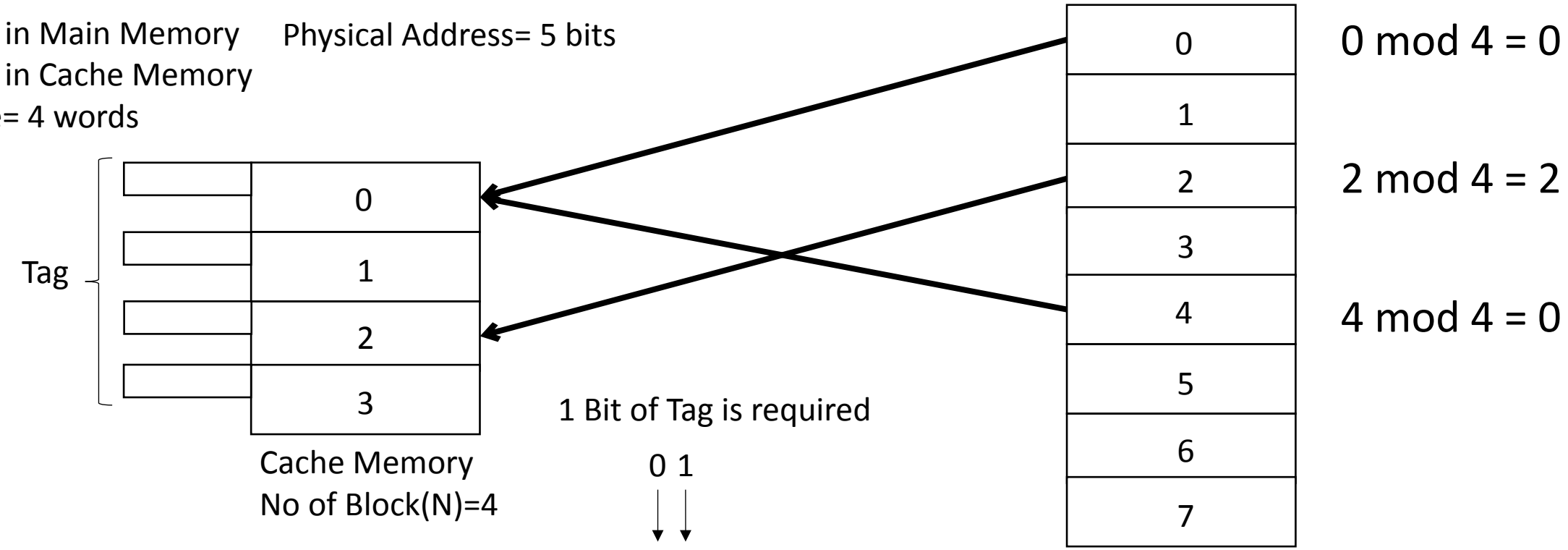
- Three types of Mapping process:
  1. Direct Mapping
  2. Associative Mapping
  3. Set – Associative Mapping

Direct Mapping

K^{th} Block of main memory is placed in $(K^{th} \bmod N)^{th}$ block of cache memory.

32 words in Main Memory
16 words in Cache Memory
Block size= 4 words

Physical Address= 5 bits



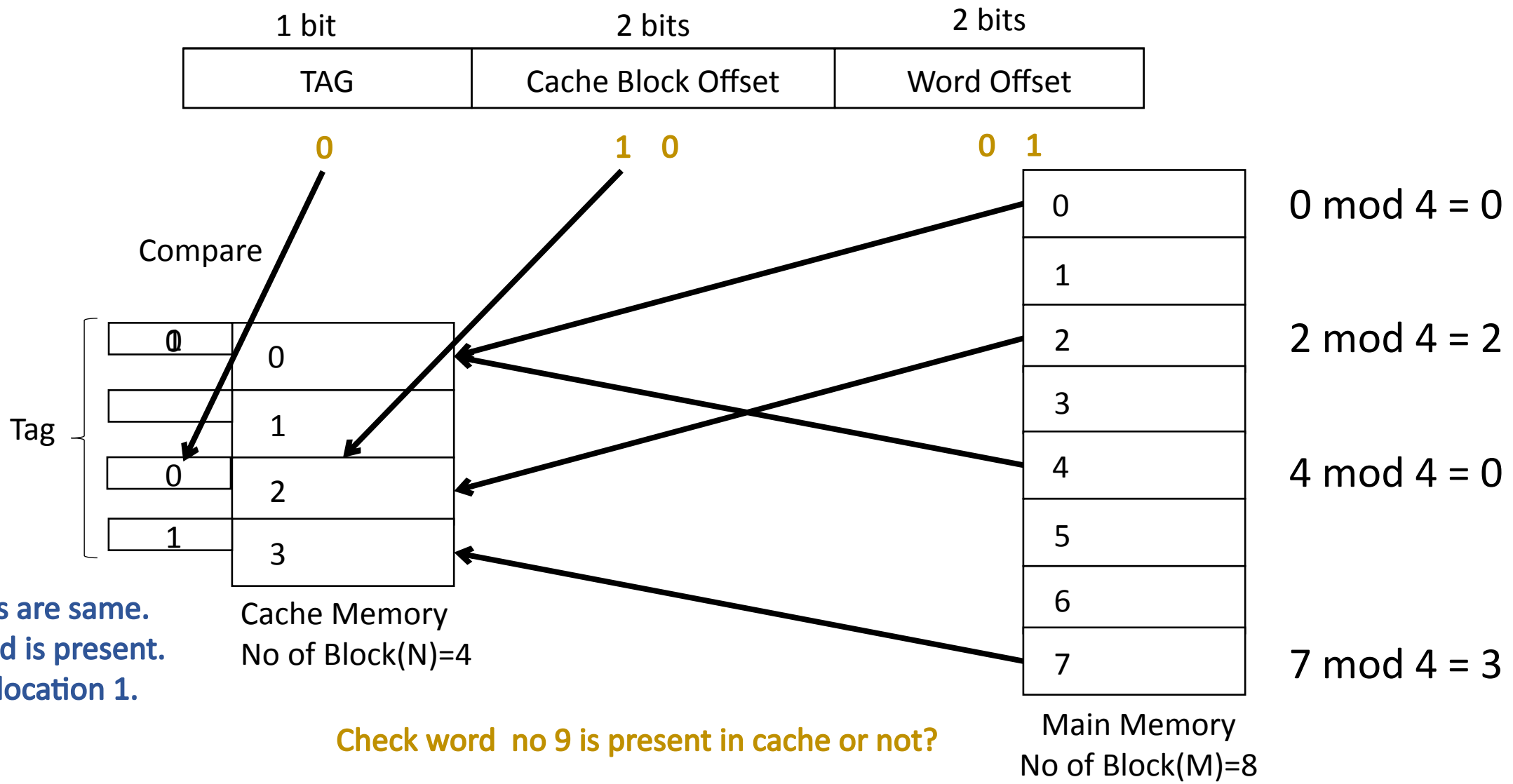
MM Blocks That Can be Placed in Block 0 of CM are {0,4}

MM Blocks That Can be Placed in Block 1 of CM are {1,5}

MM Blocks That Can be Placed in Block 2 of CM are {2,6}

MM Blocks That Can be Placed in Block 3 of CM are {3,7}

Direct Mapping



Tag bits are same.
So word is present.
It is at location 1.

Check word no 9 is present in cache or not?

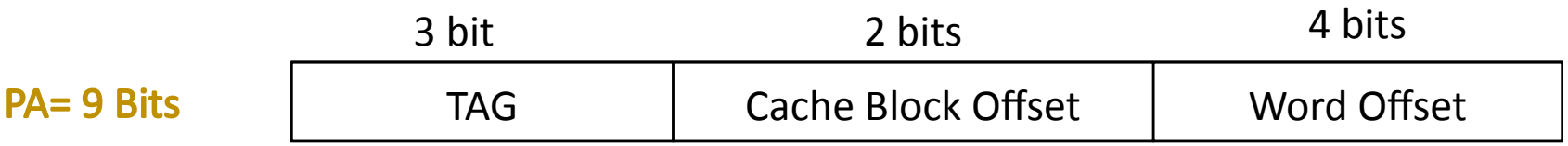
Physical Address 0 1 0 0 1

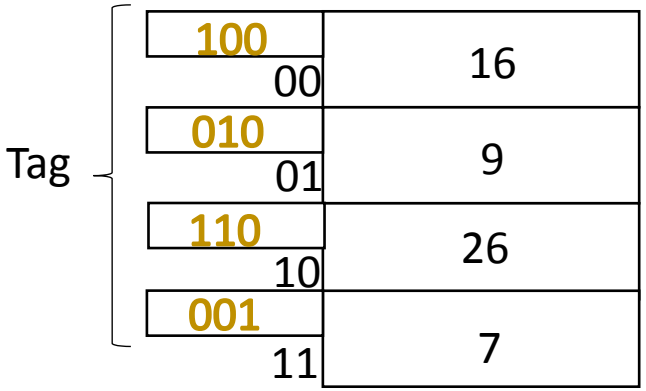
Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for cache offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Direct Mapping





Cache Memory
No of Block(N)=4

10000	16
01001	9
11010	26
00111	7

0
1
2
3
4
:
30
31

Main Memory
No of Block(M)=32

Check word is present in cache or not?

word no 237
PA: 011 10 1101 ✖
TAG is Different in 10

word no 400
PA: 110 01 0000 ✖
TAG is Different in 01

word no 145
PA: 010 01 0001

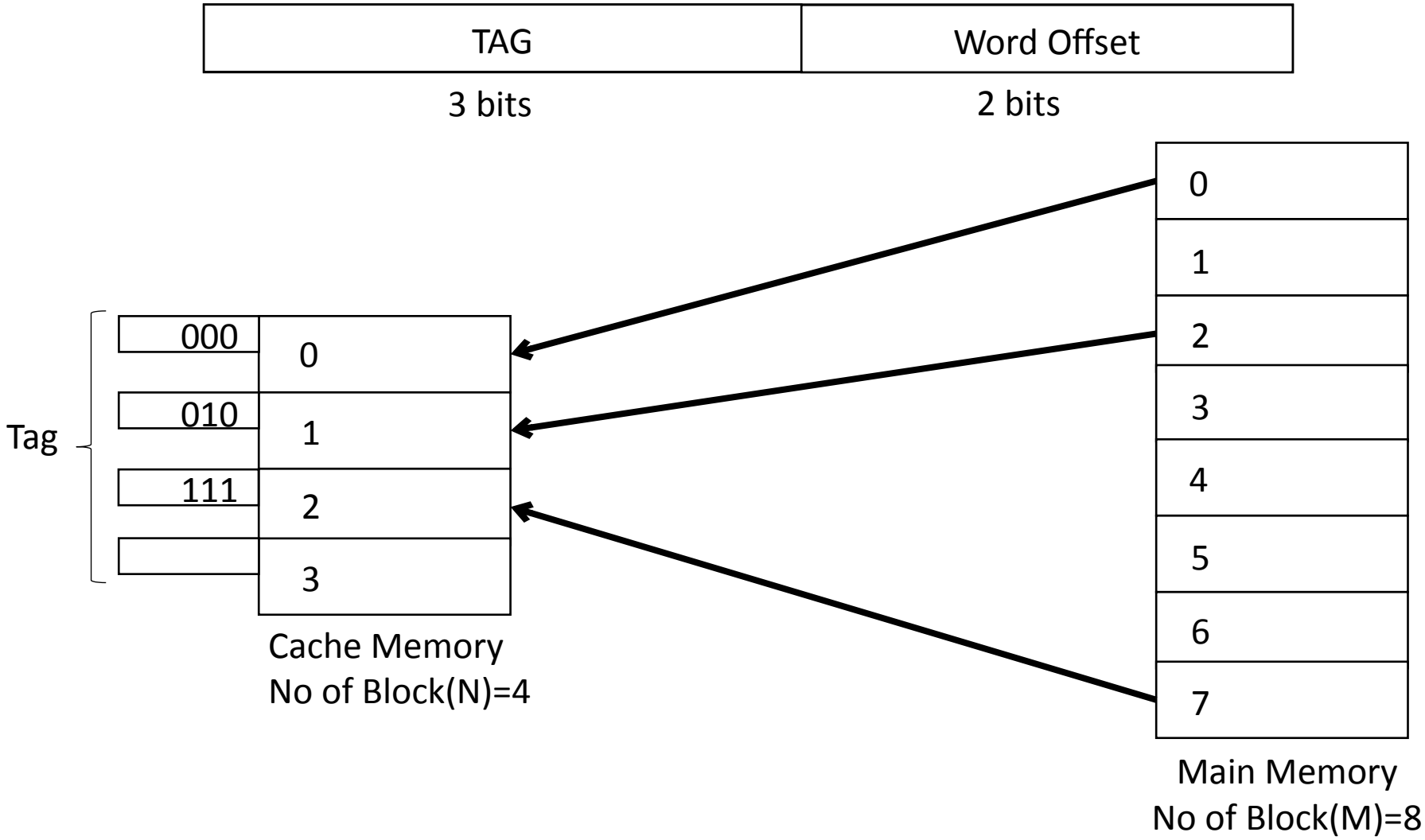
TAG is same in 01 word is at 0001 in that block

Limitation of Direct Mapping

- Conflict penalties:
  - Present of one block abstracting the incoming block inspite of the cache is free
  - MM Block ref: 0, 4, 0, 4, 8, 0, 4, 12, 4, 12

Associative Mapping

Any Block of main memory can be placed any where in the cache



Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for cache offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Set Associative Mapping

- Advantage of Direct Mapping and Associative Mapping.

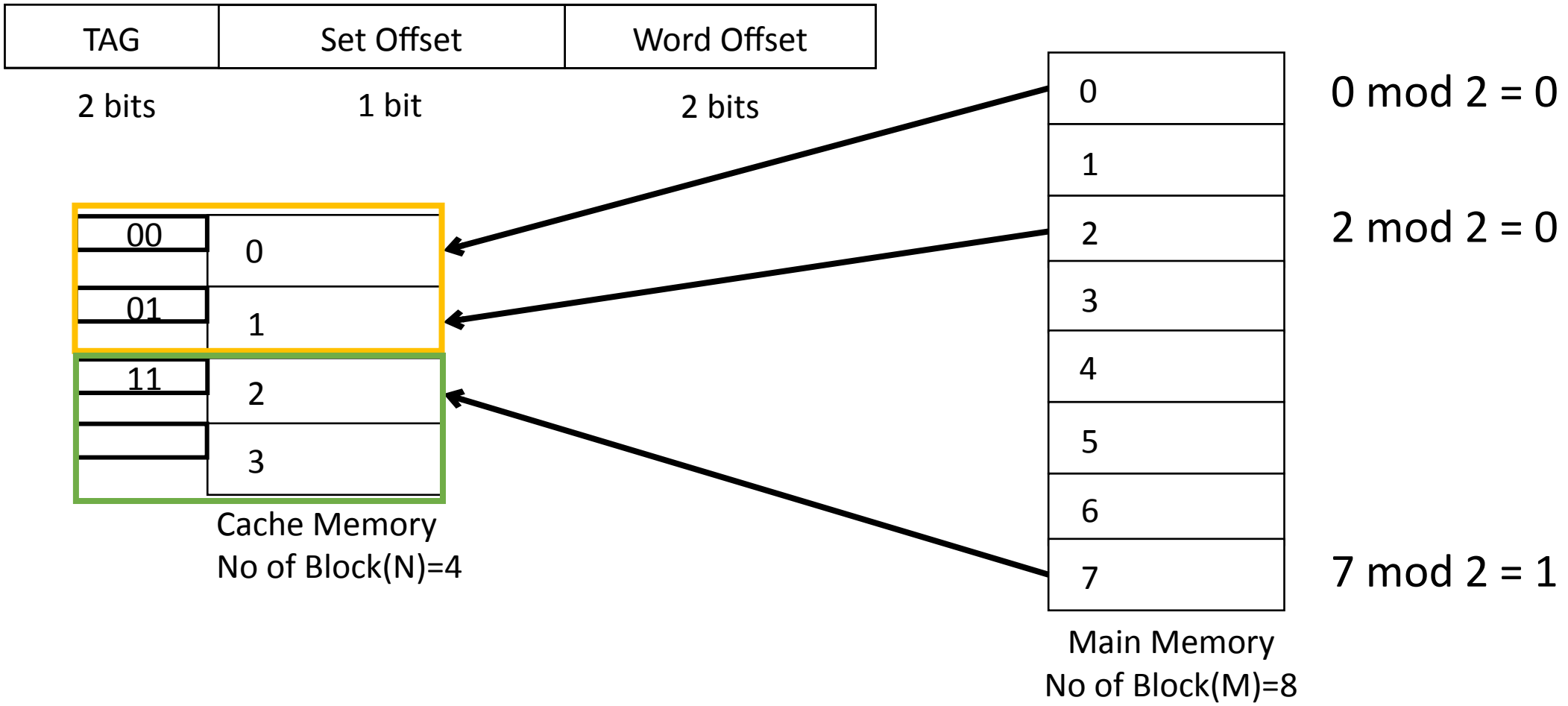
- Total number of sets in k-way set associative is:

  S =

- J^{th} Block of MM has to be placed in (J^{th} mod S)th Set. Within the set it can be placed anywhere.

2 Way Set Associative Mapping

Sets



Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words
- 2-way Set Associative

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for set offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Question

- Consider 1MB cache and 4 GB MM are partitioned into 64KB blocks.

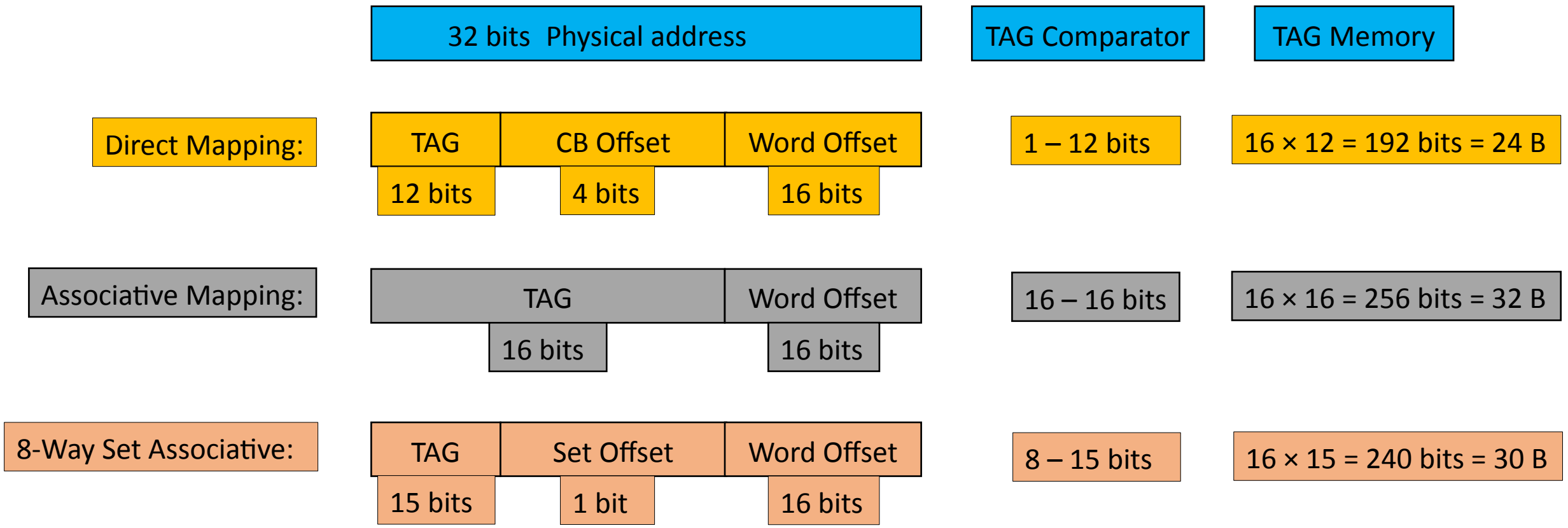
1. How many bits are required for Physical address if word size is 1B?

2. How many block are there in MM and CM?

3. How many TAG bits are required for
    1. Direct Mapping
    2. Associate Mapping
    3. 8-Way set Associative Mapping

4. How many TAG memory in Bytes are required for each case?

Answer



Observation

- What about 1 way set associative?
  - It is similar to direct mapping.
  - Because each set contains 1 block. To select set we use modulo operation. So, It will tell you in which set you have to go and there is one block only in that, which is similar to direct mapping.

- What about N-way set associative?
  - It is similar to Associative Mapping.
  - As set size is equal to total number of cache block, there is only 1 set.
  - In set you can put your block any where which is similar to associative mapping.

Writing to main memory

- Write-through
  - Useful in Direct memory access transfer

- Write-back

Cache Initialization

- Valid bit

Virtual Memory

- What is it?
- Why it is required?
- Paging overview