Memory

Part 1

Final Exam Review Session

- **Final Exam Review session**
- A) August 4th
- B) August 5th
- C) August 6th
- D) August 7th
- E) August 10th

Memory and the Ideal Scenario

- Unlimited amount of memory: Programmers Have Illusion that Large amount of Fast Memory is available.
- History Of WWII term paper
- Go to the Library – get 4 books that cover this topic well –place on your **Desk**
  - Realize do not have anything that covers Hiroshima/Nagasaki
    - Go Back and get 3 more books on this topic
- You spend the rest of the afternoon with those books on your desk, and just going through these for your information.
  - You have Fast access to those books that are sitting on your desk-
  - Because they are Close and They are Just a Few
- Very low chance that you will need to access all the books in the library with equal probability

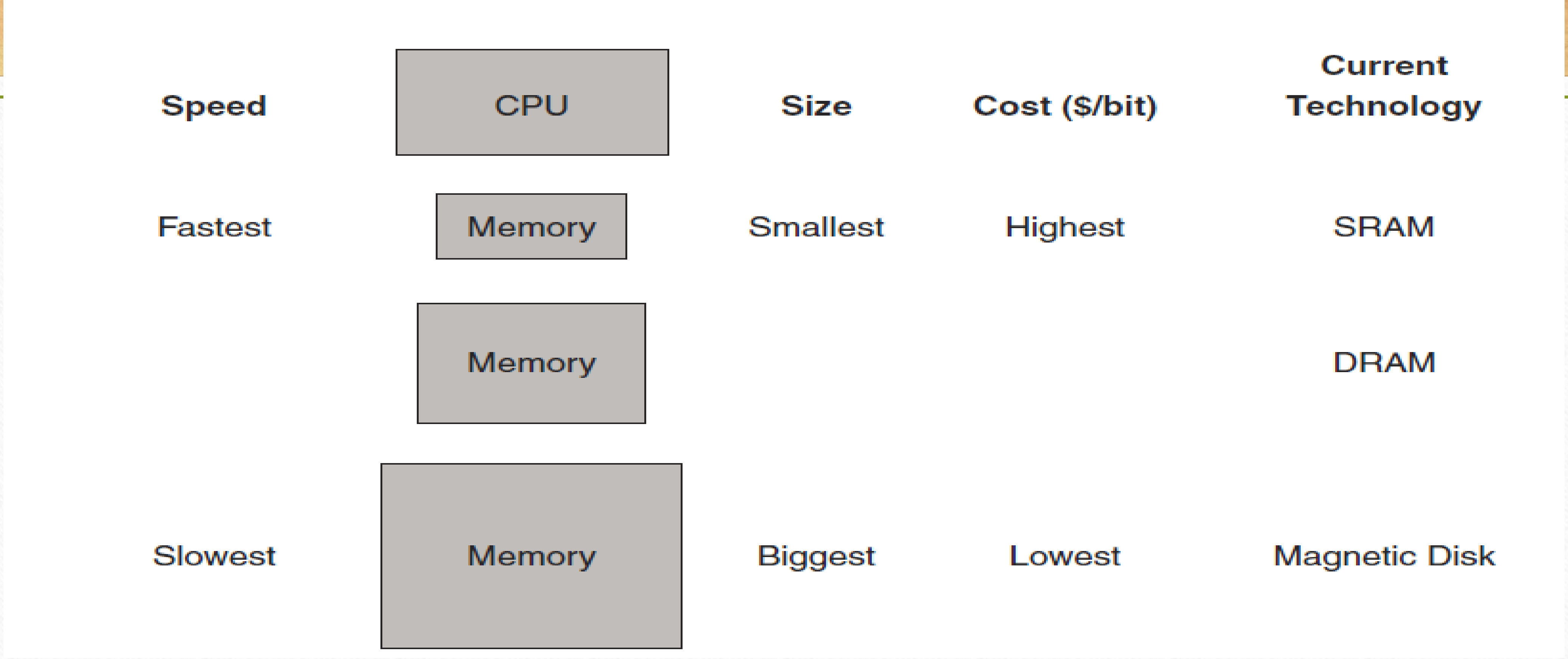
Memory and the Ideal Scenario

- Similar to a Program stored in memory
- Imagine all the code in a Program to be the Library (Large Storage Space)
- Desk is your Cache : Where you have quick access to certain small portions of the program code.
- **You will spend 80% of your time in 20% of the code**
  - CACHE IS FAST ACCESS FOR TWO REASONS
    - It is Close
    - It is small enough such that accessing any topic in those books is pretty fast.
    - If I had 100 books on my desk – not so fast anymore

Principle of Locality

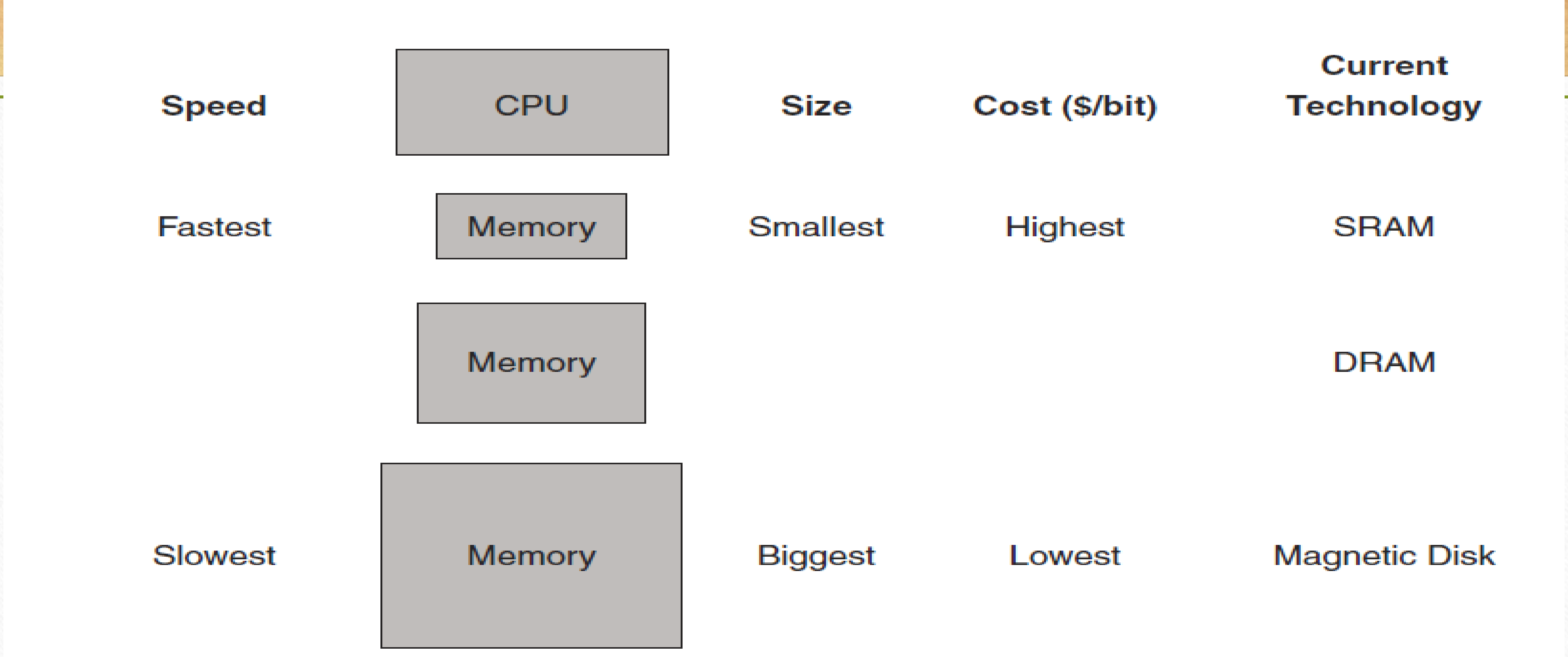
- **temporal locality** The principle stating that if a data location is referenced once then it will tend to be referenced again soon.
- **spatial locality** The locality principle stating that if a data location is referenced, data locations with nearby addresses will tend to be referenced soon.
- IF we were going to access the books in the library – at any given time each with equal probability =

idea of close nearby fast Desk would not work well.



*A memory hierarchy consists of multiple levels of memory with different speeds and sizes.
*The faster memories are more expensive per bit than the slower memories and thus smaller.

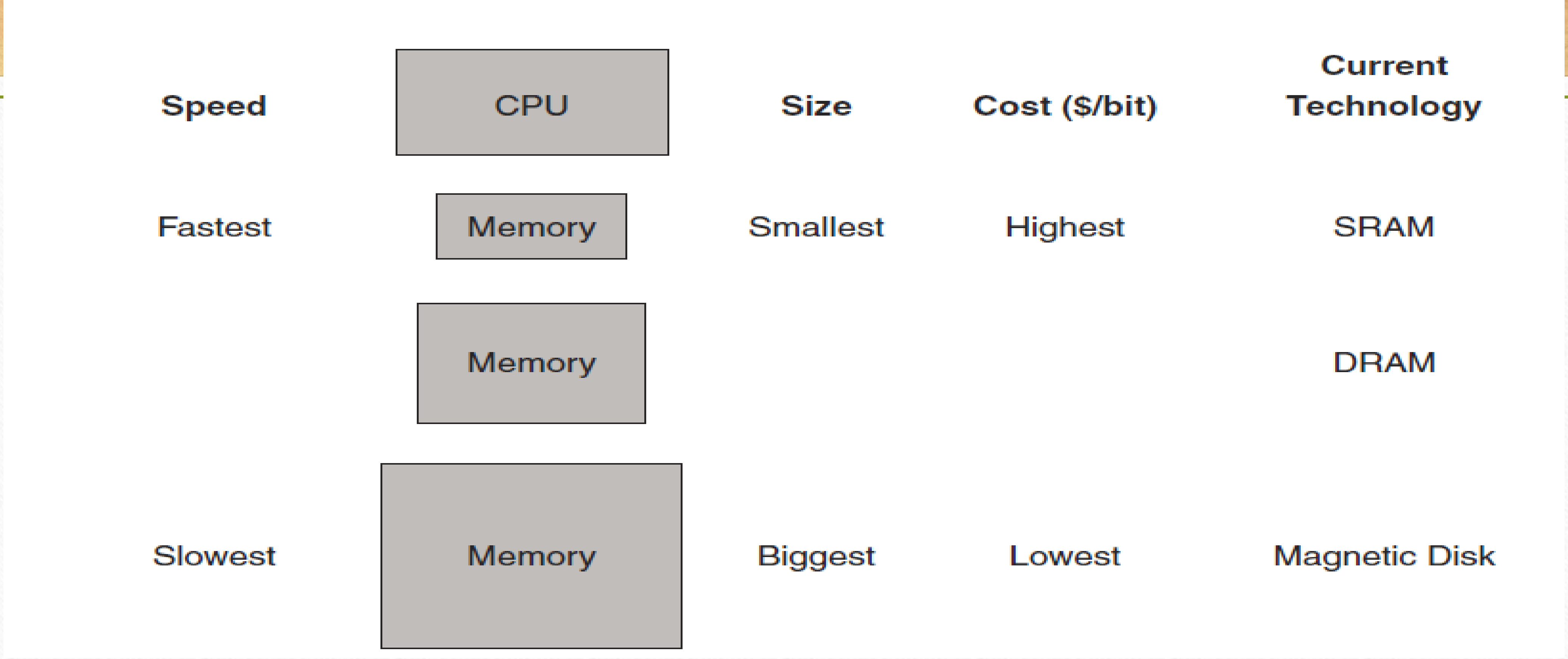
As we move away from the CPU
Memories get slower (longer access time)
And cheaper per bit



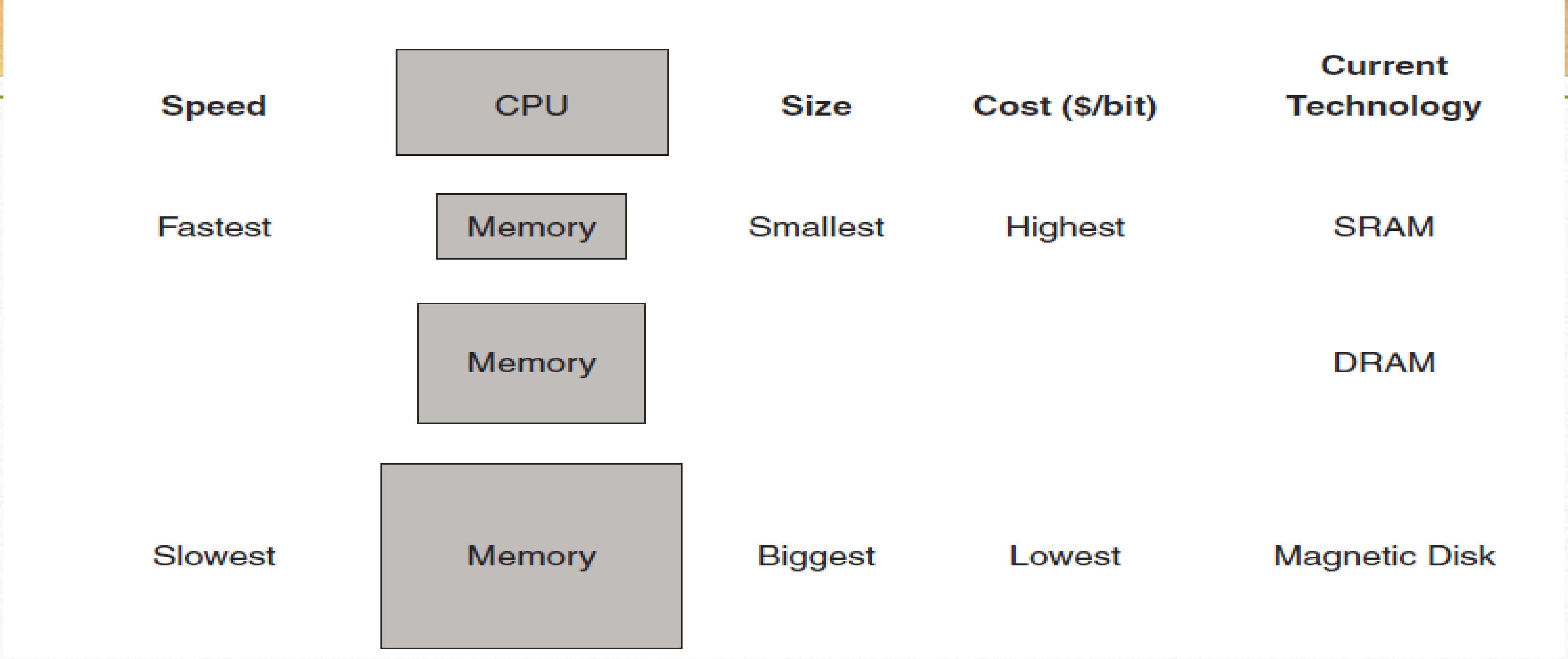
*Main memory is implemented from DRAM (dynamic random access memory)
*SRAM (static random access memory) for levels closer to the processor (caches)
DRAM is less costly per bit than SRAM, although it is substantially slower. **…. (we know why ☺)**
*The third technology, used to implement the largest and slowest level in the hierarchy, is magnetic disk.

Why is DRAM slower but cheaper than SRAM?

- A) DRAM writes are slower than reads due to capacitor refresh.. And it is cheaper because of the capacitors
- B) DRAM has slow reads due to need to refresh the capacitors. Cheaper because it uses only one transistor
- C) SRAM is faster because it is using no capacitors and more expensive because it uses 2 transistors as opposed to one
- D) DRAM is slower because of address lookup uses large MUX
- E) SRAM faster because of two level decoding



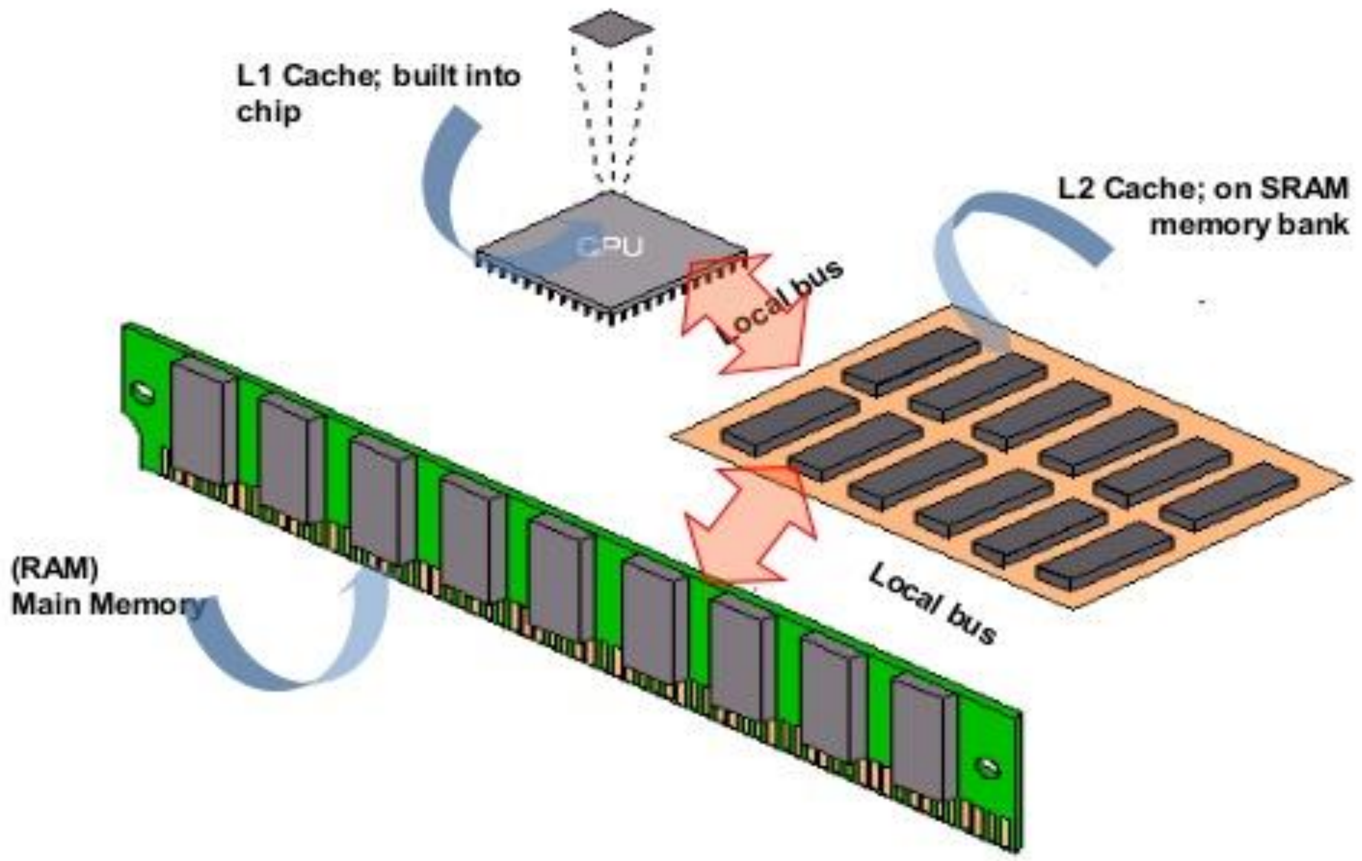
*Main memory is implemented from DRAM (dynamic random access memory)
*SRAM (static random access memory) for levels closer to the processor (caches)
DRAM is less costly per bit than SRAM, although it is substantially
slower. **(Capacitors and 1 transistor) Less silicon per bit ☺**
*The third technology, used to implement the largest and slowest level in the hierarchy, is magnetic disk.

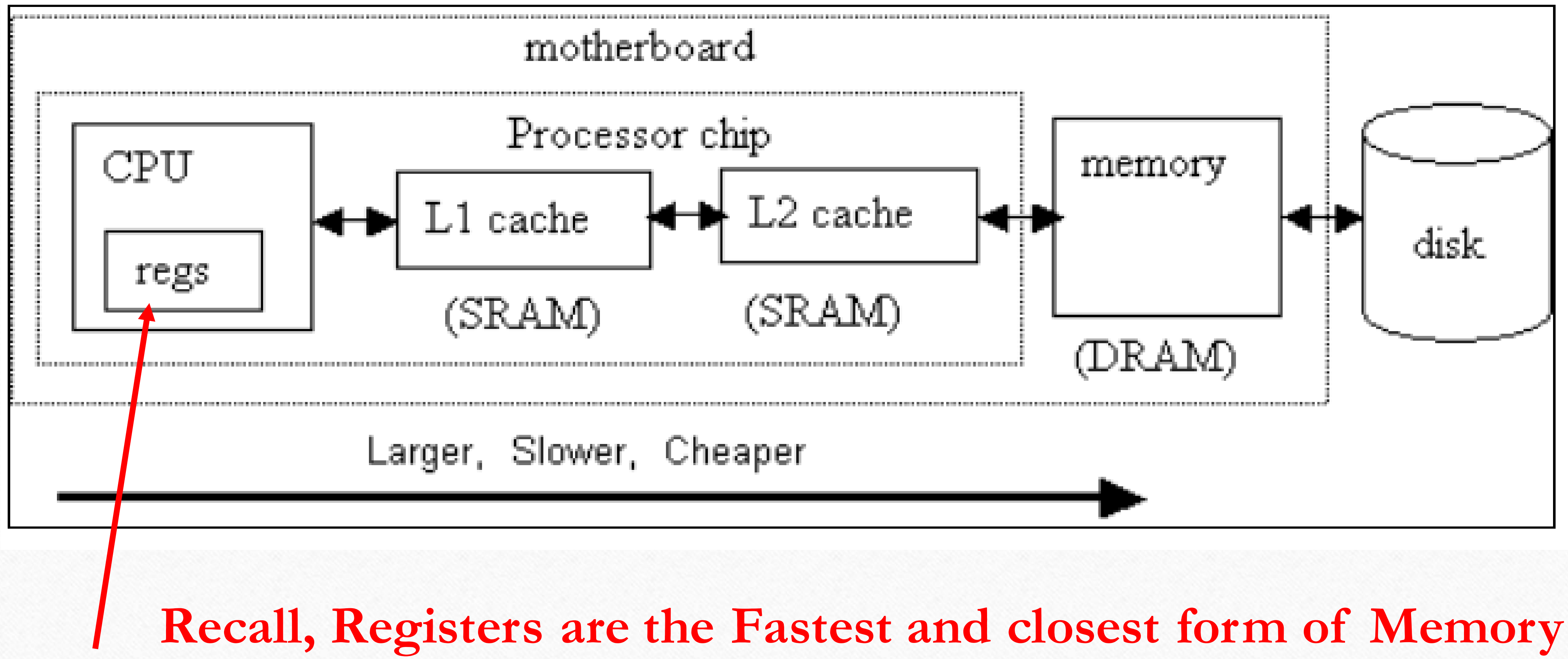


***Flash Memory : Nonvolatile. It is the secondary memory in Personal Mobile Devices**

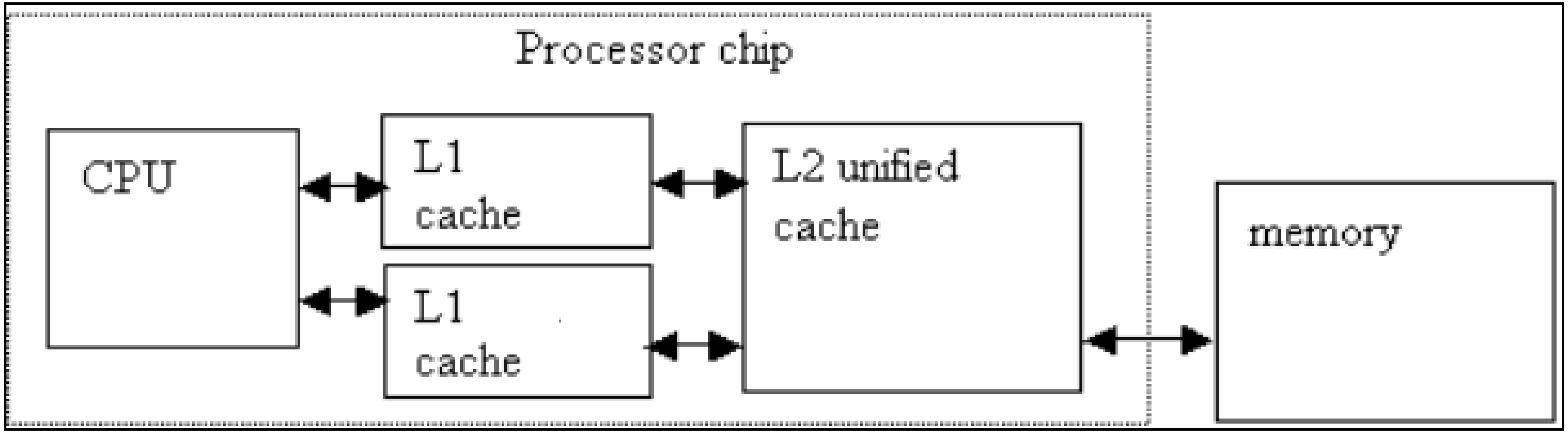
Type of Electrically Erasable Programmable Read Only Memory

The concept of cache memory





Recall, Registers are the Fastest and closest form of Memory To CPU/Datapath



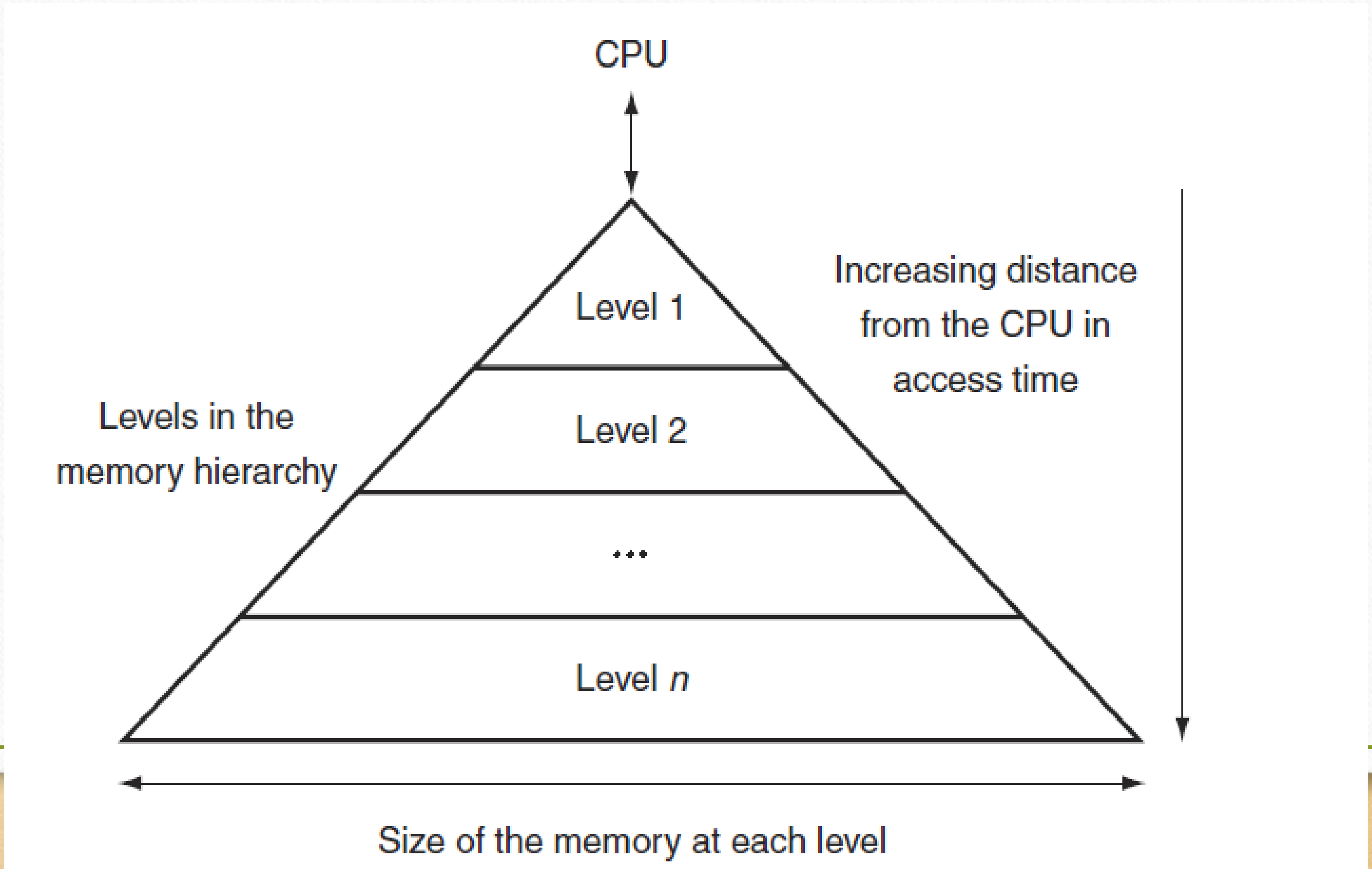
- CPU, Closest Memory (L1 caches) These can be broken up into two

Or One L1 cache

- What could the two L1 caches be used for?
- Think about the datapath….

Hierarchy:

- The memory system is organized as a *hierarchy*: a level closer to the processor is generally a *subset* of any level further away, and all the data is stored at the lowest level.


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

Between Two Adjacent Levels

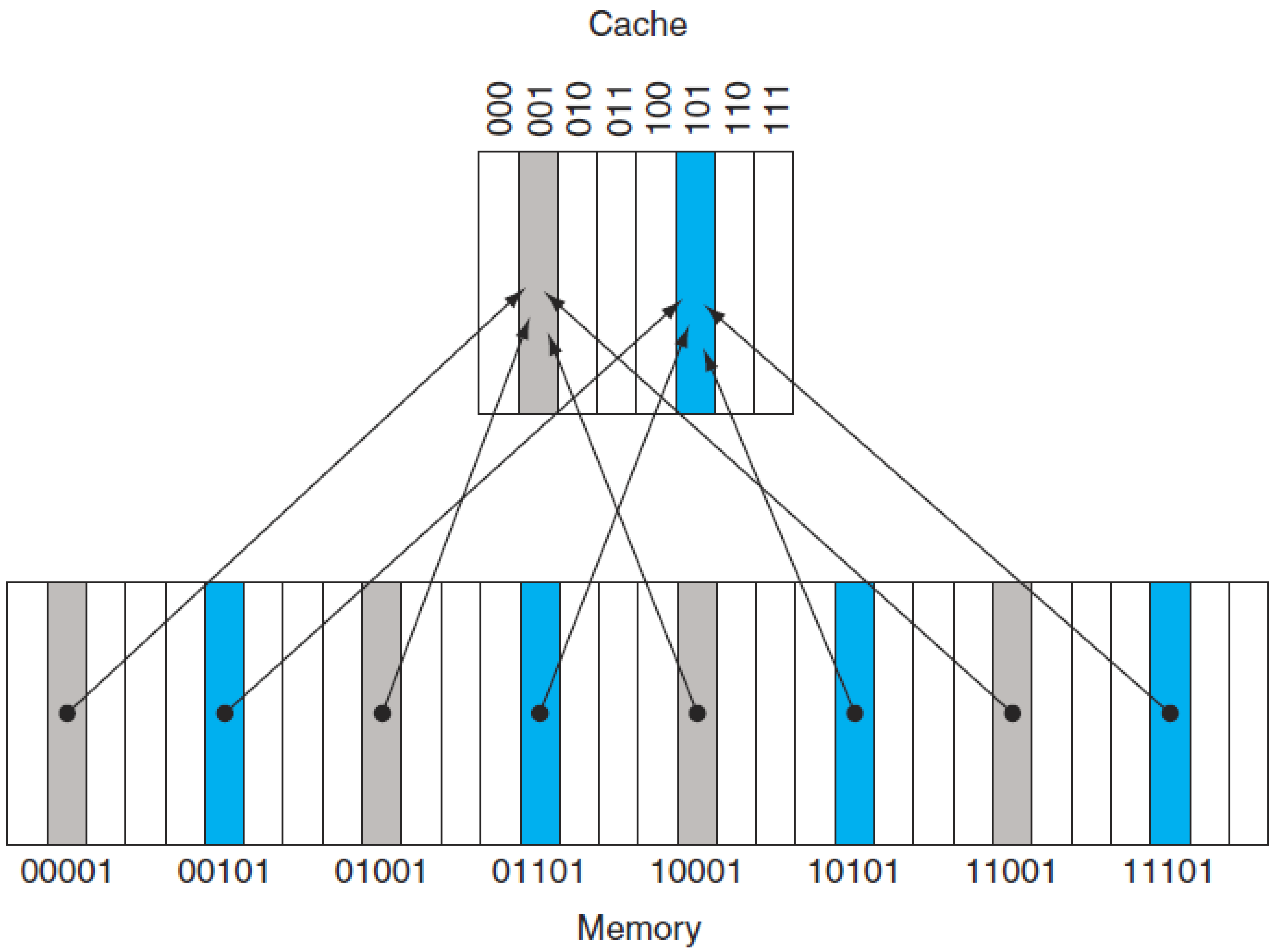
- A memory hierarchy can consist of multiple levels, but data is copied between two adjacent levels at a time, so we can focus our attention on just two levels.
- The upper level—the one closer to the processor—is smaller and faster (since it uses more expensive technology) than the lower level.
- minimum unit of information that can be either present or not present in the two-level hierarchy is called a **block**
- If the data requested by the processor appears in some block in the upper level, this is called a *HIT*
- If the data is not found in the upper level, the request is called a *MISS*.
- The lower level in the hierarchy is then accessed to retrieve the block containing the requested data

- Items not in top level are brought up when requested
- Data only copied between adjacent levels (focus on two)
- Block: minimum unit of information present/not present
- Hit: information present when requested
- Miss: information not present, must be copied up
- Terminology: hit ratio, hit time, miss penalty

Caches

- Cache: level of memory hierarchy between CPU and main memory
- Important questions:
  - How do we know if a data item is in the cache?
  - How do we find it?
- Direct mapped: each memory location is mapped to exactly one location in the cache
- Typical mapping: (block address) modulo (number of blocks in cache)
- Example: block = 1 word, cache size = 8, just take lower 3 bits of word address
- Need tag for each block in cache giving original location

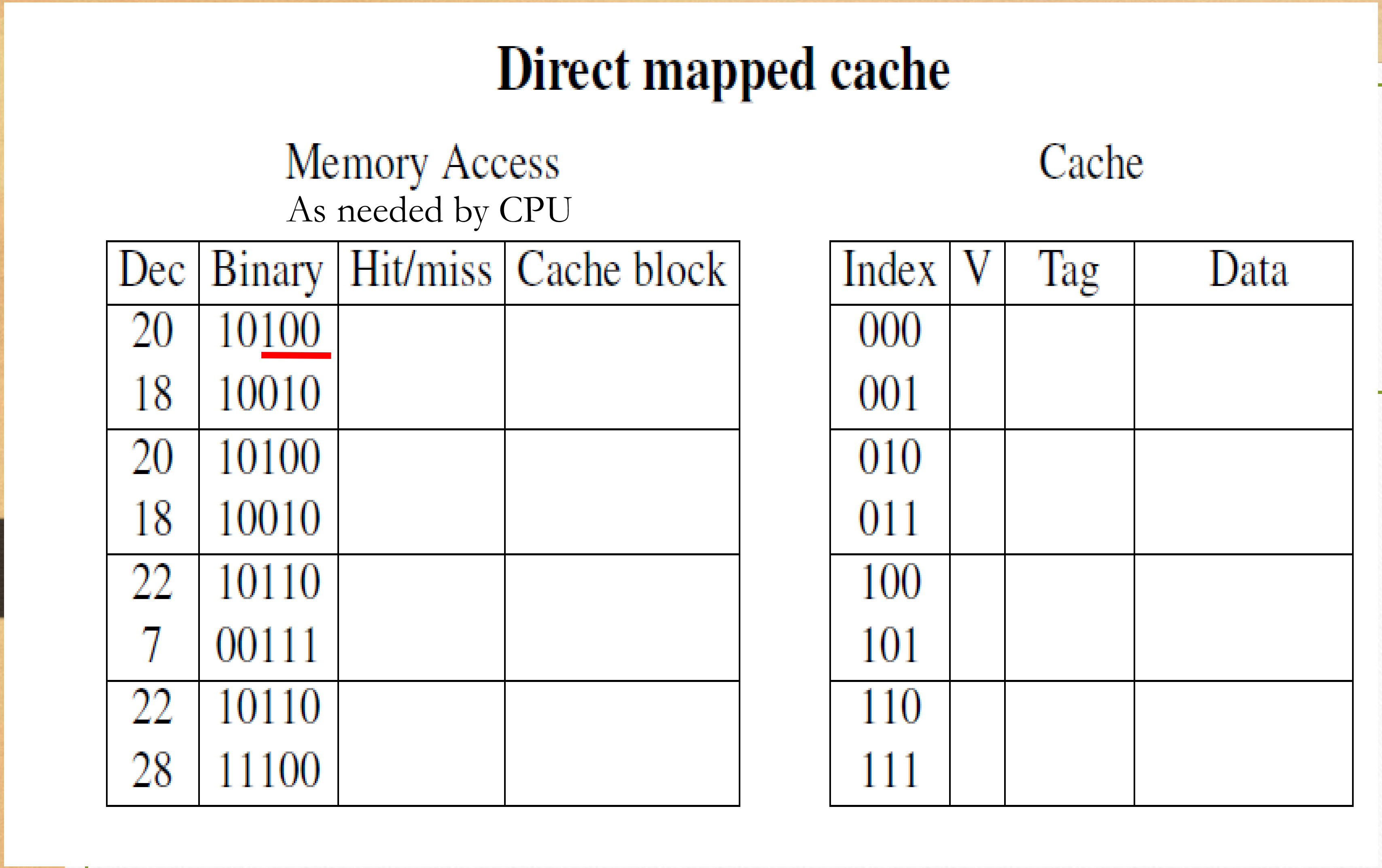
Between Cache and Main Memory : How do we know where an entry can exist in Cache.



Direct Mapped

There is only one place Where blocks in Main Memory can be mapped To Cache.

If it needs to go into Cache it can go only in This location. If some-thing else is there, it will have to be moved out



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10$\underline{100}$		
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

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

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100		
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

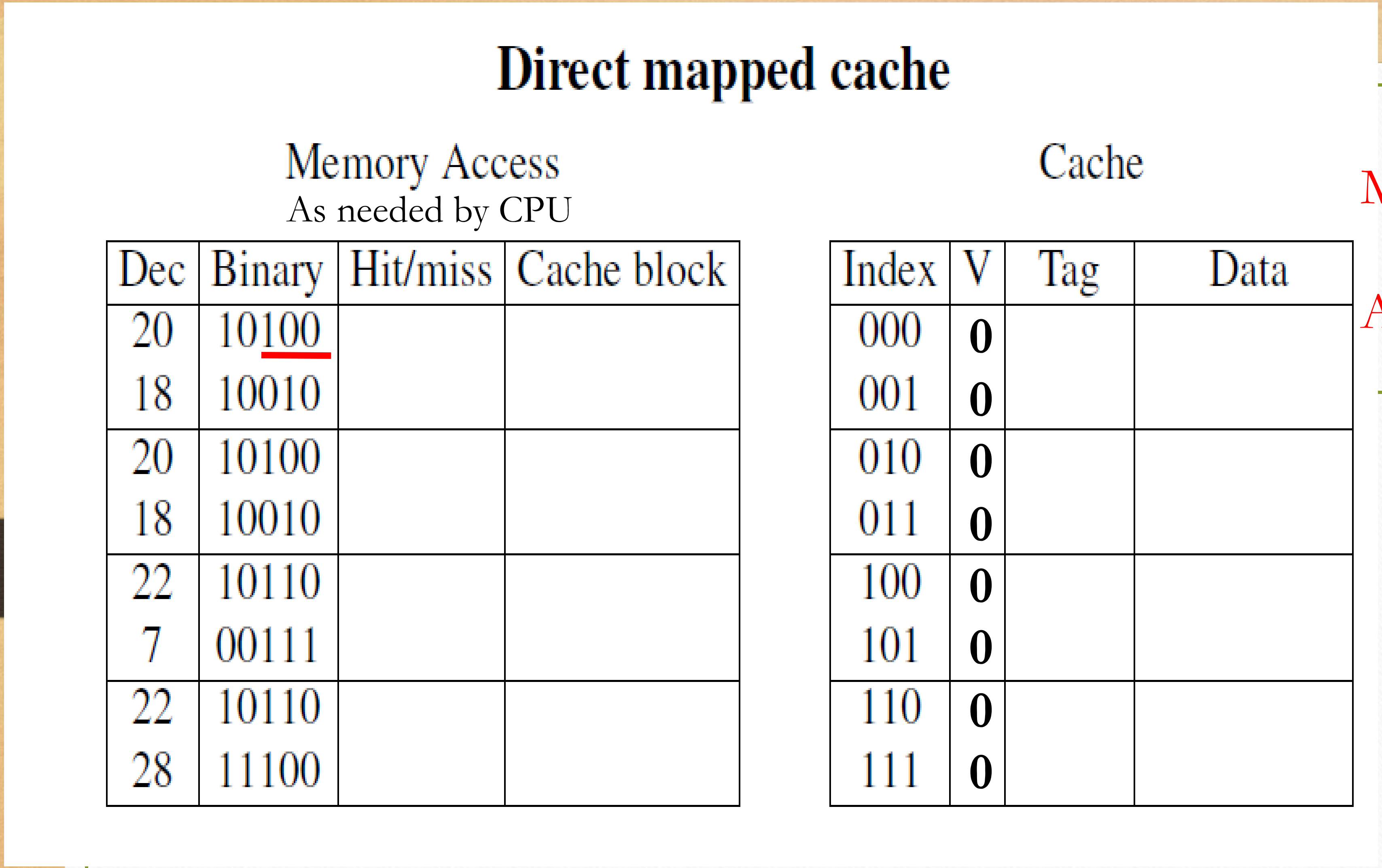
Index	V	Tag	Data
000			
001			
010			
011			
100			
101			
110			
111			

Given the first address I need
Decimal value of 20.
The bits of the binary address
Tell me where to look in cache

8 spaces in the cache,
Therefore, I look at lower 3 bits
To determine where this address
Can exist in my cache.

Main Memory

Direct mapped cache



Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10$\underline{100}$		
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

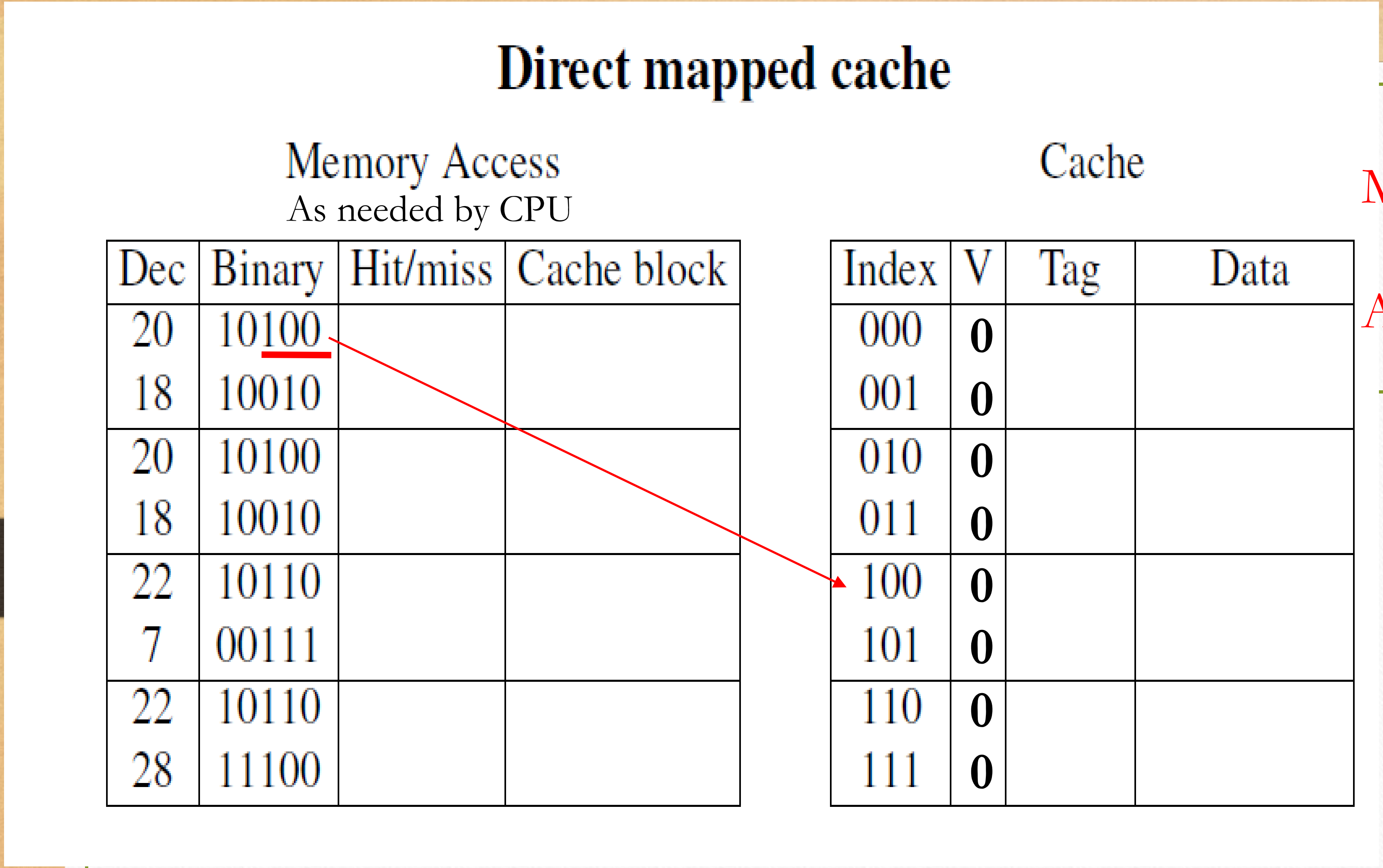
Cache

Index	V	Tag	Data
000	**0**		
001	**0**		
010	**0**		
011	**0**		
100	**0**		
101	**0**		
110	**0**		
111	**0**		

My Cache initially Empty

All Valid Bits= Off

Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100		
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

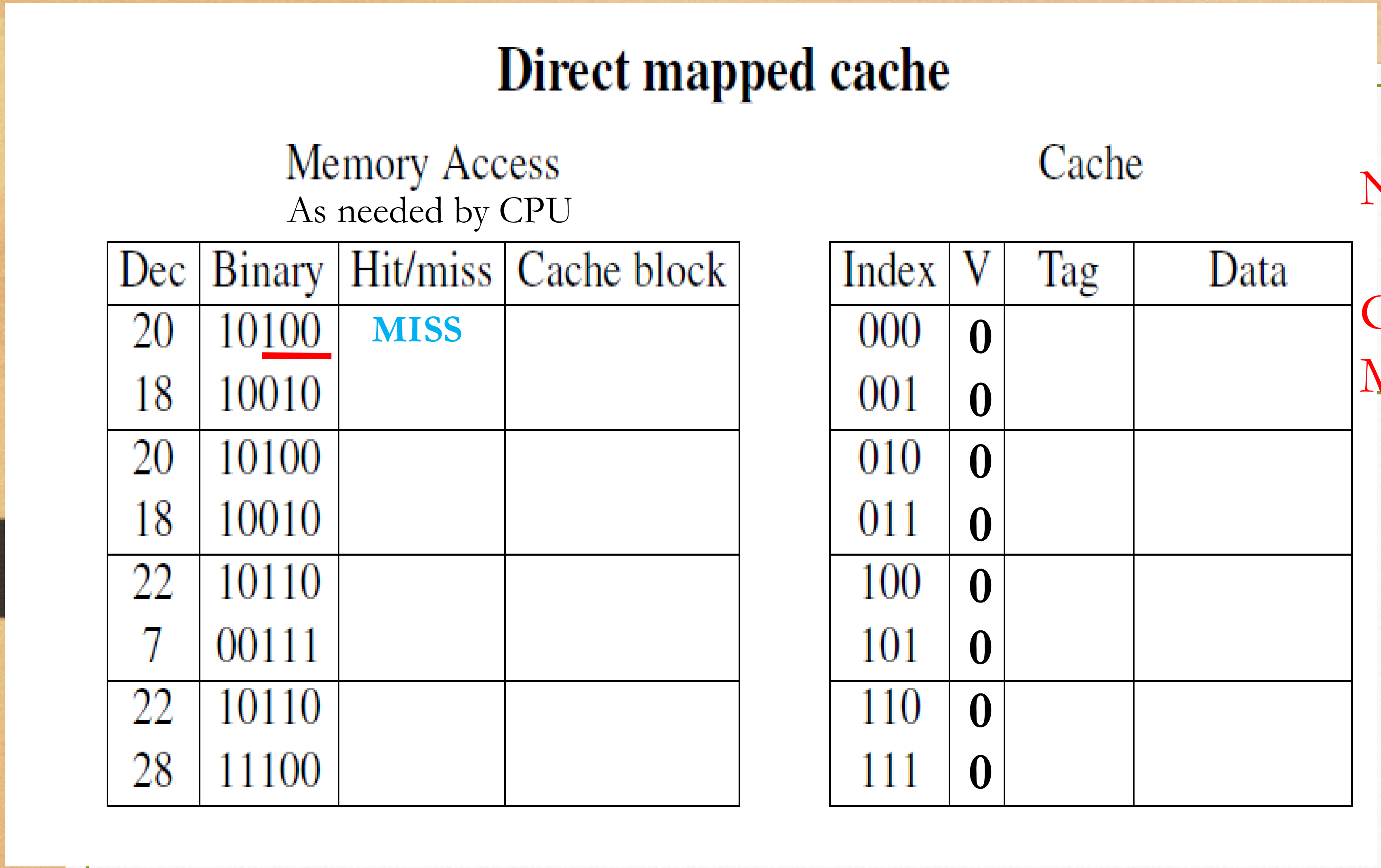
Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

My Cache initially Empty

All Valid Bits= Off

Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	**0**		
001	**0**		
010	**0**		
011	**0**		
100	**0**		
101	**0**		
110	**0**		
111	**0**		

Now Stall processor

Go Get this Data from Main Memory

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

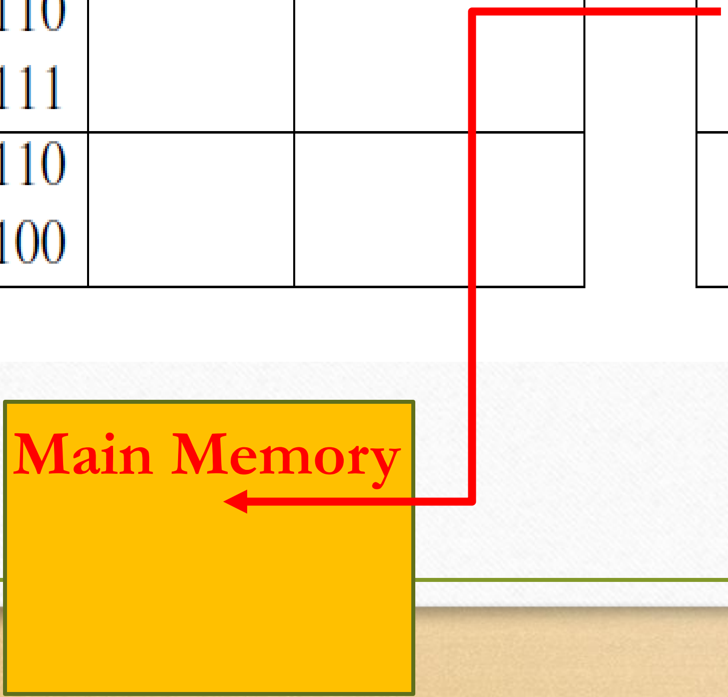
Dec	Binary	Hit/miss	Cache block
20	10<u>100</u>	MISS	
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	0		
101	0		
110	0		
111	0		

Now Stall processor

Go Get this Data from Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10$\underline{100}$	MISS	
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

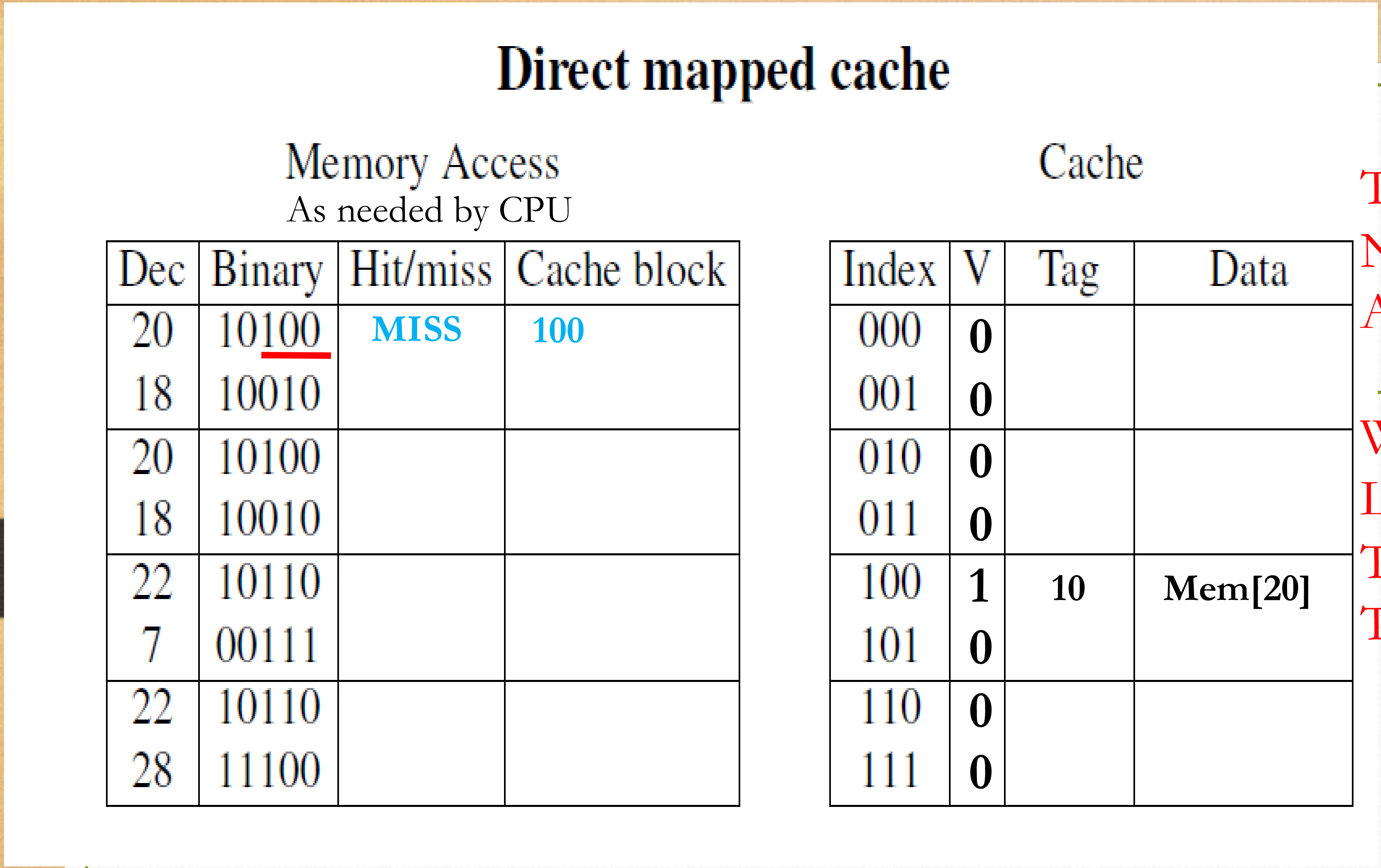
Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

TAG field is remaining Number of bits in the Address.

We used 3 to determine Location in cache, Therefore 2 bits left as TAG field.

Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10$\underline{100}$	MISS	100
18	10010		
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

TAG field is remaining Number of bits in the Address.

We used 3 to determine Location in cache, Therefore 2 bits left as TAG field.

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Next requested address
Is also a MISS the first time

Main Memory

Direct mapped cache

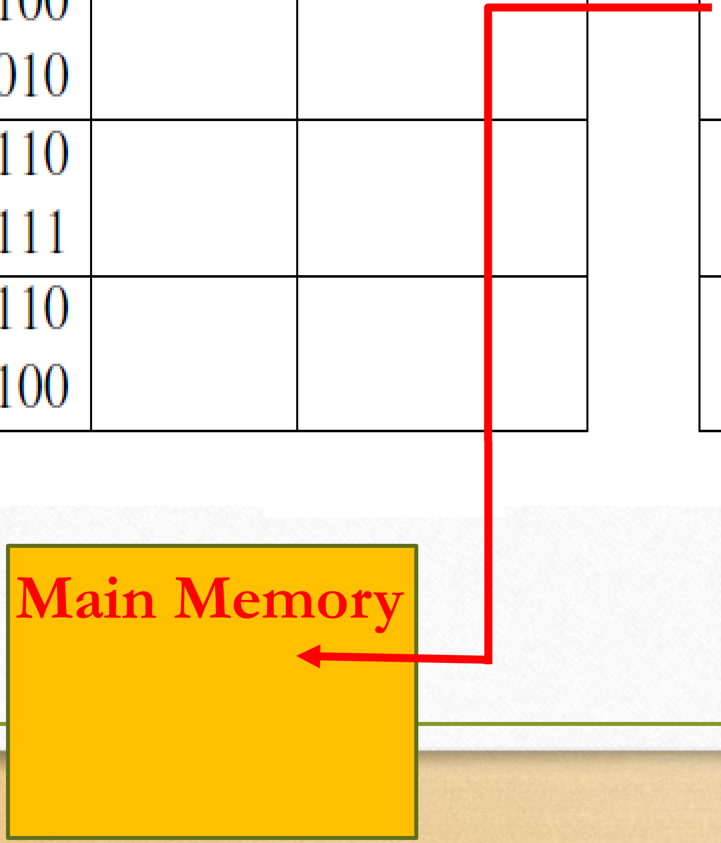
Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

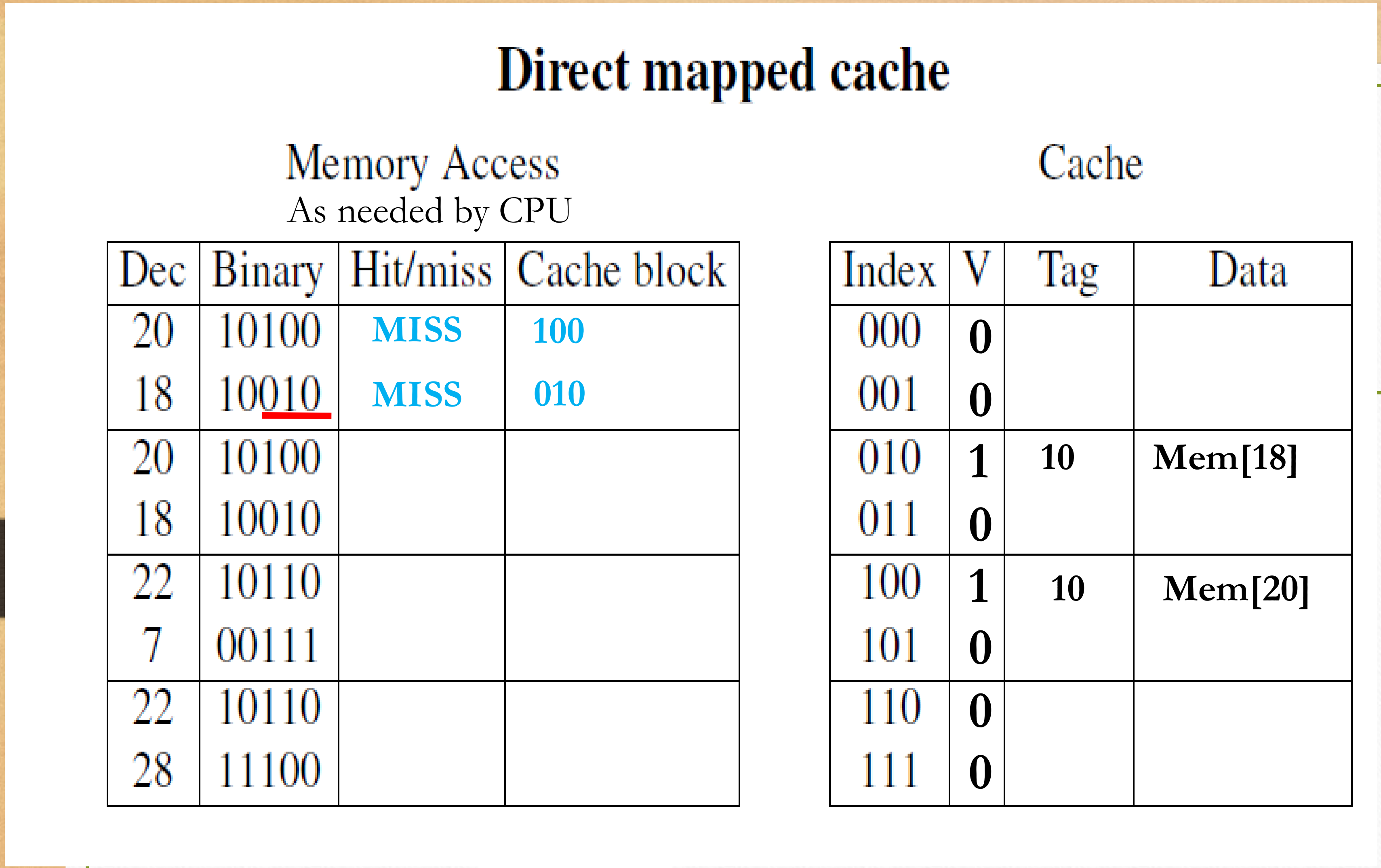
Cache

Index	V	Tag	Data
000	0		
001	0		
010	0		
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Need to Go to Main Memory



Direct mapped cache



Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	$10\underline{010}$	MISS	010
20	10100		
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010		
22	10110		
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Next requested address

First check Cache location 100

Is it Valid? Yes

Does the Tag Field Match

Yes

Hit

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110		
7	00111		
22	10110		
28	11100		

Cache

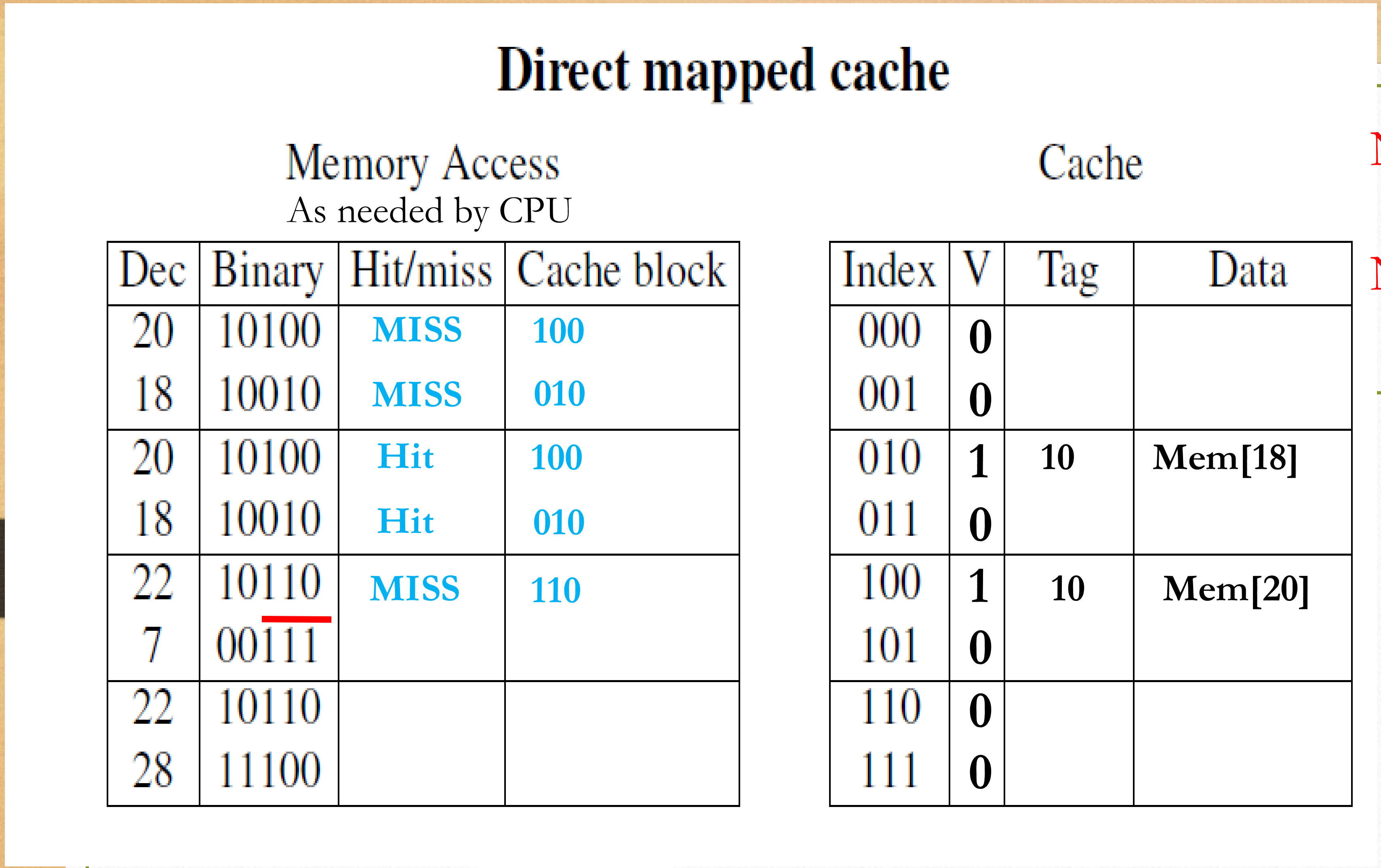
Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Next requested address

First check Cache location 010
~~Is it Valid? Yes~~
Does the Tag Field Match
Yes ➔ Hit

Now these locations
Will be Hits.
Temporal Locality
Once Accessed Once
High Likelihood to
Be accessed again

Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110	MISS	110
7	00111		
22	10110		
28	11100		

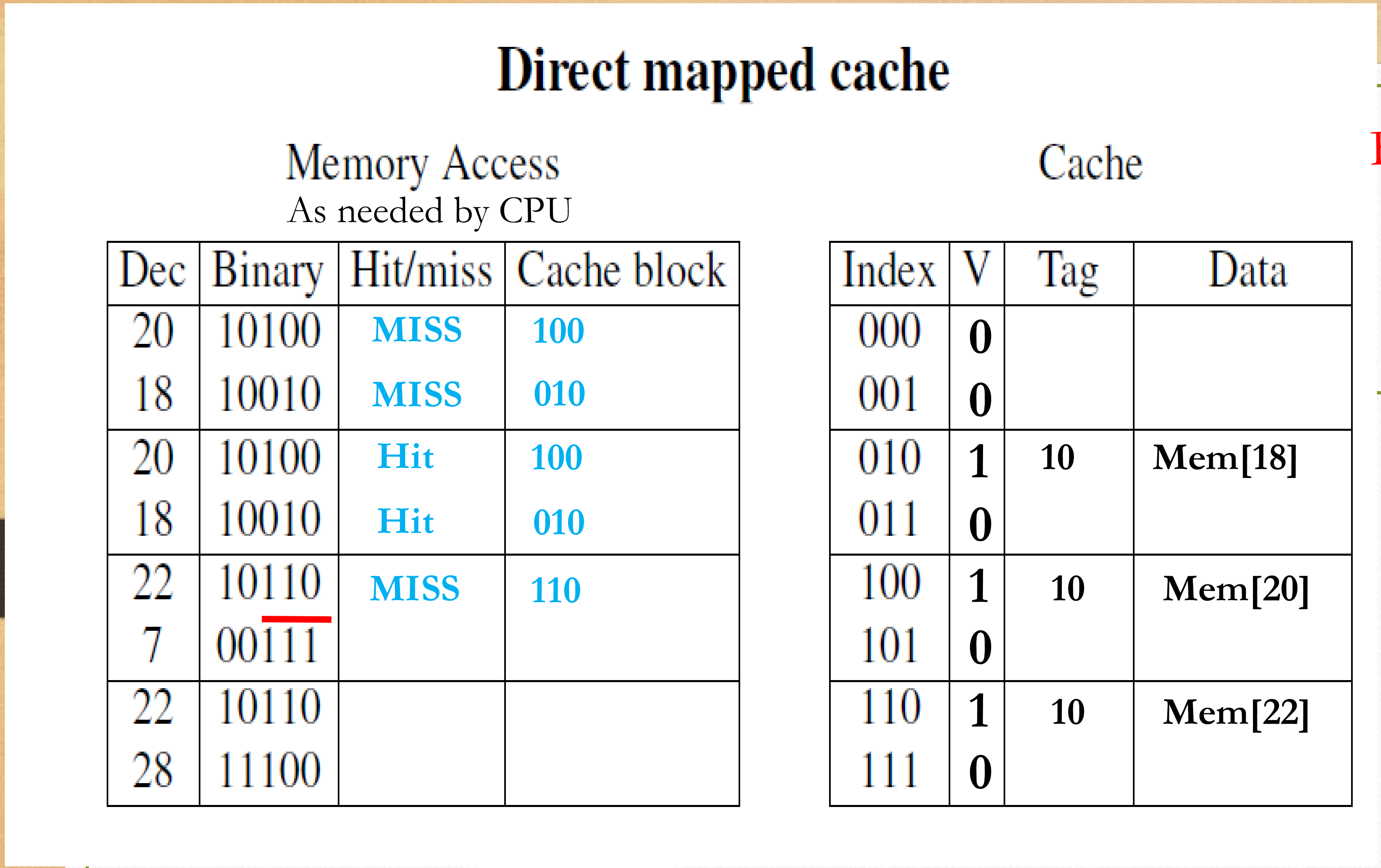
Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	0		
111	0		

Next requested address

New- will be a miss

Main Memory



Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110	MISS	110
7	00111		
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	1	10	Mem[22]
111	0		

Bring In

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	**MISS**	**100**
18	10010	**MISS**	**010**
20	10100	**Hit**	**100**
18	10010	**Hit**	**010**
22	10110	**MISS**	**110**
7	00111	**MISS**	**111**
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	**0**		
001	**0**		
010	**1**	**10**	**Mem[18]**
011	**0**		
100	**1**	**10**	**Mem[20]**
101	**0**		
110	**1**	**10**	**Mem[22]**
111	**0**		

Bring In

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110	MISS	110
7	00111	MISS	111
22	10110		
28	11100		

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	1	10	Mem[22]
111	1	00	Mem[7]

Bring In

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110	MISS	110
7	00111	MISS	111
22	10110	Hit	110
28	11100	MISS	100

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	10	Mem[20]
101	0		
110	1	10	Mem[22]
111	1	00	Mem[7]

NOW something needs
To be kicked out of cache

Main Memory

Direct mapped cache

Memory Access
As needed by CPU

Dec	Binary	Hit/miss	Cache block
20	10100	MISS	100
18	10010	MISS	010
20	10100	Hit	100
18	10010	Hit	010
22	10110	MISS	110
7	00111	MISS	111
22	10110	Hit	110
28	11100	MISS	100

Cache

Index	V	Tag	Data
000	0		
001	0		
010	1	10	Mem[18]
011	0		
100	1	11	Mem[28]
101	0		
110	1	10	Mem[22]
111	1	00	Mem[7]

NOW something needs
To be kicked out of cache

Main Memory