# CS & IT ENGINEERING

## COMPUTER ORGANIZATION AND ARCHITECTURE

### Cache Organization

Lecture No.- 04


By- Vishvadeep Gothi sir

# Recap of Previous Lecture

**Topic** Cache Write

**Topic** Write Through & Write Back Cache

**Topic** Write Allocate vs No Write Allocate

# Topics to be Covered

- **Topic** Cache Mapping
- **Topic** Direct Mapping
- **Topic** Tag & Index

[NAT]

#Q. The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations; 60 memory operand read operations and 40 memory operand write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is?

$t_{avg\ read} = 0.9 * 1 + 0.1 * 5 = 1.4\ ns$

$t_{avg\ write} = 0.9 * 2 + 0.1 * 10 = 2.8\ ns$

Total Read = 100 + 60 = 160

% of read = $\frac{160}{200} = 0.8$

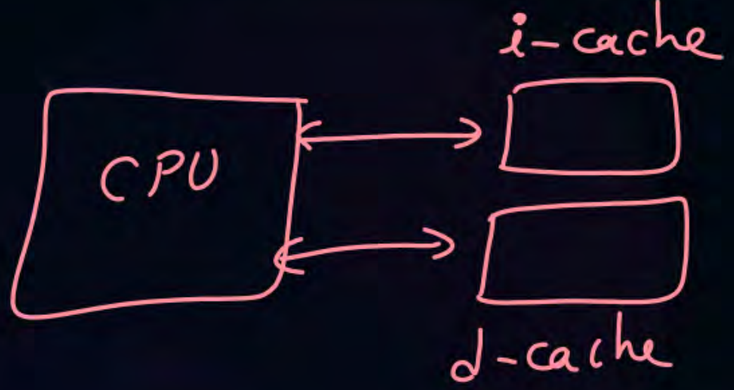
% of write = $\frac{40}{200} = 0.2$

$t_{avg} = (0.8 * 1.4) + (0.2 * 2.8)$

$= 1.68\ ns$

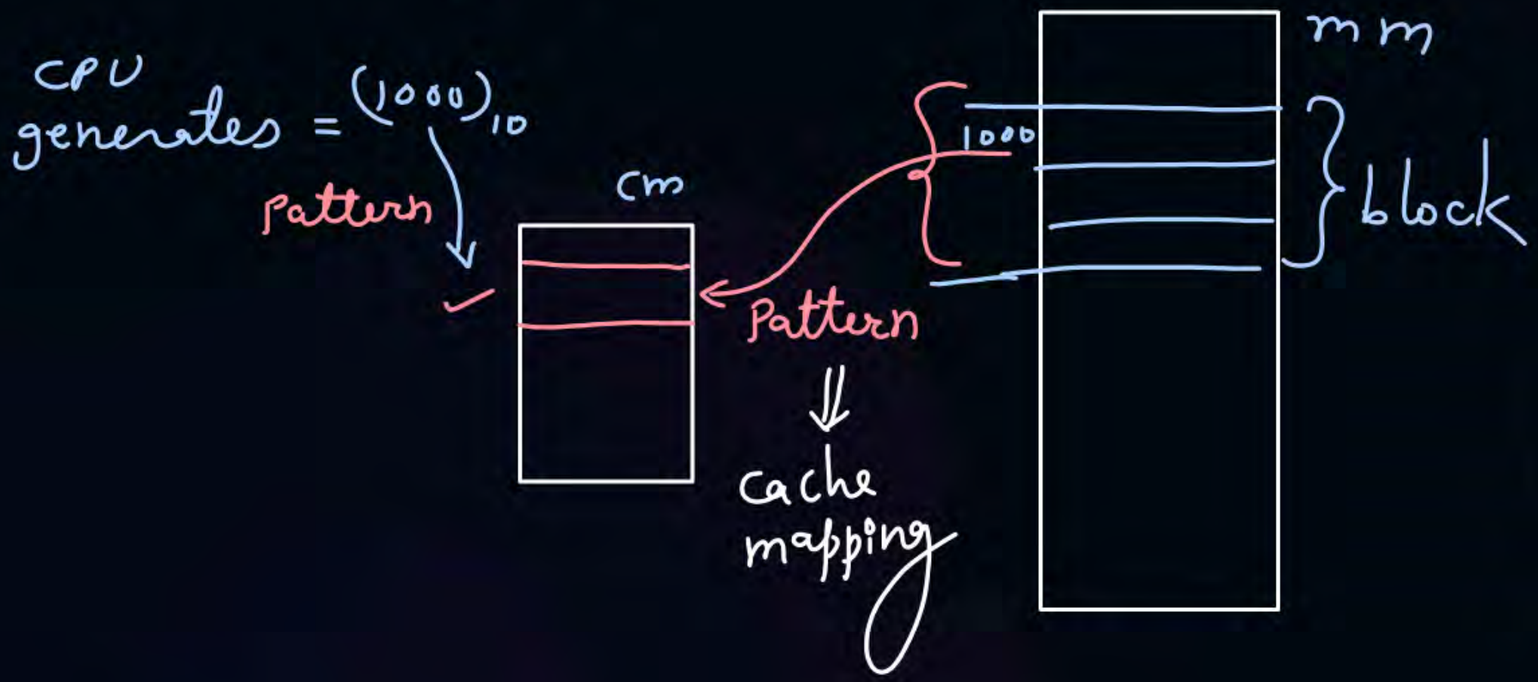
**[NAT]**

#Q. Assume a miss rate of 2% for the instruction cache and of 4% for the data cache, a miss penalty of 100 cycles for all misses, and a frequency of 36% of loads and stores. If the CPI is 2 without memory stalls, determine how much faster the processor runs with a perfect cache that never misses.



# Topic : Cache Mapping

CPU always generates m.m. address




Transformation of mm data in cache mm

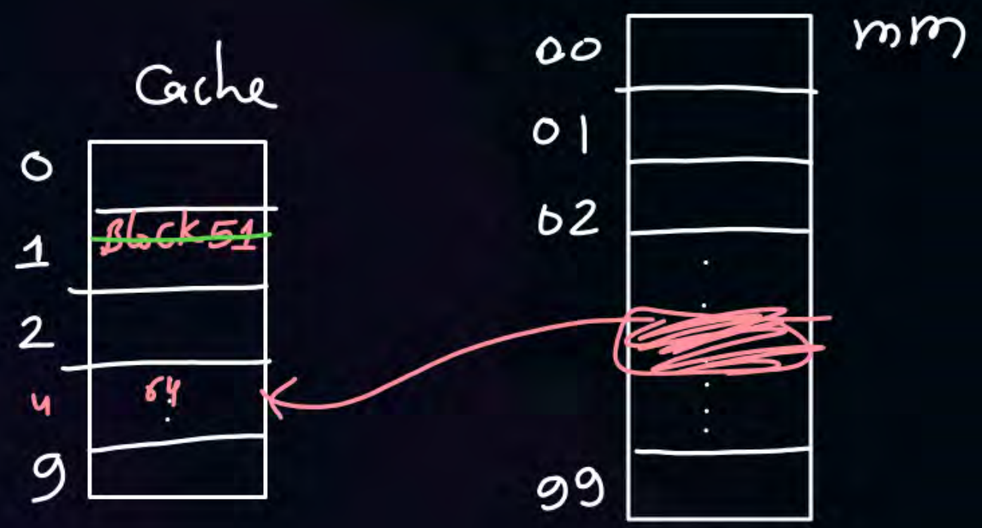
# Topic : Cache Mapping

↳ mapping is applied on blocks.

- Direct Mapping
- Set Associative Mapping
- Fully Associative Mapping

# Topic : Direct Mapping

- Blocks in cache = 10 (0-9)
- Blocks in Main memory = 100 (00-99)




# Topic : Direct Mapping

| Cache | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ~~Block 51~~ Block 91 | | | Block 64 | | | | | |

| Main Memory | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
| | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| | : | : | : | : | : | : | : | : | : | : |
| | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 |

# Topic : Direct Mapping

cache mem. block no. = (mm block no.) % no. of blocks in cache

# Topic : Direct Mapping

| CPU Request (MM block) | Mapping (CM block no.) | Hit /Miss | Comments |
|---|---|---|---|
| 51 | 51 % 10 = 1 | Miss | Bring block 51 of mm into cm at block 1. |
| 64 | 64 % 10 = 4 | Miss | Bring block 64 of mm into cm at block 4. |
| 91 | 91 % 10 = 1 | Miss | Bring block 91 of mm into cm at block 1, by replacing block 51 |

# Topic : Direct Mapping


Tag ⇒ Tag is stored to identify which block among all competitors is present in cache (currently mapped in CM)

mm block no.

| Tag | CM block no. |
|---|---|

ex:- mm block no. 51

| 5 | 1 |
|---|---|

5 → Tag

1 → CM block no.

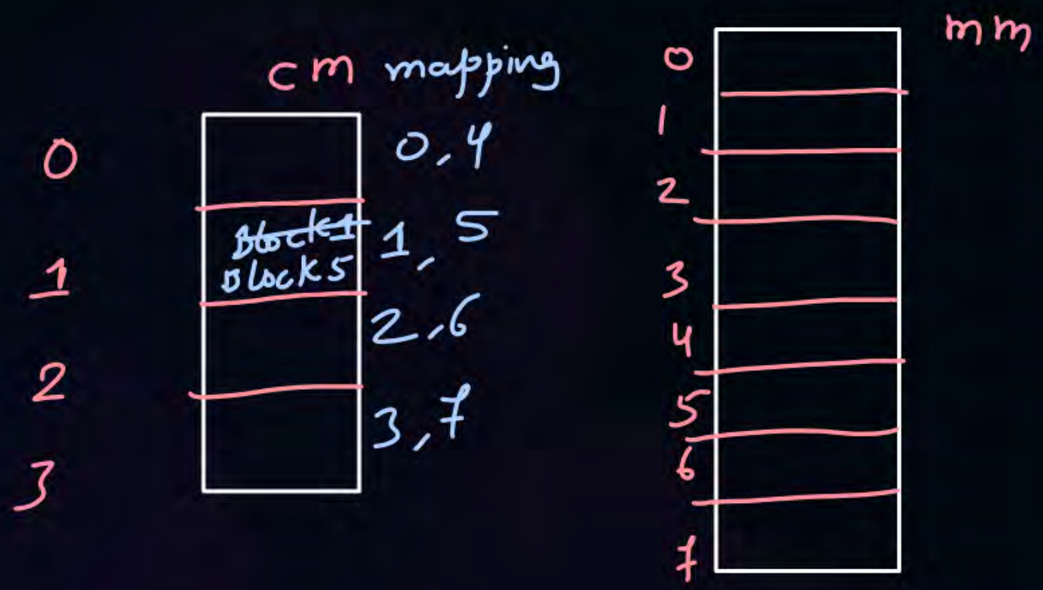
| CPU Request for mm block no. | Cm block no. | Tag | Hit/Miss | Comment |
|---|---|---|---|---|
| 51 | 51 % 10 = 1 | 5 | Miss | Bring block 51 of mm into Cm at block 1 with tag 5 |
| 91 | 91 % 10 = 1 | 9 | Miss | Bring block 91 of mm into cm at block 1, by replacing block 51 and update tag to 9. |

goto cm block no. 1 & check Tag to know which block of mm is present there.

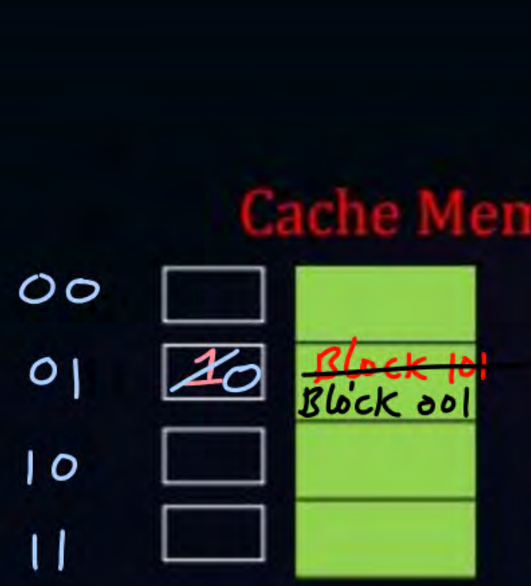
Tag in cm for present block = 5 but CPU needed block with tag = 9

# Topic : Direct Mapping

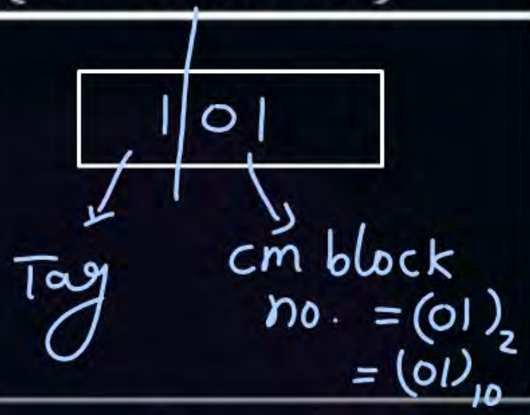
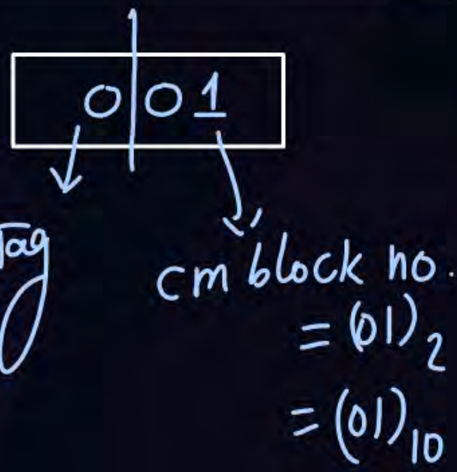
- Blocks in cache = 4 (00-11)
- Blocks in Main memory = 8 (000-111)




# Topic : Direct Mapping




# Topic : Direct Mapping

| CPU Request (MM block) | Mapping (CM block no.) | Hit /Miss | Comments |
|---|---|---|---|
| $(5)_{10} = (101)_2$ |   | Miss | Bring mm block $(101)_2$ into cm at block $(01)_2$ with tag 1 |
| $(1)_{10} = (001)_2$ |   | Miss | Bring mm block $(001)_2$ into cm at block $(01)_2$ by replacing block $(101)_2$ and update tag by 0. |

Actually CPU generates mm address which is add. of a byte.

⇓

Hence we will derive mm block no. from given mm add.

⇓

do the same as explained to search in cache for hit/miss

# 2 mins Summary

**Topic** Cache Mapping

**Topic** Direct Mapping

**Topic** Tag & Index

# Happy Learning

# THANK - YOU