5 Memory Hierarchy Design and Cache Memory

5.1 MEMORY ORGANISATION

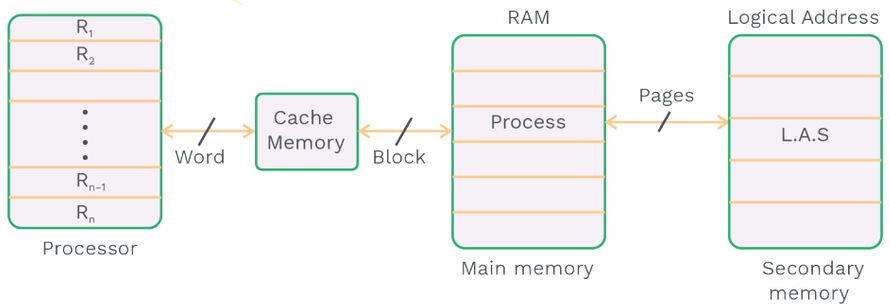


Fig. 5.1 Memory Organisation

Memory hierarchy:

Why is memory essential?

- Memory is an essential component of a computer since it's used for storing programs and necessary data.

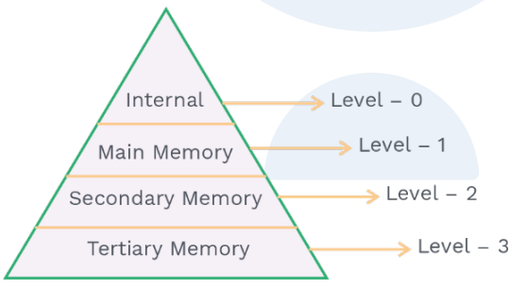


Fig. 5.2 Memory Heirarchy

- Internal memory consists of registers and cache.
- Main memory consists of RAM and ROM.
- Secondary memory consists of a Hard disk.
- Tertiary memory consists of tape.

Representation of memory:

1) In terms of storage:

Storage capacity of memory will increase from level 0 to level 3.

2) In terms of access time:

Access time of the memory is increased from level 0 to level 3. The access time of internal memory is very less negligible as compared to tertiary memory.

3) In terms of speed:
Access speed of memory decreases from level 0 to level 3. Access speed of internal memory is very fast as compared to others. Level 0 memory is very close to the CPU.

4) In terms of frequency access and per bit storage cost:
Frequency decreases from level 0 to level 3. The frequency of access level–0 is more as compared to other levels.
Per bit storage cost of level 0 to level 3 is decreased. Level–0 cost is high and level–3 cost is low.

$$\text{Average memory access time} = \frac{(\text{Fetch time} + \text{Write time}) \text{ of all instructions}}{\text{Total instructions}}$$

$$\text{Performance}(P) \propto \frac{1}{\text{Access Time}(A_T)}$$

Total time = Hit/miss latency + Access time (A_T) + Transfer Time
Hit/miss latency is negligible, and transfer time depends upon B.W. (Bandwidth)

Types of memory organisation:
Memory organisation is divided into two parts based on accessing order.
a) Hierarchical access memory organisation
b) Simultaneous access memory organisation

To understand the concept of hierarchical/simultaneous, we need to understand the concept of Hit and Miss.

Hit:
The successful access of data from cache/main memory is called hit.

Hit ratio (H):
It is the ratio of the number of successful memory access to the total number of attempts to access memory.

$$\text{Hit ratio} = \frac{\text{Number of hits}}{\text{Total number of access}} = \frac{\text{Number of hits}}{\text{Number of hits} + \text{number of misses}}$$

Miss:
The unsuccessful access of data from cache/main memory is called miss.

Miss ratio (M):

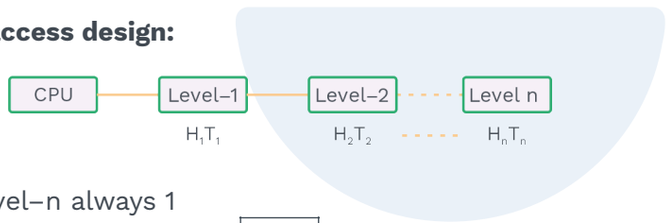
$$\boxed{1 - \text{Hit ratio}(H)}$$

a) Hierarchical access:

In a hierarchical organisation, the CPU always accesses the data from level 1 cache only.

If data is not present in level–1 cache memory, then the data is copied from level–2 to level–1. If data is not present in the level–2 cache, then data is copied from level–3 to level–2 and level–2 to level–1, then the CPU will access the data from level–1 cache, or we can say data is copied from higher (higher number of levels) level to level–1 memory.

Hierarchical access design:



Hit ratio at level–n always 1

$$\boxed{H_n = 1}$$

$$\text{Average access time } (T_{avg}) = H_1T_1 + (1-H_1)H_2\left(\overbrace{T_2}^{\text{Search Time}} + T_1\right) + (1-H_1)(1-H_2)H_3$$

$$\left(\underbrace{T_1 + T_2}_{\text{Search time}} + T_3\right) + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + \ldots\ldots\ldots\ldots + (1-H_1)(1-H_2)\ldots\ldots(1-H_{n-1})H_n(T_1 + T_2 + \ldots T_n)$$

H_1 = Hit ratio at level–1

H_2 = Hit ratio at level–2

.

.

H_n = Hit ratio at level–n

T_1 = Access time at level–1

T_2 = Access time at level–2

.

.

T_n = Access time at level–n

T_{avg} = Total time to access 1 word data from memory.

PRACTICE QUESTIONS

Q1 **Consider a system with two level cache access time of level 1, level 2, and main memory are 10 ns, 70 ns and 600 ns, respectively. If there is miss in level–1 cache then CPU copied the data from level–2 to level–1 and access it. Hit ratio of level 1 and level 2 is 0.6 and 0.8, respectively. What is the average memory access time?**

Sol: **86**

Hit rate level 1 = 0.6

Hit rate level 2 = 0.8

Access time L_1 = 10 ns

Access time L_2 = 70 ns

Access time of main memory T_m = 600 ns

$T_{avg} = H_1T_1 + (1–H_1)H_2\ (T_2+T_1) + (1–H_1)\ (1–H_2)H_3 \quad (T_m+T_2+T_1)$

$T_{avg} = 0.6 * 10 + 0.4 * 0.8 * 80 + 0.4 * 0.2 * 680$

$T_{avg} = 6 + 25.6 + 54.4$

$T_{avg} = 86$ ns

Q2 **Consider a three-level memory architecture specified below:**

Level	Access time / word	Block Size	Hit Ratio
L_1	30 ns	–	0.75
L_2	50 ns	2	0.7
L_3	100 ns	4	–

CPU accesses a block from L_1 memory. If it is a miss the block is copied from L_2 to L_1. If the block is not found in L_2, the sound block is copied from L_3 to L_2 and then L_2 to L_1. What is the effective memory access time?

a) 43 ns **b) 47 ns** **c) 45 ns** **d) 41 ns**

Sol: **d)**

Clearly, the above architecture is hierarchical one.

Effective Access Time(EAT) = $H_1T_1 + (1 – H_1)H_2(T_1 + T_2) + (1 – H_1)(1 – H_2)H_3(T_1 + T_2 + T_3)$

Now, T_1 = 30 ns (Default block size = 1 word)

T_2 = 100 ns (Block size = 2 words)

T_3 = 400 ns (Block size = 4 words)

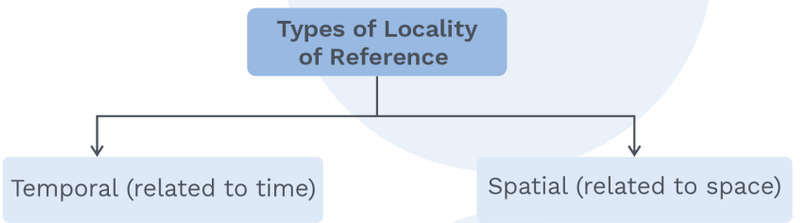
EAT = 0.95 × 30 + 0.05 × 0.7 × 130 + 0.05 × 0.3 × 1 × 530

[∵ There is always hit in L_1]

= 41 ns

Locality of reference:

When the CPU has requested one address for memory access, then that particular address or nearby address will be accessed soon, known as the locality of reference. Based on locality of reference, average memory access time decreases because programs tend to reuse data and instructions they have used recently.



Temporal locality:

If an item is referenced in memory, it will tend to be referenced again soon because of loops. LRU takes care of temporal locality.

Spatial locality:

If an item is referenced in memory, nearby items will tend to be referenced soon because, during data access, their adjacent data words are accessed in a sequence. Block size takes care of spatial locality.

Simultaneous access memory organisation:

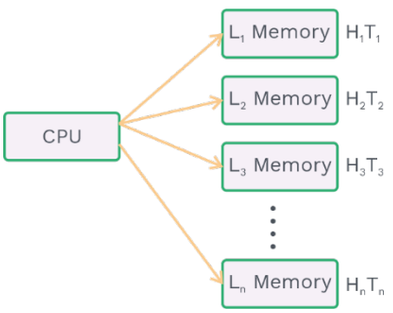


Fig. 5.3 Simultaneous Access Memory Organization

In this organisation, a CPU is directly connected to all levels of memory, CPU can directly access the data from L_i memory, L_i means the level where the data is present. Here we can't copy the data from level–n to level–1. If there is a miss in level 1 the then CPU can directly access the data from level–2 if not in level–2 then the CPU can directly access the data from level–3 and so on. Search time is ignored.

Hit ratio at level–n is 1.

$$\text{Hit ratio} = \frac{\text{Number of hits}}{\text{Total no. of access}}$$

Assume "T" represents access time and "H" is hit ratio then,

Relation:

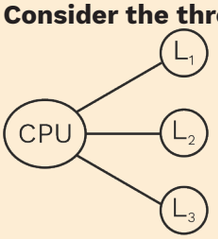
$$\text{Average access time} = T_{avg} = H_1T_1 + (1-H_1)H_2T_2 + (1-H_1)(1-H_2)H_3T_3 + \ldots\ldots$$
$$\ldots + (1-H_1)(1-H_2)\ldots\ldots(1-H_{n-1})H_nT_n$$

Throughput of memory in WPS (Words Per Second)

$$\eta_{memory} = \frac{1}{T_{avg}} \text{ WPS}$$

PRACTICE QUESTIONS

Q3 **Consider the three-level memory configuration as:**



L_1 Hit ratio = 0.8, T_{avg}/block = 30 ns

L_2 Hit ratio = 0.7, T_{avg}/block = 40 ns

L_3 Hit ratio = 1, T_{avg}/block = 50 ns

If there is a miss in L_1 memory, CPU accesses the block from the higher levels without copying the block from higher to lower levels of memory. What is the estimated effective memory access time for this configuration _____ns.

Sol: **32.6**

$T_{avg} = H_1T_1 + (1 - H_1)\ H_2\ T_2 + (1 - H_1)\ (1 - H_2)\ H_3\ T_3$

$= 0.8 \times 30 + 0.2 \times 0.7 \times 40 + 0.2 \times 0.3 \times 1 \times 50$

$= 32.6$ ns

Q4 **In a two-level simultaneous memory organization the speed of level-1 is 8 times more than level-2 memory. CPU always accesses the requested block from level-1 memory first. If the access time of level-1 memory is 30 ns less than the effective memory access time. What is the hit ratio of level-1 memory? (Assume, T_1 = 40 ns, where T_1 = Access time of level-1 memory).**

a) 0.64 **b) 0.89** **c) 0.95**

d) 0.75

Sol: **b)**

$\Rightarrow T_1 = 40$ ns

$\Rightarrow T_{avg} = (40 + 30)$ ns $= 70$ ns

Again, level-1 memory is 8 times faster than level-2 memory.

$\Rightarrow T_2 = 40 \times 8 = 320$ ns

Now, since it's a simultaneous memory model-

$\Rightarrow T_{avg} = HT_1 + (1 - H_1)\ H_2\ T_2$

$\Rightarrow 70 = H_1 \times 40 + (1 - H_1)\ 1 \times 320$

$\Rightarrow 70 + 40H_1 + 320 - 320H_1$

$\Rightarrow 280H_1 = 250$

$\Rightarrow H_1 = 25/28 = 0.89$

Cache memory:

- The speed of the main memory is very slow compared to modern processors. Cache memory is very fast, and the smallest component between CPU and main memory.
- Cache memory plays a very important role in reducing average memory access time and increase the overall throughput of the system.
- The potency of the working of the cache mechanism is established upon the principle of locality of reference. The locality of reference we have discussed previously in hierarchical access design.

Another way of defining cache memory: Based on the locality of reference concept, all the current demanded localities are kept in a smaller and faster memory called as cache.

Previous Years' Question

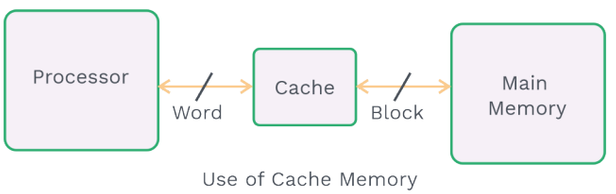
Consider a system with 2 level caches. Access times of level 1 cache, level 2 cache and main memory are 1 ns, 10 ns, and 500 ns, respectively. The hit rates of level 1 and level 2 caches are 0.8 and 0.9, respectively. What is the average access time of the system ignoring the search time within the cache? (Use Hierarchy memory access time)

a) 13.0 ns **b)** 12.8 ns
c) 12.6 ns **d)** 12.4 ns

Sol: c) **(GATE-2004–1 Mark)**

Performance of cache memory:
Performance of cache memory depends on several segments:

- Cache size
- Cache block size
- Number of levels of cache
- Cache mapping technique
- Cache replacement policy
- Cache update policy

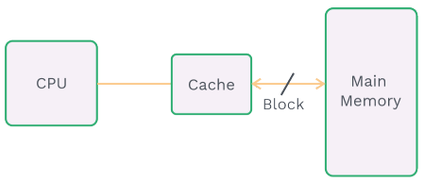


Use of Cache Memory

Fig. 5.4

Cache size:
Cache size is the amount of main memory data it can add.

Cache block size:
The number of data words/bytes in each cache block is defined as cache block size. It is always in power of 2.



(Main memory transferring the data in terms of Block)

Fig. 5.5

5.2 MULTI–LEVEL CACHE

- Multilevel cache is used to decrease/reduce the miss penalty and improve the performance in the system.
- With the help of a two–level or three–level cache (i.e., multilevel cache organization) we can reduce the miss penalty. If data is not present in the lower level cache, then the time required to transfer the data from the higher level to lower-level memory is called a miss penalty.

Definitions

Time taken to service a miss is known as miss penalty.

- Some designs are important to understand the topic.
- Suppose the CPU wants 15 memory references (MR), then calculates the access time.

System design without cache:

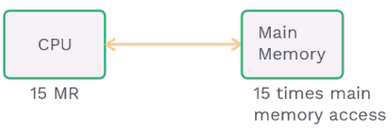


Fig. 5.6

System design With Cache:

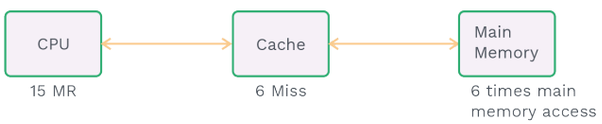


Fig. 5.7

System design with multilevel cache:

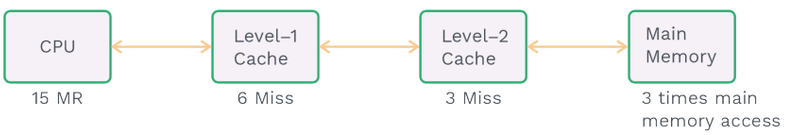


Fig. 5.8

- This system design with a multilevel cache accesses the main memory 3 times. So miss the penalty is less here comparatively.

Important points:

- Level–1 cache memory size is always a subset of level–2 cache memory size.
- Level–1 cache performance is always better than level–2 because level–1 the cache memory is very close to the CPU.
- Level–1 memory access time is less than the level–2 memory access time because level–1 cache is close to the CPU, so the CPU can access the data in the level–1 cache in no time.
- In the multilevel cache organisation, two types of miss rates we calculate:
  a) Global Miss Rate (GMR)
  b) Local Miss Rate (LMR)

Global miss rate (GMR):
Global miss rate is the ratio of total misses in cache memory (CM) and total number of CPU generated memory references at each level.

$$\text{Global Miss Rate (GMR)} = \frac{\text{Number of Miss Operation in CM}}{\text{Total Number of Memory References generated by CPU}}$$

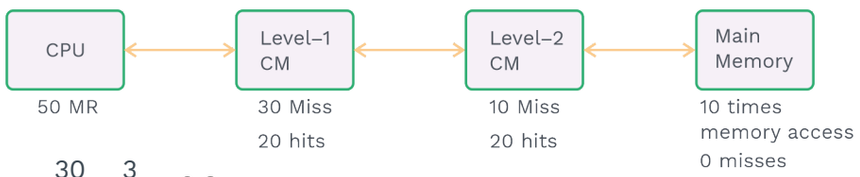
Local miss rate (LMR):
Local miss rate is the ratio of total miss operation in cache memory (CM) to total no. of access the cache at each level.

$$\text{Local Miss Rate (LMR)} = \frac{\text{Number of Miss Operation in CM}}{\text{Total Number of Access to the CM}}$$

PRACTICE QUESTIONS

Q5 **Consider two–level cache CPU generated 50 memory references. 30 miss operation present in level–1 cache and 10 miss operation, present in level–2 cache. Then calculate:**
a) Global miss rate at level–1 (GMR L_1)
b) Local miss rate at level–1 (LMR L_1)
c) Global miss rate at level–2 (GMR L_2)
d) Local miss rate at level–2 (LMR L_2)

Sol: $$\text{GMR } L_1 = \frac{\text{Number of Misses Operation at level} - 1}{\text{Total number of Memory References generated by CPU}}$$



$$GMRL_1 = \frac{30}{50} = \frac{3}{5} = 0.6$$

$$LMRL_1 = \frac{\text{Numbeer of Misses Operation at level} - 1 \text{ cache memory}}{\text{Total Number of access to the CM}}$$

$$LMRL_1 = \frac{30}{50} = \frac{3}{5} = 0.6$$

$$GMRL_2 = \frac{\text{Number of Misses Operation at level} - 2}{\text{Total number of Memory Generated by CPU}}$$

$$= \frac{10}{50} = \frac{1}{5} = 0.2$$

$$LMRL_2 = \frac{\text{Number of Misses Operation at level} - 2 \text{ cache memory}}{\text{Total number of Access to the CM}}$$

$$= \frac{10}{30} = \frac{1}{3}$$

$$= 0.33$$

Previous Years' Question

Consider a two–level cache hierarchy L1 and L2 caches. An application incurs 1.4 memory accesses per instruction on average. For this application, the miss rate of L1 cache 0.1, the L2 cache experience on average, 7 misses per 1000 instructions. The miss rate of L2 expressed correct to two decimal places is _____.

Sol: [0.05] **(GATE-2017 (Set–1) 2 Marks)**

Formula:

How to calculate the average memory access time in terms of the hit time, miss penalty and the local miss rate for 2 level cache memory.

$$\text{Average Memory Access Time}\left(T_{avg}\right) = \text{Hit time level} - 1 + \left(\begin{matrix}\text{Miss rate} \\ \text{level} - 1\end{matrix} * \begin{matrix}\text{Miss Penalty} \\ \text{level} - 1\end{matrix}\right)$$

$$\text{Miss Penalty level} - 1 = \text{Hit time level} - 2 + \left(\frac{\text{Miss rate}}{\text{level} - 2} * \frac{\text{Miss Penalty}}{\text{level} - 2}\right)$$

$$\text{Miss Penalty level} - 2 = \text{Main memory access time}$$

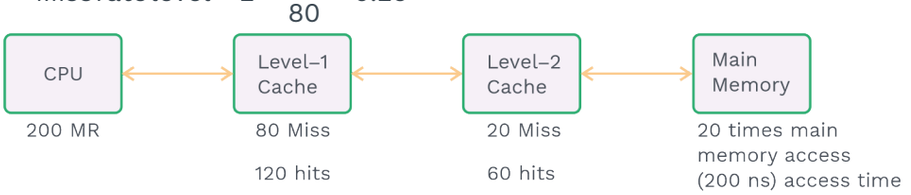
- Miss penalty level–2 is same as the main memory access time, because in the level–2, total number of miss operations is same as the number of time main memory access.

PRACTICE QUESTIONS

Q6 **Consider two level cache CPU generates 200 memory reference (MR). 80 miss operation present at level–1 cache and 20 miss operation present at level–2 cache. Hit time at level–1 and level–2 cache is 40 and 90 ns, respectively and main memory access time is 200 ns. What is the average memory access time? (Use the concept of local miss rate)**

Sol: **96**

- Level–1 miss operation = 80
- level–2 miss operation = 20
- CPU generated = 200 MR
- Hit time level–1 = 40 ns
- Hit time level–2 = 90 ns
- Main memory access time = 200 ns
- $\text{Using local miss rate concept, Miss rate level} - 1 = \frac{80}{200} = 0.4$
- $\text{Miss rate level} - 2 = \frac{20}{80} = 0.25$



$$T_{avg} = \text{Hit time level} - 1 + \left(\frac{\text{Miss rate}}{\text{level} - 1} * \frac{\text{Miss penalty}}{\text{level} - 1}\right)$$

$$= 40 + \left[0.4 * \left[\begin{matrix}\text{Hit time} \\ \text{level} - 2\end{matrix} + \begin{pmatrix}\text{Miss rate} * \text{Miss penalty} \\ \text{level} - 2 \quad \text{level} - 2\end{pmatrix}\right]\right]$$

As we know,

$$(\text{Miss penalty})_{L2} = (\text{Hit time})_{L2} + \left((\text{Miss rate})_{L2} * (\text{Miss penalty})_{L2}\right)$$

$$= 40 + \left[0.4 * \left[90 + (0.25 * \text{Main memory accessy time})\right]\right]$$

$$= 40 + \left[0.4 * \left[90 + (0.25 * 200)\right]\right]$$

$$= 40 + \left[0.4 * [90 + 50]\right]$$

$$= 40 + 56$$

$$= 96 \text{ ns}$$

Q7 **Consider a two-level cache system. CPU generates 1500 block request out of which 900 requests can be satisfied through level-1 cache. The time taken to access a block from level-1 cache is 4 ns. 400 out of the remaining requests are available in level-2 cache. Time taken by CPU to access a block from level-2 is 12 ns. If miss occurs at both level caches, CPU accesses the block from main memory in 18 ns. Then, the effective memory access time is _____ns.**

Sol: **8**

$$\text{EMAT} = \text{Hit time}_{L_1} + \text{Miss rate}_{L_1} * \text{Miss Penalty}_{L_1}$$

$$= \frac{900}{1500} * 4\text{ns} + \left(\frac{600}{1500} * \left(\text{Hit time}_{L_2} + \text{Miss rate}_{L_2} * \text{Miss Memory access time}\right)\right)$$

$$= \frac{900}{1500} * 4\text{ns} + \left(\frac{600}{1500} * \left(\frac{400}{600} * 12 + \frac{200}{600} * 18\right)\right)\text{ns}$$

$$= \frac{120}{15}\text{ns}$$

$$= 8\text{ns}$$

5.3 MAPPING TECHNIQUE:

Transferring the data (block) from the main memory to the cache memory is called mapping. There are three types of mapping:

a) Direct Mapping
b) Set Associative Mapping
c) Fully Set Associative Mapping

Note:

To denote cache memory, we will use shorthand notation as CM.

Direct mapping:

Basics:

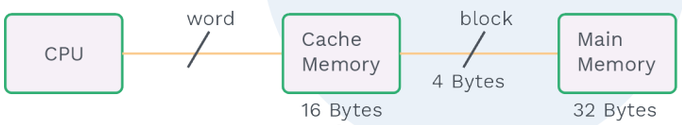


Fig. 5.9

Cache memory organization:

Let's say, cache memory size = 16 B
Block size = 4 B

$$\text{No. of blocks (lines) in CM} = \frac{\text{CM size}}{\text{Block size}} = \frac{16}{4} = 4 \text{ lines}$$

Total 16 Bytes of memory so cache index = $\log_2 16 = 4$ bits

Main memory organisation:

$$\text{No. of blocks in main memory} = \frac{\text{Main memory size}}{\text{Block size}} = \frac{32 \text{ Bytes}}{4 \text{ Bytes}} = 8$$

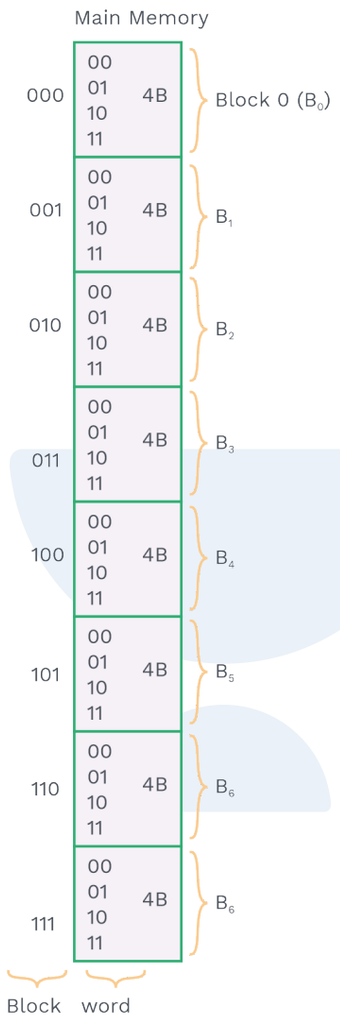
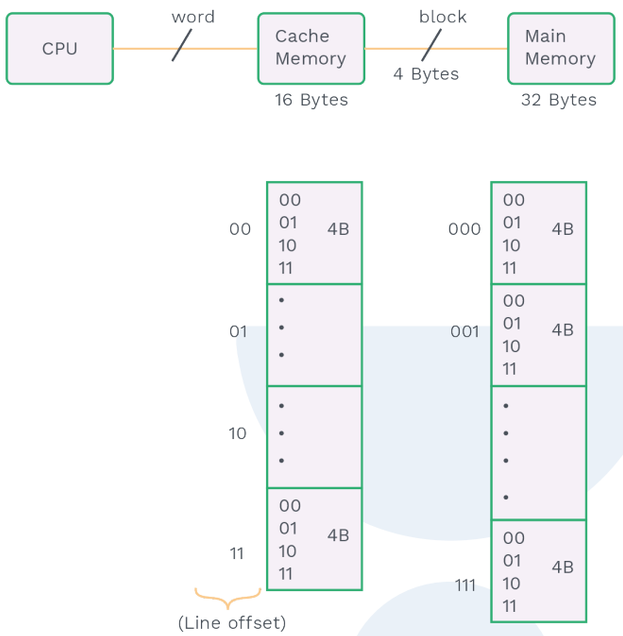


Fig. 5.10 Example of Block Organization of Main Memory

Physical address space = 32 bytes
Physical address = $\log_2 32$
= $\log_2 2^5$
= 5 bits

Note:

Mapping



"Copy" the data from MM to CM

Definition of direct mapping:

To map the data block from main memory to cache memory "Modulo (MOD)" function is used.

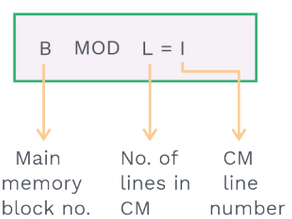
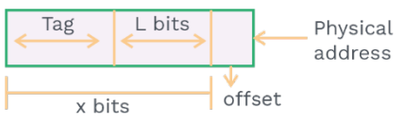


Fig. 5.11 Direct Mapping Function

Example:

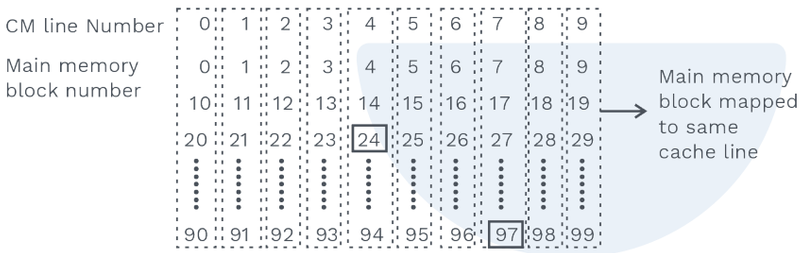
- No. of blocks in Main Memory = 2^x
- No. of cache lines = 2^L

2^x % 2^L will give least significant L bits from x bits which is nothing but the cache line number.



Example:

- CM line = 10 (0 to 9)
- Main memory blocks = 100 (0 to 99)



- Block 24 is mapped to cache line no. 4

24 mod 10 = 4

- Block 97 is mapped to cache line no. 7

97 mod 10 = 7

Calculate Word Offset, Line Offset and Tag

Block size = 4B

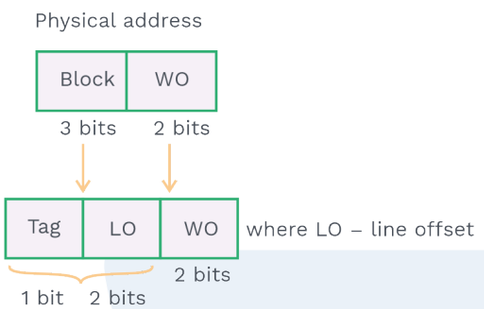
Number of blocks = 8 (0 to 7)

Number of lines = 4 (0 to 3)

Physical address = Block address + Word offset

= 3 + 2

= 5 bit



Note:

By default, we consider memory as byte addressable.

Note:

If some additional bit is given like valid, invalid, dirty, write back (updation) bits.
All these bits we add in tag space

Important formula:

$$\text{Cache memory size} = \text{No. of lines in cache memory} \times \text{Block size}$$

$$\text{Tag memory size} = \text{No. of lines in CM} \times \text{Tag space in the line}$$

$$\text{Cache index} = \text{Line offset} + \text{Word offset}$$

PRACTICE QUESTIONS

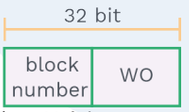
Q8 **Consider 16KB direct mapped cache organized into a 2KB data blocks. Physical address of memory is 32 bits. What is the tag size in bits and line offset(LO)? [Assume memory is byte addressable]**

a) 18, 3 **b) 21, 3** **c) 18, 11** **d) 3, 11**

Sol: **a)**

- Direct mapped cache
- Physical address = 32 bits
- Block size = 2KB = 2×2^{10} B
- Line offset = ?

Physical address format:

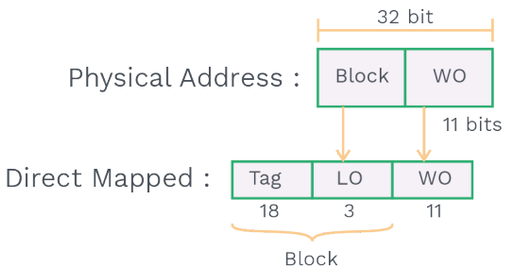


word offset (WO) = $\log_2 2^{11}$ = 11 bits

No. of lines (L) $= \frac{\text{Cache size}}{\text{Block size}}$

$= \frac{16\,\text{KB}}{2\,\text{KB}} = 8$

Line offset (LO)) = $\log_2 8$ = 3



Tag = 18 bits

LO = 3 bits

Hence option (A) is correct

Q9 **An 64KB direct mapped cache organize into 32 word blocks, word length of the CPU is 16 bits. CM data is subset of main memory of 8GB each tag is comprising with 1 valid bit and 1 update bit. What is size of tag directory and cache memory size in cache controller?**

a) 19456 bits, 32 MB **b) 19456 bits, 64 KB**

c) 9728 bits, 64 KB **d) None of these**

Sol: **b)**

- Direct mapped cache
- Cache memory size = 64 KB
- Block size = 32 Words
- Word length = 16 bit

So, Block size $= \frac{32 * 16}{8} = 64\,\text{Bytes}$

- Main memory size = 8 GB = $(2^3 \times 2^{30})$ Bytes
- Physical address = $\log_2 2^{33}$ = 33 bits
- Valid bit = 1
- Update bit = 1

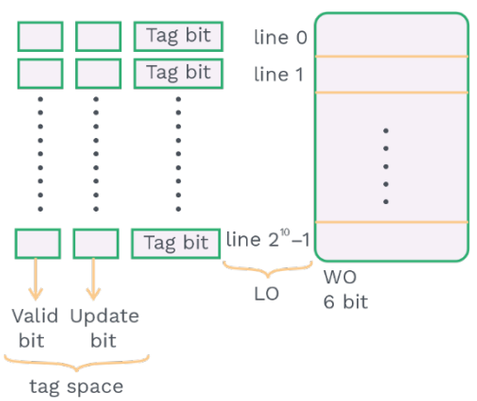
No. of lines in cache memory (L) $= \frac{\text{CM size}}{\text{block size}}$

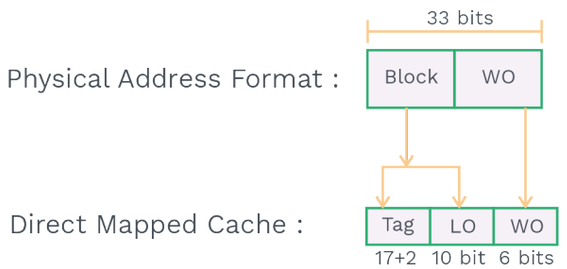
$= \frac{64\,\text{KB}}{64\,\text{bytes}}$

$= 1\text{K}$

$= 2^{10}$

Cache memory:





Tag directory size = Number of lines in cache memory * Tag space
= 2^{10} * 19 = 1024 * 19
= 19456 bits

Cache memory size = Number of lines in Cache Memory * block size
= 2^{10} * 2^{6}
= 64 KB

Hence option (B) is correct

Previous Years' Question

Consider a direct mapped cache of size 32 KB with block size 32 bytes. The CPU generates 32 bit addresses. The number of bits needed for cache indexing and the number of tag bits are respectively

a) 10, 17 **b)** 10, 22 **c)** 15, 17 **d)** 5, 17

Sol: c) **(GATE-2005 (2 Marks))**

Previous Years' Question

An 8KB direct–mapped write–back cache is organized as multiple blocks, each of size 32–bytes. The processor generates 32–bit address. The cache controlled maintains the tag information for each cache block comprising of the following.

1 Valid bit

1 Modified bit

As many bits as the minimum needed to identify the memory block mapped in the cache. What is the total size of memory needed at the cache controlled to store meta–data (tags) for the cache?

a) 4864 bits **b)** 6144 bits

c) 6656 bits **d)** 5376 bits

Sol: d) **(GATE-2011 (2 Marks))**

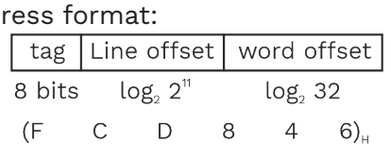
Q10 **Consider a 32-bit CPU which supports 64KB direct mapped cache. The cache is organized into 8 word blocks. The memory address $(FCD846)_H$ is mapped to cache line number-**

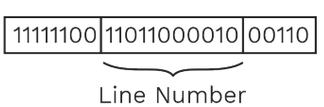
a) $(6C2)_H$ **b)** $(7C5)_H$ **c)** $(7C2)_H$ **d)** $(6C5)_H$

Sol: **a)**

Number of blocks in cache memory = $\frac{64\,KB}{8 \times 4\,B}$ = 216–5 = 211

Address format:





Line Number = 110 1100 0010

6 C 2

= $(6C2)_H$

5.4 HARDWARE DESIGN OF DIRECT MAPPING

Hardware implementation in direct mapping:

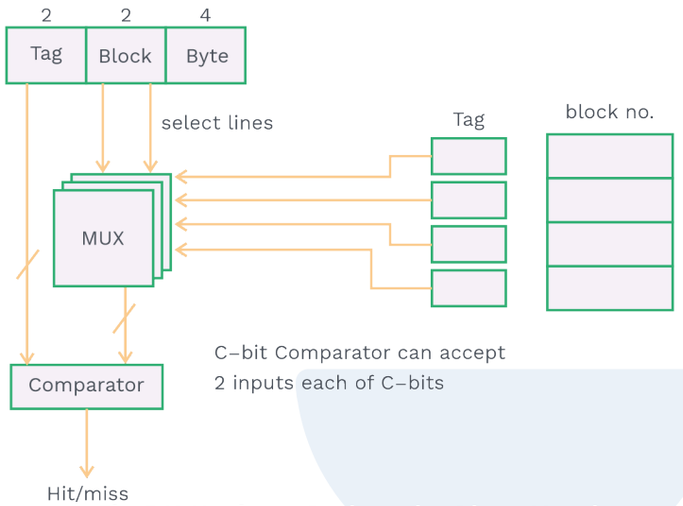


Fig. 5.12 Hardware Design using Direct Mapping

Number of select lines = Number of bits required for block number
Number of input lines = Number of blocks in CM
"C" bits comparator required
C → tag bits
here, C = 2
Number of MUX required for selecting the tag = tag bits
Size of MUX = Number of blocks in cache: 1

Previous Years' Question

Consider a machine with byte addressable memory of 2^{32} bytes divided into blocks of size 32 bytes. Assume a direct mapped cache having 512 cache lines are used with this machine. The size of the tag field in bits is
Sol: 18 **(GATE-2005 (2 Marks))**

Hit latency in direct mapping:
Total time to check whether the block hit or miss is comparator time and MUX time, is known as hit latency.

$$\text{Hit latency} = \text{Multiplexer delay} + \text{Comparator delay}$$

Set associative mapping:
To check hit/miss in set associative mapping, three things are required:

a) Multiplexer:
This mapping requires R–way sets of MUX. In each set number of MUX based on the number of sets.

b) Comparator:
In this mapping R(R–way) comparator is needed.

c) OR gate:
To check hit/miss 1 or gate, it is needed to add comparator bits.

Set associative mapping
Set associative mapping is made with the combination of direct and associative mapping. Associative mapping is a much more flexible mapping method because the main memory block has 'S' choice to place in cache memory (S means S–way set associative).

Design of 2–way set associative:

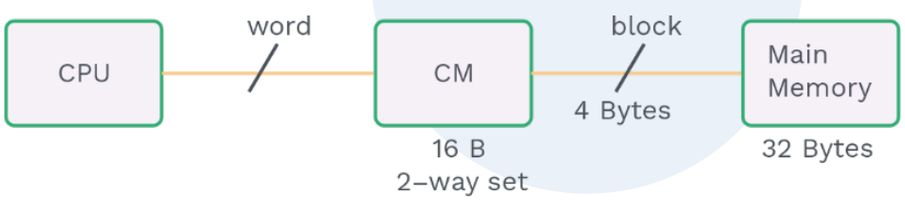


Fig. 5.13

- No. of blocks in main memory = $\frac{\text{Main memory size}}{\text{Block size}}$

$$= \frac{32\,B}{4\,B}$$

$$= 8$$

- No. of lines in cache memory (L) = $\frac{\text{CM size}}{\text{Block size}}$

$$= \frac{16\,B}{4\,B}$$

$$= 4$$

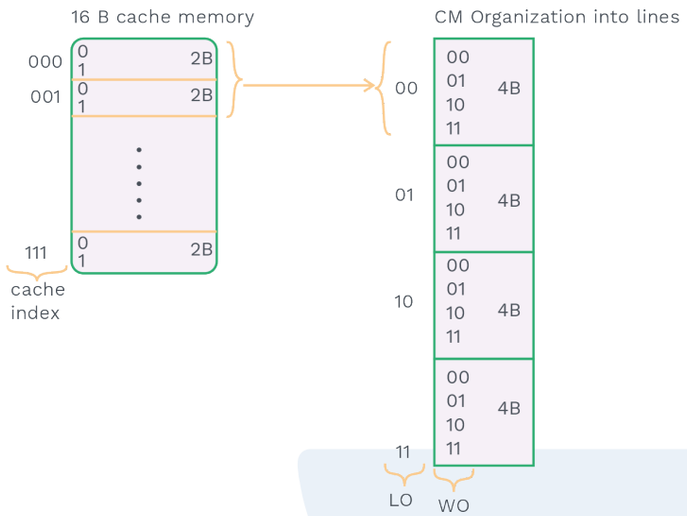


Fig. 5.14

CM organized into sets:

$$\text{No. of sets}(S) = \frac{L}{R - \text{way}} = \frac{\text{No. of lines}}{R}$$

$$= \frac{4}{2}$$

$$= 2$$

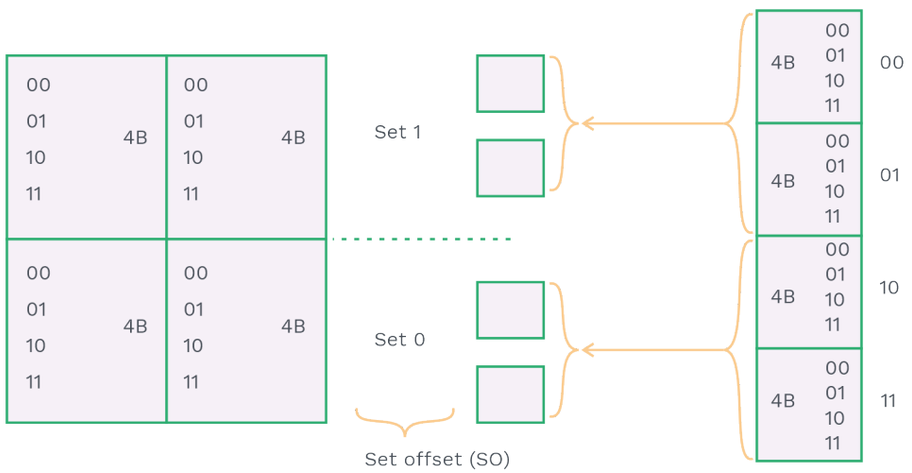


Fig. 5.15

Mapping function:

We use the mod function to map the main memory (MM) block number.

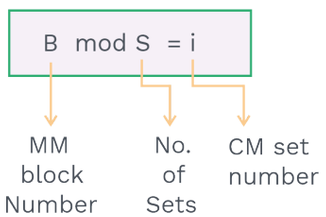
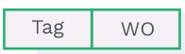


Fig. 5.16 Set Associative Mapping Function

- Main memory address: 

- Main memory = 32 Bytes, main memory address in bits = 5 bits
- Block size = 4B, Word offset = 2 bits
- No. of sets = 2, set offset = $\log_2 2 = 1$

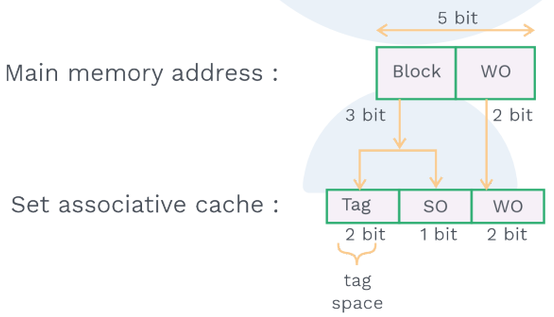


Fig. 5.17

Formulas:

$$\text{Tag directory size} = \text{Number of blocks in CM} * (\text{Tag bits} + \text{extra bit if present})$$

PRACTICE QUESTIONS

Q11 **Consider a cache with capacity of 64 KB, 8–way set associative with block size of 32 bytes. The processor sends 32 bit address to the cache controller, each tag directory entry contains 2 valid bits and 1 replacement bit. Then the size of the cache tag directory is _____ K bits.**

Sol: **44**

- 8–way set associative mapping
- CM size = 64 KB
- Block size = 32 bytes
- Physical address = 32 bits
- Number of extra bits = 2 + 1 = 3 bits

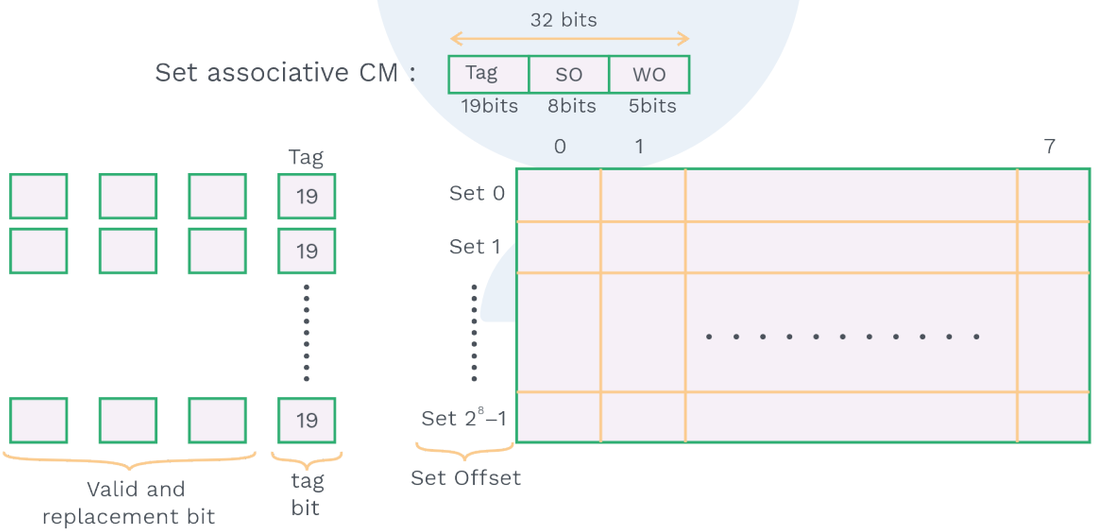
Number of lines (L) = $\frac{\text{CM size}}{\text{Block size}} = \frac{64\,\text{KB}}{32\,\text{B}} = 2^{11}$

Number of sets (S) = $\frac{L}{R - \text{way}}$

$= \frac{2^{11}}{8}$

$= 2^8$

Set associative CM :



Tag directory size = No. of blocks in CM * (Tag + extra) bits
= No. of lines * (19 + 3) bits
= 2^{11} * 22
= 44 K bits

Q12 **Consider 4–way set associative cache with the capacity of 512 byte. CPU generates 32 bits physical address and block size is 16 bytes then calculate the following term.**
a) No. of bits in the tag field

b) Size of tag directory

Sol: **800**

4–way set associative

- CM size = 512 bytes
- Block size = 16 bytes
- Physical address = 32 bits
- No. of lines (L) in cache = $\frac{\text{Cache size}}{\text{Block size}} = \frac{512\,B}{16\,B} = 32$

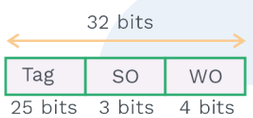
No. of sets (S) = $\frac{L}{R - way}$

$= \frac{32}{4}$

$= 8$

Set offset = $\log_2 8$
= 3 bits

Set associative cache :



a) tag space = 25 bits
b) Tag directory size = No. of lines (in CM) * Tag space
= 32 * 25
= 800 bits

Q13 **Consider a system implementing a 2-way set associative cache with 27 blocks. It uses least recently used policy for replacement. Conflict misses occur when two or more blocks contend for the same cache set. First refer-**

ence miss occurs when a block is accessed for the very first time. The memory access sequence (0, 64, 128, 64, 0, 64, 128, 64, 165, 129, 65, 1, 65, 129, 65) is repeated 2 times. The number of cache conflict misses and miss ratio in the cache are_____, _____.

a) 16, 50.00% **b) 10, 62.50%**

c) 14, 47.25% **d) 18, 56.25%**

Sol: **c)**

Number of sets = $\frac{128}{2} = 64$.

0	~~0~~ ~~128~~ ~~0~~ 128	64
1	~~1~~ ~~129~~ ~~1~~ 129	65
2		
.	.	.
.	.	.
.	.	.
63		

→

0	~~0~~ ~~128~~ ~~0~~ ~~128~~ ~~0~~ ~~128~~ ~~0~~ 128	64
1	~~1~~ ~~129~~ ~~1~~ ~~129~~ ~~1~~ ~~129~~ ~~1~~ 129	65
2		
.	.	.
.	.	.
.	.	.
63		

0 mod 64 = 0 → First ref Miss
64 mod 64 = 0 → First ref Miss
128 mod 64 = 0 → conflict miss, LRU is used, so 128 replaces 0
64 → Hit
0 mod 64 = 0 → conflict miss, LRU is used, so 0 replaces 128
64 → Hit
128 mod 64 = 0 → conflict miss, LRU is used, so 128 replaces 0.
64 → Hit
1 mod 64 = 1 → First ref miss
65 mod 64 = 1 → First ref miss
129 mod 64 = 1 → conflict miss, LRU is used, so 129 replaces 1.
65 → Hit
1 mod 64 = 1 → conflict miss, LRU is used, so 1 replaces 129.
65 → Hit
129 mod 64 = 1 → conflict miss, LRU is used, so 129 replaces 1.
65 → Hit
0 mod 64 = 0 → conflict miss, LRU is used, so 0 replaces 128.
64 → Hit
128 mod 64 = 0 → conflict miss, LRU is used, so 128 replaces 0.
64 → Hit

0 mod 64 = 0 → conflict miss
64 → Hit
128 mod 64 = 0 → conflict miss
64 → Hit
1 mod 64 = 1 → conflict miss
65 → Hit
129 mod 64 = 1 → conflict miss
65 → Hit
1 mod 64 = 1 → conflict miss
65 → Hit
129 mod 64 = 1 → conflict miss
65 → Hit

∴ Total number of references = 32
Number of misses = 18

∴ Miss ratio = $\frac{18}{32} = \frac{9}{16} = 47.25\%$

Number of conflict misses = 14

Previous Years' Question

In a k–way set associative cache, the cache is divided into v sets, each of which consists of k lines. The lines of a set are placed in the sequence of one after another. The lines in the set s are sequenced before the lines in the set (s+1). The main memory blocks are numbered 0 onwards. The main memory block numbered j must be mapped to any one of the cache lines from

a) (j mod v) * k to (j mod v) * k + (k–1)

b) (j mod v) to (j mod v) + (k–1)

c) (j mod k) to (j mod k) + (v–1)

d) (j mod k) * v to (j mod k) * v + (v–1)

Sol: a) **(GATE-2013 (1 Mark))**

5.5 HARDWARE IMPLEMENTATION OF SET ASSOCIATIVE MAPPING

Basic diagram to check hit/miss in set associative mapping:

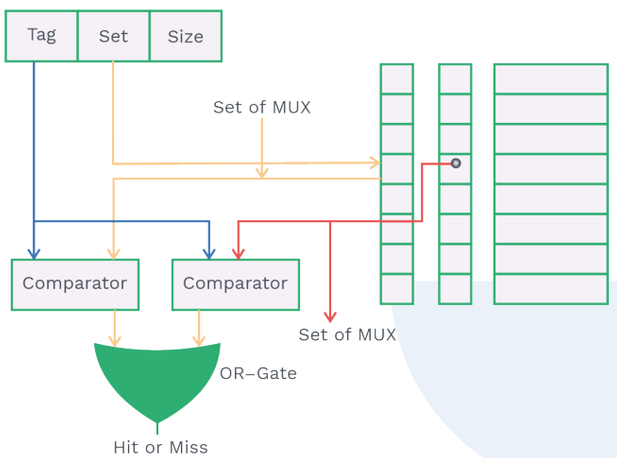


Fig. 5.18 Checking Hit/Miss in Associative Mapping

Previous Years' Question

The width of the physical address on a machine is 40 bits. The width of the tag field in a 512 KB 8–way set associative cache is _____ bits.

Sol: Range 24 to 24

(GATE-2016 (Set–2) (2 Marks))

Hardware implementation in set associative

No. of blocks in CM = 8

and 2–way set associative

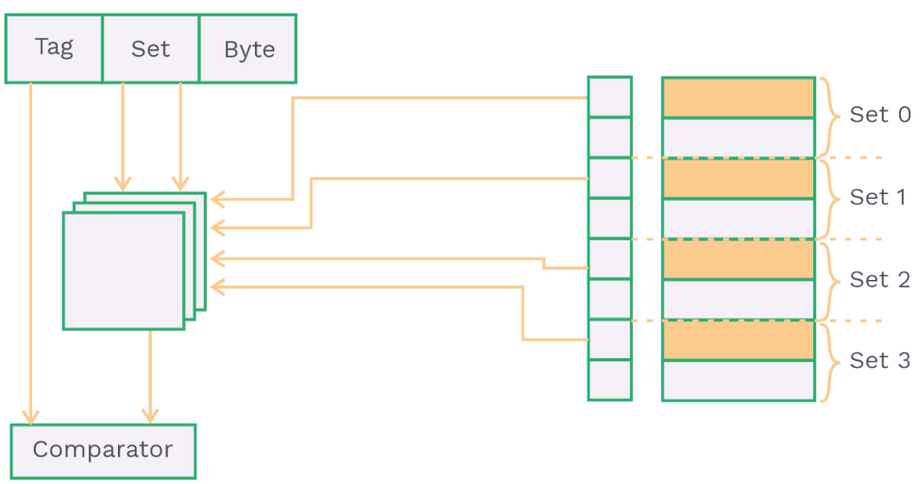


Fig. 5.19 Hardware Design used in Associative Mapping

Hardware implementation in set associative mapping

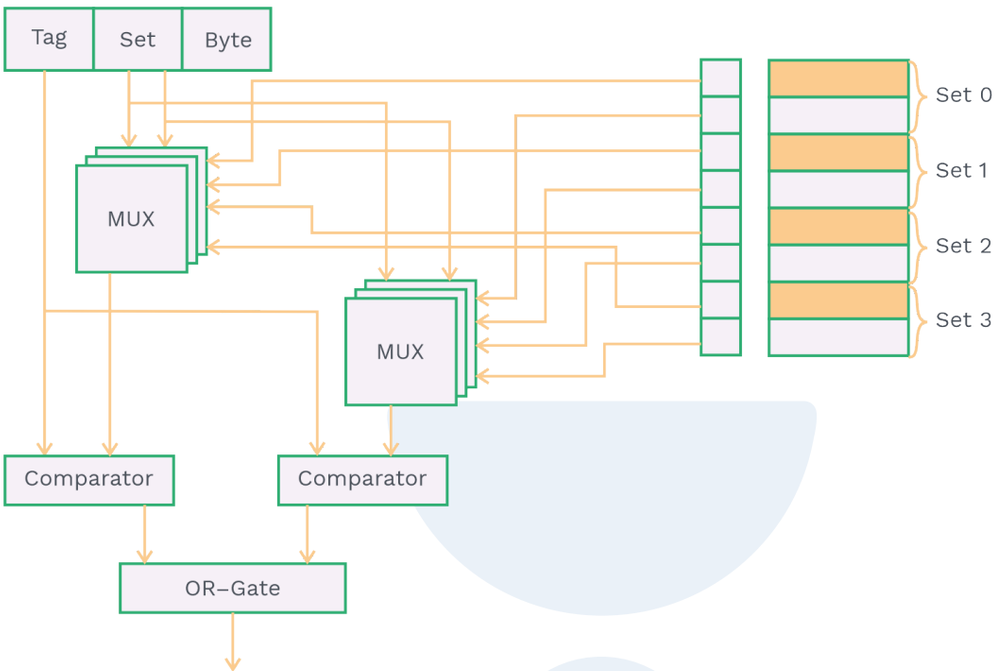


Fig. 5.20 Hardware Design using Set-Associative Mapping

Hardware Size for R–way Set Associative

a) Multiplexer (MUX):
No. of MUX = R * Number of Tag bits
Size of MUX = No. of sets in cache: 1

b) Comparator:
No. of comparator = R
Size of comparator = Number of Tag bits

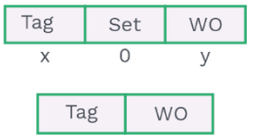
c) OR gate:
No. of gates = 1
No. of inputs = R

Hit latency in set associative mapping:

$$\text{Total time to check hit / miss (Hit latency)} = \text{MUX delay} + \text{Comparator delay} + \text{OR gate delay}$$

Fully associative mapping:
In fully associative, all the blocks of the main memory mapped with a single set. So there is no need of set index. i.e. set index bit is 0 (2^0 = 1 set)

Eg:



* Index bit comparison in all mapping techniques.

Technique	Index bit
Direct mapping	Cache line number bit
Set associative mapping	set offset bit
Fully associative mapping	0–bit

Table 5.1 Index Bits Reserved in Different Mapping Techniques

Formula:

Tag directory size = No. of cache blocks * Tag bits

PRACTICE QUESTIONS

Q14 **Consider the following parameters**
Cache memory size = 64 KB
Block size = 32 bytes
Physical address = 34 bits
Calculate the tag directory size in
a) Direct mapping
b) Fully associative mapping
c) 8–way set associative mapping

Sol: No. of blocks in CM (L) = $\frac{\text{CM size}}{\text{Block size}} = \frac{64\,\text{KB}}{32\,\text{B}}$

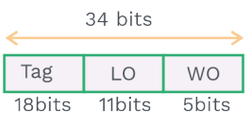
= 2K

L = 2^{11}

Line offset(LO) = 11 bits

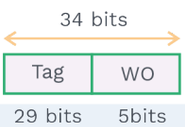
Word offset (WO) = 5 bits

a) Direct mapping:



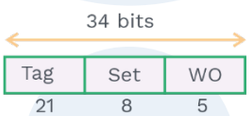
Tag directory size = No. of lines in CM * Tag space
= 2^{11} * 18
= 36 K bits

b) Fully associative cache:



Tag directory size = No. of lines in CM * Tag bits
= 2^{11} * 29
= 58 K bits

c) 8–way set associative cache:



No. of sets = $\frac{\text{No. of lines (L)}}{\text{R – way set}} = \frac{2^{11}}{8}$

= 2^8

Set offset = 8 bit

Tag directory size = No. of lines in CM * Tag space
= 2^{11} * 21
= 42K bits

Q15 **Consider fully associative cache memory size of 16KB having 4 KB data blocks. Consider the following references blocks 8, 13, 16, 6, 8, 15, 20, 23, 29, 27, 23, 16, 20 What is the hit ratio if the cache is initially empty and designed with LRU (Least recently used)?**

Sol: **15.38**

Given: Fully associative cache

CM size = 16 KB

Block size = 4 KB

No. of lines in CM (L) = $\frac{\text{CM size}}{\text{Block size}} = \frac{16\,\text{KB}}{4\,\text{KB}} = 4$

CM

~~8~~	23		
~~16~~	~~20~~	16	
~~13~~	~~15~~	27	
~~8~~	~~29~~	20	

8 – Miss
13 – Miss
16 – Miss
6 – Miss
8 – Hit
15 – Miss
20 – Miss
23 – Miss
29 – Miss
27 – Miss
23 – Hit
16 – Miss
20 – Miss

Hit ratio = $\frac{\text{No. Hits}}{\text{Total block reference}} = \frac{2}{13} * 100 = 15.38$

Previous Years' Question

A certain processor uses a fully associative cache of size 16 KB, the cache block size is 16 bytes. Assume that the main memory is byte addressable and uses a 32–bit address. How many bits are required for the tag and the index fields respectively in the addresses generated by the processor?

a) 24 bits and 0 bits **b)** 28 bits and 4 bits
c) 24 bits and 4 bits **d)** 28 bits and 0 bits

Sol: d) **(GATE-2016 (Set–2) (2 Marks))**

5.6 HARDWARE IMPLEMENTATION OF FULLY ASSOCIATIVE

Fully associative mapping:

Checking hit/miss in fully associative mapping:

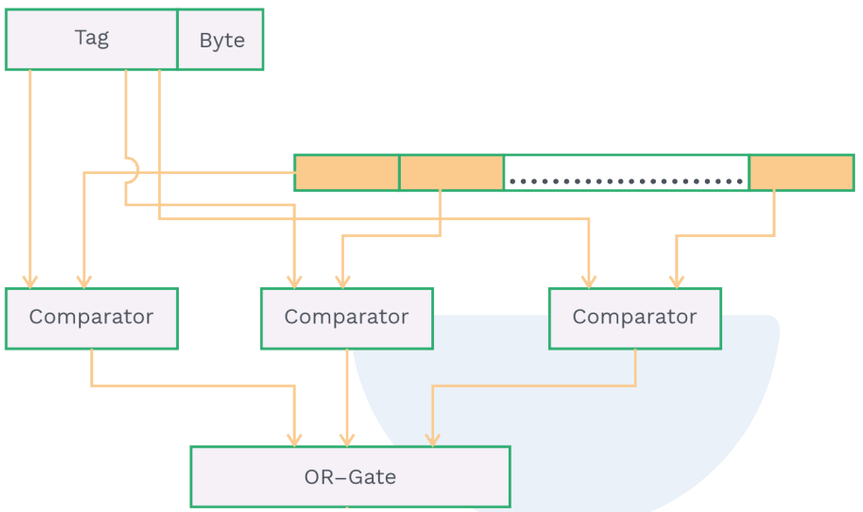


Fig. 5.21 Hardware Design used in Fully Associative Mapping

Note:

We don't need MUX here because we don't have any concept of choosing sets/lines.

Hardware size for fully associative mapping:

a) Comparator:

No. of comparator = No. of blocks in CM

Size of comparator = Tag bits

b) OR gate:

No. of OR gates = 1

Size of OR gates = No. of blocks in cache

Hit latency in fully associative mapping:

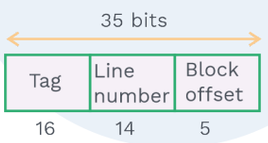
$$\text{Total time to check Hit / miss (Hit latency)} = \text{Comparator delay} + \text{OR gate delay}$$

PRACTICE QUESTIONS

Q16 **Consider 512 KB cache in direct mapping. Block size of the cache is 32 bytes and CPU generates 35 bit physical address then calculate**
No. of MUX and size of MUX in direct and also calculate same parameters in 4–way set associative cache.
No. of comparator and size of comparator in all three mapping technique.
No. of OR gate and input size of OR gate in 4–way set associative cache and fully associative cache.

Sol: Number of lines = $\frac{512\,KB}{32\,B} = \frac{2^9 \times 2^{10}}{2^5} = 2^{14}$

1) Direct mapping:



a) No. of multiplexer (MUX) = No. of bits in tag
No. of MUX = 16
b) Size of MUX = No. of blocks in CM: 1
$= 2^{14}: 1$
c) No. of comparator = 1
d) Size of comparator = 16 bits

2) 4–way set associative:

Number of sets = $\frac{2^{14}}{2^2} = 2^{12}$

a) Number of MUX = R * Number of bits in tag space
$= 4 * 18 = 72$
b) Size of MUX = No. of sets: 1 = $2^{12}: 1$
c) Number of comparator = 4
d) Size of comparator = 18 bits
e) Number of OR gate = 1
f) Number of input for OR gate = 4

3) Fully associative:

a) No. of comparator = No. of blocks = 2^{14}
b) Size of comparator = 30 bits
c) No. of OR gate = 1
d) Size of OR gate = 2^{14} inputs

Previous Years' Question

Consider two cache organizations: The first one is 32 KB 2–way set associative with 32–byte block size. The second one is of the same size but direct mapped. The size of an address is 32 bits in both cases. A 2–to–1 multiplexer has a latency of 0.6 ns while a k bit comparator has a latency of k/10 ns. The hit latency of the set associative organization is h1 while that of the direct mapped one is h2.
The value of h1 is:

a) 2.4 ns **b)** 2.3 ns
c) 1.8 ns **d)** 1.7 ns

Sol: a) **(GATE-2006 (2 Marks))**

5.7 TYPES OF CACHE MISS

1) Compulsory miss (cold start miss):

First time access of a block will always cause a miss means every first time reference block is not present in Cache Memory during program execution.

- Compulsory misses can be reduced by increasing the cache block size.

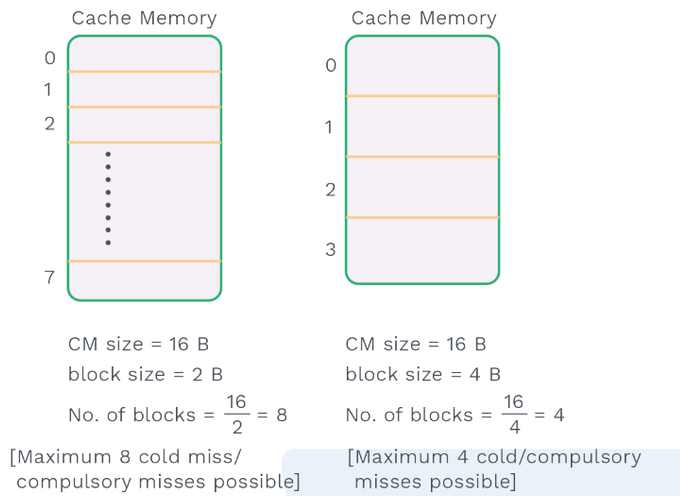


Fig. 5.22

Note:

If block size increases, miss penalty time will increase because transferring time of block will increase; so time will increase.

2) Capacity miss:

Capacity miss occurs when the cache is full, and its' not a compulsory miss. Capacity miss occurs high in a fully associative cache.

- Capacity misses can be reduced by increasing the cache memory size because the cache block will increase.

3) Conflict miss:

This miss occurs when the cache set is full, and many blocks are mapped in the same cache line/block.

- Conflict misses can be reduced by increasing (doubling) the set associativity in set associative cache.

Suppose reference blocks are: 4, 8, 12, 16, 4, 8
In direct mapping, every block in the main memory is mapped to a particular cache line. So, conflict misses are more.
It is medium in set associative mapping.

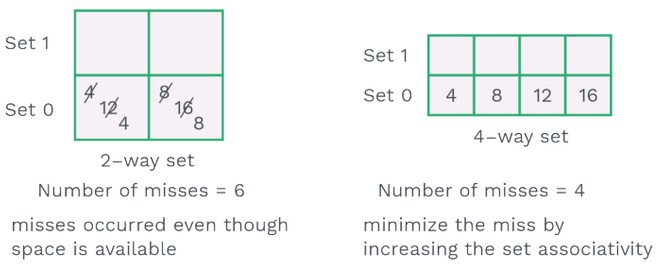


Fig. 5.23

Note:

Conflict miss does not occur in fully set–associative cache.

Previous Years' Question

Consider a 2–way set associative cache with 256 blocks and uses LRU replacement, Initially, the cache is empty. Conflict misses are those misses which occur due to the contention of multiple blocks for the same cache set. Compulsory misses occur due to first time access to the block. The following sequence of accesses to memory blocks
(0, 128, 256, 128, 0, 128, 256, 128, 1, 129, 257, 129, 1, 129, 257, 129)
is repeated 10 times. The number of conflict misses experienced by the cache is ___

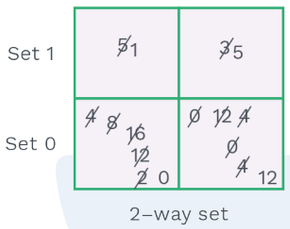
Sol: 76 **(GATE-2017 (Set – 1) 2 Marks)**

PRACTICE QUESTIONS

Q17 **Consider a 2–way set associative cache with block size 4B and cache size 16B CPU requests for main memory blocks in following orders**
4, 0, 8, 12, 16, 4, 12, 0, 5, 3, 1, 2, 4, 5, 0, 12 and 5.
Cache is initially empty then calculate these miss using LRU policy
a) Conflict miss
b) Capacity miss
c) Compulsory/cold miss

Sol: 2–way set associative

- Cache size = 16B
- Block size = 4B
- $\text{No. of blocks} = \frac{\text{Cache size}}{\text{Block size}} = \frac{16\,B}{4\,B} = 4$
- $\text{No. of sets(S)} = \frac{N}{R-\text{way}} = \frac{\text{No. of blocks}}{2} = \frac{4}{2} = 2$



2–way set

4 → Compulsory miss
0 → Compulsory miss
8 → Compulsory miss
12 → Compulsory miss
16 → Compulsory miss
} 5 compulsory miss

4 → Conflict miss
12 → Conflict miss
0 → Conflict miss
} 3 conflict miss

5 → Compulsory miss
3 → Compulsory miss
1 → Compulsory miss
2 → Compulsory miss
} 4 compulsory miss

4 → Capacity miss
5 → Capacity miss
0 → Capacity miss
12 → Capacity miss
} 4 capacity miss

5 → Hit

Total no. of conflict miss = 3
Total no. of capacity miss = 4
Total no. of compulsory miss = 9

Note:

Cache line size = Main memory block size
No. of cache lines = Number of blocks in cache memory

Cache coherence problem:
Distinct images of the same data may be present in the cache and main memory at some instant of time and if processors are allowed to update their own copies freely, then an inconsistent view of memory can result.

5.8 UPDATING TECHNIQUES

There are two updating techniques (Protocols)

1) Write through Protocol

2) Write back the protocol

These protocols are used to avoid data inconsistent problems.

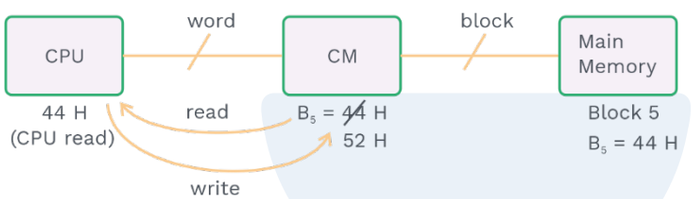


Fig. 5.24

In MM block B_5 address is 44 H but in cache B_5 = 52 H this is known as cache coherence (data inconsistent)

1) Write through protocol:

To avoid data inconsistent problem, cache and main memory location updated simultaneously by CPU.

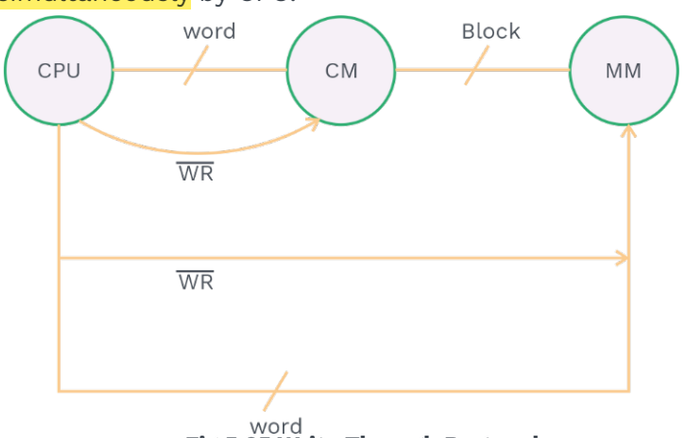


Fig 5.25 Write Through Protocol

- Here, cache and main memory both is updated simultaneously by processor After updating byte/word is accessed from cache memory.

Read cycle time:

- If time taken by cache memory to read the data is C and time taken by main memory is M. Hit ratio of read operation is H_r. Then average read cycle time is

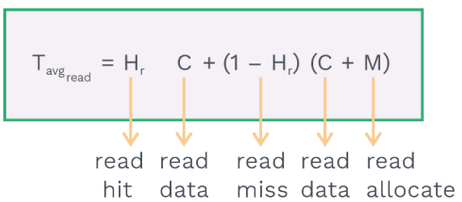


Fig. 5.26 Calculation of Read Cycle Time using Write Through Protocol

Write cycle time:

- If simultaneous write operation time is W and hit ratio of write operation is H_w the average write cycle time

Case i:

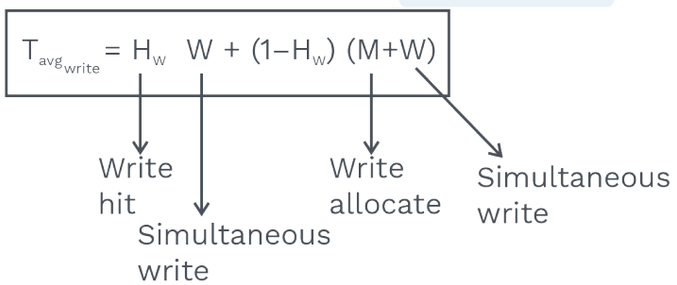
If no write allocation policy

$$T_{avg_{write}} = \text{Max}\,(\text{cache access time}, \text{Main memory access time})$$

$$T_{avg_{write}} = \text{Main memory access time}$$

Case ii:

If write allocation policy



- If frequency of read operation is F_r and frequency of write operation is F_w then the average write through protocol and efficiency.

Average time:

$$T_{avg_{write\ through}} = \left(T_{avg_{read}} * F_r\right) + \left(T_{avg_{write}} * F_w\right)$$

Efficiency (η):

$$\eta = \frac{1}{T_{avg_{write\ through}}} \text{WPS}$$ Where WPS = word/sec

Note:

$$\text{Simultaneous word updation Time}(w) = \max\left(\begin{matrix}\text{word updation,}\\ \text{time in CM}\end{matrix}\ \begin{matrix}\text{word updation}\\ \text{time in main memory}\end{matrix}\right)$$

- If hit ratio of write operation is 100% then use simultaneous access.

C → Cache Memory Access Time

M → Main Memory Access Time

$$T_{avg_{read}} = HC + (1-H)M$$

$$T_{avg_{write}} = W$$

2) Write back protocol:

In the write back protocol, the cache location is only updated and mark it as updated with an associated flag bit called dirty or modified bit. Main memory allocation is updated when the block containing this marked word is to be removed from the cache to make space for a new block. This is also known as copy back protocol.

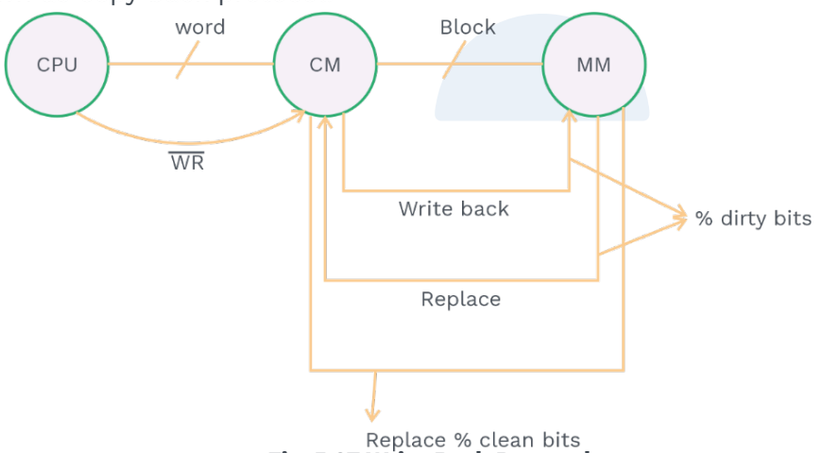


Fig. 5.27 Write Back Protocol

Read cycle time:

If the hit ratio of read operation is "H". Cache memory access time is "C" and main memory access time is "M".

$$T_{avg_{read}} = HC + (1-H)\left[\begin{matrix}\text{\% dirty}\\ \text{bits}\end{matrix}\ \begin{matrix}(C+M+M)\\ \text{read data}\ \ \text{read allocate}\ \ \text{write back}\end{matrix}\ \text{+ \% clean bits } (C+M)\right]$$

Write cycle time:
If hit ratio of write the data is H_w the calculate write cycle time

$$T_{avg_{write}} = H_w\, C + (1 - H_w)\left[\%\ \text{dirty bits}\ (C + M + M) + \%\ \text{clean bits}\ (C + M)\right]$$

(C + M + M): write data, write allocate, write back

Average access time of write back:
If frequency of read operation is "F_r" and frequency of write operation is F_w then

$$T_{avg_{writeback}} = \left(T_{avg_{read}} * F_r\right) + \left(T_{avg_{write}} * F_w\right)$$

PRACTICE QUESTIONS

Q18 **Consider 64KB 4–way set associative write through cache. CPU generates 34 bit physical address, 20% write requests and remaining are used for read requests. Hit ratio of read and write operation 50% and 60% Cache Memory access time 100 ns and main memory access time is 950 ns. What is the average memory access time when considering both read and write operation?**

Sol: **726**

- 4–way set associative cache
- CM size = 64 KB
- Physical Address = 34 bits
- F_w = 20%
- F_r = 80%
- H_r = 50%
- H_w = 60%
- C = 100 ns
- M = 950
- Block size not given assumes 1 w.
- Write through cache.

Read cycle time:

$$T_{avg_{read}} = H_r * C + (1 - H_r)(C + M)$$

= 0.5 * 100 + 0.5 * (100+950)
= 50 + 1050 * 0.5
= 50 + 525
= 575 ns

Write cycle time:

Simultaneous word updation time (w) = $\max(100, 950)$

w = 950 ns

$$T_{avg_{write}} = H_w * w + (1 - H_w)[M + w]$$

= 0.6 * 950 + 0.4 * 1900 = 570 + 760
= 1330 ns

Average access time:

$$T_{avg_{write\ through}} = (T_{avg_{read}} * F_r) + (T_{avg_{write}} * F_w)$$

= 575 * 0.8 + 1330 * 0.2 = 460 + 266
= 726 ns

Q19 **Consider a system write back cache which is used in the 36 bit CPU. Size of cache memory is 32 KB. CPU generates 60% read request and 40% write requests. Access time of cache and main memory is 60 and 800 respectively. When miss occurred in CM then 8w data block is transferred from physical memory to cache memory. Hit ratio of read and write operation is 60% and 80%. What is the average memory access time?**

Sol: **418.4**

- Write back
- Cache memory size = 32 KB
- Word size = 36 bits
- Block size = 8w
- Fr = 60% (% clean bits)
- F_w = 40% (% dirty bits)
- C = 60 ns
- M = 800 ns
- H_w = 80%
- H = 60%

Read cycle time:

$$T_{avg_{read}} = H * C + (1 - H)\left[\%\,\text{dirty bits}\,(C + M + M) + \%\,\text{clean bits}\,(C + M)\right]$$

$= .6 * 60 + .4\ [.4 * 1660 + .6 * 860]$

$= 508$ n

Write cycle time:

$$T_{avg_{write}} = H_w C + (1 - H_w)\left[\%\,\text{dirty bits}\,(C + M + M) + \%\,\text{clean bits}\,(C + M)\right]$$

$= .8 * 60 + .2\ [.4 * 1660 + .6 * 860]$

$= 284$ ns

Average access time:

$$T_{avg_{writeback}} = \left(T_{avg_{read}} * F_r\right) + \left(T_{avg_{write}} * F_w\right)$$

$= 508 * .6 + 284 * .4$

$= 418.4$ ns

Q20 **Consider a cache memory of size 16 KB and block size is 2 words. Cache is designed using write through protocol with the access time of cache memory and main memory as 20 and 250 ns, respectively. Hit ratio of read and write operations are 80% and 60%, respectively. If read cycle time is X ns and write cycle time is Y ns, then calculate the X + 6Y in nanosecond.**

a) 1420 ns **b) 295 ns** **c) 258 ns** **d) 790 ns**

Sol: **a)**

Given:

CM size = 16 KB

Block size = 2 words

$T_{CM} = 20$ ns

$T_{MM} = 250$ ns

$H_r = 80\%$

$H_w = 60\%$

Write through protocol:

Simultaneous write operation time (T_w) = Max (Word updation time in CM, word updation time in MM)

T_w = Max (20 ns, 2502)

$T_w = 125$ ns

Read cycle time (X)

read avg $T = H_r\, T_{CM} + (1 - H_r)\,(T_{CM} + T_{MM})$

$= 0.8 \times 20 + 0.2 \times 270 = 16 + 54$

$= 70$

X = 70 ns

Write cycle time (Y)

write avg T = $H_w T_w + (1 - H_w) (T_{MM} + T_w)$

= 0.6 × 125 + 0.4 × 375

= 225 ns

Y = 225 ns

So, X + 6Y = 70 + 6 × 225

= 1420 ns

Cache inclusion policies:

There is two kind of policies

1) Inclusion

2) Exclusion

1) Inclusion:

In multilevel cache, higher level (L_1) content is present in lower level (L_2). Thus, we can say the higher level cache content is subset of the lower level cache.

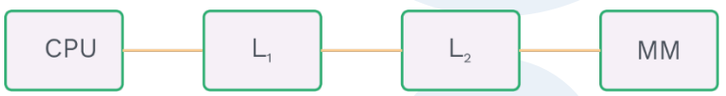


Fig. 5.28

What happened if:

a) Hit in L_1:

CPU reads the block from Level–1 only.

b) Miss in L_1 and Hit in L_2:

Copy the missed block from L_2 to L_1. There may be a block evicted from L_1 and there is no any kind of involvement of L_2 in this case.

c) Miss in L_1 and miss in L_2:

Copy the block in L_2 and L_1 from main memory.

- L_2's evicted block will be made invalid in L_1 (to make inclusion)
- L_1's evicted block has no involvement of L_2.

2) Exclusion:

In multilevel cache, higher level (L_1) content should not necessarily be present in L_2.

Fig. 5.29

What happened if:

a) Hit in Level–1 (L_1):

CPU reads the word from L_1 only.

b) Miss in L_1 and Hit in L_2:

Transfer the block from L_2 to L_1. The evicted block is moved to L_2.

c) Miss in L_1 and miss in L_2:

Copy the block from main memory to level–1 (L_1) only no need to copy in L_2 (invalid block in L_2) because of exclusion.

Chapter Summary

- Memory hierarchy design is based on size, cost, speed etc.
- In simultaneous access CPU can access the data from any level directly.
- In hierarchy design, block is transferred from lower level to higher level cache.
- Cache is small, fast and costly than main memory.
- Performance of the cache depends on:
  - **a)** Size of the cache
  - **b)** Size of the block
  - **c)** Levels of the cache
  - **d)** Mapping techniques
  - **e)** Replacement policy
  - **f)** Updation policy

- **Mapping technique:**
  - **a)** Direct mapping
  - **b)** Set associative mapping

- **Fully associative mapping:**
  - **a)** '0' bit is needed to represent the line in a fully set associative.
  - **b)** Compulsory misses can be reduced by increasing the block size.
  - **c)** Capacity miss occurs when the cache is full, and it should not be a compulsory miss. These misses can be reduced by increasing the cache size.
  - **d)** Conflict misses occurred in R–way set associative mapping; these misses can be reduced by increasing the set associativity.
  - **d)** Hit latency in direct mapping depends on two factor comparator delay and MUX delay.
  - **e)** Hit latency in R–way set associative mapping depends on three factors MUX delay, comparator delay and OR gate delay.
  - **f)** Hit latency in fully associative mapping depends on comparator and OR gate delay.

- **Cache coherence:**
  Data inconsistent problem is known as a cache coherence problem. To avoid the cache, coherence updating techniques are used (write through and write back).