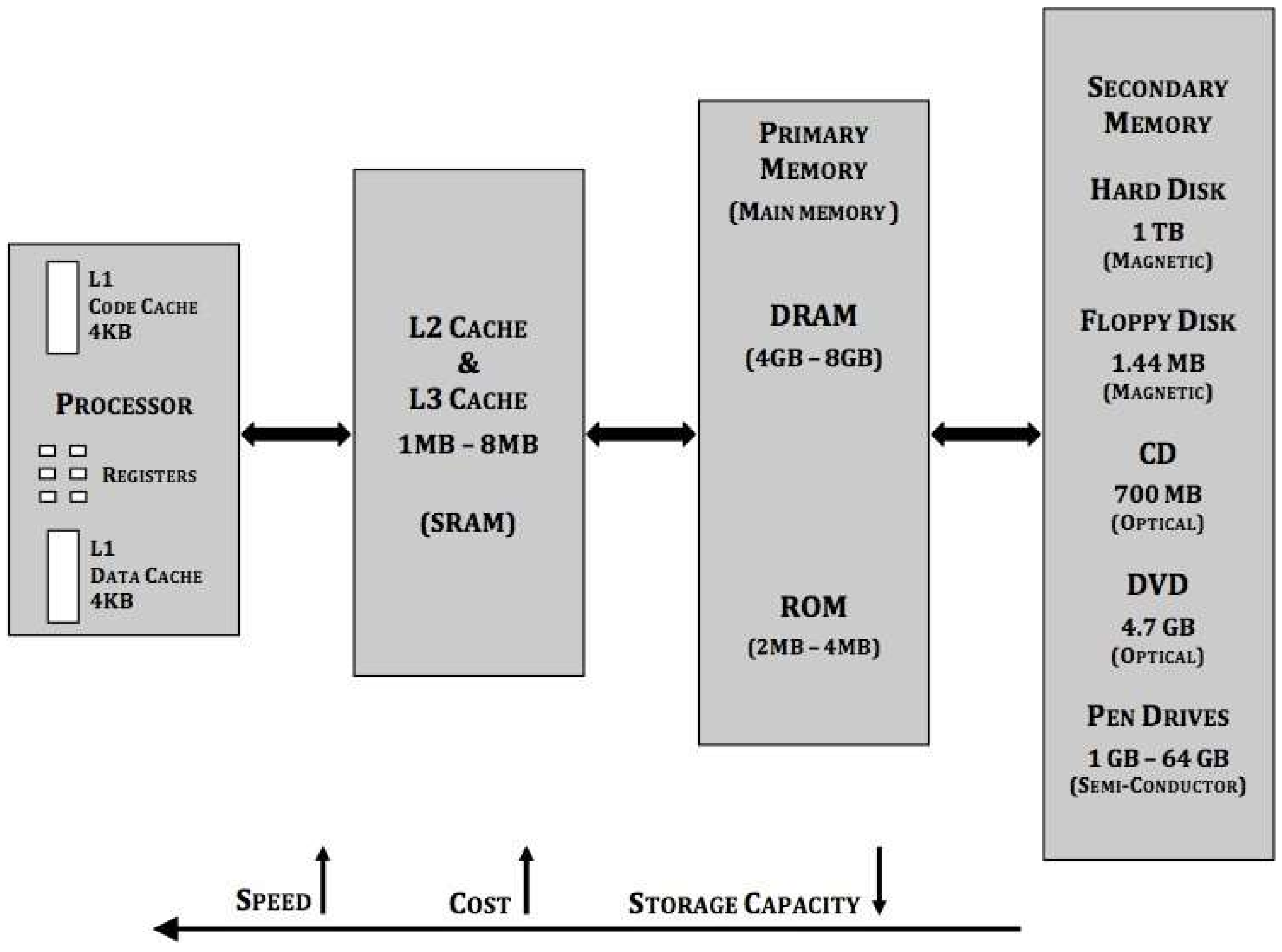
Computer Organization & Architecture
Unit- 4

Memory System Design

MEMORY HIERARCHY

1) The purpose of any memory device is to **store programs and data**.
2) **Several types** of memory devices are used in the computer forming a **Memory Hierarchy**.
3) Each plays a specific role contributing to the **speed, cost effectiveness, portability** etc.



REGISTERS

1) Registers are **present inside the processor**.
2) They are basically a **set of flip-flops**.
3) They store **data and addresses** and can **directly take part** in **arithmetic and logic operations**.
4) They are very small in size typically **just a few bytes**.

PRIMARY MEMORY

1) It is the original form of memory also called as **Main memory**.
2) It comprises of **RAM and ROM**, both are **Semi-Conductor** memories. (chip memories)
3) **ROM is non-volatile**.
   It is used is storing permanent information like the **BIOS program**.
   It is typically of **2 MB - 4 MB** in size.
4) **RAM is writable** and hence is used for **day-to-day operations**.
   Every file that we access from secondary memory, is **first loaded into RAM**.
   To provide large amount of working space RAM is **typically 4 GB - 8 GB**.

SECONDARY MEMORY

1) The main purpose of Secondary Memory is to **increase the storage capacity, at low cost**.
2) Its biggest component is the **Hard Disk**.
   This is where all the files inside a computer **are stored**.
3) It is **writeable as well as non-volatile**.
4) Typical size of a **HD is 1 TB**.
5) Disk memories are much **slower than chip memories** but are also **much cheaper**.

PORTABLE SECONDARY MEMORY

1) These are required to **physically transfer files** between computers.
2) **Floppy Disk**: It is a **magnetic form** of storage. Typical **Size is 1.44 MB**.
3) **CD**: It is an **optical form** of storage. Typical **Size is 700 MB**.
4) **DVD**: It is an **optical form** of storage. Typical **Size is 4.7 MB**.
5) **Pen Drives & Memory Cards**: It is a **semi-conductor form** of storage.
   It is composed of FLASH ROM.
   It's a special type of ROM that's writable as well as non-volatile.
   Typical **Size ranges form 1 GB - 64 GB** depending upon the cost.

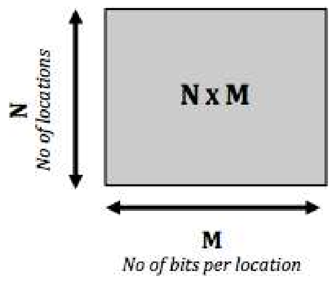
CACHE MEMORIES

1) It is the **fastest form of memory** as it uses SRAM (Static RAM).
2) The Main Memory uses DRAM (Dynamic RAM).
3) **SRAM uses flip-flops and hence is much faster than DRAM which uses capacitors.**
4) But SRAM is also **very expensive** as compared to DRAM.
5) Hence **only the current portion of the file** we need to access is copied from Main Memory (DRAM) to Cache memory (SRAM), to be directly accessed by the processor.
6) This gives **maximum performance and yet keeps the cost low**.
7) Typical size of Cache is around **2 MB – 8MB**.
8) If **code and data** are in the **same cache** then it is **unified cache** else its **called split cache**.
9) Depending upon the location of cache, it is of three types: L1, L2 and L3.
10) **L1** cache is **present inside the processor** and is a **split cache** typically **4-8 KB**.
11) L2 is present on the **same die as the processor** and is a **unified cache** typically **1 MB**.
12) L3 is present **outside the processor**. It is also **unified** and is typically of **2-8 MB**.

MEMORY CHARACTERISTICS

1) **Location**
Based on its physical location, memory is classified into three types.
- **On-Chip:** This memory is present **inside the CPU**. E.g.:: Internal Registers and **L1 Cache**.
- **Internal:** This memory is present **on the motherboard**. E.g.:: **RAM**.
- **External:** This memory is **connected to the motherboard**. E.g.:: **Hard disk**.

2) **Storage Capacity**
This indicates the **amount of data stored** in the memory.
Obviously it should be **as large as possible**.
It is represented as N x M.
Here,
N = **Number of memory locations** (no of words)
M = **Number of pits per memory location** (word size)
E.g.:: (4K x 8) means there are 4K locations of 8-bits each.



3) **Transfer Modes**
Data can be accessed from memory in two different ways.
- **Word Transfer:** Here, if CPU needs some data, it will transfer only that amount of data.
E.g.:: Data accessed from **L1 Cache.**
- **Block Transfer:** Here, if CPU needs some data, it will transfer an entire block containing that data. This makes further access to remaining data of this block much faster. This is based on Principle of Spatial Locality. A processor is most likely to access data near the current location being accessed.
E.g.:: On a **cache miss**, processor goes to **main memory** and copies a **block** containing that data.

4) **Access Modes**
Memories can allow data to be accessed in two different ways.
- **Serial Access:** Here locations are accessed one by one in a **sequential manner**.
The access time depends on how far the target location is, from the current location.
Farther the location, **more** will be its **access time**.
E.g.:: Magnetic tapes.
- **Random Access:** Here **all locations** can be directly accessed in any **random order**.
This means **all locations** have the **same access time** irrespective of their address.
E.g.:: Most modern memories like RAM.

5) **Physical Properties**
There are various Physical attributes to memory.
- **Writeable: Contents** of the memory **can be altered**. E.g.:: **RAM**
- **Non-Writeable: Contents** of the memory **cannot be altered**. E.g.:: **ROM**
- **Volatile: Contents** of the memory are **lost** when power is **switched off**. E.g.:: **RAM**
- **Non-Volatile: Contents** of the memory are **retained** when power is **switched off**. E.g.:: **ROM**
Most secondary memories like Hard disk are Writable as well as non-volatile.

6) **Access Time (t_A)**
It is the time taken between **placing the request** and **completing the data transfer**.
It should be as **less as possible**.
It is also known as **latency**.

7) **Reliability**
It is the **time** for which the memory is expected to **hold the data without any errors**.
It is measured as **MTTF: Mean Time To Failure**.
It should be as **high as possible**.

8) **Cost**
This indicates the **cost of storing data** in the memory. It is expressed as **Cost/bit**.
It must be **as low as possible**.

9) **Average Cost**
It is the total cost per bit, for the entire memory storage.
Consider a system having **two memories M_1 (RAM) & M_2 (ROM)**
If **C_1** is the cost of memory M_1 of size **S_1**
& **C_2** is the cost of memory M_2 of size **S_2**

Then the average cost of the memory is be calculated as:
$C_{AVG} = (C_1 S_1 + C_2 S_2)/ (S_1 + S_2)$

Small sizes of **expensive** memory and **large** size of **cheaper** memory **lowers** the **average cost**.

10) **Hit Ratio (H)**
Consider two memories M_1 and M_2.
M_1 is **closer** to the processor E.g.:: **RAM**, than **M_2** E.g.:: **Hard disk**.
If the **desired data is found in M_1**, then it is called a **Hit**, else it is a **Miss**.
Let **N_1** be the number of **Hits** and **N_2** the number of **Misses**.
The **Hit Ratio** H is defined as **number of hits divided by total attempts**.
$H = (N_1) / (N_1 + N_2)$
It is expressed s a percentage.
H can never be 100%. In most computers it is maintained around 98%.

From the above discussion it is clear that no single memory can satisfy all the characteristics, **hence we need a hierarchy of memories**.
Cache memories are the **fastest** but also the **most costly**.
Hard disk is **writeable** as well as **non-volatile** and is also very **inexpensive**, but is much **slower**.
CD/DVD etc. are needed for **portability**.
ROM is **nonvolatile**, and is used for **storing BIOS**.
DRAM is **writable**, f**aster than hard disk** and **cheaper than SRAM** hence forms **most part of Main Memory**.

7) Cache Mapping Techniques

Blocks are loaded from Main Memory to Cache Memory.

Cache Mapping decides which block of Main Memory comes into which block of Cache Memory.
There are several mapping techniques trying to balance between Hit Ratio, Search-time and Tag size.
Every cache block has a **Tag** indicating which block of Main Memory is mapped into that block.
A collection of such tags is called the **cache director**y, very similar to a page table.
Cache directory is a part of the cache.
Since Cache Memory is **very expensive**, we need the cache directory to be as small as possible.
That means the **Tag must be of minimum size**.
The Main Memory address, issued by the processor contains the desired block number.
This is compared to the Tag of a Cache Block, which gives the Block number that is present.
If they are equal, **it's a Hit**. If Not, the search may have to be repeated for several other blocks.
It is obvious to understand, **the number of searches must be as low as possible.**
Finally, the Mapping technique must yield maximum utilization of the Cache memory space, so that the **Hit ratio remains as High as possible**.
There are three popular Cache Mapping Techniques:

1) **Associative Mapping** also called Fully Associative Mapping
2) **Direct Mapping** also called One-Way Set Associative Mapping
3) **Set Associative Mapping** also called Two-Way Set Associative Mapping

<u>ASSOCIATIVE MAPPING (FULLY ASSOCIATIVE MAPPING)</u>

During memory operations, Blocks are loaded from Main Memory to Cache Memory.
Cache Mapping decides which block of Main Memory comes into which block of Cache Memory.
Fully Associative Mapping technique states:
Any block of Main Memory can be mapped at Any available block of Cache Memory.
There are no rules restricting the mapping at all.
This means the Full Cache is available for mapping hence the name Fully Associative.

Consider Pentium Processor Cache

Size of Main Mem:	**4GB** = **2^{32}**
Size of Cache Mem:	**8KB** = **2^{13}**
Size of Cache Block (Line):	**32** bytes (words) = **2^5**
No. of Blocks in Main Mem:	Size of Main Memory (**2^{32}**) ÷ Size of Block (**2^5**) = **2^{27}**
No. of Blocks in Cache Mem:	Size of Cache Memory (**2^{13}**) ÷ Size of Block (**2^5**) = **2^8** = **256**
Main Mem address:	**32 bits** (because main Mem is of **4GB** = **2^{32}**)

<u>Tag Size:</u>
A block of Cache Memory can **contain any block** of main memory out of a possible **2^{27} blocks**.
Hence, the **Tag** next to every block in Cache Memory must be of **27 bits**.

<u>Searches:</u>
A block of main Memory can be **mapped into any block** of Cache Memory out of **256 Blocks**.
Hence, we need to do **256 searches** in Cache Memory.

<u>Method of Searching:</u>
The Processor issues a 32 bit Main Memory address. It can be divided as:

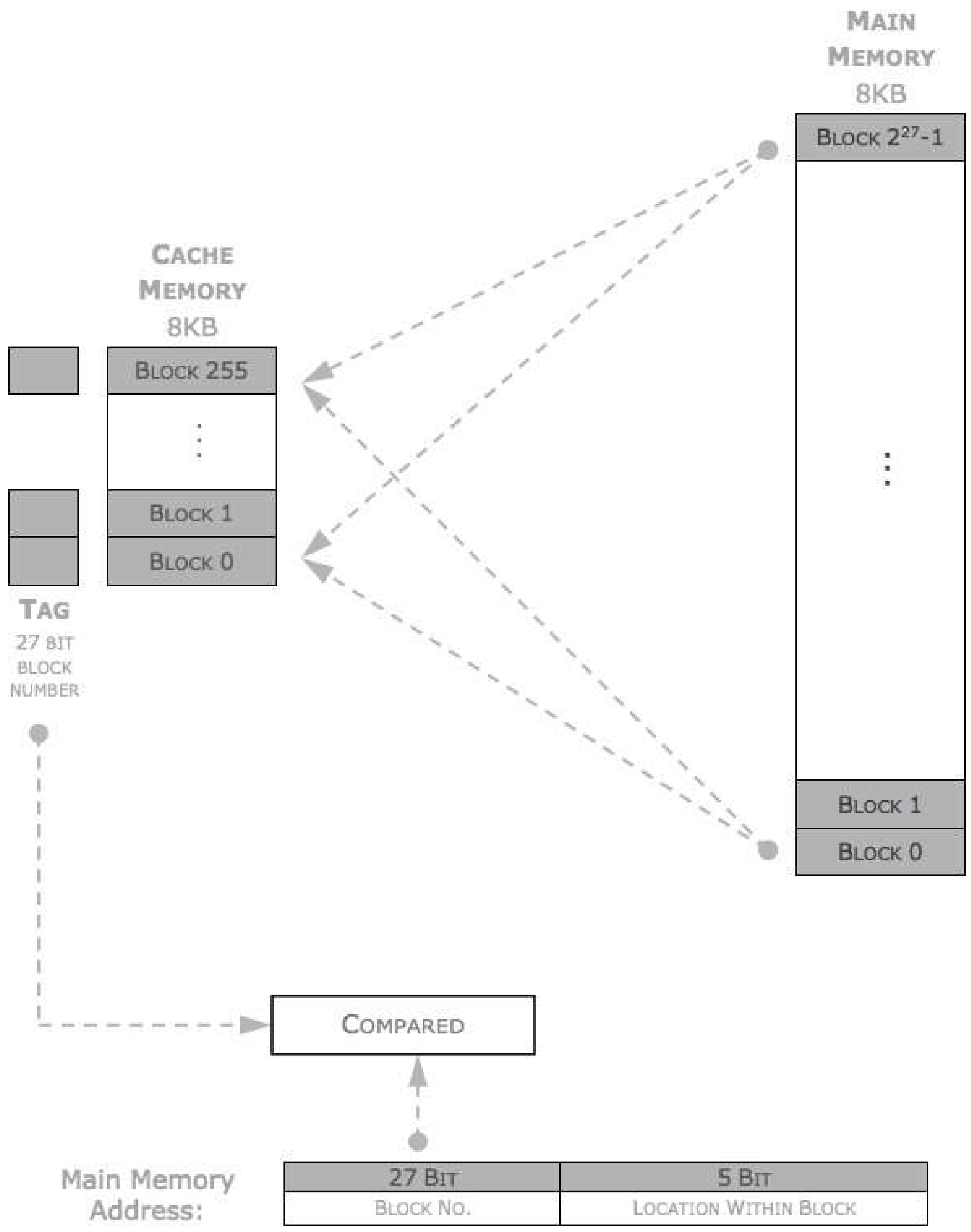
Main Memory Address:

27 BIT	**5 BIT**
Block No.	Location Within Block

This 27 bit Block number is the block number we need to search.
The Tag of each cache block also contains a 27-bit block number.
This is the block number that's present in that respective cache block.
These two block numbers are compared. **If they are equal, it's a HIT.**
If not equal, the search is repeated with the Tag of the next cache block.
This is done a total of 256 times as there are 256 blocks in Cache Memory.
If none of them match with the block number we are searching, then it's a Miss.

<u>Advantage</u>
Since the full cache is available for mapping, it causes maximum utilization of Cache Memory hence gives the **Best Hit Ratio.**

<u>Drawback</u>
Tag Size too big: **27bits.**
Searches are too many: **256.**


Main Memory 8KB
Block 2^27-1
Block 1
Block 0
Cache Memory 8KB
Block 255
Block 1
Block 0
Tag
27 bit block number
Compared
Main Memory Address:
27 Bit
Block No.
5 Bit
Location Within Block

DIRECT MAPPING (ONE WAY SET ASSOCIATIVE MAPPING)

Direct Mapping technique states:
Any block of Main Memory can only be mapped at ONE block of Cache Memory.
Since there is only one way of Mapping, its also called One Way Set Associative Mapping.
We treat the entire Cache as One Set.
The Main Memory is divided into Sets which are then subdivided into Blocks.
A Block of Main Mem. (of any set), can only be mapped into the same Block No. in Cache Mem.
This means, Block 0 of Main Memory (of any set), can only be mapped into Block 0 of Cache Memory.
In other words, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set.

Consider Pentium Processor Cache

Size of Main Mem:	**4GB** = **2^{32}**
Size of Cache Mem:	**8KB** = **2^{13}** … this is treated as One Set
Hence Size of Set:	**8KB** = **2^{13}**
Size of Cache Block (Line):	**32** bytes (words) = **2^5**
No. of Blocks in a set:	Size of Set (**2^{13}**) ÷ Size of Block (**2^5**) = **2^8** = **256**
No. of Sets in Main Mem:	Size of Main Mem (**2^{32}**) ÷ Size of Set (**2^{13}**) = **2^{19}**
No. of Sets in Cache Mem:	1
Main Mem address:	**32 bits** (because main Mem is of **4GB** = **2^{32}**)

Tag Size:
Since, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set, the Tag has to only indicate the Set No. of Main Memory, from which the block is present.
As Main Memory has 2^{19} sets, the **Tag** size is **19 bits**.

Searches:
If we need Block 0 if Main Memory, we only need to search Block 0 of Cache Memory.
Hence, we need to do only **1 search** in Cache Memory to know if it is a Hit or a Miss.

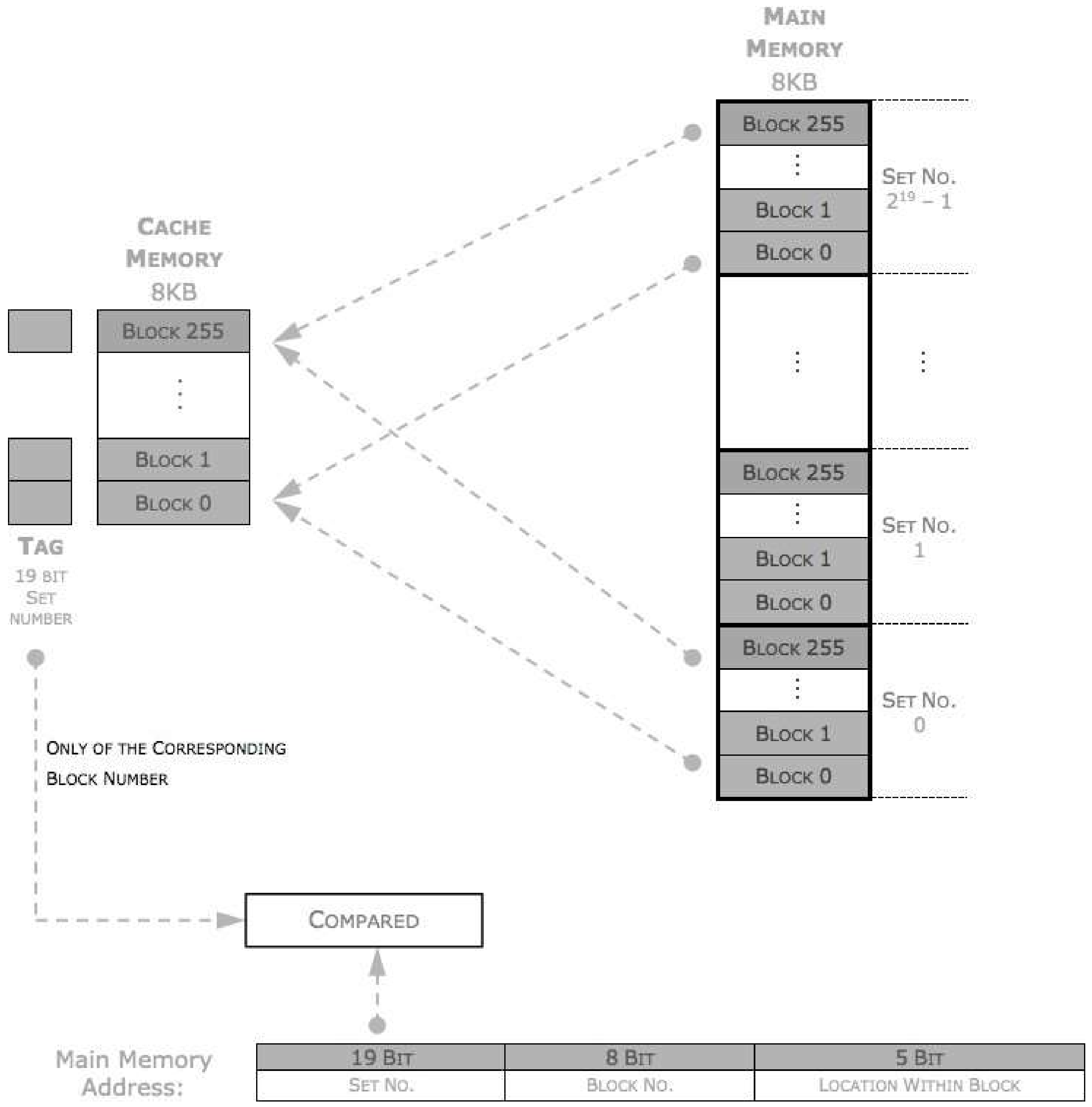
Method of Searching:
The Processor issues a 32 bit Main Memory address. It can be divided as:

Main Memory Address:	**19 BIT**	**8 BIT**	**5 BIT**
	Set No.	Block No.	Location Within Block

First we look at the block no. we need, to know where we need to search.
We then look at the Set No. that we need and compare it with the Tag of the corresponding Block no in the Cache Memory.
Assume the Main Memory address is 5:0:6. This means we need location 6 of Block 0 of set 5.
We go to Block 0 of Cache Memory.
It has a Tag, which gives the Set No of Main Memory whose Block 0 is present in Cache Memory.
These two set numbers are compared. **If they are equal, it's a HIT, else it's a Miss.**

Advantage
In **1 Search** we know if it is a Hit or a Miss. **Tag Size** = **19 bits**.

Drawback
Since the method is **very rigid**, the **Hit Ratio drops tremendously.**


Main Memory 8KB
Block 255
Block 1
Block 0
Set No. 2^19 – 1
Set No. 1
Set No. 0
Cache Memory 8KB
Tag
19 bit Set number
Only of the Corresponding Block Number
Compared
Main Memory Address:
19 Bit
Set No.
8 Bit
Block No.
5 Bit
Location Within Block

Set Associative Mapping (Two Way Set Associative Mapping)

Two way Set Associative Mapping technique states:
A block of Main Memory can only be mapped into the same corresponding Block No. of Cache Memory, in any of the two sets.
Since there are two ways of Mapping, its called Two Way Set Associative Mapping.
We treat the entire Cache as **Two Sets**. The Main Memory is divided into Sets, subdivided into Blocks.
A Block of Main Mem. (of any set), can only be mapped into the same Block No. in Cache Memory again of any set. This means, Block 0 of Main Memory (of any set), can only be mapped into Block 0 of Cache Memory, into one of its two sets.
In other words, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set.

Consider Pentium Processor Cache (This is Actually how Pentium's Cache is implemented)

Size of Main Mem:	**4GB** = **2^{32}**
Size of Cache Mem:	**8KB** = **2^{13}** … this is treated as Two Sets
Hence Size of Set:	**4KB** = **2^{12}**
Size of Cache Block (Line):	**32** bytes (words) = **2^5**
No. of Blocks in a set:	Size of Set (**2^{12}**) ÷ Size of Block (**2^5**) = **2^7** = **128**
No. of Sets in Main Mem:	Size of Main Mem (**2^{32}**) ÷ Size of Set (**2^{12}**) = **2^{20}**
No. of Sets in Cache Mem:	2
Main Mem address:	**32 bits** (because main Mem is of **4GB** = **2^{32}**)

Tag Size:
Since, Block 0 of Cache Memory can only contain Block 0 of Main Memory but of any Set, the Tag has to only indicate the Set No. of Main Memory, from which the block is present.
As Main Memory has 2^{20} sets, the **Tag** size is **20 bits**.

Searches:
If we need Block 0 if Main Memory, we only need to search Block 0 of Cache Memory, but in any of the two sets. Hence, we need **2 searches** in Cache Memory to know if it is a Hit or a Miss.

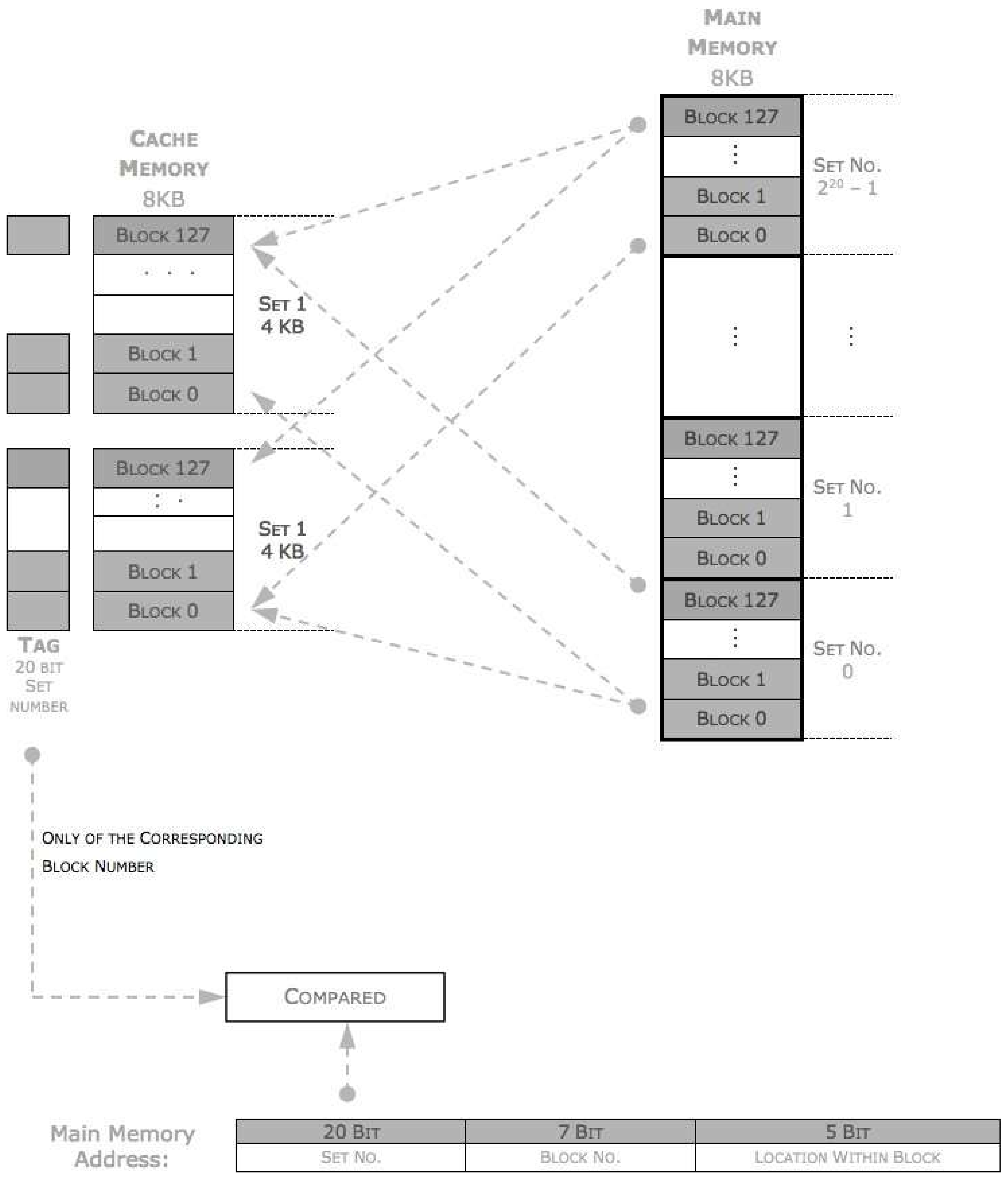
Method of Searching:
The Processor issues a 32 bit Main Memory address. It can be divided as:

Main Memory Address:	**20 BIT**	**7 BIT**	**5 BIT**
	Set No.	Block No.	Location Within Block

First we look at the block no. we need, to know where we need to search.
We then look at the Set No. that we need and compare it with the Tags of the corresponding Block no in the Cache Memory, in any of the two sets.
Assume the Main Memory address is 5:0:6. This means we need location 6 of Block 0 of set 5.
We go to Block 0 of Cache Memory, in both sets.
It has a Tag, which gives the Set No of Main Memory whose Block 0 is present in Cache Memory.
These required Set no. is compared with the two Tags. **If found in any of the 2, it's a HIT, else Miss.**

Advantage
In **2 Searches** we know if it is a Hit or a Miss. **Tag Size** = **20 bits**.

Drawback
Since the method is **flexible**, it significantly **increases** the **Hit Ratio.**


MAIN MEMORY 8KB
BLOCK 127
BLOCK 1
BLOCK 0
SET NO. $2^{20} - 1$
SET NO. 1
SET NO. 0
CACHE MEMORY 8KB
SET 1 4 KB
SET 1 4 KB
TAG 20 BIT SET NUMBER
ONLY OF THE CORRESPONDING BLOCK NUMBER
COMPARED
Main Memory Address:
20 BIT
7 BIT
5 BIT
SET NO.
BLOCK NO.
LOCATION WITHIN BLOCK

Expanding the logic of set associative cache further, we can derive the following conclusion:

No of Ways	No of Searches	Tag Size	No of Blocks in a Set
2 Way	2	20 bits	128
4 Way	4	21 bits	64
8 Way	8	22 bits	32
16 Way	16	23 bits	16
32 Way	32	24 bits	8
64 Way	64	25 bits	4
128 Way	128	26 bits	2
256 Way	256	27 bits	1
This becomes exactly the same as Fully Associative: 256 Searches, 27-bit Tag			

PAGE REPLACEMENT NUMERICALS *(Very Important)*

1) Consider Main Memory has 3 page frames (0,1,2).
Processor requires pages from Virtual Memory in the following sequence of page numbers: 2,3,2,1,5,2,4,5,3,2,5,2. Show and compare the implementation of FIFO, LRU and LFU.

FIFO	2	3	2	1	5	2	4	5	3	2	5	2	
Frame 0	2^{*}	2^{*}	2^{*}	2^{*}	5	5	5^{*}	5^{*}	3	3	3	3^{*}	Hit Ratio = 0.25
Frame 1		3	3	3	3^{*}	2	2	2	2^{*}	2^{*}	5	5	
Frame 2				1	1	1^{*}	4	4	4	4	4^{*}	2	
			HIT					**HIT**		**HIT**			

LRU	2	3	2	1	5	2	4	5	3	2	5	2	
Frame 0	2	2	2	2	2	2	2	2	3	3	3	3	Hit Ratio = 0.42
Frame 1		3	3	3	5	5	5	5	5	5	5	5	
Frame 2				1	1	1	4	4	4	2	2	2	
			HIT			**HIT**		**HIT**			**HIT**	**HIT**	

LFU	2	3	2	1	5	2	4	5	3	2	5	2	
Frame 0	2 (1)	2 (1)	2 (2)	2 (2)	2 (2)	2 (3)	2 (3)	2 (3)	2 (3)	2 (4)	2 (4)	2 (5)	Hit Ratio = 0.50
Frame 1		3 (1)	3 (1)	3 (1)	5 (1)	5 (1)	5 (1)	5 (2)	5 (2)	5 (2)	5 (3)	5 (3)	
Frame 2				1 (1)	1 (1)	1 (1)	4 (1)	4 (1)	3 (1)	3 (1)	3 (1)	3 (1)	
			HIT			**HIT**		**HIT**		**HIT**	**HIT**	**HIT**	

2) Consider Main Memory has 4 page frames (0,1,2,3)
Processor requires pages from Virtual Memory in the following sequence of page numbers: 7,5,3,2,1,0,4,1,6,7,4,2. Show and compare the implementation of FIFO, LRU and LFU.

FIFO	7	5	3	2	1	0	4	1	6	7	4	2	
Frame 0													Hit Ratio = 0.16
Frame 1													
Frame 2													
Frame 3													

LRU	7	5	3	2	1	0	4	1	6	7	4	2	
Frame 0													Hit Ratio = 0.16
Frame 1													
Frame 2													
Frame 3													

LFU	7	5	3	2	1	0	4	1	6	7	4	2	
Frame 0													Hit Ratio = 0.16
Frame 1													
Frame 2													
Frame 3													

OPTIMAL REPLACEMENT ALGORITHM

1. Another proposed replacement algorithm is called "**Optimal Replacement Algorithm**" (**OPT**)
2. We need to know beforehand, the order in which, pages will be used in the near future.
3. **The pages that will be used sooner, will be retained.**
4. **The pages that will not be used for the longest time will be replaced.**
5. Of course it is impossible to predict the pages to be used in the future.
6. But if we have had some **sample runs of the program in a simulator** then using that data as a reference, we can make safe predictions of the behavior of the program.

Consider Main Memory has 3 page frames (0,1,2).
Calculate Hits and Misses and suggest the best algorithm out of **FIFO, LRU and OPT**
Processor requires pages from Virtual Memory in the following sequence of page numbers: **4,7,3,0,1,7,3,8,5,4,5,3,4,7. (Sem 4 Comps IT Dec 2015 Exam question – 10 marks)**

FIFO	4	7	3	0	1	7	3	8	5	4	5	3	4	7
Frame 0	4 *	4 *	4 *	0	0	0 *	3	3	3 *	4	4	4	4	4 *
Frame 1		7	7	7 *	1	1	1 *	8	8	8 *	8 *	3	3	3
Frame 2			3	3	3 *	7	7	7 *	5	5	5	5 *	5 *	7
	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	**HIT**	MISS	**HIT**	MISS

Total Attempts: 14. Hits = 2. Misses (Page Faults) = 12. **Hit Ratio = 2/14 = 0.117**

LRU	4	7	3	0	1	7	3	8	5	4	5	3	4	7
Frame 0	4	4	4	0	0	0	3	3	3	4	4	4	4	4
Frame 1		7	7	7	1	1	1	8	8	8	8	3	3	3
Frame 2			3	3	3	7	7	7	5	5	5	5	5	7
	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	MISS	**HIT**	MISS	**HIT**	MISS

Total Attempts: 14. Hits = 2. Misses (Page Faults) = 12. **Hit Ratio = 2/14 = 0.117**

OPT	4	7	3	0	1	7	3	8	5	4	5	3	4	7
Frame 0	4	4	4	0	1	1	1	8	5	5	5	5	5	7
Frame 1		7	7	7	7	7	7	7	7	4	4	4	4	4
Frame 2			3	3	3	3	3	3	3	3	3	3	3	3
	MISS	MISS	MISS	MISS	MISS	**HIT**	**HIT**	MISS	Miss	Miss	**HIT**	**HIT**	**HIT**	Miss

Total Attempts: 14. Hits = 5. Misses (Page Faults) = 9. **Hit Ratio = 5/14 = 0.357**