

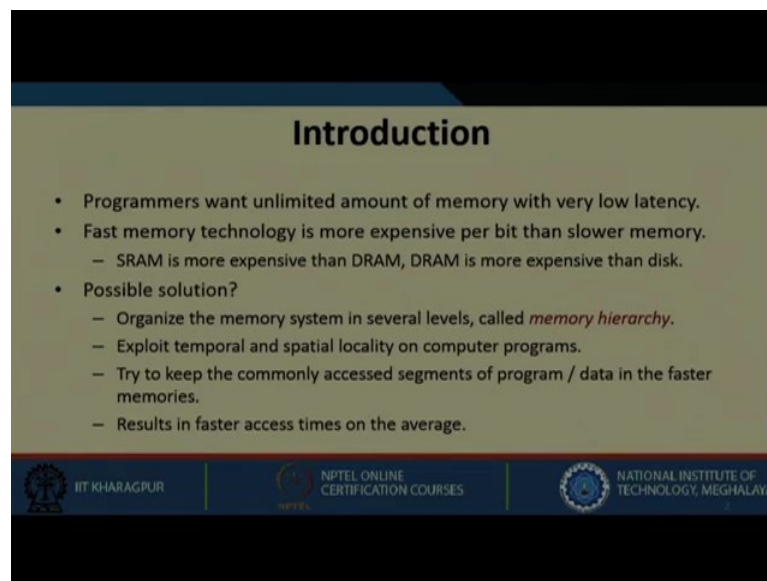
Computer Architecture and Organization
Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture - 28
Memory Hierarchy Design (Part I)

Welcome to week 6. In this week we will be looking into Memory Hierarchy Design, and of course cache memory. In previous week we discussed about memories, we discussed about static memories, dynamic memories, RAM; we also discussed about how we can actually design larger memory modules from smaller memory modules and memory interleaving.




In this week we will be looking mostly into how we can make memory faster by incorporating some strategies. One of the methods that we will be seeing in more detail is the cache memory. And we will also be focusing on memory hierarchy design.

(Refer Slide Time: 01:37)



Introduction

- Programmers want unlimited amount of memory with very low latency.
- Fast memory technology is more expensive per bit than slower memory.
 - SRAM is more expensive than DRAM, DRAM is more expensive than disk.
- Possible solution?
 - Organize the memory system in several levels, called *memory hierarchy*.
 - Exploit temporal and spatial locality on computer programs.
 - Try to keep the commonly accessed segments of program / data in the faster memories.
 - Results in faster access times on the average.

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES  NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

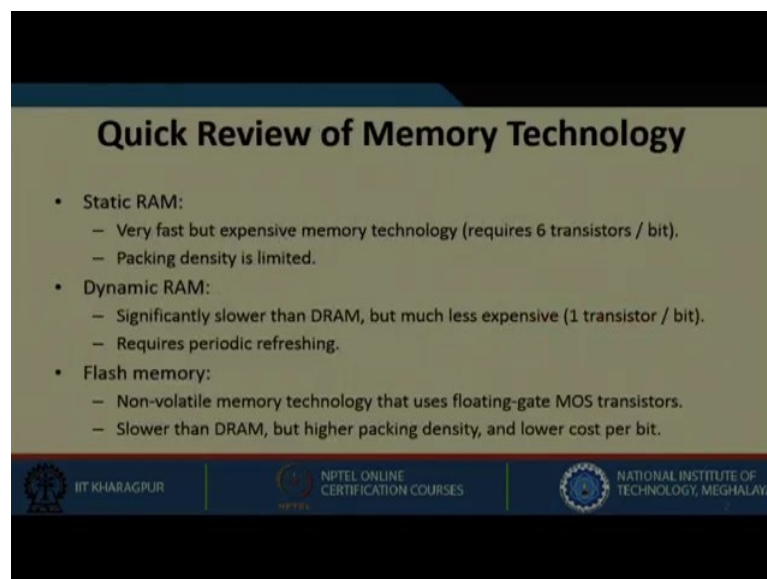
The programmers always want unlimited amount of memory with a very low latency. We need high speed and we also need more memory. We have also seen that fast memory technology is more expensive per bit than slower memory. SRAM is much more expensive than DRAM, but DRAM is again more expensive than disk. So, SRAM cannot be made much larger, where DRAM can be made much larger compared to

SRAM. Again DRAM cannot be made as large as disk, and disk speed cannot match the speed of DRAM.

So, what is the possible solution? Organize the memory system in several levels, which is called memory hierarchy, and exploit both temporal and spatial locality of computer programs. We will look into the details of what is temporal and spatial locality. And we also try to keep the commonly accessed segment of programs or data in a faster memory called cache memory. So by this, what we mean is that the frequently used data or instructions can be kept in a high speed memory, because a particular data which I am requiring now it might happen I will be requiring it after some time again.

So, instead of keeping it in a slower memory let us keep it in a fast memory and as and when required by the processor it can get it from the faster memory and not from the slower memory.

(Refer Slide Time: 03:51)



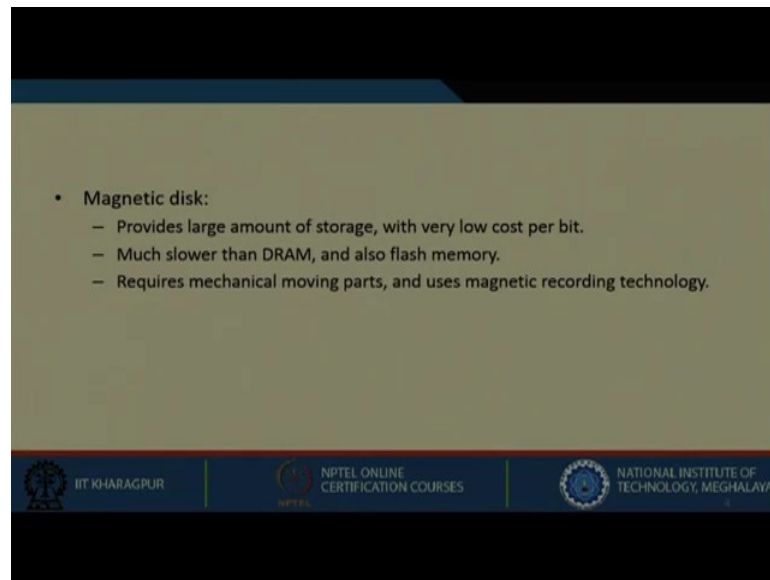
The slide is titled "Quick Review of Memory Technology" in bold black text. It contains a bulleted list of memory types and their characteristics. The background is a light beige color with a dark blue header and footer. The footer includes logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

- **Static RAM:**
 - Very fast but expensive memory technology (requires 6 transistors / bit).
 - Packing density is limited.
- **Dynamic RAM:**
 - Significantly slower than DRAM, but much less expensive (1 transistor / bit).
 - Requires periodic refreshing.
- **Flash memory:**
 - Non-volatile memory technology that uses floating-gate MOS transistors.
 - Slower than DRAM, but higher packing density, and lower cost per bit.

This results in faster access time on an average. Let us have a quick review of the memory technology that we discussed last week. Static RAM, which is very fast but expensive memory as it requires 6 transistors per bit, and the packaging density is limited. So, within a small area we cannot have very large memory in place. Whereas, dynamic RAM is significantly slower than SRAM, but much less expensive that is only one transistor per bit is required. And it also requires periodic refreshing which is not required in static RAM.

So, DRAM is much slower than SRAM, but it is much less expensive, and also it requires periodic refreshing which is not required in SRAM. And flash memory is a non-volatile memory technology that uses floating gate MOS transistors. It is of course slower than DRAM, but has higher packaging density and lower cost per bit.

(Refer Slide Time: 05:09)



• Magnetic disk:

- Provides large amount of storage, with very low cost per bit.
- Much slower than DRAM, and also flash memory.
- Requires mechanical moving parts, and uses magnetic recording technology.

The slide features a dark blue header and footer. The footer contains three logos: IIT Kharagpur, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

And magnetic disk provides large amount of storage and the cost per bit is also pretty less, but it is much slower than DRAM and also flash memory. And compared to other memories this requires a mechanical moving part and uses magnetic recording technology. The disk moves around and we actually get the data from different tracks and sectors. So, there is a moving part, whereas in DRAM or SRAM no such thing is there.

(Refer Slide Time: 05:53)

Memory Hierarchy

- The memory system is organized in several levels, using progressively faster technologies as we move towards the processor.
 - The entire addressable memory space is available in the largest (but slowest) memory (typically, magnetic disk or flash storage).
 - We incrementally add smaller (but faster) memories, each containing a subset of the data stored in the memory below it.
 - We proceed in steps towards the processor.

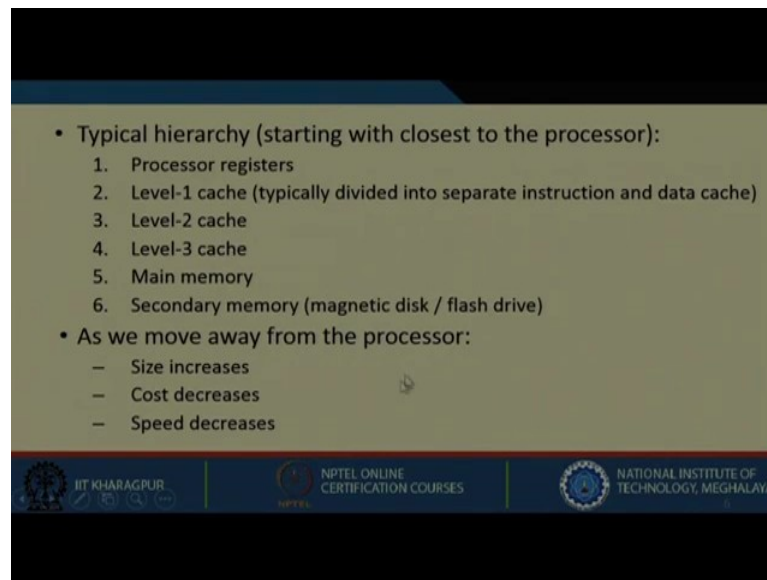
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Coming to memory hierarchy, the memory system is organized in several levels. By hierarchy we mean it is divided into many levels using progressively faster technologies as we move towards the processor. Thus there are different levels of memory, and the level that is closest to the processor is faster, and which are little further from the processor are slower.

The entire addressable memory space is available in the largest, but slowest memory; typically magnetic disk or flash storage. The addressable space can be as large as data on the disk, but we are actually implementing the levels one by one. So, at the lowest level where we have cache that is much smaller, then we go to next level which can be L2 cache or it can be main memory, it can be little larger. But how we can speed up? We can transfer data by replacing the data that is currently in that particular fast memory we will move it to the slower memory, again from the slower memory will bring to the faster memory. This is how we perform the things.

So, we incrementally add smaller, but faster memories each containing a subset of data stored in memory below it. We proceed in steps towards the processor.

(Refer Slide Time: 08:09)



• Typical hierarchy (starting with closest to the processor):

1. Processor registers
2. Level-1 cache (typically divided into separate instruction and data cache)
3. Level-2 cache
4. Level-3 cache
5. Main memory
6. Secondary memory (magnetic disk / flash drive)

• As we move away from the processor:

- Size increases
- Cost decreases
- Speed decreases

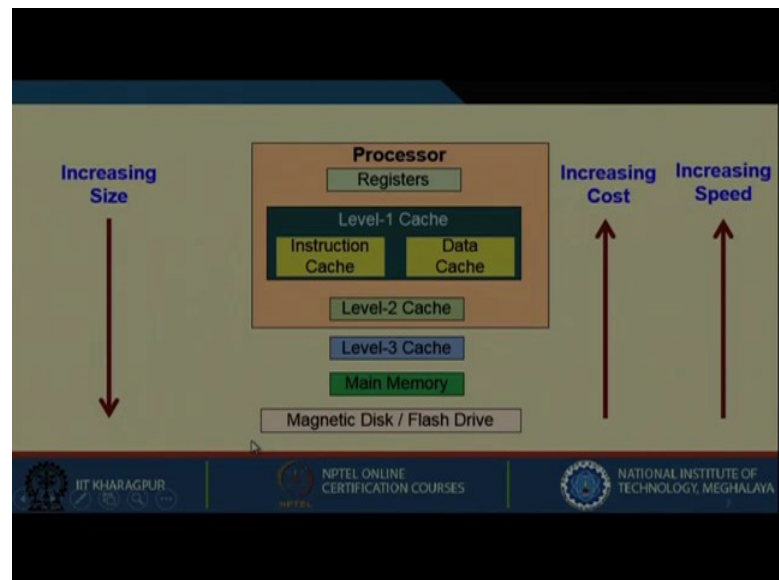
The slide footer includes logos for IIT Kharagpur, NPTEL Online Certification Courses, and the National Institute of Technology, Meghalaya.

Let us see this. Typical hierarchy starts with the one closest to processor, which are the processor registers. Then we have Level-1 cache, typically divided into separate instruction and data cache. We have already talked about Harvard and von Neumann architecture in the first week. If you recall we said that if we have separate data and instruction memory, then instruction fetch and data access can be done at the same time.

So, we typically divide Level-1 into instruction cache and data cache; we can have level 2 cache, then level 3 cache, we then have main memory, and finally we have secondary memory. So, processor cache will be the smallest one, then the level 1 cache and so on, the secondary memory will be the largest memory.

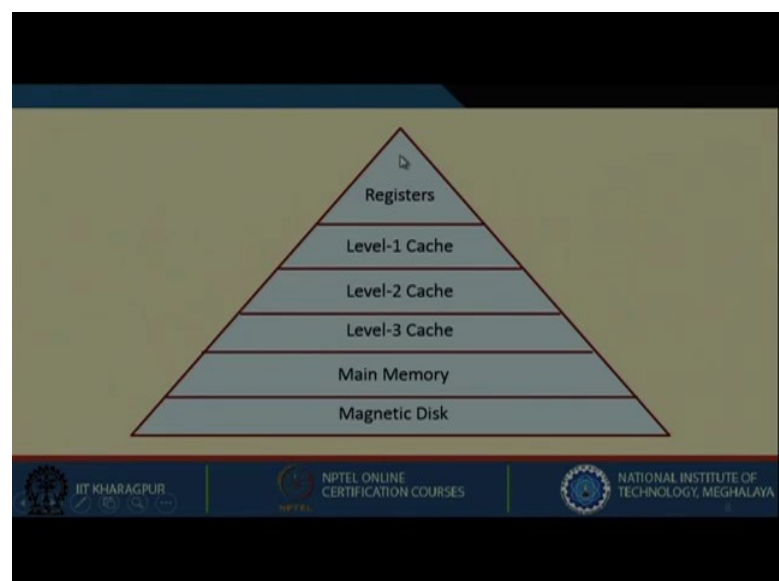
As we move away from processor the size increases. So, this is the smallest one, then the size increases little more, little more. The cost also decreases, because as you are closest to the processor it is much faster, but as we are moving away from the processor the cost slowly decreases, but at the same time the speed also decreases. So, this is the trade off you can see.

(Refer Slide Time: 10:05)



Let us see this. This is processor register, level 1 cache; we have instruction and data cache, then level 2 cache, level 3 cache, main memory and magnetic disk. As we move from processor to magnetic disk the size increases, so the size of magnetic disk is the maximum. But as you move up the speed increases as well as the cost increases; so the cost becomes much more as you are moving to the memory that is closest to the processor.

(Refer Slide Time: 10:58)



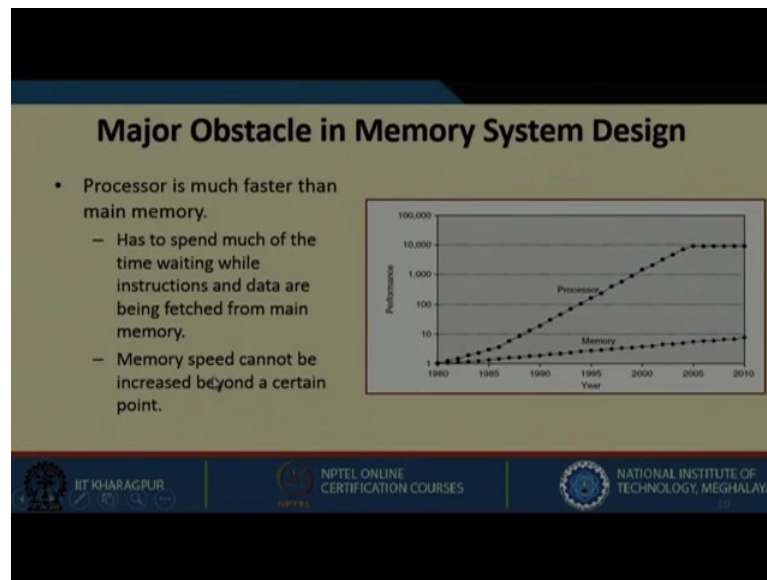
So, this is basically a pyramid structure that shows registers, then level 1 cache, then level 2 cache, and so on. So, the size is increasing, but there are few things that are also decreasing with the size.

(Refer Slide Time: 11:18)

A Comparison			
Level	Typical Access Time	Typical Capacity	Other Features
Register	300-500 ps	500-1000 B	On-chip
Level-1 cache	1-2 ns	16-64 KB	On-chip
Level-2 cache	5-20 ns	256 KB – 2 MB	On-chip
Level-3 cache	20-50 ns	1-32 MB	On or off chip
Main memory	50-100 ns	1-16 GB	
Magnetic disk	5-50 ms	100 GB – 16 TB	

Now, this is a comparison that has been made. For registers the typical access time is of the order of picosecond, level 1 cache this is 1 to 2 nanosecond, level 2 cache is 5 to 20 nanosecond. So, the access time is increasing slowly, and at the same time if you see the capacity, it is also increasing. The L1 cache and L2 cache can be on chip and L3 can be off chip or it can also be on chip.

(Refer Slide Time: 12:02)



So, what is the major obstacle in memory system design? We have already seen this slide before. Processor is much faster than main memory; this is the growth of the processor speed and this is the growth of the memory. So, basically you see this gap is always increasing. So, memory speed cannot be increased beyond a certain point that is why we are coming up with many techniques through which we can actually increase the speed.

(Refer Slide Time: 12:54)

Impact of Processor / Memory Performance Gap

Year	CPU Clock	Clock Cycle	Memory Access	Minimum CPU Stall Cycles
1986	8 MHz	125 ns	190 ns	$190 / 125 - 1 = 0.5$
1989	33 MHz	30 ns	165 ns	$165 / 30 - 1 = 4.5$
1992	60 MHz	16.6 ns	120 ns	$120 / 16.6 - 1 = 6.2$
1996	200 MHz	5 ns	110 ns	$110 / 5 - 1 = 21.0$
1998	300 MHz	3.33 ns	100 ns	$100 / 3.33 - 1 = 29.0$
2000	1 GHz	1 ns	90 ns	$90 / 1 - 1 = 89.0$
2002	2 GHz	0.5 ns	80 ns	$80 / 0.5 - 1 = 159.0$
2004	3 GHz	0.33 ns	60 ns	$60 / 0.33 - 1 = 179.0$

Ideal memory access time = 1 CPU cycle
Real memory access time \gg 1 CPU cycle

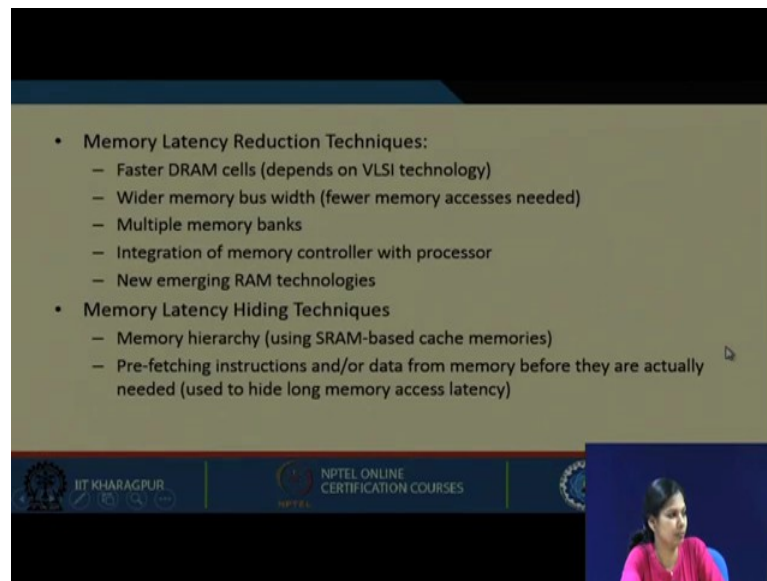
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us see the impact of processor and memory performance gap over the years. You can see this is the CPU clock, this will be the clock cycle time, and this is the memory access

time. So, what is happening is that the clock cycle time and the memory access time gap is increasing; the processor clock speed is becoming higher, with that the clock cycle time is getting reduced, but the gap between clock cycle time and the memory access time is more.

The minimum CPU stall cycles can be given by this. The data is provided till 2004 that shows that minimum CPU stall cycle will be 179.

(Refer Slide Time: 14:21)

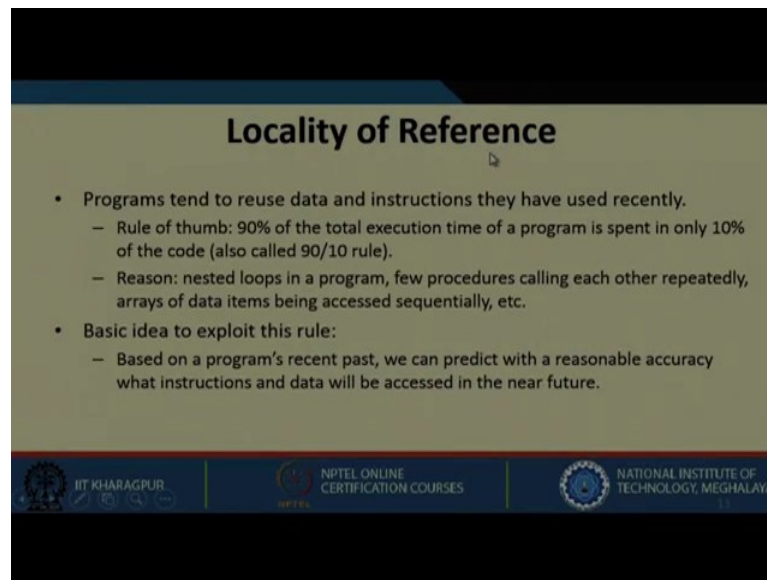


- Memory Latency Reduction Techniques:
 - Faster DRAM cells (depends on VLSI technology)
 - Wider memory bus width (fewer memory accesses needed)
 - Multiple memory banks
 - Integration of memory controller with processor
 - New emerging RAM technologies
- Memory Latency Hiding Techniques
 - Memory hierarchy (using SRAM-based cache memories)
 - Pre-fetching instructions and/or data from memory before they are actually needed (used to hide long memory access latency)

Memory latency reduction techniques say how we can reduce the access time. If it is reduced what are the techniques that can be used. One is faster DRAM cell that will depend on VLSI technology, and wider memory bus width with fewer memory access needed. So, we access once and we get the data all together.

So, actually you are using multiple memory banks with memory interleaving, integration of memory controller with processor, we can also use some emerging RAM technologies. And under memory latency hiding techniques we have memory hierarchy using SRAM-based cache memory. So, we are having a fast memory and we will see that most of the access will be made to this particular memory. Prefetching instruction or data from memory before they are actually needed will also help. Prefetching is a technique that can be used to hide this memory latency.

(Refer Slide Time: 15:50)



Locality of Reference

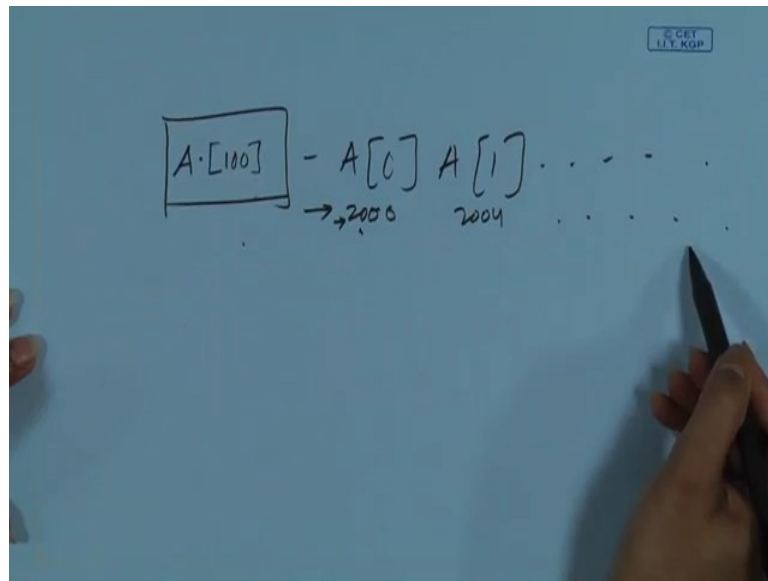
- Programs tend to reuse data and instructions they have used recently.
 - Rule of thumb: 90% of the total execution time of a program is spent in only 10% of the code (also called 90/10 rule).
 - Reason: nested loops in a program, few procedures calling each other repeatedly, arrays of data items being accessed sequentially, etc.
- Basic idea to exploit this rule:
 - Based on a program's recent past, we can predict with a reasonable accuracy what instructions and data will be accessed in the near future.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, we come to locality of reference. There is a property that programs tend to reuse data and instruction they have used recently; that means, an instruction that is used at time t it is much likely that it will be used again at some point of time very soon. This is called locality of reference; the rule of thumb says that 90% of the total execution time of the program is spend in only 10% of the code. This is called 90/10 rule.

So, only 10% of the code is been used because of loops. If you consider a loop where certain statement and certain instructions are getting executed repeatedly. If we bring those instructions into some faster memory, you can actually have a better access time because you have brought the data from a slower memory into a faster memory, and now you are accessing repeatedly from the faster memory. That is why cache can be helpful in such scenario although we are bringing the data from one memory to another memory, but in turn we are getting advantage out of it.

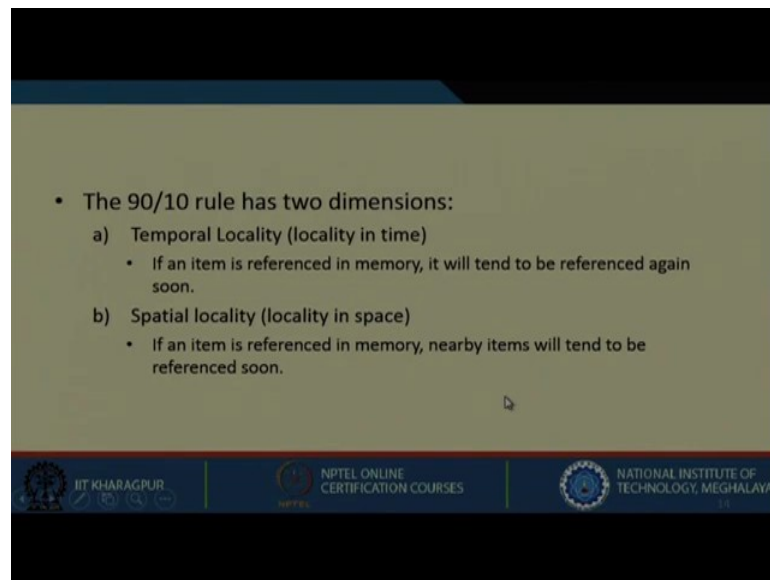
(Refer Slide Time: 18:12)



Let us say this is an array of 100 elements. So, you will access first element, you will access next element, and so on. So, actually you are making an access to some memory location let us say 2000, then 2004, and so on. So, if an instruction or data is required which is in some location, it is expected that data in the nearby locations is also required in the near future. So, instead of bringing only this we can bring the whole set of array together into the cache. So, this is where it helps and this locality of reference is coming into picture. So, there are two things; we call it spatial locality of reference and temporal locality of reference.

The basic idea is that based on program's recent past we can predict with a reasonable accuracy what instructions and data will be accessed in near future.

(Refer Slide Time: 19:35)



The slide displays the 90/10 rule dimensions in a bulleted format. The main bullet point is 'The 90/10 rule has two dimensions:', followed by two sub-points: 'a) Temporal Locality (locality in time)' and 'b) Spatial locality (locality in space)'. Each sub-point has a further bullet point explaining the concept. The slide also features a footer with logos for IIT Kharagpur, NPTEL, and the National Institute of Technology, Meghalaya.

- The 90/10 rule has two dimensions:
 - a) Temporal Locality (locality in time)
 - If an item is referenced in memory, it will tend to be referenced again soon.
 - b) Spatial locality (locality in space)
 - If an item is referenced in memory, nearby items will tend to be referenced soon.

IIT KHARAGPUR
NPTEL
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

The 90/10 rule has two dimensions, one is called temporal locality --- locality in time; that means, if I am accessing an element at time t it is likely that I will be accessing that same element at time $t + \text{something}$. So, if an item is referenced in memory it will tend to be referenced again very soon because of loops.

So, if an item is referenced in memory nearby items will tend to be referenced soon; that means, let us say we have written a code and that code takes some, say 20 words, to store that particular program. Now, if you take one word at a time it will not help because when you are bringing one particular word it is likely that we require 19 more words associated with that program. So, why not to bring the entire thing into cache memory such that next time when you are accessing you will get it from the cache memory and not from the main memory. So, this is spatial locality.

(Refer Slide Time: 21:11)

(a) Temporal Locality

- Recently executed instructions are likely to be executed again very soon.
- Example: computing factorial of a number.

```
fact = 1;  
for k = 1 to N  
    fact = fact * k;
```

```
ADDI    $t1,$zero,1  
ADDI    $t2,$zero,N  
ADDI    $t3,$zero,1  
Loop:   MUL    $t1,$t1,$t3  
        ADDI    $t3,$t3,1  
        SGT     $t4,$t3,$t2  
        BNEZ    $t4,Loop
```

- The four instructions in the loop are executed more frequently than the others.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Let us take this example of temporal locality; recently executed instructions are likely to be executed again very soon. The example is computing factorial of a number.

It says that because of loop structure this instruction will not be required for us to bring it from a slower memory, because we will bring it once from slowest memory to fastest memory and then we will keep it there. So, this is temporal locality this is an example of temporal locality.

(Refer Slide Time: 24:04)

(b) Spatial Locality

- Instructions residing close to a recently executing instruction are likely to be executed soon.
- Example: accessing elements of an array.

```
sum = 0;  
for k = 1 to N  
    sum = sum + A[k];
```

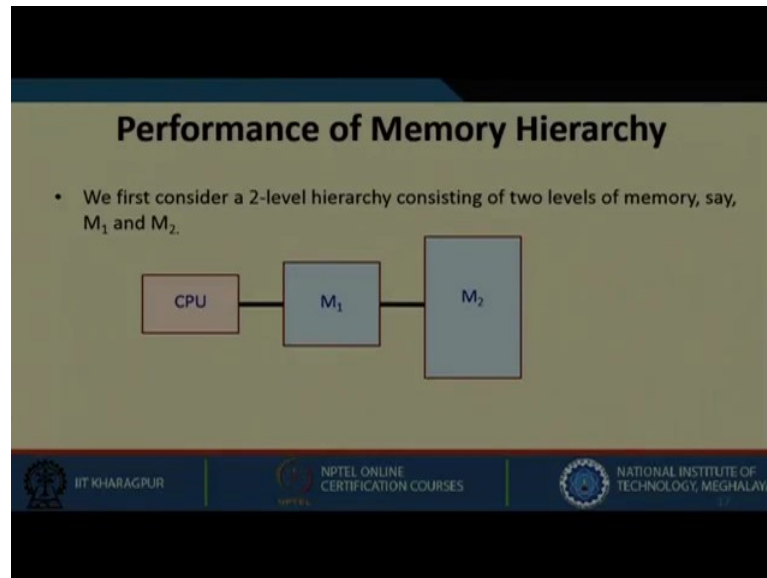
```
SUB      $t1,$t1,$t1  
ADDI     $t2,$zero,N  
ADDI     $t3,$zero,1  
ADDI     $t5,$zero,A  
Loop:    LW      $t8,0($t5)  
        ADD     $t1,$t1,$t8  
        ADDI    $t3,$t3,1  
        SGT     $t4,$t3,$t2  
        BNEZ    $t4,Loop
```

- Performance can be improved by copying the array into cache memory.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

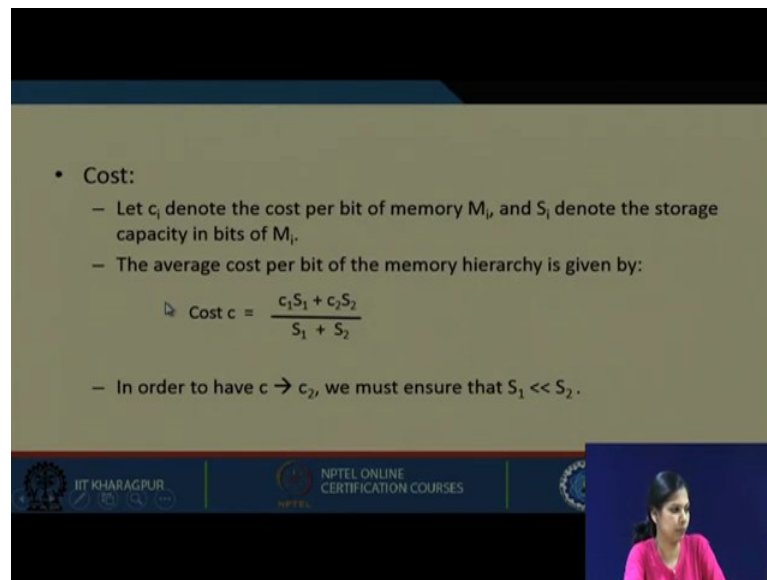
Now, let us see spatial locality. Instructions residing close to recently executing instructions are likely to be executed soon; that means, this instruction is in close proximity of other instruction. So, if I am bringing this particular instruction it is also better that we bring some more instruction that is in the close proximity of this instruction.

(Refer Slide Time: 26:58)



We first consider a 2 level hierarchy consisting of two levels of memory, say M_1 and M_2 . CPU is first hitting M_1 , and if it is found here it will take it send the data to CPU, and if it is not found it is brought from M_2 to M_1 , and then may be transferred to CPU.

(Refer Slide Time: 27:26)



• Cost:

- Let c_i denote the cost per bit of memory M_i , and S_i denote the storage capacity in bits of M_i .
- The average cost per bit of the memory hierarchy is given by:

$$\text{Cost } c = \frac{c_1 S_1 + c_2 S_2}{S_1 + S_2}$$

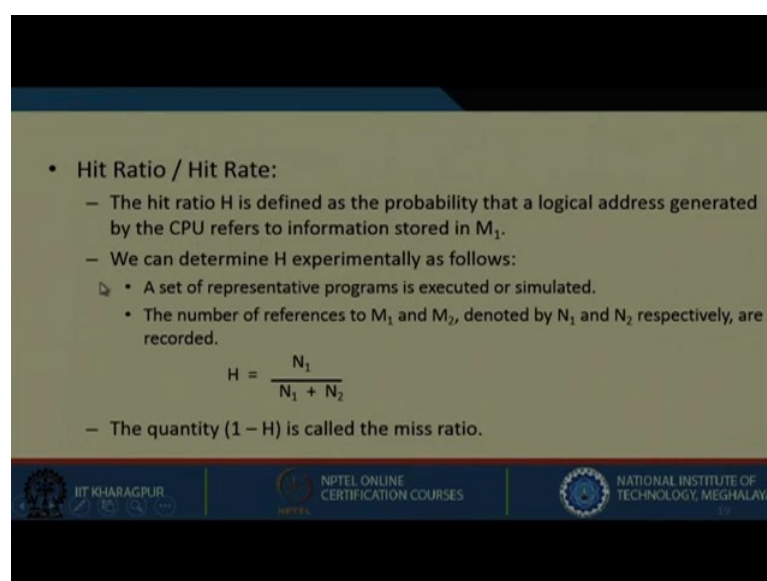
- In order to have $c \rightarrow c_2$, we must ensure that $S_1 \ll S_2$.

The slide is part of an NPTEL presentation from IIT Kharagpur. It features a blue header and footer with logos for IIT Kharagpur, NPTEL, and the National Institute of Technology, Meghalaya. A small video inset in the bottom right corner shows a female presenter.

So, how we can calculate the cost? Let c_i denote the cost per bit of memory M_i , and S_i denote the storage capacity in bits of M_i . The average cost per bit of the memory hierarchy is given by this expression.

What we are trying to say is that c will be roughly equivalent to c_2 ; that is, cost should be less, but for that we must ensure that S_1 is much less than S_2 , the size of M_1 memory should be less than size of M_2 .

(Refer Slide Time: 28:49)



• Hit Ratio / Hit Rate:

- The hit ratio H is defined as the probability that a logical address generated by the CPU refers to information stored in M_1 .
- We can determine H experimentally as follows:

- A set of representative programs is executed or simulated.
- The number of references to M_1 and M_2 , denoted by N_1 and N_2 respectively, are recorded.

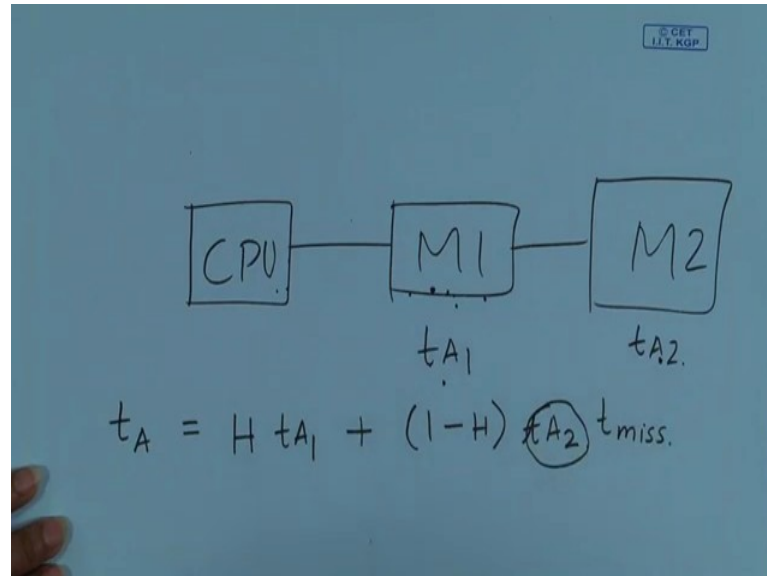
$$H = \frac{N_1}{N_1 + N_2}$$

- The quantity $(1 - H)$ is called the miss ratio.

This slide continues the NPTEL presentation from IIT Kharagpur. It includes the same blue header and footer with logos for IIT Kharagpur, NPTEL, and the National Institute of Technology, Meghalaya. The video inset in the bottom right corner shows the same female presenter.

Coming to hit ratio or hit rate what do you mean by that? The hit ratio H is defined as the probability that a logical address generated by the CPU refers to information stored in M .

(Refer Slide Time: 29:15)

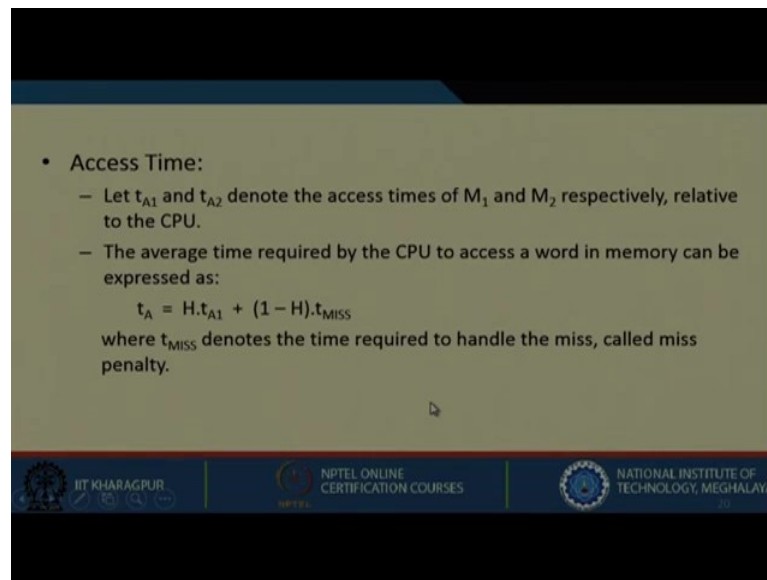


So, let us see this; this is your CPU and you have M1 level and you have M2 level, and we are saying the CPU will be hitting this particular memory first. This means CPU will hit M1 and it will get the data from M1; that is hit ratio. So, the percentage time the data is found in M1 is the hit ratio.

So, hit ratio H is defined as the probability that logical address generated by the CPU refers to the information stored in M1; that means, the data which I am looking for is present in M1. We can determine H experimentally as follows. A set of representative programs is executed or simulated; then the number of references to M1 and M2 denoted by n_1 and n_2 respectively is measured, and then hit ratio can be n_1 divided by $n_1 + n_2$.

The quantity $1-H$ is called the miss ratio; that means, the number of times it is not found in M1 cache.

(Refer Slide Time: 31:10)



• Access Time:

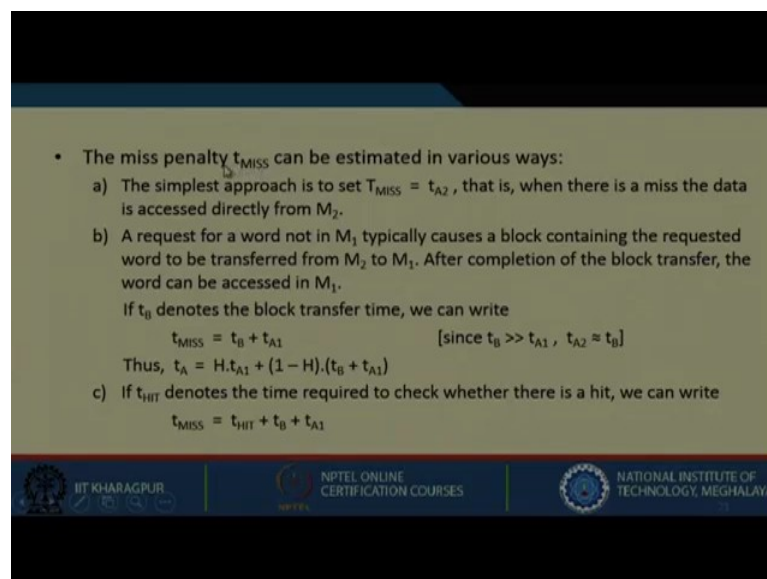
- Let t_{A1} and t_{A2} denote the access times of M_1 and M_2 respectively, relative to the CPU.
- The average time required by the CPU to access a word in memory can be expressed as:
$$t_A = H.t_{A1} + (1 - H).t_{MISS}$$
where t_{MISS} denotes the time required to handle the miss, called miss penalty.

20

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, now let us see the access time. Let t_{A1} and t_{A2} denote the access times of M_1 and M_2 respectively relative to CPU. How we can actually tell the average time required by CPU to access the word. It is given by this expression.

(Refer Slide Time: 33:56)



• The miss penalty t_{MISS} can be estimated in various ways:

- The simplest approach is to set $t_{MISS} = t_{A2}$, that is, when there is a miss the data is accessed directly from M_2 .
- A request for a word not in M_1 typically causes a block containing the requested word to be transferred from M_2 to M_1 . After completion of the block transfer, the word can be accessed in M_1 .
If t_B denotes the block transfer time, we can write
$$t_{MISS} = t_B + t_{A1} \quad [\text{since } t_B \gg t_{A1}, t_{A2} \approx t_B]$$
Thus, $t_A = H.t_{A1} + (1 - H).(t_B + t_{A1})$
- If t_{HIT} denotes the time required to check whether there is a hit, we can write
$$t_{MISS} = t_{HIT} + t_B + t_{A1}$$

21

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

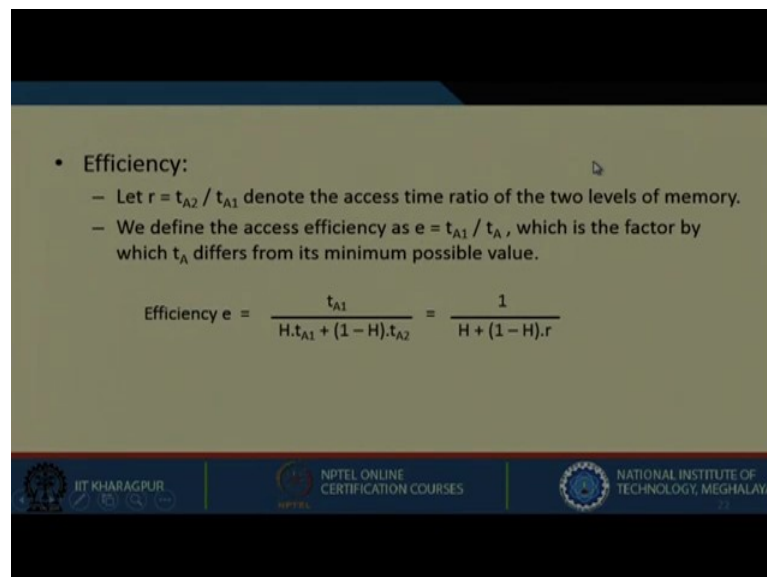
The miss penalty t_{miss} can be estimated in various ways. The simplest approach is to set t_{miss} as t_{A2} , that is, when there is a miss the data is accessed directly from M_2 . So, a request for a word not in M_1 typically causes a block containing the requested word to

be transferred from M2 to M1. After completion of the block transfer the word can be accessed from M1.

When accessing a particular word, generally we do not transfer a single word rather we transfer a block of word. So, the block containing that particular word should be transferred to the cache, and then from the cache it will be transferred to the processor. So, this is what is said a request for the word not in M1 typically causes a block containing the requested word to be transferred from M2 to M1.

So, first the block is transferred from M2 to M1, and after completion of the block transfer the word can be accessed in M1. t_B denotes the block transfer time. The expression is shown.

(Refer Slide Time: 36:53)



• Efficiency:

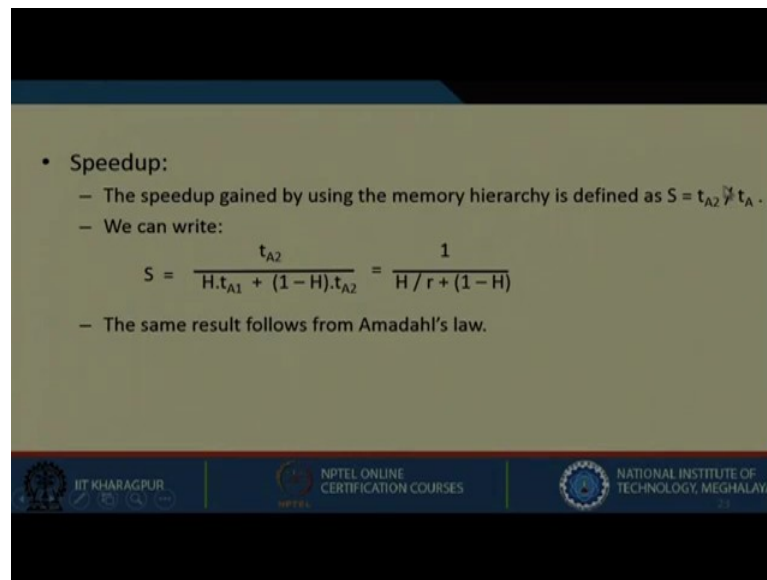
- Let $r = t_{A2} / t_{A1}$ denote the access time ratio of the two levels of memory.
- We define the access efficiency as $e = t_{A1} / t_A$, which is the factor by which t_A differs from its minimum possible value.

$$\text{Efficiency } e = \frac{t_{A1}}{H \cdot t_{A1} + (1 - H) \cdot t_{A2}} = \frac{1}{H + (1 - H) \cdot r}$$

The slide footer contains three logos: IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA.

Now, what is efficiency? Let us consider r as the access time ratio of the 2 levels. Efficiency is defined as shown in the expression.

(Refer Slide Time: 38:05)

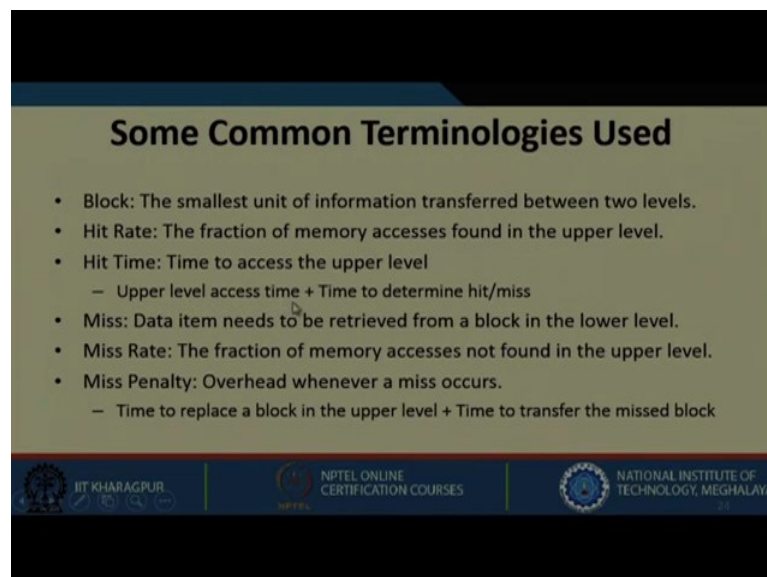


- **Speedup:**
 - The speedup gained by using the memory hierarchy is defined as $S = t_{A2} / t_A$.
 - We can write:
$$S = \frac{t_{A2}}{H \cdot t_{A1} + (1-H) \cdot t_{A2}} = \frac{1}{H/r + (1-H)}$$
 - The same result follows from Amadahl's law.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, coming to speedup, the speedup gained by using memory hierarchy is time-old divided by time-new.

(Refer Slide Time: 39:28)



Some Common Terminologies Used

- **Block:** The smallest unit of information transferred between two levels.
- **Hit Rate:** The fraction of memory accesses found in the upper level.
- **Hit Time:** Time to access the upper level
 - Upper level access time + Time to determine hit/miss
- **Miss:** Data item needs to be retrieved from a block in the lower level.
- **Miss Rate:** The fraction of memory accesses not found in the upper level.
- **Miss Penalty:** Overhead whenever a miss occurs.
 - Time to replace a block in the upper level + Time to transfer the missed block

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

So, there are some common terminologies that we must know for rest of the lectures for this week. What is block? --- the smallest unit of information transferred between 2 levels. Hit Rate --- the fraction of memory accesses found in upper level. Hit Time --- the time to access the upper level. And so on.

So, these are some of the terminologies that we will be using throughout the week 6 lectures.

Thank you.