

**Real Time Systems**  
**Professor Durga Prasad Mohapatra**  
**Department of Computer Science and Engineering**  
**National Institute of Technology Rourkela**  
**Lecture 46**  
**Benchmarking Real - Time Systems**

Good afternoon to all of you. So, today we will discuss about how to benchmark real time computer systems.

(Refer Slide Time: 00:26)

**CONCEPTS COVERED**

- Rheapstone Metric
- Interrupt Processing Overhead
- Tridimensional Measure
- Deterministic Benchmarks
- Latency Benchmarks

The slide features a dark blue header with the title 'CONCEPTS COVERED' in white. Below the title is a list of five items, each preceded by a right-pointing arrowhead. The background is white with a decorative blue and green geometric shape on the right side. At the bottom, there are logos for NIT Rourkela and NPTEL.

So, here you will see these some metrics such as Rheapstone metric then we will discuss about interrupt processing overhead. So, another metric called as tridimensional measure, then other categories of metrics which are deterministic benchmarks and latency benchmarks.

(Refer Slide Time: 00:44)

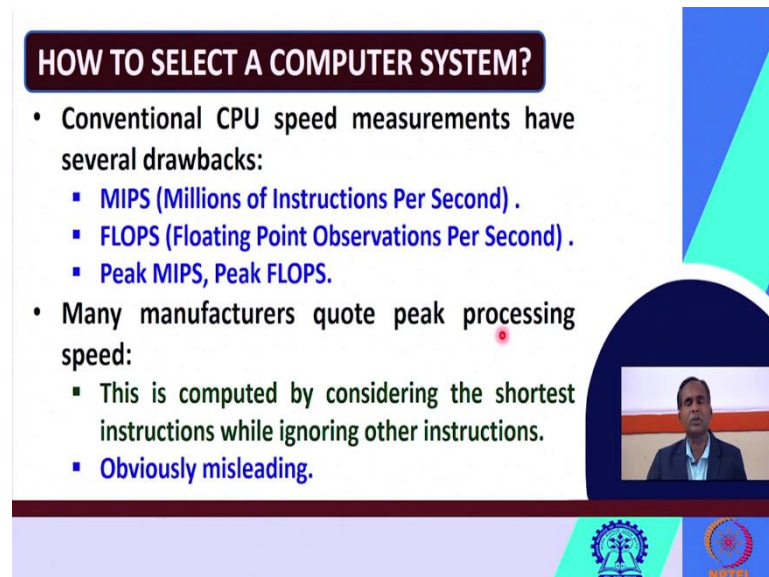
**KEYWORDS**

- SPEC (Standard Performance Evaluation Corporation)
- MIPS (Millions of Instructions Per Second )
- Interrupt Loading
- Timer Jitter
- Bintime

The slide features a dark blue header with the title 'KEYWORDS' in white. Below the title is a list of five items, each preceded by a right-pointing arrowhead. The background is white with a decorative blue and green geometric shape on the right side. At the bottom, there are logos for NIT Rourkela and NPTEL, and a small video inset showing the professor.

These key words we will discuss here SPEC, MIPS, interrupt loading, timer jitter, bintime, etcetera.

(Refer Slide Time: 00:51)



**HOW TO SELECT A COMPUTER SYSTEM?**

- Conventional CPU speed measurements have several drawbacks:
  - MIPS (Millions of Instructions Per Second) .
  - FLOPS (Floating Point Observations Per Second) .
  - Peak MIPS, Peak FLOPS.
- Many manufacturers quote peak processing speed:
  - This is computed by considering the shortest instructions while ignoring other instructions.
  - Obviously misleading.

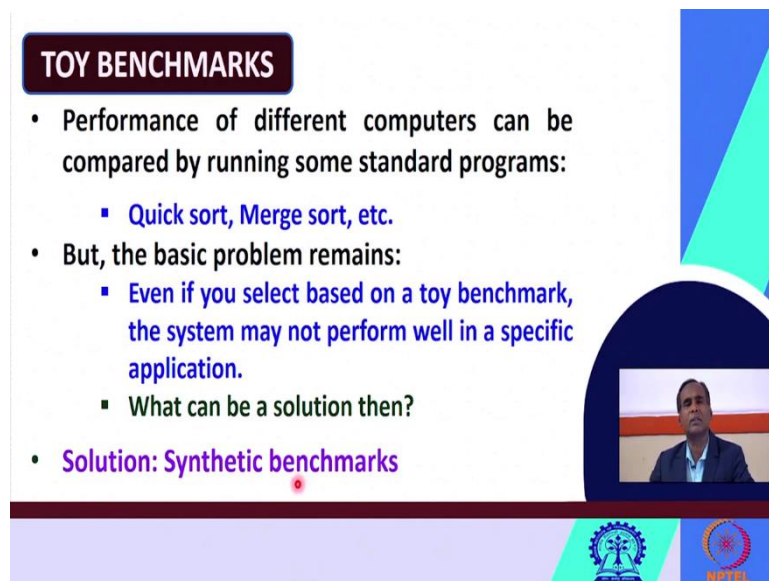
The slide features a video inset of a man speaking, a blue and green geometric design on the right, and logos for IIT Bombay and NPTEL at the bottom.

So, now, let us start with how to benchmark real time computer systems. So, before going to discuss how to benchmark the real time computer systems let us first look at how to select the ordinary computer systems. So, suppose you are the owner of a, or you are working for a company and that company has given you the assignment of selecting the best computer out of the available computers in the market then what you will do, you will take some benchmarks you will take some measures based on which you will select which computer will be best computer.

So, what measures or what metrics you can use, you can use the traditional conventional measures such as the CPU speed. So, we can use the conventional CPU speed measurements such as MIPS or FLOPS. So, MIPS stands for millions of instructions per second, FLOPS stands for floating point observations per second like this, but these traditional CPU speed measurements they have several drawbacks you can see those drawbacks yourself then what happened that since this measurement have some drawbacks, so then these vendors what they have done, they have quoted the like Peak MIPS, and the Peak Flops, etcetera.

So, what is this Peak MIPS that means they are taking the observations or the peak time. So, many manufacturers they have quoted what the peak processing speed, so in order to attract more customers, but you know these peak values are computed by considering the shortest instructions and while they have ignored the other instruction. So, obviously, these kinds of measures they will mislead you. So, obviously these kinds of measurements are misleading.

(Refer Slide Time: 02:30)



**TOY BENCHMARKS**

- Performance of different computers can be compared by running some standard programs:
  - Quick sort, Merge sort, etc.
- But, the basic problem remains:
  - Even if you select based on a toy benchmark, the system may not perform well in a specific application.
  - What can be a solution then?
- **Solution: Synthetic benchmarks**

The slide features a video inset of a man speaking, a decorative blue and green geometric shape on the right, and logos for IIT Bombay and NPTEL at the bottom.

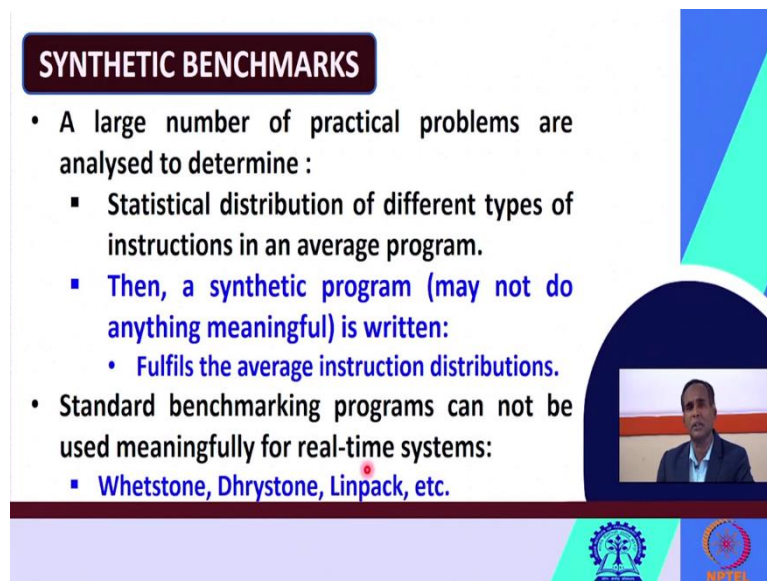
So, then next what happened? So, then many toy benchmarks have occurred. So, here the performance of different computers can be compared by running some standard programs. So, then what these vendors what they have done?

They have given the performance of different computers and they have compared them by running some standard programs, so these are called as toy benchmarks. So, toy benchmarks, they provide the performance of different computers which can be compared by running some standard programs such as quick sort, merge sort, bubble sort, etcetera.

So, but the basic problem still remains, if you select the best computer based on a toy benchmark, then what will happen, this system may not perform well in a particular application in this specific application, then what is the solution, what you should do?

So you should go for you should look at the synthetic benchmarks. So now today in this class, we will see some of the synthetic benchmarks. So, those benchmarks or those measures, those matrix you can select, you can use to select to rank to compare the different computer systems available in the market.

(Refer Slide Time: 03:39)



**SYNTHETIC BENCHMARKS**

- A large number of practical problems are analysed to determine :
  - Statistical distribution of different types of instructions in an average program.
  - Then, a synthetic program (may not do anything meaningful) is written:
    - Fulfils the average instruction distributions.
- Standard benchmarking programs can not be used meaningfully for real-time systems:
  - Whetstone, Dhrystone, Linpack, etc.

The slide features a video inset of a man speaking, and logos for IIT Bombay and NPTEL at the bottom.

So synthetic, let us see the difference or the details of the synthetic benchmarks. So, these synthetic benchmarks, how they are prepared? A large number of practical problems they are analyzed to determine the statistical distribution of different types of instructions in an average program. So, how this synthetic benchmark is prepared. So, here are a large number of practical problems they are analyzed, why they are analyzed? To determine the statistical distribution of different types of instructions in an average program.

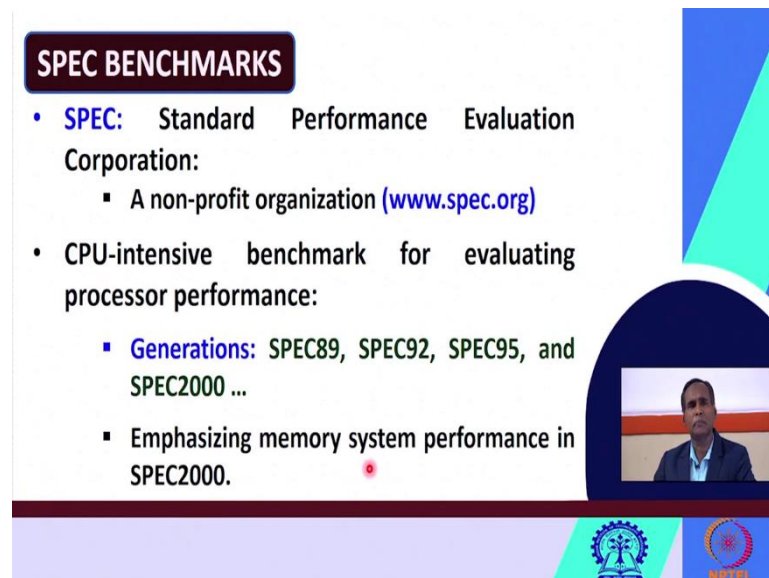
The different types of instructions could be suppose the arithmetic operations, arithmetic instructions are 20 percent. These IO operations say 10 percent, register operations say 30 percent like this. So, in this way, so a large number of practical problems are analyzed to determine the statistical distribution of different types of instructions in an average program.

Then what is being done using this obtained information, then a synthetic program is written, this synthetic program may or may not do any meaningful job, but the synthetic program is written why, this synthetic program is written which fulfils the average instruction or instruction distribution, so this synthetic program which is written based on the obtained information, this fulfils the average instruction distributions.

So, synthetic programmer, it is written, that is why this the name of the benchmark, is synthetic benchmark. So, there are several standard benchmark programs. So, some of the standard benchmark programs are Whetstone, Dhrystone, linpack, etcetera, but the standard benchmark programs they cannot be used meaningfully successfully for comparing or for measuring the performance of real time systems.

So, we may discuss some of the advanced, measures advanced benchmarks for comparing or for measuring the performance of real time systems. So, this synthetic benchmark, let us see, actually it was, there is a what you can say organization, there is what prepared these synthetic benchmarks.

(Refer Slide Time: 05:45)



**SPEC BENCHMARKS**

- **SPEC:** Standard Performance Evaluation Corporation:
  - A non-profit organization ([www.spec.org](http://www.spec.org))
- CPU-intensive benchmark for evaluating processor performance:
  - **Generations:** SPEC89, SPEC92, SPEC95, and SPEC2000 ...
  - Emphasizing memory system performance in SPEC2000.

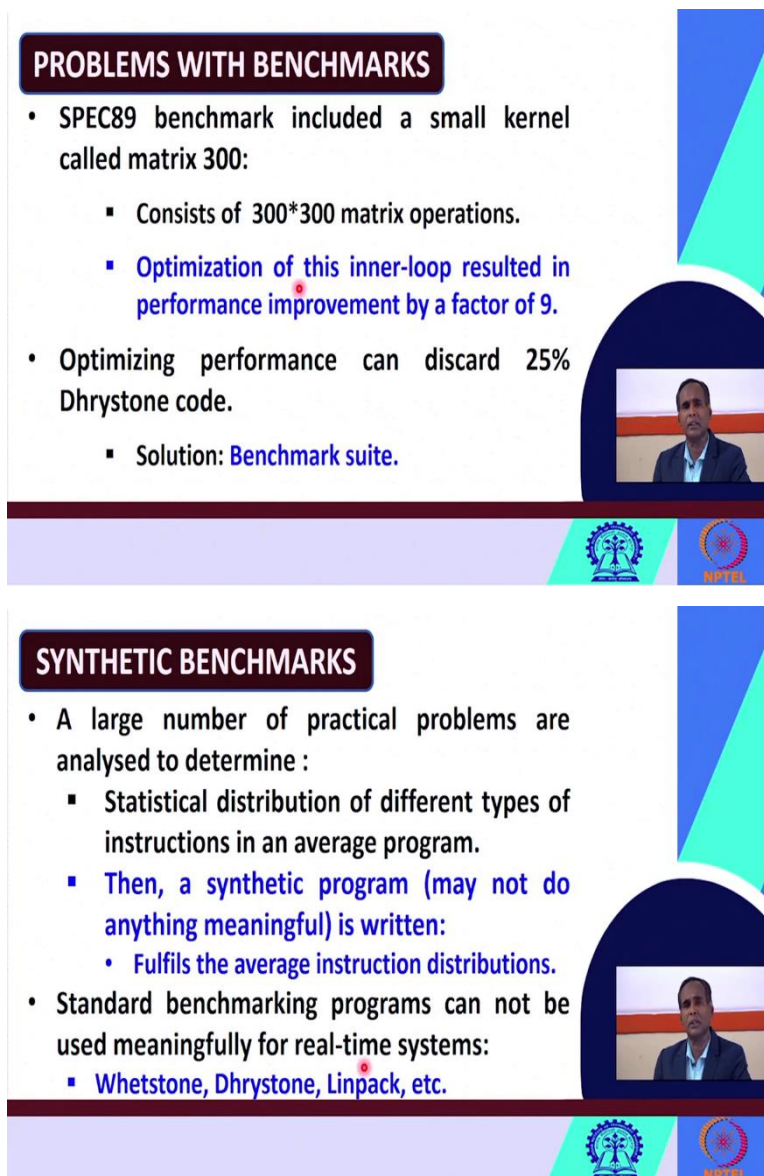
The slide features a video inset of a man in a suit and tie, and logos for IIT Bombay and NPTEL at the bottom.

One such organization in SPEC. So, SPEC stands for Standard Performance Evaluation Corporation, this is a non-profit making organization. So, normally they are associated with these synthetic benchmarks. So, you can see the different standards given by the SPEC, or the different SPEC benchmarks marks from this website [www.spec.org](http://www.spec.org). So, this spec ID is involved in a CPU intensive benchmark for evaluating the processor performance.

So, SPEC is involved, this is a non-profit making organization, it is involved in CPU intensive benchmark for evaluating the processor performance. That there are you can see the history like this the generations of SPEC like this, so SPEC it was I think started in SPEC89, then SPEC92, then 95, 2000 then onwards 2006, 2008, 2010, etcetera.

So, if you will see this 2000 version onwards. So, in SPEC2000 they have emphasized the memory system performance, how to measure the performance of the memory system. So, in SPEC2000, they have emphasized the memory system performance.

(Refer Slide Time: 06:53)



The image shows two presentation slides. The top slide is titled "PROBLEMS WITH BENCHMARKS" and contains a bulleted list. The bottom slide is titled "SYNTHETIC BENCHMARKS" and also contains a bulleted list. Both slides feature a small video inset of a man speaking in the bottom right corner. At the bottom of each slide, there are logos for IIT Bombay and NPTEL.

### PROBLEMS WITH BENCHMARKS

- SPEC89 benchmark included a small kernel called matrix 300:
  - Consists of 300\*300 matrix operations.
  - Optimization of this inner-loop resulted in performance improvement by a factor of 9.
- Optimizing performance can discard 25% Dhrystone code.
  - Solution: Benchmark suite.

### SYNTHETIC BENCHMARKS

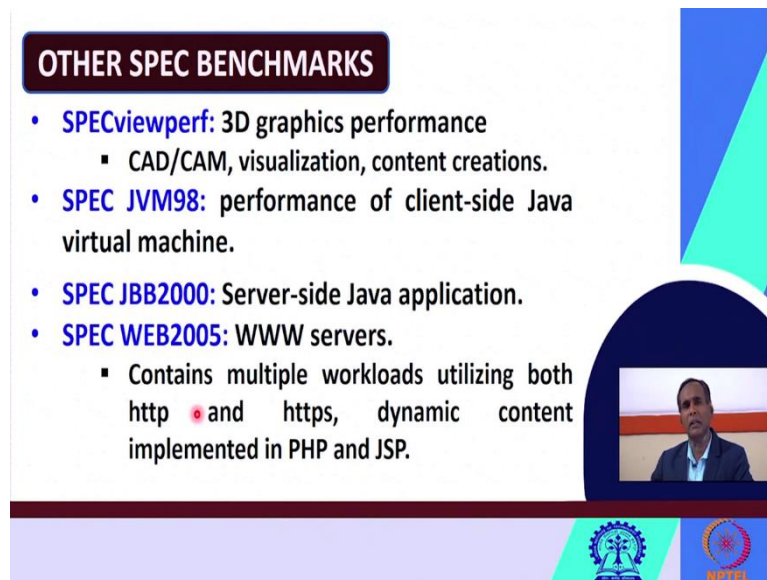
- A large number of practical problems are analysed to determine :
  - Statistical distribution of different types of instructions in an average program.
  - Then, a synthetic program (may not do anything meaningful) is written:
    - Fulfils the average instruction distributions.
- Standard benchmarking programs can not be used meaningfully for real-time systems:
  - Whetstone, Dhrystone, Linpack, etc.

Now, let us see what are the problems with the benchmarks. So, SPEC89 benchmark it has include a small kernel called matrix 300 and which consisted of 300 into 300 matrix operations, the optimization of this inner loop resulted in a performance improvement by a factor of 9. So, optimization of this inner loop it resulted in performance improvement almost by a factor of 9. So, optimizing performance can discard but the problem we are discussing out the problem with the benchmarks.

So, while this optimization is done, I have already told you this benchmark consists of a small kernel called as matrix 300, it consists of 300x300 matrix operations. And the optimization of this inner loop it has resulted in a performance improvement almost by a factor of 9.

But while the optimization it is being done, this optimizing performance, it can discard 25 percent of the dhrystone code, I have already told different examples like whetstone, dhrystone, linpack. So, while optimizing the performance, it can discard 25 percent of the dhrystone code. So, let us see the solution, we will go for the benchmark suite.

(Refer Slide Time: 08:06)



**OTHER SPEC BENCHMARKS**

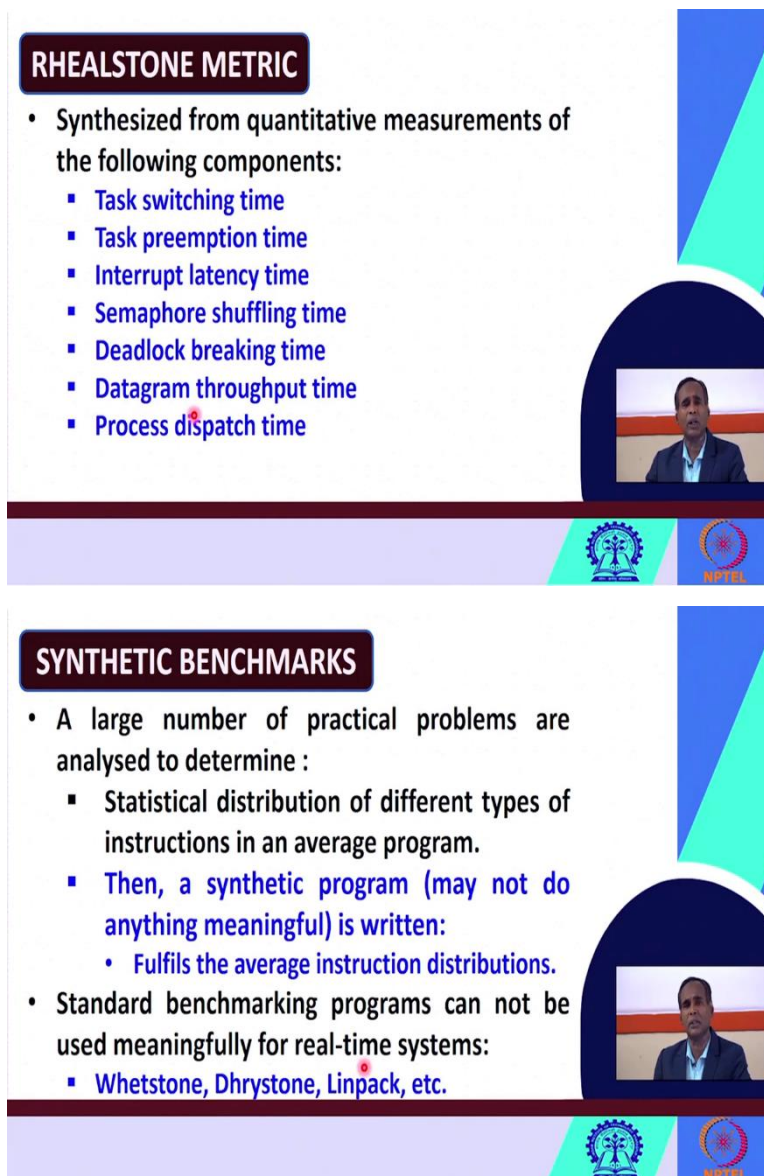
- **SPECviewperf:** 3D graphics performance
  - CAD/CAM, visualization, content creations.
- **SPEC JVM98:** performance of client-side Java virtual machine.
- **SPEC JBB2000:** Server-side Java application.
- **SPEC WEB2005:** WWW servers.
  - Contains multiple workloads utilizing both http and https, dynamic content implemented in PHP and JSP.

The slide features a dark blue header with the title 'OTHER SPEC BENCHMARKS' in white. The content is a bulleted list with blue and black text. A small video inset of a man is visible on the right side of the slide. The bottom of the slide has a light blue footer with logos for IIT Bombay and NPTEL.

There are other spec benchmarks such as SPECveiwperf, normally it deals with the 3D graphics performance, CAD CAM visualization, content creation, etcetera. SPEC JVM98, it can be used for measuring the performance of client side Java virtual machines. SPEC JBB2000, it can be used to measure the performance of server side Java applications. SPEC WEB2005, it deals with the WWW servers.

It contains multiple workloads, utilizing both HTTP as well as HTTPS, dynamic content implemented in PHP and JSP. So, these are some of the what other SPEC benchmarks are available. Even some more SPEC benchmarks are available, you can look at that website for the details.

(Refer Slide Time: 08:52)



The image shows two presentation slides. The top slide is titled 'RHEALSTONE METRIC' and lists seven components: Task switching time, Task preemption time, Interrupt latency time, Semaphore shuffling time, Deadlock breaking time, Datagram throughput time, and Process dispatch time. The bottom slide is titled 'SYNTHETIC BENCHMARKS' and discusses the analysis of practical problems to determine instruction distributions and the limitations of standard benchmarking programs for real-time systems. Both slides feature a video inset of a speaker and logos for IIT Bombay and NPTEL.

### RHEALSTONE METRIC

- Synthesized from quantitative measurements of the following components:
  - Task switching time
  - Task preemption time
  - Interrupt latency time
  - Semaphore shuffling time
  - Deadlock breaking time
  - Datagram throughput time
  - Process dispatch time

### SYNTHETIC BENCHMARKS

- A large number of practical problems are analysed to determine :
  - Statistical distribution of different types of instructions in an average program.
  - Then, a synthetic program (may not do anything meaningful) is written:
    - Fulfils the average instruction distributions.
- Standard benchmarking programs can not be used meaningfully for real-time systems:
  - Whetstone, Dhrystone, Linpack, etc.

Now, we will come to the other metric, the most popularly used metric I have already told the examples like whetstone, dhrystone, linpack, etcetera. Today we will discuss one of the most popular benchmarks is the rhealstone metric. So, this rhealstone metric, this is synthesized from quantitative measurement of the following components.

So, the components are task switching time, task preemption time, interrupt latency time, semaphore shuffling time, Deadlock breaking time, datagram throughput time, process dispatch time. So, out of this many of the definitions you have already known in the earlier classes, we will just quickly look at again, we will summarize these times, these metrics.



(Refer Slide Time: 08:52)

**TASK SWITCHING TIME**

- Average time that the system takes to switch between two independent tasks.
- Determined by the efficiency of kernel data structure.

The slide includes a Gantt chart illustrating task execution. It shows three tasks: T1 (blue), T2 (pink), and T1 (blue). Vertical dashed lines indicate the start and end of each task. The time intervals between these lines represent the switching time. A red dot on the second T1 bar indicates a specific point in time. The slide also features a video inset of a speaker and logos for IIT Bombay and NPTEL at the bottom.

Let us start to task switching time, what do you mean by task switching time. So, this task switching time it is defined as the average time that the system takes to switch between two independent tasks. So, if there are two independent tasks, the average time that the system will take to switch from one task to another task, we call it a switching time. How it can be determined?

The switching time determined by the efficiency of kernel data structure. The switching time is determined by the efficiency of kernel data structure. Let us take an example like this, suppose there are three tasks here T1, T2, as I have already told you, it is defined as the system which takes to switch between two independent tasks. So, T1 is a task, T2 is independent task.

So, how much average time, how much time the system will take to switch from T1 to T2 or from T2 to T1 like that. So, you take the average value, this will give you the task switching time. So, this period from T1 to T2, this is, similarly from T2 to T1, when it is switching the system is switching from either from T1 to T2 or from T2 to T1. How much time it is taking? You find out the average value, this average time is known as the task switching time.

(Refer Slide Time: 10:48)

**TASK PREEMPTION TIME**

- Average time it takes to transfer control from a low priority task to a high priority task.
- Consists of 3 components:
  - Task switching time,
  - Time to recognize the event causing a higher priority task to become ready, and
  - Time to dispatch it.
- It is normally larger than task switching time.

**TASK SWITCHING TIME**

- Average time that the system takes to switch between two independent tasks.
- Determined by the efficiency of kernel data structure.

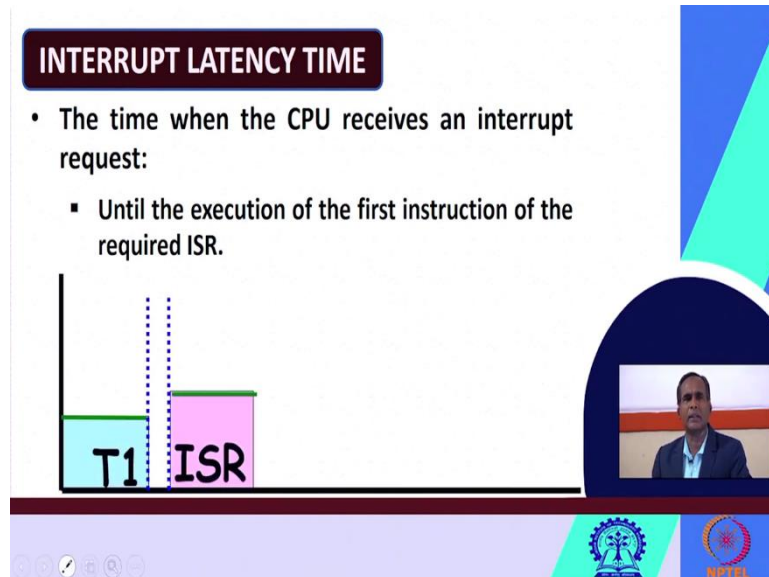
The diagram shows a horizontal timeline with three colored bars representing task execution: a blue bar labeled 'T1', a red bar labeled 'T2', and another blue bar labeled 'T1'. Vertical dashed lines indicate the start and end of each task. A red dot is placed on the timeline between the end of the first T1 bar and the start of the T2 bar, representing the task switching time.

Then you will see the next time, next metric that is the task preemption time, this you have already what is preemption, what the task preemption you have already known in the earlier classes. So, this metric is defined as the average time that it takes to transfer the control from where, from one low priority task to on the high priority task.

So, the task preemption time is defined as the average time that it takes to transfer the control from, from a low priority task to a high priority tasks. So, this task preemption time, it consists of three important items, one is the task switching time, then the time to recognize the event what, causing or enabling a higher priority task to become ready and what is the time to dispatch it.

So, this time will task preemption time normally should be larger than the task switching time. What is task switching time? Already you have seen here. So now, this preemption time, three components out of that 1 is task switching time, so obviously so this task preemption time, it will be larger than the task switching time.

(Refer Slide Time: 11:51)

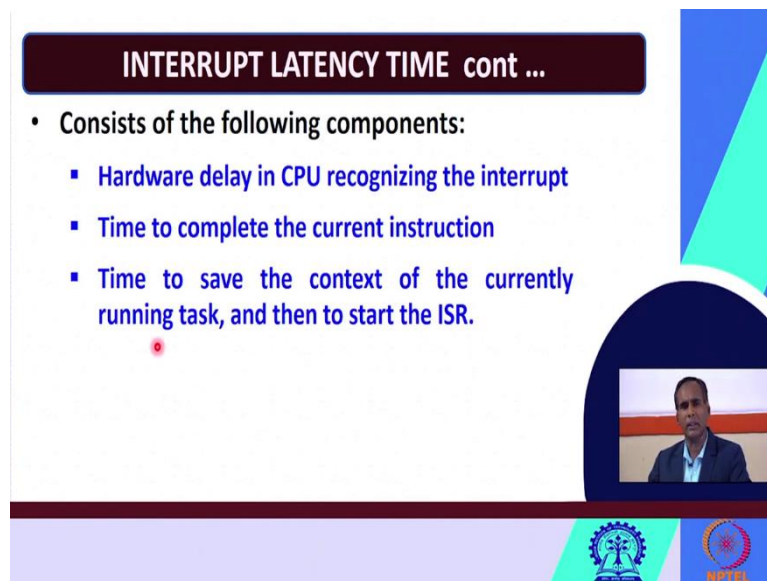


Now, let us see about this interrupt latency, so you have seen task preemption time let us see the interrupt latency time. So, the interrupt latency time is defined as the time when the CPU receives an interrupt request until the execution of the first instruction of the required ISR. I am repeating again. So, interrupt latency time is defined as the time when the CPU it receives an interrupt request and the execution of the first instruction of the required ISR.

So, when the CPU receives an interrupt request, from that until the execution of the first instruction of the required ISR, interrupt service routine, that time is called as what interrupt latency time. So, this is defined as the time when the CPU receives an interrupt request, till what until the execution of the first instruction of the desired ISR.

If you will see, in figure looks like this. So, suppose T1 is a task here. Now, when CPU receives an interrupt request, then you will see till the point when the execution of the first instruction of the required ISR, we call it interrupt latency time. So now, suppose the T1, at the end of this T1 CPU receives an interrupt request and if this is T1 and this line is T2, at T2 ISR starts what execution, the first instruction of ISR start execution then this difference is known as interrupt latency time.

(Refer Slide Time: 13:15)



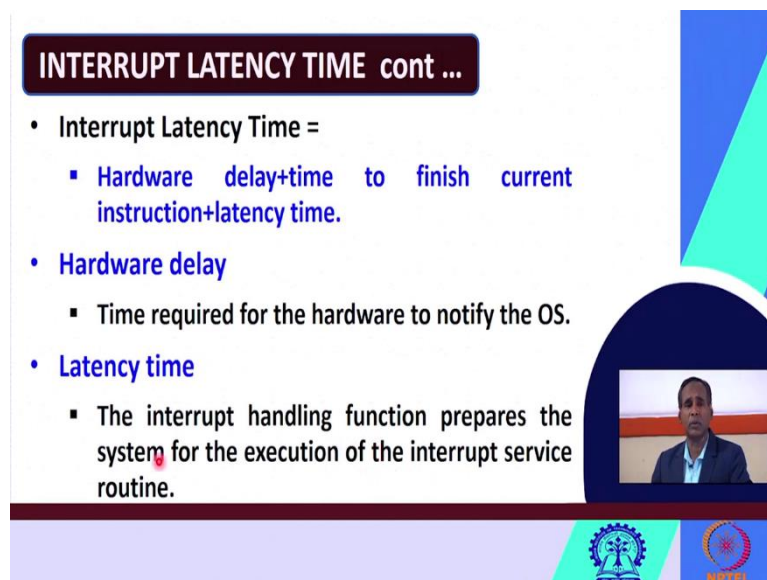
**INTERRUPT LATENCY TIME cont ...**

- Consists of the following components:
  - Hardware delay in CPU recognizing the interrupt
  - Time to complete the current instruction
  - Time to save the context of the currently running task, and then to start the ISR.

The slide features a dark blue header with the title in white. The main content is on a white background with blue text. A small video inset of a man in a suit is in the bottom right. Logos for IIT Bombay and NPTEL are at the bottom.

So, interrupt latency time consists of the following components, the hardware delay in CPU recognizing the interrupt, then the time to complete the current instruction, then the time to save the context of the current running tasks and then to start the corresponding ISR, interrupt service routine.

(Refer Slide Time: 13:35)



**INTERRUPT LATENCY TIME cont ...**

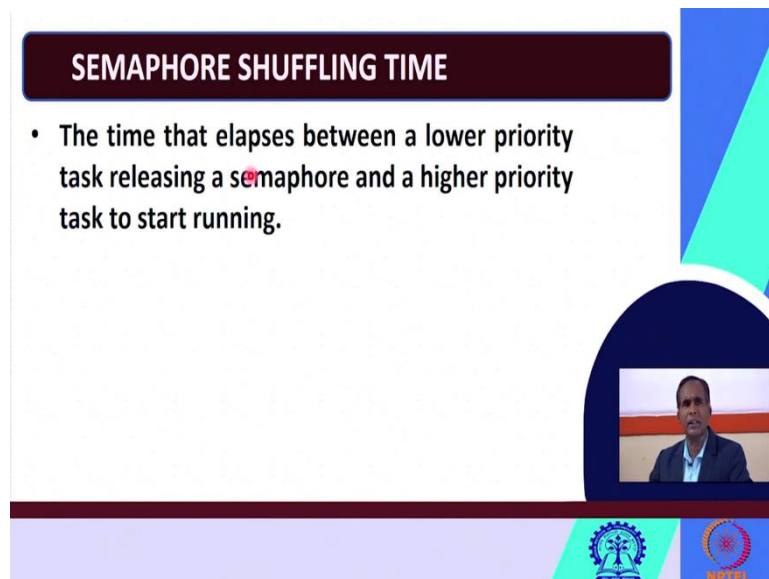
- Interrupt Latency Time =
  - Hardware delay+time to finish current instruction+latency time.
- Hardware delay
  - Time required for the hardware to notify the OS.
- Latency time
  - The interrupt handling function prepares the system for the execution of the interrupt service routine.

The slide features a dark blue header with the title in white. The main content is on a white background with blue text. A small video inset of a man in a suit is in the bottom right. Logos for IIT Bombay and NPTEL are at the bottom.

So, I can summarize as follows. Interrupt latency time can be defined as hardware delay plus time to finish the current instruction plus the latency time, what do you mean by hardware delay, this is the time required for the hardware to notify the OS. So, what, how much time will be required for the hardware to notify to the operating system that we call are the hardware delay.

Latency time, this latency time what it does, the interrupt handling function, it prepares the system for the execution of the interrupt service routine. So, latency time is associated with the following. The interrupt handling function, it prepares the whole system for the execution of the interrupt service routine, so how much time it takes. So, these three times you can add hardware delay plus time to finish the current instruction plus the latency time this will give you the interrupt latency time.

(Refer Slide Time: 14:23)



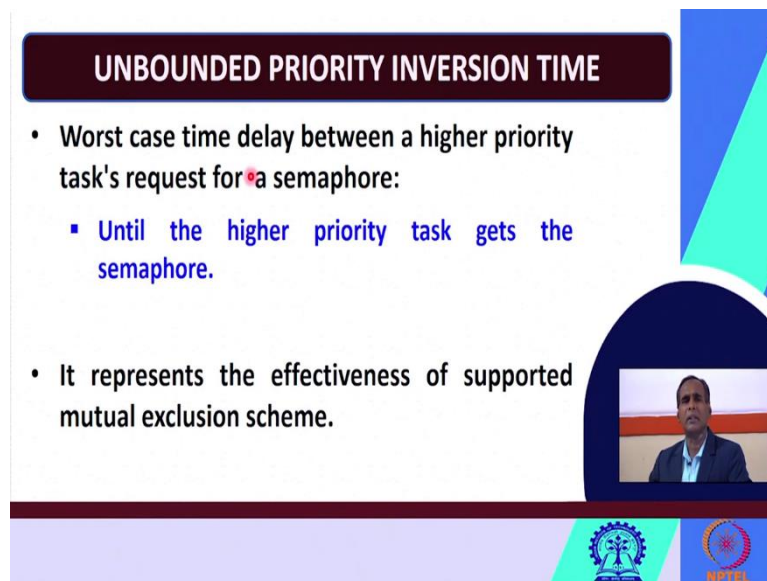
**SEMAPHORE SHUFFLING TIME**

- The time that elapses between a lower priority task releasing a semaphore and a higher priority task to start running.

The slide features a dark blue header with the title in white. Below the title is a bulleted definition. A video inset in the bottom right shows a man in a suit. The slide is decorated with geometric shapes in blue, green, and red, and logos for IIT Bombay and NPTEL at the bottom.

Then the next metric is semaphore shuffling time. So, it is defined as the time which elapses between the lower priority task, releasing a semaphore and then when a high priority task, it starts running, is not it? So, suppose the low priority task T1 is there, it is holding the resource or the semaphore, then a high priority task T2 comes, so now this time gap when these because high priority task now it will wait, because T1 is holding the resource. So, now the time from which the T1 task releases the semaphore and the high priority task T2 it acquires that it starts running that time difference is known as semaphore shuffling time.

(Refer Slide Time: 15:04)



**UNBOUNDED PRIORITY INVERSION TIME**

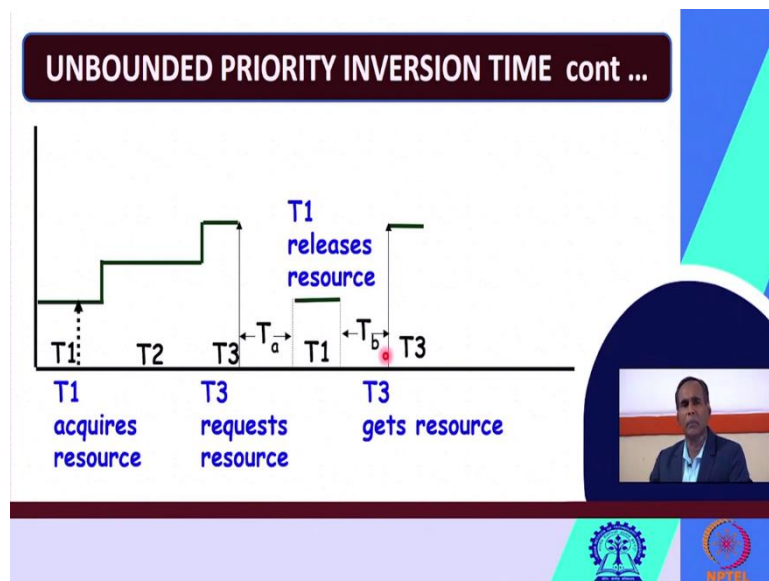
- Worst case time delay between a higher priority task's request for a semaphore:
  - Until the higher priority task gets the semaphore.
- It represents the effectiveness of supported mutual exclusion scheme.

The slide features a dark blue header with the title in white. The main content is on a white background with two bullet points. A small video inset in the bottom right shows a man speaking. The footer contains logos for IIT Bombay and NPTEL.

Next metric is unbounded priority inversion time. So, unbounded priority inversion you have already known earlier. So, unbounded priority inversion time it is defined as the worst-case time delay between a high priority tasks request for a semaphore until the high priority task gets the semaphore. I am repeating again.

So, this unbounded priority inversion time, it is defined at the worst-case time delay between what, between the time when a high priority tasks request for the semaphore until that high priority task gets the semaphore. This is defined as unbounded priority inversion time. It represents the effectiveness of the supported mutual exclusion scheme. So, which mutual exclusion scheme is supported by your system, it represents the effectiveness of that supported mutual exclusion scheme?

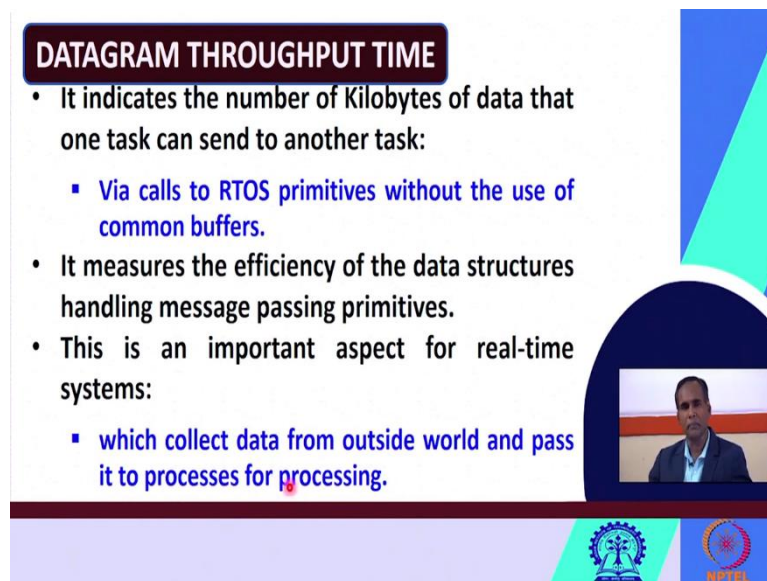
(Refer Slide Time: 15:50)



Let us look at pictorially. So, suppose here task T1 is there, you can see this low priority task, it acquires a resource. T2 again is high priority task, T3 still a high priority task. So now, this T3 is waiting because T1 is holding the resource. Now, after the so, how much time it is required, so T3 waits now it is requested for the resource. So, as soon as this request which is receives at the system or this what the scheduler what will happen then T1 will try to complete its job then it will release.

So, suppose at this point it releases, the T1 releases the resource, then after releasing the resource, then the resource will be assigned to, it will be assigned to T3 because T3 has already made a request. So, now at the point when T3 gets the resource, so this difference will take. So, now two difference you see, when T3 request for a resource and when this time interval plus when T1 releases the resource, and the T3 acquires the resource, so this time interval you say  $T_b$ . So, now this unbounded priority inversion time is equal to  $T_a + T_b$ . So, this is the definition unbounded priority inversion time.

(Refer Slide Time: 17:03)



**DATAGRAM THROUGHPUT TIME**

- It indicates the number of Kilobytes of data that one task can send to another task:
  - Via calls to RTOS primitives without the use of common buffers.
- It measures the efficiency of the data structures handling message passing primitives.
- This is an important aspect for real-time systems:
  - which collect data from outside world and pass it to processes for processing.

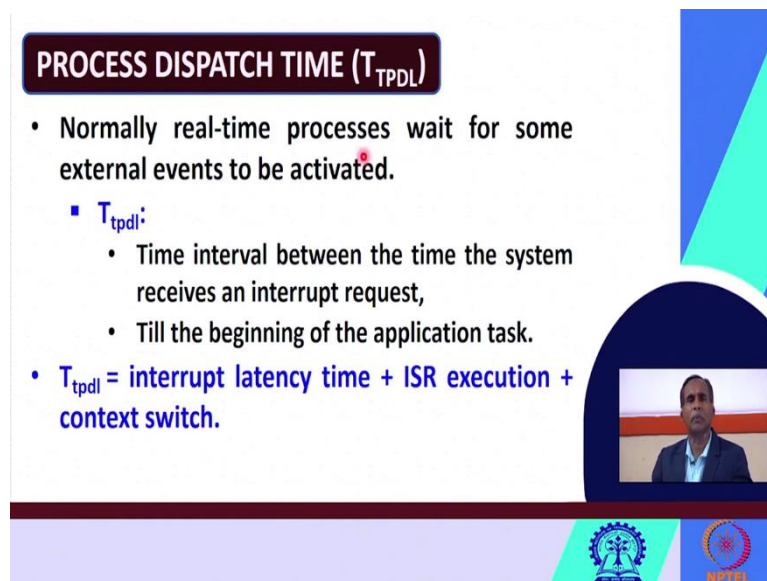
The slide features a dark blue header with the title in white. The main content is on a white background with blue and red accents. A small video inset shows a man in a suit. At the bottom, there are logos for IIT Bombay and NPTEL.

Next metric is datagram throughput time. So, it indicates what, this datagram throughput time it indicates the number of kilobytes of data that one task can send to another task. So, this throughput time it indicates what, the number of the kilobytes of data which one task can send to another task via what, via the calls to the real time operating system primitives without using the common buffers, might be some pointers or something else.

So, this measure metrics these, this metric measures the efficiency of the data structures on which are handling the message passing primitives. So, this metric data gram throughput time is a very important aspect for real time systems, which collect data from outside world and pass those data to the different processes for processing. So, this is very much, this metric is very much important for real time systems.



(Refer Slide Time: 18:02)



**PROCESS DISPATCH TIME ( $T_{TPDL}$ )**

- Normally real-time processes wait for some external events to be activated.
- $T_{tpdl}$ :
  - Time interval between the time the system receives an interrupt request,
  - Till the beginning of the application task.
- $T_{tpdl} = \text{interrupt latency time} + \text{ISR execution} + \text{context switch}.$

The slide features a video inset of a man in a suit and two logos at the bottom: the Indian Institute of Technology (IIT) logo and the NPTEL logo.

Then we will see another metric called as process dispatch time. So normally, the real time processes they wait for some external events to be activated, is not it? So, normally the real time processes they wait for some external events to be activated or triggered, we may use the symbol tpdL. So, your time for process dispatch latency time, this L is latency time maybe. So, this tpdL is defined as the time interval between the time the system receives an interrupt request and the beginning of the application task.

So, the process dispatch latency time, it is defined at the time interval between the time when the system receives an interrupt request and the beginning of the application tasks. So, this process response time can be written in the following mathematical formula, this is equal to the interrupt latency time plus this ISR execution interrupt service routine execution how much time it takes plus the context which how much time it takes for the context switch.

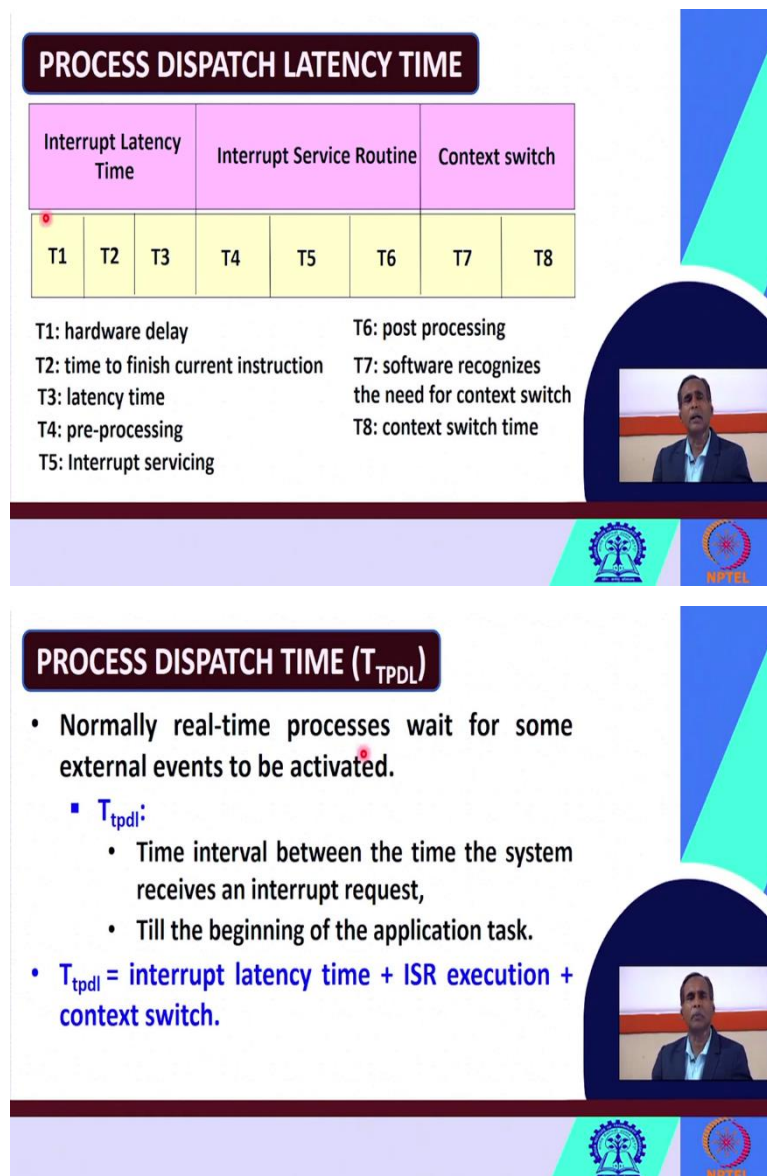
So, this is how you can compute this process dispatch time. Out of this, how to calculate the interrupt latency time, I have already shown you. ISR execution you know and context switching time also you know.

(Refer Slide Time: 19:16)

### PROCESS DISPATCH LATENCY TIME

Interrupt Latency Time			Interrupt Service Routine			Context switch	
T1	T2	T3	T4	T5	T6	T7	T8

T1: hardware delay  
T2: time to finish current instruction  
T3: latency time  
T4: pre-processing  
T5: Interrupt servicing  
T6: post processing  
T7: software recognizes the need for context switch  
T8: context switch time



The slide features a title 'PROCESS DISPATCH LATENCY TIME' in a dark blue box. Below it is a table with three columns: 'Interrupt Latency Time', 'Interrupt Service Routine', and 'Context switch'. The first column contains T1, T2, and T3; the second contains T4, T5, and T6; the third contains T7 and T8. Below the table, definitions for each T are provided. The slide also includes a small video inset of a man in a suit, the IIT Bombay logo, and the NPTEL logo.

### PROCESS DISPATCH TIME ( $T_{TPDL}$ )

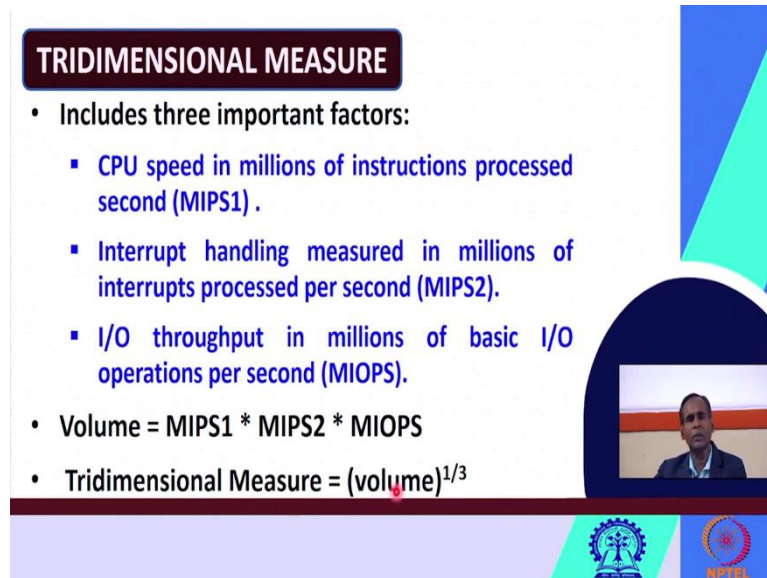
- Normally real-time processes wait for some external events to be activated.
  - $T_{tpdl}$ :
    - Time interval between the time the system receives an interrupt request,
    - Till the beginning of the application task.
- $T_{tpdl} = \text{interrupt latency time} + \text{ISR execution} + \text{context switch}$ .

So, let us see in pictorially manner. Process dispatch latency time can be considered the sum of these three things interrupt latency time, ISR execution, and context switch. So, interrupt latency time consists of three things I have already told you, hardware delay, time to finish the current instruction, and latency time. Interrupt service routine consists of the what, the pre-processing required plus the time required for interrupt servicing plus the post processing the time required for post processing.

Context switch again consists of two important times, the T7 here it deals with the or might indicates the software recognizes that, the software recognizes the need for a context switch and the T8 is the actual context switch time. So, in this way, the process dispatch latency time

can be computed if you know the interrupt latency time, the interrupt service routine execution, and the context switch.

(Refer Slide Time: 20:08)



**TRIDIMENSIONAL MEASURE**

- Includes three important factors:
  - CPU speed in millions of instructions processed second (MIPS1).
  - Interrupt handling measured in millions of interrupts processed per second (MIPS2).
  - I/O throughput in millions of basic I/O operations per second (MIOPS).
- Volume = MIPS1 \* MIPS2 \* MIOPS
- Tridimensional Measure =  $(\text{volume})^{1/3}$

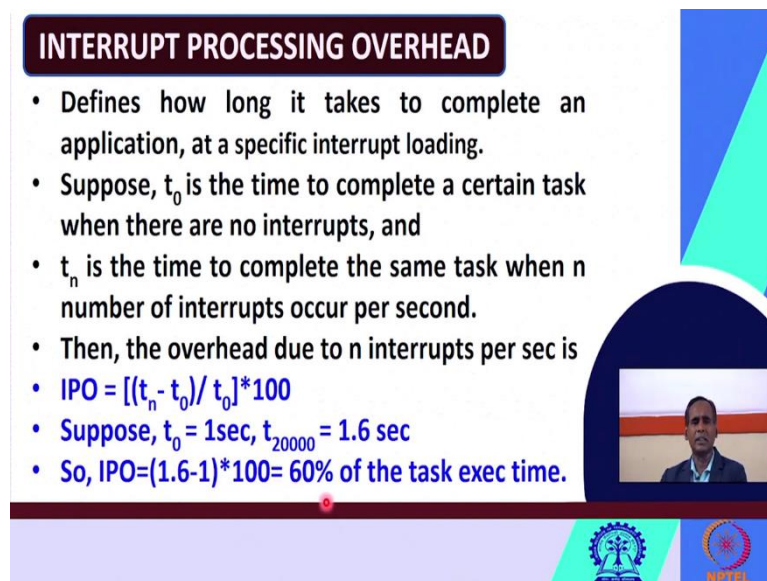
The slide features a blue and green geometric design on the right side, a small video inset of a speaker, and logos for IIT Bombay and NPTEL at the bottom.

Now, we will go to another category of metrics, this metric is known as tridimensional measure. So, this measure includes three important factors number one, the CPU speed in millions of instructions processed per second.

Similarly, then the second component is interrupt handling measured in millions of interrupts process per second, it is in millions of instructions processed per second, this is the millions of interrupts processed per second. Please mark the difference. And the IO throughput in millions of basic IO operations per second or MIOPS.

So, in tridimensional measure we first compute the volume, this volume is equal to MIPS1 \* MIPS2 into MIOPS, then you can find out the tridimensional measure by taking the cube root volume, so tridimensional measure is equal to cube root of volume. In this way, you can also compute the tridimensional measure, which is another metric for computing the performance of the real time systems.

(Refer Slide Time: 21:13)



**INTERRUPT PROCESSING OVERHEAD**

- Defines how long it takes to complete an application, at a specific interrupt loading.
- Suppose,  $t_0$  is the time to complete a certain task when there are no interrupts, and
- $t_n$  is the time to complete the same task when  $n$  number of interrupts occur per second.
- Then, the overhead due to  $n$  interrupts per sec is
- $IPO = [(t_n - t_0) / t_0] * 100$
- Suppose,  $t_0 = 1\text{sec}$ ,  $t_{20000} = 1.6\text{ sec}$
- So,  $IPO = (1.6 - 1) * 100 = 60\%$  of the task exec time.

The slide features a video inset of a man speaking, a blue and green geometric design on the right, and logos for IIT Bombay and NPTEL at the bottom.

We will see another metric called interrupted processing overhead. So, this overhead defines how long it takes to complete an application at a particular interrupt loading. Now, let us find out a small mathematical formula for that. Suppose,  $t_0$  is the time required to complete a certain task, a particular task when there are no interrupts, that is why I am measuring 0. So, no interrupts and  $t_n$  it is the time to complete the same tasks when there are  $n$  number of interrupts, when  $n$  number of interrupts occurs, occur per second this is  $t_n$ .

So, then the overhead due to  $n$  number of interrupts per second can become computed as IPO interrupt processing overhead is equal to  $(t_n - t_0 / t_0) * 100$ , normally it is specified in terms of percentage. So, what is  $t_n$ ? We have already known what is  $t_0$  we have already known suppose let us assume that the  $t_0$  is equal to 1 second and  $t_n$ . What is  $t_n$ ? Time to complete the same task when  $n$  number of interrupts occur per second suppose there are 20,000, suppose 20,000 interrupts occur per second. And let us assume that  $t$  of 20,000 equal to 1.6 second.

Then what is the interrupt processing overhead? So, interrupt processing overhead is equal to  $t_n$  minus  $t_0$  that means 1.6 minus 1 divided by, I missed, divided by how much it is  $t_0$ ,  $t_0$  is equal to how much, 1, so that is why I have omitted 1 here because divided by 1 same thing into 100, which is equal to coming to be almost 60 percent of the task execution time. So, in this way, you can compute the interrupt processing overhead. There are  $n$  number of interrupts occur per second.

(Refer Slide Time: 22:51)

**REAL-TIME OPERATING SYSTEM BENCHMARKS**

- Two categories of benchmarks are important:
  - Deterministic benchmarks.
  - Latency benchmarks.

The slide features a dark blue header with the title in white. The main content is on a white background with a blue and green geometric design on the right. A small video inset of a speaker is in the bottom right. Logos for IIT Bombay and NPTEL are at the bottom.

Now, we will go to the benchmarks for the real time systems. Now, let us say the real time operating system benchmarks. There are two important categories of benchmarks which are very much important for real time operating systems. One is deterministic benchmarks. Another is the latency benchmarks. Let us see about this deterministic benchmark.

(Refer Slide Time: 23:12)

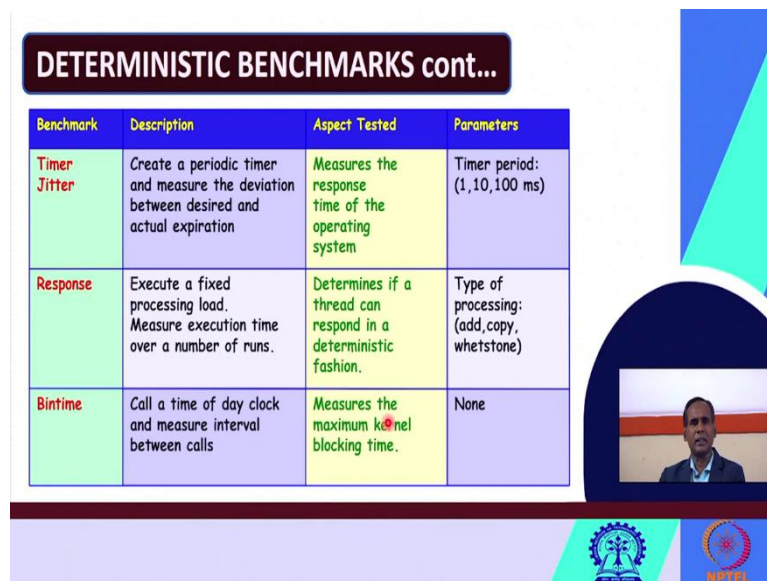
**DETERMINISTIC BENCHMARKS**

- Measure the determinism of an operating system services:
  - Timer Jitter,
  - Response, and
  - Bintime.

The slide features a dark blue header with the title in white. The main content is on a white background with a blue and green geometric design on the right. A small video inset of a speaker is in the bottom right. Logos for IIT Bombay and NPTEL are at the bottom.

So, this deterministic benchmark, what does it do? These benchmarks they measure the determinism of an operating system service. So, deterministic benchmarks they measure the determinism of an operating system service. And which benchmarks it includes? So, the following benchmarks are included under deterministic benchmarks, like timer jitter, response, and the bintime. Let us start with first, what do you mean by timer jitter.

(Refer Slide Time: 23:42)



The slide features a table with four columns: Benchmark, Description, Aspect Tested, and Parameters. The table contains three rows of data. To the right of the table is a video inset showing a man speaking. At the bottom of the slide are logos for IIT Bombay and NPTEL.

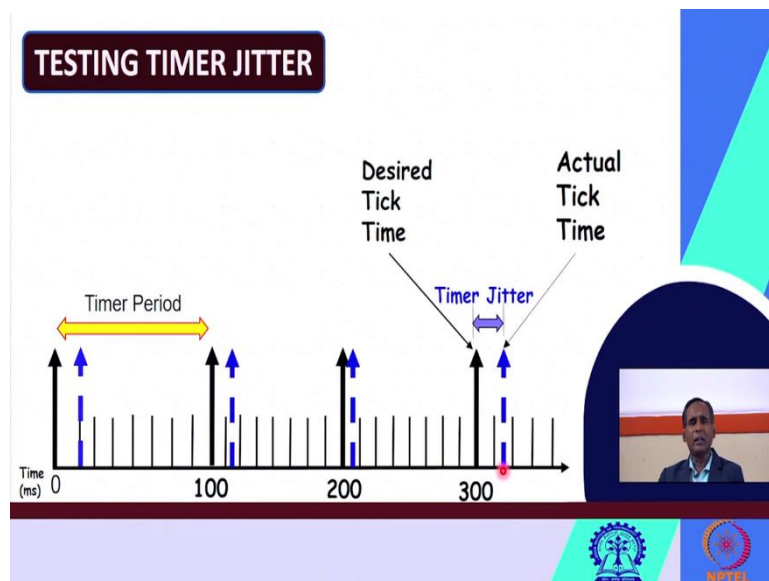
Benchmark	Description	Aspect Tested	Parameters
Timer Jitter	Create a periodic timer and measure the deviation between desired and actual expiration	Measures the response time of the operating system	Timer period: (1, 10, 100 ms)
Response	Execute a fixed processing load. Measure execution time over a number of runs.	Determines if a thread can respond in a deterministic fashion.	Type of processing: (add, copy, whetstone)
Bintime	Call a time of day clock and measure interval between calls	Measures the maximum kernel blocking time.	None

So, so, this I have already told you deterministic benchmarks what includes the following benchmarks timer jitter, response, and bintime. It will say timer jitter, what does it do? It creates a periodic timer and it measures the deviation between the desired expiration and the actual expiration. So, what is the desired value? What is the actual value? Find out the difference you will get this timer jitter. Jitter means roughly we are saying delay.

So, what aspects are tested here? It measures the response time of the operating system. It measures the response time of the operating system. And what parameters it use? Time period maybe of the order of 1, 10, or 100 milliseconds. Then response, what does it do? It executes a fixed processing load. It measures the execution time over a number of runs or over a fixed number of runs. What aspects it tests? It determines if a thread can respond in a deterministic fashion or not.

Then what parameters can be used, like the type of processing for example, whether add, copy, whetstone those operations can be considered in response to benchmark. Then bintime, what does it do? This benchmark calls a time of a day clock. So, please remember this is very important, bintime benchmark it calls a time of a day clock and it measures the interval between the calls. So, what aspects are tested by bintime? So, it measures the maximum kernel blocking time. And what parameters are there? No parameters are here. Now, let us see the timer jitter first.

(Refer Slide Time: 25:20)



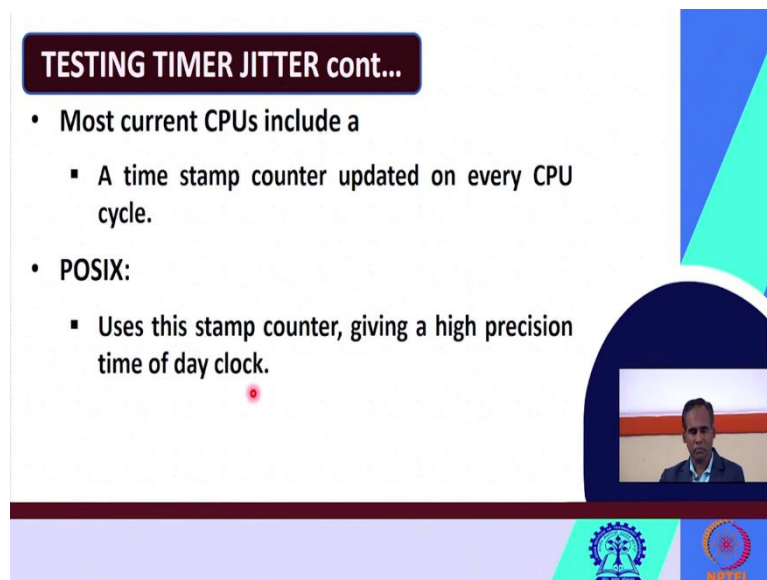
So, how to test the timer jitter, testing timer jitter? You see, so, in the X-axis we have taken time in terms of milliseconds. So, this is the time period, this block of the time period like this. Suppose, the desired tick time should be here at 300 millisecond you should get the tick time. This is the desired tick time but actually you are getting the tick time after some time, then this difference that means actual tick time - desired tick time, you find out the difference this difference is known as the timer jitter. This is known as a timer jitter.

(Refer Slide Time: 25:56)

So, how to go for testing timer jitter? First you create a timer, then set it to expire at some period, some specific period when it expires then you determine what is the actual expiration time, then you find out the deviation, and then you compute this deviation. This deviation

between the actual and desired expiration time is called as jitter. This deviation between the actual and the desired expiration time is known as jitter. In this way you can find out the, you can find out this timer jitter.

(Refer Slide Time: 26:27)



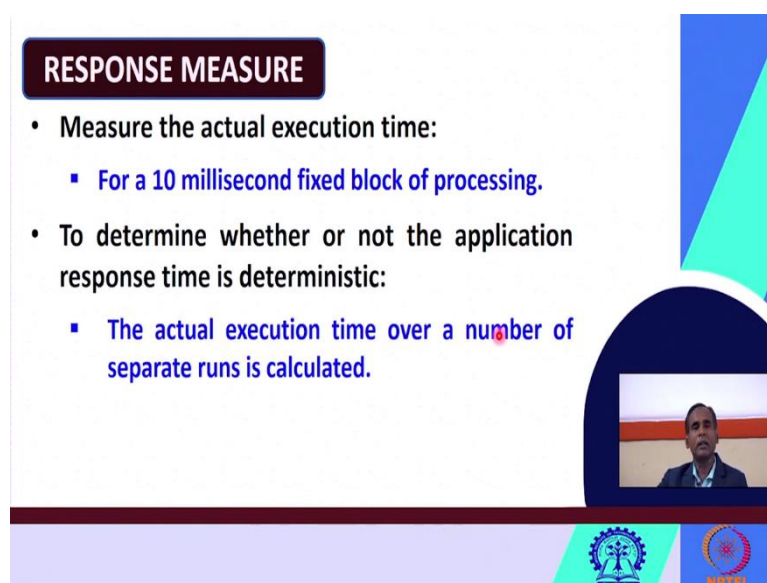
**TESTING TIMER JITTER cont...**

- Most current CPUs include a
  - A time stamp counter updated on every CPU cycle.
- POSIX:
  - Uses this stamp counter, giving a high precision time of day clock.

The slide features a dark blue header with the title in white. The main content is a white area with a dark blue border on the right and bottom. A video inset of a man speaking is located in the bottom right corner. Logos for IIT Bombay and NPTEL are at the bottom.

Most current CPUs they include a timestamp counter updated on every CPU cycle. Most of the existing CPUs they include a timestamp counter which is updated on every CPU cycle. We have already known POSIX. POSIX uses this stamp counter and gives a high precision time of the day clock. POSIX standard uses this stamp counter and it gives a high precision time of the day clock. This is about this timer jitter.

(Refer Slide Time: 26:57)



**RESPONSE MEASURE**

- Measure the actual execution time:
  - For a 10 millisecond fixed block of processing.
- To determine whether or not the application response time is deterministic:
  - The actual execution time over a number of separate runs is calculated.

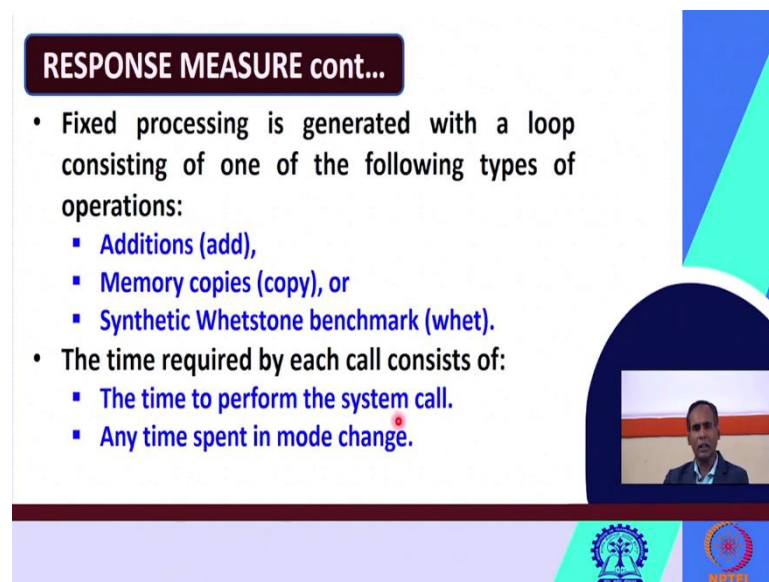
The slide features a dark blue header with the title in white. The main content is a white area with a dark blue border on the right and bottom. A video inset of a man speaking is located in the bottom right corner. Logos for IIT Bombay and NPTEL are at the bottom.



Now, let us quickly look at the response measure. It measures the actual execution time. Response measure what does it do? It measures the actual execution time for a 10-millisecond fixed block of processing. To determine whether or not the application response time it is deterministic or not what is being done, the actual execution time over a number of separate ones is calculated.

In order to know that to whether or not the application response time is deterministic, what you have to do or what is been done? The actual execution time over a number of separate runs is calculated then you can determine whether this application response time is deterministic or not.

(Refer Slide Time: 27:42)



**RESPONSE MEASURE cont...**

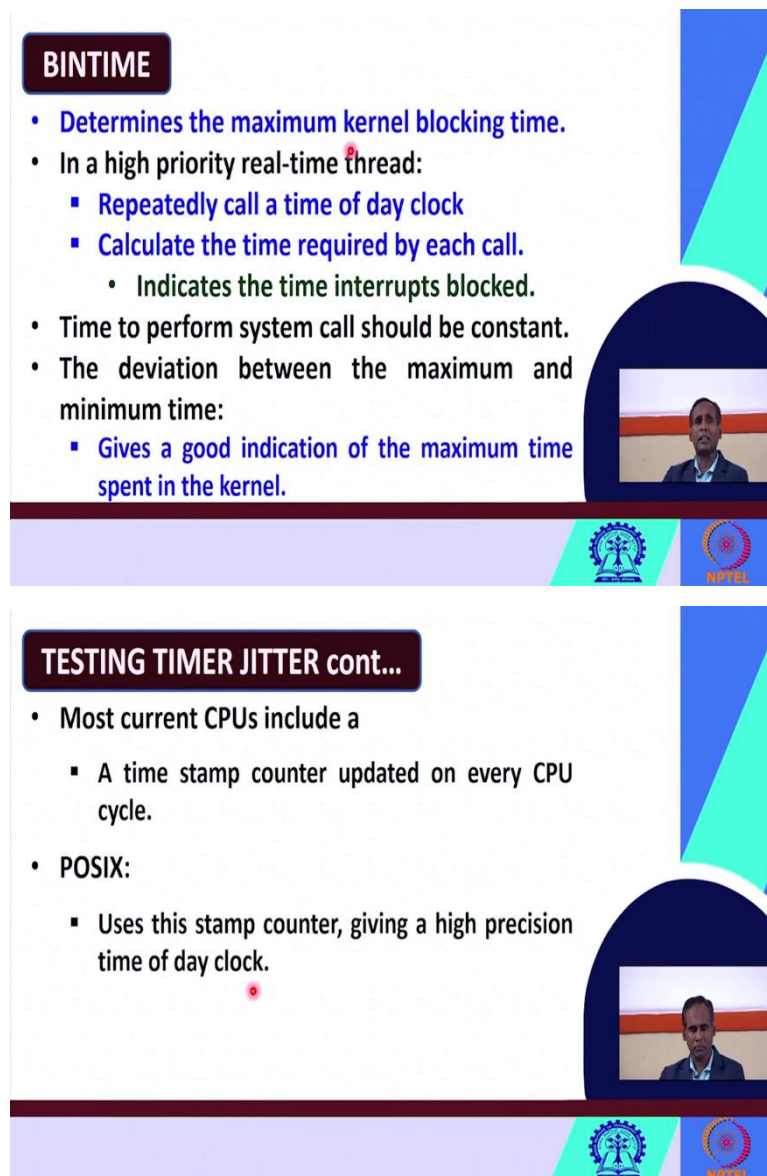
- Fixed processing is generated with a loop consisting of one of the following types of operations:
  - Additions (add),
  - Memory copies (copy), or
  - Synthetic Whetstone benchmark (whet).
- The time required by each call consists of:
  - The time to perform the system call.
  - Any time spent in mode change.

The slide features a video inset of a man speaking, set against a background with blue and green geometric shapes. Logos for IIT Bombay and NIPTEL are visible at the bottom.

So, fixed processing is generated with a loop consisting of one of the following types of operations. So, in response measure what is being done. So, fixed processing is generated with a loop consisting of the following types of operations. You may consider this addition operation or the memory copies operation, it is denoted as copy, or synthetic whetstone benchmark maybe which is denoted as whet.

The time required by each call consists of two things. The time required by each call consists of two important components first one the time to perform the system call and second one any time that is spent in the mode change. That means you are changing the mode from one to another. So, the time you required by each call consists of the time to perform the system call and any time that is spent in a changing the mode.

(Refer Slide Time: 28:31)



The image shows two presentation slides. The top slide is titled 'BINTIME' and lists several bullet points. The bottom slide is titled 'TESTING TIMER JITTER cont...' and also lists bullet points. Both slides feature a video inset of a speaker in the bottom right corner and logos for IIT Bombay and NPTEL at the bottom.

**BINTIME**

- Determines the maximum kernel blocking time.
- In a high priority real-time thread:
  - Repeatedly call a time of day clock
  - Calculate the time required by each call.
    - Indicates the time interrupts blocked.
- Time to perform system call should be constant.
- The deviation between the maximum and minimum time:
  - Gives a good indication of the maximum time spent in the kernel.

**TESTING TIMER JITTER cont...**

- Most current CPUs include a
  - A time stamp counter updated on every CPU cycle.
- POSIX:
  - Uses this stamp counter, giving a high precision time of day clock.

We will say the last benchmark in the suite, deterministic benchmark. That is bintime. It determines the maximum kernel blocking time. So, bintime it determines what, it determines the maximum kernel blocking time. If you are considering a high priority real time thread, what does it do? In a high priority real time thread, it repeatedly calls a time of day clock.

I have already told you the time of day clock which is used in this previous one, I have already told you time of day clock in this timer jitter I have already told you this time of a day of clock, we have already seen, is not it? This we have already seen it uses a time of the day clock we have already seen.

And now, this I have already told you this time of day clock. Then what I was saying, in a high priority real time thread, it repeatedly calls time of the day clock, it calculates the time required

by each call. These indicates the timer interrupts blocked. So, this will indicate, this indicates the time interrupts blocked. The time to perform system call should be constant.

Please remember the meantime, the time to perform the system call should be constant it should not change. The deviation between the maximum and the minimum time it will give a good indication of the maximum time spent in kernel. Because I have already told you bintime, it determines what? It determines the maximum kernel blocking time. So, this deviation between the maximum and the minimum time it gives you a good indication of the maximum time spent in the kernel.

(Refer Slide Time: 30:05)

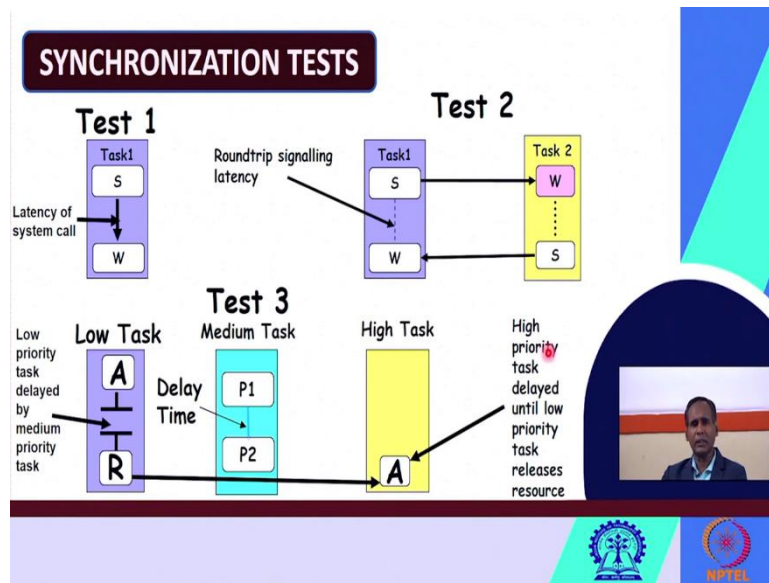
Benchmark	Description	Aspect Tested	Parameters
Sync	Measure the latency of thread to thread or process to process synchronization	Measures the context switching time between threads and processes	Type of semaphore: (POSIX named/ unnamed semaphore, etc)
Message passing	Measure the latency of sending data from thread to thread or from process to process	Measures the possible throughput of data between processes and threads	Data buffer size; process to process or thread to thread
RT Signals	Measure the latency of real time signals between two processes	Measures the latency of POSIX real-time signals	None

Then let us quickly see about the second category benchmarks for real time system, that is latency benchmarks. So, a latency benchmark includes three individual benchmarks, one is a Sync, which stands for synchronization, then message passing, then RT signals. The descriptions are quite easy, aspects tested are quite obvious, and the parameters like this Sync uses type of the semaphore.

Basically, Sync what does it do? It measures the latency of thread to thread or process to process synchronization. It measures the context switching time between what, between the threads and processes. And the parameters used are like the types of semaphore or POSIX named or unnamed semaphore, etcetera. Then message passing this benchmark it measures the latency of sending data from thread to thread or from process to process. It measures the possible throughput of data between the processes and the threads.

And the parameters what parameters it use? It uses the following parameters like data buffer size, process to process or thread to thread. Similarly, RT signal, RT signals it measures the latency of real time signals between any two processes. What aspects are tested? It measures the latency of the POSIX real time systems. And parameters is, no parameters are used. Now, we will see the most important one, this is the synchronization.

(Refer Slide Time: 31:23)



So, synchronization test, how does it appear, the synchronization tests I have shown in this figure, here you can see there are three tests are involved test 1, test 2 and test 3. So, in test 1 we are dealing with latency of system call like this and in test 2 round trip, signaling latency. Similarly, test 3 we check about this whether the low priority tasks are delayed by any medium priority tasks or not and we are also checking this priority inversion, high priority task delayed until the low priority task releases resource, that we measure that we test. Now, let us see how we do, let us see the test individually.

(Refer Slide Time: 32:00)

**SYNCHRONIZATION TESTS**

- **Test 1 :** To measure the latency of semaphore system calls
  - A single thread signals and then waits (W) on a semaphore.
- **Test 2 :**
  - Uses semaphores to signal between two threads.
  - The threads are either in a single, or two different processes.
  - Context switching time =  $(T2/n) - T1$ .

**SYNCHRONIZATION TESTS**

**Test 1**

Latency of system call

Task1

S

W

**Test 2**

Roundtrip signalling latency

Task1

S

W

Task 2

W

S

**Test 3**

Low priority task delayed by medium priority task

Low Task

A

R

Delay Time

Medium Task

P1

P2

High Task

A

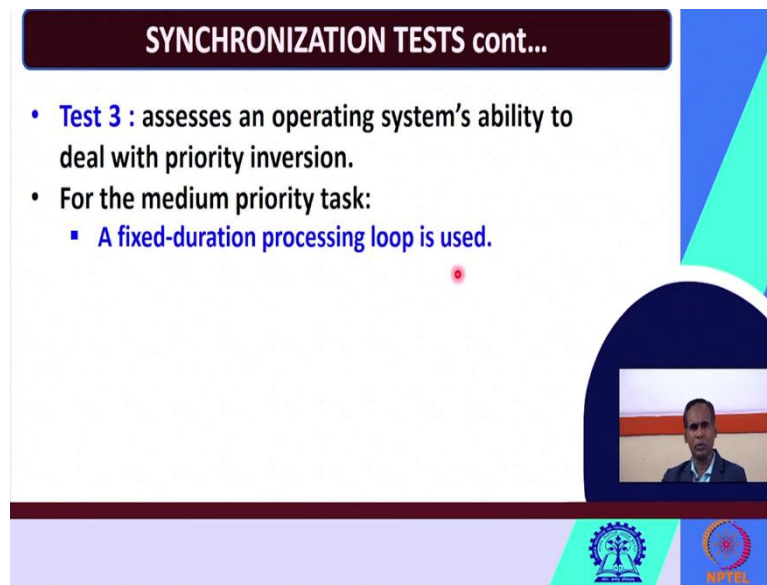
High priority task delayed until low priority task releases resource

Test 1, what it is test 1, I have already told you, we are checking this the latency of a system call, we are examining the latency of the system call. In test 1, we are performing the following things, test 1 is used to measure the latency of the semaphore system calls. We have already known about this latency. So, test 1 is used to measure the latency of the semaphore system calls.

Here a single thread it signals and then it waits on a semaphore. And in test 2 what we do, this is the test 2, when say two threads maybe there. Test 2 it uses semaphores to signal. Test 2 uses semaphores two signals between two threads, the threads are either in a single process or two different processes.

The threads they may be either in a single process or two different processes. The context switching time here in test 2 can be calculated as follows. In test 2, the context switching time can be calculated as what this  $T2 / 2 - T1$ . Already we have seen  $T2$  and  $T1$ , so context switching time is equal to  $T2 / 2 - T1$ . So, in this way we see what is performed in test 2, next we will see how it performed in test 3.

(Refer Slide Time: 33:13)



**SYNCHRONIZATION TESTS cont...**

- **Test 3** : assesses an operating system's ability to deal with priority inversion.
- For the medium priority task:
  - A fixed-duration processing loop is used.

The slide features a dark blue header with the title in white. The main content is on a white background with blue and green geometric accents on the right. A small red dot is visible below the second bullet point. A video inset in the bottom right shows a man speaking. Logos for IIT Bombay and NPTEL are at the bottom.

Test 3, it assesses an operating systems ability to deal with priority inversion to what extent the operating system it deal with the priority inversion. You have already known the different priority inversion mechanism. So, the PIP, PCP, etcetera. For the medium priority task, a fixed duration processing loop can be used. For the medium priority task, a fixed duration processing loop is used.

(Refer Slide Time: 33:37)

**SYNCHRONIZATION TESTS cont...**

- If a priority inversion occurs:
  - The time between when a low priority task acquires the resource and when the high priority task receives it will be high.
- If OS synchronization mechanism prevents a priority inversion:
  - Then this time will be negligible.

The slide features a video inset of a man speaking, and logos for IIT Bombay and NPTEL at the bottom right.

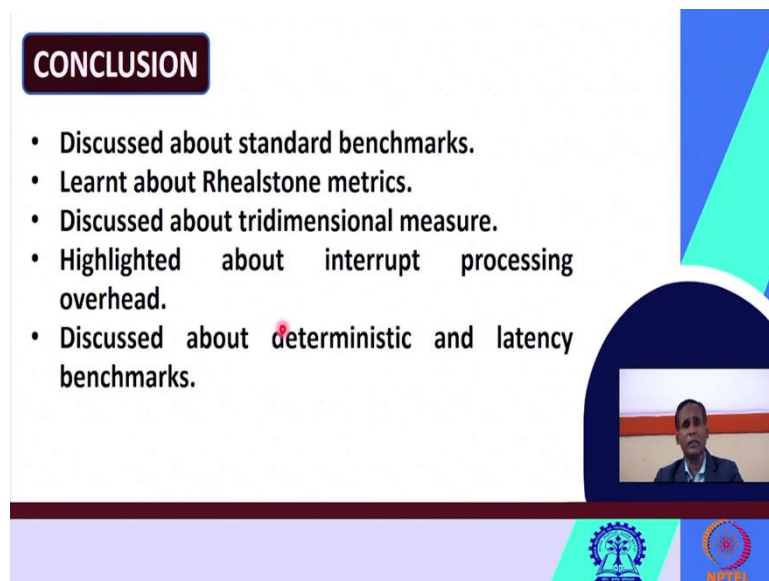
If a priority inversion of course, because I have already told you test 3, it is used to assess the operating systems ability to deal with the priority inversion. To what extent it deals with priority inversion. If a priority inversion occurs, then what will happen, the time between when a low priority, you know already priority inversion in earlier classes.

So, here the time between, if a priority inversion occurs then the time between when a low priority task acquires the resource when the high price task receives the resource it will be very high. So, if a priority inversion occurs, then the time between, when a low priority task it acquires the resource when the high priority task receives it will be very high.

So, it is a very simple thing I have already known that whenever priority inversion generally occur, then the time between the two things that means when low priority task acquire a resource and when the high priority task receives it, it will be very high, if operating systems synchronization mechanisms prevents these priority inversions, then this time will be very much negligible.

If the operating system or this real time operating system synchronization mechanism it prevents this priority inversion, in fact, it should prevent, then this time will be very negligible. If operating systems requires some mechanisms, they prevent this priority inversion then this time will be very much negligible.

(Refer Slide Time: 35:00)



**CONCLUSION**

- Discussed about standard benchmarks.
- Learnt about Rheapstone metrics.
- Discussed about tridimensional measure.
- Highlighted about interrupt processing overhead.
- Discussed about deterministic and latency benchmarks.

The slide features a video inset of a man speaking in the bottom right corner. At the bottom, there are logos for IIT Bombay and NPTEL.

So, today we have discussed the different standard benchmarks, we have seen about Rheapstone metrics, tridimensional measure, IPO, what your deterministic benchmarks, latency benchmarks. So, various standard benchmarks we have seen. Particular learnt about the different metrics which are coming under the Rheapstone metrics, then we have discussed about the tridimensional measure, and interrupted processing over at IPO.

We have also discussed two important metrics for real time operating system, two important metrics or two important benchmarks for real time systems. Those are what deterministic and latency benchmarks. We have discussed about two important benchmarks for real time systems, which are named as deterministic and latency benchmarks.



(Refer Slide Time: 35:42)



**REFERENCES**

1. Rajib Mall, Real-Time Systems: Theory and Practice, 1st Edition, 2007, Pearson Education
2. C. M. Krishna & K. G. Shin, Real-Time Systems, 2017, Tata McGraw Hill Education

The slide features a dark red header with the word 'REFERENCES' in white. Below the header, two references are listed in black text. A small red dot is positioned between the two references. On the right side, there is a video inset showing a man in a suit. The slide is decorated with a blue and green geometric design on the right and a dark red footer containing logos for IIT Bombay and NPTEL.

We have taken from these books, these details. Thank you very much.