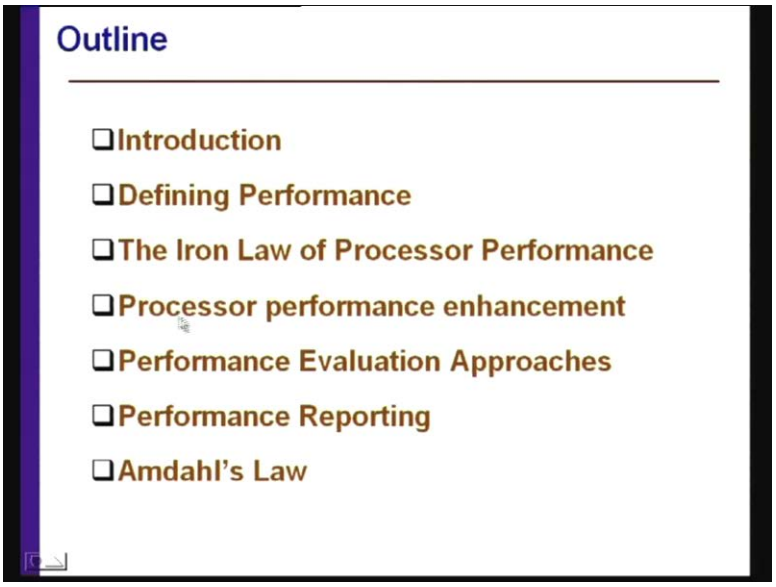
High Performance Computer Architecture
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 2
Performance

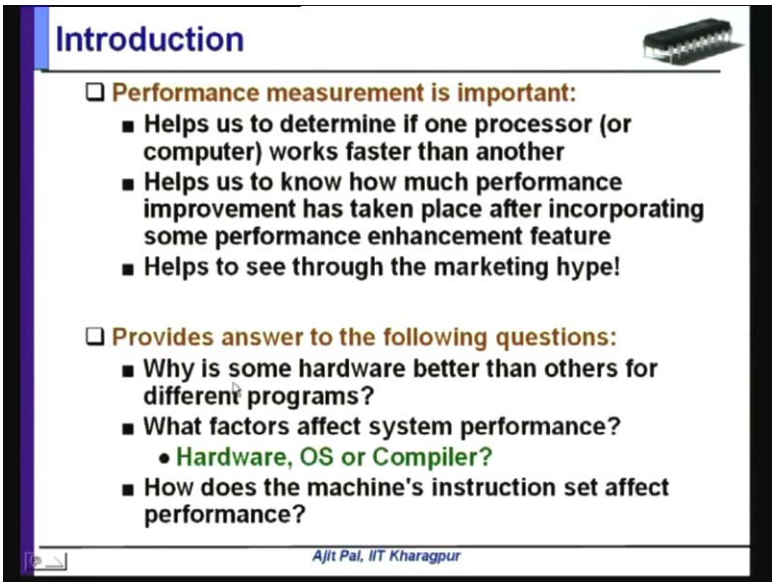
Hello and welcome to today's lecture on performance and obviously, by that I mean performance of the computer system.

(Refer Slide Time: 01:02)



And here is the outline of today's lecture after giving a brief introduction, I shall define what durably mean by performance. Then, I shall discuss about the iron law of processor performance, the various factors on which the performance depends. And then, we shall discuss, what do we really mean by processor performance enhancement and then I shall discuss about performance evaluation approaches, after the performance is defined, how can we evaluate performance and the various approaches that can be used. And another very important aspect is performance reporting, how you can report the performance by using a single number. And finally, I shall conclude my lecture by discussing Amdahl's law, which is related to performance measurement.

(Refer Slide Time: 02:09)



Performance measurement is very important because, it helps us to define one processor to define, if one processor works faster than the other that means, we are discussing about the high performance computer architecture. Obviously, we shall discussing various techniques by which the performance of the processor can be improved or enhanced. And in that context the performance plays a very key role and we have to see how really you can measure performance and see it.

So, also it helps us to know how much performance improvement has taken place after in cooperating some performance enhancement feature. So, you will see that we shall be incorporating various performance enhancement features, whether in a compiler or in the organization of the computer like pipe lining and various other things. And by doing so how the performance is improving, that we shall know from this performance measurement. And it also helps to see through the marketing hype, you know there is the marketing hype that whenever a new processor introduces people say this is so much better than other processors.

So, this how true is that hype that can be evaluated with the help of, with the help of performance measurement. It will also provide answers to the following questions, number one is why is some hardware better than others for different programs. So, you will be measuring performance; obviously, by running programs and how a particular processor is performing better, for a particular program than other program. So, that

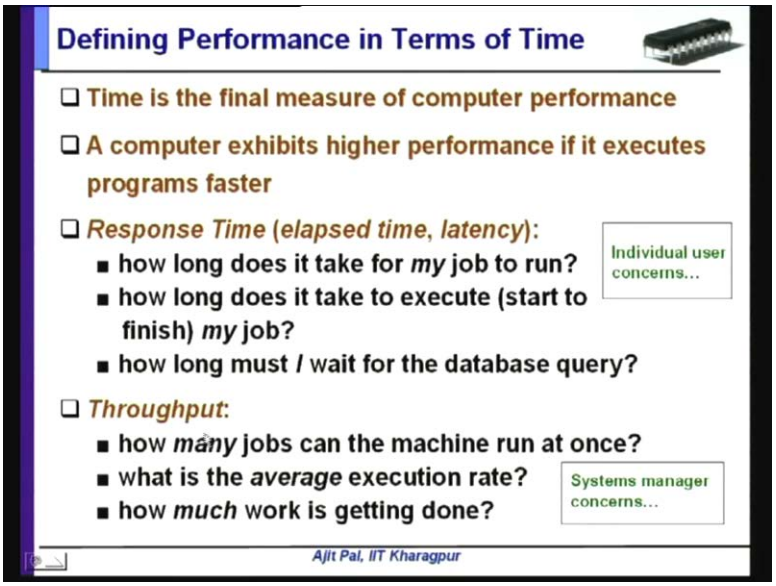
particular analysis is very important and that answer will get from this particular topic. Then, what factors affect system performance?

You will find that there are various factors which affect the performance obviously, the first thing is the hardware. Hardware means the processor by with which the weight is implemented and incorporating arithmetic logic unit then, register file, controlling unit and so on. So, the hard ware which implements the processor that will definitely play very important rule. Second important parameter is operating system, operating system will schedule various tasks to the processor and obviously, operating system will also play a very important role in deciding the performance of the processor. And also we know, we will be using whenever writing program in a high level language. It is like c, FORTRON and other programming languages, it has to be converted into machine language before you can run on a processor.

And obviously, the efficiency of the compiler, the performance of the compiler will also affect the overall performance of the system. So, these are the various factors which will affect the performance. And last but not the least, how does machines instruction set affects performance? You know a particular processor is characterized with the help of a instruction set, as I briefly discuss in my last lecture.

Now, that instruction set can be different for example, it can be a for a risc processor there can be one type of instruction set, for sisc processor instruction set is different. And as the depending on the complexity and nature of instruction which we, which we shall call as instruction set architecture, the, how the performance depends on that. So, that, that, those thing those aspects are to be studied.

(Refer Slide Time: 06:22)



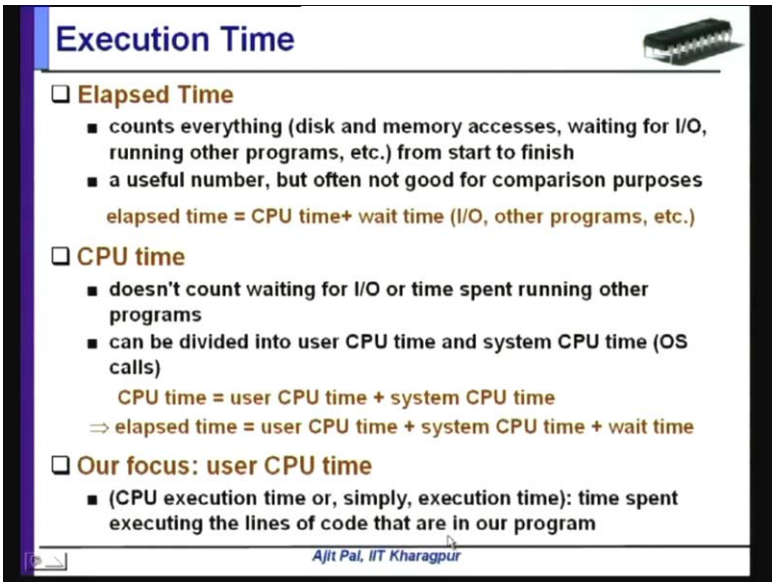
Question is how do you really measure performance obviously, it is time, time and time. Time is the ultimate measure of performance and what kind of time? You will find that a computer exhibits higher performance, if it executes the program faster. Obviously, whenever you are trying to measure a performance, you will be measuring the execution time. Faster is the execution time, the performance is better so, but whenever you measure the time, there are various ways you can do it. For example if you, if you think from this individual prospecting then, response time or left time is factor, which is important to an individual user.

An individual user will submit a job, submit a task and after the task is submitted when he get the result, that is the last time, that time is important when user. Because, that is the time for which one has to wait for getting the result. On the other hand for a system manager the perspective is different they are from system managers perspective, through put is the most important parameter. Because he is more interested in a, in how many jobs can be machine can run at once that means, in a multiuser, multi programming environment a computer is executing many tasks of many users.

And obviously, how many tasks per unit time is executed by the processor, that is important to the system manager. And what is the average execution time and how much work is getting done by the computer. So, you can see these two are not really same, response time and through put this two are not same and from individual point of view,

will be interested in response time, for system management point of view will be more interesting in throughput.

(Refer Slide Time: 08:43)



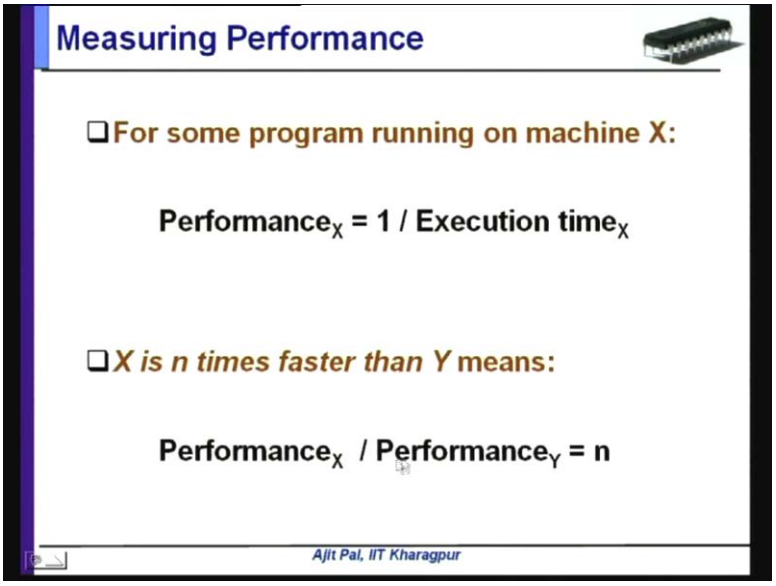
Let us see what you really mean by elapsed time. So, it counts everything these can memory axis waiting for IO running other program etcetera from start to finish. That means, a job will be submitted and a fraction of CPU time, a particular user will get during the running of his task. And obviously, there will be other times like the time required speech from one task to another task, waiting time for IO and the operating system time, these are the various times that will come. So, you can state it in terms of a number and that number elapsed time is CPU time plus weight time, weight time can be for waiting for IO, it can be waiting for other programs running etcetera.

It can be for because of phase fault, later on we shall discuss about all these things, then comes the CPU times. So, the CPU time does not count waiting for IO or times spend running other program. So, it is simply finds how the time required to perform a particular task. So, it can be divided into CPU time, user CPU times plus system CPU time. Obviously, operating system cause will be involved, when a particular user is running a program. So, CPU time is equal to user CPU time plus system CPU time, then elapsed time is equal to user CPU time plus system CPU time plus wait time.

So, that means elapsed time fees particular user encounters can be divided into three component, user CPU time plus system CPU time plus weight time, but for this

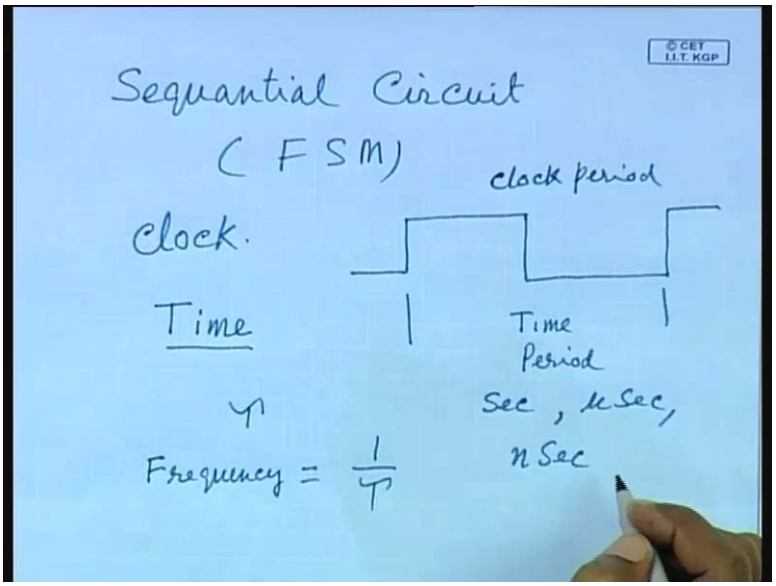
particular course, our focus is on user CPU time. We shall not be bothered about system time or the weight time, when the processor is running somebody else's job. So, we shall be primary concerned about user CPU time, that is the CPU execution time or simply we shall call it execution time. So, that is time spend executing the lines of code that are in our program. So, a particular user is running a program and how much time is required to run that particular program, that is the time that will be used as the as a measure of performance.

(Refer Slide Time: 11:15)



Now, whenever you try to measure a performance, you know performance is a relative thing for some program running on machine x is equal to performance x is equal to 1 by execution time. So, execution time and performance is inversely related because, the larger the execution time, performance is worse, smaller is the execution time performance time better. So that means, a larger execution time will lead to inferior performance. Now, whenever you compare x with y, x is n times faster than y by that we mean performance y is equal to n. So, this is how you shall try to measure performance.

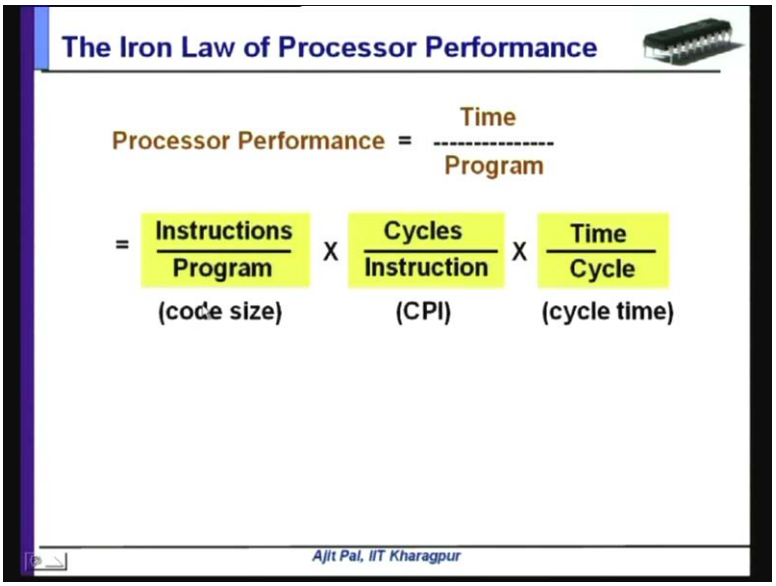
(Refer Slide Time: 12:11)



Now before I go into these topics, another very important aspect I should tell, you know you may be knowing that a computer is nothing but a sequential circuit, a sequential circuit or finite state machine will require a clock. So, we are interested about time, how do you relate time, actually it is related to clock, a computer is controlled to by a clock, a processor. So, a clock is nothing but a repetitive, you know and this is called the time period. Time period of the clock usually it is stated in terms of second or may be whenever it is small, it is stated in terms of micro second or sometimes it is nano second as the speed is increasing.

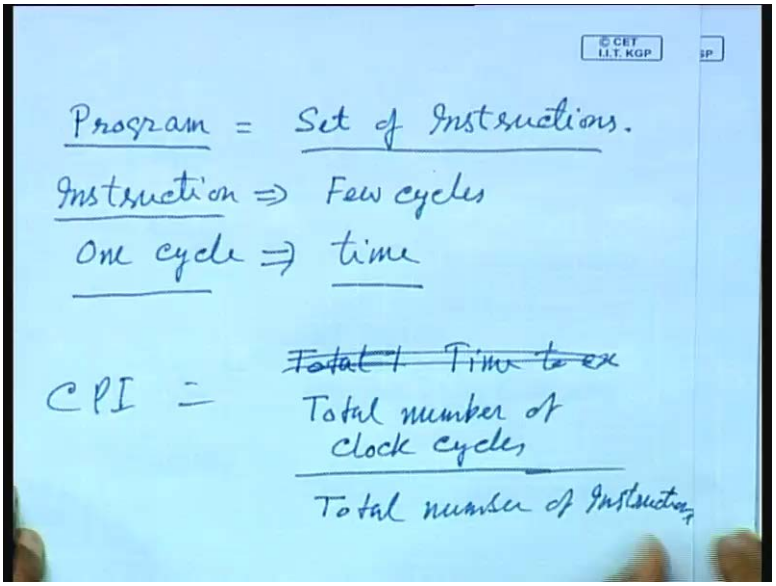
So, it is some form of time in terms of second may be micro second, nano second. So, this is, this is, this may be called one clock period or time period. So, if tau is the time period then the frequency of the clock is related to the time period in this manner, is 1 by tau. So, sometimes you will see we shall try to express the execution time in terms of number of clocks because, ultimately the processor is controlled by a clock and number of clocks is related to the execution time.

(Refer Slide Time: 14:13)



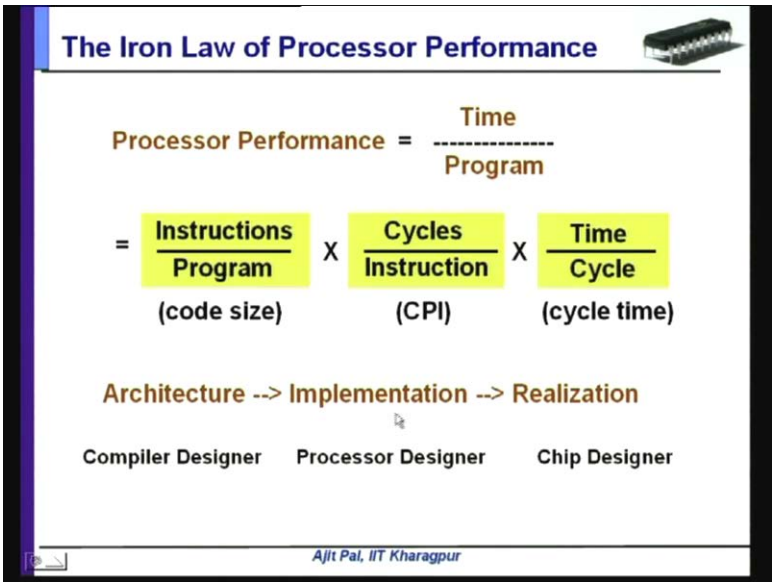
So, let us see how we are going to express performance in terms of this various, this clock frequency, clock time period. So, processor performance is equal to time required to execute a program and it comprises three components. So, we can say that you know a program is a set of instructions, then an instruction requires few cycles may be one, few cycles, for few clock cycles to executor instructions then, one cycle will require some time.

(Refer Slide Time: 15:20)



So, you can see we have got three important parameters a program can be decomposed into a number of instructions, which you may call instruction count. Then each instruction, instruction will require one or few cycles and then each cycle will take some time.

(Refer Slide Time: 15:39)



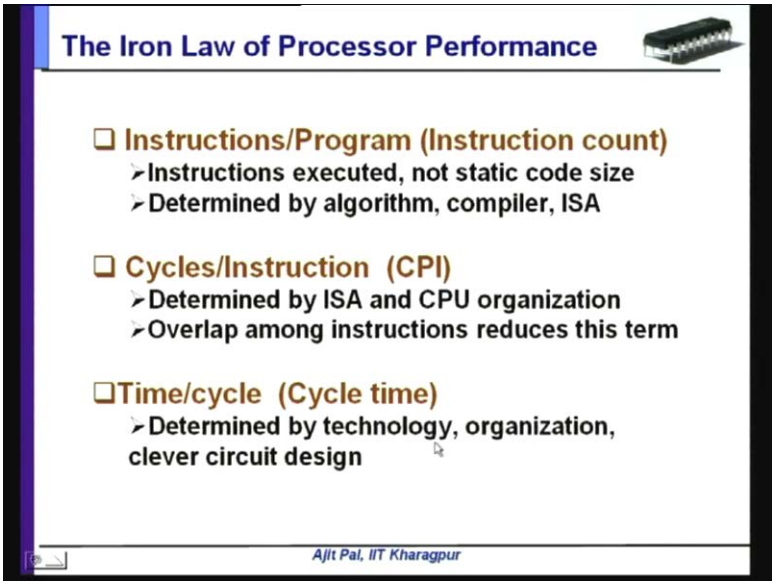
So, this is what is shown here. So, that means, the processor performance is nothing but instruction, number of instructions program you may call it instruction count. So, it depends on the size of the code and then, the number of cycles per instruction that is known as CPI of cycles for instruction. So, that CPI, the cycle for instruction can be one, can be more than one and later on we shall see it can be less than one as well. So, and then cycle time, clock cycle time I have already explained that is the time period, it is the clock cycle time, that time period is can also change and.

So, it is a you have to consider all these three parameters and you will see that these three parameters is affected by three important aspects. One is your architecture, architecture is represented by the instructions set architecture, I have deeply discussed in my last lecture and in a next lecture actually elaborate in more details. Then, the, the processor is implemented for a given instruction set architecture. So, processor, instruction set processor is essentially is represented by the implementation. So, then comes the realization, realization is implementing the, a particular processor, instruction set

processor with the help of some electron circuit may be transistors or integrated circuit or VLSI chip.

So, you can say there are several designs are involved in the, in the, in executing a program or in the design of the system. Compiler designer, processor designer and chip designer and their design will affect the overall processor performance. So, these three factors are to be considered when we considering processor performance. So, actually this is known as iron law of processor performance, product of these electrons, code size that is instruction counts CPI and cycle time.

(Refer Slide Time: 18:20)



Now, that number of instruction per program or the instruction count will depend on obviously, you may be asking what you really mean by number of instruction per program. Say a program that source code will consist of several instruction, now that is static in nature, but whenever you execute then we call it that number of instructions that is executed by the processor. So, that means the size of the source code as got nothing to do with the, or is really independent on the dynamic size of the code that is executed by the processor. So, we are more interested on the dynamic size, not the static size of the code.

So, static size of the code can be very small say, you have written a program where it is looping. So, we looping within the source size is this, but it may be looking thousand times. So that means, this code will be repeated thousand times so, you have to take into

consideration that the dynamic size, that the total number of instruction is getting executed. So, do not get confused with the source code size. So, it is essentially the instructions executed not the static code size. So, this particular factor instruction count is dependent on three important parameters. It is first of all it depends on the algorithm, your implementation a particular algorithm and the way the algorithm is implemented, you can define some better algorithm to reduce the size of the code.

It also depend on the compiler, complier can perform a number of optimizations to reduce the size of the code. So, those who have attended a course on compiler, they must have studied various compiler optimization techniques. So, the size of the dynamic code will dependent on the compiler then, last but not the least it will depend on the instruction set architecture that means, instruction which can be executed by the processor. So, you can see instruction count is dependent on several factor then, cycles for instruction is dependent on the ISA and the CPU organization. That means, that instruction set up architecture as I said it can complex instruction set architecture, it can be reduce instruction set architecture.

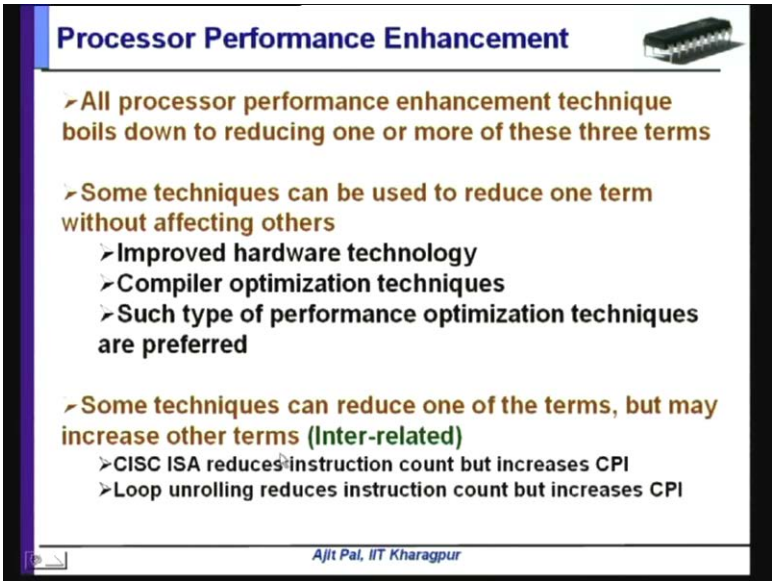
So, depending on the complexity of the instructions the cycles that will be that means the average number of cycles that is required by the instructions. That means, how do you measure CPI, CPI you measure CPI is equal to total number of instructions executed or you can say time to execute number of cycles, so not this. Total number of instruction cycles, total number of not instruction number of, I should say clock cycles by total number of instructions. So, these are the ratios will decide CPU. So, total number of clock cycle by total number of instructions for a particular program. So, it is the dynamic, that number is dynamic, dynamic in the sense while executing what is the total number of instructions.

So, it is it determined by the ISA and the CPU organization, the way the CPU is organized later on we shall discuss about in detail about this and overlap among instructions reduces this term. So, we shall discuss about different technique for reducing the CPI like techniques like pipelining and other things, by which this CPI can be reduced. Then the last time, the cycle term is determined by the technology organization and design. So, first factor is determined by the technology, technology means the VLSI technology, as you know the various site technology is improving over time.

And you may have heard of Moore's law, I briefly mention in my last lecture about it and as for most law as you know, the size of the device is reducing every 18 months the size is becoming half. And as the size becoming half, the capacitance is getting reduced and then it is becoming faster. So, the technology is determining the cycle time, cycle time in the earlier it was, it was micro second now it has become nano second. Because, clock rate of the processor as you know earlier it was, the clock frequency was few mega hertz.

Now, the clock frequency of the modern processor is splatted in terms of giga hertz, how it happens? That is primary because of the advancement of technology and of course, it is also dependent on the organization particularly pipe line and clever circuit design. So, you can see these three factors, together decides the processor performance.

(Refer Slide Time: 24:25)

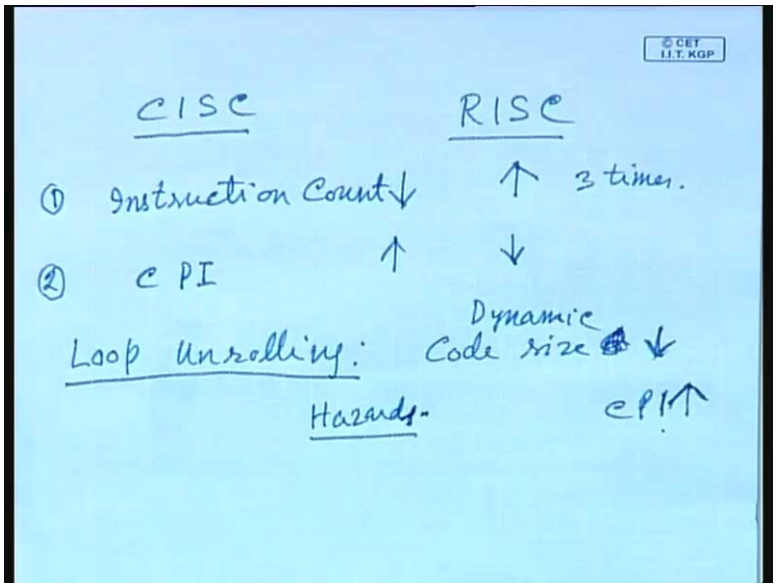


And obviously, whenever you, you decide about improving, to improve the performance of processor, you will see that all processor performance enhancement technique voice down to reducing one or more of these three terms. So, in the here we have discussed of three terms, you will see that whenever you try to improve processor performance either we try to reduce the instruction count. Or we try to reduce the CPI or we try to reduce the cycle time or sometimes more than of these two times together we try to. So, now the question is some techniques can be used to reduce one term without effecting others.

In other words what I am trying to tell, there exist some techniques which does not affect others. For example, whenever the technology is enhanced, when you are going from one technology generation to next technology generation, circuit is becoming faster. So, other things are reaming same. So, the cycle time is getting reduced, but CPI will be remaining unaffected and also the instruction count also will remain be affected, if only the technology enhancement is considered. So, the one is input hardware technology similarly, whenever you go for complier occupation techniques, you remove some dead code and various other compiler optimization techniques we use, what happens by using this techniques?

The instruction count reduces without effecting CPI or cycle time. So, such type of performance optimization techniques are preferred because, these techniques does not affect other parameter. Only it affects only one of the three parameters. So, these are preferred however, there are, there exits other technique which are inter related some technique can reduce one of the terms, but may increase other terms.

(Refer Slide Time: 26:50)



So, let me explain with the help of the example. Let us consider CISC and RISC. So, in whenever you go for complex instruction set architecture, we know that instruction count reduces. So, instruction count reduces whenever you for CISC so, on the other hand in case of RISC this instruction count increases. Because, number of instructions required for a RISC CISC processor is roughly three times, may be three times more than CISC

processors. So, we find that one particular parameter is decreasing for CISC and increasing for RISC.

On the other hand if we consider CPI, cycles per instruction then we will find whenever you are executing a complex instruction obviously, the number of cycles required to execute the complex instructions, number of cycles would be more. So, it will increase this CPI, on the other hand the risc processor since the instruction are simple and you know the number of cycles required to execute a simple instruction obviously, the CPI will reduce. So, you can see whenever we go for, we go for and decide I mean compare between CISC and RISC, you can see for one particular parameter is getting reduced. On the other hand the other particular parameter is increasing.

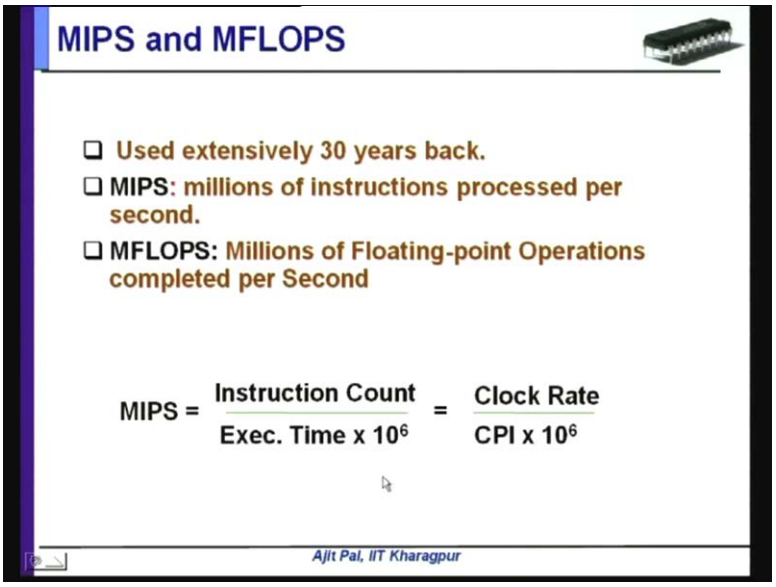
So, as consequence we cannot really say that this is better, this is inferior. Because they are inter dividend there is some kind of inter relations. SO, CISC ISA reduces instruction count, but in CPI similarly another technique we shall discuss later on, which is known as loop unrolling. So, with the help of loop unrolling, what we do a particular loop is unrolled say for example, a loop without unrolling may require the lesser memory in your program. And obviously, the whenever you do loop unrolling what happens? The number of instruction increases, the code size increases, code size increases, no sorry, the dynamic code size will reduce.

Because, whenever you do loop unrolling this static code size reduce, but increases, but dynamic code size reduces. Because, many loop unrolling those decisions, wherever you take, if there will be decision making things, those decision makings will be reduced whenever you go for loop unrolling. So, you will see the code size will reduce that dynamic code size, I should write dynamic code size, dynamic code size will reduce in a loop unrolling technique. That means, you can say that instruction count will reduce however, what will happen? Whenever you are executing the program, because of the, you know the static size of the code that you have to load in a program, what can happen? The, it will lead to increase in CPI because of, because of you know hazards whenever the static code size increases then, the hazard increases, as the results the cycle per instruction increases.

So, later on we shall discuss more detail about loop unrolling and we shall see how dynamic code size reduces, but CPU can increases because of various hazard that may

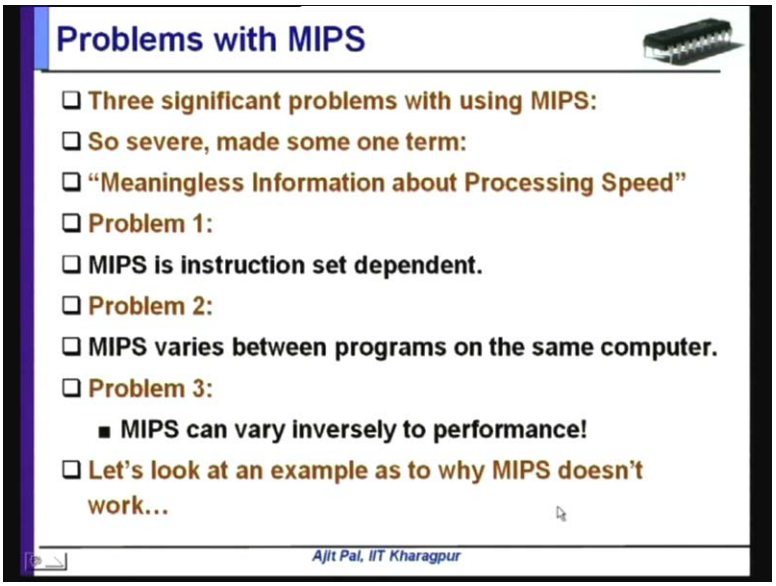
occur whenever you try to execute a code. So, the loop unrolling reduces instruction count, but increases CPI. So, you can see there are many factors which are interrelated and in such cases, we have to be very careful whenever you try to measure the performance by using these techniques.

(Refer Slide Time: 31:42)



Earlier, one very important parameter was MIPS or mega flops. MIPS stands for million instruction per second and MFLOPS stands for million floating point operations per second. So, these two were extensively used 30 years back as a measure of processor performance that means, higher the MIPS rating, the one used to consider, why processor used to be considered faster. Similarly, higher the m flops rating, a processor used to be considered faster.

(Refer Slide Time: 32:49)



But it has some draw backs, we shall see, what do you really mean by MIPS? MIPS is instruction count by excision time into 10 to the power 6 or you can say that clock rate by CPI cycles per instruction into 10 to the power 6. So, MIPS can be calculated by executing a program and it can be found used for comparison, but let us see what kind of problem face whenever you use MIPS and as a measure for performance. We encounter three significant problems when we use MIPS and these problems are so severe that, somebody commented meaningless information about processing speed.

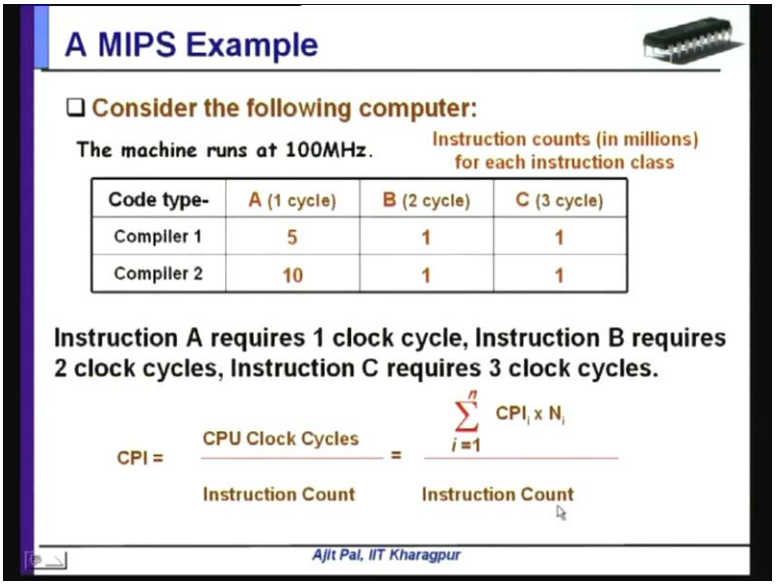
So, although for many years MIPS are used as a matrix performance measurement for long time. One important factor is MIPS is instruction set independent, as I have already told that instruction set architecture plays a very important role in deciding the performance of the processor. But it can be shown that MIPS is independent of the instruction set architecture, the reasons for that is you know it is dependent on the technology of the processor. We have seen that clock rate by CPI into 10 to the power 6.

So, it can be, it can be instruction set MIPS is instructions set dependent. That means, the instruction set dependency is occurring because, you can see CPI is there, CPI is instruction set department. And as a consequence, the MIPS is dependent on the instruction set, simply we cannot really tell in terms of MIPS, we have taken, we have to also take into consider instruction set architecture. Second is MIPS varies between programs on the same computer that means, whenever you take different programs,

different programs and run on a single processor then, we will find that MIPS rating is different for different programs.

That means, you cannot really tell that the higher the MIPS rating means, this is better because, for a particular the value of MIPS may be better and for another program value of MIPS maybe inferior, which is shall illustrate with example. Third problem is MIPS can be very inversely to perform, I mean that is the reason why somebody commented meaningless information about processing speed. So, let us illustrate with the help of an example and why MIPS does not work.

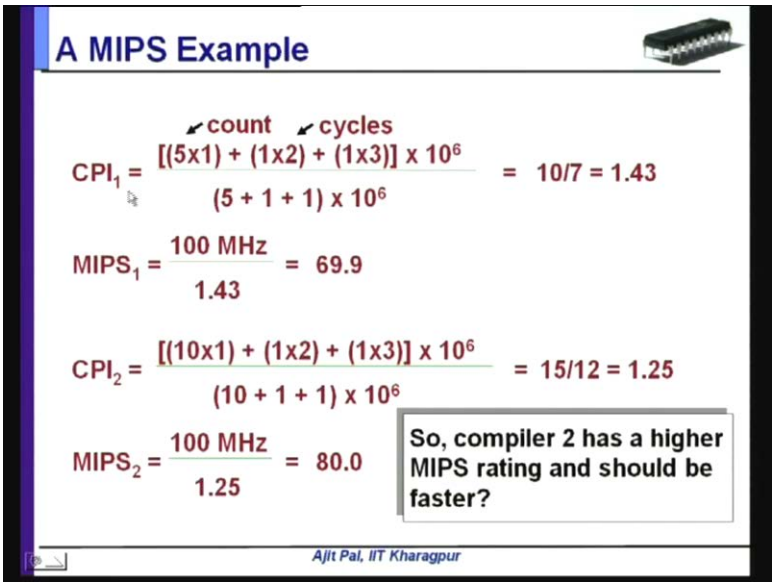
(Refer Slide Time: 35:32)



So, let us consider the following computer and we are using two compilers, compiler one and compiler two which are design for the same processor, same computer. And we have got three types of instructions category a, category b and category c. Category a instructions require one cycles, categories b instructions requires two cycles, category c requires three cycles. Now a compiler one generates category a instructions, which is 5 and category b generate 1 instruction of a category b and category c another 1 instruction.

On the other hand the compiler 2 generates 10 instruction of category 1, 1 instruction of category b and 1 instruction of category c. So, your CPI is equal to CPU clock cycles by instructions count, that is your CPI and now what you can do? You can measure the total number of CPI into n, take the summation by the instruction count for all instruction.

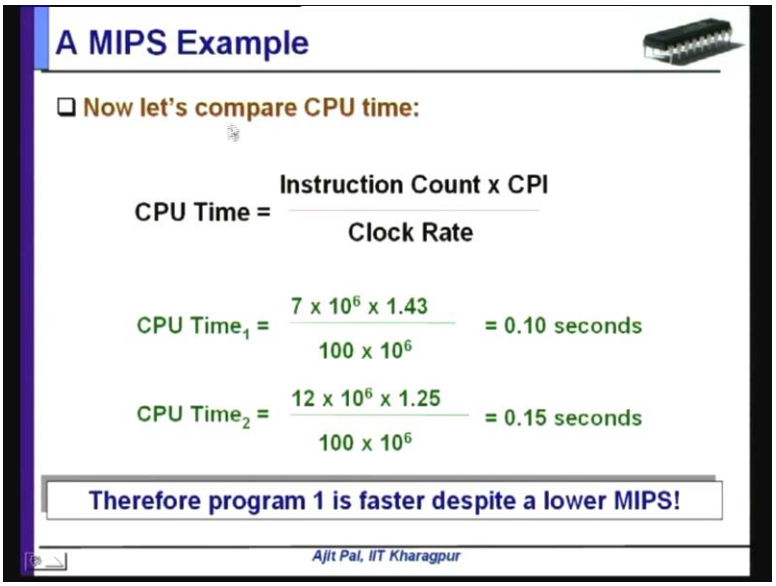
(Refer Slide Time: 36:55)



And then, you can find out the CPI for the compiler 1 and CPI for compiler 2. So, for CPI for compiler 1 is, we have seen that the type of instructions is 5 that means, I mean 5 is the number of instructions. So, it requires one cycle then 1 instruction of 2 cycles and another 1 is instruction for 3 cycle into 10 to the power 6 and CPI and total of CPI is 5 plus 1 plus 1 into 10 plus 6. So, you get the CPI, cycles per instruction is 1.43 that is for compiler 1 and so MIPS rating will be 100 megahertz by 1.43 that means. So, it is operating at 100 mega hertz. So, 100 megahertz by 1.43 we get MIPS rating of 69.9, that is MIPS rating for compiler 1 on the same processor.

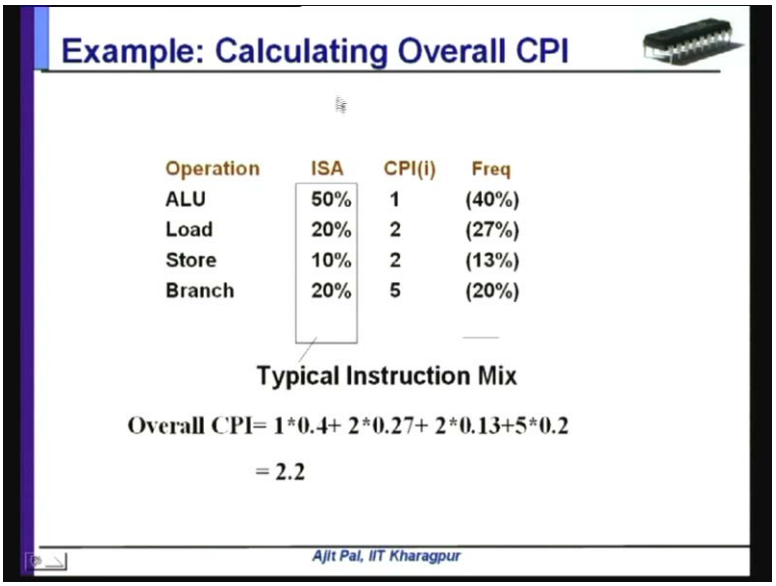
Now CPI for compiler 1 can be calculated in a similar way 10 into 1 plus 1 into 2 plus 1 into 3 summation of that into 10 by 6 by 10 plus 1 plus 1 into 10 by 6 that is the total number of instruction, that is the instruction count and this gives you a CPI of 1.25. So, we find that MIPS rating for compiler 2 on the same processor is 80. So, 100 megahertz by 1.25. So, we find that compiler 2 has a higher MIPS rating. So, it should be faster because, MIPS rating for compiler 1 for the processor is 69.1 and here it is 80.0. So, compiler 2 has a higher MIPS rating and should be faster.

(Refer Slide Time: 38:57).



Now, let us see whenever we translate it in terms of CPU time, that instruction count into CPI by clock rate for the complier 1, we get that is 0.10 second, that is the execution time. And for the complier code, the generated by compiler 2 execution time is 0.15 second. So, we find in this case earlier we found that MIPS rating that compiler 2 was giving you better performance, but on the other hand you know the CPU time corresponding to code generated by compiler 1 is lesser. So, therefore, program one is faster despite lower MIPS rating. So, we can say MIPS rating is not really reflecting the processor performance.

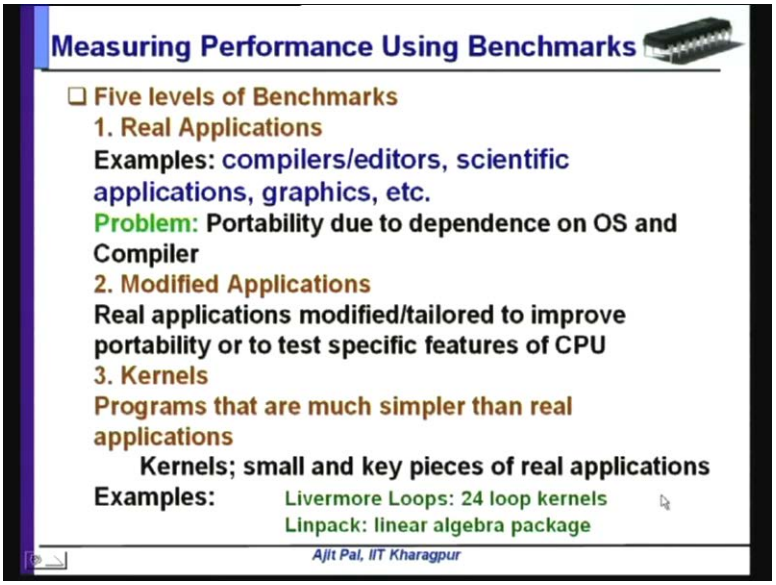
(Refer Slide Time: 39:48)



And what you can do, you can calculate overall CPI this way, this is the instruction architecture. Different types of instructions ALU operation is 50 percent, load instruction is 20 percent, store instruction 10 percent, branch instruction 20 percent and these are the corresponding CPI and these are the frequency of appearance in a particular program.

So, for a particular instruction, for a particular instruction, for the particular instruction mix and you can find out the overall CPI in this way 1 into 0.4 plus 2 into 0.27 plus 2 into 0.13 plus 5 into 0.2. So, this gives you overall CPI of 2.2. So, for a particular program so, we find this how one can calculate CPI.

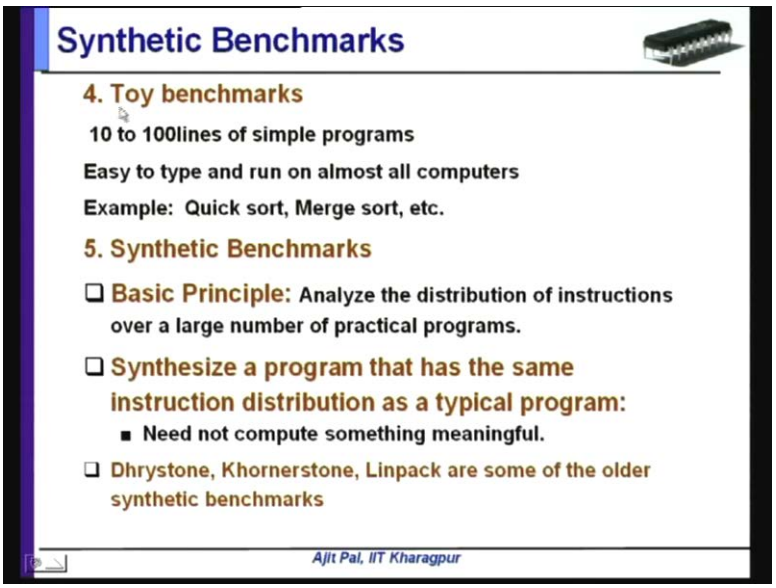
(Refer Slide Time: 40:44)



Now, whenever you try to measure performance, you have to use some program, what kind of program and these programs are known as benchmark program. So, you can see the benchmark programs can be, can have 5 different level. Number 1 is real applications, real applications that will be running in your computer, in your day today like compilers, editors, various scientific programs, graphics application and so on. And unfortunately for these real applications there is a problem of portability because, these application will be dependent on the operating system, as well as the compiler. For different computers the operating system can be different, compiler can be different, as result portability is a problem whenever try to compile the performance of real applications.

So instead of that, one can consider modified application that means, you take, you consider a particular application then, you modify a particular application and tailor it and improve the portability. So that, the portability is improved or it can test specific features of the CPU. So, specific features of the CPU that means, graphic feature or digital signal application DSP features that may be present. Those particular aspects can be specially tested by modifying the applications then, the third level of benchmark known as kernels. Kernels are very small and key pieces of real applications and since these program is very simple can be 10 to 100 lines of code and examples those are Livermore loops, 24 loop kernels and LINPAC linear algebra package. This can be used as a for the measure of the performance. So, these are known as the kernels.

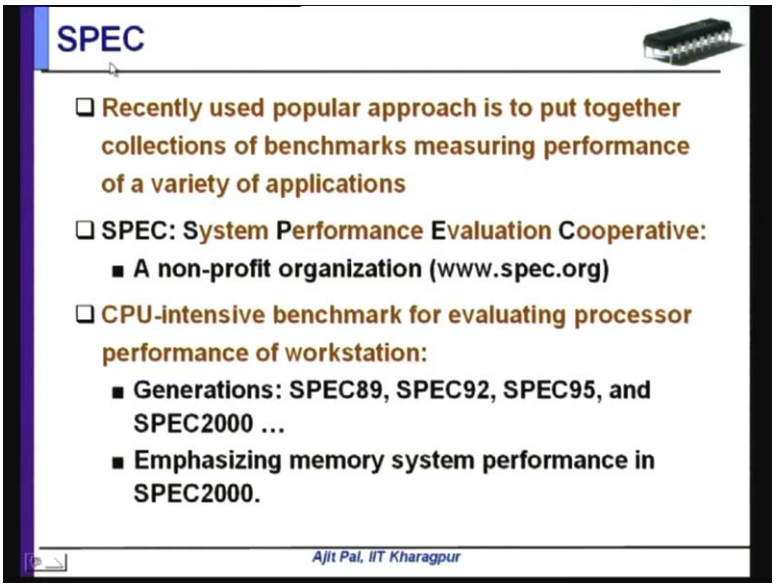
(Refer Slide Time: 43:01)



And the forth category of toy benchmarks, which are also simple program may be 10 to 100 lines of code and which are easy to type and run on almost all computers. And these are the applications which are typically given as assignment in your, to the students may be in the first year. Like quick sort, merge sort these program can be used, can be consider as toy benchmark, which can be use for the purpose of testing. However, there is an another level, which is known as synthetic benchmark. So, synthetic benchmark means you have created benchmark to analyze the distribution of instructions over a large number of practical programs.

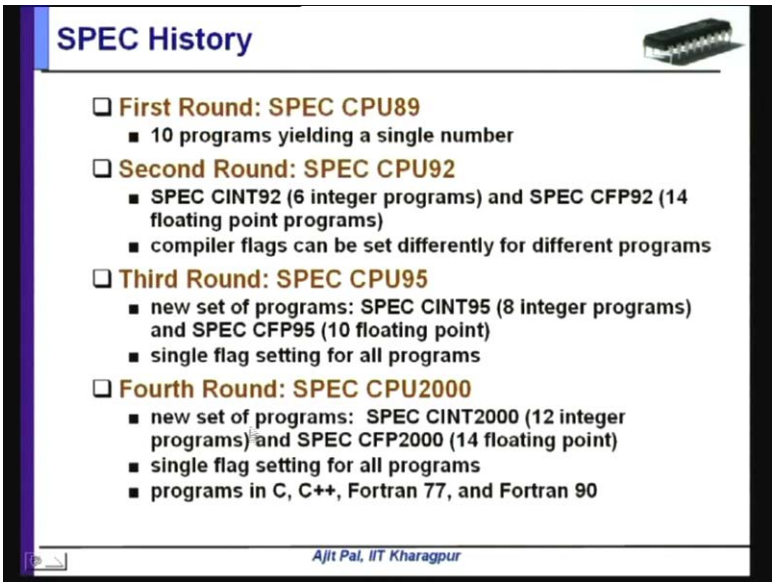
That means, you have instruction of architecture, even to test how different type of instructions are executed by the processor. So, some synthetic benchmarks are created and synthesize a program that has the same instruction distribution as a typical program. However, these programs have no real meaning to an user because, these are, they do not give you any meaningful result and examples of this synthetic program are Dhrystone Khorner stone, LINPAC these are the some older benchmark problems.

(Refer Slide Time: 44:30)



Now a day's however, people depend on SPEC, SPEC is system performance evaluation cooperative. So, recently this is the recently used popular approach where, a collection of benchmark are put together to measure the performance of the variety of applications. So, here we are not dependent on a particular application, we have chosen application from different fields which are used to measure the performance. So, SPEC is a nonprofit organization, this is the website www.spec.org and they have developed benchmark programs, which can be CPU intensive benchmark for evaluating processor performance for workstation. So, different benchmark programs have been developed for the measurement of performance of work stations, servers and so on.

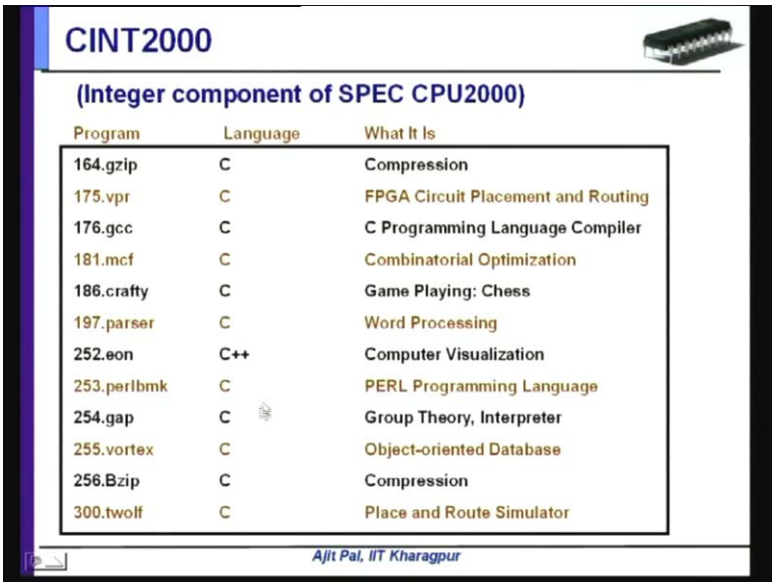
(Refer Slide Time: 45:31)



So, this is the history of SPEC first round was SPEC CPU 89, 10 programs yielding a single number. Then second round SPEC CPU 92, where 6 integer program and floating points where used. Then of course, compiler flags can be set differently for different programs in this particular case. Third round was SPEC CPU 95 so, you can see have been enhanced, the benchmark programs have been change at the processor technology improved. So, here 8 integer program and 10 floating point programs are used and in this particular case single flag setting is allowed for all programs. You have seen in the previous case compiler flag settings can be different for different programs.

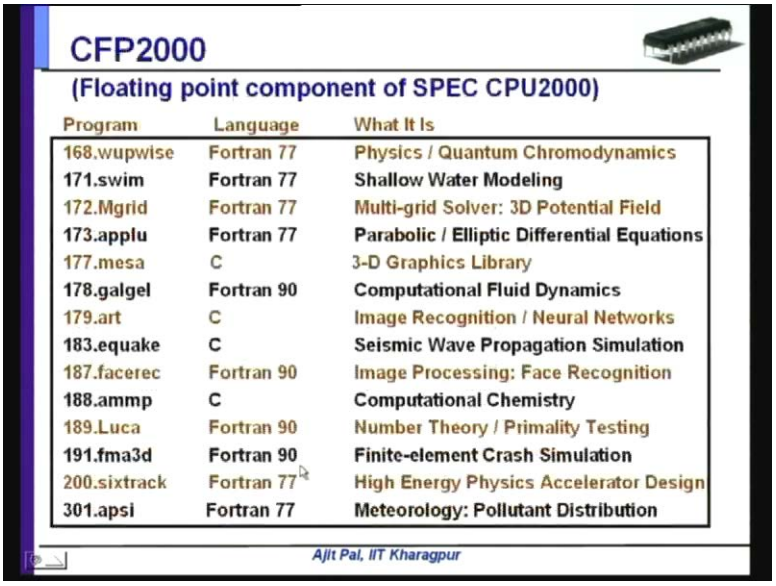
Then, the fourth round is SPEC CPU 2000, which is presently used SPEC CINT 2000 has got 12 integer programs and SPEC CFP 2000 has 14 floating point and single flag setting for all programs and these programs are written either c, either in c or c plus plus or Fortran 77 or Fortran 90.

(Refer Slide Time: 46:50)



So, here is the list for integer component of SPEC CPU 2000. So, you can see there are 12 program getting on c or c plus plus and performing different function like compression, FPGA circuit placement and routing, c programming language compiler and so on.

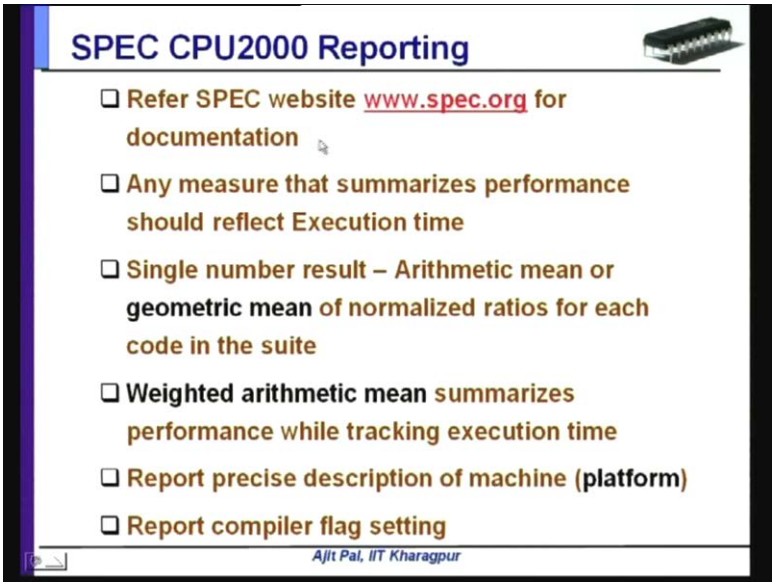
(Refer Slide Time: 47:10)



Then you have got the floating point component of SPECCPU 2000, where you have got 14 different programs written in c or Fortran 77 or Fortran 90. And various functions which is performed are given here. So, you can see physics, quantum, chromo dynamics,

shallow water modeling. So, various applications from different fields of scientific computing, image recognition, seismic wave propagation, image processing, computational chemistry, number theory. So, instead of floating I mean considering a one particular application. So, different applications of different fields have been taken to evaluate the performance.

(Refer Slide Time: 47:53)



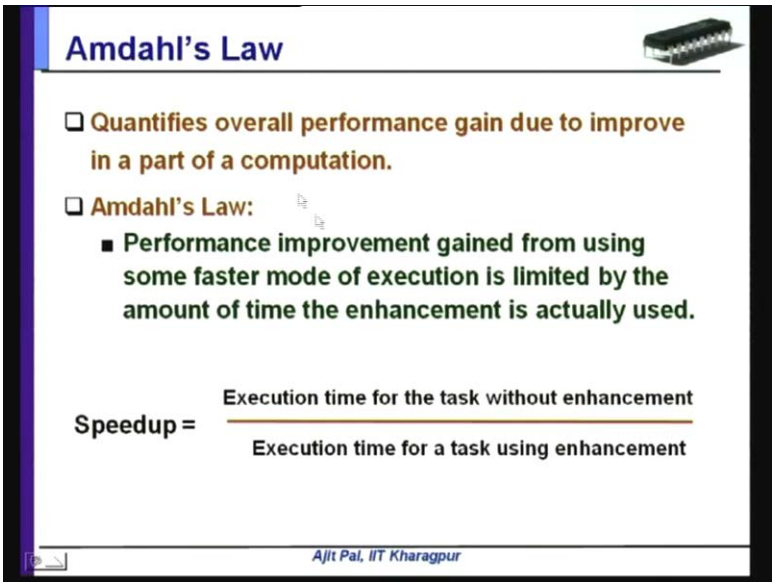
Then comes the question of the reporting, how do you really report with the help of single number? So, you have run may be 14, 12 integer programs and 14 floating programs and those, they are to be compiled and a single number has to be used to give the measure of performance, how can be done? So obviously, whenever you want to do that, you can visit website for more detail for documentation and whatever measure we use it should reflect the execution time. So, the single number result can be either arithmetic mean or it can be geometric mean of normalized ratio for each code in the suite. It has been found that arithmetic mean although gives you some measure of execution time, it is not very good.

Because, if you take one computer as reference, you get one value, if you taken another computer is reference, you get different value. So, arithmetic is I mean some have lacuna or pitfall. Similarly, geometric mean is although it is good because, it gives a single number, but unfortunately it does not give you the measure of execution time. Another term that is used is weighted arithmetic mean, which summaries performance while

tracking execution time. So, this has been found to be good. So, in addition to using a benchmarks weight, what do you have to do? You have to report precise description of the machine.

Because, platform is plays a very important role, what CPU you are using, what processor you are using, what is the on chip case, what is the off chip case, what is the main memory size all these parameters will affect the execution time. So, this particular whenever you report, you have to give precise description of a machine because, if you change some parameter of machine, if you increase the case memory size execution time can be different. So, platform information has to be provided whenever you report the performance. Then comes the compiler flag setting so, report compiler, what compiler flag setting has been used, whenever use different compilers for different programs like c, c plus plus, Fortran 77 seven or Fortran 90.

(Refer Slide Time: 50:36)

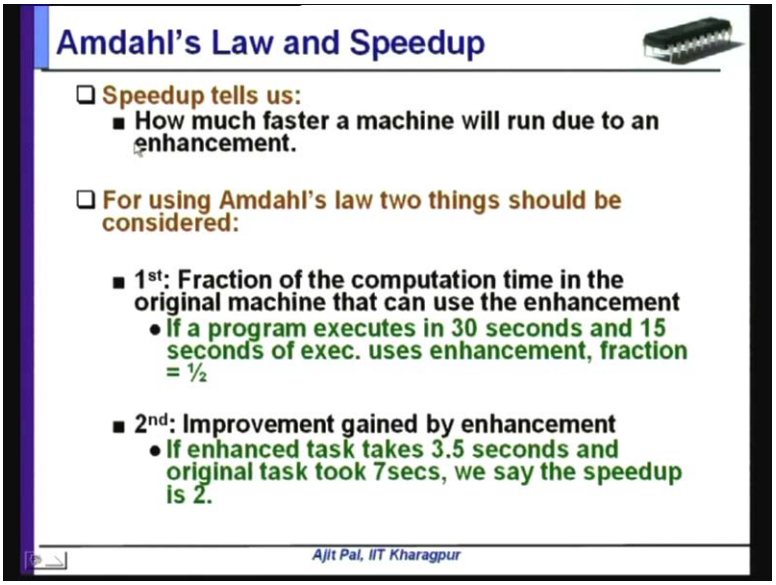


Our discussion will not be complete without considering Amdahl's law. So, it quantifies overall performance gain due to improve in a part of computation. Normally you know, we cannot really improve performance of all the aspects. For example, we may improve floating point processing by adding a coprocessor. So, only performance on floating point program execution will improve, but not the other type of programs. So, in that context Amdahl's law states that performance improvement gained from using some

faster mode of execution is limited by the amount of time, the enhancement is actually used.

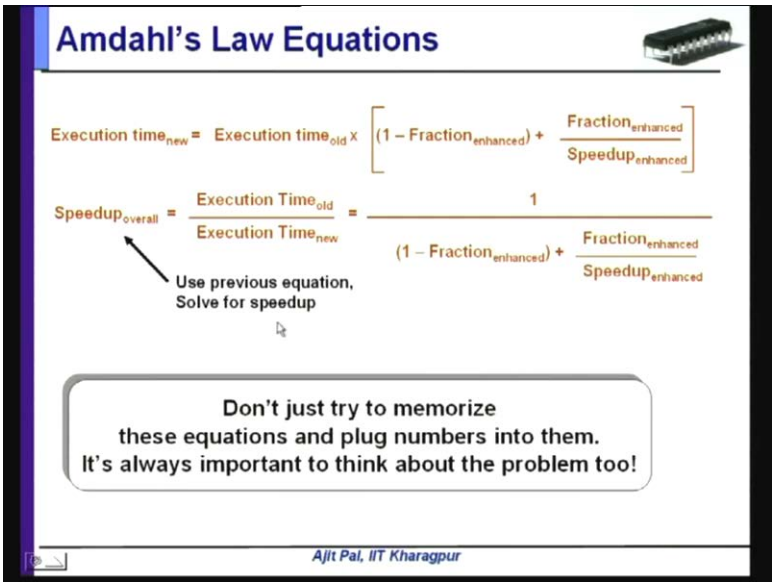
That means, you have to consider the gain that has taken place for that part only. So, you can say that speed up that is achieve by improving performance of a particular aspect is called to the ratio of execution time for the task, without enhancement by execution time for the task using enhancement. That enhancement can come in different forms, may be in the form of the, CPU, the floating point processor and or the compiler or various other aspects.

(Refer Slide Time: 52:08)



So, speed up tells us how much faster a machine will run due to enhancement and whenever we use Amdahl's law two things you should be, one should consider. Number one is fraction of the computation time in the original machine, that can use the enhancement. So, if a program executes 30 seconds and 15 second of execution uses enhancement then, that fraction is half. That means, that entire program may not be using that enhancement that has been incorporated in the system. So, that is what is stated in this. Second is improvement gained by enhancement, we have to consider overall enhancement, if enhancement task 3.5 seconds and original task took 7 seconds, we say speed up is 2.

(Refer Slide Time: 53:03)



Now let us see the formula that we use for the speed up. So, we can say that execution time for the new system after enhancement is equal to execution time old into 1 minus fraction that is the enhanced, plus fraction that is enhance by speed up that is enhanced. So, fraction of enhancement that takes place is taken in to consideration in this formula and you can find out the execution time using this formula. And then, you can find out the overall seep up, that is called, that is equal to execution time by the old system and by the execution time by the new system. And that you can find out from this, that is equal to 1 by 1 minus fraction enhancement this one, plus fraction enhancement by speed up enhanced.

So, this formula there is do not try to just memorize it these equations and these equation and plug numbers into them. So, what you should do, it is always important to think about the problems too. You have to consider what problem you are testing for what application those problems are decided and accordingly, you have to choose, you have to see on what aspect the improvement has to be done and that will lead to find enhancement in performance. So, we can summaries our lectures by mentioning the points to remember of this lecture. First of all we have seen that processor performance is dependent on three factors, the code size that is instruction count then into CPI cycle per instruction and cycle time.

So, you have to consider all these three together whenever you try to compare processor performance and particularly, we have seen these terms are inter related. So, you have to minimize time, which is the product not the isolated terms. So, you may reduce cycle time, but it may affect others as you have already seen. So, you have to consider all these three factor together.

So then, the use of bench mark suite to measure performance I have already told the use of SPEC. And you have to report with the help of single number and to do that we have seen, you can use different techniques, but we have seen that weighted arithmetic mean gives you good result because, it ultimately tracks the execution time. So, with this we have come to the end of today's lecture on processor performance, in our next lecture we shall discuss about instruction set architecture.

Thank you.