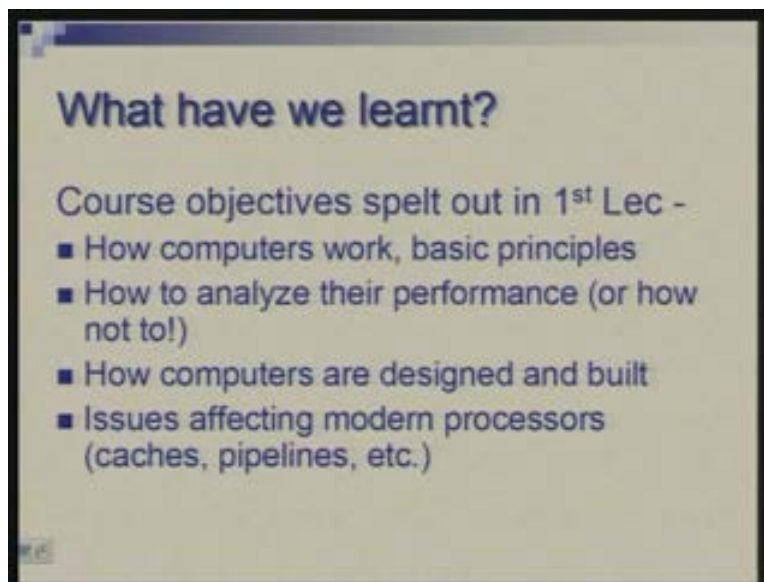


Computer Architecture
Prof. Anshul Kumar
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture - 38
Concluding Remarks

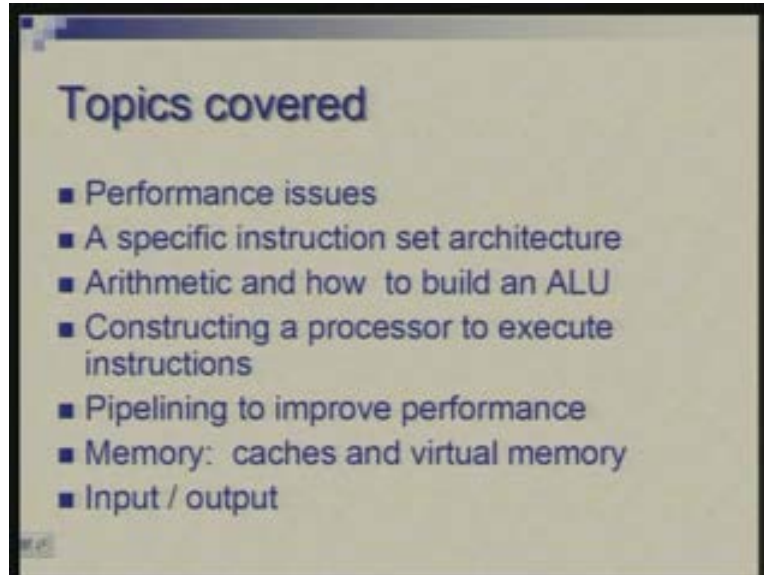
This is the last lecture on computer architecture and we conclude series of lectures. So today we will go back to what we had set out to do in this course and see where do we stand. So let us try to review what have we learnt in this course.

(Refer Slide Time: 1:22)



As it was spelt out in the first lecture we had listed a few course objectives. The objective was try to learn how computer work; what are the basic principles; how we analyze the performance; how we design and build the computers and how do we improve performance; what are the issues which have to be kept in mind in the modern processors: issues like caches or memory hierarchy, pipelining and so on.

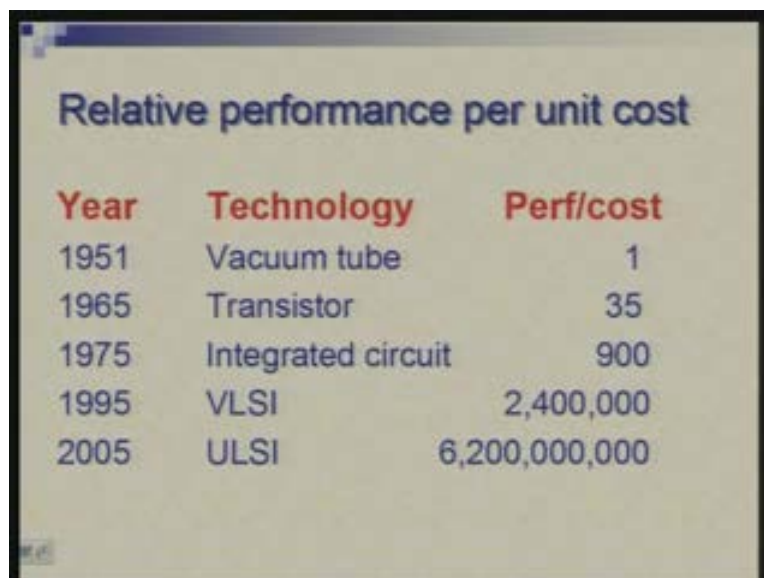
(Refer Slide Time: 2:10)



Topics covered	
■	Performance issues
■	A specific instruction set architecture
■	Arithmetic and how to build an ALU
■	Constructing a processor to execute instructions
■	Pipelining to improve performance
■	Memory: caches and virtual memory
■	Input / output

So, in summary we have covered various topics. We talked about how performance is defined. We took a specific instruction set and discussed some core instruction; then we went out to build an ALU which will carry out those instructions and then we constructed a complete processor to execute these instructions. We saw the concept of pipelining to improve the performance. We saw that memory is hierarchical not just a flat array of words and finally to complete the picture we talked about input output.

(Refer Slide Time: 2:43)

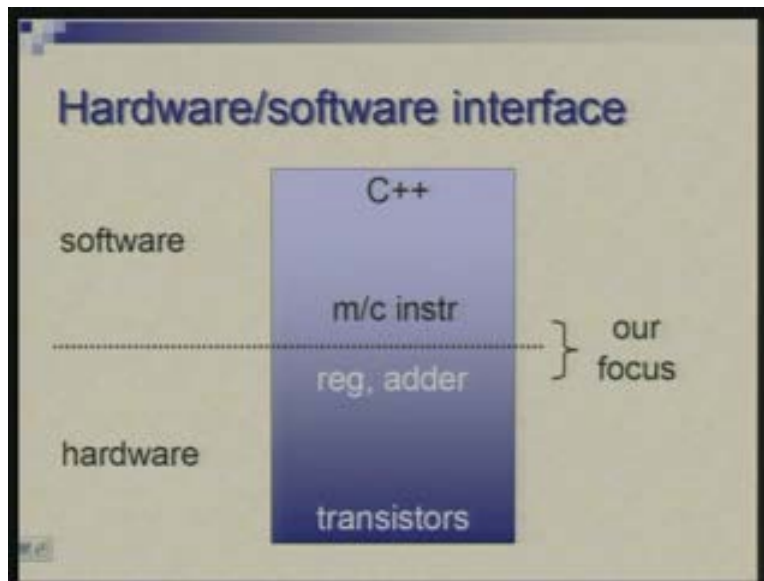


Relative performance per unit cost		
Year	Technology	Perf/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit	900
1995	VLSI	2,400,000
2005	ULSI	6,200,000,000

We began by looking at how the technology has been moving over several decades and this was the table which we had looked at on the first lecture. I have added one more

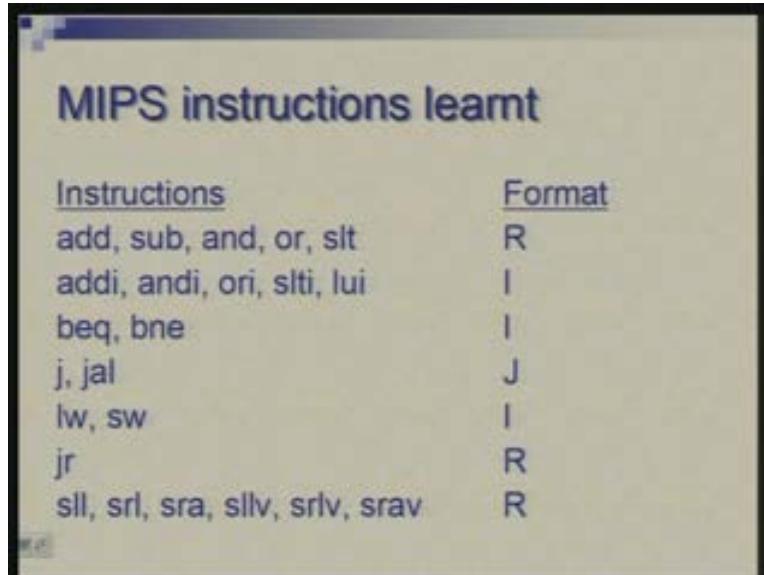
entry here the last one. So you would notice that there has been an exponential growth in performance what you can get out of a single monetary unit so performance per rupee or per dollar. The figures here are of course relative values. So, taking what you get from a vacuum tube system as 1 the other figure has been projected. So there has been an exponential growth; this has also divided the computers historically into generations.

(Refer Slide Time: 03:30)



So our focus while talking of architecture has been on the interface of hardware and software where the hardware and software meet which is essentially machine instructions as seen from the program end and the building blocks like ALUs, adders, registers, multiplexers, buses and so on as seen from the hardware end.

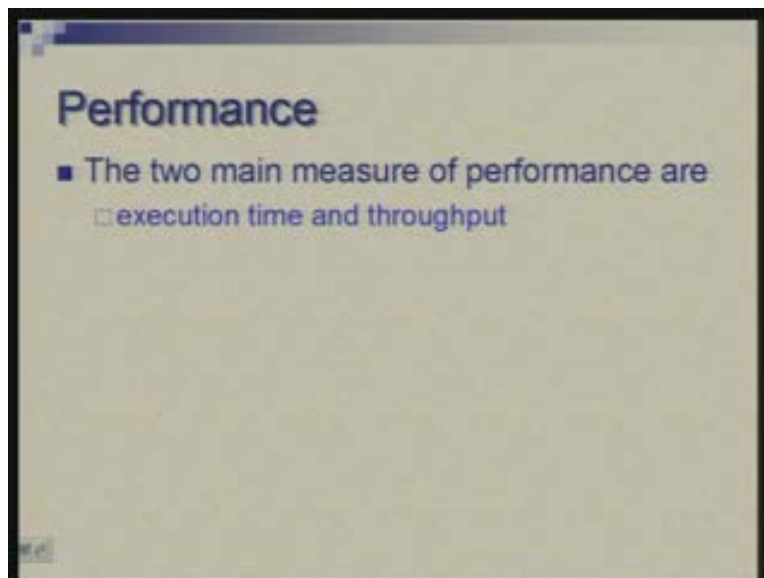
(Refer Slide Time: 3:58)

A presentation slide titled "MIPS instructions learnt" showing a list of MIPS instructions and their corresponding formats. The instructions are grouped into three categories: R-format (add, sub, and, or, slt, jr), I-format (addi, andi, ori, slti, lui, lw, sw, beq, bne), and J-format (j, jal).

<u>Instructions</u>	<u>Format</u>
add, sub, and, or, slt	R
addi, andi, ori, slti, lui	I
beq, bne	I
j, jal	J
lw, sw	I
jr	R
sll, srl, sra, slv, srlv, srav	R

The small set of instruction where we really focused is shown here. These are for MIPS processor and it is a processor of..... what is called risk variety reduce instructions at computer. On the whole there are very few instructions and simple instructions. Each instruction tries to do one particular task and not mix many things. The advantage of such choice is in building a fast processor; ease of doing pipelining and also ease of generating code. So there are three formats only in this: R, I and J and the instruction we talked about nicely fall into these three categories.

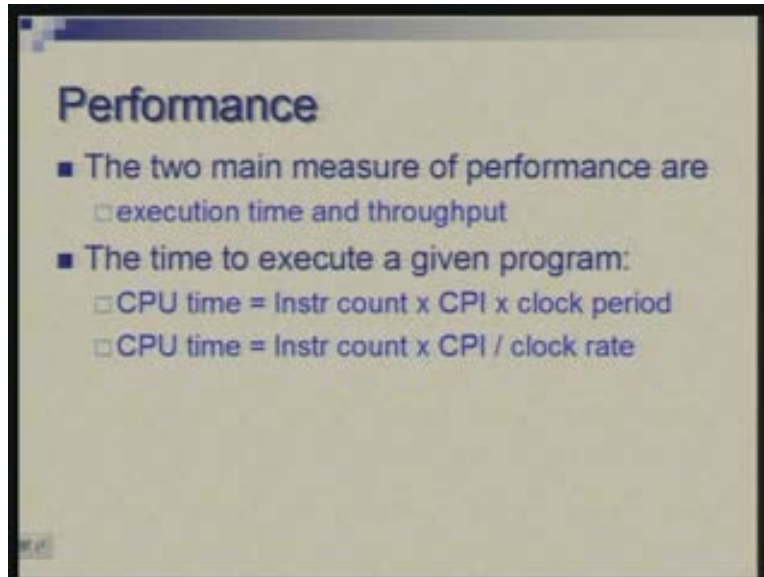
(Refer Slide Time: 4:46)

A presentation slide titled "Performance" with a bullet point stating that the two main measures of performance are execution time and throughput.

<u>Performance</u>
■ The two main measure of performance are <ul style="list-style-type: none">□ execution time and throughput

The performance was an important issue which we discussed in the beginning and lot of discussion was pertaining to how to improve performance. Of course there other aspects when you design a system we have not gone into all of those. particularly power consumption is very important and unless that is checked the power being produced or power being dissipated in the processor chip per let us say square centimeters increasing tremendously and you need elaborate arrangement for ((.....5:22.....)) otherwise system cannot work.

(Refer Slide Time: 5:50)



So, from performance point of view we looked at two possible directions: execution time and throughput. Throughput is a measure which is at a gross level when you look at collectively several tasks being executed by a processor. But for an individual user execution time is important which can be related to three basic parameters: the instructions executed in a program; the instruction count, cycles per instruction on an average and the clock period or clock rate.

The factor which influence these are varied: the compiler which generates a code, instruction set architecture, what instructions you have chosen, micro-architecture that means how you implement those instruction and the underlying technology.

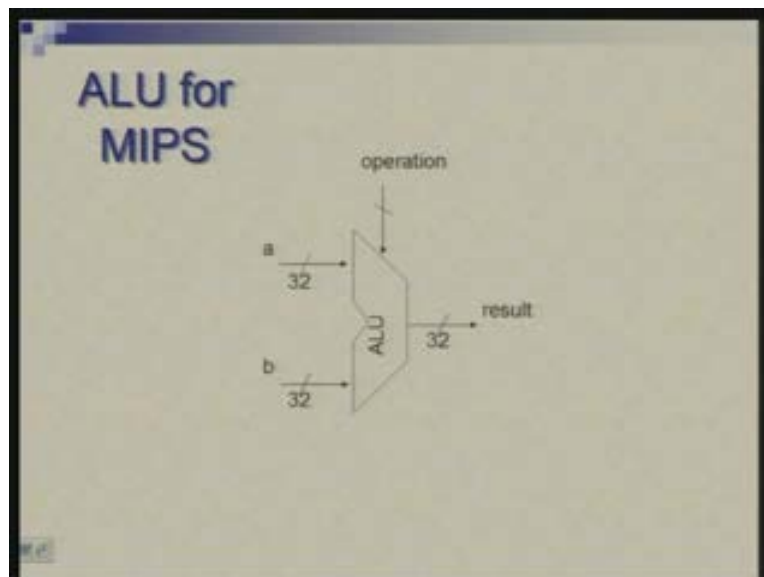
(Refer Slide Time: 6:15)

Performance

- The two main measure of performance are
 - execution time and throughput
- The time to execute a given program:
 - $\text{CPU time} = \text{Instr count} \times \text{CPI} \times \text{clock period}$
 - $\text{CPU time} = \text{Instr count} \times \text{CPI} / \text{clock rate}$
- Affected by compiler, ISA, μA , technology.
- When trying to improve performance, look at what occurs frequently => make the common case fast.

So, when we try to improve performance it is important that we look at what is the common case and try to make that fast; if you cannot make everything fast at least make the common case fast. So now, having studied a few instructions from programming point of view and developing some understanding of what performance means, we had set out to design the ALU first and then the complete processor.

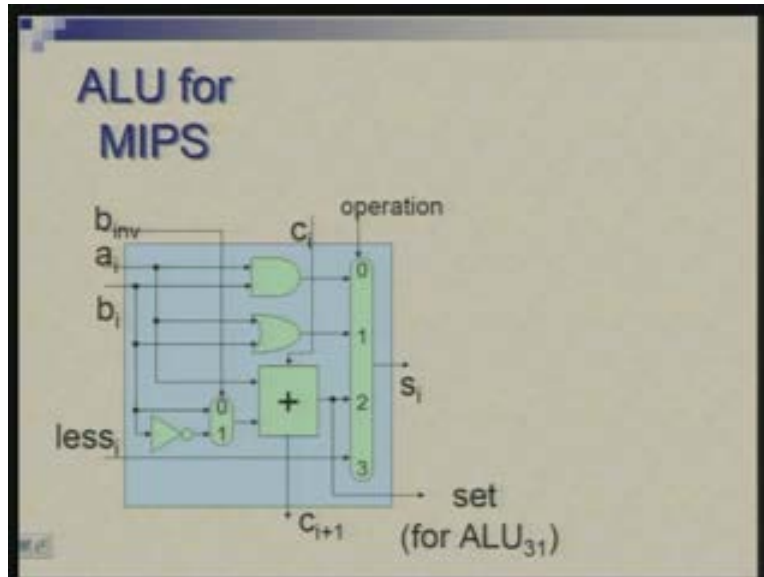
(Refer Slide Time: 06:49)



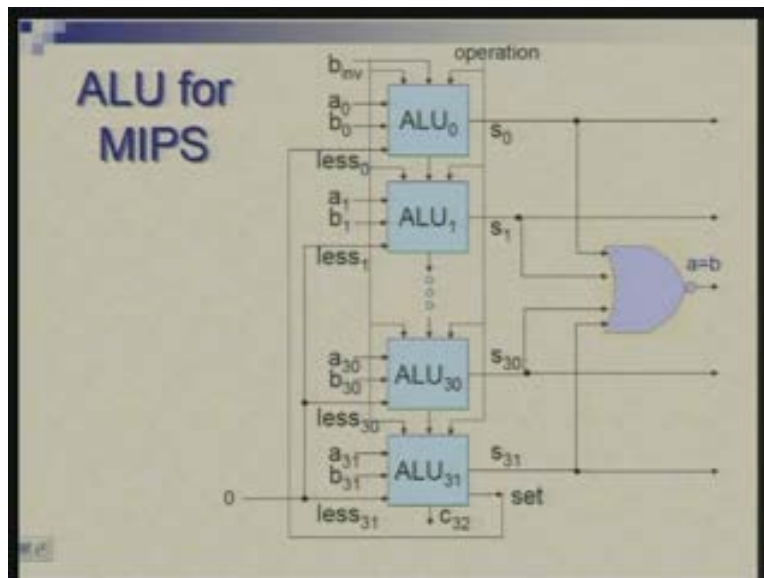
So ALU is a heart of the processor which carries out all arithmetic and logical operations and its design was done by looking at one cell which can be repeated for every bit and the construction was done by simply looking at various functions it has to perform, put

hardware in place for each of these and then have a multiplexer to select one of the outcomes. So effectively it does all the operations simultaneously but you look at the result of only one of these.

(Refer Slide Time: 7:00)

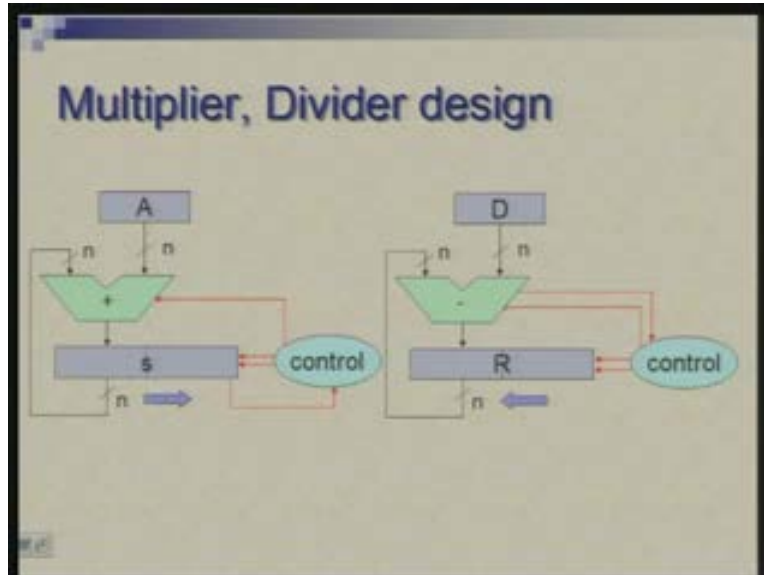


(Refer Slide Time: 07:25)



And this is repeated thirty-two times; the additional logic shown here which does the equality comparison so this is the ALU we had worked with. While doing so we had ignored somewhat complex operations like multiplier and divider for which we primarily discussed iterative algorithms which take thirty-two steps and in each step try to produce one bit of the result.

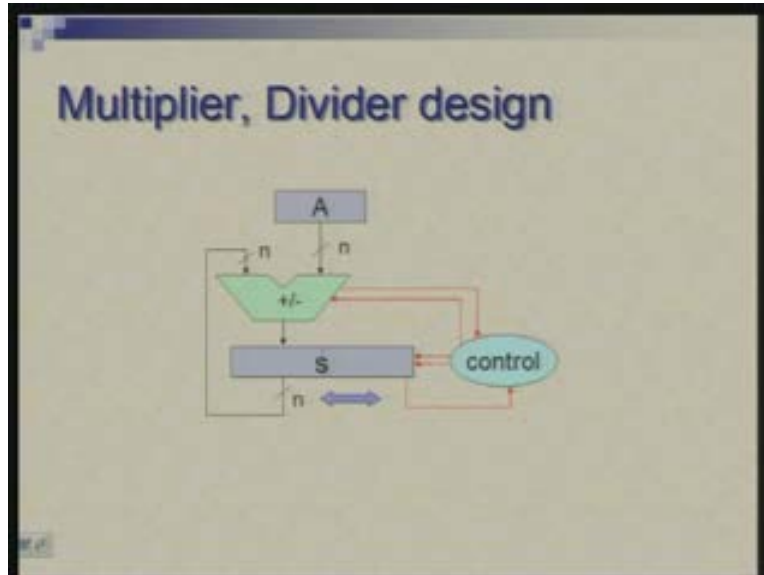
(Refer Slide Time: 08:02)



So this is a multiplier design; this is one of the divider design we had and you can notice similarity of the structure here. We had lot of variations. For example, division was done in a restoring manner, non-restoring manner so this is a non- restoring division.

It is usually possible to merge the multiplier hardware and divider hardware. If you look at the commonality there is lot in common. The key difference is that the big register at the bottom shifts right in one case and left in other case. So if you make it possible to shift it both ways and organize the control appropriately then make this (Refer Slide Time: 8:49) and this unit is capable of doing addition or subtraction both so you can have a single unit which does multiplication or division depending upon how you exercise the control.

(Refer Slide Time: 08:55)



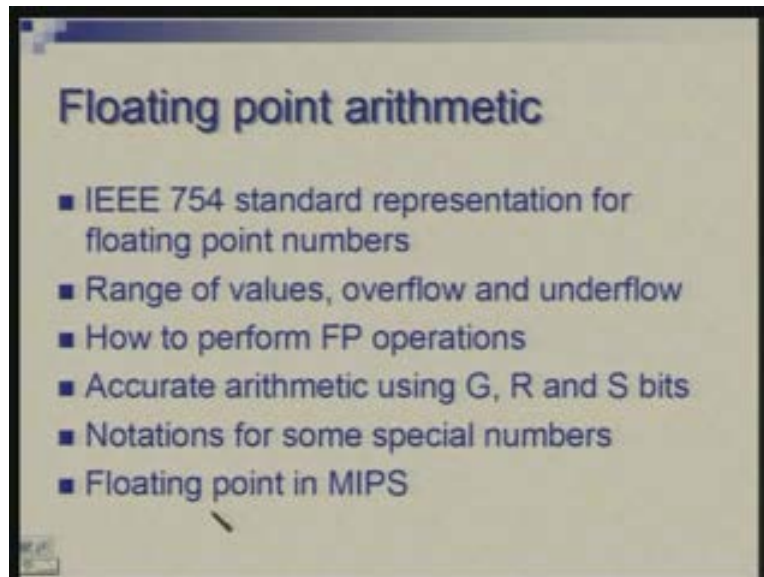
(Refer Slide Time: 09:00)

Speeding up arithmetic

- Speeding up addition
 - Ripple carry adder (carry propagate adder): $O(n)$
 - Carry look ahead: $O(\log n)$
- Speeding up array multiplier
 - Using carry propagate adders: $\sim 3 n d$
 - Using carry save adders: $\sim 2 n d$
(here d = delay of 1 bit adder)

So this design, we saw how we could speed it up. **You could** For speeding up addition which you did earlier you could use carry look ahead technique which takes from time of the order of $O(n)$ to $O(\log n)$ and that is a tremendous improvement. To speed up multiplication we looked at the possibility of using carry save adders in an array form. So instead of doing iteratively you expand it out into adders which add all the partial products and use carry save additions so that improves the delay by a significant factor.

(Refer Slide Time: 9:50)

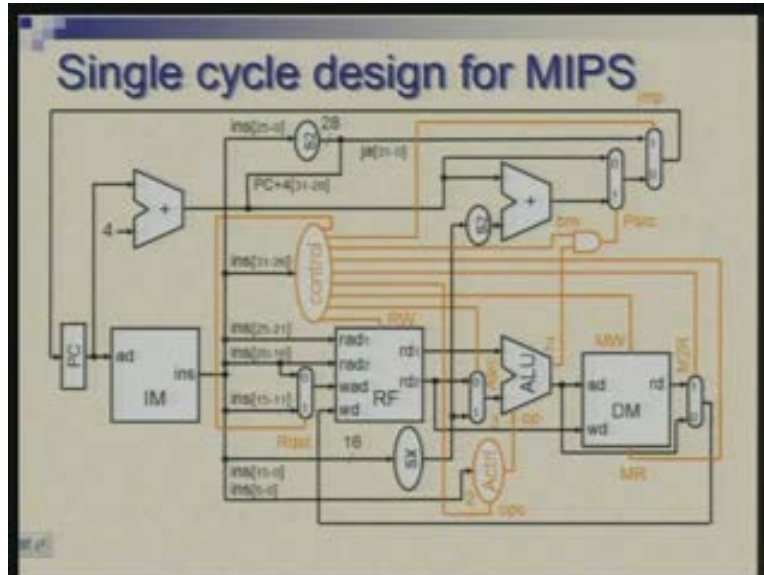


Then, noticing the limitation of integer operation we talked about floating point operation. There are varieties of ways in which you can represent numbers and our focus was on a standard method which is commonly followed now that is IEEE 754 standard. There are issues of how you detect overflow and underflow. So underflow is an additional feature which occurs in floating point. We saw how these operations are performed. We also looked at the need for accurate arithmetic with the help of guard, round and sticky bits.

The notation provides for some special numbers such as de-normalized numbers, infinity and something which is indeterminate and so on and we had a brief look at how floating point operations are done; anyways what are the instructions which support this type of arithmetic.

Then, having understood how basic operations are done, we looked at the entire process of instruction execution starting from fetch, fetching up of instruction, picking up the operands, doing the operation and then putting the result back in place.

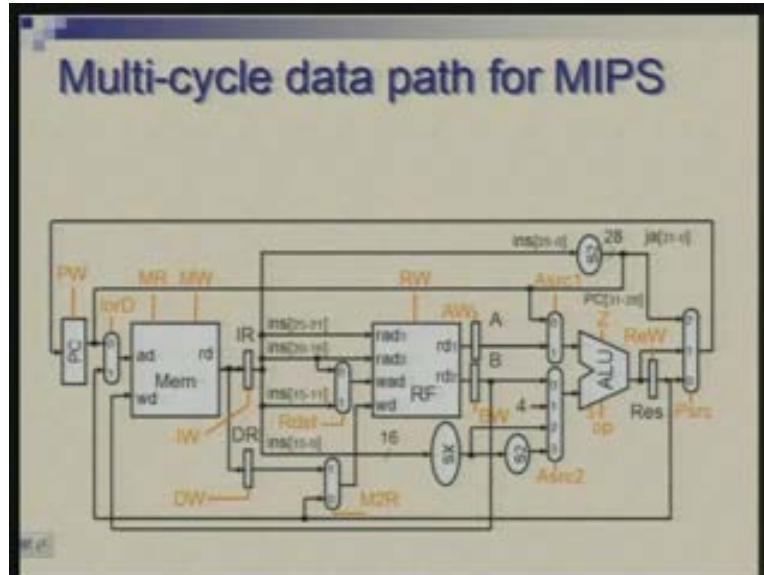
(Refer Slide Time: 11:11)



The first design we looked at was a single cycle design where the entire instruction execution is finished in a single cycle and the control here is a simple combinational circuit which looks at opcode and associated function bits and tries to generate control of various components within the data path. So what you are seeing in black and white is the data path and what you are seeing in orange is the control. Hence, when you design you have a clear distinction made between the data and the control which is the task substantially.

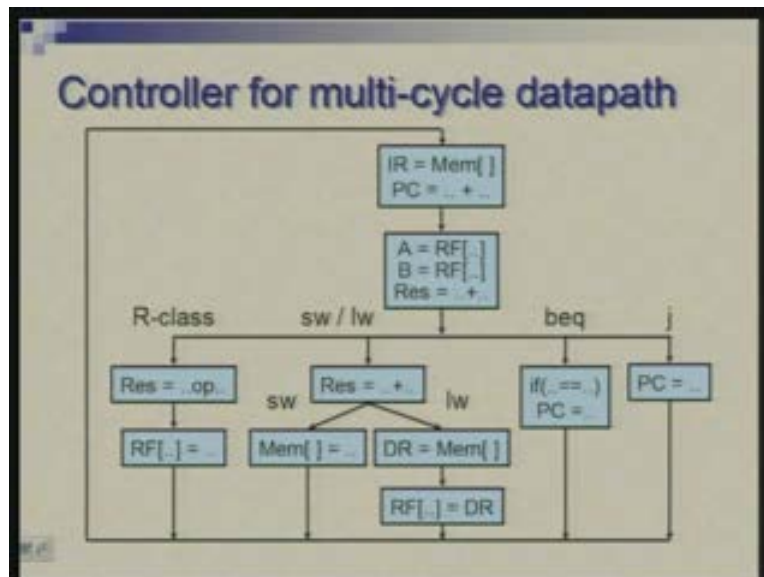
So, having designed this we notice the limitations of such a design. Various points of view and went on to a design which is multi-cycle design. So that is to improve the performance; helps in sharing the resources and also makes it possible to implement a much larger variety of instruction.

(Refer Slide Time: 11:45)



So in this the controls..... although not shown here; is a sequential circuit which steps through various states and takes the instruction through various cycles.

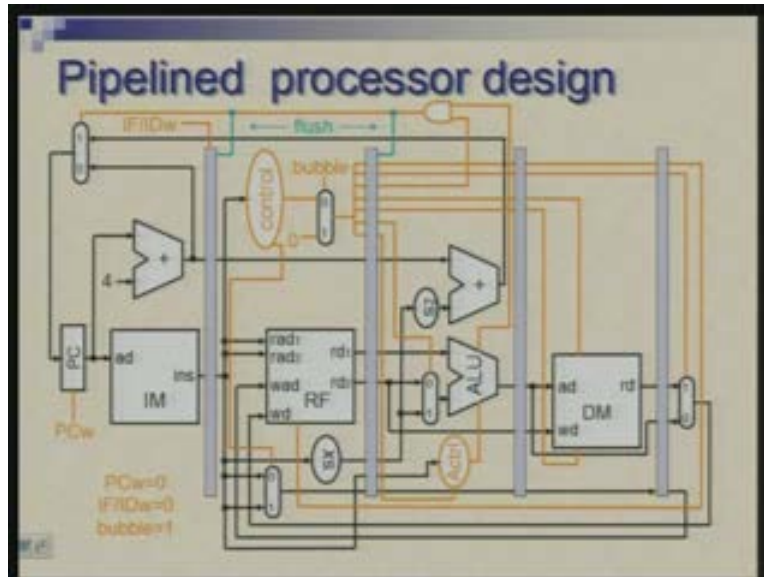
(Refer Slide Time: 12:29)



So this flow chart actually indicates how various instructions are done in multiple cycles. Each box represents activity which is done in one clock cycle. So now here it is very crucial to see how instruction execution is divided into cycles because **that determines** that influences both the CPI. So CPI is very obvious, for example, if there is an instruction which is following let us say this path it takes five cycles; **you have five** you go through five boxes but what you have put in these boxes determines how wide the

clock has to be so clock period as to accommodate the slowest of the activity which you have put in that box. So you have divide these things, distribute these things in a uniform manner as far as possible.

(Refer Slide Time: 13:24)

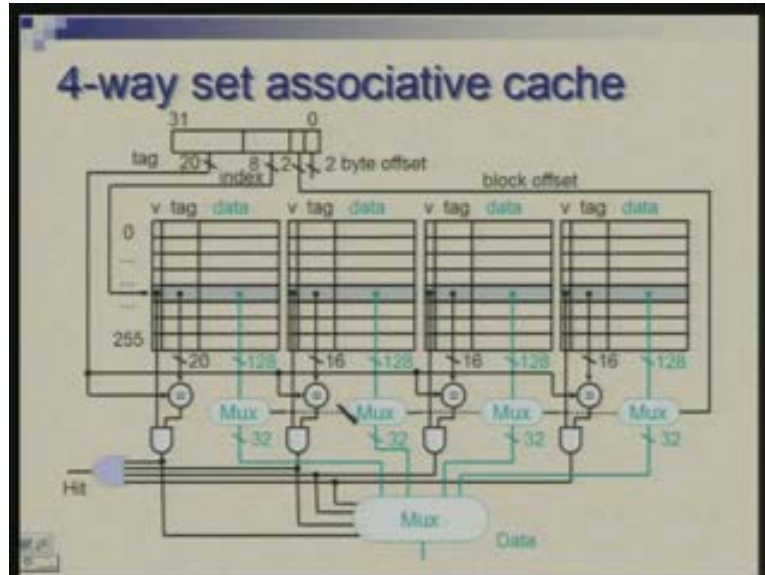


Now, going further we looked at pipelining of the design where each individual instruction take multiple cycles but instructions get issued hopefully every cycle. And this design was obtained in a very simple manner by looking at the single cycle design and introducing registers which separate pipeline stages. So in terms of concept it is very simple but we immediately notice some hazards: structural hazards, data hazards and control hazards.

Structural hazards are relatively simpler to handle. You try to provide enough resources so that there are no conflicts occur but data and control hazards have quite inherent partly in the hardware which you design and partly in the way program logic is there. So one needs to provide for extra control to take care of these things.

For example, for flushing out instructions which are fetched when you are trying to take a **branch** decision. So the wrong instruction may get fetched and you have to flush them later on. For handling data hazards you try to reduce the delay by forwarding data so that the instruction which is dependent and looking for some data gets it as early as possible. The data forwarding paths are not shown here but they are a number of possible paths so that data flow from instruction to instruction can be done as fast as possible.

(Refer Slide Time: 15:08)



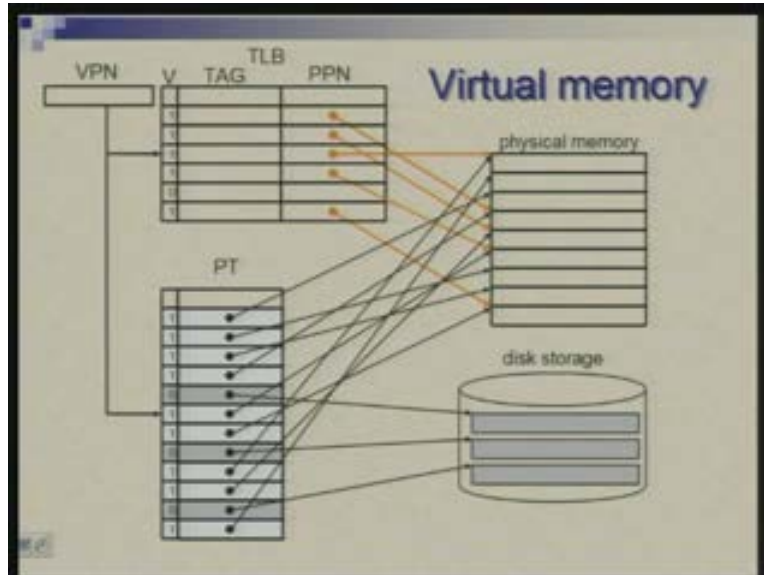
After having spent several lectures on processor design we went on to discussion on memory which was seen to be as a hierarchical structure as **na** flat structure and the basic idea there was to try to get advantage of fast SRAM and try to get the bulk of inexpensive and-dense DRAM at one end and also taking it further get the advantage of size of what you can get on a disk at an inexpensive manner.

So a typical cache organization is shown here (Refer Slide Time: 15:54). This one in particular shows a 4-way set associative cache. We looked at three organizations: direct map cache, set associative cache and fully associative cache. So, most commonly what is used in cache is set associative cache with varying degree of associativity. this one is 4-way associative in the sense that a given block of data or instructions can be placed in one of the four alternative slots and this number 4 gives you the kind of flexibility you have in placing a block; the larger the number, larger the flexibility. A larger number also implies more hardware cost because you need to look at more places in an associative manner or by doing parallel comparison. But this parallel comparison and associative search also has an effect on the clock periods. So typically as we increase the degree of associativity the clock period becomes larger and larger which means there is advantage on the impact of performance.

The positive impact on performance is that by giving more flexibility you reduce the number of misses; you are unlikely to throw away data which you would have brought earlier but are throwing away because of conflict because that space is required by something else.

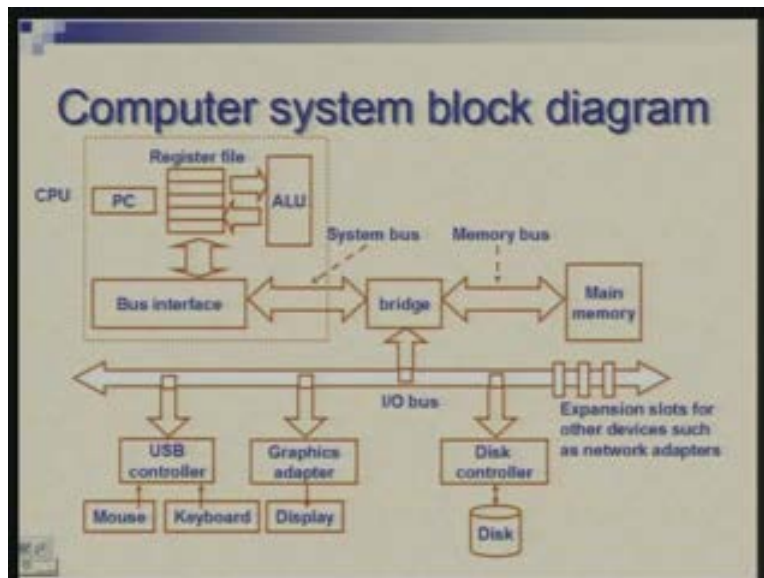
The other level which is commonly used, other level of hierarchy is virtual memory where main memory forms one level and the disk stores the next level.

(Refer Slide Time: 17:44)



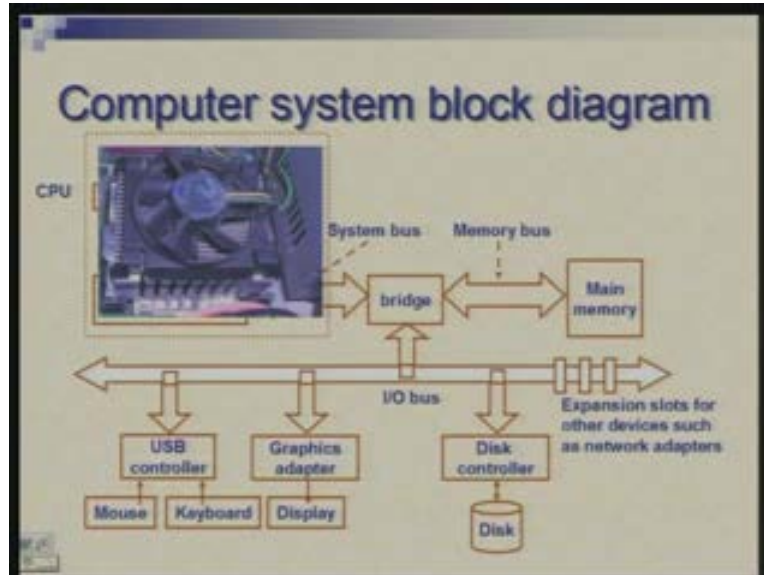
So mapping here is done through a page table which also is in the memory either in physical or in virtual. To speed up this process of accessing virtual memory we have an associative buffer called TLB or Translation Lookaside Buffer.

(Refer Slide Time: 18:08)



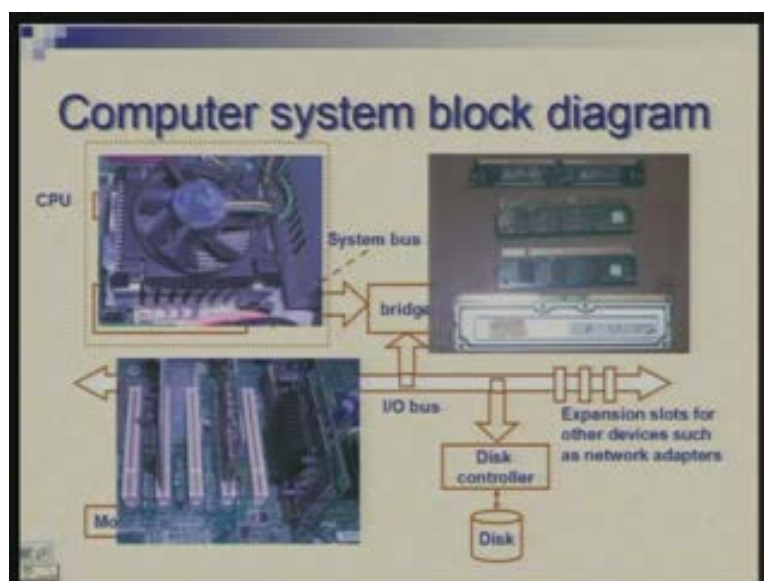
Finally we moved on to I/O. So putting I/O together you form the complete system and this is a diagram which we projected in the first lecture. You have various parts here; (Refer Slide Time: 18:26) there is a processor, there is a memory, there is a bus and there is an I/O subsystem. So, physically you can relate it to what you see if you open a computer.

(Refer Slide Time: 18:38)



So this is a processor with the heat sink and fan; you do not actually see the processor; it is covered with heat sink and a fan on top of it. These are memory modules (Refer Slide Time: 18:51); these are PCI slot which is a bus we had seen and some of these slots have been plugged in. this probably is the network card and I think this takes care of I am not sure maybe keyboard and mouse. So this one is actually not on PCI this is a Graphics Display Card.

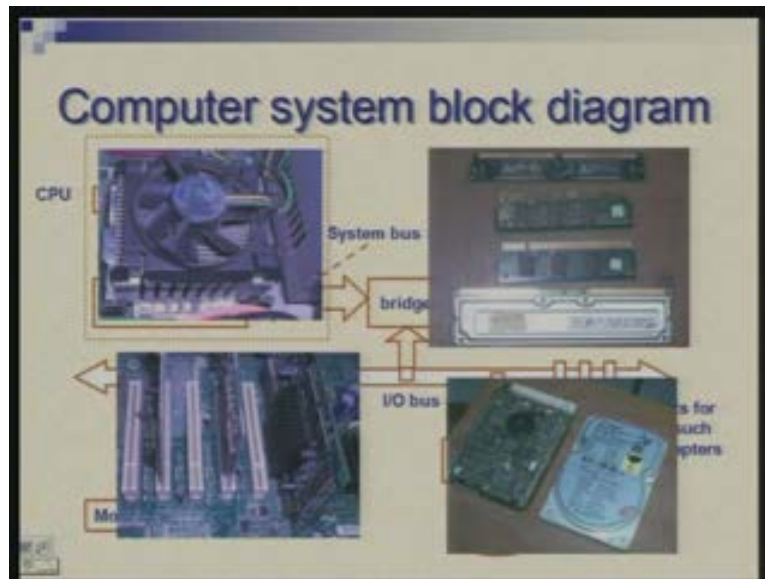
(Refer Slide Time: 19:03)



So, some of these things could often be on the board. This is not a very recent board, it will be some two three years so what you are seeing here as a separate card could also be

part of motherboard. But as we have seen that the graphics display is the most demanding in terms of performance and it is connected as close as possible to the processor memory interconnection whereas other things might sit on backplane bus or an I/O bus.

(Refer Slide Time: 20:06)



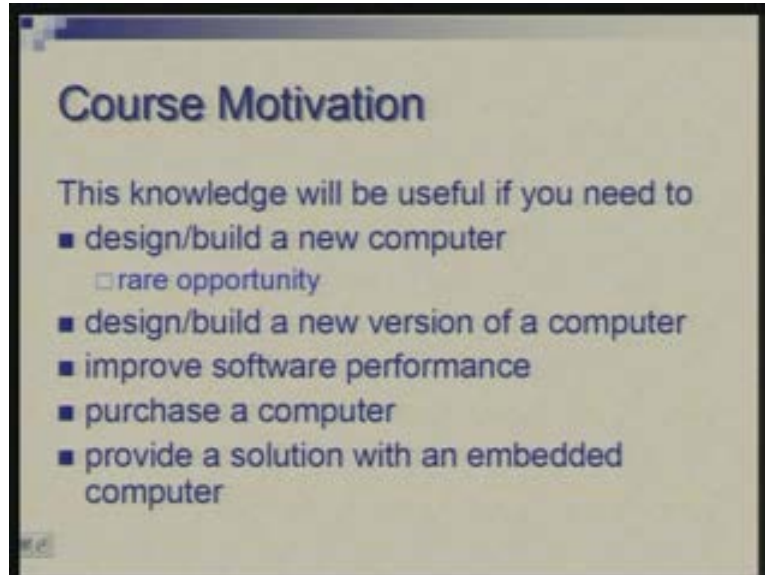
So these are I/O devices, this is disk drive.

(Refer Slide Time: 20:17)



This is a complete motherboard.

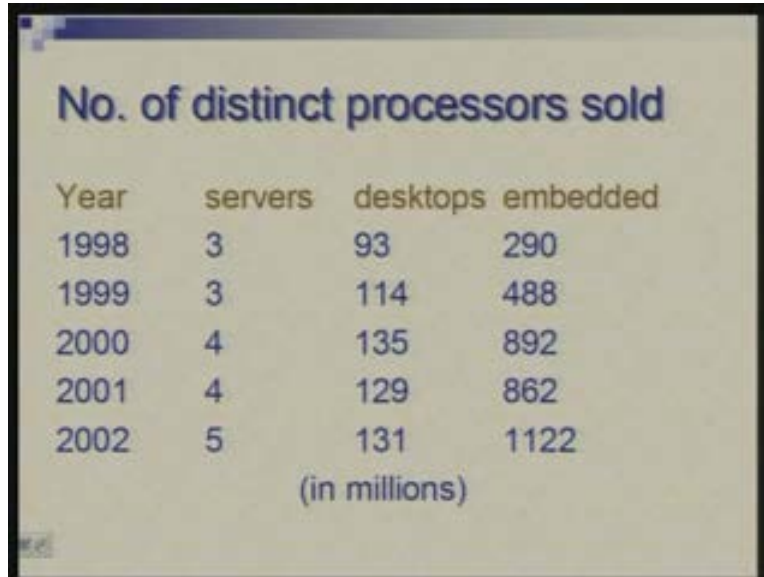
(Refer Slide Time: 20:30)



Now, why have we studied all this? These are the points of motivation which we had looked at. So, ideal thing would be that you get an opportunity to design a new architecture, new instruction set, build the processor and so on but practically speaking that would be a very rare opportunity not many new processor designs takes place. More common, a more likely situation is that you might design or build a new version of an old architecture or from software point of view this knowledge could be useful to write better software either at application level, if you have an understanding of what architectural compulsions are you can actually write better software or if you are in compiler operating system area again then of course a deep understanding of architecture is very very important.

Even if you are not doing a technical development of hardware software, understanding of architecture and performance issues is important when you are for example purchasing a computer. So now all these points were with desktop computing or general purpose computing in mind. But if you look at embedded computers then you might be doing much more of this, there are **much more** many more opportunities, it could be basically designing; you might end up in designing a new processor or actually building a complete solution out of a processor used as a component. So it is interesting at this stage to look at the statistics here about how many processors of various kinds typically get sold every year.

(Refer Slide Time: 22:30)



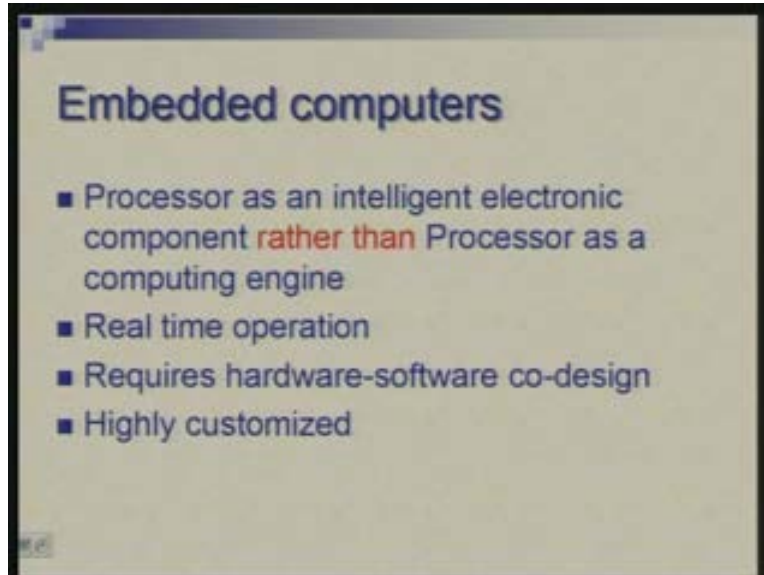
Year	servers	desktops	embedded
1998	3	93	290
1999	3	114	488
2000	4	135	892
2001	4	129	862
2002	5	131	1122

(in millions)

There are three classes of systems are put here in this table: servers, desktops and embedded processors. So the figures are in millions and they are given year wise; over last over five consecutive years. There are several observations you can make. One observation is that the volume is varying by orders as you go from server to desktop to embedded the numbers here are many more; the second observation is that while servers are growing very very slowly the desktop appear to be almost saturating but embedded if you look at this only as an ((.....23:25)) but if you broadly look at the curve there is a very fast growth.

So when you talk of desktops and you talk of availability of these machines per person, you will talk of very small numbers. For example, if you look at our own community here the number would be either 1 or may be little less than 1 or little more than 1 per person. But embedded could be much much larger. As I mentioned earlier you could have embedded processors in various other devices, various appliances, your mobile phone and so on so number you can conceive of per person could easily go to double digits. So what are these embedded computers?

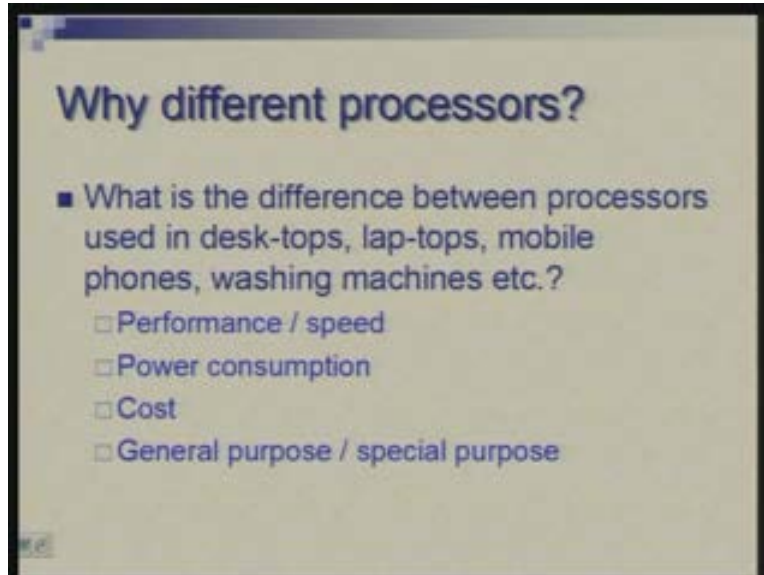
(Refer Slide Time: 24:25)



Basically here you have, these are processors which are not seen as a general purpose computing device. So they do some intelligent function but you do not have users sitting there and programming them so they typically do a fixed function or a fixed set of functions; they operate in real time, interact with the environment and the design process is more involved. We have not discussed that. Some of the courses which you might do as ((.....24:56)) later on would address that issue.

These typically require one to address the hardware and software design issues simultaneously as a combined hardware software co-design problem. These are very size, rate, power and cost sensitive and therefore they have to be highly customized. So I see tremendous opportunities of doing some interesting work in this area where the knowledge of this course would definitely be useful.

(Refer Slide Time: 25:33)



The question is why different processors; why do you need different processors for different types of things. We talked of servers, desktop, embedded. What is the difference between processors used in desktops, laptops, mobile phones and appliances like washing machines etc?

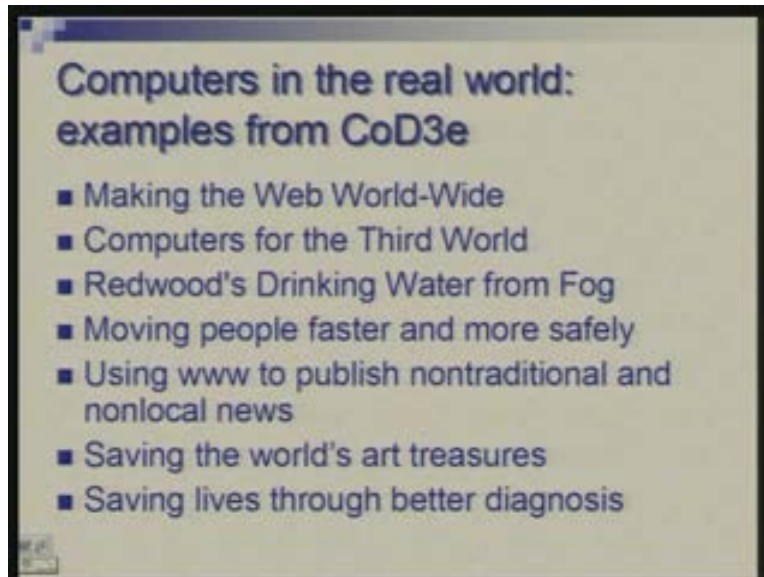
The difference comes in various aspects in performance or speed; difference in power consumption, cost and whether processor is for a general purpose operation or it has to do a special purpose.

(Refer Slide Time: 26:11)



This is a picture which I have shown earlier that shows the Pentium processor and a micro-controller. This is an 8-bit processor; it is actually a processor plus memory, the program could be stored in the memory which is there and you would typically store a fixed program and it could repeatedly just execute that.

(Refer Slide Time: 26:43)



I looked at the third edition of the textbooks we have been following; it talks of lots of real life examples where computers are used. In most cases these are embedded computers; sometimes they are augmented by general purpose computing also. So we will look some of these briefly and I will encourage you to see these interesting possibilities. I will go through some of these and towards end add a few from my own personal experience. Let us look at some of these. This talks of an attempt to take computers to masses. So basically a low cost PC which has been designed by a group of people for villages, the idea here was to make it portable in low cost so that affordability for people with small means could be there.

(Refer Slide Time: 28:11)

Making the Web World-Wide



- \$400 PC devised by Lee Thorn, the head of the Jhai Foundation, for Lao villagers
- No moving / delicate parts
- Flash-memory instead of a hard disk, LCD screen, 486-type processor
- Can be powered by a car battery charged with bicycle cranks
- Wireless Internet cards connect to a solar-powered hilltop relay station
- Passes the signals on to a computer in town that is connected to the Lao phone system and to the Internet
- The Linux-based software that will run the computers is in the final stages of being "localised" into Lao

It has no moving parts and it is rugged, there are no delicate things like CRT or hard disks. It uses flash memory instead of hard disk, LCD screen and an inexpensive Intel 486 processor which is couple of generations behind. But since those are available **in cheap** with very low cost that was used, powered by a car battery which can be charged by a bicycle; wireless internet cards which connect to a solar powered hilltop relay station and it connects to the telephone system and also to the internet. The OS is Linux and it is being localized in the sense that in the local language its communication is in local language so that people who do not know English can also use it.

(Refer Slide Time: 29:44)

Computers for the Third World

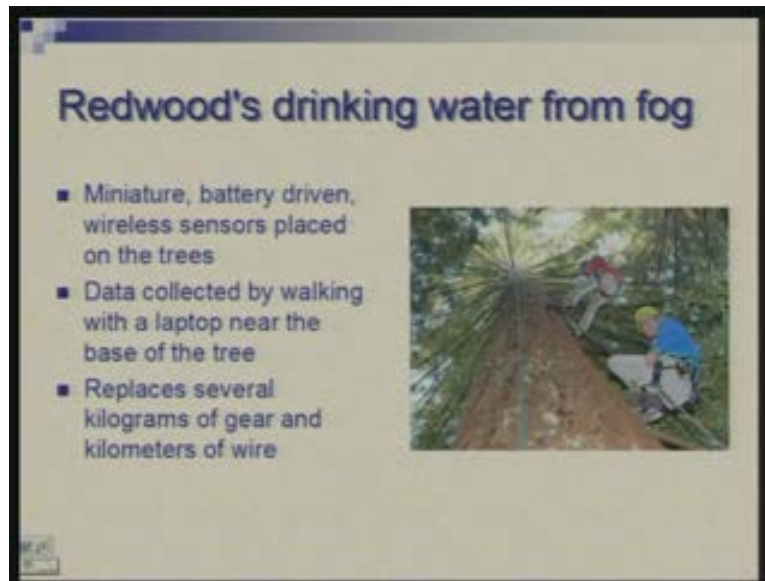
- Simputer (simple, inexpensive, multilingual computer) designed for villagers
- price - \$250 to \$300
- conceived by a team of computer scientists at the IISc, Bangalore
- Intel Strong-ARM processor (low power consumption)
- 64 MB RAM, 32 MB flash
- modem, Linux operating system
- touch-sensitive screen, no keyboard, text into speech.



Another attempt is from Bangalore. You must have heard of Simputer. How many have heard of Simputer?

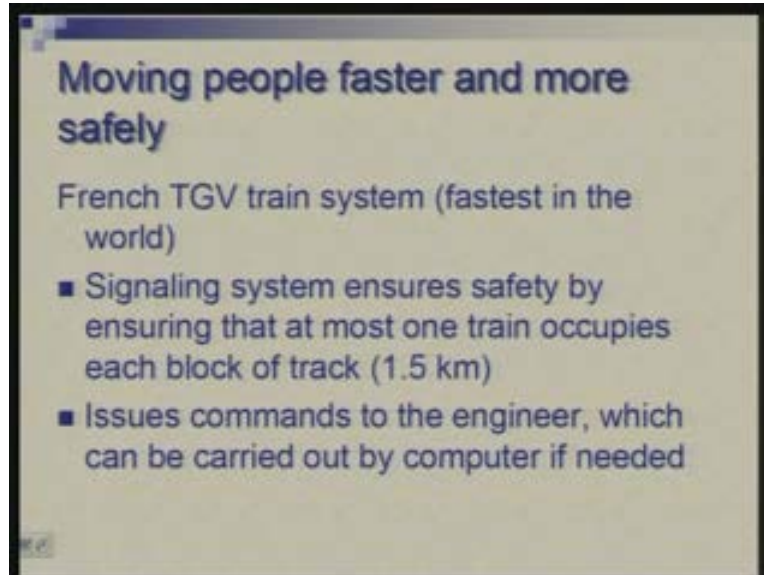
So you would recognize this. Let us say, again it is an attempt to build a small low cost computer and this was actually initiated by a team in IISc Bangalore; a startup which came out of IISc Bangalore. The processor used here is strong arm which has low power consumption; again it is flash based, there is no hard disk, no moving part, this plays LCD, it has a modem, Linux OS and input output, there is no keyboard this is a touch pad so you can either work through icons or it also has text to speech capability so that..... there is no speech input but speech output is there so even if you cannot read text for an illiterate person that could be a reasonable medium of communication. So the information coming from the computer can be spoken out and the input could be by touching the icon; you can look at the pictures and convey the input as you can see what is happening here.

(Refer Slide Time: 31:06)



This is an example of where computers and modern technology can help in carrying out some scientific experiments. What you are seeing here is a redwood tree and some scientists wanted to study the biological phenomenon here for example how these trees becomes so tall. These are really joint trees and they wanted to study whether these trees absorb moisture from atmosphere from fog. So, initial attempts were to place lot of heavy equipment on this with lot of cables running around but then subsequently it was replaced by some simple sensors, miniature battery driven wireless sensors which were placed on trees and because they are wireless the scientists could move around with laptop near the base and the data which is collected could be simply communicated without having to climb or without running huge cables.

(Refer Slide Time: 32:11)



This is an example of computers trying to bring safety into automobiles or in transportation system. So the example is of a French TGV train system which is the fastest system and the key to safety here is the signaling system which is all computer based. What it tries to do is that the entire track system is divided into segments or blocks of size 1.5 kilometer so at places they are even shorter. And essentially sensors and computers and the signaling mechanism tries to ensure that in each block or in each section there is at most one train because more than one would mean a disaster and the system issues commands to the driver who is supposed to continue or stop and so on but there is also a computer there which can take the commands and act if necessary if the human operator fails to listen.

So let me go back to this list and mention some of the other examples. I am not putting the details here. So this is an example of news medium which is which is different from traditional TV or newspaper medium. So basically from web sources the news could be collected and published automatically and that crosses the typical regional boundaries and regional biases so it could be a totally different nontraditional nonlocal source of news.

The next one is about how computers can help in preserving the old pieces of art by using image processing techniques. And the last one is use of computers in diagnosis through MRI and such techniques.

I want to add at the end some interesting applications we have been involved in developing.

(Refer Slide Time: 35:02)



Together with the startup **we have** you must have heard of critical solutions which is in our **business incubation** unit. So these are pictures of device for scanning underbelly of a car or any vehicle in general for security. The security is becoming an increasingly important concern because of terrorism there. So one would like to prevent some dangerous thing being attached to the bottom of a car and then the car being brought into a premise and it might explode later on. So you need to scan the underbelly of the car. The traditional approach is to use a stick and a mirror; you might have seen that security people would carry a mirror and then try to do that. The problem with that is the illumination is very poor, it is generally dark, there is not I mean one goes through the drill but does not really help you much and at any given time you see only a very small part of a car so exhaustive checking is never possible.

So the idea here is to have basically a camera based system. What you are seeing in this picture are four units. each unit these bright dots you see are LED displays LED sources which emit light and in between you are not seeing very clearly but there is a digital camera video camera sitting there (Refer Slide Time: 36:41) so these lights illuminate..... this is a device which is placed in a pit like this and as vehicle drives over it the bottom gets illuminated and cameras take picture. But what happens is that because the gap between the camera and the bottom of the car is not too much each camera gets only a limited field of view it sees only part of the car. So, as car is passing a camera would see only a small part and each camera actually captures a video of a strip of the car.

Now these video signals, you can see some cabling here, these are sent to a computer which is somewhere in the side in a control room and each video first of all, each frame of the video is taken together and you generate a single picture you stitch these together, it requires sophisticated computer vision techniques to form a strip of image and then these strips are put together to form a complete image. So what you get is like this and **if you care** if you see carefully you might be able to see a boundary of (Refer Slide Time:

38:02) although I have shown here but this image was taken with 3 so the whole image is divided into three strips actually but the stitching is done nicely so that **you do not really** unless you are careful you do not see the boundaries.

So you could see the role of a computer, it is a dedicated application where a computer is performing single but a fairly sophisticated task of creating this image out of three videos.

(Refer Slide Time: 38:38)



Another example is to automate the kind of process you would have seen on the entry exit gates one like IIT. So here **if you** those who have driven cars across the gate you know that the cars which are authorized which are registered with our security they carry a sticker on the windscreen and the person who is driving is supposed to carry a card and **as you** as you pass through the gate the security man has to check the card, look at the identification number in the card, identification number on that sticker which you have and see if the right person is driving the right car.

So now it is a workable system in principle, but what happens in practice is that it is impossible to do it thoroughly. So you know, once you start this you put this kind of process in place it works right nicely for a few days and then after that the security may say okay go. There is **no time** no patience in people who are driving to wait and have it checked and also no patience in the security people to really go through it thoroughly. So here is an attempt to do this automatically.

What is required is that you have a device installed in the car which carries a small computer. In fact it is a variation of what I showed you here; not in this packaging but in a different packaging. This has what is called an EPROM Erasable and Programmable Read Only Memory. We have used a version of this which uses flash. But basically it is the same processor with the same instruction. So, a processor is sitting there which

communicates with this pair of devices which is installed on a pole near the gate. So this is connected to a PC and basically the main thing here is an infrared source an infrared sender and receiver. This also is capable of sending and receiving infrared signals. And instead of a card where the number is printed which carries my identity as a driver I would be carrying a smart card.

I would probably have one in my pocket; I will show you how it looks like.

(Refer Slide Time: 41:55)



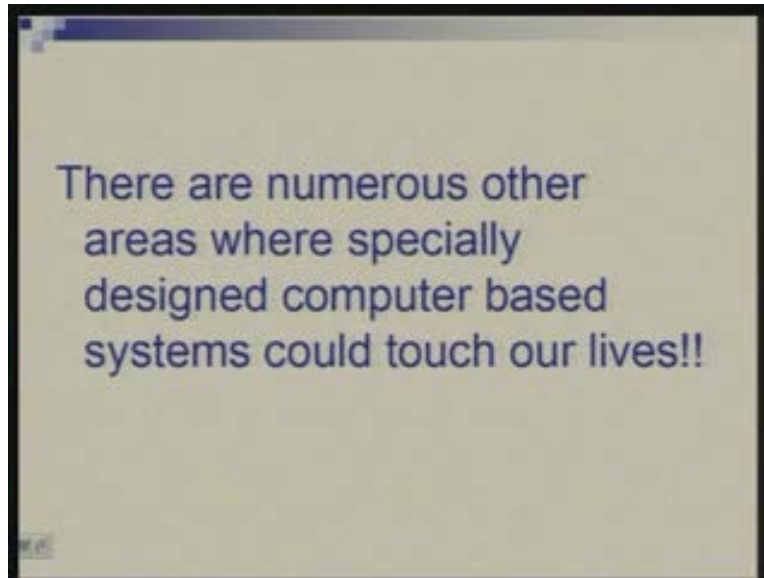
This is a smart card and what you see, this golden thing is basically a chip which has a tiny processor and some memory in it, rest of it just plastic to make it a credit card size thing. So basically this carries the identity of the person who is driving. Well, these cards are used in a variety of applications so they could carry for example electronic cash also. Electronic cash is essentially nothing but a set of bits of 1s and 0s. So in this case this carries identity of the person who is driving the car and this identity is also known to the control computer which is connected to the gate.

The identity of the car is actually embedded in the microprocessor which is there inside this device. So this has to be inserted there and before you pass through the gate you insert this and as you are passing through the gate this device which is on the gate will interrogate and seek the identity so identity of the car and the driver would be encrypted and communicated to this and then **the two** the two will be checked against a database which you have in the computer and then the computer can be used to either turn a light green or red or open a barrier. So there are interesting design issues which come up here.

Basically since this is a battery driven device (Refer Slide Time: 43:48) **the power consumption** reducing the power consumption becomes extremely important so you have to have this operation organized such that the power consumption is minimized. **I cannot**

go into details but of course since these are the last two hours of in-house development and those who are interested can always contact us and find out more.

(Refer Slide Time: 44:24)



So I will like to close at this note that there are numerous other areas where specially designed computer based system could touch our lives. They are very interesting areas, very challenging and there is lot of excitement which could be lying ahead of you, thank you.