Microprocessors and Microcontrollers

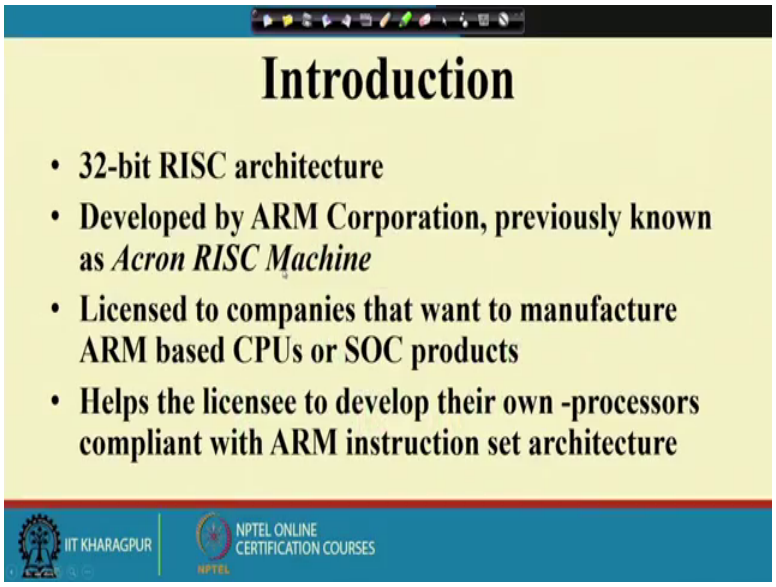
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 41
ARM

We will next look into another very advanced microcontroller which is known as ARM microcontroller and it is one of the very popular ones, because of the fact that many of the devices that you find today, many of the mobile phones, I think cameras and all that, for particularly the power is a concern. So, we want to go for low power operation. At the same time we want to achieve very high speed of operation. So, those cases, these ARM processors have become very popular. And we will see that it has got many interesting features that make this processor design very unique in that sense, and we can use it as per our requirement.

So, unlike say 8051 or 8085 or other processor that we have discussed, where this entire chip always have to use in case of ARM we will find that there are way outs, like if you do not need a certain portion of the device, so you can turn off those portions. So, in your chip you may not include those portions, and you can have a chip which is just catering to your requirement, but still following the philosophy of ARM.

(Refer Slide Time: 01:30)



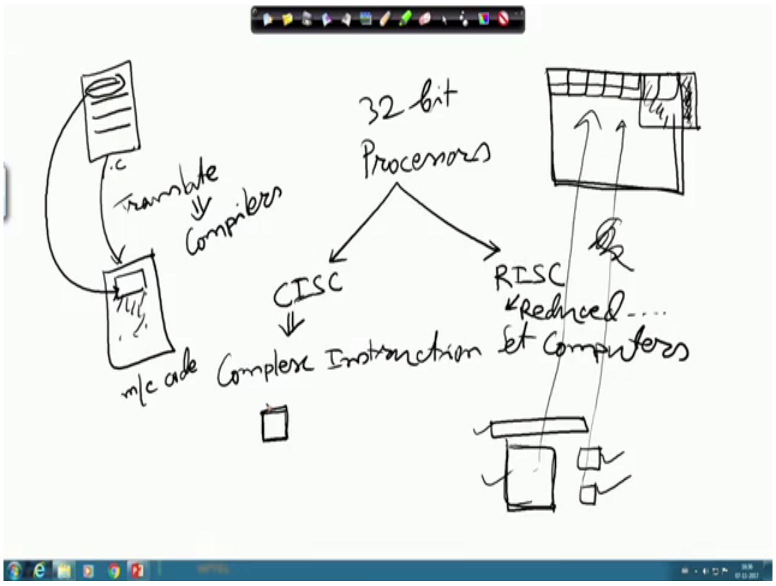
Introduction

- 32-bit RISC architecture
- Developed by ARM Corporation, previously known as *Acron RISC Machine*
- Licensed to companies that want to manufacture ARM based CPUs or SOC products
- Helps the licensee to develop their own -processors compliant with ARM instruction set architecture

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

So, we will look into this microcontroller. So, this ARM microcontroller is a 32 bit RISC architecture. So, RISC architecture means that reduced instruction set computers. So, its a 32 bit processor. So, this is already understood. Unlike say 8051 or 8085 which are 8 bit processors. So, if you look into this 32 bit processors. So the 32 bit processor means the internal data on which it operates, so it is 32 bit.

(Refer Slide Time: 01:58)



Now, we have got these processors. So, if you classify then there are two major classes. So, that on the basis of the instructions set that we have; one is called the CISC processor or complex instruction set computers and another is called the RISC processor or reduced instruction set computers. So, this is complex instruction set computers, and this one is the, RISC is the reduced instruction set computer. So, this is tends for the reduced instruction set computers.

So, what is the major difference between the two, the point is, that as a processor designer. So, you can design new processors, we can design advanced processors, you can think about the instructions that you have in that processor and those instructions may be very complex in nature, but where is this processor going to be utilized. So, this processor is going to be utilized, when you have written a program in some high level language. Suppose I have written a program in c language and then that c language program. So, I am translating and that translation leads me to the corresponding machine code, and in this machine code I have got the instructions given in the processor.

So, who is doing this translation? So, this translation is done by the tool called compilers, this is done by the tools called compilers. Now this compiler development process in some sense, you can see that every instruction that you have in this high level language program. So, that is getting convert it into a set of instructions into this machine language program. So, we can say that every instruction of this high level language statement, high level language program will get converted into a set of machine language programs, machine language statements.

So, given this job, so if in some sense I can say that the job is similar to problem like this, that suppose we are you are given an area, a rectangular area is given, and you are given some patterns. So, you are given some, say some rectangular patterns. So, maybe this pattern, then say you have got says this pattern, then you have got say this small pattern, so you are given, and a number of such patterns are given, and your job is to fill up this block using these patterns. So, using this pattern, so you have to fill up these blocks.

Now if I try to fill up then what will happen is. Suppose I put a big block here, then this part, this remaining part I may not find a suitable block. So, I put a block which is something like this, and then this is the wasted part and later on I will cut it off. So, in some sense I can say that if this big rectangle that I have drawn, so, if this is the program for which, I am trying to generate the code, and these templates that I have. So, they are considered to be the machine instructions, then as if I am trying to cover this my program logic by means of these rectangles, by these instructions, but some of the instructions are complex and some of the instructions are simple.

But I could not find an exact match. So, for this part of the calculation, I use this instruction, and maybe some of the operands of this instruction I make them zero. So, that this part does not have any meaning for my program. So, this way I can do it. So, essentially what is happening is that, these instructions I am using for that purpose.

Now, consider the other situation, where the same problem is given to you, but the blocks that you have, they are all of a single size and they are quite small, but they are of all single size. Then you can just do a very nice feeling like this, we can do a nice feeling this fashion, and it is very much likely that the amount of wastage that you will have will be much less compared to the previous case.

So, in case of RISC, this is what happens. Now so, in case of RISC, this individual instructions are like this boxes, and in all these boxes, they are similar in nature. So, all the instructions they are a similar in nature that, their sizes are more or less same, their time needed the execution times are more or less same. So, that way all these instructions they are of more or less same size and same duration.

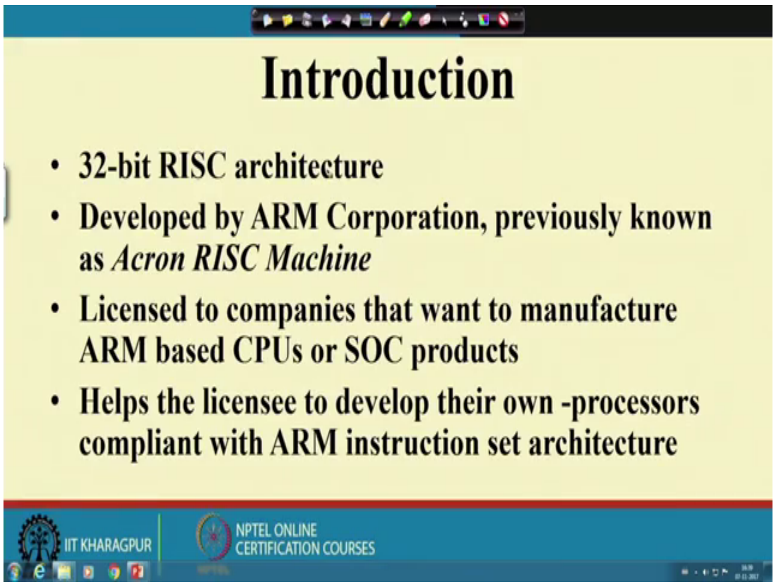
Unlike the CISC processors, where the instructions are of arbitrary sizes, these the sizes may vary. For example, if you look into say 8086 processor you will find that the move instruction itself has got sizes from 2 bytes to 6 bytes. So, as a result the execution, the fetching of that instruction itself will take different amount of time, if we look into the execution of instructions, there is some instructions; for example, which does a simple shift operation, that may take only three cycles, whereas, if you are taking the multiplication operation. So, that is going to take say 60 to 70 cycles. So, that way it depends, depending upon the instruction the execution time and the size will vary.

So, can we do something better? So, previously this processor design team and the compiler design team. So, they used to work independently, and then this processor designers they used to come up with complex instructions, but in more many cases those complex instructions are not being used by the compiler designers. So, this RISC philosophy came from that point. So, where the compiler designers required the instructions to be of equal size, and these instructions to be of equal duration, so that we can very easily predict the cost of equation, cost of these instructions, and we can very easily predict the size of the resulting program, then the execution time for the resulting program. So, you can tell those things very easily.

So, this is the objective with which this RISC computer scheme and one contrasting difference between these CISC base is in terms of the instruction sizes. Another contrasting difference that you will find is in terms of the registers. In case of RISC processors, we will find that the processor has got large number of registers; unlike CISC. So, compared to CISC, these RISC processors have got more number of registers that helps in the instruction execution. As we have seen that if the operands of some instruction, they are available in the CPU registers then the execution will be the fastest.

So, that way this RISC processor, if it has large number of registers then the compiler designer can put a number of temporary variables onto those registers and make the system fast, make the design fast. So, this is what is the philosophy behind designing, this RISC processor.

(Refer Slide Time: 09:57)



Introduction

- 32-bit RISC architecture
- Developed by ARM Corporation, previously known as *Acron RISC Machine*
- Licensed to companies that want to manufacture ARM based CPUs or SOC products
- Helps the licensee to develop their own -processors compliant with ARM instruction set architecture

So, this ARM is a originally proposed to be a 32 bit RISC architecture, is developed by ARM corporation. Previously known as Acron RISC machine, it is licensed to companies that want to manufacture ARM based a CPUs or SOC products.

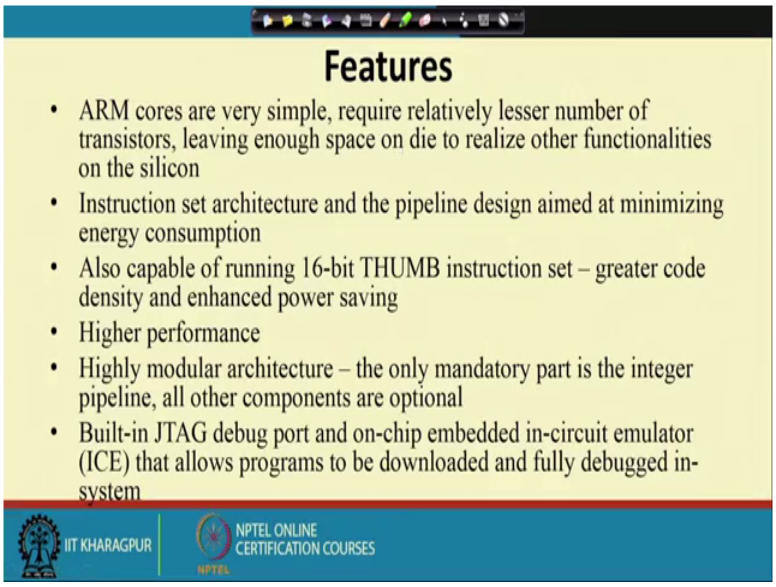
Now, this is another very interesting thing. So, you will never find any chip whose name is ARM, you will find chips like say 8051, 8085 numbered like that, but you will not find any chip which is named as ARM, because this ARM company, they do not manufacture any chip. What they do? They do the design and that design is licensed to various other companies, like you can find in XP, you can find say ST microelectronics STM series and all that, you can find different series, but none of them will be marked as ARM.

So, they have their own processor and they are modified to suit the requirement that they have. And so, the ARM based CPU designs we can, so the soft course are available in terms of the description. So, you can take that from the ARM, by paying the license fee, and then you can modify it as per your requirement. So, for system on chip type of design also, that it is very much useful; like in system on chip, what happens is that, unlike a board based design or PCB based design, where this memory, processor then other peripherals, that they are put as separate chips on the PCB in case of system on chip design, so entire system is put onto a single silicon wafer. So, that way all communications, they become on chip communication and the system operates at a higher rate.

But, for that matter you need to get the soft cores from different vendors and then we have to fabricate them on to the chip. So, this is good, one if you have got ARM type of platform, then if you can take the portions of ARM, which are meaningful for your case and you can discard the others. So, that way you can have good utility of the wafer space that you have. Unlike this board base design, where if you take say 8051 chip. So, the entire chip has to be there on the board. So, you do not have any other option.

So, this type of philosophy, this helps the licensee to develop their own processors complaint with the ARM instruction set architecture. So, first feature is the ARM cores are very simple, and they require relatively lesser number of transistors.

(Refer Slide Time: 12:37)



Features

- ARM cores are very simple, require relatively lesser number of transistors, leaving enough space on die to realize other functionalities on the silicon
- Instruction set architecture and the pipeline design aimed at minimizing energy consumption
- Also capable of running 16-bit THUMB instruction set – greater code density and enhanced power saving
- Higher performance
- Highly modular architecture – the only mandatory part is the integer pipeline, all other components are optional
- Built-in JTAG debug port and on-chip embedded in-circuit emulator (ICE) that allows programs to be downloaded and fully debugged in-system

So, the cores are made very simple. So, if it requires less number of transistors, means, as I said that in the overall silicon wafer you have got lot of space available for other components to be put into. So, you have got enough space to realize other functionalities on the silicon flow.

Now, instruction set architecture and pipeline design aimed at minimizing energy consumption. So, this is another important thing. So, instruction set architecture, like the instructions that you have, the registers and all that and the whole design is a pipeline design, and the overall goal is to minimize the energy consumption. So, this power is a very important concern, when we have got these embedded applications and that way this power minimization becomes an important issue. So, energy consumption, if we can minimize, then definitely it is helpful. Like today if we are buying a cell phone, a typical criteria for buying it is the size of the phone, and in that size. So, that size, one part is the display.

So, we want reasonably large display, but at the same time another part of it is the battery. So, you will find that the battery size is not reducing at the same rate at the other circuitry of the design. So, if you put a smaller size battery, means the system will not be able to run for a long time without recharge. So, if you can have this power consumption low; that means, you can have this battery size reduced or you can increase the time after which you recharge the device.

Also this ARM processor they are capable of 16 bit thumb instructions set. So, unlike other processors, ARM processor has got two instruction set; one is the, one instruction set is called ARM, which is a 32 bit instruction set. Another instruction set is called thumb which is a 16 bit inspection set. So, the major advantage that you gain, by having a 16 bit instruction set over a 32 bit is that, your memory is organized as 32 bit word.

So, if you are accessing the memory. So, in one axis you are getting two instructions now. So, that way it becomes, so the number of memory accesses can be reduced, it can be halved and in one axis you are getting two instructions. So, that is how the whole operation can be made faster, and these transitions on the buses will also be less.

So, you do not have to do so many memory accesses. So, this bus activity will also be low as a result, power consumption will also be low performance you can get higher performance, because these processors, we will see that there are many interesting design features that makes the processor faster.

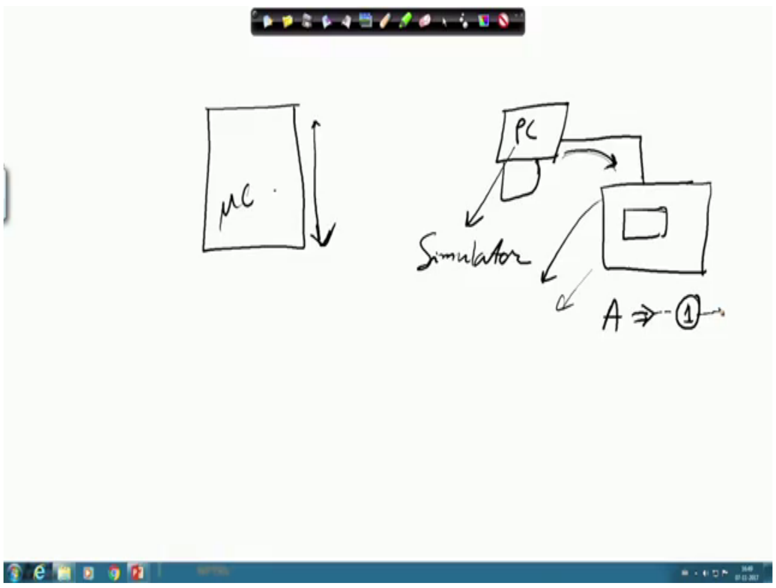
Modular architecture. So, modular architecture means that the entire design, it can be divided into sub modules. And as per your requirement, so you can tell that I need this module or I do not need that module. So, you can take that module. For example, you can find that the securities over this, ARM processor, the advanced version they even have got the security modules built into it. So, if you do not need the security modules for small embedded applications.

So, you do not put those security modules into your operation. Similarly there are modules that are responsible for signal processing job, the DSP algorithms will be running better here. So, for your case you may not be requiring the DSP algorithms to be running there. So, you can ignore those, the DSP sub modules.

So, that way it is modular architecture. So, you can select the set of modules that you need for your design only mandatory part is the integer pipeline. So, the part of the system that does the integer operations, so that part is mandatory. So, any integer instructions will be executed, so the addition, subtraction, multiplication, division with integers. So, that part will be there, others are all optional, so you may or may not use it.

Another very interesting feature that this ARM processor has, is the built in JTAG port. So, this built in JTAG port. So, this is used for in circuit emulation. So, let me just clarify this, what is it. Say suppose I am developing some application ok.

(Refer Slide Time: 17:30)



So, this is the application code that I have written in some microcontroller programming language I have written it, and this is developed sitting on a PC. So, this is a PC on which, I have developed my program there. So, this is a PC on which I have developed my program. And then, so this PC or this personal computer, it can have a simulator further, it can have a simulator for the microcontroller, and I can simulate this program and check whether this program is running correctly or not, whether it is giving with the desired values and all that.

After that, so normally the situation is like this, that there is a development platform, development board onto which we can download this compiled program, once we are satisfied with the simulation that my program is working correctly, so I can download this program and now on this, once the program has been downloaded, then I can take this system out and this can act as the stand alone system where. For example if I am doing a traffic light controller example. So, here after doing the simulation I will find, it is fine it is working file. So, I download this program via some downloading mechanism, and then I take this system to the road, and there I install it and try to run the traffic light controller program.

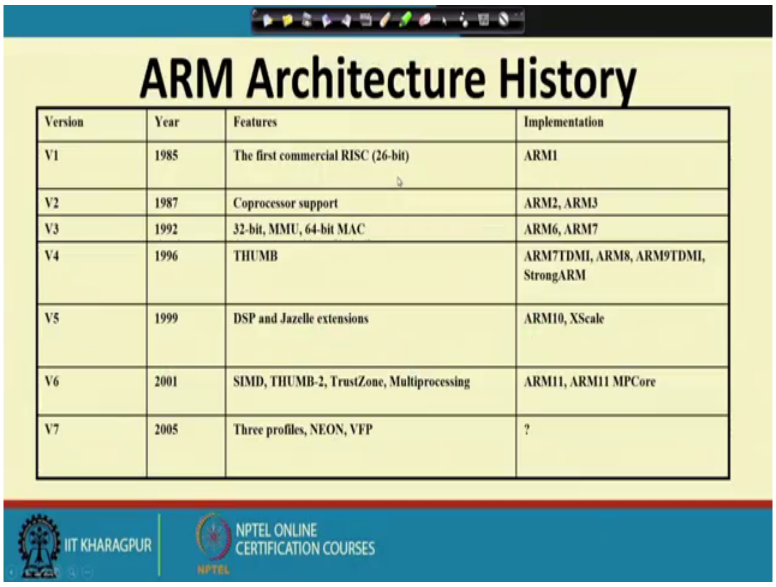
Now, the difficulty is that, while doing that, so the program, when I am simulating, so, it is working fine, but when I have downloaded onto the actual cheap actual, it say 8051 microcontroller, it may not be working correctly, why that microcontroller itself might has some fault developed. For example, it has got that a resistor, that A resistor may be one particular bit of it, say bit number 4 of it is permanently 1 or permanently 0. So, we cannot change that bit. So, that bit is always 1.

So, naturally that microcontroller will not be able to run this program correctly, because for the A register the bit is always, that particular bit is always 1. So, in simulation you cannot catch this error, because in simulation everything is running on that PC only. So, you cannot do anything, but when you are downloading it on to the target and then you are trying to see whether it is running correctly or not, you may find that it is not running correctly, and you need to find out the reason, like why is it not working correctly, and for that matter, if you suspect that the A register might be having some problem, you need to see the content of A register.

Now, how do you do that? There must be some mechanism, by which this PC will be able to talk to this processor now and it will be able to get the content of various registers within it, for check the value. So, this is exactly what is meant by this in circuit emulation sort of thing. So, there is a JTAG port. In fact, of many these advanced processors, they now have this JTAG port, that is for the debugging purpose, and we have got this in circuit emulator.

So, then you can, this will allow programs to download and fully debug in system, that way the system is operating, so you can take a snapshot of various registers through this mechanism, ICE mechanism and you can then try to analyze that what is the problem. So, this is very important feature when you are designing embedded application, using this, using some development platform like some PC.

(Refer Slide Time: 21:34)



ARM Architecture History

Version	Year	Features	Implementation
V1	1985	The first commercial RISC (26-bit)	ARM1
V2	1987	Coprocessor support	ARM2, ARM3
V3	1992	32-bit, MMU, 64-bit MAC	ARM6, ARM7
V4	1996	THUMB	ARM7TDMI, ARM8, ARM9TDMI, StrongARM
V5	1999	DSP and Jazelle extensions	ARM10, XScale
V6	2001	SIMD, THUMB-2, TrustZone, Multiprocessing	ARM11, ARM11 MPCore
V7	2005	Three profiles, NEON, VFP	?

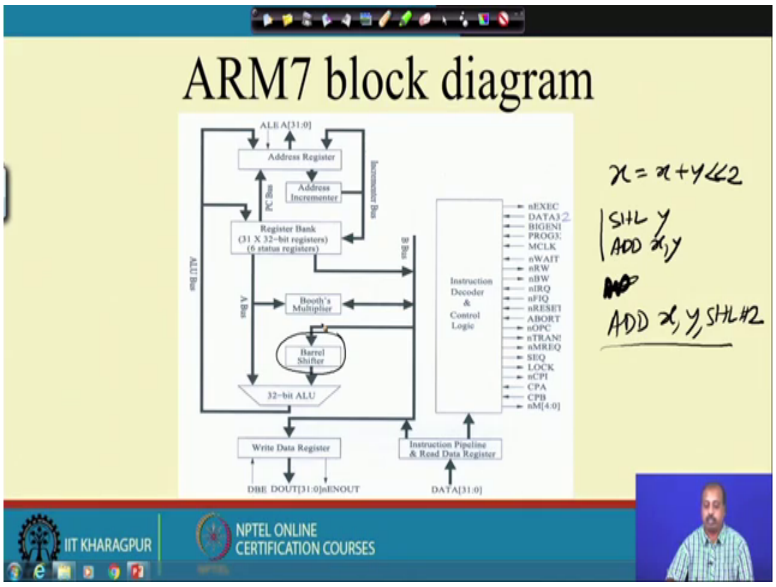
A quick look at the brief history of this ARM processor so, it started in 1985, as a 26 bit commercial RISC, first commercial RISC. So, 26 bit that was called ARM 1 in 1987 the version 2 came with the coprocessor support. So, that the processor. So, it was implemented in ARM 2, ARM 3, type of processors. Then in 1992 we get the version 3. So, that has 32 bit processor now. And there is a memory management unit, then there is a multiply accumulate unit. So, that has been introduced. So, that is useful for mainly for the signal processing applications, and the very implementations are ARM 6, ARM 7.

Then in 1996 the version 4 came. So, that is that had the thumb instruction set, and then the along with also we have got the thumb instruction set as well, and the implementations we have this ARM 7 TDMI, ARM 8, ARM 9, strong ARM. So, these are various implementations that are available for this that has got the thumb instruction set. 1999 we have got this DSP and Jazelle extensions. So, these are some additional co-processing facilities and available in ARM 10 processors.

Then, we have got this 2001 SIMD, Thumb 2, is another extended version of thumb instruction set, thumb 2 came, then for security this trust zone came, then this multiprocessing facilities came. So, they are available in ARM 11 and type of processors, then we have got, after that the ARM developed three different profiles. So, for the server side, they have got ARM, they have got these A series. So, then for the microcontrollers they have M series and for real time processing there R series, that there are three different profile scheme, and many other additional features, like say this security and these floating point operation features they got introduced. So, that way it is in the operatives.

So, right now or the version of ARM, that we have for microcontroller operation are these cortex series and cortex m 3, m 4 type of processors. So, that are the current versions that we have. So, we will look into this ARM architecture why is it so interesting, why is it maybe better than many of the existing processors.

(Refer Slide Time: 24:07)



ARM7 block diagram

$x = x + y \ll 2$

SHL y
ADD x, y

ADD x, y, SHL #2

So, if you look into the block diagram, then this is the thing, that you can have this, there is, I should say, I will start with the ALU. So, there is a 32 bit ALU that is there and there is a register bank that has got 31, 32 bit registers plus 6 status registers. So, I said that with a RISC architecture and its registered reach architecture. So, you see there are 31, 32 bit registers plus 6 status register that way there are a large number of registers.

Then ALU is 32 bit and so for operation, you can have one data coming through these A bus to this ALU, another data can come, either from this register, or can come from this memory through this or from the instruction is through this one, and that is that we generally called the B bus and there is a barrel shifter put before this ALU. So, barrel shifter is used for shifting the bits of the operand.

So, in one instruction, so you can send the first operand directly, second operand you can say that the second operand is actually shifted by some bits, like if you want to do an operation, say x equal to x plus y left shifted by 2 bits. So, if normal processor, first of all you have to do a shift left y and then you have to say like add x comma y. So, two instructions will be needed, but in case of ARM processors. So, you can simply say add x comma y comma SHL hash 2. So, something like this.

So, this will mean that the second operand. So, it will be left shifted by 2 bits and then only the addition will be done, by a single instruction we can do this thing. So, this is another, so this barrel shifter being on top of this ALU. So, this helps in this process ok.

So, apart from that we have got, say this, other interesting point to note is the ALE signal is there, ALE signal in case of other processors that we have seen. So, they were going out from the processor, because that was used for demultiplexing the address data, but here the purpose of ALE signal is just something else, here the if the memory chip is slow. So, it can request the processor to keep the address on the address bar.

So, accordingly it can raise the ALE signal. So, that the processor will keep the content of this address register available on the ALE on this address bus, till this ALE signal is high. So, this ALE direction is towards the processor, unlike previously what we had, was ALE was coming out from the processor going to the other control to the memory chips.