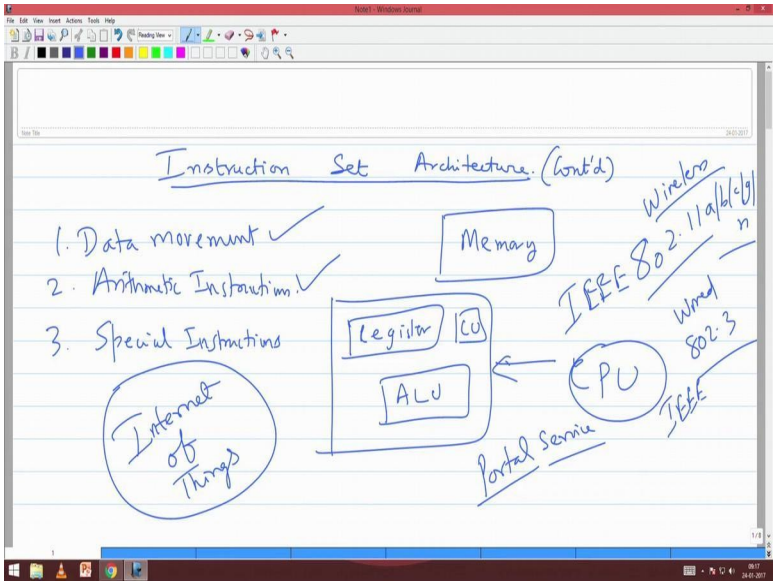
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 10
Instruction Set Architecture – continued

So, good morning, so we will continue with the instruction set architecture. So, where we left yesterday was on the risk consist, we completed some inside into what is the risk architecture and what is the CISC architecture and what is the implication of having a risk architecture on the compiler versus hardware and vice versa. So, these are some of the things that we talked of and importantly we understood that there is something call and up code and then operant and every instruction. Now today we will now see what are the type of instructions and how did they come into practice etcetera. So, every instruction that is added into a system has a background. There is a legacy there is a history why that instruction essentially comes into the thing and understanding that is extremely crucial for you to become a good compilerator.

So, when you study code generation. So, you did code generation in the last semester here. So, one of the important things that if you want to really become a very good you know right a very efficient compiler or you want to work in the area of compilation one interesting thing that you should actually developed that interest is that you should understand the legacy behind every instruction why this instruction is that. So, let us go and do that. So, it is going to be interesting history. So, let us see. So, what are the types of machine language instructions that you can conceive of one thing is in a system there are as I told you there is memory and inside there are registers and then there is an ALU and there is a control unit which controls the movement of data here and there.

(Refer Slide Time: 02:03)



So, this is the organization of a simple computer now this collectively you call this as the CPU central processing unit. So, the type of instructions could be one of the important instruction would be data movement instructions I move data from memory to register, register to memory right etcetera. The second thing would be arithmetic instructions like your add subtract etcetera then there are certain you know special instructions right your data movement I do not need to justify why you need data movement if all f it is not there you cannot even move data from memory to register or register to memory arithmetic instruction again I need not justify why you need an arithmetic instruction right add subtract multiply these are all basic things floating point where it is of this.

Now, what is more crucial for us to understand these special instructions? So, what are these special instructions why are these necessary what is the legacy behind introduction of these special instructions once you understand that because finally, whether you do algorithms today you said some cryptography there is something call modular exponentiation that you will be doing as a part of it now there are specific instructions exclusively added to your system today to solve certain cryptography problems right. So, these are called crypto acceleration instructions there are specific instructions that are added to your instruction set for doing media applications like your you know rendering of your screen right doing certain computations on that. So, those are called multimedia

instruction. So, processing your audio file there are specific instructions that are done for

doing certain data base transactions the specific instructions that are done to do web bays transactions.

So, today I buy as a processor which is for a server class machine was as a processor that is for a work station class machine what is a difference right there are certain transaction based instructions that could be part of your server class machine which will not be as a part of here normal work station class machine right today I buy a processor for a real time system versus a normal system right already I we discusses in the class. So, what is the distinguishing factor between a processor that is design for a real time system versus processor that is design for a normal system we discussed this in the class which spend for 5-10 minutes on this, what is that distinguishing feature?

Student: Depth of a some;

Student: Depth of a sometime and;

Now, these are definition of what is the real difference between a real processor that is made for a real time system versus a processor that is done for a almost fast interrupts excellent. So, interrupt and inter processing instruction should be very very very as I in the case of a real time process. So, as I move across applications my thrust on what I am going to do with the system changes right for example, the computer that I am talking today is not just one that you see on a desktop or your server or your you know or your mobile phone today these are all equivalent for you. So, whatever you do on a desktop today you are majority; majority of the things in addition to talking you are able to do only mobile phone. So, these are all some classes of processors that you are seeing here, but then if we take a network switch or a router again there is a computer there; there is a processor there.

Now, the type of instructions that processor will execute will be extremely different from the type of instructions that a normal you know fancy computing processor will do right for example, what will happen in a network process can you just think of it this is just an introduction to a networking course; can you just (Refer Time: 06:40) somebody can tell me what is what is inside a router or a switch how many people have seen a router can you put your hands up right all of your seen a router. So, what is inside a router there is a CPU there is a processor there is lot of memory put inside the router. So, there is a

computing environmental can you just guess what are a type of instructions that processor will execute.

Student: (Refer Time: 07:05)

Packet; excellent packet processing who gave this answer can I just good excellent, so, packet processing is a very important activity of a network processing. So, what you mean by packet processing? See it is another things like let us take a Wi-Fi router right the way a packet is transfer bit from this system say to a Wi-Fi and from there it is transmitted to on a network. So, this protocol is different from a wired protocol right. So, typically and access this is called an access point you should have seen that or whatever you see as a hotspot; hotspot again is Wi-Fi on both side or u s b plus Wi-Fi. So, let us take a switch. So, there will be one protocol one I triply standard which will be transmitting from here another I triply standard which will convert it to a wired protocol the Wi-Fi is 8 o 2 11. So, I will just note all these things because these are term something which will be useful in future also 8 o 2 11 a b c g n. So, many things are there. So, these are difference standards.

While you look at this is the wireless standard the wire standard is 8 o 2 3 if I remember correct. So, I triply these are all I triply standards now one of the important packet processing function is to convert and 8 o 2 eleven a packet into an 8 o 2 3 packet right; 8 o 2 11 a will have some pro some header 8 o 2 3 will have some other type of header. So, this conversion from one format to another format is a very very important functionality of a network processor correct. So, this is sometime you call it as a portal service what do you mean by portal service I am porting one format to another format and vice versa if some packet come from the wire network to the wireless network then I have to convert. So, not of packet processing functionalities like this form the basis and this we will not see in a normal processer you see in a network processer.

So, when I want to design a network processor the instructions set for it, it will different from the instructions set that we see on a normal system. So, we can we guess of some other processors other then these now processors that you see on what we call as this internet of things (Refer Time: 09:37) these are all call microcontrollers small processors one of the distinguishing feature between a normal desktop or workstation processor and a control controller is it will not have extensive memory management etcetera it will

basically it will not given run and operating system or there is necessity for me to run and operating system because the functionalities is very spurcific right of course, it is programmable, but the functionalities specific take some data collect some data do some small processing on it and send it up.

So, these are these are processors for what we call as internet of things up of course, this is one major password and its going to be a major business time permits sometime in the class I will give you some very excellent videos and what is internet of things. So, internet of things are basically I want to major pollution I put. So, many pollution sensors across the city and there will be some processor which will be responsible for pinging that sensor getting that pollution value in that area collect all these things and sent to a centralized server and then there will be a big data analytics that you do your machine learning and all these things and you just signed out you get a big chart in ensuring this area of this city has this much etcetera.

So, what is the processor that I am putting on those device that processor will basically do only one functionality it will it will ping the sensor get a value do some normalization or something there and communicated to over a network to a server. So, that is the very very specific simple thing that it does. So, normally there will not be any operating system that is executing one that if I do not have a an operating system then many of the instructions can go on. So, it will be a very very simple light weight processor with a few instructions and what is the big advantage you get here in other way if I refresh the question why cannot I put an Intel processor for the I o t device server class processor.

Student: Very faster to.

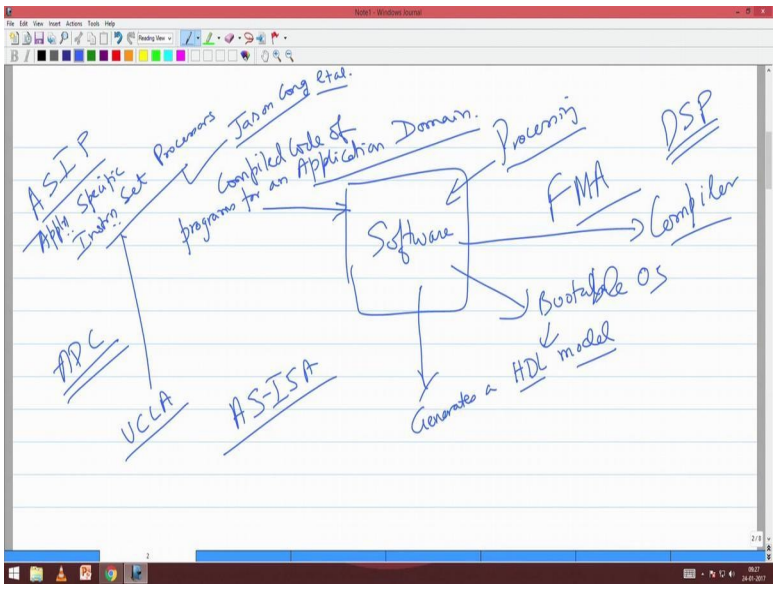
Faster now I no need. So, I am going to ping that for say once in every 5 minutes. So, I got I will get say 4 bytes in every 5 minutes power right.

So, I if I put a the big processor that it will conceive lot of power which I cannot offer right. So, I basically these things will work on some solar cell small small solar cell and it will work. So, it will be in the overall all total consumption which should not exit milliwatts right, I cannot put a watt processer that. So, these are all some of the big things that come up and everything every decision finally, reflects in the form of an instructions said to you because the instruction said is the interface by which I can

basically access the hardware please understand how do I go and access the hardware how do I make the hardware do something it is true the instruction set.

So, all my engineering decision all my functionality based decisions everything finally, reflex in the form of a instruction set and that is why study of instructions set architecture I understanding the legacy some amount of history is extremely crucial. So, I just gave you a brief overview of different types of computing elements that you see in practice today and how these are basically you know what are the instructions there and what are some of the functionalities there. So, now, let me just go into one very interesting how people made business out of this.

(Refer Slide Time: 13:33)



So, this is a very excellent business idea. So, there was a company which made as software this software does the following it takes as input yeah compiled code of programs from an application domain.

So, the application domain can be say web applications right financial applications networking application cryptography applications. So, one application domain it took it took all the compiled code in some language it will be compiled for some 6 8 6 processor or it can be an intermediate representation. So, when you do a compilation as we saw in the previous

semester from the source language it comes to some intermediate representation and then it goes to the machine language. So, I could take that

intermediate representation. So, just if may add the (Refer Time: 14:40) what was that intermediate representation Raghav.

Student: V m.

Last time;

Student: V m.

V m; so, there was there was a virtual machine. So, we will have some intermediate representation. So, it will take that compile code than it does some processing and find out what are the instructions that could make the execution of such type of codes you know; what are the machine instructions that could make the execution of such type of codes extremely fast? So, it saw from patterns right it is a lot of patterns I will give some example as you pro. So, it is a lot of patterns. So, if there is a pattern take this data to that that to this and I had this then some pattern is that that entire pattern I will capture it as a single instruction for example, multiply and add if one instruction that you do for lot of digital signal processing you multiply and immediately add it. So, why should have a separate instruction for multiply an another instruction for it let me just say. So, there sometimes called FMA fused multiply add just do it one short. So, that was a new instruction.

So, suppose I multiply then I add I take double the amount of time If I just do one shot multiply plus add then the that say big how do you design hardware for that that is another different ball game all together, but if I can do it in one shot I fetch the data I multiply and add with something this use multiply and add can be say double that double faster than the individual multiply then followed by add and if I do this FMA if I if I want to do this multiply and add say one million times right that is say each takes some time cycles it will take ten million cycles if I do multiply followed by add while it will take only 5 million cycles if I do fused multiply and add.

So, there are some basic operations which are repeated many many times and if I do that if I could capture it in a in a single instruction and do it faster than I get lot of advantage. So, if I am going to make a DSP processor digital signal processing processor. So, all that you do here is DSP right your mobile phone has a DSP engine right it is a one which will convert your radio to digital when you talk converting from digital to reduce. So,

when the when your when your signal which is an analogue signal gets converted into digital using something call a d c analog digital converter right a d c is right this digital signal to is to be processed and vice versa when I want to send this signal back. So, I need to process that signal. So, that it needs certain criteria.

So, this DSP processors have this FMA right you do not see this FMA and say an Intel processor, but you will see in DSP processors. So, what is particular software do is to process these instructions find certain patterns and come out with a new instruction set which is called application specific ASISA application specific instructions set architecture.

Application specific instruction set architecture then what it does it immediately generates yeah HDL model last time you have all done your hardware description language right you described your hardware in some HDL. So, it generates that HDL model automatically.

So, that you could go and go to the FAB and make the processor it also gives you a bootable opera bootable operating system that will run on top of this thing. So, it will give you a processor which you can take in a in a hardware description language it will be a soft description which you can take to a fabrication unit and make a processor last semester you have all seen how HDL will be then it will also give you a operating system which will boot on this and it will also give you a c compiler or compiler in some language in which you can write code.

So, what you need to do to the software give a compiled code of several applications which you intend to execute on a processor I want to make a processor for some application domain.

You take all the possible programs give it to this fellow that fellow will process it will find out it will find out some common patterns it will generate and instructions at architecture completely for your domain exclusively for your domain you will not only generate the HDL model it will generate by hardware model which you can take to the fabrication unit and make a processor it will also give you a voice which is could be booted on top of it is also give you a compiler by which you can write programs or it compile and run it all these things will do. So, this is now the genesis of this particular

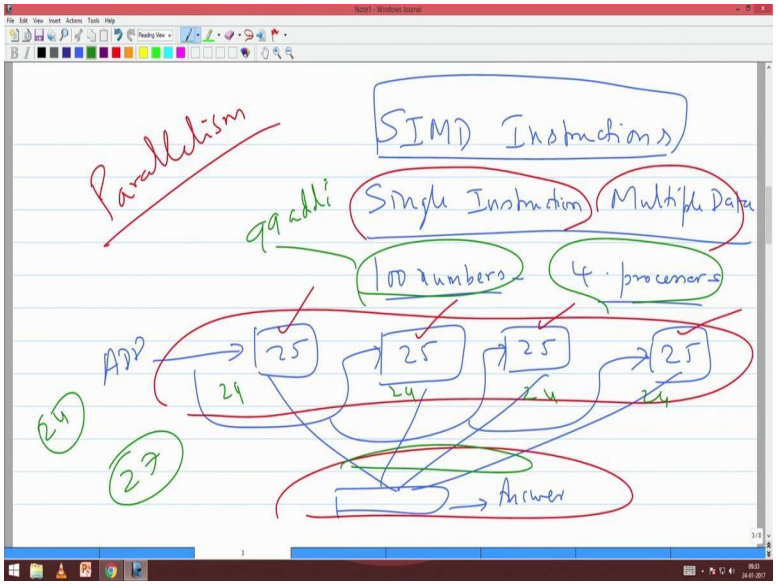
software came from please search for ASIPS application specific instruction set processors.

Application specific instruction set processors by Jason Cong Etal; Etal means etcetera her many authors for the paper this team is from UCLA university of California at Los Angeles. So, just look at this right. So, I want all of you do p h d do not go for some job after do something interesting one now I will tell you what is the commercial model for this I was I do not know fortunate enough to sit in one comity where we purchase this for some purpose for some Indian agency. So, what happens is once you this software will be there. So, you give all these programs there then finally, all this done then it will ask you for this is a licensed software. So, there is one button call submit once I submit then the hardware will come there is dell model will come the bootable software will come and the compiler will come and I can click it only ones after that the license will expire next time I cannot do and the clock the cost of clicking once can you guess 10 million dollars.

So, I will go an click once I will get a hardware is real model I will get an operating system I will also get a compiler for it everything I will get tomorrow I can go to the fab and make a processor I will have my own processor industry that that the cost of one click.

So, I will by mistake you go and click this button that all over ten million dollars go is extremely important. So, this is business. So, this is business. So, this how instructions sets have come up in that right. So, this is very very important. So, there is enough motivation for has to understand instructions set architecture and now we have to male it you need to really understand why should I study this.

(Refer Slide Time: 22:39)



So, let us first start with what we call as SIMD instructions there are some f u does not instructions in your excised its processor right we would not talk more about risk here because it is not fun. So, we will talk about is you have the; I believe I have certain believes I will tell you later.

So, let us talk about is no. So, this is SIMD instruction. So, what is SIMD stands for single instruction multiple data single instruction multiple data. So, what is single instruction multiple data mean suppose I say I have 100 numbers and 4 processors. So, I go and say give 25 numbers to each and I say add one single instruction to all of them add each will give me a result which again one of these process will do an add and give me the answer. So, let us not bother about this part of the story, but if you look at this part of the story no I had different sets of data right hundred numbers 1 to 25 was with this fellow, 26 to 50 was this fellow, 51 to 75 was with this fellow and 76 to 100 was instruction.

So, I had multiple data, but I give gave only one instruction to this say add right. So, this is a very trivial, but very important examples of what I mean by single instruction add multiple data different sets of 25. So, I give only one instruction, but it will act on multiple sets of data. So, this is call SIMD this is one example of what we call as parallelism

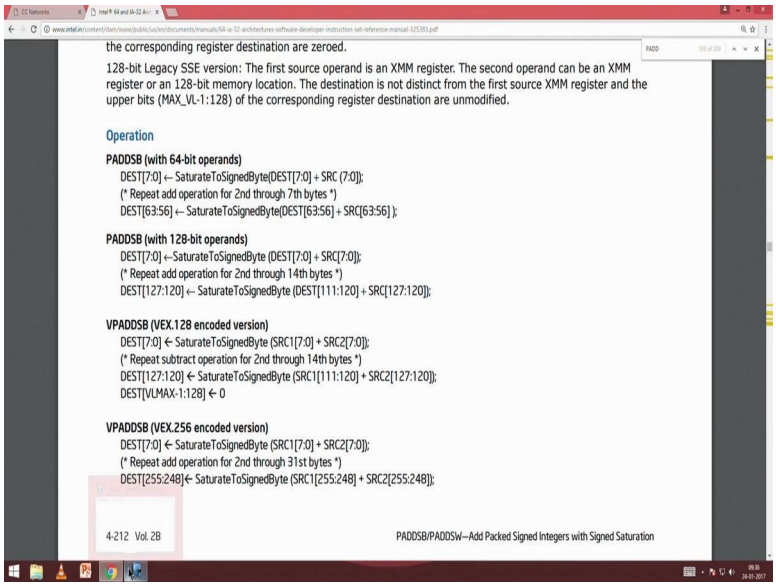
correct. So, I am exploiting the parallelism in my program. So, what has happened is if I add these hundred number sequentially what would have happened I

would have taken ninety nine additions to do that right here I am doing 24 additions in parallel. So, the total time required is only time required for 24 addition plus say 3 addition. So, 27 additions I can if because I have add this 4 in properly I will add this sequential. So, I am exploiting if I have 4 processors I am exploiting this 4 processors to basically rules.

So, what is saw as recursive doubling is also parallelism that right if I add one processor to do the prefix on would have taken order n time if you add n processors then it took me order login time right and that is also an example of a SIMD correct because I did only one single instruction of adding my neighbor and making me point to the neighbors neighbor every step if you remember recursive doubling in a very correct you are remember recursive doubling. So, now, that we see SIMD here where SIMD comes in place can you guess where SIMD can come in here already done one SIMD in your assignment in your third semester where you do them.

Now, where do you see SIMD last semester you did I and ride logical end of n bits with the another set of n bits right you did logical end of end of 2 2 n bits did you do right. So, that is paralism. So, there are n bits. So, 30 bits and I do 32 bit to 30 2 bits I can end in one shot. So, that is that is an example of a paralism.

(Refer Slide Time: 26:57)



So, when we look at media multimedia today. So, what happens I bring another screen on top of this right what happened here every pixel of this screen every piece pixel of

this screen got superimposed on some pixel all of this and that could be something common if I am having this little less. So, I am just doing I am just processing. So, let me say this is some 640 cross some 320.

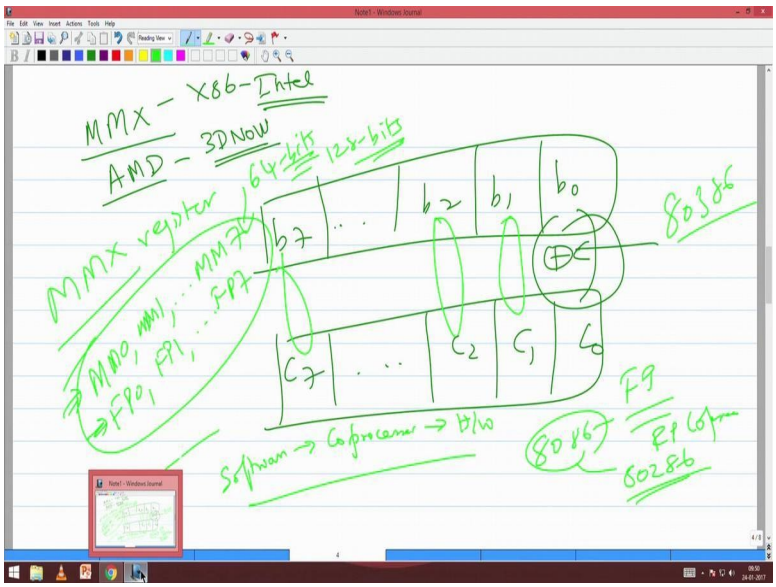
So, 600 into 300 is what around 180,000; this 180,000 pixels that I see here color is getting superimposed by another 180,000 here. So, what I do on each pixel each pixel today let us assume it is one bite it cannot be just because this is a color monitor. So, I add to represent the color and so, many things that let us for just for our understanding let us say each pixel is one bit I want to process 180,000 bites. So, I have one bite there another bite here one bite of which the general this one bite here another one bite here in this case I am fully masking this with that, but in case I will do half masking of this is if this screen is half of this I do not I do not yeah. So, I can see some general here and something here you see this.

So, one 180,000 pixels I have to do processing I have 2 pixels one 180,000 belonging to this manual one 180,000 belonging to that micro subgenre and have to do some computations on this impact this if I do one bit after another what will happened when I press this will come like slow motion action replay, but if I want fast the movement I click or I think it should go and I just bring the pen here immediately I want to see right. So, suppose I want those type of re responses right I am very happy. So, imagine here one time I click this it will the slowly then I have to teach this scores for 3 semesters. So, by the every time on it a new screen and if I keep increasing the resolution I want. So, crystal clear things is going take lot of time.

So, this is an example where I can do paralism you un understand this where I can do paralism right and I am doing the same thing take this bit this bit and super impose on top of a if this is merging or partially merging then take some of this and some of this and show. So, you the same action the same algorithm the time executing on pixel k is this algorithm and going to execute on pixel k plus 1 understand this follow. So, this is a very very interesting example of SIMD and what you get out of it today I the moment I go and that is this I am getting that screen while why go and touch this I am getting the next screen this is because I could do some amount of parallel processing the response is very quick we understand this is a very realistic example that I could do.

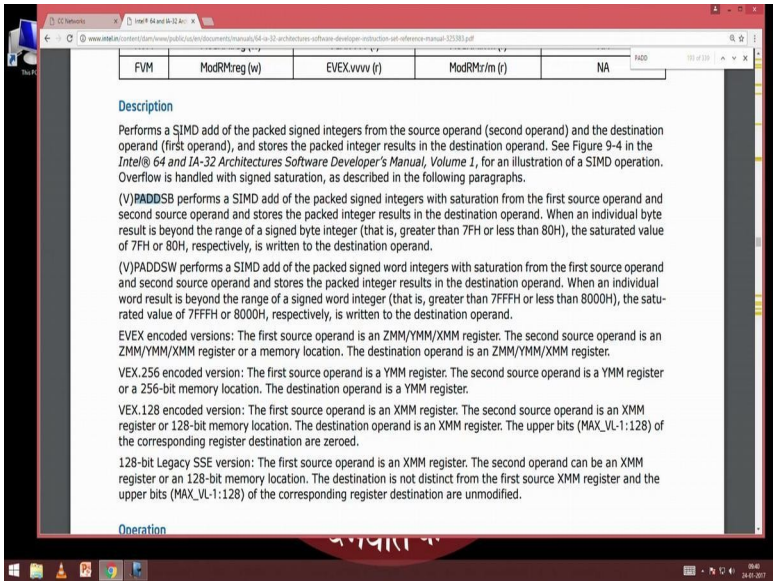
Now, furthers Intel and AMD came out with what we call as MMS instructions multimedia extension MMS is multimedia extends for extension this was brought by x 8 6 Intel and AMD where 2 good competitors good competitors.

(Refer Slide Time: 30:17)



So, when Intel came out with this I am talking about you know around late ninety's AMD came out with what you call as 3 D now instructions because the 3 D graphics was coming up in a big way right. So, these are basically use for basically these are all graphics and you know media processing instructions now I will just show you something that exist even today. So, this is this PADDSB. So, let us go into what is what does this mean please note here. So, this is central volume.

(Refer Slide Time: 31:10)



So, I will not. So, I will not read this completely for you there was, but please note some very important words here oh I cannot this not work if performs a SIMD add what do you mean by performing an SIMD add let us go at one of this.

So, I have a 64 bit operand. So, this will split this 64 bit operand into 8-8 bites. So, will 8 such operations. So, I will have 2 registers of size 64 bit in which I will have 8-8 byte. So, I will have 8-8 bites or 8 bytes, I will split that 64 bit (Refer Time: 31:47) 8 bytes another 64 bit also into 8 bytes and I will do operations the same operation like I will add the respective bytes here right you understand this. So, this is p add with lot of thing. So, there are some 3 to 4 dozen such instructions to these are all name SIMD instructions now let me go and do an example of this. So, I have this 64 bits I will split it has b 0, b 1, b 2, till b 7 and I could have another 64 bit which I will call it as c 0, c 1, c 2, to c 7 and I will put an operator here this b 0 and c 0 will operate together when this is operating b one and c one will operate here then b 2 c 2 and b some (Refer Time: 32:43) altogether.

So, in one shot I can finish of 8 bytes processing of 8 bytes assuming that your screen was each pixel was a byte. So, I can do whatever I could do sequentially I could do 8 times faster I will be able to follow this now these registers are called as MMX registers in Intel they are they are named as m m 0, m m 1 to m m 7 right, you have a floating

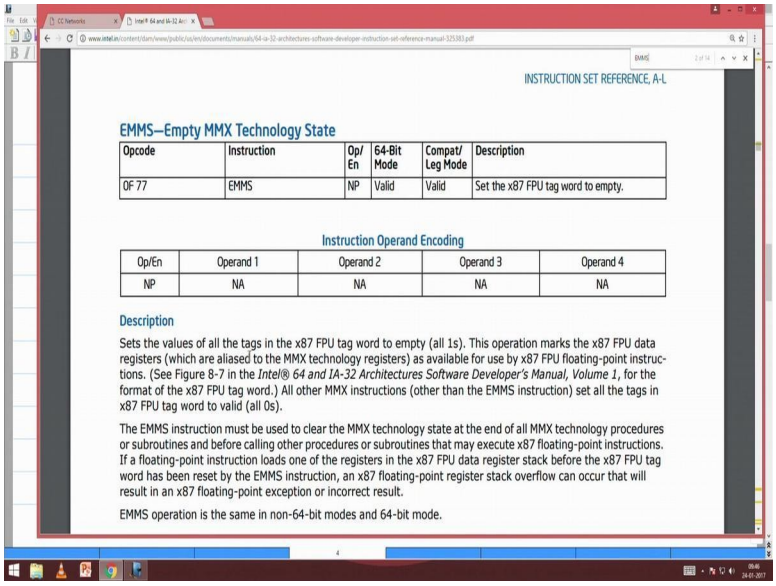
point registers f p 0, f p 1 to f b 7 like that there are for these are in addition to a general purpose registers like e a x e b x e c x c d x, last class you should have done some

assembly programming did you do that in overlap right, now in addition to those e a x e b x these are the MMX and all these in our earlier process or 64 bits now you have one twenty 8 bits also.

Now, when Intel started introducing this MMX instructions right please understand we started which last time he said right your e a x is one register of 64 bits as 32 bits in which there are a x is the first is 16 bit a h and a l are the first and second 8 bits right. So, there is there was a re use of the a h and a l register to realize x and there was a re use of a x to realize e a x right we explained you in the first class, similarly here when Intel first introduced they way they reuse the floating point register for MMX. So that means, your Intel processor can run once in floating point mode and then if you want to switch to and running multimedia more then you have to do some instruction to switch the state in able to get this you able to get this right I have I am first executing in floating point and I want to switch to multimedia then I need to have a special instructions which is switch it in to a multimedia mode.

So, when the processor company want to introduce a new instruction said it has lot of implications I need to change my complete control structures I need to change my data path etcetera. So, because I am adding a new set of a registers I need to change the hardware right. So, the way and instruction is introduced is as follows first they will try and use the existing infrastructure if its works very well then they will make it as a you know they will add new in infrastructure. So, when I want to execute MMX instructions the infrastructure I need is first registers and then I need some execution units the execution units are pretty simple because they are add subtract etcetera, but I need registers. So, what Intel decide at the first step was to use the FPU register. So, if you look into Intel manual very interesting like if you go and search for; so, EMMSMT MMX technologies state yeah.

(Refer Slide Time: 36:15)



So, if you read this says the values of all the tags in the x 8 7 FPU tag this is floating point unit tag word to empty this operation marks the x 8 7 FPU data registers which are aliased to the MMX technology registers please note this for aliased right aliased means they are the same as available for use by x 8 7 FPU floating point instruction right. So, if I am executive multimedia instruction then I have to put EMMS then I can now start using floating point. So, that is how when I want to introduce and instruction into the architecture I re use the states right these are lot of complexity there a way. So, a processor can either do MMX or it can do floating point, but not both together and whenever I want to move from MMX to floating point I need to you know store the resistor values back and we know execute this MMS then do floating point again get back to multimedia.

So, but these are some of the interesting things that happened in the history right another interesting thing of course, I have been telling you. So, for example, floating point floating point was not part of your processing unit at all till your 8 8 6. So, till 8086 floating point was done floating point was mimic using integer arithmetic yesterday I told you how you will add to floating point numbers multiply 2 floating point numbers divide etcetera right. So, this was basically done using integer arithmetic there was no specific floating point instruction then suddenly it found out that engineering computations need

more accuracy in precisions. So, the 7 5 4 standard came and

subsequently or along with it almost concurrently I need parallel Intel introduce this 82 86 which is called a floating point co processor.

So, this is not part of here 8086 this is an 8286 floating point co processor now when I am starting to execute a program on 8086 and I want to do a floating point operation the processors had the facility to check whether 8286 exist if it exist then it will take an get it executed on the hardware if it does not exists then it will mimic it using software libraries right. So, I had to compile for an 8086 program if I am writing floating point programs involving floating point operation I need to compile once for 8086 with 8286 and another way of compilation for 8086 without 8286.

So, just because floating point was good the technology was not matured enough to just go and immediately including in your instructions at architecture. So, they made a careful decision put it on a separate co processer if that files only that co processor will file not the main system then they if stabilized it over a period of time and then today your post 83 86 the systems did have both the floating point and the integer arithmetic put into the same processor. So, essentially yesterday's software is today's co processer and tomorrow's hardware. So, this is a transition yesterdays software actually becomes today's coprocessor and then it becomes tomorrow's hardware.

This is also the case with graphics right. So, all the graphics was done using software at some point of time then they found out. So, if I want to draw a line we will go like this slowly imagine here now I want to go up with this and this every line gets drawn like this at all it will take yes right. So, then lot of functionalities where shifted to the graphics fellow like this fellow will say go and do something the main process will just say go or do some graphics functionality do and hidden surface elimination the graphic processor would do that I think I mentioned in last semester course when the main processors running at 30 pipe line of size 30 the graphics processor was running with pipe line of forty 5 or 48. So, the graphic processor was much more complex in the main process right, but initial it was software then it became co process today many of the graphic functionalities do come inside the chip I do it is not still part of your CPU is different, but it is inside a chip inside a single chip.

I also have the graphics co processor made that is why its call system on chip the chips you get today the integrated circuits we do not call it just CPU you call it system on

chips. So, that is another very interesting. So, as you see SIMD was the new class of instructions that were introduced.

Thank you.