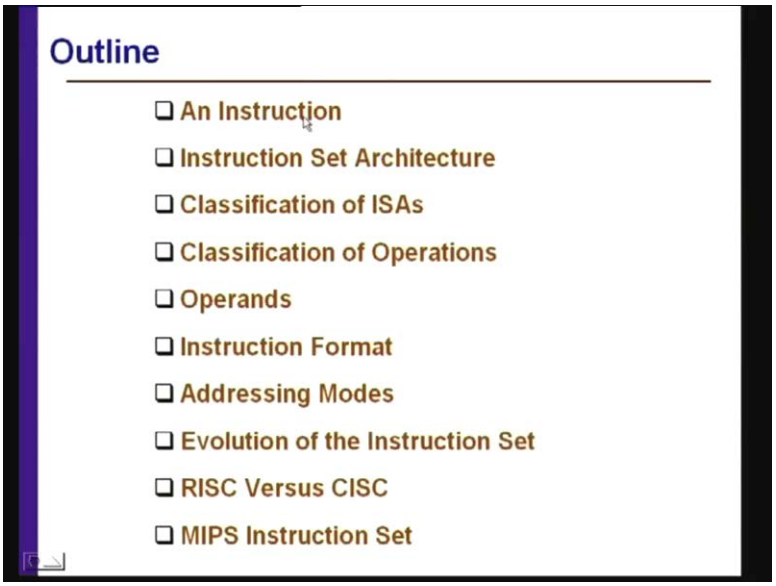
High Performance Computer Architecture
Prof. Ajit Pal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

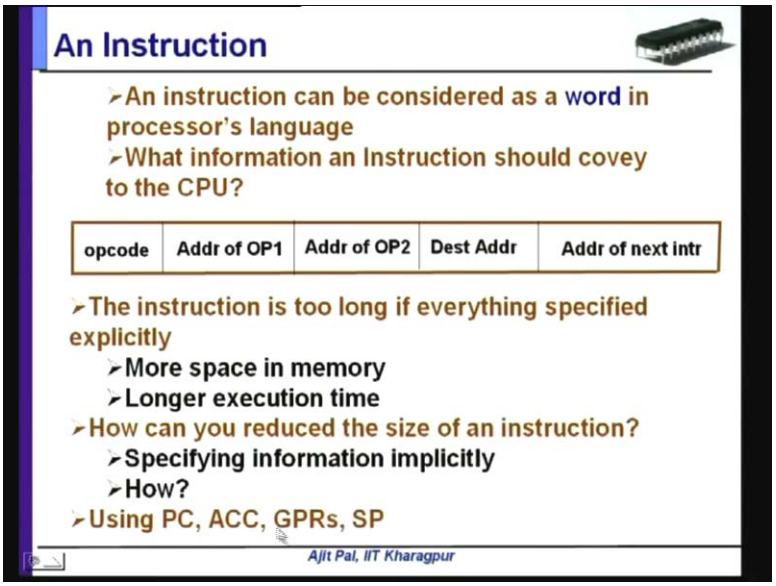
Lecture - 3
Instruction Set Architecture

(Refer Slide Time: 01:04)



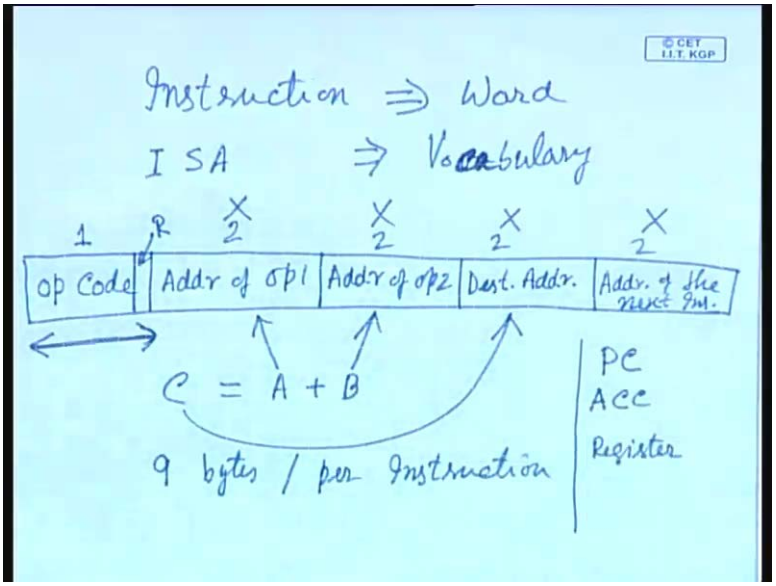
Hello and welcome to today's lecture on instruction set architecture and in this lecture I shall discuss various topics such as what do you mean by an instruction. Then, what do you mean by instruction set architecture, classification of instruction architecture and as sport of the instruction at architecture will find there are you have to perform various operations. So, there is classification of operations then classification of operands instruction format addressing modes, then evolution of the instruction set. Then, there are two bases classes of the instruction set architecture RISC verses CISC discuss and finally, I shall discuss about the MIPS instruction set architecture.

(Refer Slide Time: 01:45)



So, let me start with by an instruction what you really mean by an instruction you know whenever you want to talk to somebody you have to know the language of that person. So, whenever you are writing a program you are essentially communicating with the processer and you have to talk the language the processor understands the language. The processor understands is essentially represented by instruction set architecture, so an instruction can be considered as a word.

(Refer Slide Time: 02:28)



Instruction is equivalent to a word, in conventional you know language, and then instruction set architecture can be considered as the vocabulary for the processors. So, you can say that an instruction can be consider as a word in process language and question naturally arises what information an instruction should covey to the CPU as we know in our traditional language a word coveys a meaning. So, similarly, an instruction should convey something to the processer what it should convey let us try to understand first of all an instruction can be represented by an instruction format.

Different components of the instruction can be represented by different fields, first thing that an instruction should convey to the processer is the op code the operation to be performed by the processer. You know the task of a processer is to execute instruction one after the other obviously the instruction should tell what operation to be performed. Later on we shall see the various type of operation that can be perform by a processer like adipindict operations logical operations and like that.

Then, apart from the op code, the other operates or the other fields of the instruction are number one is address of operant one. You know whenever you have to perform some operations say let say assume you have to perform c is equal to a plus b question is where from you will get a that is the address of operant of 1, then comes the question of where from you will get b. So, address of operant two-third is you have to store this results somewhere c value of c, so that is known as destination address. So, these are the four things and instruction should convey to n to the processer, finally this instruction execution is over what processer should do now processer has to phase the next instruction where from it will get.

So, that information should also be present here, so address of the next instruction, so this are the different fields of an instruction if everything is conveyed explicitly. Now, that means what I am trying to tell you if all the information likes source address of operant one so address of operant to destination address of the next instruction. If everything is explicitly specified as spot of the instruction, and then instruction will be too long. For example, say suppose this op code requires 1 byte and address may require 2, 4 four bytes, let us assume it requires 2 byte for simplicity 2 byte, 2 byte and 2 byte.

So, you can see even in this very simple format it requires 9 bytes per instruction now, so 9 if 9 bytes is required for a single instruction and as you know a program consist of a

large number of instructions written one after the other. So, a program will require very large memory size because each instruction is requiring 5 bytes if all the fields are specified explicitly. Then, they have to store in the memory and as you know from the memory have to transfer to the processer whenever the execution take place. So, not only the program will occupy very large memory because is each instruction is long to fetch them from the memory to the CPU.

It will take long time because if the one size is 3, you know 16 bit, you can fetch only 2 bytes at a time. So, you will require, I mean say for 9 bytes you will require at least you know 5 cycles, 5 machine cycles to fetch it. So, each instruction fetching will require long time in other words execution time will be long. So, storage requirement is long process is time processing time is long if all the fields are specified explicitly in that case what is the alternative is to specify implicitly rather than explicitly how can it be done.

For example, you can use a special register known as program counter pc program counter and it is implied that address of the next instruction is always present in the program counter. When the processer is initialized, I mean whenever you put reset button then program counter will be loaded by the address of the next instruction to be executed. Then, as one instruction is fetch and it is executed, it will automatically load the address of the next instruction in the program counter. So, program counter is considering to be this source of the next instruction, so you do not require this field.

This field is no longer required in the instruction if you see is providing the address of this next instruction, so it is implied that program counter will be provide the address of the next instruction. So, this field is not required, now let us consider another possibility say you are using three memory locations to store the sources source of the operant and distention address, what you can say that address of operant one will be the destination address you make the assumption that address of the operant one will be the destination address where the result will be stored.

So, in such case there is no need for this field destination address field is not required, now the question of a 2 this two fields address of operant 1 and address of operant 2 can you get read of them? You can get read of them in this way you assume that one of the operands will be always will be always able available from the resistant known as accumulator acc accumulator so accumulator will hold. One of the two operant always

that means you have to load the load one of two operant in the accumulator before you execute an instruction. So, whenever you make that assumption maybe this particular address of operant 1 is not required.

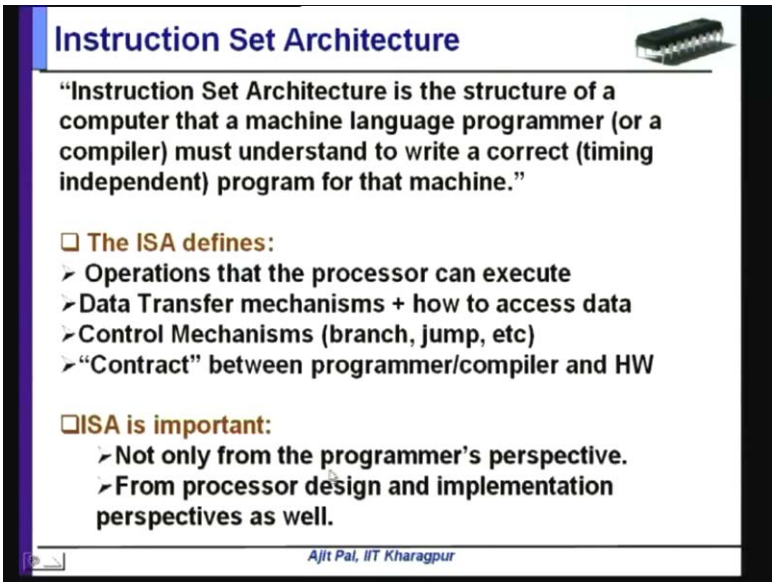
However, you will require another field address of operant two will be required as part of the instruction can you get read of this you can you cannot really get read of this fully, but what you can do instead of specifying the full address. You can specify the address in a in a various way, for example what you can do you can use a register as the address of the operant two in such a case may be the op code can be little bigger, say some a part of the op code can be used to specify the register which will be the address of the operant 2 in such a case.

Again, you will be able to overcome this, I mean this field not be present in an instruction, so you find that ultimately you have been able to have an instruction with only one field which specify the op code. Remaining things can be implicit, implicitly specified available from some special purpose register processers knows where from they will get it. Also, what if you register, then the size will be small the reason for that is you know size of the memories pretty long may be 64 kilo bytes minimum few megabytes for now a days few gigabytes. So, you will require 16 beta address or 32 beta address as it happens in present the computers.

So, instead of 32 bit, if the number of register available is maybe say 16 only 4 bites is required, so four bites can be provided as part of the op code field. So, in such a case you will require a very small instructions size this is how the instruction size can be reduced. So, based on this discussion, we shall see you know different techniques have involved at the c modes and another things. So, instruction is too long if everything is specified explicitly, so it requires more space in memory.

It requires longer execution time question is how can you reduced reduce, how can you reduce the size of the instruction specifying information. Implicitly, as I have already told by using program counter by using accumulator by using general purpose register and stark pointer I shall discuss about it little later to implement some special data structures, so this is the basic idea about an instruction.

(Refer Slide Time: 13:32)



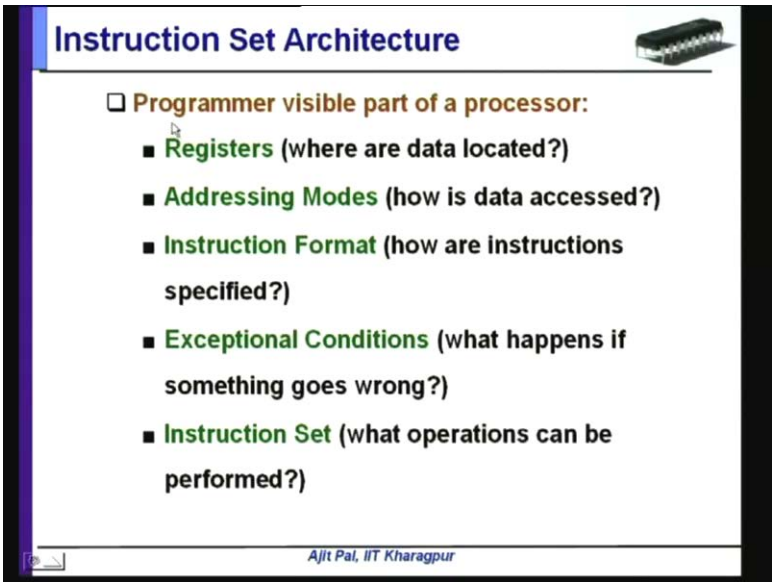
Now, coming to instruction set architecture what it really means an instruction set architecture is a structure of a computer that a machine language program are must understand to write a correct program for that machine. So, instruction set architecture is equivalent to vocabulary of the processer which the programmer must know and with the help of each he can write a program in a simple language or machine language. So, what the instruction architecture defines, it defines the operation that the processer can execute various data transfer mechanism and how to excess data either from memory or from registers.

Then, various control mechanism like branch jump and so on, so it is essentially a contract between programmer and compiler and the hardware. So, you have got hardware and software and instruction set architecture is the essential a contract between the hardware and software. Knowledge of instruction set architecture is important not only from the programmer's perspective. That means if the programmer does not know the instruction set architecture, he cannot write program in machine language.

Not only is that important, it is also important from another perspective from processer design and impletion perspective as well. That means not only a programmer should know an instruction set architecture, but the designer of the processer should also know it because design is essentially the specification. It sharps at this specification to the designer because this is what the processer has to do have implemented those

instructions. So, this instruction can be executed by that processer, so the instruction set architecture essentially subs as specification to the designer of the processer.

(Refer Slide Time: 15:42)



Now, let us focus on the programmer visible part of a processer number one is registers that means a process with help of this registers. I mean there is at the registers are available where data are located you can store data you can access data with the help of the instructions. Then, the instruction set architecture also provide addressing modes various addressing modes with the help of each data can be access either from register or form memory.

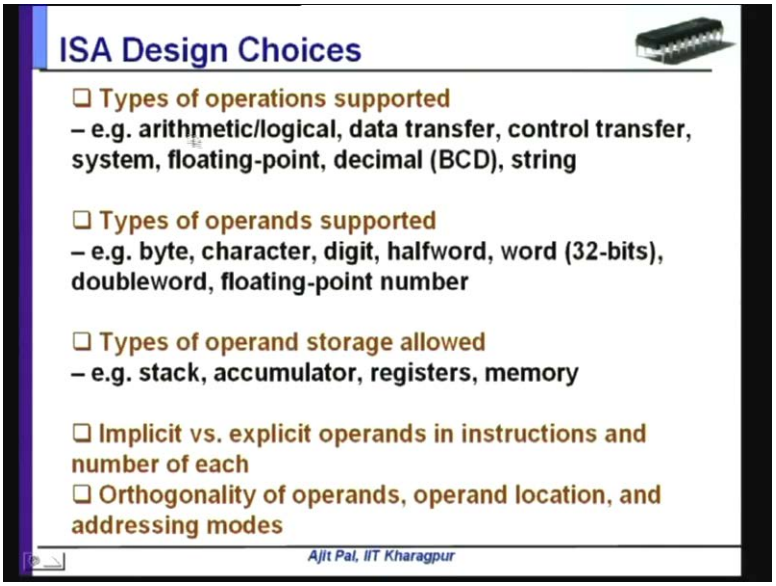
Then, instruction format you will see that and I have already mention about the instruction format, he may call it at instruction format the various fields of an instructions a were with the help of which can specify a various things. Then, it should be also know exceptional conditions what do you mean by exceptional conditions exception can occur in two ways, exception can be generated from the outside world. For example, interrupt it can be generated by an I O or it can be generated by a by user, so reset interrupted inputs are the external interrupts or exception are coming from outsides of the processer.

In addition to that, some exception are generated from within the processer, so whenever in instruction is executed. There are may be something wrong, how that code that is being executed may not be op code may be invalid code because of wrong alignment of

the memory. That can happen or while performing, you know execution of an instruction there are some situation like divide by 0. So, those situations are known as exception generated within the processer whenever an instruction is executed by the processer.

So, whenever that happens what happens if something goes wrong that also has as to be you know specified by the I mean should be known to the programmer. So, whenever this exception happens for example, if interrupted occurs it knows that there is a interrupt of a sub routine there is the specified, specific address to which it will jump. Then, all this to together represent this instruction said what operation can be performed, so this are the different parts programmer visible parts provided by the instructions set architecture.

(Refer Slide Time: 18:36)



Now, there are various instruction set architecture design choices for example, types of operation supported, you know the processor has to do some data processing what kind of data the processer can process. So, this can be like arithmetical logical data transfer control transfer system calls floating point operation addition multiplication division and decimal addition, subtraction, decimal operation, string operation, bit manipulation operation. So, these are the various type of the operation which can be supported, so it is not necessary a processor should everything depending on the application its design.

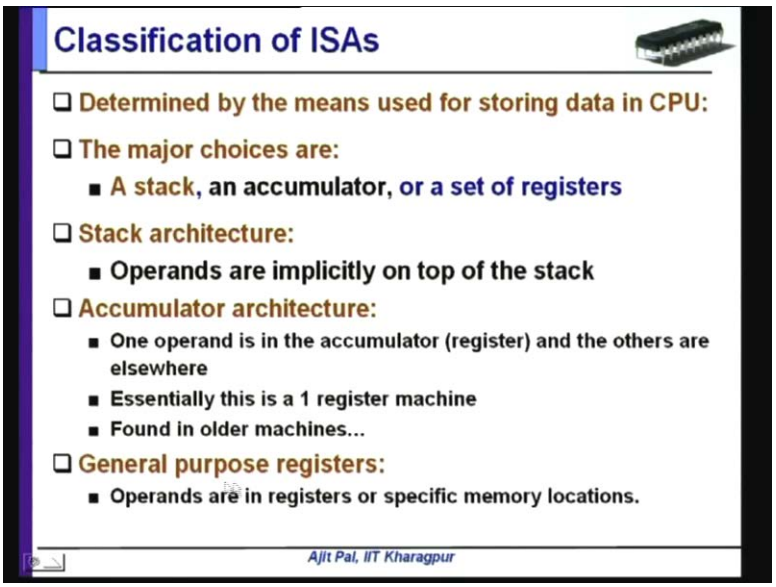
One can decide that this is the subset of the operation a process instruction set architecture will provide and accordingly the process has to implementer. So, it depends

on the application for which a processor design, then comes the types of operant supported. So, types of operant means whether it is byte operant the operation can be perform on bytes or it can be performed on 16 bytes or it can be performed on 32 bytes. So, it can be byte character digit half word double word and also floating point number you can see the operands sizes can vary.

Therefore, format can also vary, some can be the fix point operand's and some can be floating point numbers then types of operand's storage allowed again where the operand's will be stored obviously either in register or on memory. These are the two alternatives available, so there can be various type of storage facility like stack accumulator then registers can be up two types special propose register, general purpose register, then memory by memory. We really mean the main memory perform the process or axis various operant and also instruction, then as I have already told there is the possibility of specifying either explicitly or implicitly.

So, implicit verses explicit operand's in instruction and numbers of each, I have already explain this particular feature how instead of specifying everything explicitly you can specified some of the things in implicit manner then orthogonally operand's. That means whether each operation will support all different types of addressing modes like that so that that defines the orthogonality of operands. Then, operands location and addressing mode, so these are the various design choices of a instruction set architecture.

(Refer Slide Time: 21:58)

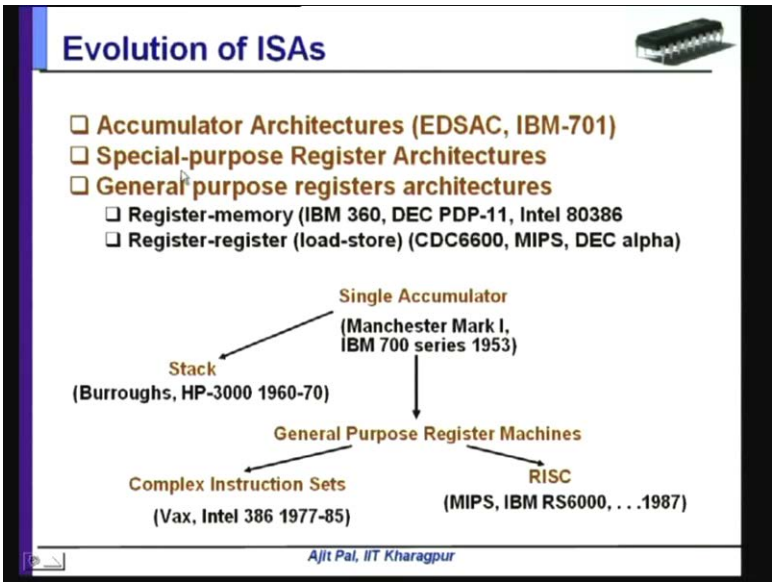


Now, let us have some kind of classification of instruction set architecture, so it is actually determine by the means used for storing data in CPU. So, I have already told you can store register or memory for storing operants. Depending on that, this instruction set architecture can be classified broadly into three types like stack architecture accumulator based architecture and register based architecture. So, there are three possible alternatives in case of stack architecture operands are implicitly on top of the stack, so everything you are doing with the help of a stock stack and you can access from the top of the stack.

So, as you know stack is the data structure last in first out type of data structure, so you can access always from top of this stack and then it you can be accumulator at this architecture. So, as I have already one operant is in the accumulator which is the special purposes register and others are elsewhere. I mean it can be either in some register or it can be some memory locations and essentially this is one register machine. That means whenever it is accumulator based machine, it is implies that the processer has got only one register. That is why it is called one register base machine and this is particularly available in older machine, you know in earlier years the implementation of registers was very costly hardware was costly.

So, only one register was allowed to be provided as part of the processer and that is how the accumulator base processor, where popular in the earlier of computers were subsequently as we shall discuss that restriction was overcome. Then, the general purpose registers comes, where operands are in the register or specific memory locations. So, it is a general purpose, you have a set of register may be 16 or 13 register, they are general purpose in nature.

(Refer Slide Time: 24:30)



So, you can access operands from the registers or it can be also from specific memory location. So, you can see how the instructions set architecture has involved over the years as I said in the early years it was all accumulator based architecture like ESDSAC, IBM 701, this were single accumulated based or IBM 700 series. Back in 1953 these were all accumulator based architecture that means you have only one accumulator and operands is always available taken from the accumulator and result also stored in the accumulator. However, this second operands can be taken from memory, and then came special purpose register architecture.

I have already mentioned about this special purpose register like a program counter stack pointer and there are some more special purposes register provided a from where you can access operands. Then, came the era of general purpose register architecture, so where it can be register memory type.

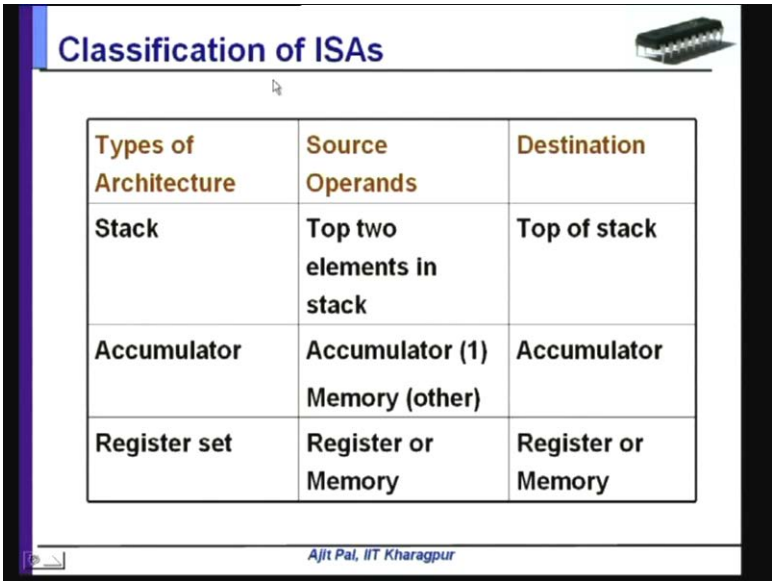
That means one operands is taken from the register second operant is from the memory, so these are known as register memory architecture. For example, process like IBM 360, DEC PDP 11, Intel 80386, these are all you know belong to this this class and they are essentially you know one operant is taken from the register operant is taken from the memory. They belong to the category of SISC architecture complex instruction set architecture. So, then came subsequently the RISC architecture where it is register

architecture that means both the offers and taken from register and result is also store in the register, register architecture.

That means both the operant are taken from the registers and result is also stored in the register. That means whenever you performing various operation at arithmetic and logical operations the always the operands are taken from the register result is also stored in the register. So, this is known as register, register architecture and this are also known as load store architecture because you have to perform. You know loading of the register from the memory with the help of explicitly instruction and stored the results from the register to the memory with the help of storage instruction.

So, those this are also known as loads store architecture and the RISC processer belongs to this category and they are also known has load store architecture like CDC 6, 6, 0, 0 MIPS processer DEC alpha. These are all register, register architecture processer, so you see it is started with simple and then gradually it has become complex.

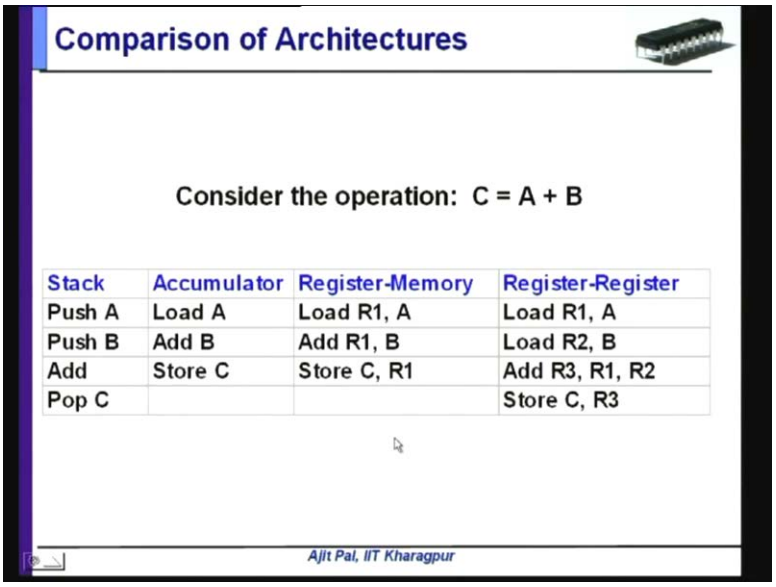
(Refer Slide Time: 27:25)



As I have already told, we can do the classification in three ways three ways stack based where the operands are taken from the top of this stack top to elements in stacks. Destination is also top of this stack, everything is done with the help of this stack in accumulator base the processer as I have already told, and one of the operands is from the accumulator. Other operands is from the memory and result is store in the accumulator and in case of register a general register, register memory or register,

register processer. It can be register or memory and destination also can be register or memory that means source of operant and destination both can be register or memory.

(Refer Slide Time: 28:15)



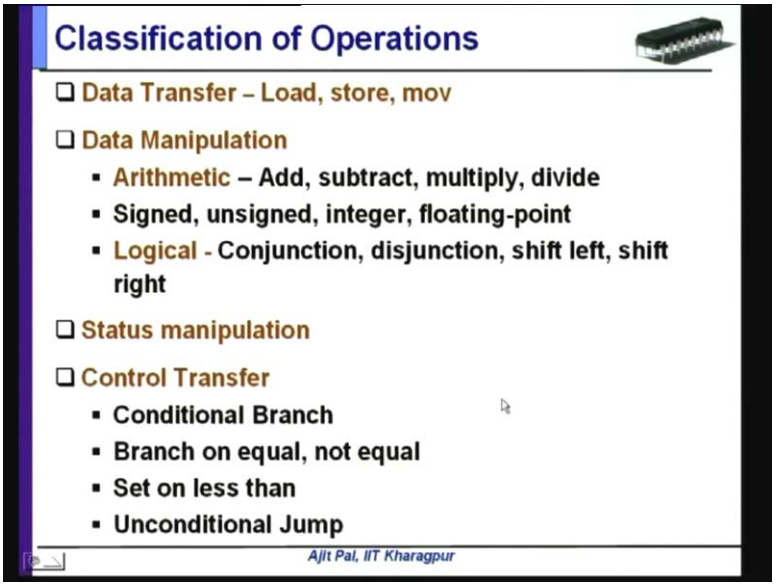
For example, let me illustrate these three architectures with the help of this very simple operation. Consider you have to perform this simple processing c is equal to a plus b in case of stack machine this will be the 4 instructions push a, push b add and pop c. So, this is how the operands will be taken and the processing will be performing then in case of accumulator based machines, first you have to load the accumulator from the memory. Then, you will be adding the one index from taken from the accumulator and second operant is from the memory and you are performing the operation, then storing the result in accumulator, then result can be stored in memory locations c.

So, this is accumulator memory accumulated type of processer architecture and then register memory in case of register memory as we can see you have got the register r 1 which is the general purpose register where you are loading one of the operands. Then, you are performing the operations adding contain the register r 1 with contain the memory location b and then you are storing the result.

Here, the result is available in register r 1 and result store in memory location c in the third instructions and in case of register, register architecture you are always doing with the help of register. So, accumulator is loaded I mean register r 1 is loaded with the first operands, and then you are loading the second operands in register r 2. So, this is the

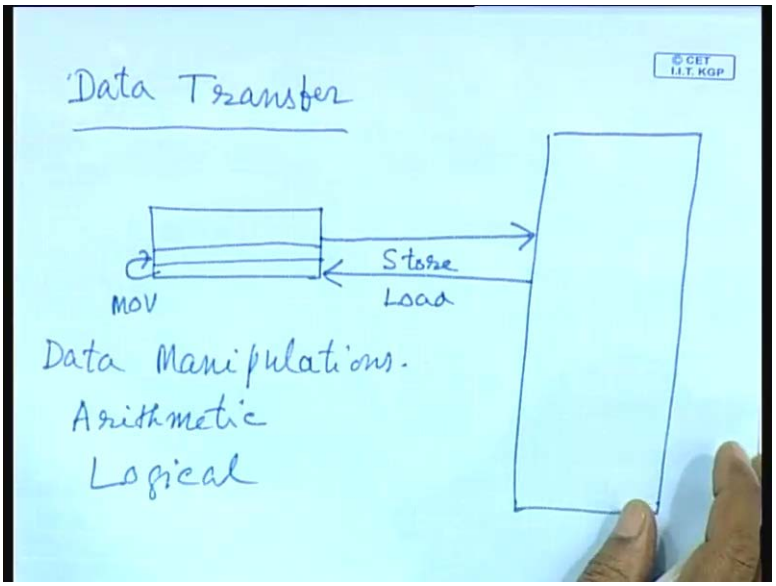
essentially that loads store architecture and then you are performing the addition operation result is stored in another register r 3 by adding contain of r 1 with contain of r 2. Then, you can store the result with the help with the help of store instruction in memory locations c. So, you can see this is the register, register architecture or loads store architecture, so we have compare four different architecture that is possible.

(Refer Slide Time: 30:33)



Now, let us come to the classification of operations the operations can be classified into four different categories, as you can see data transfer data transfer means you are move that data transfer is taking place from register to memory or memory to register.

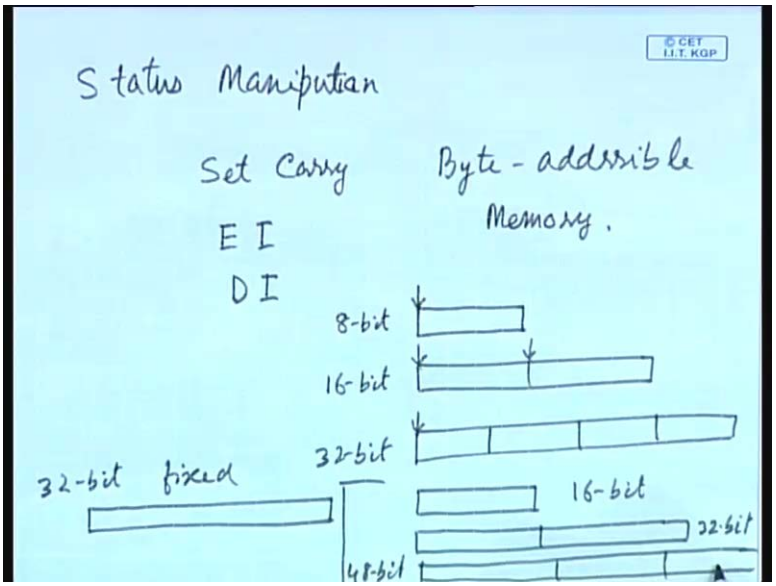
(Refer Slide Time: 31:02)



So, we can say the data transfer so you have got some registers and memory, so this is your storage place. So, it will be either transferring from register to memory, so it can be called store or you will be transferring from memory to register load or from one register to another register. So, this is your normally known as move, so this are the three possible transfer the data transfer that can take place between the between register and memory or within the registers. Then, the data manipulation operations data manipulation, so data manipulation operations can be broadly divided into arithmetic operations or logical operations arithmetic operations are like add subtract multiply divide.

So, again you have got a different possibilities sign or unsigned integer or floating point. So, you can have addition on unsigned addition or signed addition or multiply sign multiply unsigned multiply or integral multiply or floating point a arithmetic operations. So, you have got several alternatives, similarly you have got logical like conjunction disjunction shift left shift right and or like that. So, these are the various data manipulation operations where you are performing manipulation of the data the value of data is changing. Then, there are some instructions which are known as status manipulation what do you mean by status manipulation.

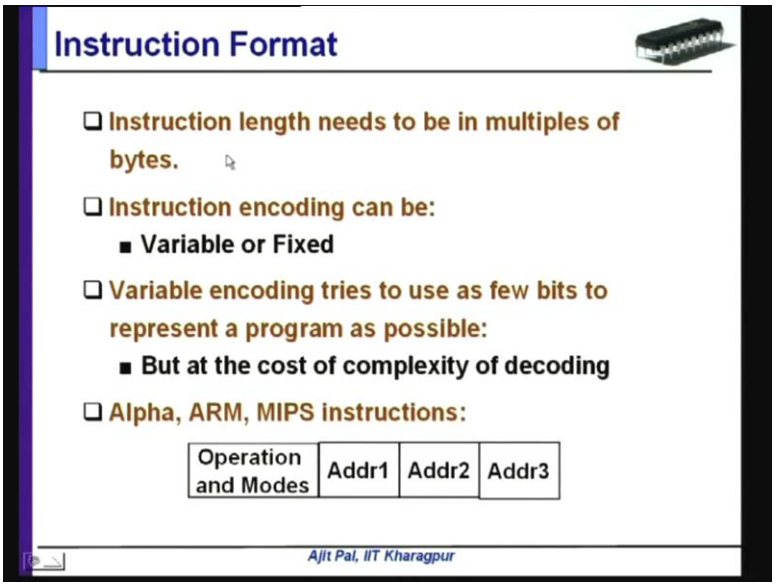
(Refer Slide Time: 33:23)



In case of status manipulation instructions, neither you are transferring data note transfer of data take place no manipulation of data take place. That means neither the data is moved from register to memory or memory to register nor you are changing the value, what happens this status of the processing changes. There are some instructions like that, for example, set carry there is carry flag bit as you known. So, set carry flag bit here the status of the processer changes or say enable interrupt or disable interrupt.

So, this are some of this status manipulation instruction with the help of which status of the computation can be change, but neither data transfer nor data manipulation take place. Then comes the control transfer instructions which can be conditional branch like branch on equal branch on not equal set on less than or unconditional jump. So, there can be various types of control transfer instruction, now operation which can confirm, so this are the different classification of operation that is provided by the instruction set architecture.

(Refer Slide Time: 35:05)



Now comes the instruction format whenever we are considering instruction format it is very important that the length of the instructions should be multiple bytes why it is necessary to be a multiple bytes because you will be storing instruction in memory. The accessible unit from the memory is byte that is why we call it byte addressable byte addressable memory. So, your instruction has to be either 8 bit 1 byte or it can be 16 bites for 16, normally it is a 8 bit for eight bit processer, 16 bit for 16 bit processer. That means 1 byte or 16 bit 2 bits or it can be 32 bit 4 bits, so there are it has to be multiple bytes. So, you can access it from this point and edit or from this point or form this point and you can get a complete instruction.

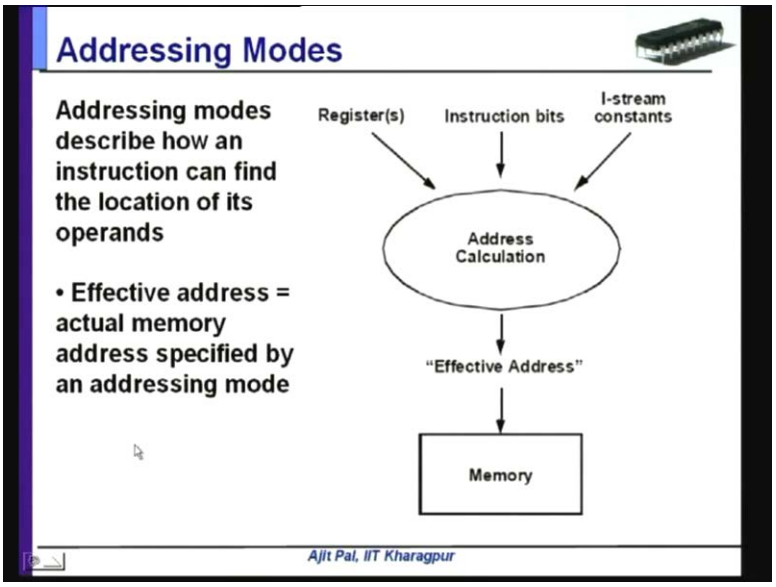
That is why I was telling that alignment is important if it is red from the middle for example, whenever the instruction is 16 bit and if you read from here you will not really get the instructions in a properly. So, whenever you try to execute it, it will give error or exception, now instruction and coding can be variable or fixed there can be fixed format. In other words, what I am trying to tell an instruction can be of fix size say 32 bit fixed all instruction are of 32 bit. So, that is one possibility another possibility is that it can be multiple of some bytes say one can be of 16 bit, one instruction format another can be 32 bit and third one can be 48 bit.

That means in these three cases in this particular case that instruction can be of one bytes size 2 bytes sizes or 3 bytes sizes. So, we call it variable format, so instruction we will

call it variable format, so you can either have variable format or fixed format particularly. Later on, we shall see we can see this 6 processers have variable format, on the other hand the RISC processor have fixed format in general that is true. So, variable format leads to vary variable encoding tries use few bits as represent a program as possible, but at the cost of complicity of the decoding. So, that means here the complexity of the decoding is present whenever you use variable format.

Later, we shall discus on MIPS instruction; we will see that DEC alpha ARM MIPS instruction. They have fixed instruction sixes because they belong to the RISC category this processers and this is the general instruction format for this processer up code. Then, three addresses one, address two, address three, they can be usually, they represent the register, but we should discuss about various format with the help of the example particularly MIPS instruction.

(Refer Slide Time: 39:29)

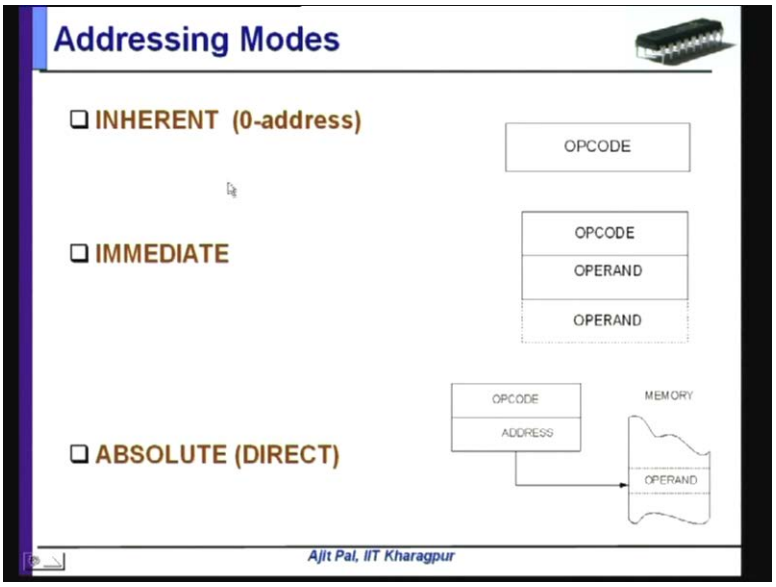


Now, address modes describe how an instruction can find the location of the operands, so you can have varieties of addressing modes to facilitate accessing of operands from registers as well as from memory. You will see that either from register be that means you have to do calculation of the address from where you will get the operant and that is called the effective address. So, effective address is calculated whenever you are operands are in memory, what you can do that effective address calculation may involve

readings some value from register and some part of the instruction that can be used two compute the effective address.

Later on I shall discuss about the how effective address is calculated in various situations our in different addressing modes. So, effective address is essentially the actual memory address this specifying specified by an addressing modes the mechanism by phase the effective address is calculated varies from one addressing mode to the another addressing mode.

(Refer Slide Time: 40:50)



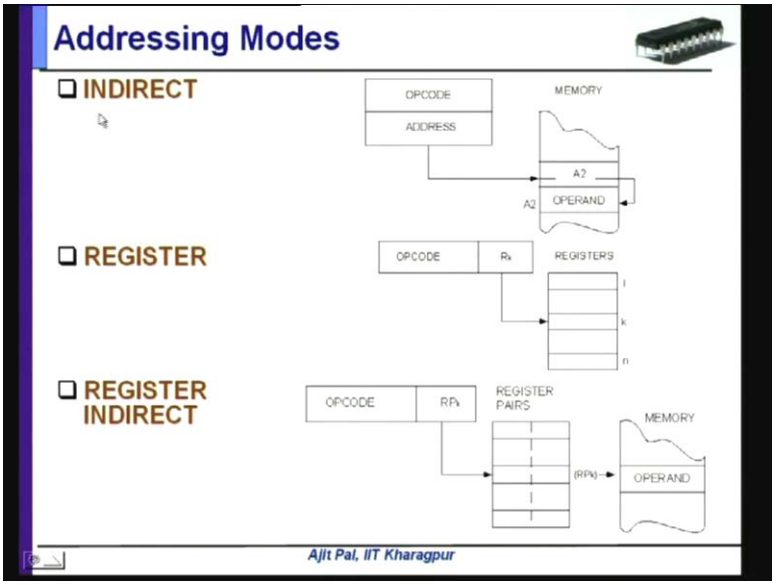
Here, I shall discuss about the various addressing modes, for example inherent addressing 0 addresses. So, here for example you have nothing but up code, so rest of the things are implicit obviously one of the operands is in the accumulator. So, it can involve a an accumulator based instruction since such a case say it can be say implement of a value implement the content of the accumulator or compliment the content of the accumulator. In such a case it involves only one operator, so if it is in the accumulator that is implemented or inverted and result will be also in the accumulator.

So, there is no need to have any other operands address, so that is why this are called this is known as inherent a 0 address modes only the up codes need to be explicitly specified rest of information are implicit. Then, an immediate addressing the operand itself profile is part that mean you have got op code and first operand may be in the accumulator, second operand is provided as part of the instruction an explicitly. That means you will

add the contain of the accumulator, I mean if it is add operation with the with the operands which is provided as part of the instruction.

So, this is the second addressing mode third addressing mode is known as absolute or direct addressing mode in such a case as you can see the address of the operands particular address of the second operand first operand is a. Assume in the accumulator address of the operand second is available from the memory and that address is explicit the specified and as part of the instruction, so that is why it is called absolute or direct addressing.

(Refer Slide Time: 42:46)



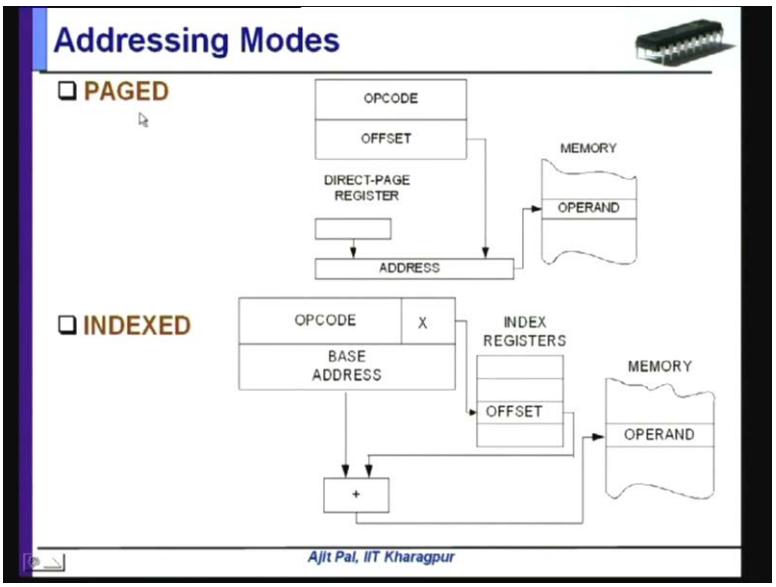
Then, you can have to incorporate little bit of flexibility, you can have indirect addressing in indirect addressing instead of specifying the address from the memory, sorry instead of specifying address. Explicitly, what you can do that address can have the address of the operands instead of the operand as you can see here in this case this is the address, but this the address does not contain the operand. The address contain the address of the operand, so you got a kind of indirection, so this address is pointing to another memory equation which is the address of the operand so that a 2 is the address of the operand.

So, again you have to read from memory location a to locate the operand, so this is known as indirect addressing essentially to provide little bit of flexibility in addressing. Then, you can have register addressing as I have already told some parts of the op code

your small portion of the op code, I mean the that instruction the op code is little smaller. Here, you can provide the register name register number and which will give provide the operands, so here is your register, set of register or register bank whatever you call it. This is pointing to the operand, I mean register where the operand is stores, then you can have register in direct.

So, just like your indirect addressing here also that register is not having the operands but, is having the address of the operands. So, you can sometimes use the concept of register appear because we are single register may not able to hold the entire address. For example, you are register is 16 bit, but address is of 32, so you require a pair of register in profile the full address. So, such a case we use the concept of register pair, so as you can see here a register pair is holding the address and that address can be used to phase the operand. So, that means effective address here is coming from the registers, so this is how in different situation the effective address generated as you can see.

(Refer Slide Time: 45:23)

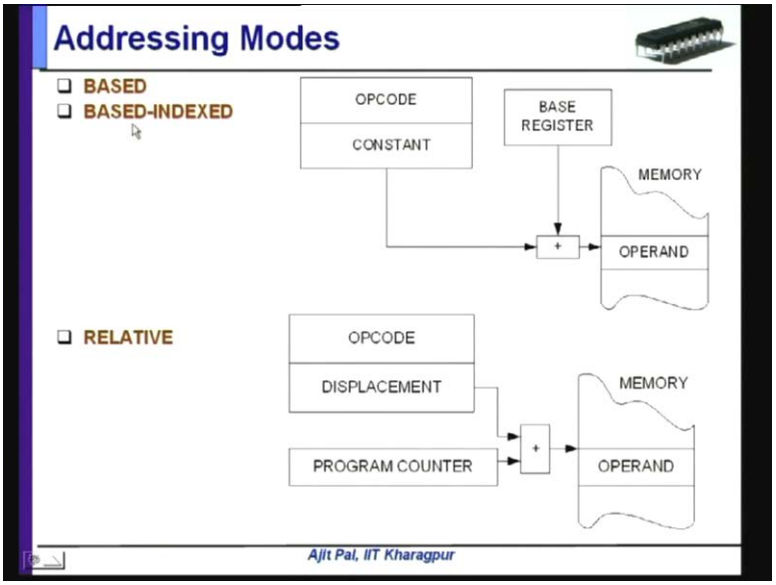


Then comes the paged addressing, sometimes we use the concept of paging and particularly to reduce the size of the instruction we use different pages or you can use a page register. So, in such a case an offset is provided and then direct page register is used this two together provides the full address where operands is available. So, this is known as paged addressing that is done with the page of special purpose register which is available in the as part of the processer. Then, index addressing is used to facilitated

implementation of data structures like a q and as you can see we use a special purpose register x.

So, this register gives you the upset value and another register base address is provided as part of the instruction this together is used to generate the address and you get the operand here, this is very useful for accessing ray implements. So, the index register will provide that base register, this base address is the starting address of clearly and then index register will join to different elements of the ray. So, base of this addresses fixed and then you can change the index register value to point to different a elements. If its 0 its point to the 0, 8 elements, if it is one it points to be first element like that, so in this way you can access and a ray a very convenient by index addressing I said that this is useful implement some data structure like r a q and so on.

(Refer Slide Time: 47:20)



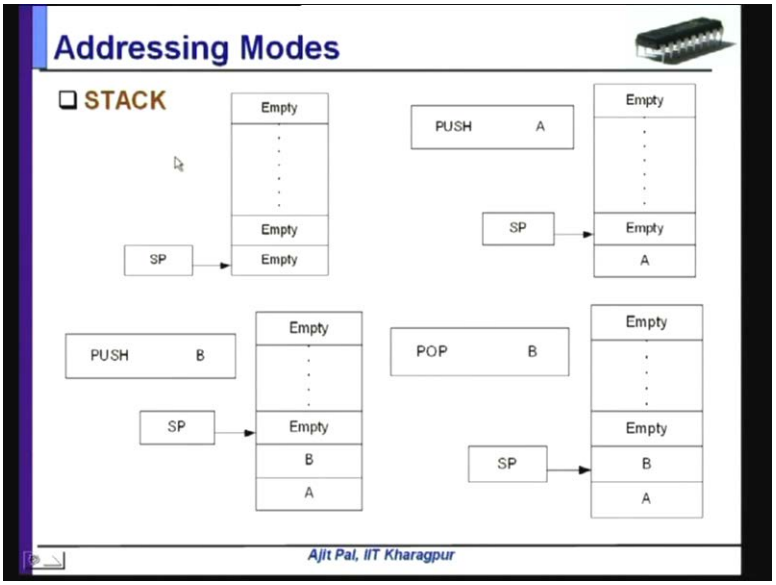
Then comes the based addressing based addressing, where a special purpose register is none as based register is used and the constant is added to that to point to the operands. So, this particular earlier what have you seen that register value was changing, now this value will change, I mean this base register is added with the constant to generate operands at this.

This is particularly used in situation like re location, now sometimes you have to do relocation in a multi user environment. Then, you require best addressing to change the content of the based register to relook at the instruction to different parts of the memory.

So, then you can combine based indexed that in this you can have indexed register and a based register both of them can be used simultaneous to have based in addressing.

Then, you can have relative addressing relative addressing is you are essentially providing an address with respect to some register particularly program counter with respect to program counters. So, op code and the displacement is provided as part of the instruction which is added with the program counter to generate the effective address and which is the address of the operand. So, this is known as relative or when you call the program counter register is used then you call it PC relative.

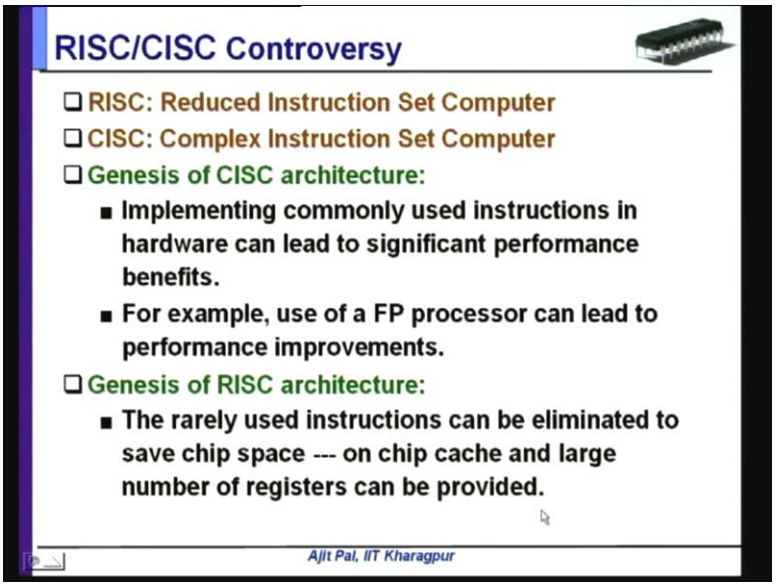
(Refer Slide Time: 48:52)



Then, there are various addressing mode related to stack, so you have got a stack pointer which is pointing to the bottom of the top of this stack and you know whenever you post some element and it is decrement and that is stored. So, you can perform push a, so you can sit this pointing to it is static point requirement to point to the next locations and value of a is stored here. Then, another push you can see it is again implemented and top of tag his having the much holding the value b. Similarly, you can perform pop operation, so with the help of the push and pop operation you can store or load operands from in a memory.

So, whenever you do pop operation you do can read it from this location then it will point to next location. So, in this way you can see stage you can do the help of this two instruction push and pop.

(Refer Slide Time: 49:57)



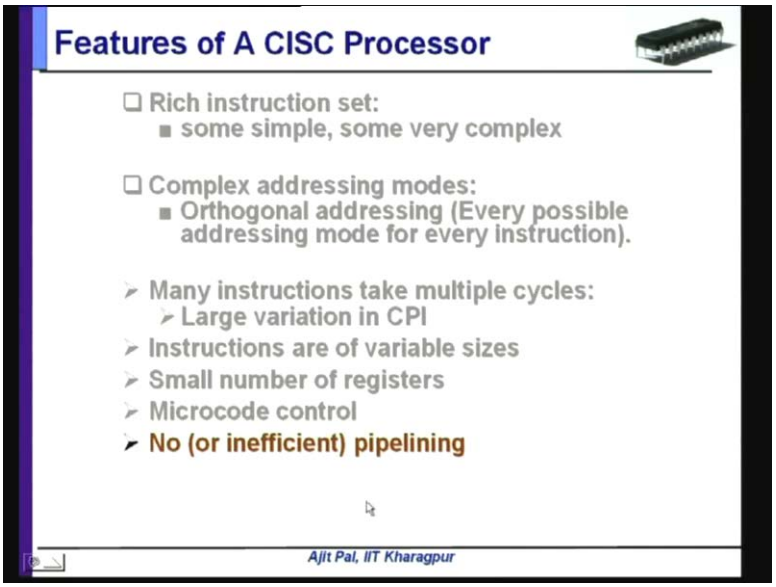
So, these are the various addressing modes, but our discussion will not be completed without the mentioning about RISC and CISC controversy. I have already told that there are two possible architecture, one is known as RISC instruction set computer is architecture another is complexes instruction set computer or RISC architecture and what is the genesis of a CISC architecture. So, implementing commonly using instruction in hard ware can lead to significant performance benefits that is means what you are trying to do you are trying to minimize the semantic gap with high level languages. That means whatever the high level language instruction can perform you are trying to do with the help of machine language and instruction by making the instruction more and more complex.

As a result single high level language instruction will lead to very few machine language instructions may be one or few. So, this semantic gap between the high level language and machine language reduce that is the basic idea behind complex instruction set architecture. So, what is happen as the as the various program in language involved and various operation which are provided statements can performed in high level languages those are implemented a complex instruction. So, that is the genesis of the this architecture for example, use of a 14 point processer can lead do performance improvement and they will be complex 14 point operation which can be specified with the help of complex instruction.

On the other hand, in case of RISC architecture the rarely used instructions can be eliminated to save chip space on chip case cache and large number of registers can be provided. Here, there is a story behind it particularly IBM did some study on various type of instruction which are used by the compiler. You know you are writing a program in high level language then compilers is generating the machine language code. So, which instructions are used by the compilers different compilers, it was found that a lot of stimulation works were carried out by IBM before. It was found that very complex instructions are really used the other compilers use only simple instruction for generating the object code.

So, what was decided the instruction which is really used by the complier try to implement them. So, they remove that complex instruction leading to what is to known as RISC processer. Here, complex instruction where moved and only a simple instruction which are commonly used by the complier are retained and that is the genesis of RISC architecture.

(Refer Slide Time: 53:09)

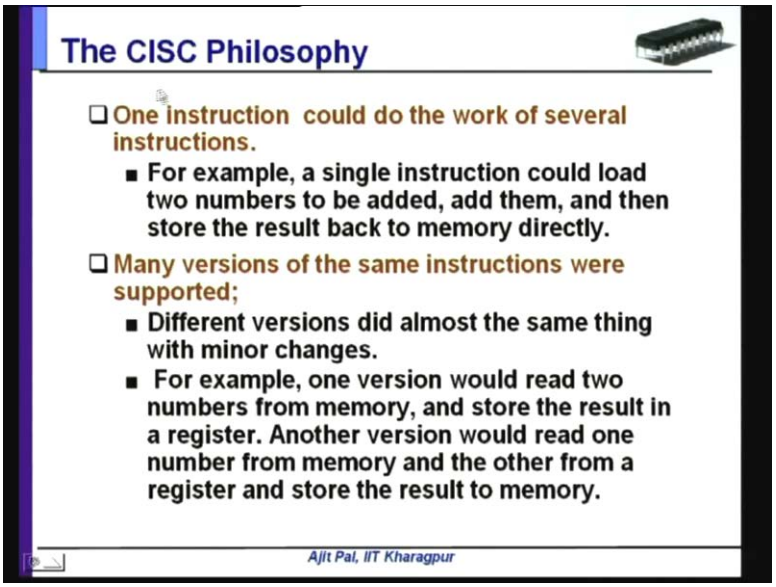


These are the features of rich architecture rich instructions set, some simple some can be very complex addressing modes. Then, many instruction taken multiple cycle as I told they can be the instruction format can be complex and I mean variable and there can be a large variation of CPI, some instruction may take 1 cycle, some instructions may take 2 cycles. They may take a 10 cycles depending on the complexity of the instructions, then

instruction of variable sizes I have told then small number of registers because it uses primarily a resist.

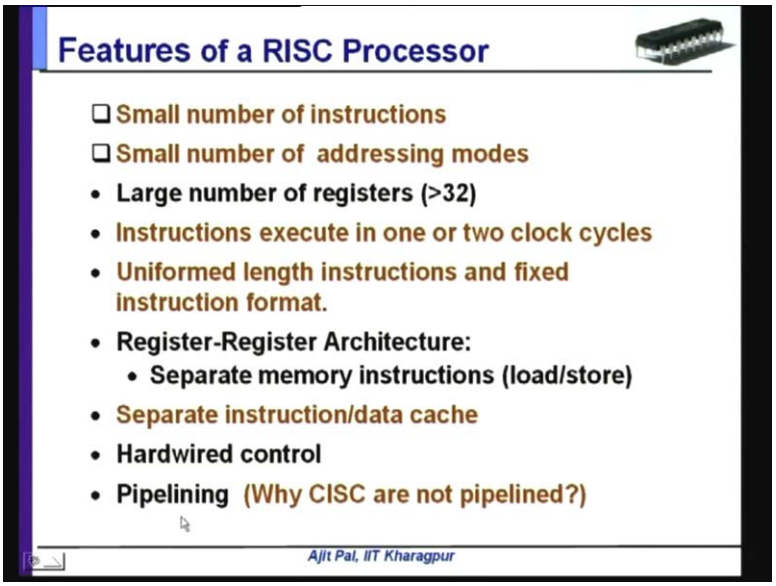
It is a register memory architecture, so one operands in the register other operands is from the memory. That is why you require small number of registers and micro code control; you know that the control unit is implemented with by micro programming micro programming control unit. Instead of hard ware control unity you may have studied there are two ways of implementing control unit one is hardware control unit and another is micro program control unit. Normally, this CISC processer use micro program control unit and it is very difficult to implement pipelining in CISC processer.

(Refer Slide Time: 54:33)



So, obviously you one instruction could do work of several instructions because this quite complex and there will be many variation of RISC instructions because it involves memory and register.

(Refer Slide Time: 54:47)



On the other hand, the RISC processer have small number of instruction small number of addressing mode large number of register this is one very important features of RISC processer. They requires very few large number of register and instructions executive one or two cycle clock cycles you will find later on when I shall discuss the MIPS processer architecture, you will see all the instructions will require a single format. Also, it will involve only 1 cycle or 2 cycles for these executions, then uniform length instruction and fixed instruction format as I already told and is essentially register, register architecture where you require separate load and storage instructions.

That is why this called load store architecture and separate instruction and data cache this is a very important feature. Later on, I shall discuss on little more detail, you will require two separate cases cache memories one for instruction one for data instead of a single a cache memory. Then, the control unit has to be hardwired instead of micro program control unit it has to be hardware control unit. So, these are the features and finally, I shall discuss in detail later on when we shall discuss about pipelining.

You will see the RISC processer can be RISC instructions can be very easily pipe line compare in contrast two CISC processer. So, implementing pipeline architecture for RISC processer is very difficult which can be very easily done in case of RISC processer. So, with this let us come to the end of instruction set architecture in by next

class I shall continue our discussion on RISC, CISC, and also consider a representative processor architecture that is the MIPS processer architecture in my next class.

Thank you.