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

Lecture - 21
Control Hazard, Branch Prediction

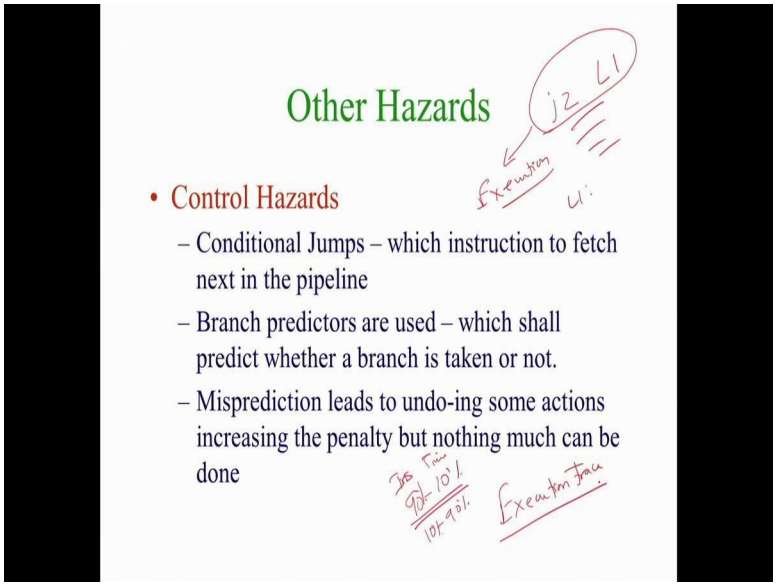
So, good morning. So, last class we saw about data hazards, they are what could happen if the program is run as such without any you know controlled by the hardware, if it can lead to a you know wrong result and that is what we termed as hazard. And all the hazards that were present and for which we give solutions were because of data. If you look at the instruction right there is an opcode and operand and the all the hazards that we that we have talked of was because of the operands so that is why we call them data hazards.

The next thing is next type of hazards is what we call as a control hazard what is the control hazard. Suppose, I have an conditional jump right you have used lot of conditional jumps hopefully now yes or no? Hello?

Student: Yes

Ok. So, what would happen in the context of a conditional jump I fetch the instruction when I am decoding the instruction and that is the point where I will be knowing it is a jump or not right. And then what could happen I do not what would be the next instruction to fetch do I know because the conditional jump is taken then it will be some other instruction that I need to fetch. If the conditional jump is not taken then it will be the next instruction that is following the conditional jump. So, when I fetch a jump instruction I will never know whether it is a what is the next instruction to fetch, correct, should be the next instruction, should it be the instruction that follows or should it be the target instruction. So, jump on a condition to some address right that is called the target address. Should it be the instruction stored in the target address or should it be the next instruction, this I cannot find out when there is a when you see a conditional jump. When will I know that I will know that there is a the target address only when the previous instruction finishes its execution stage.

(Refer Slide Time: 02:37)



When I have jump on zero to some label L 1, and then I have several instructions and L 1 only. So, the next instruction to be fetched will be known only when this particular instruction finishes execution; till that I cannot even fetch the next instruction. Are you able to follow? So, in contrast to the you know the true dependency that we talked yesterday, we are talked about true data dependency raw where I have to wait only at the data fetch stage and that also I will get by I will get it fast because of the operand forwarding. But in this case, when I see a conditional jump, I cannot even fetch the next instruction. So, I have to wait for at least four cycles, assuming each one will take one cycle I need to wait for at least four cycles till I get that.

So, going back to our context of what we did yesterday there was an instruction fetch unit which was bring up six instruction a set at a time. Imagine one middle instruction being a jump right then the other two fetches that we have we are fetch these instructions at a time the remaining two's instruction cannot even be fetched. So, when I am fetching a set of instruction and the moment, I see a jump the remaining instructions may become useless may become useless may be correct or may become useless. So, the pipeline essentially is stopped or it is stalled because I see a conditional jump and that is why this hazard the conditional jumps are called control instructions, and that is

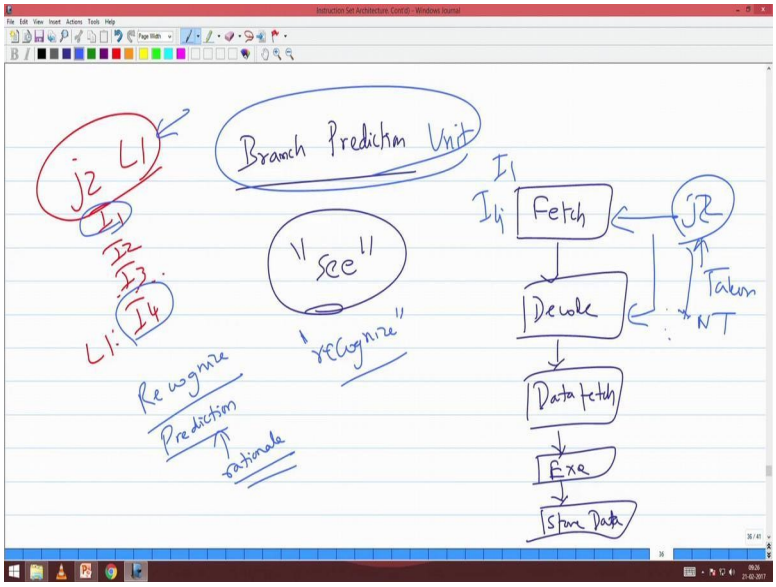
why this hazard is called a control hazard. Are you able to follow?

Now what can we do every time I see a conditional jump instruction if I am going to wait for three cycles. And in a program you know there is a I think I have mentioned this in the previous class, there is something called a 90-10 rule 90 percent of the instructions. It will work only for 20 percent of the time will take only 10 percent of the time there will be one core nucleus part of the program, which will take 90 percent of the time. So, there is an opposite 10-90 ninety which implies another 10-90 rule. The 10 percent of the instruction will take 90 percent of the time.

So, how will 10 percent of the instruction take 90 percent of the time there will be some loops the will be conditional control statements which will make it repeatedly execute. So, you take any program except hello world right every program will have an if else statement, while statement, for statement right any sensible program which does even simple finding the maximum of two number will have an if statement. So, there will be lot of control statements and inside a program. So, when I take the profile or what you call as the execution trace, please note this word this is a very important word, execution trace of a program, execution trace is not the assembly program this thing.

Execution trace is what are all the instructions that are executed from the start of the program till the end of the program. So, there will be some 10 percent of the program which will be repeated lot of times. So, I start the program which address is executed which instruction is executed next, next, next, next, next, next till the end of the program. So, you will see many instructions repeatedly executed. So, when you take the execution trace of the program, you will see lot of such conditional branch; and for every conditional branch if I am going to wait for three cycles then it becomes a nightmare my performance will go for toss. So, I have to do something about this and that leads us to what we call as a branch prediction.

(Refer Slide Time: 06:45)



Now, we will go back to this mode please take notes. So, what is branch prediction? So, let us take this there is an instruction fetch stage, there is a decode, then decode data fetch, execute and then store data. Now, when I see a branch, I want to go and predict what, what will I predict about branch here will you get married or not married or what you will what will you predict about a branch, whether it will be taken or not taken correct right. So, I am seeing a conditional branch, when I see a conditional branch I will put see with quote, unquote and how do you implement this see it is a big story I will say that.

When I see a conditional branch at the fetch stage immediately I should go and predict hey this fellow will be taken or this fellow will not taken. What you mean by taken, the branch condition will become true and what I need to fetch next is not the instruction that is following it, it should be the instruction to which it is going to jump to it. Let us take example here jump one is at some L 1. So, I have instruction one, instruction two, instruction three and I have L 1 here which is instruction four. So, I can have many things here.

The moment is see j is at L 1, I will have a extra hardware which is called a branch

predictor or branch prediction unit. And that branch prediction unit will tell me this jump

will take for sure, that is the moment it says it take then what should I fetch next I should fetch I 4, because it will take. If this branch prediction unit says it will not take then what should I fetch, I should I 1. So, the decision of whether the branch is to be taken or not taken where should I make that decision in the fetch stage itself then only I can convey the decision immediately to the program counter which will start fetching the next instruction. If I make the decision that jump instruction came here, if I make that decision when decision comes somewhere here then I have to stall the pipeline. So, the decision has to be taken where in the fetch stage itself. The moment some instruction comes I should somehow see recognize, I should somehow recognize that it is a branch instruction and then I have to predict the hardware the branch prediction hardware will predict whether this conditional jump instruction will be taken or not taken.

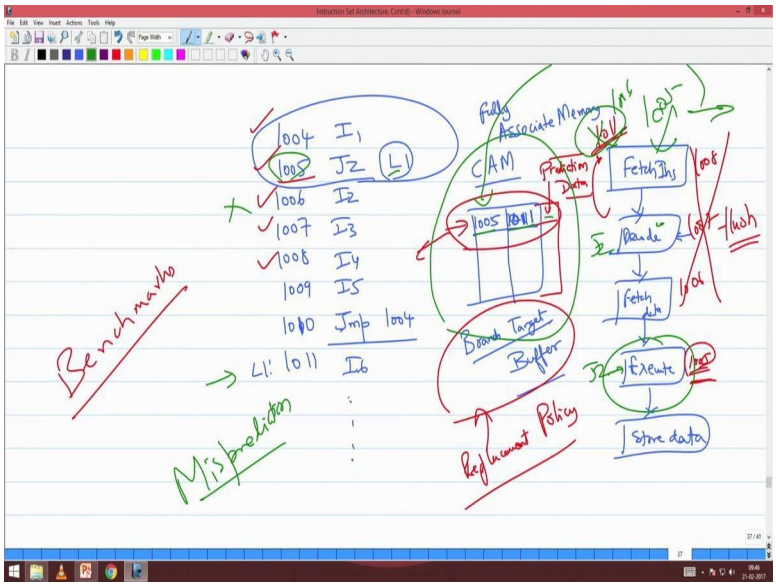
And if it is going to be taken if this fellow says yes it will be taken then I should fetch I 1, so the next instruction I 4. So, at this point I will predict suppose I say taken when this instruction actually comes here I 4 will be fetched. If this fellow says not taken then when this instruction actually comes here, I 1 will be fetched, correct, all followed. So, now, what should we, what are the steps now number one should recognize that it is a jump instruction and then I should do the prediction. And most importantly for this prediction I need a rationale I need some rationale for doing this prediction. So, these steps we will explain in detail.

Now, the most important challenge here is that, how will I recognize it is going to be a conditional jump instruction. If it is unconditional jump, I do not have any issue. The problem comes only if it is a conditional jump instruction as of now. How will I recognize if it is a conditional jump instruction before even decoding the instruction? I need to decode to find out it is a conditional jump that is why you have the stage called decode stage. How will I even recognize that it is a jump instruction conditional jump instruction before I decode that instruction? Is it a important question, yes or no?

Even before decoding the instruction somehow I should find out that it is a conditional jump how will I know. I cannot decode at the fetch stage forget it and I cannot do any special decode to find out if it is a jump instruction and all. Any extra hardware I am already that fetch is like a monkey, I am going to fetch six fellows I have to resolve

whether there is data dependency so many things I have to do there let me just carefully see please understand this very, very crucial, very slowly I will do this.

(Refer Slide Time: 12:17)



Now, let us give some address 1004, 1005, 1006, 1007, 1008. Now, we have some instruction one then I have jump z, some L 1. What is this structure equivalent in c?

Student: loop

Loop, what loop? It should be while loop or a for loop, not a do while, because the condition is checked at the beginning right followed. So, should be a for loop or a while loop. So, you do some condition check here then you do a comparison here; and based on that idea you again jump to so L 1 would be somewhere here. So, let me change (Refer Time: 13:25).

The first time the instructions are coming. So, let me again put this, we have to repeat this n times imposition, fetch, decode, fetch ins decode, fetch data, execute. Now, first I 1 will be fetched, I 1 will be decoded, I 1 will be I 1 data will be fetched, I 1 will be executed right now jump will be fetched first times. So, 1004 will be fetched when 1005 is fetched please understand that will not know it is a conditional jump, but the moment I

come to this decode stage here I will understand it is a conditional jump right. And I will

also understand where it has to jump. At the decode stage I will know that if condition is true I need to jump at where L 1 that is 1011.

At that point I will create what you call as a content addressable memory, yesterday we saw that right can is also called associative fully associate memory both are same FAM or CAM are synonyms fully associative memory. In this memory, I will go and make an at the decode stage; first time I do not know that is a jump. When the jump reaches the decode stage, I know that it is a conditional jump; I am only handling the conditional jumps; I am not unconditional jump, there is nothing to predict, any way it will be taken. You all know what difference between conditional and unconditional correct. This is an unconditional jump there is no condition involved if it comes here it will take, but only the conditional, I have a doubt whether it will take or not.

Now, the first time I see this conditional jump, I will store this 1005 with L 1 the target address in this case this is 1011 correct. I will store 1005 with 1011 this is actually also called as branch target buffer. So, I will store 1005 and I will say it has to jump to 1011 then I will be executing I 2, I 3, I 4, I 5 again I will come to 1004. Next time, when I am coming to 1005 every time when I am coming when I get a new address in the program counter, I will go this CAM and say hey is this address stored in new. If this address is stored in new then I know that it is a conditional jump right. Are you able to follow? So, every time the decode unit decodes a new conditional jump instruction; new in the sense the first time the conditional jump instruction is executed in this program immediately that pc the address of where it is stored along with the target address L 1 that it will store in the branch target buffer. This is a content addressable memory.

Every time when I am fetching an instruction I will go to the fellow to CAM to the BTB branch target buffer and I ask, hey is this stored in new if that fellow says yes then I know it is a conditional branch. I not only know it is a conditional branch, I will also know if that branch is taken where I should jump. So, the content addressable memory as I explained you yesterday will give you not only the address, but it will also give you some data associated with that query element, yesterday I discussed the content addressable memory in the context of memory aliasing same thing here.

So, what will happen now let us quickly and simulate this 1004 comes. So, first time 1004 comes then 1005, 1005 gets entered here. Then 1005 comes to this place for execution. At this point, first time what will happen 1004 comes, 1005 comes, then 1006 1007, 1008 it will come in this direction. First time it will go here and it will execute when the branch is taken if the branch is taken then all that I have fetched when 1005 is here I would have fetched 1006, 1007, 1008 or bunch more lot more instruction I could have fetched. Now, if the branch is taken at this point, then I have to load all these things are wrong, I have to flush the pipeline this is called flushing right.

So, first time itself if this jump is going to be taken then what will happen this is 1005, this is I come to this stage and this is where I know whether it is going to be taken or not by that time I will have 1006, 1007, 1008 already in the pipeline. Now, if I say it is going to it is taken that means, fetching of those instruction 1006, 1007, 1008 are all wrong. So, I have to go and handle that. If when I reach this point and I find that it is not taken then I can basically continue. Then all the processing I have done for 1006, 1007, 1008 does not go as a waste. You get it, are you able to follow right? Now, that means, along with this target address I will also have one prediction data, this prediction data will tell me whether the branch is going to be taken or not taken, and that is where I am building some intelligence into this.

So, at the first stage, so let us assume again 1005 is going to be executed, when 1005 is getting executed it will go an ask this CAM hey is it. So, it will ask this query is 1005 stored in new, if 1005 is stored in it then we know that it is a conditional branch. We also know if that branch is going to be taken which target address I need to go, and I will also know whether that what is with high probability that the branch will be taken or not taken. I will have some prediction. If the prediction says yes it should it is going to be taken. So, immediately I will go and pump the program counter to be 1011. So, the next instruction that is going to fetched would be I 6. If the prediction says it is going to be taken and if the prediction is correct then what happens I have saved three cycles. If the prediction is wrong, I waste three cycles, correct. Are you able to follow yes or no?

Similarly, so this is the entire scenario. So, what we have seen now is how will the fetch stage recognize that this is jump conditional jump instruction. See that it is a conditional

jump instruction. It basically uses a content addressable memory to solve this problem. Are you able to follow? Yes. So, now, the challenge is that this content addressable memory will have limited number of entries now what is size of the content addressable memory that I need to put so that you know all my branches are taken care of. If the size is limited then what will happen then you will not have enough space to store if you have hundred thousand branches and you have only ten entries then we are in a (Refer Time: 21:51).

So, how do you make a decision of what should be the you know size of the content addressable memory. When you design an architecture what should be the number of entries in a content addressable memory that itself is a very, very important decision that architecture decision you want to make. So, when you study and course on advanced computer architecture for your M.Tech elective you are expected to be taught on what how do you design CAMS. How do you design branch target buffers, what would be the size of this branch target buffer, how do you come out with this size, [FL] night dream I got 1024 entries can I put? No, because CAM why cannot I put large content addressable memories because they consume lot of power the chip will burn, so and it will also consume lot of area. So, lot of decision making has to come and that is where benchmarks play a crucial role.

So, there are different benchmarks and these benchmarks basically when you run them it will it will tell you if I keep the branch target buffer as 10 entries what will be my performance. If I make it 20 entries what will be my performance, 30 entries what will be my performance and so on and so forth. So, it will kill you performance. Beyond some point you will see that area is too large below some point that the area is very less and, but your performance goes for a toss.

Please note that it there are many, many branches and ultimately your 1005 gets out of this cam for some reason right. Then the next time when you bring 1005 it will not be recognized again this whole recognition problem has to restart. And this branch target buffer will also need to have a replacement policy. What is the replacement policy? Replacement policy is that I need to know if all the entries are full and a new entry is coming which entry should I evict to put this new entry I should make a decision of

which entry should I evict. So, the we will talk about replacement policies in the context of cache in great detail, but there are lot of replacement policies, and all these things I will cover post quiz two when we do memory management of caches, cache organization in caching.

Student: Sir

Yeah.

Student: sir

Yes.

Student: I did not get how we are able to get that gather conditional jump and else statements that is (Refer Time: 24:35)

Yeah, see the first time this 1005 instruction comes in into your fetch instruction unit fetch unit, it will not recognize that it is a conditional jump. Then it the 1005 goes through the decode unit decode unit will recognize, it is a conditional jump. Then it will go to the content addressable memory this part of the content addressable memory and it will go and store 1005 and the target address 1011 along with some prediction data default prediction data. It will store it in the content addressable memory. Now, every time some instruction is fetched as and when the instruction is fetched, the value of the program counter the address where that instruction is stored is also given as a query to this fully associative memory. So, the next time the 1005 is fetched here when the 1005 is coming and falling into this unit that same time that query will also go to this content addressable memory. Hey 1005 is there are not, I am fetching 1005 it is coming from memory I am also asking the CAM is 1005, there or not. This cam will say second time it will say yes it is there then it will also have give not only not only, it will say yes it is there, but it will also say that if this conditional branch is going to be executed taken. Then this is the target to which is need to dumb, and I need to also say this is the prediction that this you know this conditional branch will be taken or not.

So, two data come at the fetch instruction stage itself. So, the fetch instruction unit in addition to all those it has been doing in the past that we have described, it will also say now this is a conditional jump instruction and this is the target address and let us look at the prediction oh it is predicted to be taken. So, the next instruction I should fetch is not 1006, but should be 1011. So, when the jump instruction goes to the decode stage at that point itself, the next instruction that will be fetched will be 1011. But suppose it says this is 1005, it is a conditional jump instruction, but the prediction says it will not be taken then it will not do anything the jump will come here and then whatever you are predicted so the prediction is not taken. So, 1006 will be fetched right. Do you understand?

Now, what will happen jump eventually will come here to the execute. At this point, you will realize actually what is going to happen, something has been predicted and something is actually going to happen. What actually happens is different from what is actually predicted, if there is difference then what we need to do, we have to go and flush out the pipeline, and restart the pipeline with the correct one. If there is a miss prediction, this is what we call miss prediction, when will I realize that there is a miss prediction when I come to this execute stage. What I have predicted and what is actually happening is different I said not taken, but it is actually taken. So, all the three instructions that are there in the pipeline has to be flushed out. What I say now I said taken, but it is not taken so again so these when I realized that there is a miss prediction, I will go and undo what I have done in the past (Refer Time: 28:26). So, I have flushed the pipeline and restart it. So, in the miss prediction I have a penalty in this case of three cycles at least. you are getting this? So, this is how the control hazard is handled.

So, if one minute if I am not going if I am going to reduce the miss prediction rate then I am going to get excellent performance right if I am going to have lot of miss predictions then my performance actually goes for a toss, are you getting this? So, the ultimate objective for me is to go and reduce the miss prediction rate and that is what we will work on. Ok done.

Thanks.