Computer Architecture
Prof. MainakChaudhuri
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

Lecture - 15
Basic pipelining, branch prediction

Let me recap where we finished last time.We talked about these two level branch predictors, where we have first level table, which can have multiple entries or can I just say a single entry. And the second level table also can have. Well the second level table usually has multiple entries.And you also came up with taxonomy of branch predictors, where you can call the first level table as global per set or per branch, depending on how you organize the first level table that is in.The first level table you can just add one single entry of recording histories that you called a global table.If we have a first level table such that a set of branches, takes one entry, then that would be called per set first level table.And you could have a gigantic first level table where you have all the entry number of entries equal to the number of branches.

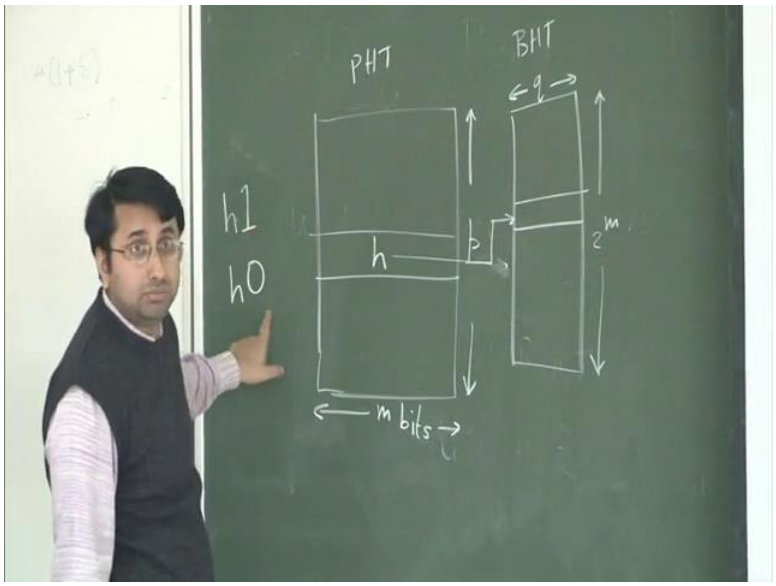
Similarly your second level table could be global. Here global means, all the table entriesit shared by your histories. So, as such there is nonspecific table for a particular history. So, each history gets the access upon single table. Similarly you can this type of per set table per branch table get in tothisarchitecture in more detail, and the question is how do you update this tables. So, there are two most one is static, which we actually do not update the compiler generates certain things, and whenever the program starts, you have some way ofslowly the compiler generatedinto this table or actually you can actually learn what is happening and adaptive be updated. And we have alsotalked about one special case that is g share, sowe discuss again later today, and you can combine multiple of these. So, we see that also.

So, in general a direction predictor, will see it as a function which takes a branch pc and a global history register, and it is your outcome either 0 or 1. So, these one is essentially encoding your history over sliding in of all the registers. And the way it is implemented is that, if can I multiple direction predictor components i r showing three of them at a branch. So, each of these will give you a 0 1 outcome, and you have a branch pc and a global history register, you to pick plus selector which you select all of these outcomes at

based on whatever values of these two things. So, ultimately what you will get of course, is a 0 1 outcome. So, look at today how exactly what kind of selectors it may design, how we actually war. So, it is a basic idea here that I have bunch of direction predictors, and I have a selector which selects one of these predictors. So, will not try to come up with any function of these as such, which is select models.It is possible to actually design functions, sothat you actually attach weights to which of these predictions, and come up with the function that combines all these things, and gives you a binary outcomes that is possible also, will not get to that

See if you open up one of these direction predictors. So, they look like exactly same as this.They take the branch pc and G H R andequal to 1. And the selector takesthis3 dimensional predictor of 0 and 1s; that are coming out of these direction predictors, and it also takes the branch pc.The pc of the branch instruction that kind of predict, and the global history register the current content of that and (( )). So, what is DPN? So, we talked about this last time.We have the first level;we saw the pattern history table. So, that essentially stores your history patterns. and what you get out of this is a history. So, table will talk about that today, and the second level table which is a branch history table, takes that history as input, and what comes outis a 0 1 output, because this is the description of the pattern history table, which takes the branch pc and global history register, and it gives you the branch pc and the history. So, the pc just passes through.And the branch history table takes the pc and the history, whatever is output here and gives you 0 1.

(Refer Slide Time: 05:51)



So, what does the pattern history table look like.It is a table of histories. So, this notation says that each history is m (( )). So, we are having a sliding window of m bits. We are looking at the last m outcomes of the branch. And how many entries do I have p p suggest. So, is this quotation clear to everybody. So, the pattern history table will look like this.There are p entries here, and this is m bits. And this one is going to store some history.It maybe history of particular branch, itmaybe the combined history of multiple branches.For example, if you are if you storing a global history, then you would be essentially storing the history of all the branches, whenever a branch shows up, it will store the of the branch here, and it would be shifting in this direction, sowhenever we see a new branch, we will shift this by one position and shifting the new branch as outcome; is it clear, the pattern history table.

Similarly, the B H T is the table of saturating counters; each one is q bits white, and it has to have 2 to the power of m entries, why is that.Why is this relationship.IfI have n bits of history here why should we have 2 to the power of m entries, even if you have forgotten can you, rein built that correlation why is that. Total possible patterns, why is that important; that is what I use the index. I use one of these entries to index, that is why I need2 to the power of m entries here right this is m bits long, so I need 2 to the power of; this is q. So, is this anatomy clear to everybody. So, this very generalizeddirection predictor, and what was I saying is that we have (( )) and selected we should be choosing one of the predictions.

The bimodal predictor is a special case, where the P H T is not. There is no P H T. So, essentially what happens is that, the B H T takes the branch pc, this is not existent and gives you a 0 1 outcome, P H T does not exist actually. So, does anybody remember what the bimodal predictor actually does. So, before that, so I say that these are saturating counters, what are they counting, does anybody remember, counters count right.What are these saturating counters count (( )) is it. So, given a particular entry here, I have a corresponding counter here right, what is that counter count.He says that it is a number of types this particular pattern is occurring, is that correct. How do Iupdate that counter, does anybody remember.It always increments, it always decreases, what happens to the counter.Somebodysorryandotherwise re decrement

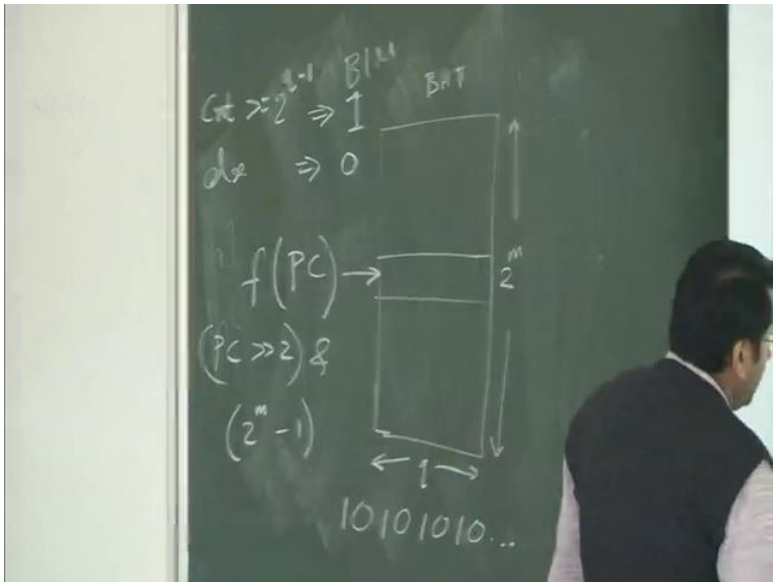
Student: Decrement.

So, now, can you tell me what if I look at the content of this counter at any point any time, what is it tell me.

Difference between taken and not taken, going (( ))

Well after that pattern. So, it is telling me the difference in the number of times, and seeing h 1, and of course, 2h 0 if h is the content of this history. Is that correct.

(Refer Slide Time: 10:52).



Say in other words is this still telling me the probability of,let us say getting a 1 give it h. So, try it if I want to extract that from this. And since it is a q bit y counter, I can only

say anything about these particular operands in the last 2 to the power of q occurrences of h. I cannot tell you really what happened particularly,that is not possible. So, out of 2 to the power of q occurrence last occurrence of h, I can tell you how many timesone happened, minus how many time 0 happened; is that clear. So, the bimodal predict does not have a P H T, so what does it do. So, bimodal predictor, it has the B H T. What is this function of pc, what function make sense.Now last time; exactly yes.You will remove the last 2 bits, because they are always 0, and then take the remaining m bitsthat the lower m bits. So, essentially a pc shifted by 2, and even 2 to the power of m minus 1. So, any m bits exactly. So, is there any justification of taking a lower m bits, does anybody have an answer for that.Why do not I pick the upper m bits from pc.

Student: there will be problem for new instructions. So, they will same for two branches, there is a very high likelihood if I pick, the upper m bits they maybe same for two branches.Whereas, if I pick the lower 1s since they are changing most right, the chances are that the branches in a. So, essentially what I am saying (( )), I am saying that I can guaranteemapping on this tableif I, which is how many instructions, 2 power m instructions. 2 to the power of, out ofin a local 2 to the power of m instructions; however, branch is you giving, I will give you the here, there will no collision;that is guarantying actually. And that essentially determines your over, what size of region you can accurately predict branches right. So, whereas, if I pick the upper m bits. So, or if I moved this m bits towards the upper side of pc, I will startlooking this property gradually. So, chances are that two branches may collide only single unit, which is not true that I do not want that.

So, otherwise here what this counter is counting, essentially that given this particular branch, how many times it has been taken, how many times it has been not taken.And looking at this sounder I can know in which direction is biased. So, that is why it is called a bimodal counter, because it works great for branches which have bimodal behavior. So, either highly biased towards taken, or highly biased towards not taken. if the branch is somewhere in the middle you are likely to make mispredictions, so it is better.For example, if you have a alternating branch which is like this 1 0 1 0 1 0 1 0 etcetera right, what will be the prediction accuracy for this branch.

Student: 50 percent accuracy.

Is it 0 percent accuracy or 50 percent accuracy?

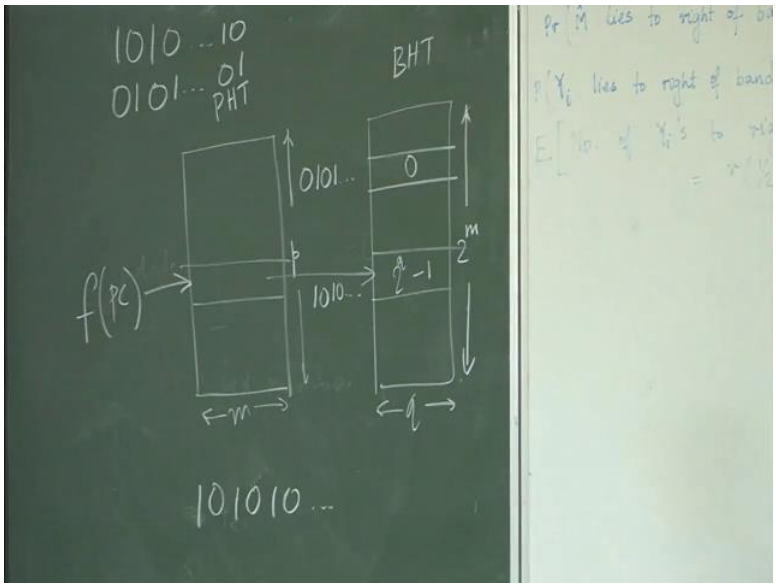
Student; depends on the initial value

I visualize these to, let say 0, then 50 percent. When do you get 0 percent

Student: if you just look at the previous value, and predict with that is on the (())

ι forgot to tell. I taught that was essentially what is the prediction function? If my count is bigger than equal to 2 to the power of q minus 1, I predict 1, else predict 0. So, if I am up the midpoint, I say it is 1, otherwise 0 why is the reason is that, whenever I get a taken indication, I increment the counter,whenever I get a not taken indication I decrement the counter.

(Refer Slide Time: 15:57)



So, when do I get 0, if I initialize the (( )) to given this particular register, right. So, if I initialize the alternate 2 to the power of q minus 1.

Minus 1 done yes; 2 to the power of q minus 1 minus 1 then I am going to get all wrong right. So, depending on initialization you maybe 100 percent wrong, or you can get 50 percent wrong. So, the best thing that you can do is 50 percent correct.It cannot be better than that wherever you initialize you can change that actually. So, that is the role bimodal predictor. So, let us take a look at now the different predictors. Before I go there I just want toname it, and take this alternating branch little more, supply of generally two

level predictor that you talking more you have P H T. So, if I take that branch orwhich has its alternating pattern what will happen.Now till I get anything better in. So, this index functions here. So, although here I mentionbranch pc and g h r. in most cases will not be using the g h r at all the branch pc only. So, here then if a function pc. So, in this case this function is going to take the lower p bits, lower long p bits, after removing the last 3 bits. So, what happens to this 1 let us say a branch has this particular history.Any hope of improving here, compare to bimodal predictor.Forget about the first amount comes.Assume that we are state in state, the history is filled up yeah.

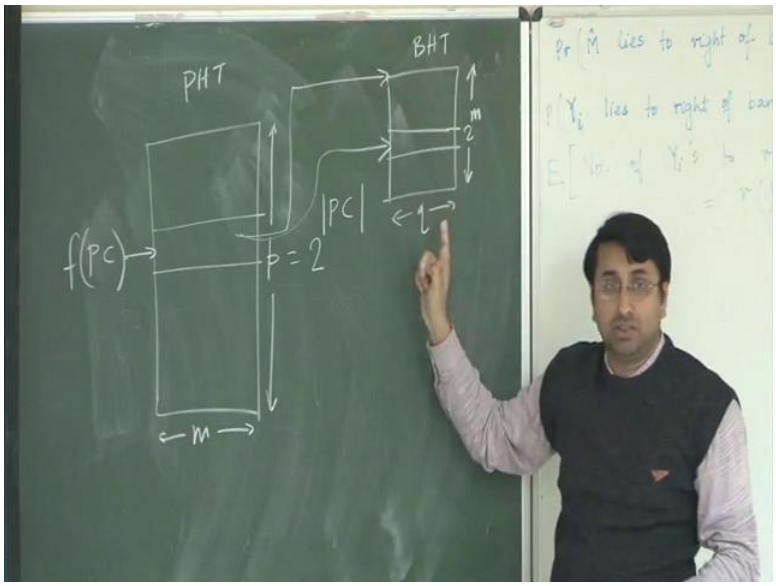
Student: wherever there is a Pythagoras fraction 0 1 0 1 it will be…

What is the sliding m bit into here. So, how many possible patterns are there, different m bit patterns 2.What are they.Starting from 1 we will assuming anything about m or even.Let us assume something right, let suppose the other pattern is. These are two possible patterns, if I slide in m bit to window over this history

Student: sir what will map to different location B H T.

So, first of all both of both of these will reside here right, in the same entering the P H T. So, as I see the branch, sowe make to be either see this pattern, sitting here, or we see this pattern sitting there, so then both of these. So, these two patterns will get two different counters here. So, let us suppose that this counter gets attach to, let see which one is the bigger value this one right. So, this one will have the smaller value. So, let us attach this one here, and let us suppose that this onewill bigger value over here. So, what will be the contents of these two counters, because both have wrong (( )). Is it, upper one is, what will the content of this one. Say the static state what I will be using here0 right 0, and this is2 to the power q minus 1 right. So, in the steady state once I have lost the midpoint, I will always give you the correct prediction for this branch, all the time, in 100 percent accuracyright.Is it clear to everybody. So, I will just go through this taxonomy. So, first one waspap, which essentially means that you have a per branch pattern history table, and the per branch B H T. So, let us see whatthat looks like.

(Refer Slide Time: 20:00)



So, this is my P H T. I have m bits of history, and I have P H T here, where small p is equal to the number of branches in that program.Now clearly give given your, if you put yourself in the position of a processor designer, you have no idea about what programs that are going to run.Probably there are there are would be many programs that going to run in this processor.Since actually not possible to escape its small p. So, what you will do here if you 1 create a pap predictor, what is small p, because it going to be hard coded predictor right.It is not that you know before running a program you can figure out what small p and you can increase your size of prediction; no not possible actually. So, what is this what will you do.What is the constant.

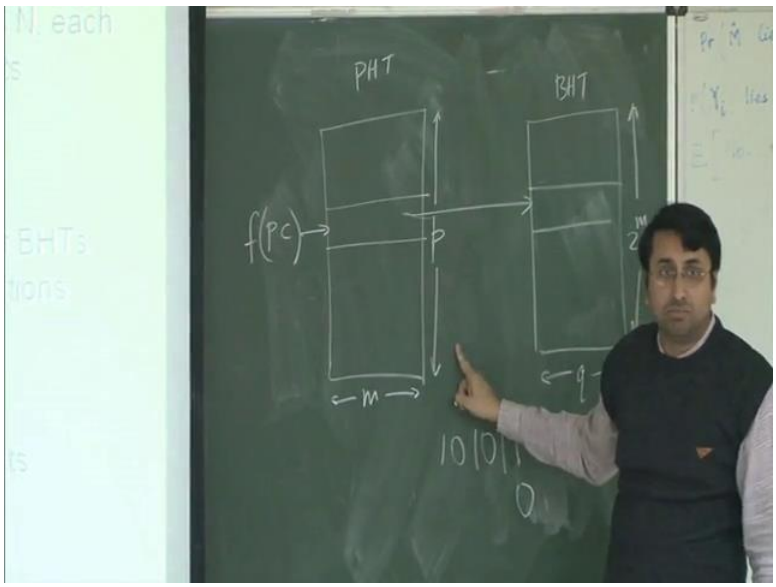
Student: (( ))

Why no B H T will not work then actually.What is the worst case.For each instruction, why (( )) that is the best I can do in fact, here. So, p would be equal to essentially what 2 to the power of size of my pc right shifted byq bits less. So, that is my first level table. So, each branch gets to one entry. So, that is the p stands for them, per branch P H T. So, there cannot be any collision here, guaranteed. What does each every point to, its point to a table, it points to 1 B H T, and the B H T is q bits long, and 2 to the m bits, 2 to the m entries. So, whenever you get this, this one is branch pc. So, whenever you get this branch, you have as many B H Ts here right; small p number of B H Ts here. You look

up that corresponding B H T, and you will index with this particular history, and B H T be the outcome . So, that is the total size of your pap predictor, first level table is mp bits.

And the second level each table is2 to the m bits here p. So, if you assume that you have N is the worst case number ofbranch instructions which will be probably 2 to the power of space of pc. mN plus Nq to the power of m bits.it is a gigantic predictor, pay in large.Is this clear to everybody, pap, any question in the anatomy of pap.So, now, pas essentially tries to collapse some of these B H Ts, what is doing is.It is skipping a P H T unchanged with pattern history table. There is 1 B H T for a set of branches. So, essentially what it do is. So, previously we are taking the pc and index in into a B H T you are figuring out which B H T to access based on the branch pc. Now a bunch of branches will actually use a single B H T. So, essentially what you do is, we applyhash function on your pc to figure out which B H T to access. So, this is what you get.You will get p over s number of B H Ts, and this is your total size of your predictor.

(Refer Slide Time: 24:06).

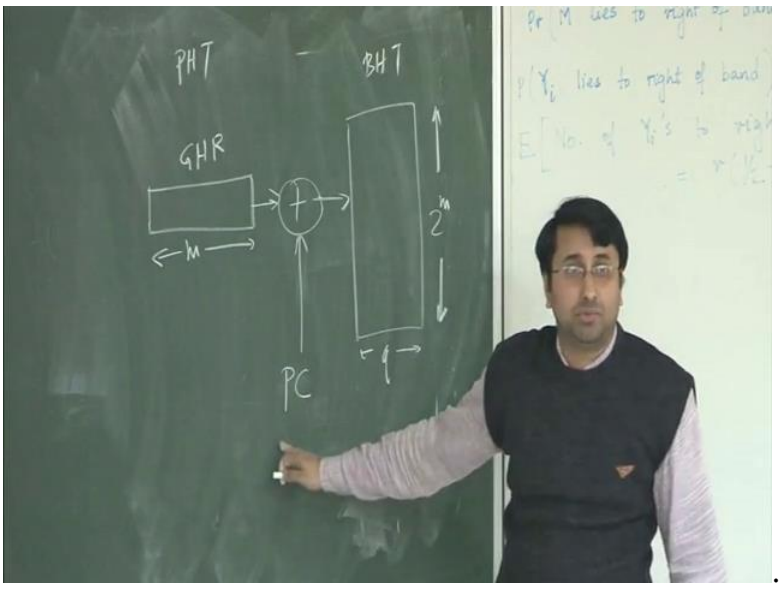


Now pag says that. Well I just give you 1 B H T that is it ok. So, that is the B H T predictor, whatever history you have here you use that index into this table.There is only 1 B H T. So, global B H T, shared by (( )). So, what is (( )). So, here essentially you give complete isolation, a particular branch cannot long from behavior of other branches which maybe good or bad, it is good if you have in the best of your branches which is there if you are correlating branches.They will never for example, correlating

branches.Here you start to have some amount of sharing with the histories. So, essentially what now can happen is, thattwo different histories may actually reinforce the volume of particular counter, that way start happen in now, and that case even puts here, but remember that the negative sign is that, they may be destructive interference, a particular. So, there will be two branches they have different behaviors. So, 1 may be giving the following that 1 0 1 0 1, after that somebody has a 1, and some other branches is 0 after the same history.They will have a big interferencein the same counter we have 1 0 1 0 1, this particular counter will sometimes get incremented sometimes get decremented. So, there is a chance of making mispredictions, that is ok

So, now if you try reduce the size of your (()). So, you can do the same thing.You can say just like the B H T you can say that well I am going to give, instead of giving each branch 1 history entry here, I can give a set of branches 1 is. So, that is your sap predictor, which has the first level table is5 entries per set of branches.Whereas, the second level table is again per branch, that is possible actually. So, essentiallythey still ejects into a 1 B H T, then you can get you get these kind ofsizefor the both the predictor. We can do sas where both P H T and B H T are each entries for a set of branches, and per set of histories, and you can do sag where. So, this is where were the most popular predictors actually, where you have 1 global B H T indexed by the P H T entry contents. So, it looks like this actually, where the only thing is that, p here is not really equal to the number of static branches some number.

(Refer Slide Time: 27:14)



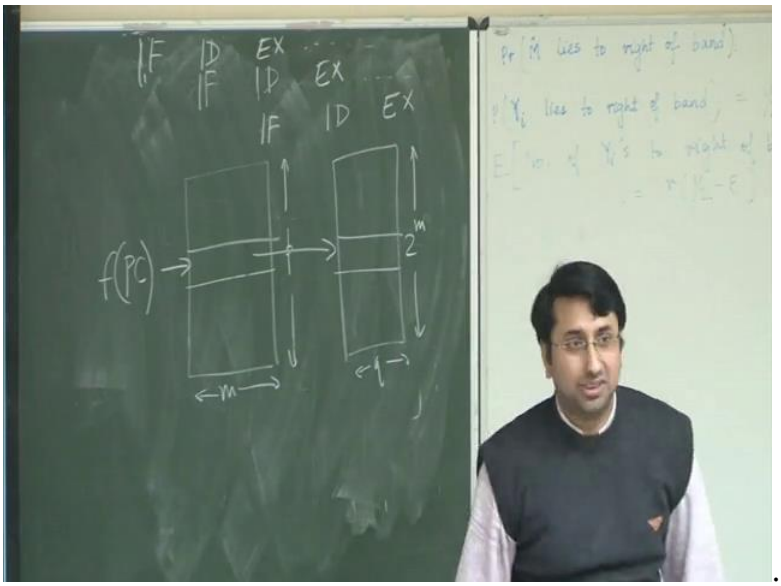
.

So, set of branches will map you 1 entry, and then you can make the P H T global. So, what is the global P H T, just one entry? It is often called the g h r global history register. So, here small p is equal to 1. So, this is my P H T. So, gap would essentially have, stillwhat would gap. So, you have a array of B H Ts, each will be indexed by this alternate. So, these are all 2 to the power of m entries, and how do you pick which P H T to index, based on the branch pc. So, each branch gets a separate P H T. Then you can try collapsing the of the B H Ts you get a gas predictor, and finally, you can have a gag predictor; that is again a very popular predictor where your B H T.

So, notice that when you make the P H T global, you do not use the branch pc any more well, except probably in these twocases, where you select the B H T based of the branches, but here there is no use of branch pc. So, this predictor is very good in discovering correlation between different branches.Oneexample we saw last time, so go back and revisit that example and see actually gag predictor will give you very accurate prediction for that particular background.Any question on this; is it clear to everybody how I update it the P H T B H T, and how I actually make a prediction.

So, given a small piece of program you can calculate the branch misprediction rate for a given predictor, I would assume that. So, there is special predictor called g share, is very similar to gag, except its index function is changed a little bit.It does not directly use the g h r that will use the function ofpc and g h r. So, essentially what you have to do is, you prepare a mp hash of the pcand show that with the g h r. I mean I am showing sorry you can do some other function, but the way g share was predict, g share was proposed for. It was like this which B H T is indexed with branch pc xor g h r. So, here you essentially shift out the last 2 bits, and take the next m bits xor and the g h r.

(Refer Slide Time: 30:43)



So how do you update the direction predictor. So, remember that your branch case executed in the ex stage of the pipe, before that you do not know the correct outcome of the branch. So, really cannot update the predictor, before you reach the ex stage, we execute the branch and now we know what the outcome of the branch is. So, you can go and update. So, first the B H T is updated, by indexing it with the old P H T entry. So, let us take an example.SupposeI have a fact predictor.

So, there is a branchwith some pc which has used applying the hash function, and it gave me this particular entry. So, I get an n bit history out of it,and this is p bits I use that index into my B H T 2 to the power of m entries which is and based on this counter content, I gave you a prediction whenever in the decode stage when you looked up the predictor. So, finally, I execute the branch. Next still that figure out if the prediction was correct or wrong. So, now, it is time to update the predictor, because that is what the predictor is going to run the correct thing right.You have to keep on updating correct prediction, the correct outcome. So, first I am saying that you update the P H T by indexing it with the old P H T entry by old P H T entry. I mean what about P H T entry you got when you index into thepredictor at the time of decoding, why is that, why cannot I just go now look up the P H T and whatever entry I get here I use that index is the P H T, and increment it if the branch is taken decrement it.If the branch is not taken why cannot I do that, why do I have to remember my P H T, how can it change. You cannot change the predictor before the ex-stage, that is not possible, what you will.

Student: no you have instruction which have come.

It mayit predicts does not change anything. It does not change anything, P H T entry will be different, as per different branch. So, we have a branch instruction which waits through the pipeline. So, I predict to the branch here by looking it up.Now the branch which is ex stage executes. I know the correct outcome, I should be updating the predictor now. So, what I am asking is that when I predicted the branch here, I index to the pitch P H T. I got some history at that point of the branch, I will say that I have to use that index in the P H T. Now I cannot just go and look up the P H T, and get the history and get the B H T, why is that. You are saying something can you repeat that.

Student: (( )) that fetched.

Butit does not update anything.No it in the p h. It does not modify anything. Index this some other entry give or a prediction, does not modify is not it here the tables.Is the problem to problem to hear anybody or it does not make any sense at all.Does it make any sense.Why am I talking about this problem, or it is not a problem.You thing the same instruction is getting fetched, same pc. But remember that the branch predictor is not modified until this stage. I cannot modify the predictor, because I do not have anything modify to it. I can only lead the content that does not make any difference. Previous instruction; as look at the previous. So, if this instruction indexed into the P H T and the same history, it should have actually modified the history before it gets the chance to access actually, it gets a chance to modify. So, essentially what happening now is that in this particular cycle, this branch is making a prediction, while this branch is actually updatingthis. So, depending on what happens first, you may or may not get the correct outcome. So, essentially the way it is going to work is that, this modification is going to happen actually here in this cycle, after the branch it has completed execution.

So, you cannot really rely on the correct content of p h t, because that you will need to a wrong history, if you update the B H T with that. You have to take the P H T contents when you made the prediction, because next time what I am trying to do actually.Next time when I encounter the same history, I should be able to give a correct predictionsactually right. So, which is why I have to copy this P H T along with the branch instruction. Let me go to the pipeline, and that has to index to the B H T and update the predictor right.

Sir, yes as if we are running the same branchinstruction multiple.

No its not,it is a hash function then it be collisions,do not forget that. They will multiple branches map to the same history actually that is possible.

Student: but again they need to be 2 to the power m bits or 2 to the power m p distinct, because you have.

No why this is just the entries, and p is not equal to number of branches. So, the hash function is I am taking the lower lock p bits of the pc, after removing the last 2 bits. Yes control lock p bits have to be common, yes that is right,that is possible yes. So, is this clear to everybody, this problem, why let to carry forward them P H T.

So, you first modify the B H T, and then you go and update the P H T. So, how do you modify the B H T, if the branch is taken, you increment this counter, if it is not taken you decrement the counter, how you modify the P H T. You shift this out by 1 bit, and shift in the new outcome, 1 for taken 0 for not taken.The question is, why do I do it in this order actually, or does the matter at all.can I update the P H T first and then the B H T, does it matter yeah. So, is it ok, can I update in arbitrary order.No, why not.What is the problem.

It will affect the MIPS.

You can assume that this is roughly with respect to other branches. So, why cannot it flip this, the order what can I. it should be fine exactly. So, since I anyway using the old P H T entry, does not it matter, which order I access then. In fact, I updateparallel you say it up.

Student: So, the old P H T is only used for indexing the B H T; yes. And by updating the P H T would not use the old P H T store to.

No of course not. I will index into the table I am, whatever I get you shifted out and the new entry. So this order is what important. In fact, I will update them in parallel say time. The next question is that arises is, should the predictor be updated speculatively with the predicted outcome; that is when I make a prediction here, why update the predictor at the same time immediately, is there any gain of doing that, because it seem sowrong. I do not know what my predictor outcomewhether it is correct or wrong. Now I

am asking that can I update the predictor with the wrong outcome. So, there are pros and cons, can somebody think about it.Now what I am saying is that instead ofthe updating ex stage can I update it here, because the predictor is going to give me a prediction, that the predictor thinks and the branch is taken.CanI take that outcome when actually increment is further, and change this P H T.

So, that isone big problem, that if the predictor is wrong then we have corrupted the state of the predictor (( )) correct any more. So, that isone problem is there any advantage of doing that; anything else. Next instruction will be, if I do not do it is there a problem. So, if I update it correctly then you are saying that next instruction may benefit why is that, because of the prediction otherwise entire section is during that prediction of the previous one. So, you are saying that if the next instruction in the branch, map to the same entry. This may actually benefit by seeing the new prediction, there is a correlation between this.

So, can you think of any example of this. So, to help you, suppose we have a very long pipeline, the time from make a prediction to the time you take to update the predictor is very long many cycles in it. So, can you elaborate little bit more, is the same branch I am executing over and over, but I have several instructions in between there is a fall to body.So, let start first branchnothing here, sothat all zeros, I have initialized everything 0, will index be the first entry will give me some prediction likely to be all .

So, then the branch goes on through the pipeline, before you gets to ex stage I fetch the next that is possible actually, because I am actually fetching through the loop, I may actually end of fetching many iterations, even before the first loop branch executes;that is possible depending on how many pipelines I have here right, the next one branch shown up, you still see the all 0giving the same wrong prediction. So, is it clear now that actually would have helped, to update the predictor provided I had a correct outcome. So, that is the very common example of. So, I gave a very simple example here you cannot copy that branch were essentially, you have two branches which are correlated to each other. So, you want to update the predictor with the first branches outcome even before the second branch actually predicts. So, that second branch may have higher chance to predict correctly, but of course, this is all true if I update it correctly.

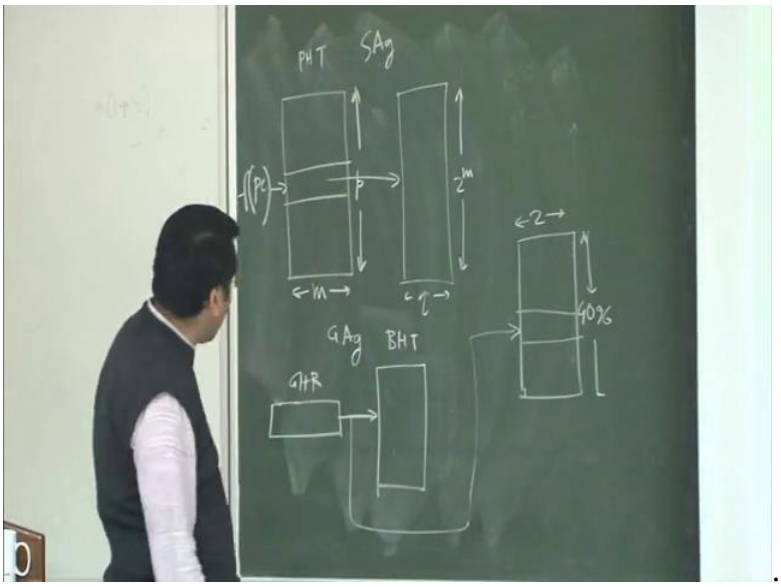
What happens if I do a wrong update, how do I fix the predictor. I have to fix the predictor now. So, finally, I go to the ex-stage and figure out that I had misprediction. Now I have two things to do, one is of course, I have to remove some instructions, that I have fetched wrongly, second is that I go and repair the predictor, now how do I do it, but on the wrong path I might have updated many P H T entries. So, I will list out all of one's one by one, flush everything, how do to that yeah. Check point, exactly keep the copy of this P H T yet that is copy the P H T or every branch, whenever I have a misprediction I will copy the, you know the contents of the check point from then remaining, it is not done in any processor, because of the big over it. So, essentially you are check pointing mp bits P H T, is usually large in size, and for every branch you have a check point, because any branch can go wrong whichever branch goes wrong, you have to restore the check point of that branch, but what is done is, if you have a gag predictor you have just one P H T entry, that is easy to check. So, a gag predictor you will often find the g h r is update is speculatively, because it helps a lot.You will copy the g h r that just m bits that easy to copy.

What about the B H T. So, there are two questions to ask now. He has speculatively updated predictor do I at all update the B H T, does not make sense advantage, and if yes; how do I fix it, if I update it in wrong way. So, first you might want to think a easier one, suppose I do not update the B H T what do I lose, do I lose anything, then assure that you are in steady state, do not ever think about the corner cases that you know your con starting or entering.In the steady state is there any advantage of updating the B H T. I have a taken prediction, I go and increment the B H T by 1. If I am correct do I gain anything, I do not, my predictions still remains taken,Iincrement it.IfI did not increment the Iwould have been taken anyway. So, see does not really buy me any extra accuracy by incrementing or decrementing the B H T entry as such. If I have a not taken prediction if I even if I do not decrement the B H T entry, I will still get a not taken prediction next time, because I was not taking a prediction.

So, B H T is never updated, you do not do that, because it does not buy you any extra accuracy it actually gives you a big headache that how you face the register. So, whenever there is a speculative update of a predictor, you only update the P H T, provided you have the single entry, or maybe small number of entries, otherwise you never do that actually. Any question on updation of the predictor. Is it clear, see. So I

wanted to talk about one hybrid predictor ,where essentially we have two components and that is a selective, which you select one of these. So, today you find this predictors in pretty much every processor, and that goal of having hybrid is that, starting predictors try to address something type of branches, and if you want tocoverage, you probably the combining multiple different types of predictors. So, a very common example is combining sag predictor with a gag predictor. So, just to remind you what they are.

(Refer Slide Time: 47:00)



So, this is a sag predictor.We use the pc or hash function of pc to, and the gag predictor has the semi P H T entry which is the global history register, then it get us into a B H T, that is a gag predictor. So, as you can see here sag predictor, you only learned that correlation between different branches by accident, if they actually map to the same entry. And that learning may it with the destructive or could be constructive; you really do not have any control over that. So, these 1s are often called local predictors, because they are essentially designed to capture the pattern of a single branch.Well here you are mixing all the history of all the branches. So, these are actually go and capture global correlations. So, these are often called global predictors.

So, if make sense combining these two and have a single predictor.So, that you can predict the globally correlated branches correctly, and as well as locally correlated branches correctly. So, it combines the local and global predictor. So, here you have the parameters listed. So, this is exactly what was implemented in the alpha 2 1 2 6 4

processor. Say essentially has thousand 24 entries; that is your small pwith 10 bits each, that is small m in P H T. And the second level has thousand 24 entry B H T it has to, because that is 2 to the power of 10 each being 3 bits; that is q . So, these are 3 bits saturate counters, and these arebits. The gag has the 12 bit global history register and a 4B H T which is to be power aftereach being 2 bits. So, whenever you have a branch, what you will do.You set the pc the local predictor, it will tell you something 0 or not.You also activate this particular predictor. We get the B H T the g h r and tell you some out, you have to select one of these.

So, there is a meta predictor or selector, that is the bimodal predictor, with 4096 entries indexed by g h r. So, there is a bimodal predictor. So, here these are twocounters 4096 entries indexed by the g h r, and its job is basically to tell you which one is correct, or which one is likely to be correct. So, it that is why it is called a meta predictor.Itspredicting the predictor outcome right. So, now, need to carry 2 P H T entries along the pipeline, because you have to update two things.You have to copy the g h r, you copy the P H T entries here , when you when you turns to update ,it will need their they own copies. How do you update this component.

So, finally, the branch executes.You know the correct outcome, what is update protocol, what make sense. would you update both the predictors all the time. So, there are 4 situations, the sag is correct, the gag is wrong. So, this my sag, this my gag; sag is correct gag is wrong, gag is correct sag is wrong, both are correct both are wrong.Can you tell me what will you do in each of the cases, also you have to clear out how to update. So, essentially how we update the chooser, the selector essentially.

So, there are three things you have to update, how you will update them. So, let us say the first stage; sag is correct gag is wrong, tell me. You do not update the sag at all, what you reinforce its learning. So, if you take the sag in isolation, if we had only that predictor what you update whether it is correct you should. You always update the predictor. So, you always update the sag component and the gag component with the correct outcome always, what about the selector. Both are the output is same, like both are correct we would not update thebimodal predictor, and when both are wrong even then we want to update the bimodal predictor. Why is that? Then it makesno incrementing both or decrementing both.

So, when the sag is correct gag is wrong what you do. We increment the counter of set. When the counter is the g h r.G h r is used to index in the bimodal predictor.When the sag is correct we decrement the counter saying that increment the counter saying that. So, it will be the opposite thing, whatever weight is yeah exactly. So, while sag is correct gag is wrong, I will go and increment the counter, and when the gag is correct and sag is wrong, I will decrement. So, what will be the select protocol now. So, whenever I get the counter value, I compared it against the midpoint and if it is over the midpoint I will choose the sag, otherwise I will choose the gag, and when both are correct we do not update the selector, because the selector you know has really nothing to do with this case, both are correct, and when both are wrong, then also nothing to do with the case.