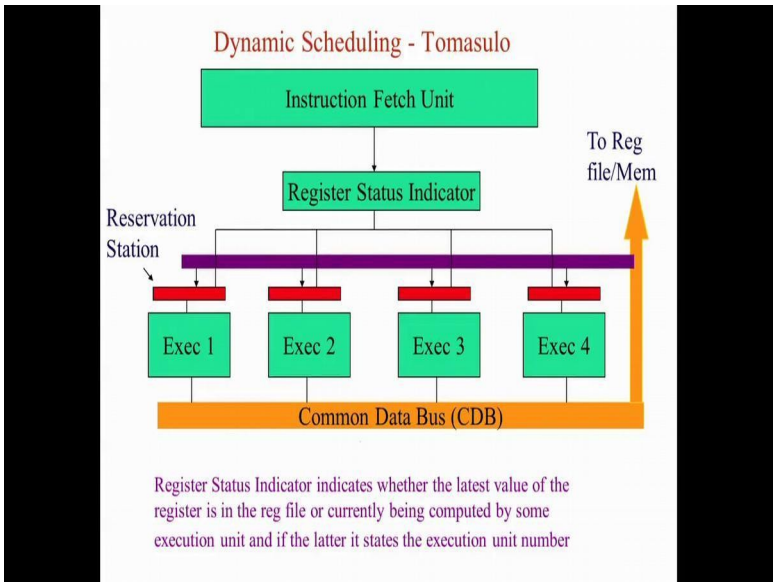
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 20
Dynamic Instruction Scheduling (Part - I)

So we stopped at this point in the last class.

(Refer Slide Time: 00:20)



We explained what the instruction fetch unit does it can bring more than one instruction at a time. The registers status indicator, there is an entry for every register and if that entry is 0 that means the latest value is there in the register file otherwise it is in that num it is the execution unit number which is currently executing. And then there are reservation stations marked red. So, if a instruction has to wait for some answer it will wait there and it will know which unit is having

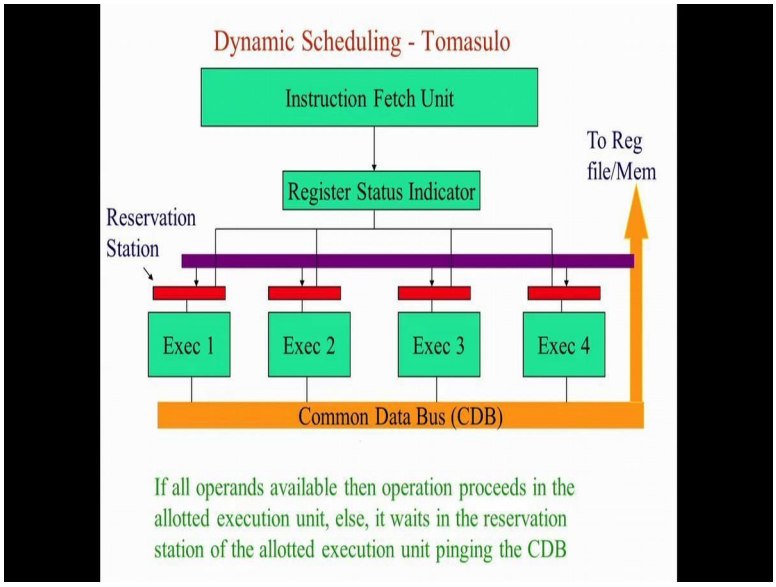
that and then every execution unit when it finishes it puts the result in that common data base and that along with the unit number so that the result along with the unit number will go to the register or memory. And on that way it will also through that you know purple path and every reservation unit can see that, right.

So, if the reservation unit is waiting for somebody then it will get it here itself before going to the register file. And this is precisely what we told as the operand forwarding. Now, we will now go and explain through an example how this hardware is going to take care of the raw hazard: I almost explained you operand forwarding we will see with an example. The WAR hazard and the WAW hazard: it will also do register renaming it will also do some renaming and it will handle these WAR and WAW hazard right and all these things we will see using an example.

The instructions are fetched one by one and decoded: one by one in the sense that in one cycle there will be many instructions that will be decoded and you will find the type of the operand and if there is an execution unit which can do that it will be pushed to that execution unit. So, there are multiple execution unit it is not necessary that every execution unit will do every operation. So, there can be 2 units specifically dedicated for load and store, there can be 2 units specifically dedicated for integer and logic arithmetic, there can be an unit dedicated for floating point for multimedia etcetera.

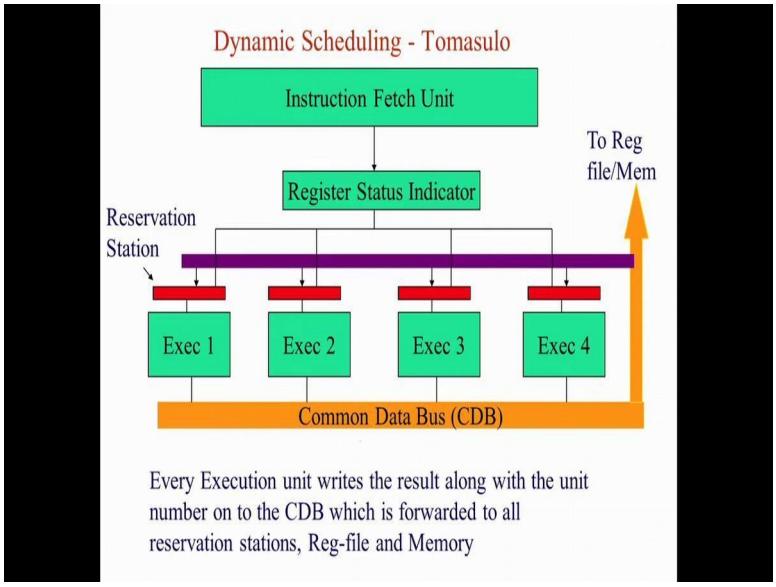
So, I will fetch one by one in the sense I will fetch all the instructions right, but I have to in one cycle I will process it in a such a order that I will go and allocate if there is an I will decode the instruction and if there is an execution unit which can handle it I will put it. So, in one cycle this instruction fetch unit can process more than one instruction please also understand that I have to process the instruction one by one because then only I will understand the data dependency you understand right then only I can understand the data dependency. So, let us go the register status indicator again indicates whether the latest value of the register is in the register file or it is currently being computed by some execution unit.

(Refer Slide Time: 03:55)



And if the later is true that is it is computed by some execution unit then the register status indicator will store the number of that execution unit. So, that somebody wants to use that we will know which execution unit is currently processing it. If all the operands are available then the operation proceeds in the allocation execution unit allocated execution unit else it will wait in the reservation station of the allotted execution unit and it will be pinging the common data bus.

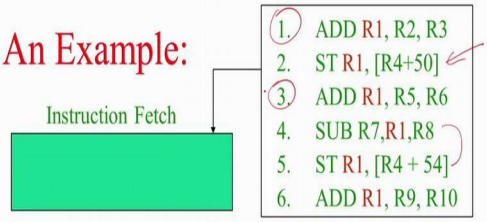
(Refer Slide Time: 04:32)



Every execution unit, once it is finishes it will write the result along with unit number because if somebody is waiting for it they will know this unit is waiting on to the CDB which is forwarded to all the reservation stations through this purple thing and it to the register file and memory I think we have done it 3-4 times. Now you will know almost by heart.

(Refer Slide Time: 04:40)

An Example:



1. ADD R1, R2, R3
2. ST R1, [R4+50]
3. ADD R1, R5, R6
4. SUB R7,R1,R8
5. ST R1, [R4 + 54]
6. ADD R1, R9, R10

Register Status Indicator

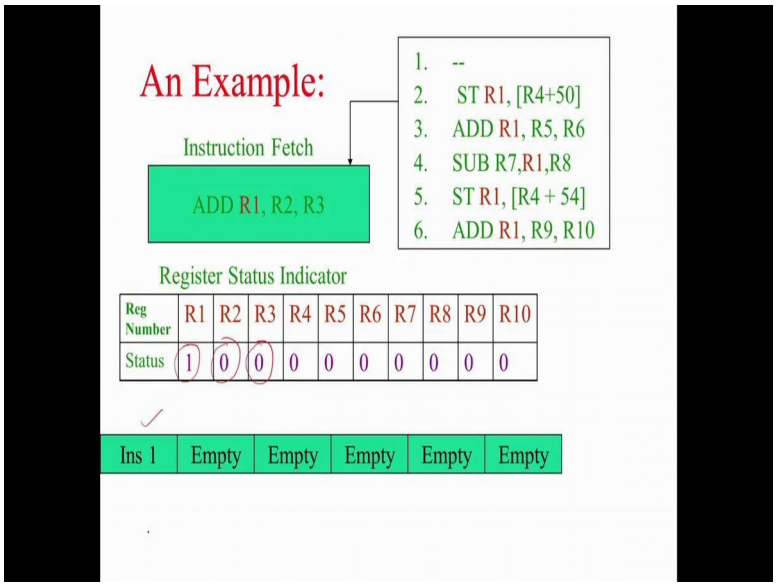
Reg Number	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
Status	0	0	0	0	0	0	0	0	0	0

Empty	Empty	Empty	Empty	Empty	Empty

Now, let us see we will take. So, these are the 6 instructions that needs to be executed add R 1 R 2 R 3 store R 1 R 4 plus R 50 store the result of R 1 into R 4 plus 50 add R 1 R 5 R 6 subtract R 7 R 1 R 8 store the result of R 2 to R 4 plus 50 4 add R 1 R 9 R 10. So, I have just restricted no there are many things let us see how it is going to execute. Now these instructions are processed in one cycle 6 instructions are processed in one cycle. Let us assume for the sake of argument that these are all the reservation stations currently all this there are 6 units let us assume that there are 6 units and all the units are currently empty, right.

Now we are going to start executing and the register status indicator are all initialized to 0; that means, the latest value of all the registers right are available in the register file. So, this is one snapshot of the code we will just take it and see how it is going to be executed. Now what I am going to do is I will give 6 slides; in the 6 slides I will tell you what is happening to each one of these instructions, but one in practice what will happen is all the 6 instructions will come at a time and all the operations that I am going to talk of in the next 6 slides will happen in one cycle I need to get more than one instruction to execute per cycle.

(Refer Slide Time: 06:22)

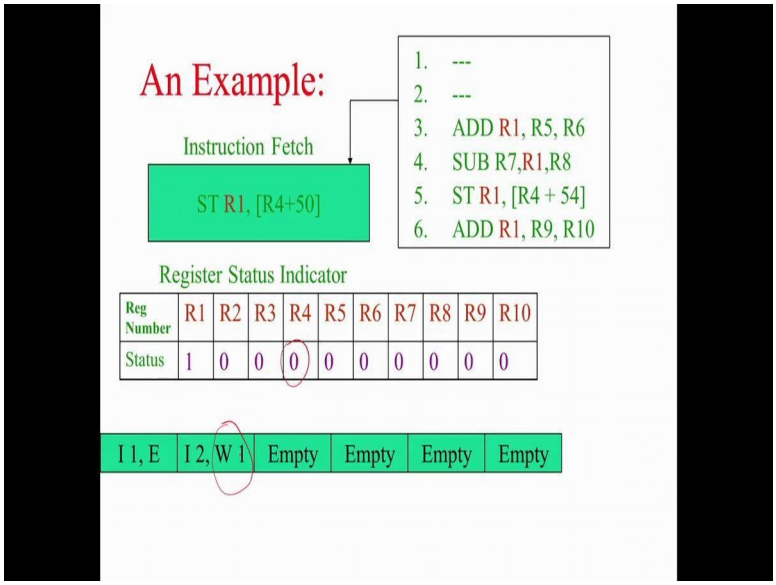


Are you able to follow what I am trying to say write now what will happen first add R 1

R 2 R 3, now R 2 and R 3 what is the value of R 2 R 3 they are 0s R 2 and R 3 are 0s; that means, they are available. So, immediately what will happen it will be allocated to unit number one look here right now where is the latest value of R 1 it is not in R 1 it is currently being computed by unit number one and that is why you get this one here. So, the moment add R 1 R 2 R 3 comes R 2 and R 3 are checked they are all 0s; that means, the latest value is available in R 2 R and R 3.

Now what you need to do I am computing the value of R 1 here I am going to add R 2 R 3 and the answer is going to be one. So, there is an addition unit. So, I will now give it to that addition unit to go and fetch the data and do that, but I should tell the instructions that are following [FL] R 1s value is not in R 1 it is currently being computed in instruction unit execution unit number one are you able to follow and that is why I have put for R 1 the value one, now right. Now next all of you followed this very clear no doubts now in the same cycle please note it is not going to be another cycle in the same cycle now store R 1 R 4 plus R 50; this one.

(Refer Slide Time: 08:06)



Right now what is right? Now what is this store going to do it is going to read from R 1

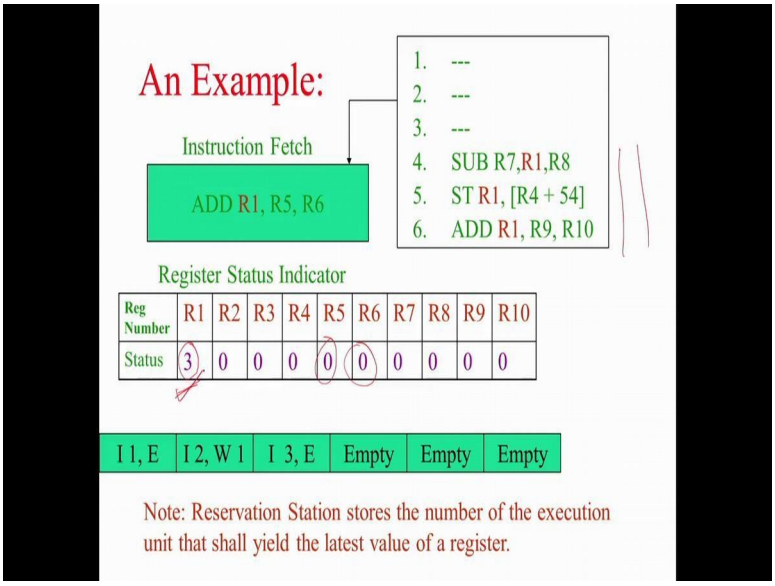
please it is also going to read from R 4 if R 4 is being computed be some I should you know. So, it is going to read from R 1 it is also going to read from R 4 please note

because I need to compute the address right and then it is going to store in R 4 plus 50 right. Now R 4 is available to it right R 4 is available to it fine, but R 1 it comes and sees the registers status indicator has 1. That means, the latest value of R 1 is currently being computer or is going to be computed in execution unit 1. So, it will go and say I will wait for one. So, it is waiting for one.

So, the I 2 is allocated to execution unit two, but it is not proceeding it is waiting for the result of one you all understand how I got this w one obvious right. So, I come here I need to read R 1 and also R 4 this is very very important please not that I have read R 1 and also R 4. Now when I am trying to read R 4 R 4 is already available, because it is status indicator is 0. So, R plus 50 I know the address right, now I have to write into that address what should I write R 1 I have to write now where is R 1 is not in the R 1 it is currently being computed be execution unit 1. So, I am waiting for it fine followed any doubt.

Now, what is going to happen I am going to get add or 1 R 5 R 6. Please note R 5 and R 6 they are available, right.

(Refer Slide Time: 10:01)



So, we can start execution, but now I make the register indicator 3 I allocate I add 2

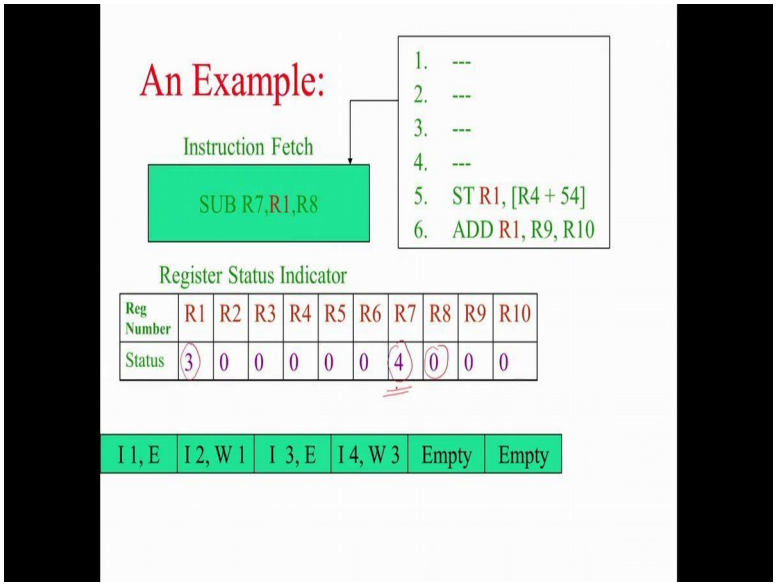
instruction execution unit 3 and I say the latest value of register one is now in 3 subsequent instructions after this fellow if they want to read R 1 where should I wait for I should wait for 3 please note that this I 2 is already waiting for one this I 3 whatever is going to follow I 3 will now stat waiting for 3 I do not need one anymore here because, from now on if somebody wants to read R 1 he has to look at execution unit 3, because the latest value of R 1 for these instructions R to be executed R available with execution unit 3 is being computed execution unit 3 are able to follow, right.

So, that is why I make the register indicator 3 is it fine. So, let me just do it once more. So, I start with this is the program that I want to execute. Please note, please give undivided attention right now I i get add R 1 R 2 R 3 R 2 and R 3 are already available in the register file because R 2 and R 3 are 0s right. So, now, i; so, this instruction all the operands necessary for this instruction are available. So, I can start executing the instruction.

I start executing the instruction in execution unit 1, but in the register indicator status I say that the latest value of R 1 is not in R 1, but it is in execution unit 1 next step I get store R 1 R 4 plus R 50. So, I have to read 2 values R 1 and R 4 R 4 is already available correct you know why I have to read R 4 right I have to compute the address. So, R 4 is already available R 1 is not there where is R 1 it is in execute it is currently computed in execution unit 1 who is telling this the register status indicator is telling.

So, I allocate this instruction to execution unit 2 and make it wait for one right it is waiting for one and it is allocated for 2. Now the next instructions add R 1 R 5 R 6 R 5 and R 6 are available. So, and there is an execution unit also available. So, your third instruction set can start executing in the execution unit 3. Now for subsequent instructions which are going to use R 1; R 1 is no more in 1 R 1 is now in 3. So, I go and erase one and put 3 here are you able to follow the very [FL] thing right it is not even I by millionth of your pulley problem in physics.

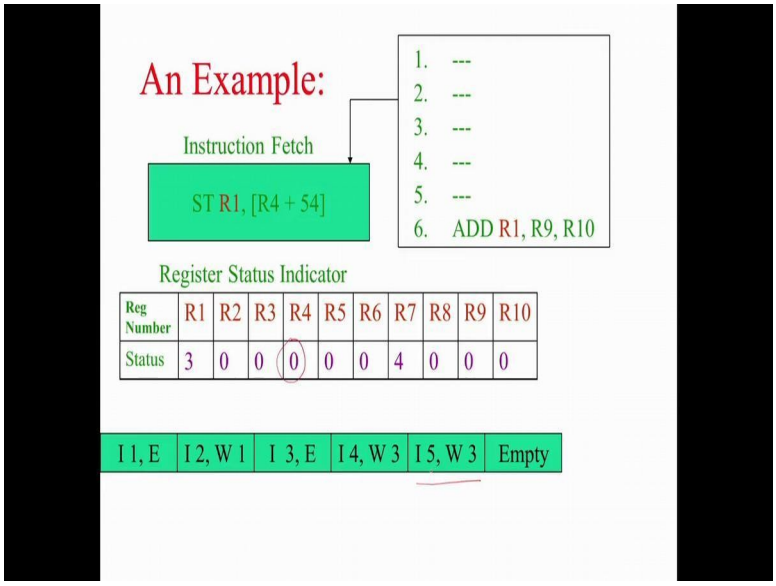
(Refer Slide Time: 13:25)



Now sub R 7 sub R 7 R 1 R 8 now R 8 is available R 1 is not available it is still in 3. So, the fourth instruction will be allocated to execution unit 4 and that will be waiting for the third fellow to complete right now and the latest value of R 7 where it is going to there it is going to be in 4. So, your register status indicator is now updated to 4. Now this fourth instruction cannot proceed to execute, because it is waiting for R 1 and that R 1 is currently computed by execution unit 3 correct and the result of that is going to be written in R 7, so in the register status indicator for R 7.

Now stores 4 because somebody else who wants R 7 beyond then they have to wait for 4 to complete you are is it fine. Now the next is it any doubt (Refer Time: 14:34) fine can I proceed, yes.

(Refer Slide Time: 14:39)

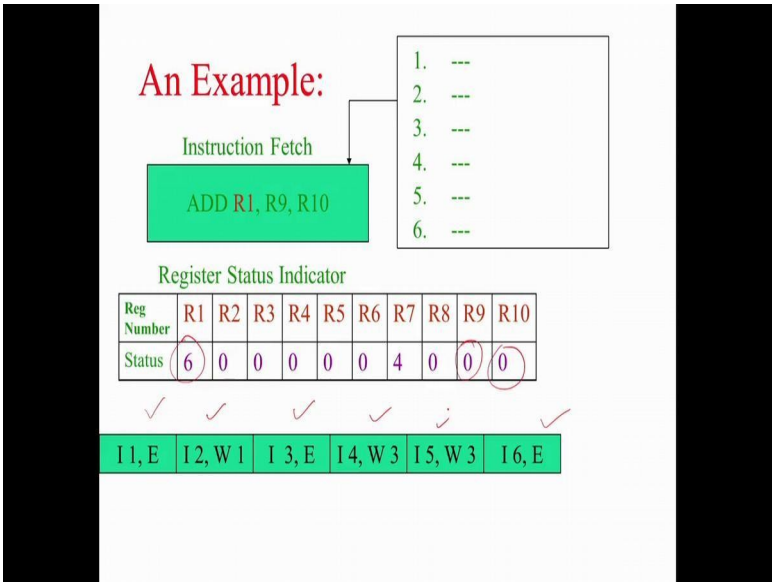


Student: Yes sir.

Now next one, now store R 1 R 4 plus 54, right: now I have to read R 4 and I have to read R 1 R 4 is already available because the register status indicator is register status indicator is 0 R 4 is already available, but R 1 is currently being computed by whom execution unit.

R 1 is R 1 is currently being computed by execution unit 3. So, I will go I 5 will also now go and wait for execution unit are you following R 1 is currently being computed by execution unit 3.

(Refer Slide Time: 15:29)



So, I 5 should also wait for execution unit 3 right now next is add R 1 R 9 R 10 right R 1 R 9 R 10 note that R 9 and R 10 are available correct. So, this get this is start executing in it is execution unit 6 and the latest value of R 1 beyond that somebody wants we will now be available in execution unit number 6 you all get this now what is happening all these operations should happen in one cycle, then only you will get that super scalar architecture right. Now how do we construct such a fetch unit that I will keep it for computer architecture course? We will not deal it in computer organisation course that is really advanced topic right, we how to about doing it, but this is going to happen and when it happens now what will what are the things.

Please note that I 1 I 3 and I 6 are executing while I 2 and I 4 and I 5 are waiting. Please note that I 6 executes before I 2 I 4 and I 5 I 3 executes before I 2 that is why we call it as out of order execution out of whose order compilers order compiler has given some order, but we have now violated that order and we are going out of order execution. Now all the fellows who have the true dependency or what we call as true dependency is equivalent to raw right read after write all those 3 follows are waiting they have to true dependencies these 3 fellows are waiting right let us go back.

These 3 fellows are waiting the one sorry this fellow sorry this fellow is waiting for one

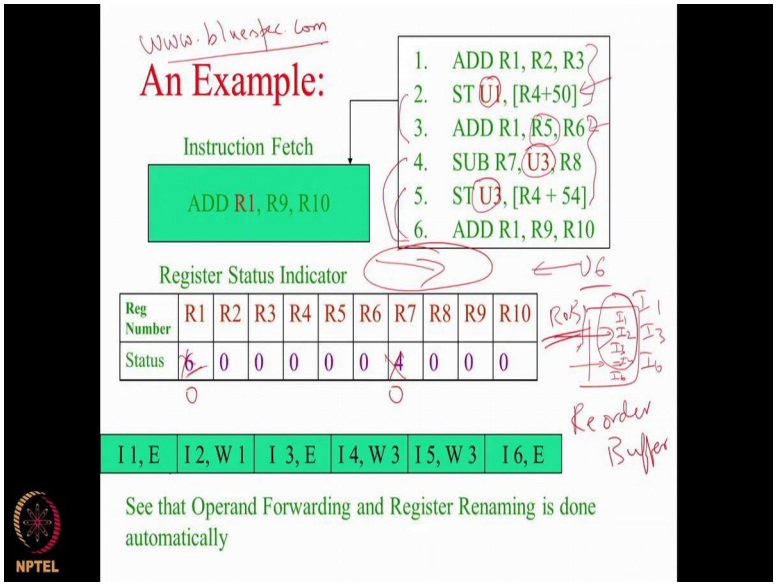
to complete and these 2 fellows are waiting for 3 to complete correct. So, the true dependencies are handled there is. So, this hardware what did I tell this hardware has to handle read after write and write after read and write after write read after write it is handled here if I am dependent on a previous instruction and that instruction is currently being computed I will wait. Now what will happen when finishes it will write the answer on to the common data bus in to the purple bus we saw and 2 is waiting for one answer one the moment it sees one it will go and grabs that and it will start executing.

Similarly when 3 finishes, it will write the answer on the common data bus the moment 4 and 5 they are waiting for 3 they will be pinging the common data bus when it comes immediately it will start executing. So, when the answer is going to the register or memory file it is also forwarded to this and they will catch and start handling. So, read after write hazard is completely handled here and this is how operation forwarding works any doubt in this.

Student: (Refer Time: 18:39).

No doubt, right, yes or no can I proceed any doubts no; now how is; now I am having out of order execution how is the WAR and the WAW hazard being handled how is the write after read and write after write hazard being handled that is quite straight forward. Now what has essentially happened I 1 is executing I 2s R 1 is waiting for unit 1.

(Refer Slide Time: 19:08)



I 2s R 1 is actually waiting for unit 1 I 3 executing I 3's I 4 and 5's R 1 are basically waiting for unit 3; that means, what I have renamed there R 1's with U 3 and this R 1's with U 1 are you getting this. Similarly now, subsequently if there is an R 1 here it will be waiting for unit 6. So, what is this? R 1 got renamed by U 1 these R 1 got renamed by U 3 correct right register renaming happens by default correct then what happens once the instruction finishes it will write in the common data bus and it will also go and see if there is any register waiting for it for example, when the sixth instruction finishes right finishes it will go and if this value is 6 all the value that is 6 it will go and make it 0.

That means 6 instruction is completed the latest value of R 1 is in register now it is in the register file it will go and make that 0. So, when a instruction finishes and if there is any register status which is indicating that fellow right when and unit finishes if that fellow is there on the register status indicator then it will be it will make 0. For example, if 4 finishes then it will go and make this 0, unless somebody comes if one finishes it will not make this 0, because by the time one finishes this already became 6. So, whenever I am writing something into the register file unit K is writing into register file it will go and check all these register status indicator if any of those fellows have the value K, it will go and make it 0, fine.

So, what has happened here your operand forwarding happened you are waiting for a result when there is a true data dependency that happened and whenever

there was a named dependency like WAR; WAW for example.

There is a WAR between 2 and 3 2 is reading while 3 is writing the WAR hazard between 2 and 3 has been resolved by renaming 2 right the WAW hazard between sorry the WAW hazard the WAR hazard between you know 6 and 5 and 6 and 4 are got out by U 3 renaming this.

So, there will be and right are you are you getting this. So, there will be no, you know all the write after read hazard just disappears and this renaming happens in (Refer Time: 22:21) are you are you with this. Now what happens is- now when I am writing to the register file when I am writing back in to the register file please note that this one 3 and 6 are writing to R 1 1 3 and 6 are writing to R 1. Now 3 may finish before 1 or whatever there could be some other one here which is suppose we are waiting for R 5 somewhere essentially 3 may finish before one and 6 may finish because here you know all the fellows who are writing on to R 1 are executing simultaneously, but it may be the case where one fellow needs to wait also.

So, then what could happen is. So, what we have after this is something called a re order buffer reorder buffer this will be in the path this will be in this sorry this is called rob reorder buffer that will be here; so the instructions as they finish right. The instructions as they finish will be loaded in to this. For example, in our case I 1 will finish I 1 then I 3 then I 6 I 1 I 3 I 6 will finish in one slot. So, I 1 will be here then there will be one gap then I 3 then 2 gaps and I 6 will be here.

Then your I 2; I 2 will finish I 2 I 4 and I 5 also can finish I 2 is a store may be I 4 can finish and so on right. So, the moment I finish I go and commit all the instructions above it in one shot. So, that that is the reorder buffer, because what will happen is that at the end at this point see after 6 when I am viewing them result of R 1 it should be the answer got by 6 I am taking care of all the execution within the micro architecture, but when I am writing to the register file I should reduce I should have the latest value there right.

So, what I do is even though I complete instruction earlier I keep it in a reorder buffer

and assignment and instruction finishes all the instruction all the set of consecutive

instructions from the top I go and commit it. So, I 1 finishes I 3 finishes I 6 finishes the movement I 2 finishes I 2 I 3 will be committed the movement I 4 and I 5 finishes I 4 I 5 I 6 will be committed that will do in one shot, right. So, by using this reorder buffer I will also get a consistency in my register file. So, that when I am looking at this point when I am looking at this point here I will get a latest value of R 1 when I want to read it right, yes.

Student: What is the necessity of (Refer Time: 25:51) committing the instructions as soon as you find them find that the top few of them are consecutive.

The top that is no more necessary right, I want to have reorder buffer reorder buffer is also finite length. So, I do not want the reorder buffer I cannot have infinite length of reorder buffer. So, when I find the some instruction can be committed, because all the instructions above it has are completed I would love to go and flush it out, because reorder buffer is of a finite set. So, what should be the size of the reorder buffer 6 at any point of time 6 instructions will be executed? So, I will have it as 10 or 12 so, that some instructions can wait for long, so, the other instructions can proceed. So, I will keep on populating that reorder buffer once it finishes I have to go and flesh it. So, the construction of the fetch unit and the construction of the reorder buffer are topic of advance computer architecture.

So, if we take the elective on computer architecture you are expected to be taught on this you know this is 2, 2, 2, 2 large for this we do not have time for this, but you can go and read how this instruction fetch unit can fetch 6 instruction at a time resolve it. So, again logic that is basically logic you can start designing you can use your corn of map and also design that and then how do we go and commit how can see the common data bus in a single cycle has to commit many instructions and how the reorder buffer works. So, these things are you; you can design it, it is all the concept is this, but you could have you know different ways of handling that right question.

Student: What is meaning of reorder method committing something?

Reorder buffer committing means that it is there in the reorder buffer it is waiting for

some instruction about to finish once it finishes I flush it out and put that value into R 1. So, that is what we mean by committing it latest value of R 1 will not be in the R 1. It will be in the reorder buffer once this is over then I will go and commit it R 1. So, some interesting points come up the thing is that when the register status indicator is 0 right, when the register status indicator is 0. That means, the latest value of R 1 as for as; that means, the latest value of R 1 as for as we have considered. So, far we said it is the register files it need not be really in the register file it can also be in the reorder buffer.

So, I have to go and check the reorder buffer hey R 1 is there no then go back to that. So, if you if you guys interested then what you can do is that you need to learn a language called blue spec I am writing above here w w w dot blue spec dot com you will lean a language called blue spec and then look at our you know Shakthi program where we have done a super scalar thing. So, you can actually go through the code and see how we construct reorder buffers how do we construct instruction fetches damn tough thing right, but then the language which is. So, expressive that you can understand it quickly right, but you have to learn language blue spec.

If you have to do it in a low level language like Verilog or the or the hardware description language you did see please note that this is to be this is not a C program correct instruction fetch the logic I am talking all these are going to be realised on hardware right. So, I need to write the logic for the instruction fetch I need to write the logic for the reorder buffer. So, that is not going to be a joke verifying this see actually again I repeat right this is not software what happens in software I have already told you right you make a bug. Now you make the user find out what the bug is you correct all the bug and resell to him as a next version of software within hardware we cannot do that. So, how do we even verify that this is working correctly the already we have seen atomic instructions correct.

So, where we have seen race conditions this and this trying to go in parallel now in hardware this is going to happen now I am going to get multiple instruction I have to see how those race conditions should be avoid synchronisation to happen lot of things come up there and that is really advanced. So, if you right; so you can go and look at our out of order implementation same somewhere if you can learn this then next year you can look

at it and appreciate how it is being done. So, we have handled all WAR WAW and raw we have done operand forwarding we have done register renaming. So, what is not taught which is also not taught everywhere in the globe as a part of your curriculum means; how this instruction fetch unit and the reorder buffers are basically designed the real digital design then everything else is taught.