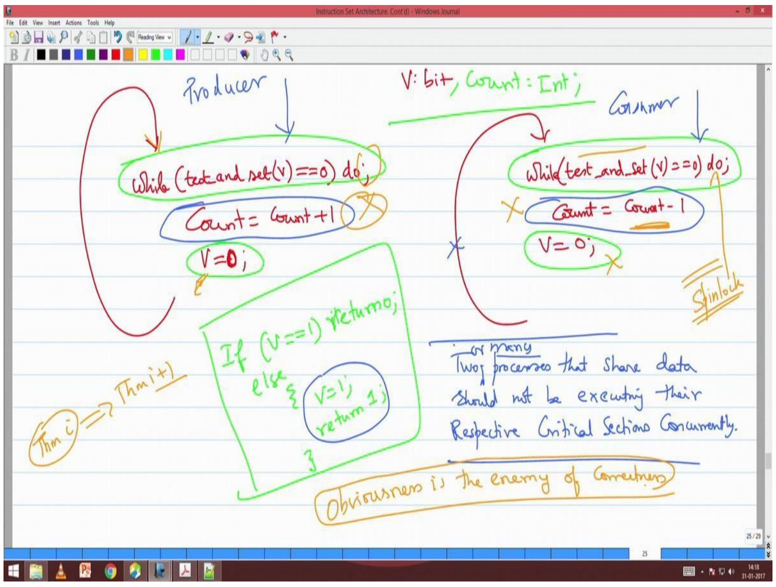
Computer Organization and Architecture

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

Lecture - 14
Atomic and Predicated Instructions (Contd.)

[FL]. So, yesterday we were last class yesterday right yeah yesterday we were talking about test and set ok.

(Refer Slide Time: 00:26)



V: bit, Count: Int;

Producer

```
While (test_and_set(V)==0) do;
Count = Count+1
V=0;
```

Consumer

```
While(test_and_set(V)==0) do;
Count = Count-1
V=0;
```

Spinlock

```
If (V==1) return 0;
else {
    V=1;
    return 1;
}
```

Thm i => Thm i+1

Two or many processes that share data should not be executing their Respective Critical Sections Concurrently.

Obviousness is the enemy of Correctness

Count equal to count minus 1 I had a shared bit b which was a bit. So, what I did here was while test and set V equal to 0 do here also while test and set V equal to 0 do. So, what I have done here is that I just added this instruction and this instruction right. So, what will test and set V do if V is equal to one. So, I may just keep interchanging this for out understanding today may be it is different from what I taught yesterday does not matter V equal to 1 it will return 0 else it will make V equal to 1 and return one if V equal to 1 it will be always be returning 0 and else V equal to 1. So, this the test and set operation and V is a shared variable count is also a shared that is not a bit, but it is an integer and this will also keep.

So, this is the producer this is the consumer; now how will you prove see what we want what was the consistency thing my consistency will be completely adhere to; that means, the number of elements in the buffer is equal to count, if I ensure that these two statements are not executed concurrently. In some sense when producer is executing count equal to count plus 1 consumer should not execute count equal to count minus 1 and vice verse, these two group parts are called the critical sections of both. In some sense what you essentially mean here is that two process is that share data should not be executing their respective concurrent sections respective critical sections concurrently, this is the statement this statement, you will not find in any book I will stand 10 days upside down if you find it right.

So, two processes that share data in this case one or so, I can now say two or many processes I can just extend it, two or many processes that share data in this case count should not be executing their respective critical sections. So, producer’s critical section is count is equal to count plus 1 consumer is count equal to minus 1. There should not be in their critical section at the same time I gone they may be executing non executing for this is static. So, I will ask where were you if I ask producer he should not say I am in count equal to count plus 1 and if I ask the consumer you should not say count equal to count minus 1. So, when producer enters count equals count plus 1 your consumer should be in that red loop here anywhere outside that sections right and vice verse, when producer when consumer enters the critical section I should show that this should be outside.

So, how will you prove this you are understanding my question? When the producer is inside the blue box consumer should be on the red arrow, when the consumer is inside the blue box the producer should be in the red arrow how will you prove this. So, the first thing is I will give you. So, these are all proofs in concurrency if you do distributed data base you want to do machine analytics and all you should understand how data base works right. So, machine learning all these things you all go and study you will all become all of you are going to take machine learning. So, I am sure 100 percent, but when you do machine learning and all you should understand data base and when you want to understand data base you should understand distributed data base right data base which is spread across many places right and then you want to prove anything on these type of distributed computing. This is one proof you should remember right this proof is

slightly different from the other proofs that we see.

Normally how do you prove you use induction contradiction and all now we will use something different something it is like Maggie it is different ok [FL]. Now first thing let first time let producer and consumer try to enter this fellow, if they want to enter the blue box they have to come through that while loop correct. When they want to come through that while loop this fellow will be executing test and set this fellow also will try and execute test and set, initially V will be 0 right. So, for exactly one fellow I will get a one not for both exactly one fellow. So, he will also execute test and set and he will also try and execute both of them will try and execute, but somebody whoever has the CPU first that is the fellow who will get V equal to he will get one and come out of this while loop while the other fellow will not.

Who ensures this? When both of them try to execute test and set and there could be context switch between them, exactly one fellow will get you know will come out of this while loop; that means, exactly one fellow will make V equal to one will execute this part who will ensures this.

Student: (Refer Time: 07:28).

The test and set is an atomic instruction, the atomicity that is ensured by the architecture. Please understand it is not the software that is giving you this the hardware has that atomicity and that atomicity basically ensures that one of these fellow will enter not both now I am entered. So, let me check some other colour colour colour what colour what do you choose orange. Now I am inside when I am inside this fellow has to go on this (Refer Time: 08:01) because I have V I have made it one this test and set has made V 1 and nobody else is going to make it 0. So, this test and set will be what it will be doing? It will be trying to it will keep on executing and every time it executes it will be getting only 0 because V is 1 and this fellow cannot make V 0 by the routine right look at the test and set routine if V is 1 I has to keep returning 0 it cannot adjust the value of V.

So, this fellow will keep on executing while this fellow enter the critical section. So, as long as this fellow is going to be in this critical section are you understanding this fellow cannot come out of this while loop right you will be executing please note there is a semicolon here. So, this while loop will keep on going on a merry go round, or sadder sad go round. So, whatever fine. So, me going on a merry go round in operating system parlance is called spin lock, lecture will teach about this great in great detail in the next semester when you do operating system, but spin lock I will ask him to ask you what a spin lock then I will find out who all smiled. So, this is called spin lock. So, why is we will come to that now what will happen this fellow finished this critical section, then he make V equal to 0. The once he made V equal to 0 then the next time this while is going to execute he will make he will get he will come out he will get V as 0. So, he will make V 1 and come out.

So, once this fellow enters right again this fellow will come here he will be doing test and set as long as this fellow is here he has the lock and this fellow has to wait correct this fellow has to keep waiting. Now only after he finishes this consumer finishes count equal to count he will come and make V equal to 0 then this fellow will come. So, when this fellow is on the red this fellow can be in the blue when this fellow is in the this fellow is in the blue, this fellow has to be in red and this fellow is in the blue this fellow has to be in red, this is a proof this is formal proof are you able to understand and please this proof the nucleus of this proof if that is wrong everything is wrong right. Somebody wrote a paper where there are 10 theorems theorem I implied theorem I plus one. So, 10 theorems I proved everything was correct except theorem 0, you know what happens everything goes. So, be careful first theorem that you prove you should be. So, note this down in your things you will not appreciate it now obviousness is the enemy of correctness yeah. I may ask one n some question what are all the jokes other things I may have tell at least three jokes, I whatever you call p j's or whatever you may.

So, that is what you should be extremely careful in this first thing. So, the nucleus of this proof rise in the fact that test and set is atomic, if that is not atomic this whole thing goes. So, what where is the ragavas why cannot to put increment memory right there is only one single instruction. This is not one single instruction internally I will come I will basically deal with it in some great detail as we go to memory management later, but please note that this is a toy example for me to explain why it is working right. This critical section whatever you put inside will be a big program thug program. So, there

will two thug programs inside. So, it is not an it is going to be a simple thing, but to explain you I am. So, when you do the operating system course again you will be having something called a super block, the Unix file system has a super block that super block is one structure which you need to there will be lot of concurrent transactions on this concurrent means that it is shared by many processes any fellow wants to open a file has to touch the super block in some way or other right that super block has to be you have synchronise it.

So, how well do you synchronise? If you say entire super block I block. So, I put a entire code of the super block inside this critical section for everybody then what will happen? You touch the super block until you finish that fellow cannot do then the system will be working what slowly ok ha because the entire operation becomes. So, that is the crux. So, when you are trying to understand file systems the operating system is also very good for machine learning please understand do not neglect operating systems because there you will understand how synchronisation see under see the entire data analytics, and machine learning is done on top of data bases the best way for you to the best data base you have is a file system in my opinion ok.

Now, data base architects may disagree with me, but you have to understand the file system. So, I need this type of see what is happening here? Only one fellow can get in means what you are basically stopping that concurrent execution I am serialising it are you understanding there is a word I am serialising the access to the respective con serialisation you will learn about this in both data base, and operating system I am serialising it here I am this fellow can enter then only the next fellow then only this fellow next are you getting this. So, if by putting these type of barriers and synchronisation I serialise the whole operation and if I make too much of serialisation then what happens your performance suffers quite a bit. So, these are all some very interesting issues you will see. Now I want to ask two questions before I go to the next predicated instructions.

So, now see your brain has to start thinking concurrently, it should not just look at the producer process it should not just look the at the consumer processes it should look at it together and that is a training you need because today you do not have a single core

system simply we want go and buy a 83, 86 core it is 5000 dollars last piece in the museum right. So, even your mobile phone today have 4 8 cores. So, you now have to. So, the next generation when you go start programming whatever you are going to do next 4 to 5 years you will get more money or you will get research problems wherever you are you will find your life interesting, once you start thinking concurrently. I think that is one thing that is why we all want to bring up these type of issues. So, why I am teaching this in computer organisation I am sure many many 99 percent of the computer c o courses will not teach this why we are teaching this here is because we need you have to start thinking concurrently. Now think concurrently and tell me one problem in this in this implementation of this producer and consumer I will give you two.

Student: When this is in critical section and then there is a switch over to the consumer then it gets stuck in the do loop. So, instead of being over there it can come back to the producer.

One good thing yeah this is a answer, but it is not real answer I am asking for an, but any way this is a good answer this spin lock is just waste of time right correct I am just doing I am just doing while again while do while do while nothing then there is there is only a semicolon there the spin law actually is a waste of time. So, many times you go and evaluate systems become slow right wherever you are in you are in Goldman Sachs or Uber or whatever job you take hopefully Intel or a m d or arm system suddenly becomes slow, but you will find CPU utilization to be 100 percent no transaction is happening. So, what is this CPU doing dancing a jig or what it will be on this type of locks. So, spin lock essentially see the CPU is executing this again and again are you able to appreciate this while loop is executed by the CPU the same PC again comes again again again again and goes, but then spin lock essentially kills performance. So, it is useless for us. So, now, what next operating system will teach you I will not go in to that; so spin lock is a problem.

But test and set will be used that context of synchronisation without spin lock. So, they introduce concepts like semaphore etc we will let the of course will teach you. Next what is another interesting problem? Here you have to again you are talking about one process you are talking about one while loop, but in the context of other while loop having

executed, but can you see both together and give me some interesting thing.

Student: (Refer Time: 17:23).

Still that this is this both the consumer and that execute on the.

Student: On different.

On different cores. So, the atomicity is based on a on a data. So, the memory management has to take care of it.

Student: (Refer Time: 17:42).

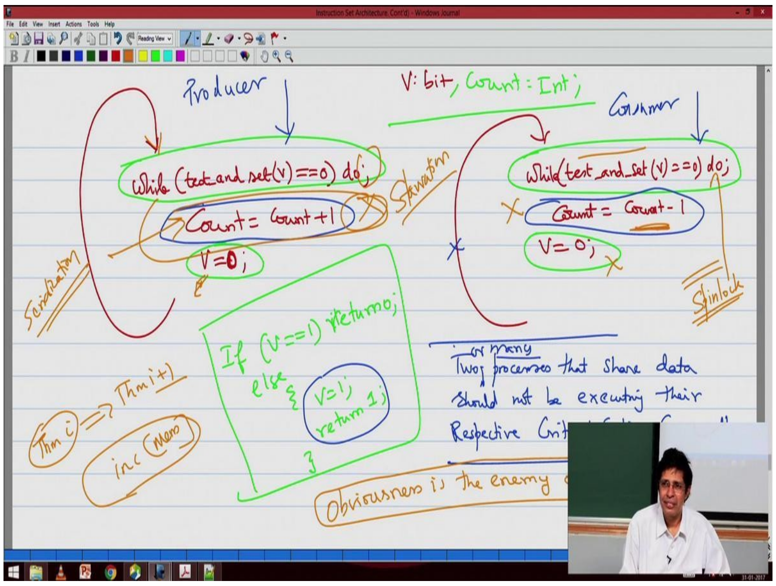
It is atomic in the sense if I am access if I am having instruction on particular variable right. So, that implementation should be. So, what is grade about this implementation I have 2 cores, 2 cores that thus test and set we one is doing this and one fellow another fellow is doing test and set now I have to serialise that. So, it is a part of what you call as some moment of what you call as like you know I will not use that word now it will confuse people, but there should be some sort of coordination that need to happen.

Can you tell me some other problem like functionally? So, functional issue it is not an implementation issue. When you log in to Gmail you want the mail box come up right what if the mail box does not come up. Can you tell me some issue with these two programs I gave you a example also.

Student: While loop keeps on executing and (Refer Time: 18:52).

Very good very good give a clap for that guy man good suddenly sound cut good. So, one fellow can be made to starve, imagine I got I am a very greedy producer I or my operating system. So, luck is that I get V equal to 0 I enter, and I make I finish of my critical section and this fellow is waiting here, before he could execute the next version of that while loop I can again come here test and come in. So, I can there is a possibility because the context switch should happen at the right time so, that this fellow comes out. So, this red path is so fast that I keep on producing and this fellow consumer keeps on waiting and if suppose I have one million buffer of one million size. So, one million times this fellow will keep waiting possibility there is a theoretical possibility.

(Refer Slide Time: 19:48)

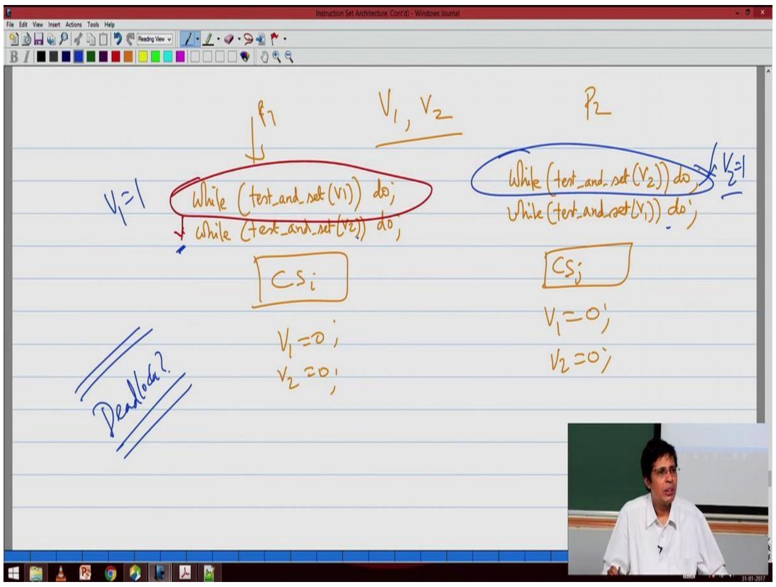


```
While (test_and_set(V)==0) do;
Count = Count + 1
V=0;

While(test_and_set(V)==0) do;
Count = Count - 1
V=0;

If (V==1) return 0;
else
{ V=1;
  return 1;
}
```

Two or many processes that share data should not be executing their Respective Crit

So, there is possibility of what we call as starvation on one fellow like consumer also, if then a producer cannot produce then the consumer can be consumed. So, you are you are able to follow. So, all these things need to be addressed. So, using the same test and set instruction we will come out with solutions that does not have starve starvation that that will not land on dead lock.

(Refer Slide Time: 20:28)


Instruction Set Architecture, Cont'd) - Windows Journal

V_1, V_2

P_1

```
V1=1
while (test_and_set(v1)) do;
while (test_and_set(v2)) do;
CSi
V1=0;
V2=0;
```

P_2

```
while (test_and_set(V2)) do; V2=1
while (test_and_set(V1)) do;
CSj
V1=0;
V2=0;
```

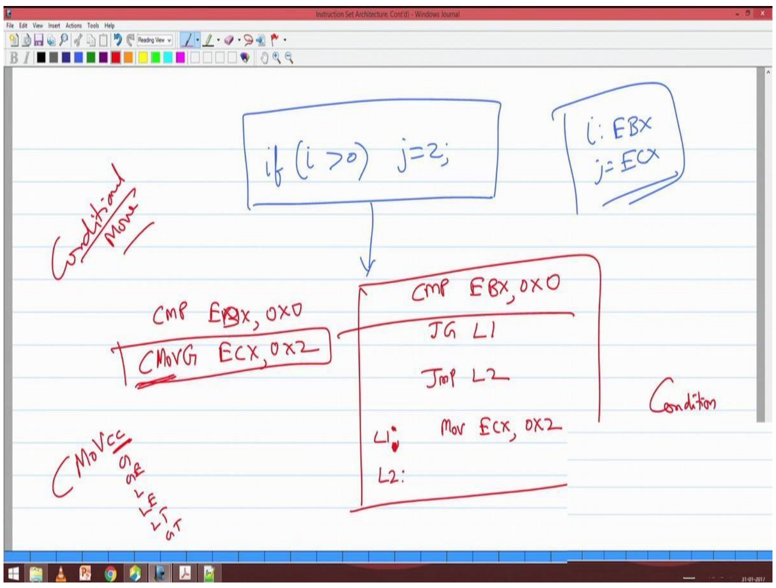
Deadlock?

So, can you create a dead lock with this, what is a dead lock? I will test you. So, let us say there are two bits V 1 comma V 2 this is one program P 1 somebody else wrote P 2 where he said while now you understand what will happen now this fellow comes here. So, luck as it he finishes and he is about to come here when he is about to come here context switch happens and this fellow comes here. So, this fellow I have made V 1 equal to 1 and he is here this fellow has made V 2 equal to 1 and he is here. So, this fellow is waiting for V 2 to become 0 which only this fellow can make because V 2 is in his hand and this fellow is waiting for V 1 equal to 9 very simple thing. So, there is I am waiting for you are waiting for me nobody is ready to merge then we get in to a dead lock correct. So, this is a example of a dead lock ok.

So, now what is there to study in operating system? All these things you have to study. So, I have to give a way of synchronisation that will not starve me that will not leave me to a dead locks so many things are you able to appreciate these points ok. So, again I repeat giving a facility is like building dam writing proper software is to fill it with good water otherwise only dam will be there nothing will be there right. Now test and set is a very beautiful dam they handle synchronisation, but then you have to very careful when you use it, if you do not use it correctly you will if you do not if you cannot visualize think in parallel. So, what was the problem? Like when these fellows wrote the program they did not visualize what will happen when we both execute together, that concurrency is not visualized and that is why you land up with such type of problem. Are you getting what I am saying right? So, the most important thing is that whether you learn c o not at this point start looking at you know parallel programs concurrent programs and this will help you in both your networking and o s course in a very big way.

Operating system is a outstanding example of multiple processes execute together and they also share data followed are you able to get it. So, why atomic instruction there is a great need from the operating system end and then the operating system has to be extra careful by while using this facility. Now I will just spend another 5 to 10 minutes maximum on doing this.

(Refer Slide Time: 24:11)



```
if (i >0)  j=2;
```

```
i: EBX
j= ECX
```

```
CMP  EBX, 0X0
CMOVG  ECX, 0X2
```

```
    CMP  EBX, 0X0
    JG  L1
    JMP  L2
L1:  MOV  ECX, 0X2
L2:
```

Now that you have all done some assembly program let me say I is currently in EBX, j is in ECX how do you translate this program? Compare jump greater than true l 1, l 1 bellow ECX not necessarily the best way to transfer. So, I can say jump less than or equal to l 1 and put whatever. So, anyway, this is one statement now there is something called compare EBX 0 x 0, C move G. So, C move is called conditional move do this moment if the greater than flag is set this G is. So, in the Intel instruction set architecture you will see C move c c small c c there you can put G, GE, L, LE, LT, GT all these things; c c are all the condition code c c means condition codes. So, 1 2 3 4, 4 instructions actually became one instruction.

So, what are the advantages? So, this is called a conditional instruction or what we call as predicated instruction, that instruction will execute when some condition is satisfied some predicate is satisfied we call it as condition is satisfied as for as quick understanding. So, what is the condition here GGE and all. So, this C move this conditional move will execute only if that condition is satisfied. So, what do you gain and loose here shorter code and why shorter code is very important because your program size will be less for every every large loop I can basically go and put this predicated instruction and things will start working. The second thing is we are going to do is pipelining, these jumps and typically this conditional jumps will introduce what we call as you know control hazards remember this name we will talk about this in the next few classes. So, these jumps these conditional jumps like j G and all we will create something called control hazards. So, it is better to avoid you know these type of jump instructions. So, there this conditional move will play a role, the disadvantage is if the condition is false this entire instruction is a waste this is fine the condition is true then the status we wrote ok.

So, with this we come to an end of the instruction set architecture discussion on instruction set architecture, we have seen different addressing modes we have seen different types of instructions and then we actually good took a case study of multimedia instructions atomic instructions and you know the predicated instructions, these are new instruction varieties that are introduced in to a instruction set architecture with a view to reduce the total memory consumption. So, with we come to the end and in the next class we will start on pipelining.

Thank you.