Computer Organization and Architecture

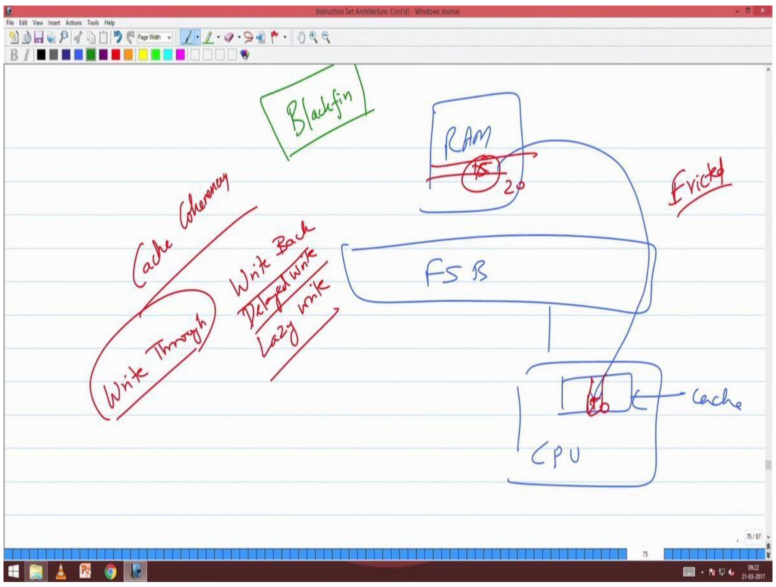
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 34
Cache - Cache Coherency, Dual Ported Cache

So, yesterday we saw the simple matrix addition algorithm that just swapping of the loop can actually make the algorithm run much faster, while the other way it can kill performance. So, these are some very very small, but very important settle integrity is that we need to settle details that we need to keep in mind, when we write compilers and this type of a knowledge will come and only when you have this global view of how and you know system is organized. So, that is the most important point. So, just learning organization in isolation will not serve any purpose. So, this is one very interesting stuff.

So, let us now go and start. So, what are the challenge in this whole cache organization right.

(Refer Slide Time: 01:04)



So, let us see there is a ram there is a front side bus, and there is a CPU and here is the cache. nNow when I want to read an location I fetch it from the ram and put it in the cache and I keep accessing it. Now what is the problem here what could be one very important problem program will not write. So, the CPU will only interact with cache it will not good. So, yes. So, this problem of what he set is called cache coherency problem. I fetch something here it was some 15, I fetch this it was 15 here also now I changed it to 20 when will I change that to 20.

So, there is a difference. So, for a given memory location please understand that there are 2 versions of values that are stored one in the memory one in the cache correct right. So, if I touch this if I change in the cache then the version that is stored in the cache is different from the version that is stored in the memory. So, there is a difference right the latest value will be always in the cache while the you know a older value can be there in the memory right. So, this difference is basically called the cache coherency problem right. As far as I have only one processor this cache coherency is not going to be a big issue for me because always I am going to read from the cache and if it is there in the cache anyway I am going to get the latest value, but when I go to multi cores where I have 4 cores and they are all sharing a memory, then there is going to be an issue we will come to that when we address multi core memory organization, which will follow say from next Monday ok.

So, cache coherency problem is addressed in 2 ways traditionally there are the solution to cache coherency forms the basis of classification of a cache right. So, there are based on just address in cache coherency, cache is classified into write through caches and write back caches ok.

Student: (Refer Time: 03:40).

No only one processor, but when 2 process start accessing. So, you have a cache I have a cache you have taken 1000 from the memory some location 1000 and you go and change that value I am trying to access 1000 you have that 1000. I do not have access to your cache. So, these type of problems will come very interesting and I will tell you some simple protocols how to handle it you will we will discuss this as we go through the next part.

But now just a single processor now still there is a cache coherency is still exists, there are 2 versions of the same memory one in the memory itself and another is the another in the cache and both carry different values, but one thing that we should notice the latest value is always in the cache. Now based on how to handle this coherency problem caches are classified into 2 write through cache and a write back cache right. In a write through cache when I write into a location I also go and immediately update the memory.

So, when I change this to 20, I also immediately go and change to 20 here. In a write back cache I only change this and whenever this is evicted.

So, this word evicted is used evicted means that something else collaged into this and so, I have to throw it off meaning I am evicting of whenever I am evicting it I have to go and write it write it. So, this is right back or this is also called the delayed write or so, many names lazy write so many names lazy write protocol and I am same. So, do you get this right? So, I write back in in the case of write through when I change this location immediately I will go and changes in the memory.

The biggest disadvantage of write back of write through is when I do not actually you know enjoy the speed of the cache access, every write is equalant to a original memory it is as if whether the cache existed or not right. It does not matter because every write I have to go and write I have to access memory and keep writing into it your are you getting this, but on the other hand if it is a write back cache which is the state of the art today, I only write back when I am whenever I am getting evicted. As long as I am not getting evicted or I am not getting invalidated right I do not really go and write back it will be there in the cache itself. So, this is this write back is also called as delayed write or lazy write I am lazy to write into the memory I will go and do it at a later point. So, this is a difference.

Now please understand that that write back is more you know heavily practice now, and with write back alone we can get that big performance with write through we may not get that much big performance. But then there is an impractical thing that the amount of things that you read is much higher than the amount of thing you write correct right. So, the this is the second we can check up check this up, the amount of loads is lesser than the amount of stores sorry it is more than the number of amount of stores load is read from memory store is write into memory that we can see right even the amount you read and amount of notes you write right fine. So, write through and write back. So, let us now concentrate on the other types of cache; see what happens here before going to that let us see because of cache what is going to happen. So, one more point here, now in the example that I have talked here I have talking about one single location correct.

Now this write through policy is very good when I am talking of one location, but if you say I have a block right. So, today my granularity is not one location right what I fetch from the memory into the cache or what I write back into the memory whatever I evict, that unit of transfer between ram and the cache is not just the one byte it is one block.

Now, when I am looking at a block then this right through will kill because when I write into one location I mean I am writing the entire block back correct. So, otherwise I should have an assume it what I read is a block what write is a byte and imagine this all these things have is not a software c language program, but it should be execute you know hardware to do this. So, it becomes extremely difficult to construct or you know design such hardware right. So, this write through policy becomes a much more headache when I am looking at block caches right where my unit of transfer is a block from the memory to this thing and vice versa.

While in the case of you know the write back that is where pretty happy. So, I will evict whenever I evict I would have done enough changes in the block and then it is good that i. So, this is extremely this is one point that I wanted to tell here. Write through will not increase efficiency write through is still there because at early stages people were not having proper mechanisms to actually address the cache coherency issue right the point is. So, when I am really talking of somewhere will write through work right. So, this will be in some of the DSP processors we have seen write through this digital signal processing server where there is very very few writes you keep only reading right. So, it is something like that is that is the reason why I came out with this reduction function, you read 100 numbers and you output one number it is an addition some of this right. So, DSP process will have this fewest multiply add a plus b into c that is a multiplication plus and addition. So, this is an operation.

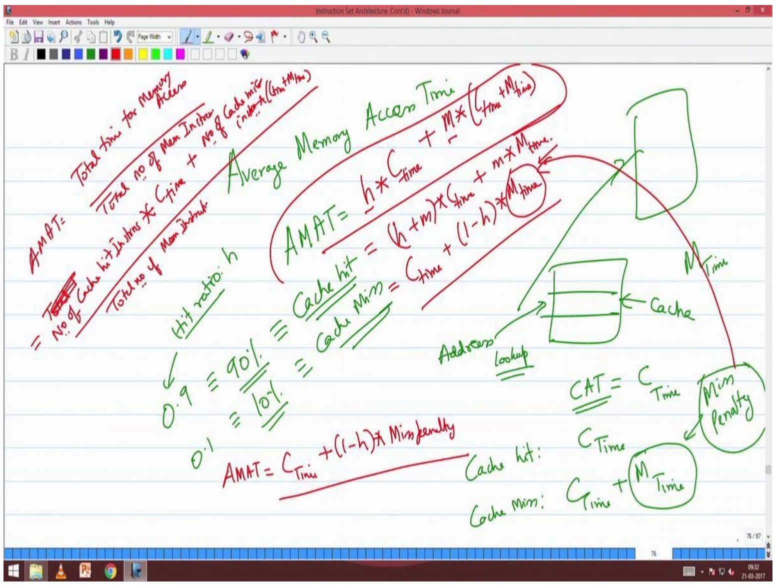
So, then what will happen is you keep on reading lot of things you keep reading right from the memory and then multiple read you do everything, but finally, go and write only one. So, the number of nodes will be phenomenally hide that the number of stores right and that is the application for which it is destined for that DSP process that is the domain. So, why should I break my head putting all these write back policy? See implementing write back is going to be an heightener. Just coming back write back is going to be a very very difficult implementation strategy right when should I shoot this mosquito with a revolver right that is the question that comes when you go and look at DSP processors I have few writes [FL]. So, should I do all these stuff right put some heavy thing look at I r u all these non-sense, why should I do let me go and write through that is a very that is a most intelligent decision you can take right.

Student: (Refer Time: 11:09).

I could also have I actually decided can I just keep accessing something from ram, but number of reads are quite high for example, I start doing a simulation the number of repeated reads of a location is also high right. So, I keep reading the same location again again again again right for say some nitrations for then finally, I will only go and put only one word one value back. So, the reads to write is very that that ratio is very high. So, that these are the cases where write through will work right. So, DSP processors are some examples, I think back fin had black fin there are processors called black fin why I do not know, but name I do not know like why my aunt named me come Shruti like that something like. So, black fin, you can look at black fin. Actually there you can even completely disable cache and use the cache itself like a memory ok some lot of interesting things can come fine.

Now, now we will do some analytics, I been talking so, much practice. So, let me talk some theory you have an algorithm example also this days. So, let me just talk some theory. So, we have been telling that if I have a cache I am going to get lot of advantage now what is that advantage right. My memory access time is going to reduce I cannot say every memory access time there will be some memory access where the it is not in the cache then I will going to get lot of I am going to actually suffer quite a bit if cache is not there I have to go to memory and fetch it, but, if I take a entire program, the better way of assessing the impact of cache is to have this parameter namely what we call as average memory access time or what we call as AMAT correct.

(Refer Slide Time: 13:11)


AMAT: Total time for Memory Access / Total no of Mem Instrn
= No of Cache hit Instrns * C_{time} + No of Cache miss instrns * $(C_{time} + M_{time})$ / Total no of Mem Instrns
Average Memory Access Time
Hit ratio: h
0.9 ≡ 90% ≡ Cache hit = $(h+m) * C_{time} + m * M_{time}$
0.1 ≡ 10% ≡ Cache Miss = $C_{time} + (1-h) * M_{time}$
$AMAT = h * C_{time} + m * (C_{time} + M_{time})$
$AMAT = C_{Time} + (1-h) * Miss\ penalty$
Address
Lookup
Cache
M_{Time}
$CAT = C_{Time}$
Miss Penalty
Cache hit: C_{Time}
Cache Miss: $C_{Time} + M_{Time}$

I am interested in the average memory access time if you take worst case it will be. So, I have to just go to memory and bring it right. If there is a miss then I have to go to memory and bring it so that is going to be much costlier for me and even when I want to read one byte what I am going to read I am read some 256 bytes. So, the worst case memory access is going to be very large for me. So, what I am interested is overall I am spend some k times. Now I am interested in what is the total time spent by the for memory access divided by the total number of instructions that access total number of loads stores right total amount of memory spend for load stores total amount of time spent for load stores divided by the total number of load stores, that is my average memory access time.

Now, I have to look at I have t get out with the very expression for an average memory access time in the presence of a cache are you getting this right so, AMAT just not memory access time, but AMAT now. In this context let us go and see that what will happen. So, this is a cache I have an address with me, and I do a look up on this cache look up on this cache whether it is there or not. So, that will be. So, how much time I will spend I will spend cache access time right, this is called let me call it as this is the cache access time let me call it C or C T C time right.

If it is not there so, if a cache hit happens the time is C time right. If it is not there if cache miss means what will happen? I will go to the memory and fetch it form there that is m time. So, if there is a cache miss what is the total time spend C time plus M time.

Student: (Refer Time: 15:58).

Oh that is included in that we will see that is this memory time right there will be some constant time that memory will have because there will be you know constant number of memory access. So, we will just take it as M time. So, this is C time plus M time. So, what I M time alone this is called miss penalty, the penalty I pay that is the extra time I spend because there was a miss is called miss penalty right.

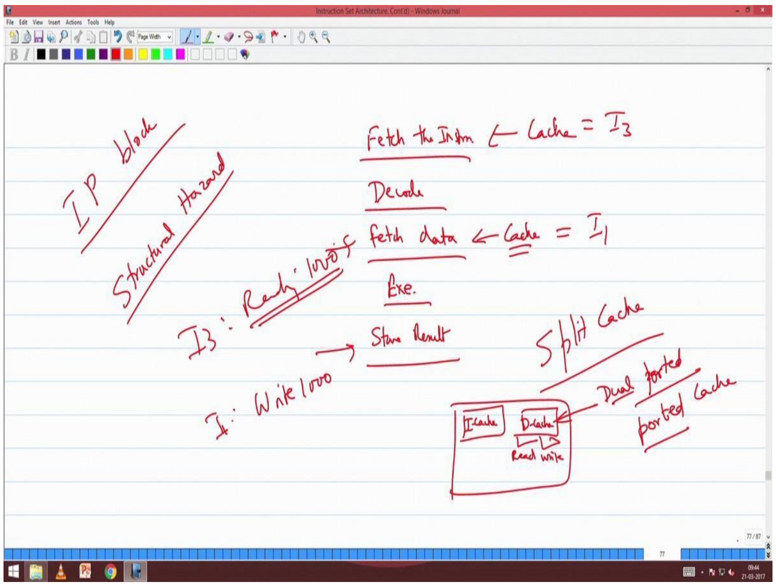
Now, what I do is 90 percent of my memory access is a cache hit and 10 percent is a cache miss. Let us say in the overall execution of all the memory access instruction 90 percent of that was a cache hit and 10 percent is a cache miss. This 90 percent which is equivalent to 0.9 and this is equivalent to 0.1 this is called hit ratio this is called miss ratio let me call this as h let me call this as small m. So, your average memory access time is nothing, but hit ratio in to C time plus miss ratio into C time plus M time how do you get this are you able to are you able to understand how I am getting this how.

So, what is an average memory access time? Total time for memory access divided by total number of memory instructions correct this is equal to total time of number of cache hit instructions into number of cache hit instructions into cache C time plus number of cache miss instruction into C time plus M time whole divided by total number of memory access instructions because number of cache hit instructions if that is cache hit I would suspend C time there. So, that is number of cache hit instructions in to C time plus number of cache miss instructions, instructions which are not avail instruction or data which landed up with cache miss that would suspend C time plus M time.

Now, number of cache hit instruction by total number of instruction is hit ratio number of cache miss instruction by total number of instruction is miss ratio right you got this. So, this is how I derived this AMAT is equal to h in to C time plus M into C time plus M time this is nothing, but h plus m in to C time plus m into M time what is h plus m 1. So, this is cache access time plus 1 minus h into memory access. This M time is actually called as what miss penalty. So, just to sum up AMAT is equal to cache access time plus

1 minus h into miss penalty right got this. So, just copy this if you have any questions in this particular derivation you may ask me.

(Refer Slide Time: 21:51)


Instruction Set Architecture Cont'd) - Windows Journal
IP block
Structural Hazard
Fetch the Instn ← Cache = I3
Decode
I3 : Read 1000 → fetch data ← Cache = I1
Exe.
I1 : Write 1000 → Store Result
Split Cache
I-cache
D-cache
Read
write
Dual ~~ported~~
ported Cache

Now, because of cache what could happen? So, let us start with let us go back to over pipeline see pipeline will not leave us still eternity, now what are the five stages fetch the instruction.

Student: Decode.

Decode fetch data execute store result now this type of an execution profile in your control unit in your CPU along with cache when both of them coexist what would be the issue what would be the most show stopping issue. Now you are done that assignment there no then what did you do in fetch the instruction what did you do in fetching the instruction what will you do normally.

Student: Take it from (Refer Time: 23:01).

Take it from.

Student: (Refer Time: 23:17).

Wrong when this in this case in our CPU from ram from ram means what from now in this context after yesterday todays class ram means some cache this has to access cache now you tell me how you know the answer.

One stage is over second stage third stage just look at it then what happens next stage it will decoding next stage fetch data from where.

Student: Cache memory cache.

Who is fetching data when some who is fetching instruction what is fetching will. So, can you just see there is some problem here when I ones data is being fetched I 3 is being fetched I 3 as an instruction is fetched and I ones data is also fetched from where.

Student: (Refer Time: 23:58).

From the same cache. So, what happens? Both follows are trying to access the cache correct both fellows are trying to read from the same cache; that means, there is a sharing of resource and when I am saying that resource after now see serialize it. So, I should allow the first fellow to fetch data from the cache then allow the. So, there are 5 unit right one unit is fetching the instruction one unit is decoding another unit is fetching data. So, I have to serialize the action between fetching data and fetching instruction both cannot go concurrently, I have only one cache I have only one interface. How can two fellows come and read and write read from we at the same point of time are you getting this point yes or no right what are you able to follow what I am saying right.

So, how can these 2 fellows come and keep accessing me at the same time cannot right. So, what sort of hazard it is, this is going to stop the pipeline because I have only one cache I one is asking for data from the same cache at the same time, the fetching data is saying fetch I from the same cache right. So, there is a contention for the cache right. So, this type of hazard is called.

Student: Structural hazard.

Structural hazard very good right. This structural hazard we have seen earlier, I have when there is no execution unit that I have ten add instruction that I could execute together, but I have only 4 add units. So, that remaining 6 has to wait this is structural hazard. But the structural hazard had another interesting case where structural hazard will come is that when I am implementing, the pipeline there will be one stage it is asking for a data another fellow is asking for data both fellows are asking for an instruction and both fellows are attacking the same cache and that will lead to a structural hazard yes or no right.

So, now how do we solve this problem there you go into what we call as a split cache what is split cache nothing great. So, what I do I have one cache called I cache another thing called D cache what is I cache.

Student: Instruction cache.

Instruction cache d cache.

Student: Data cache.

So, in if I fetch data I will put it in D cache if I fetch instruction I will put it in I cache. So, I and d may not I and d will not even be same though. So, instruction cannot be data; data cannot be instruction right. So, when I do the fetch one minute when I do the fetch I fetch only from the I cache when I do a data will fetch only from the d cache. So, these 2 are accessing 2 different caches, and hence this structural hazard is avoided is it fully avoided yes a what man go and see there more stages I have write there.

I one is storing and somebody is fetching data. So, that is why this d cache will be a multi ported cache one is say read port another is a write port. So, this cache normally will be a dual ported p o r t e d it will be a dual ported cache dual ported in the sense that one port can be a read port another port can be a write port right. Now what is the problem when I have a dual ported cache?

Student: (Refer Time: 27:31).

There will be data; obviously, right when this fellow is trying to read this fellow is trying to write both are reading and writing from the same location there will be a data dependency. Now there is another way by which it is actually handled this this is actually a data hazard where I am writing and reading right this is a read after write hazard and we have already know how we are handling read after write hazard, but when I am looking at memory when I am looking at cache as a separate entity because today cache is a separate block I can make an IP and keep selling that I P I have to look for this. So, how will I solve this problem?

When there is a tie I will allow give preference to the write correct it is obvious because every time I read I want to get the latest value of the location. So, when there is a write and a read for the same location I will give preference to write. So, that is how dual ported caches are built. So, what we have seen now I have I cache and a d cache why did a splatted otherwise in a pipelined environment I will get a structural hazard. Now the next step is that even after a split it from my cache if split it into I cache and d cache, again I will have an issue between the fetch data and the store result stage correct. The way I handle that is by having a dual ported cache and in this dual port again I make a story saying that not story it is a reality that I should have higher preference for write when I read and write from the same location.

Why do you give preference to write because let us say I am writing into 1000. So, I one is writing into 1000, I 3 is reading from 1000, what should be the value of 1000 be it should be I ones result right because always when I am somebody is writing he is he will be an instruction before you. So, between a write and a read I have to give preference for write.

Student: (Refer Time: 30:12).

Yeah those things are there. So, that is what I am saying if the if you keep the tomasulo algorithm that we are already seen, then this type of resolution itself is not necessary because there itself this read read read after write would have been recognized, and the hardware would have stopped it. But when I am generating a cache, cache is an intellectual property block IP block. So, this cache is also an intellectual property block. So, people may caches separately independent of how the CPU is done and they sell it. So, they make money now and they want they do such a thing then they have to handle this because they may give it to you a dead processor right they may not give to intelligent give tomasulo processor, they may give it to some stupid processor which will not handle all these type of things. So, then when it comes then the cache has to handle.

So, every stage we will put this type of you know you know safe guards to see that your program ultimately gives the effect of having be executed in a in the way it is listed what you call as the program order. So, that is why you know even though we when we give a dual ported cache write will be given more privilege that read right. So, if there is a tie between write and read, write will happen first then read.