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

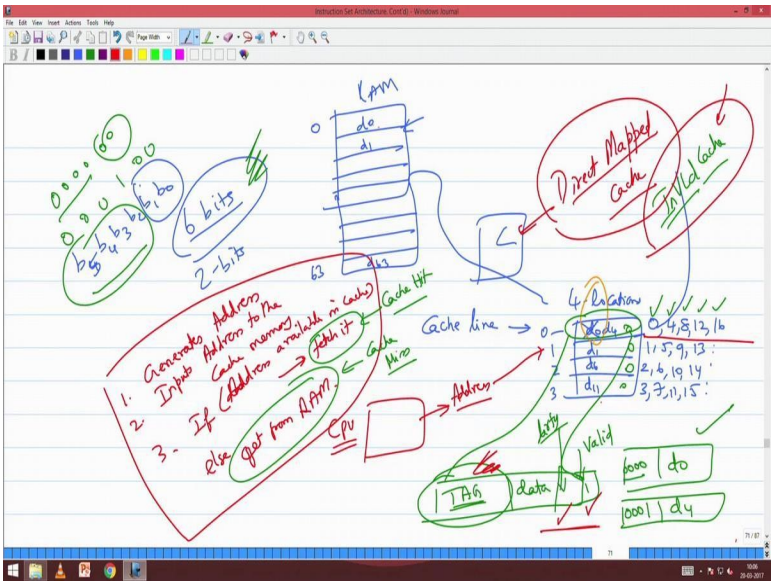
**Lecture – 33**
**Cache Organisation**

Good morning. So, last class we.

Student: (Refer Time: 00:19).

Discussed about cache, and we solve this algorithm that I put on the screen whatever I have written in red.

(Refer Slide Time: 00:21)




So, this CPU actually generates the address right. It gets converted into an effective address. So, it generates a logical address it gets converted into an effective address, for the further for the current class let us assume that there is no paging. So, it generates an address, and then this address is input into the cache memory, and if this address is available in the cache, we go and fetch it. This is called a cache hit. If the

address is not available in the cache, then we go and fetch it from the RAM, put it into the cache and then consume it then it is called a cache miss. And how do we go and check whether a particular address is there in the cache or not? We gave a very simple example of a 6 bit memory whatever I am marking here; let me use the green pen here.

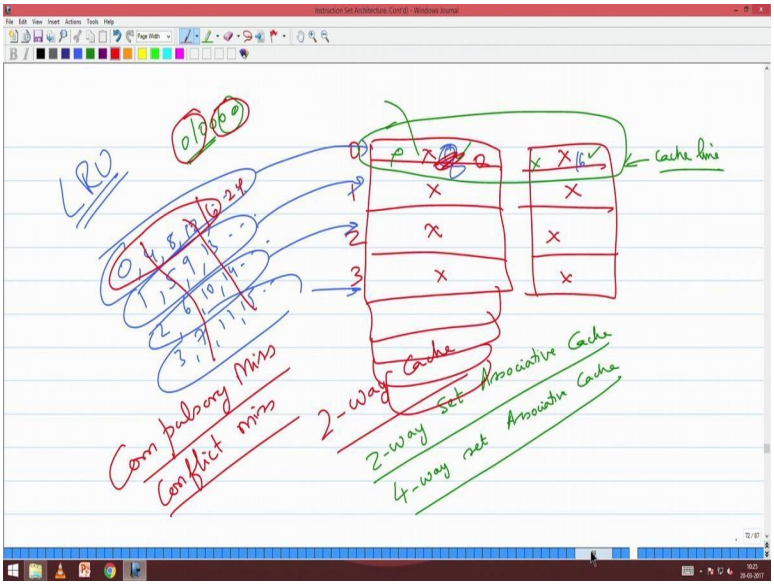
So, in which and I have a 2 bit 4 location cache. So, I use the last 2 bits into index into that location, and the first 4 bits as tag bits, as you see here. I am marking in green. The first 4 bits as tag bit, the last 2 bits are used index into the location, and then there is a valid bit and then invalid sorry, a valid bit and a dirty bit for obvious reasons. We have we have discussed this in the last class.

And every instruction set architecture again I am marking in red here on your right and top, has an instruction called invalidate cache, which will go and make the valid bit 0. So, when I generate an address in this context I take the last 2 bits and index into this location, if I find the valid bit to be 0, then it is a cache miss. If I find the first that, if the valid bit is one then some address is stored there, I take the first 4 bits namely b 2 to b 5, and compare with the tag if they match, then it is a cache hit, otherwise it is a cache miss right. So, this is how cache is basically work.

Now, let us go and enhance this idea. So, this is called a directly mapped cache or direct map cache, because there is nothing that I am translating here. I am just taking the direct address that is generated by the CPU just using 2 bits and directly mapping it onto the cache. What are the serious disadvantage of this is that there is lot of collisions right. Collision in the sense 0 4 8 12 16 all these addresses collide, collide they are all stored in the same if at all they are going to be stored in the cache they should be stored in zeroth location. So, when 0 is there I want to now access 4 0 has to be taken out and 4 has to be put there, this is called collision right.

So, in data structure course you would have learned hash function right. And collision this is a very, very trivial case of hashing. We cannot do we do not want to do anything more than this because, it is done at an instruction level. I do not want you know a Montgomery hash function to sit there k or a. I want to have a very, very simple mapping mechanism; I do not want a very complex mechanism because it will hit what? It will hit the timing performance, your instruction execution or your throughput number of instructions for clock cycle will go down right. And that is why I would like to have a very, very simple hash function to start it, are you able to follow?

(Refer Slide Time: 04:20)




Now [FL] let us go to the next part, let us see what is advanced in this? Now, as you see here in the zeroth location 0 4 8 12 so many addresses mapped on to 0. So, many on to 1, So many on to 2, and So many on to 3. And what happen if 0 is stored now 4 has to be store immediately at replace 4. Instead what I will have is, instead of one set of locations; let me have a pair of locations. So, these addresses this 0 4 8 12 can be either stored here or it can be stored here right.

Similarly 1 5 9 13 can be either stored here or here, here or here, here or here. So, this is called a 2 way cache. How do you, how do you implement it? So, I take 0 suppose, I take some address let us say it is mapped on to this location, let us say some 16. 16 gets mapped on to this, 4 if at all 16, 16 is stored it should be store either here or it should be stored here. So, I know which line. So, I told you right, last class this is called a cache line right. I know which line this address could be stored if at all it is stored. I go to that line and ask away, is this address stored in your line? That means, I am query into memory locations concurrently and asking a question, is this tag bit, right? Is the tag bit, of say let us say 16 what are the 2 or 3 bits right. 16 is 1 1 0 0 0 let us see right. This is 16 right, right yes or no? So, this is stored in location 0. So, I will go and ask is 0 1 0 0 stored as a tag in either this 0, in this location or in this location.

What do you mean by store? The valid bit should be one and the tag should be 0 1 0 0 I am asking this question. So, in the past you have seen several memories where do you asked such type of question, if I ask such type of a question what is that memory called?

Student: (Refer Time: 07:35).

Student: Content address.

Content address memory or associative memory. Now each line here should be implemented as an associative memory. So, I index into that line like a normal memory. Because I take these 2 0 bits and say I have go and look into this location. The last 2 bits in this case point to that location. So, it is like a normal memory tag. After I get that line in that line there are several sets the several versions of this memory. So, I go and ask these fellows, hey is my tag stored? There is the valid bit one and my tag is stored. So, there what I am asking I am asking, whether content I am giving the content and asking whether that content is stored there.

So, horizontally this memory is implemented as a contest addressable memory, vertically it is a normal memory. So, each of these is called a way right. So, this is 2 way and since it is an associative memory, it is not a fully associative memory, it is associative on each of these we call it sets. So, this is get 2 way set associative cache. Is that 2 way set associative cache, modern processors will have something like 4 way set associative cache, 4 way set. So, I will have 4 such versions of this memory.

So, a particular address can either be stored here or here or here or here right. Any one of these caches it can store right. If all the suppose I have 2 way set associative 0 is occupied here 0 sits here and say sorry, say some 8 sits here and say some 16 sits here. Now I want access 4 then within this cache line, I have to do some replacement algorithm. It can be least recently used I told you right. LRU least recently used or such algorithms. I will cover one such algorithm in the class as an implementation, lot more details can be taken up in a advanced architecture course. But at least here I will cover how do we implement in hardware one such class replacement is there, I will take the most recent one followed by model Intel and AMD processes and I will teach you right.

Never class today. So, 8 and 16 are stored in that line now 0 has to stay in one of these fellows. Instead of just to one now I am made it two. So now, between 8 and 16 I have to

decide whom to move right. So, the hardware has to make now this is no software basis this is not operating system. The hardware has to give make a choice between whom to move should I move 0 or should I move out 8 or should I move out 16. So, there are lot of space by which hardware can decide you have that is called a cache replacement policy. So, the hardware has a policy based on which decides whether I move 8 out or 16 out to accommodate say 4 right.

So, this is notion of a 4 way set associative cache, and there why I have a 4 way sets so, all the modern processors to the best of our knowledge, they will no revile their architecture. So, Intel latest what is the cache? We do not know, they do not revile it. But intuitively we see that it is basically a 4 way set associative cache, is not more than 4 way. Now what can drive your intuition? Suppose you are given this problem of saying decide the number of ways in your set associative cache.

How will you come out with that? How will you answer this question? To be more specific 5 way will not yield any benefit, it may yield some many minus scale benefit, beyond 4 way the incremental benefit is very less compared to the amount of power you have going to consume please note if I have set associative if I have a content addressable memory it takes lot of hardware at consumes lot of power. Now it is not just one content addressable memory, for every line I have one content addressable memory, are you bale to follow? Right, yes or no? Right, So I use these 2 last 2 bits to index then I do a complete content based search on that line. And So, I have a content based content addressable memory on every horizontal line. So, essentially the power consumption is going to be much, much higher. So now, how do you decide what would intuitively, how will you try and answer this question? Yes.

Student: (Refer Time: 13:09) say you said.

Give me answer to my question.

Student: (Refer Time: 13:12).

You are actually giving an answer to my question. I am asking you how many ways how do you decide on the number of ways, yes it actually depends upon the cache usage pattern, correct?

Student: (Refer Time: 13:24)

Yeah we can try. So, we have something called architecture simulators, to which you run the program and it will tell you all the traces. Like, which memory location is accessed, that it. Now you can take just the memory traces. I do not really bother about when I am looking at memory management, I do not bother about what application is running right. It may be some application to by a buffalo, or it can be some hard and ethical harking thing. I do not really calk we can do a scientific computation. As far as I am concerned I have the memory traces. Now I take these memory traces and put it into my simulator and find out how if this is the cache size what would be the cache if it is the cache size what would be the cache size. So, I can make that type of lot of things I will come to that memory simulation also I will try and cover towards the end.

So now good the first step is, I have an infrastructure to find out for the programs that are interesting to me. I making an architecture for some domain right. For all the programs that are interesting to the domain. I know the memory traces. Memory traces means, what are all the memory locations it will access both instructions and data that I have the trace. Now given this how do you count? Why do you think 4 way is still sufficient even for? So, what will happen if I increase this? This is very short. So, what will happen if I increase this? Come on tell me Kavya, what will happen if I make this more? Suppose I make this 8 right. Then this will become 3 bits right. And this will become 3 bits right. So, what will happen 0 the next fellow will be this will go off, 0 next fellow will be 8, next fellow would be 16, next fellow will be 24, correct? So, the distance between these addresses increase; that means what?

Student: (Refer Time: 15:43)

Yeah so, for reasonable size program, normally you run program everything would not be 4 GB hundred GB right. Everything would be small in size. So, for reasonably you know large programs still, the address spread will be so much that by the time you are all the 4 ways get full right. All the 4 ways get full, there will be no conflicts right. So, if I am looking at say, a say some 512 MB cache, suppose I am looking at 512 cache for example, I 3 cache suppose I able to get 512 MB cache and it is 4 way also correct. So, 2 GB of cache itself, I have on chip suppose I have. Now what will happen by the time that 4 ways get filled up? Your program would end up meaning your size of the program will

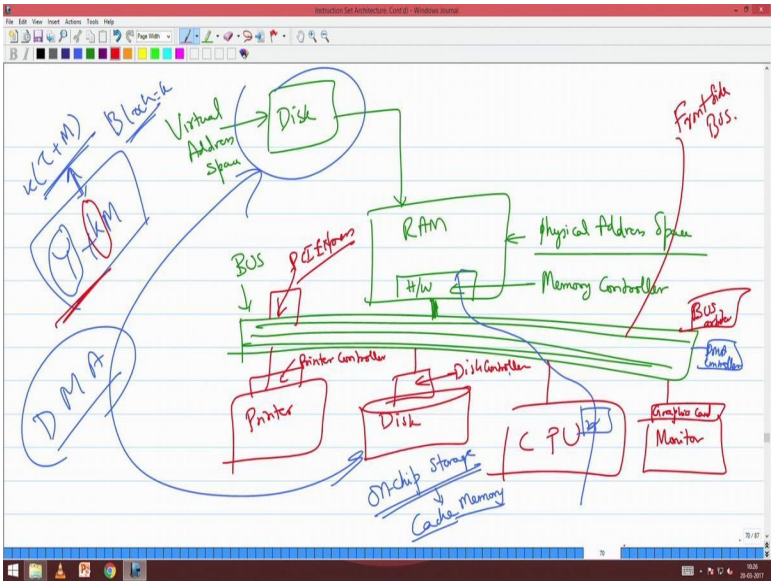
not be as large as filling up the all 4 ways of every line, are you able to get this, right? So, that is a very, very important thing. So, so what would happen is first time when either fetching the instruction or fetching the data it will be a miss, because it is there in the memory. It cannot suddenly it is avatar cannot come in the cache. First time I am fetching from memory to the you know, the to the to the processor it has to be a miss. So, this is called a compulsory miss. So, if you go to any processor company they will ask, what are the different types of cache misses? First thing you say compulsory miss. It has to happen, first time I am accessing any instruction, first time I am accessing any data that miss has to be there. So, that is called compulsory.

After that so, I brought in 8 I brought in 16 now 0 came. So, 8 moved out. Now I have to again bring 8 right. So, because it has moved out once, it has come in then again it as moved out this type of a miss is called conflict miss. This miss happened though I had brought it once, this hit happened because there was a conflict. It was thorn out and again I have to bring it. So, there are 2 types of cache misses, one is called a compulsory miss another is called a conflict miss right? Now what will happen is, as I keep increasing my size I make this stall and I make this fat right.

Then what happens right. My compulsory misses are always going to be there, but my conflict miss will reduce right. Because the spread the distance between the addresses are the distance between the least address on the maximum address now become now if I put a 4 way cache here, then the distance actually become 0 to 24. And I just double this size. While it was just 4 and only 2 way cache then I had the distance was just 4 or distance was just 8.

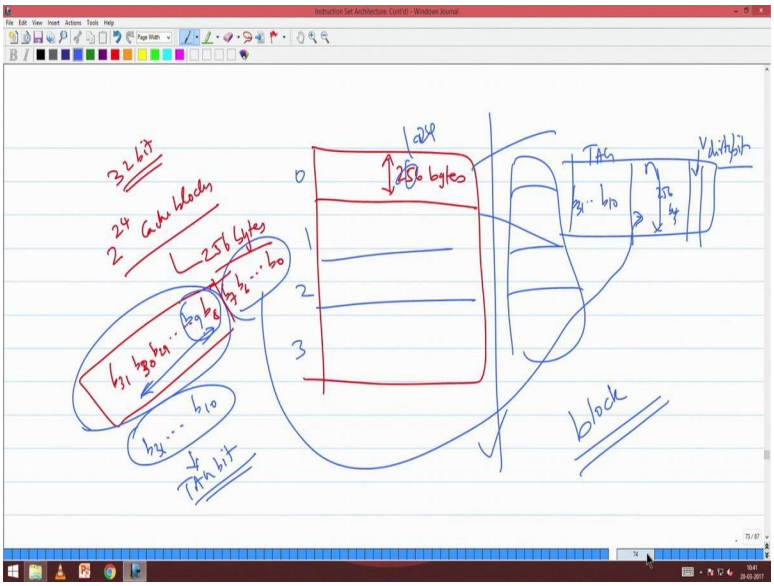
Now, if something above 24 comes, then only this 0 8 16 and 24 one of them has to get replaced right. And if this is too large 8 is a very small size or minus (Refer Time: 19:21) minor very minus size. So, this is quite large, then the number of conflict misses can essentially get. So, one of the questions is, that what is it that you have trying to optimize? I am trying to minimize the number of conflict misses. So, we are trying to optimize the number of conflict misses. Now can you give you an algorithm? Let us just take one way cache. Let us just take a one way cache. Can you give me an algorithm? Which will be extremely benefited by the before just going into it, now what we said was we are bringing 1 byte right. 0 means I am bringing 0 4, I am bringing 4 etcetera, but what we discussed in the previous class was I would love to bring k bytes at a time right.

(Refer Slide Time: 20:24)




And that is beneficial for me, because I have the principle of locality. I have principle of port special locality and temporal locality. Instead of bringing 1 byte I can bring k byte, by this what I am trying to hide, I am trying to hide the latency tau that we are seeing on the side. And I am reducing this tau right. Otherwise tau gets multiplied by k. So, what will happen here it is almost like paging here.

(Refer Slide Time: 21:13)



So, instead of so, each line will not be 1 byte it will be one block. So, one block can be any let us for us sake of argument, assume that this is going to be 256 bytes right. So, if I

am bringing one location I do not bring that location alone. I bring that block which contains that location. So, what is 256 bytes? What is 256, 2 power 8. So, if I take a 32 bit address space, how many cache blocks it will have? It will have 2 power 24 cache blocks, correct? Each of 256 bytes.

So, if I if I generate an address say b 31, b 30, b 20, 9 to b 8, b 7, b 6, to b naught, this will this whole thing will give me the cache block, which block it belongs to right. And right so, it can tell me which block it belongs right. These are these are not necessary for me. So, and within this I will now have a so, let us say I have 4 fellows here. So, 0 1 2 right. So, this first with this 24 bytes will tell me which block it is going to belong. Now b 8 and b 9 will tell me between the 0 1 2 3 where it will fit. So, I will go and what will be my tag bit? B 31 to b 10 with b my tag bit right

So, if I just blue moon of this zoom one of this, tag bit it will be b 31 to b 10 will be tag. And this will totally store 256 bytes, and will have one single valid bit and one single dirty bits that. So, when you generate a since there are when generate a 32 bit address the first 8 bytes 8 bits we are not bothered. This reaming is because each block is 256 bytes. Now I am looking at the remaining b 31 to b 10 b 31 to b 8 , within this b 31 to b 8 there are 4 locations in this cache. So, I take the first 2 bits to find out which location with this if at all this block is to be stored, which location will it be store. So, I take that first 2 bits, in those 2 bits then I have b 10 to b 31. I go to that location and check if the tag corresponds to this and the valid bit is one.

So now I will get a 256 bytes block. To this I input this 8 bit last 8 bits into this 256 bytes to find the exact location, to access the exact location in that dimension right.

Student: Sir what is the valid bit in the sense?

Valid bit says, you will see otherwise anything is valid no everything is 0 and ones everything is valid.

Student: Valid for the first addresses.

It is valid for the entire block, in this case the valid bit and dirty bits for the entire block. So, even if you go and change one location in that block and when it has to be replaced entire thing as to be copied

Student: (Refer Time: 25:22) say we want to access some 10th instruction in this 256.

You will would not the cache will only see 256 bits. If you go and change in that the anything the whole thing is, I told you right? If thousand liters of milk one drop of cropper position you will call It only poison, you go and say this entire 256 bytes you can change one bit gone you go. And even write see it is already stored in see just fun you write 2 again, still that fellow is says is dirty right. The moment you write into that location the write access come to the left location it is becomes dirty.

So, when I replace everything has to be written back. So, let me just repeat once more So that you know you are very clear. We were dealing with 1 byte now we are dealing with 256 bytes. Why I am dealing with 256 bytes? Because there is a latency hiding I am going to do, correct? I am going to hide latency and it is very beneficial for me as of now right. Because I have the notions of temporal and spatial locality, that is given empirically proved on the programs ok.

There is another thing that I am going to talk of little later, where without these type of block caches, the whole operating system will become defunct meaning, it becomes highly in affective. I will tell about that later. But at least these 2 advantages we should, for that reason what do we do we if I want to bring one address it is not there in the cache I will bring that 256 bytes to which it is associated. What do you mean by the association? You have 32 bit address the first 24 bits from the most significant bit will tell me which block it is. So, from that suppose I take this is a 4 location cache I take the last 2 bits b 8 and b 9 index into which location here, and the remaining things b 10 to b 31 I store it as a back. Please note that for all the 256 addresses, this b 10 to b 31 will be the tag, b 8 and b 9 will be the index, correct? For all the 256 bytes, because these 24 bytes are fixed for those 256 bytes correct, are you getting this? So, so the tag will be b 10 to b 31 and your b 8 and b 9 will tell you what index and then there is.

Now, this is a one way direct mapped cache. Now if I make it 4 ways same thing I will do. So, I can store it So, I can add one more way here and the same thing. So, so I now I am query again that that 20, 20, 22 bits right. Totally 8 bits goes off here, 2 bits go off here ten. So, 22 bits I query on each of these location for the tag. I ask whether you are valid and this tag there. You will say if it is yes, then I use the last 8 bits to index into that location and basically. So, this is so, this is how yeah, k way set associative block cache

floor. So, this is called a block cache rather than a byte cache, because every time I unit of transfer, what this block? This cache looks at this minimum 256 bytes. It cannot resolve individual bytes in terms of so, whenever I write into some location essentially I am going and dirtying that block. And so, if that location has to be replaced then the entire thing has to be returned and then fresh thing has to come ok.

Now, how do you fix this? I am coming to question. So, how do you fix these 256 bytes? If you fix this 256 bytes based on my performance requirement; if there is a cache miss, if I say I will put 1 MB block right. Then I will be spending bringing 1 MB of data here, your temporal locality again know anything more than threshold nectar also becomes poisons. Like that if you put 1 MB right. So, you will bring all these things special locality will not exist for 1 MB. So, I cannot be. So, far of this 1 MB I cannot remember right. So, I will not write a program with spatial temporal locality of 1 MB, maximum I will look at say some 10 instructions away or 20 instructions, 256 k 256 bytes to So, or 512 bytes would be something much conceivable. Or even say they can go up to k's, 1 k or 2 k, but I cannot say 1 MB at all So that much memory the programmer does not have in terms of visualizing a program.

So, 1 MB cache block will not work. So, again this is all experimentations. So then so, what basically happens is there is an architecture simulator. In that simulator you run all your programs and find out. So, what are all the parameters no how many ways I could have? What is the size of a block? These 2 are parameters we have seen. And what is the height of the memory. So, how long should it be? How fat should I be? And you know, what is the block size? So, 3 parameters already we have.

So, these 3 parameters we want juggling here and there and then come up with some optimal choice. And finally, say if I put some 4, 5, 10 ways and all see then your area also should your power also chosen. So, there is an implementation aspect of it. And also when I have a large set associative memory my look up also takes more time. So, a performance also gets you know, when I make it more and more fat, your performance becomes lower. Because now it is a set associative memory, so I have to my performance essentially takes a task right. So now, you understood cache modeling right. So, I have to look at area, I have to look at power, I have to look at performance all from the circuit side. And again from an application side I should look at what would be the block size, what would be the number of ways what should be the height of the memory 3

parameters So that I get best performance. So, between these 2 I have to strike a balance.

So, already we have 6 parameters and each would have some 4 5 variance.

So now imagine hundreds and thousands of paper we can generate just by looking at these of this policy that policy response that is that is the entire story about cache. Now I will take one or 2 stories and tell you later. Now I am giving you this let us just say one way each is 256 bytes or whatever 512 bytes or one kb 1024 byte. Can you tell me a program that will kill this? And can you tell me a program that will boost this? If at all I want to evaluate the performance. So, I have put 1024 bytes one way, if I want to evaluate the performance of the cache what sort of things will I start doing. What would be that person that program that can evaluate this? Loop, plus loop is the one because I am looking at locality. Do and then loop what would be what will the loop do?

Student: It depends on where exactly

Do not it depends on where that I know. Now I am asking you what is that thing which will make it depend on what you are say, are you get my question?

Student: This is the location of (Refer Time: 33:37).

How will you, how will you, loop is excellent Kavya told loop, now you say next.

Student: Function call inside the loop.

Function call inside loop will, function see I want this to be killed. I am killing essential cache trust and so. So, it should be something huge.

Student: (Refer Time: 33:59) I have an integer array.

Array, why integer?

Student: (Refer Time: 34:08).

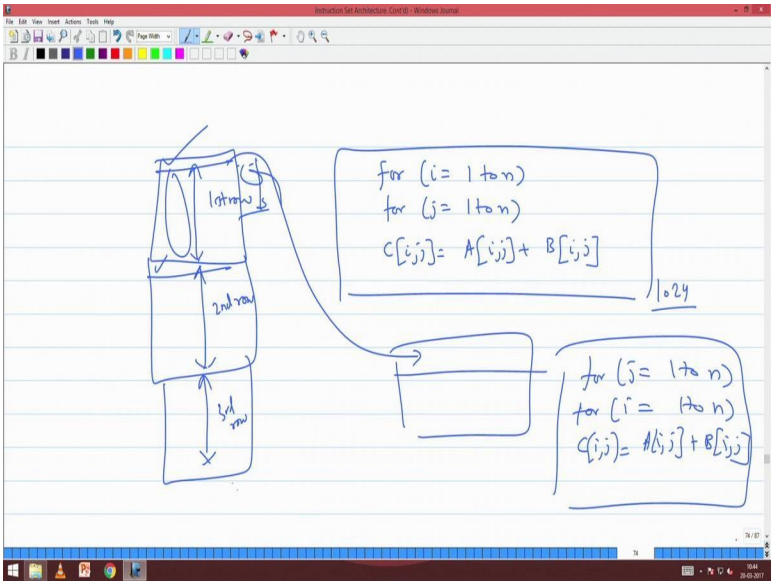
Floating point or anything, array. Just array is enough. Array is a linear structure. Now you saying you should kill, it should enhance. So, what is a non-linear version of an array?

Student: (Refer Time: 34:24).

Version of an array, linear one dimensional matrix corrects? Are you able to get this? So, I should have loop acting on a matrix. Can you give me some example matrix addition?

Student: Matrix addition.

(Refer Slide Time: 34:56)




Very good man, so, matrix addition what will happen? Suppose guys carefully look into this. This is what we do in matrix addition right? Now can you guess, now how is a matrix stored in the memory?

Student: (Refer Time: 35:31)

Row major order that is one way call a major order, so what will happen? So, this is first row, then second row, then third row. Like that in the memory it will be stored. So, here first row first element this taken. So, I am reading it row wise. So, when I fetch the first element 1024 elements should be fetched. So, thousand so, first element I fetch it is a cache miss, then I will have 1023 hits. Because along with the first elements 1024 elements will be fetched correct? Along with the first elements, so, into the cache along with this first element 1024 elements will be fetched to the block cache. So, for every 1024 access there will be only one cache miss, the reaming 1023 will be cache hit, and similarly for your B matrix and similarly for your C matrix.

Suppose I wrote it as for j equal to 1 to n for i equal to 1 to n, C i j is equal to A i j plus B i

j. That is all both are equivalent programs correct? Both will now what will happen? I

will go and bring the first element, then that entire row will be wrong, but I need the second column I am access, in this way what will happen? Your matrix is accessed column wise. First element I will bring, I will bring 1024 things inside. But none of them will be used, I want something else, assuming this matrix is going to be a 10 thousand byte, thousand matrix, which you see normally if you do other modeling will see 10 thousand (Refer Time: 37:43)

First element I will bring 1024 fellows I am brought in, but no use, it is a miss and that 1023 data is also no use, because the next fellow is going to be in the next column. So, this is the first element, this is the second element none of this I am going to access you are getting this. By the time I come back and come to this second element for all my luck this would have gone up, where a huge matrix that I am trying to access. So, something else should have come in read it (Refer Time: 38:18).

So, what will happen is for every, I can I can find tune the size of the matrix in such a way that, for every access I am going to get a this. While here for there is only one occur every one miss I have 1023 hits. The same program, same functionality it depends on who it depends on the compiler, whether it is going to affect the column wise column major storage or a row major storage. If we will does a column major storage and you access, if you do a column major storage and access it row wise, or if you do a row major storage and access it column wise, then I am going to have a big soon correct. So, this is what we mean by a compiler optimization. In fact, of compiler on the architecture where does the complier and the architecture talk with each other. This is one more very interesting example, are you able to follow?

So, we have talked about static instruction scheduling, dynamic instruction scheduling is that compiler can be dirty complier has to be extremely intelligent right. In a static instruction scheduling complier has to be intelligent. In dynamic instruction scheduling complier can be intelligent right. Now here complier also depends upon the architecture. So, if you ask a question why in the processor company like IBM Intel etcetera, why there is the complier team? What are their works? Of course, they have to write complier which will translate.

But more than translation there is something called architecture optimization. There are lot of architecture optimization topics which you will learn in the complier course. But

one such team because we are talking about memory management here is that the complier should know, should allocate and drive the code like this right; should allocate the memory in the same way as it driving. If arrays the compiler by chance is storing the arrays as column wise, then when it is compiling these 2 it has to swap this j and i you see that there is a (Refer Time: 40:34) are you able to understand?

So, how the complier and the architecture, if they fight with each other, if they do not know about each other then we may enter into task right, it may take he has to execute this, that if I know each other and I want to work by, by then what happens then it can do a good job. So, this is one very interesting and important example about the interaction of a compiler with a architecture. We will take it forward see what I want is please revise these things. Because this cache has So many things, tomorrow I will talk about inclusive exclusive cache and slowly will make this topic, another 2 3 days you need to have the continuity please understand what I have thought now.

Thanks.