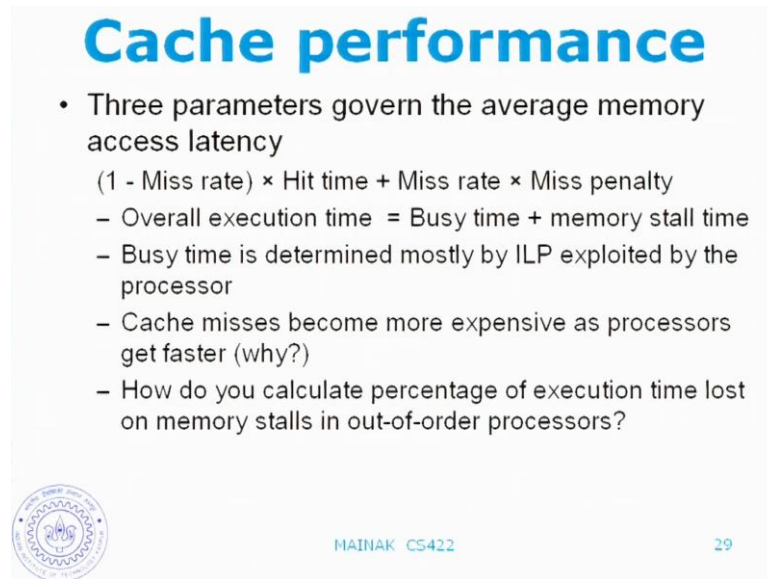


Computer Architecture
Prof. Mainak Chaudhuri.
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur


Lecture - 28
Topics in memory system, DRAM and SRAM Technology

(Refer Slide Time: 00:13)



Cache performance

- Three parameters govern the average memory access latency
 - (1 - Miss rate) × Hit time + Miss rate × Miss penalty
 - Overall execution time = Busy time + memory stall time
 - Busy time is determined mostly by ILP exploited by the processor
 - Cache misses become more expensive as processors get faster (why?)
 - How do you calculate percentage of execution time lost on memory stalls in out-of-order processors?


 MAINAK CS422 29

We discussed 3 parameters. So, today what a do is I will over some these 3 topics memory systems before we want to a discussion of ...

(Refer Slide Time: 00:31)

Exclusive hierarchy

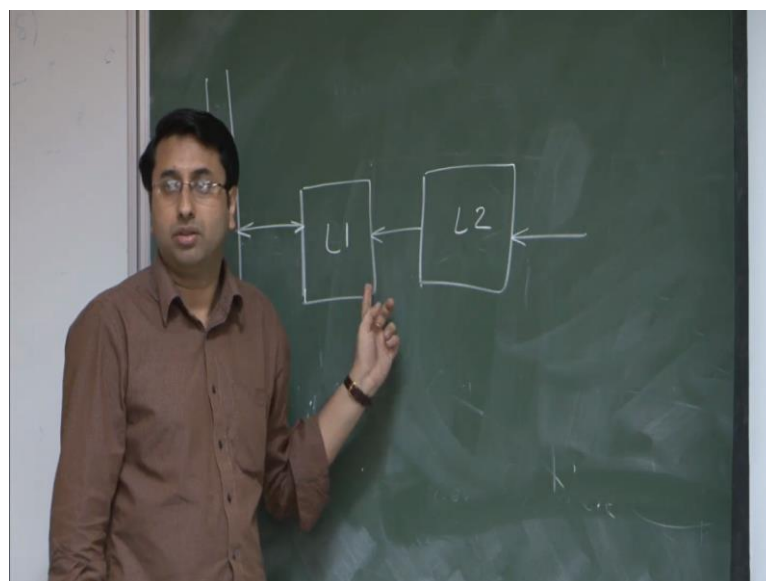
- Drawbacks of inclusion
 - Back-invalidations and replication across levels of hierarchy
- Trade-off between inclusion and exclusion
 - Effective capacity vs. bandwidth
- Bandwidth-efficient exclusion
 - Bypass potential dead blocks
- Middle-ground
 - Non-inclusive/non-exclusive



MAINAK CS422 37

So, this I discuss a last time about an exclusive hierarchy talking about there is some more keep which exclusive and at the reason for these was that conclusion has a couple of programs right. So, what is that there is replication across level, there is 2 level of that L1 is a subset of L2 which means necessarily there will be wastage of spaces.

(Refer Slide Time: 01:21)



The second problem is that where you add a block for the other level cache where add a block for here we have to inherit the block L1 also and otherwise L1 may not be a subset of L2. Now, it is a little problematic because you are essentially taking away a block from L1 cache simply because you are adding something to L2 cache, but if you do a very well happen that the block in the L1 cache is heavily used by the processor. So, if you examine more carefully what is the cache block which is very heavily used by the processor.

So, what will happen the processor pipeline is here and is essentially access the block for L1 if you have several who is the L2 will not get to know, because you do not get the access a block in L1. So, what will happen? So, what may happen is that these block may eventually become the block in the L2 side because there is no access L2 cache which may become the ... And eventually the L2 cache may actually block because it a block and when it is block to maintain it will also inherit the block for L1 which may actually hard perform is because actually this is a very hard block doing access by the processor. So, these are called back invalidations; invalidations triggered by inclusion

So, what will happen by such a block immediately the processor will actually will access the block have a caches and beat them block that we. And this cycle it will not repeat again this will access the block for involve and you will see the is the code block to become where are you eventually as the again a. So, ultimately what will happen is an because of his the back evaluation you will see extra cache which you would not have if you head about in future. So, these are called back in terminations of or sub because of just because of your before using extra cache machines? Is that clear?

If you have a exclusive hierarchy this problem both are both of them problems it is actually you do not have a duplication across happens. And you do not have to worry about this back to evaluation whenever you able to cross level 2 look at this because there is no guaranty about the subset always these are exclusive in the disjoint. So, wherever the block is L2 it is not a L1 you know that, so to block from L2 you can just with alphabetic. So, what is the state of between inclusion and exclusion this 1 also we discussed last time that exclusion lies in capacity because, but it request for bandwidth. Because whenever a block is evicted from L1 it has to allocated to the L2. So, it has to

travels the the pass whatever there is between L1 to L2 and it will go to L2 on every every which is not presently. Now, it is possible to make exclusive caches magnet efficient essentially the the idea is that when you edit a block from L1 you put a pattern pattern here which you audited.


The blocks that to get L1 and basically figure out which blocks are not not likely to be reviews in future and those blacks will not actually go to L2 they will just be dropped, because there is a point keeping this blocks in the caches. So, I am not going to the detail of this particular sitting here you can essentially the idea is that you will monitor the access inside of this block and makes of that you decide when did the block is or may be live. So, there is a middle ground that is offering processors that is called non inclusive to nonexclusive what is that essentially the idea is that your L1 is not a L2, but it is not also this 2. So, what happens in this case is that when you bring a block from memory you fill it to L2 fill it to L1 all right, so but for you ethic something for to L2 you do not for L1.

So, essentially what happen doing is that I am taking away this part of this process is back in relations I still have a where the blocks are duplicate across the also the hierarchy. So, when you edit a block L2 you do not this is where the subset property gets violated. So, essentially what you get this either exclusive or nonexclusive.

(Refer Slide Time: 06:26)

Trace cache

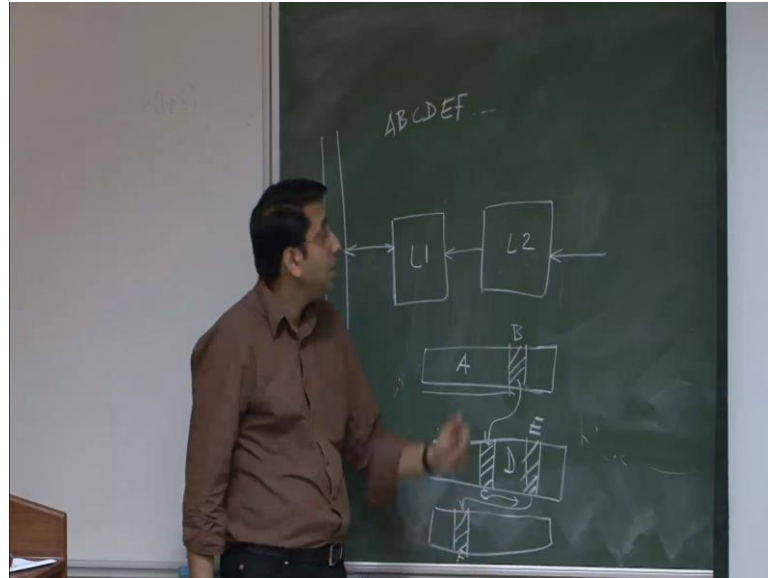
- Need to supply “good” instructions quickly
 - Often predicted target of a branch falls in a different cache block and may be in the middle of the block
 - Wastes trailing part of the cache block having the branch
 - Wastes header part of the target cache block
 - Worst part: cannot fetch from the target in the same cycle even if the prediction is correct (multi-ported icache is very hard to design at high frequency)
 - *Build traces dynamically*: Branch prediction validation becomes part of the hit test
 - Introduced in Pentium 4: it does not have an L1 instruction cache, only has a trace cache
 - While building the trace lines, Pentium 4 also translates IA32 instructions to micro-ops (off the critical path)
 - Downside of trace cache: may contain duplicate partial traces (why?)



MAINAK CS422 38

So, as I said Essentially goes to some a distinct topic today the second point is trace cache this is use in details processors because it was including pentium 4. Essentially the idea was that we need to supply the good instructions quickly to the pipeline if you by good instructions I mean you most almost always you want to remain of the current processors actually correct transportations are you may in the current process, because if you are know at essentially processing instructions we should be eventually to wasting time. So, off course what will to achieve these to have small project another to have trace cache. So, what is this? So, often predict a target of a branch also the differ cache and may be in the middle of a branch. So, these are we also talked about last time.

(Refer Slide Time: 07:37)



So, essentially we were currently executing an instruction in this cache block and let us suppose branch instructions and the target of this instruction falls in the middle of some other cache right. So, you this instruction and the next instruction execute will be this one. So, clearly this part of the cache will be waste.

Waste of the trailing part of the cache block having the branch is, but it wastes the header part of the target cache this block and the worst part is that you cannot fetch from the target in the same cycle even if the prediction is correct. So, the same reason is that you usually single cache multiple cache of same cycle you can have multi ported instructions caches. So, talk about multi ported caches, but now on is it is very hard to design the high frequency because additional course usually slowdown reverse structures. So, what pentium 4 did was it bit faces dynamically? So, the branch branch prediction validation becomes part of the hit test. So, will it does that on the whenever you have a branch instructions like this 1 you find the target.

Alright and you prepared the trace of instructions were this particular instructions activate tight of this instructions. So, essentially now, you are preparing a trace cache block where these instructions activate tight of these instructions you are preparing this trace is on the fly as a you in count of these branches and targets. So, it has introduced in

pentium 4 which did not have L1 instruction cache the only has a trace cache. And while building the trace lines pentium 4 also translate I a 32 instructions to micro operations off the critical path. So, that it actually saves lot of reporting time. So, is the is the is the happenings. So, I am executing like this and the branch. So, then I jump here and then I again execute like this make it I encounter 1 more branch here then I jump the some other cache block is the target here.

So, what will happen is that the what will trace what do the trace have. So, let us let us smart this one's this my this is my portion is A this is B this instructions is C this portion is D this E this is F and so on. So, traceful have A B C D E F etcetera. So, the trace will not have this part and this trace will be prepared on the block and these will basically become a trace cash block stored in. So, next time whenever you encounter A you will a get out whole cash block out, but how an our question is next time when I encounter B will he go to see again that may not be the case actually.

It may go to some other place by may be actually, so that pass pass correct. So, that is the point here trans prediction evaluation becomes part of the hit test. So, the trace cache law not only takes the of the, but it also checks the branch predictions that the that the incase decide trace. They will also have to match otherwise the trace time is the rejected and it will actually call back to the traditional execution here.

Now, 1 major problem trace cache was that it may contain duplicate partial traces what is that if you contain duplicate partial traces. So, same piece of course, may be part of 2 different part right we can the execute to this path to you can execute this path and you can have a common portion of the of both of these traces right. They may happier as 2 different trace likes in 2 different portion of trace caches.

So, this is really hard to abort. So, it is really hard the above at to duplication portion anyway. So, I will I will keep it to that only if you really want to dell preparing to trace cache let me know I can give you a paper which talks about it as if you want to know more about exact architecture that became 4 space cache let me know.

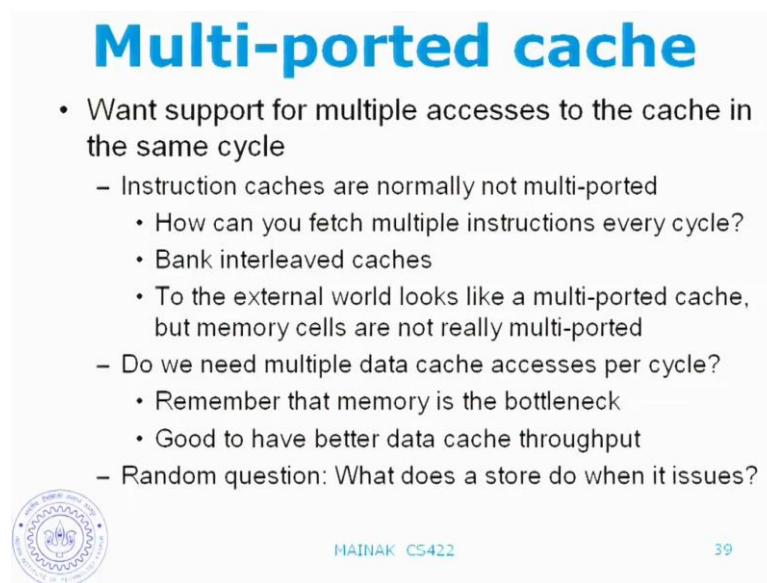
Sir. So, trace cache.

Yes.

That is right. So, interesting the pentium 4 did not have an 1 instructions it would has a 2 caches.


Sir, trace have to be.

(Refer Slide Time: 13:07)



Multi-ported cache

- Want support for multiple accesses to the cache in the same cycle
 - Instruction caches are normally not multi-ported
 - How can you fetch multiple instructions every cycle?
 - Bank interleaved caches
 - To the external world looks like a multi-ported cache, but memory cells are not really multi-ported
 - Do we need multiple data cache accesses per cycle?
 - Remember that memory is the bottleneck
 - Good to have better data cache throughput
 - Random question: What does a store do when it issues?

 MAINAK CS422 39

Any other question all right we discuss little bit about multi ported caches. So, let me try some offer some more detail here. So, what is a multi ported cache? Essentially the idea is a very simple you want to support multiple accesses to the cache the same cycle to essentially want to have multiple parallel access to the cache. How do you do that? You need 1 4 per access that is does the whole point. So, that I can I can access if I would both report cache and may be able to access 2 different caches in the same cycle alright. So, instructions caches are normally not multi ported. So, then the question is how can I fetch multiple instructions every cycle, because we say is that this is this is 1 important aspect as the super scanner processor. So, what simple solution is that you have multiple instructions in a cache? So, you can just bring 1 cache.

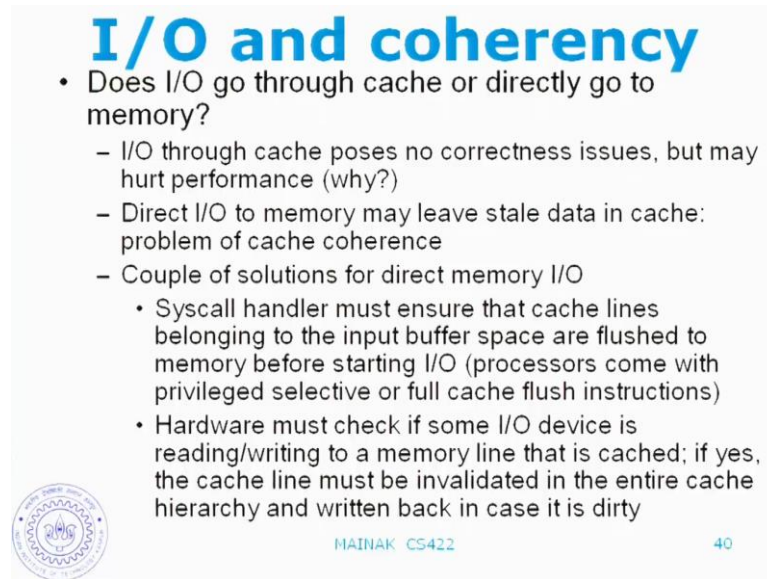
Hopefully you will have to the many instructions then the cache right if you really want

to have the effect of multi ported cache at least is to some extent you can have badly cache. For example: if we have 2 two separate bands of the instructions cache you can access both the bands in parallel and you can be each band single ported ignore 1. So, that will still keep you some benefit multiport gives you the full freedom because for example, if you find that you want you access 2 caches on falling on the same band that you not get out we should be allow in the true rural ported cache. So, that is what you lose actually. So, 2 external world practically cache world as looks like multi ported cache, but memory cells are not actually not multi ported. So, they are just single ported cache.

So, you can get some around of, but not poor. So, that is about instruction cache the question is now do we need multiple data cache accesses per cycle enhancer is normally yes because memory is the bottleneck. So, it always helps to have multiple component access to solute the data cache. So, it is good to have better data cache to put. So, normally you find that data caches are duel ported at most duel ported normally is an not more than 2 port these all we have already discussed what are the 2 orientations. So, we might want to remain that is why this question is here that is about multi ported caches. So, keep it mind instructions caches are normally not multi ported will be receive this question again while discussing that.


So, there will see another motivation of having multi ported instruction cache. So, see how to that data caches are normally multi ported, but number of ports is not.

(Refer Slide Time: 16:18)



I/O and coherency

- Does I/O go through cache or directly go to memory?
 - I/O through cache poses no correctness issues, but may hurt performance (why?)
 - Direct I/O to memory may leave stale data in cache: problem of cache coherence
 - Couple of solutions for direct memory I/O
 - Syscall handler must ensure that cache lines belonging to the input buffer space are flushed to memory before starting I/O (processors come with privileged selective or full cache flush instructions)
 - Hardware must check if some I/O device is reading/writing to a memory line that is cached; if yes, the cache line must be invalidated in the entire cache hierarchy and written back in case it is dirty

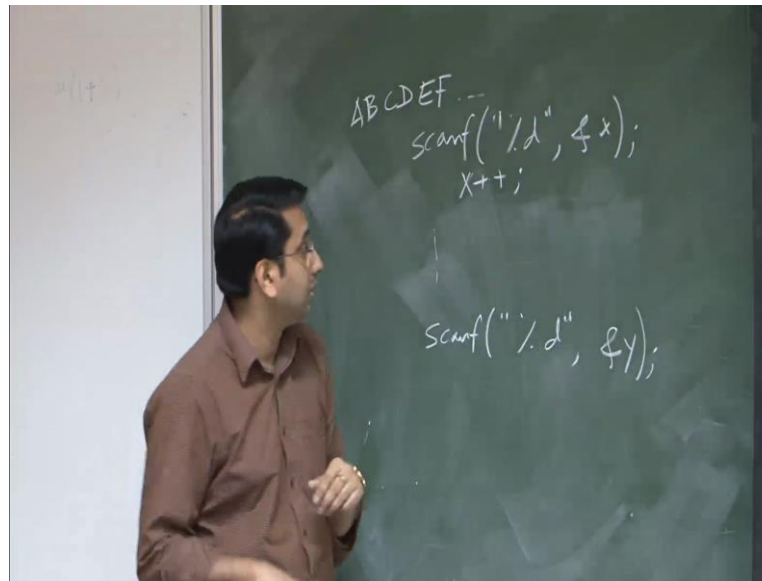


MAINAK CS422 40

So, a little bit about input the output. So, am just touch upon the few changes that interact with the memory hierarchy we have not discuss I o, but I assume that all of you have seen at least some of Io input output. So, essentially the point here is that the device that the computer task which essentially input output devices like keyboard we were display device CD ROM drives USB drives speakers etcetera right. So, the question now is that question trying to ask here is as follow.

So, let me keep you an example, suppose am reading in some in keyboard so the user is happening anything. So, how does it work usually, but the data that that is been will be store in the keyboard controller small buffer and when the buffer gets pull it will send in the interrupt in the CPU. So, the CPU then pick up the data copies the data from the keyboards internal buffer in to the memory right. So, the question now is that it copies in the memory or to the cache where does the final memory where does the data go? So, once the data can copied from the CPU can it can parts an data it can interpret the data is noted wants program ones.

(Refer Slide Time: 17:57)



For example please say scan if percent d and percent x right. So, what does that what ultimately it the put the value at that particular address whatever is specified there alright. The questions is this address is cacheable or not does it gone to the cache or does it not because the next what I may want to do is I may say that x plus right. That is the CPU instructions and x will be copied into cache and normally proper do not right. So, this particular buffer, so essentially there are 2 copies are going on at the first path will involve a system call which the operating system will copy the data from your keyboard in to condone space buffer alright. And then, the condone space buffer will be copied to in this particular address at the content buffer the question is where is the current space buffer is it inside the cache or not.

What you think either this address is the off course in the space this will be the cacheable? Now, able to reacts where is the current space buffer is it? What does the intuition payment is it cash or not buffer used for doing I know?

Is cache.

Sorry, silence no x, x is the x is the percent x is usual space address.

Yeah.

So, it is of course cacheable I would be inside cache the question is. So, their is intermediate copy they just going on right copy the data copy the data from keyboard buffer into a space buffer. That is the Io buffer and then the current space buffers content will be copied to that particular address and that complete scan. If the question is where is a buffer is it inside the cache or it is not in the cache?

It has to be cache.

It has to be cache, why?

45 mark and cache we can do it. Hardware system don not know but operating system knows, operating system may choose to make the Io buffer un cacheable. So, it generate a translation where the attribute makes you safe and is the un cacheable that possible what you do separately if I make it a cacheable is very take a long time exactly. So, here that talking about single variable, but we may take the bunch of inputs for example, we may for file.

So, then take a long time to copied from the condone space to the user space . So, now, space buffers are cacheable alright. So, now, that list to be a problem after a while I do suppose I do so am fine and happy I have great x the continue then I after while I say scan if percent d and y right. So, remember then the current space buffer is now cached alright. So, while and use the input it is going to be a cache right it is already in the cache.

The problem is at now when I read when I try to be y I may actually get the same value because x make it copy to y, but the current in buffer the copy that inside the cache. So, this called the problem. So, that is that is what is being summaries here.

Yeah.

Right ok.

No, but the stream contents to be copied in the internal normal space buffer right the processor.

Before where does the copy from the keyboard controller to the from the current space.

That is right.

That much have that much execute before the remote control.

That is right.

Yes.

That is right that is right yes right exactly. So, that is exactly what is being says is in Io is to cache poses no correctness issues, but may hard performance.

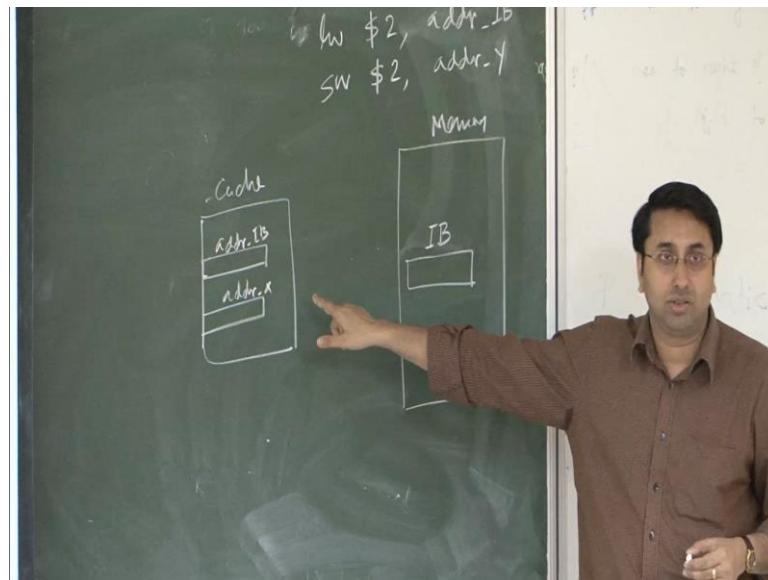
So, essentially the point is that when you put the buffering cache as he has already mention there will there will be on Io space there will be current space right to the path which is already in cache alright. So, when the processor in try to copy from that buffer it will get the correct contents and put the value in the address of y, but direct Io to memory may be its still data in cache.

So, here the point is that this suppose the buffer actually was in memory it was not cache, but processor has to copy x from the buffer. So, what does we do in the process it is actually cache the buffer because what will what will it what kind of instructions. It will say load some certain address from the buffer space and then store that value in to this particular x. So, in care of and the load will goes to the cache alright.

So, the point is that what we have done is that you have essentially copied the buffer in to the cache from memory and then going to do this Io suppose the Io have to his directly

to memory. So, will actually copied the keyboard buffer content directly to memory buffer all right will not the cache another processor again does the same process. It is start copying from the current space buffer to this address, but now cache alright.

(Refer Slide Time: 24:38)



So, let me show you doing the so this is my memory. So, this is the cache, so let us talk the scan so let us talk about the these about. So, what does the do? Copies the keyboard controllers buffer contents into memory that suppose this is my input buffer here imports the data buffer keyboard controller buffer, then the processors job. So, now, the system called hanger essentially has done its job it is from the keyboard in to the condone space buffer; the other processor has to copy the content of this buffer into that particular address. So, what does do it is generate base instructions to in the most important 1 to be. So, if you load from these address do some register. So, address underscore I column all right and then what you will do it will do store work in 2 address of x that completes the first all right.

So, what is the effect of this? Essentially by these to instructions I in to 2 things I made a copy of address Ip inside by the cache. The cache is containing that an would be the copy of address x inside the cache now the second buffers alright. So, what does the do it raise the keyboard controllers buffers copies the content of remember that am doing all right

as a system called hanger. Now, what do the processor has to do? Processor has to copy these contents into a percent y. So, it is starts a process away it does exactly the same thing except that this will now be y.

So, now, what would happen as soon as execute this instruction it will get a cachet whatever the is here would be loaded to dollar 2 which actually wrong. And then, it will execute the this store it is essentially y will get the value of x whatever value of the x the actual contents are actually still sitting here process does not see. So, it is a problem the Io you memory maybe it is same data in cache. And the other hand if the Io actually happen through the cache. Then this problem would arrive because a condone space to input buffer could actually the cacheable theory. I will always get the correct content from the cache I would not be make in 2 different topics.

Processer.

They can you can makes a Io device access a cache were actually solve this problem if you actually to do is eventually to do that. Because, if you say that well I do not allow my Io devices processer caches, but then a come up with solution for this. So, how do solve this here couple of solutions only Io. The system handover must ensure that cache belonging to input buffer spaces are cache to memory before starting the input. Essentially, what I am saying that these particular cache time should be fast from the cache before the input output operations. So, processors come with prevail edged selective or full cache flush instructions.

So, you can make of them to do is the second solution is that hardware must check if some Io device is reading or writing to a memory line that is currently cached. So, this all requires for extra hardware support that is whatever address that flows between the Io device. Then memory must be by processor cache and if that particular access currently cache it has to remove that cache you face. The cache time must be evaluated you may entire cache hierarchy and get back and cache.

So, this is a very attractive system to the operating system designer because it does not interactive in the processor cache the Io does not interfere caches. Because, there are

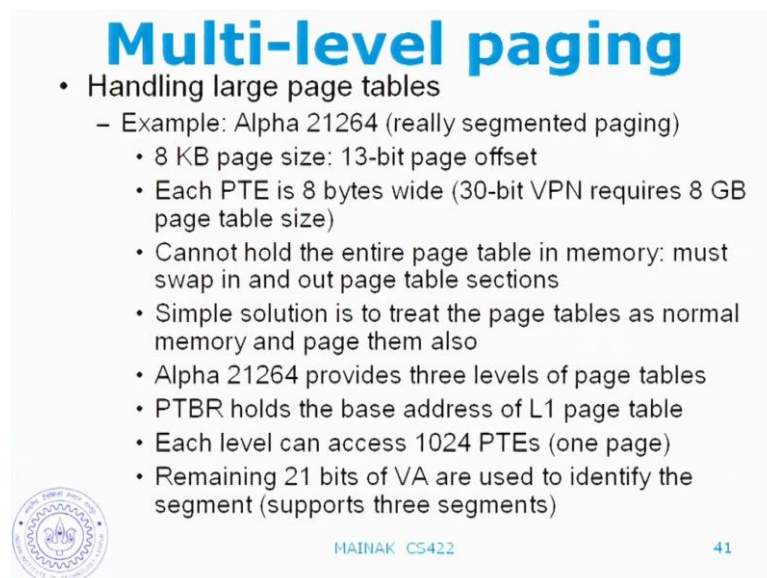
many pieces of data that the processor cache should not have many and that can be about it completely if you do to Io in memory the counts.

You have to do this same problems are I given the example of input here the same problem arrives the outputs also exact same problem. You may interpreting on the on display device something which you did not some more data. You may will pickup input output in more detail there again will come back to this problem with more example. So, any questions of this is this clear. So, here what here the take away the point is that even in a single processor accept you have to worry about the problem of cache which people often talk about in content of multi processors. But, single processor you have to support some minimal amount of cache queries to they able to do Io authorized because the Io device is can be seen as individual processors.

Essentially, now you have a make the processor you have certain other Io devices that can be depend manipulated data. So, do have some solution for progress questions required the it is.


You can. You can make to the way you want. So, depend the operating system exactly which data is cacheable usually that comes a part of ...

(Refer Slide Time: 31:14)



Multi-level paging

- Handling large page tables
 - Example: Alpha 21264 (really segmented paging)
 - 8 KB page size: 13-bit page offset
 - Each PTE is 8 bytes wide (30-bit VPN requires 8 GB page table size)
 - Cannot hold the entire page table in memory: must swap in and out page table sections
 - Simple solution is to treat the page tables as normal memory and page them also
 - Alpha 21264 provides three levels of page tables
 - PTBR holds the base address of L1 page table
 - Each level can access 1024 PTEs (one page)
 - Remaining 21 bits of VA are used to identify the segment (supports three segments)

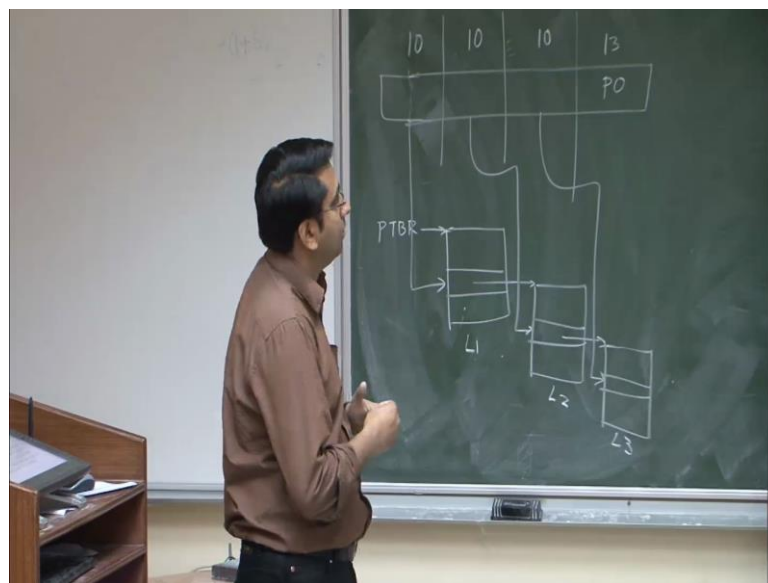


HAINAK CS422 41

So, I wanted to touch upon multi level paging how many documents seeing this process of operating system multi level paging. So, that is good. So, highest quickly go this 6 I discuss paging. So, there is a problem which handling large page. So, let us start to understand what are the problems is. So, in alpha 21264 for this example taken is directly from you can go back and make in more detail this is just a summary of that. So, alpha 21264 it will be segment paging you able to ignore that 1 we will assume that.

It has 8 kilobyte 8 size which means it is a bit page offset all right each. Each page entry is 8 bytes wide. So, it has a 32 bit page number. So, that gives you a 43 virtual address actually it is more an upper bits some other parts processors. So, a segment is a normally page in upper beats specified segment. So, anyways so all purpose what is the important is that you have thirteen bit virtual number all right, which 8 bit page level entries this needs to 8 gigabytes all right to 30 multiply. So, you can imagine just to support virtual memory you have to 8 gigabyte of memory space for make a sense. So, you cannot hold the entire page table in memory what you do is you must copy the swap and out each table sections. So, what? So, simple solution is to treat the page tables as memory and page them also. So, 21264 provides 3 levels of page tables page tables base register holds the base address of L1 page table each level can access 1024 page table entries.

(Refer Slide Time: 33:53)



So, these for like 13 bit o s page offset. So, that picture called history picture. So, each of these point to the levels of tables. So, the page tables base register called the base address of a L1 page table. So, each of the table can hold 1024 page table entries. So, essentially these can bits for 10 were in L1 that alright. So, what does these entry telling any guess and here it is not there what should hold what going to know.

Process.

The page table for the next.

The base address of the address table the base of L2 this 1 can see the offset in to it this 1 holds the base of L3 what does this hold contain.

Base address of the page.

Base address of the page.

Base address of the page the physical page number alright.

So, is it not the base right alright. So, that is that is how you access, so essentially all you need to do is 3 tables in a memory how may got they each 1 contents 1024 page table entries. Each page table entry how much?

8 bites.

So, how many are in this table?

Yeah

8 kilobyte

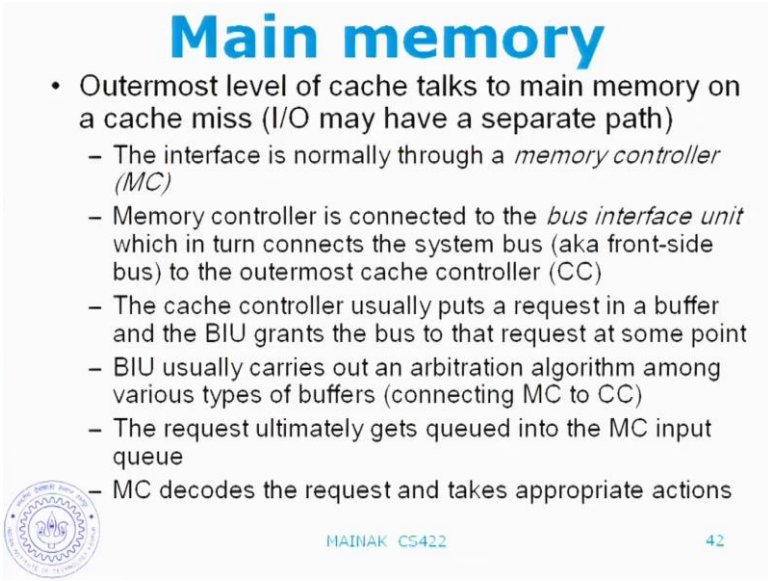
8 kilobyte

How much is that a page?

I whole 3 page that is it. So, usually the page tables are size is that they become a page. So, that is the page alright. So, that any point that that hold 1 page of each of history there was instead of holding 8 kilobyte of page telling just keep to 3 pages now alpha 264 is a 64 bit processor these only gives you 43 bits of the virtual access what about the upper bits. So, remaining 21 bits virtual useful are to identify the segment it support 3 segment.


There are many un used that is the 3 segments need to be identified is in 21 bits. So, we can read up example in your book. So, it has more details about 64 portion memory.

(Refer Slide Time: 37:23)



Main memory

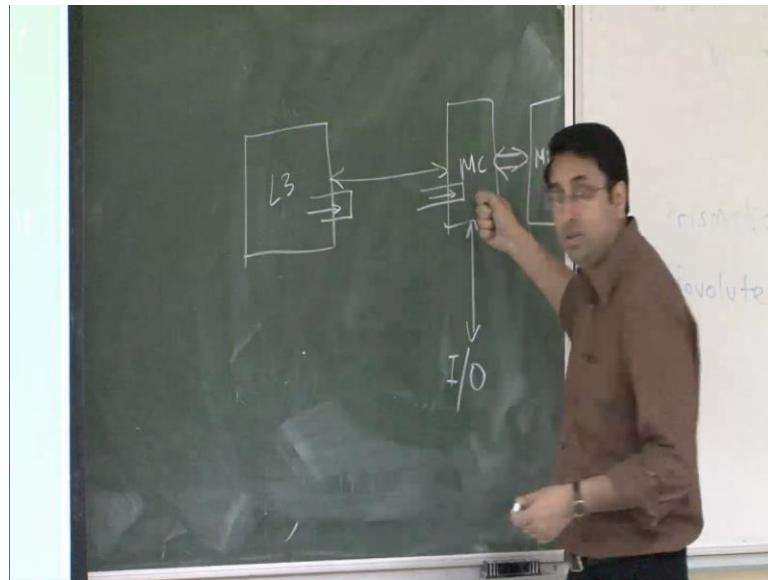
- Outermost level of cache talks to main memory on a cache miss (I/O may have a separate path)
 - The interface is normally through a *memory controller (MC)*
 - Memory controller is connected to the *bus interface unit* which in turn connects the system bus (aka front-side bus) to the outermost cache controller (CC)
 - The cache controller usually puts a request in a buffer and the BIU grants the bus to that request at some point
 - BIU usually carries out an arbitration algorithm among various types of buffers (connecting MC to CC)
 - The request ultimately gets queued into the MC input queue
 - MC decodes the request and takes appropriate actions

 MAINAK CS422 42

Now, will switch a little bit. So, a essentially a moving in bit out of the out of the processor were them look at main alright. So, what they will try. So, the out of most level of the cache hierarchy talks to make memory of the cache all right. For example, if I add the L1 and L2 only on L2 bits and talk to the main memory otherwise am happy with my cache are and off course the Io devices they have the separate parts to make memory now.

The interface to memory is normally through a memory controller memory controller is connected to the mass interface, which connect the system mass also is to know as front side bus to the out of most cache controller alright.

(Refer Slide Time: 38:18)



So, let us suppose at this is my L3 cache I have L2 and L1 before it and this is my memory controller and this is my memory. And L3 controller talks to the memory controller there may be other paths to with the Io device talk to the memory controller alright. So, this is the pass which connects the L3 memory controller, but the cache controller that is the L3 cache controller usually to is a request in a buffer.

So, taking about this particular pass, so that is a q here were the 3 will put the requests and eventually the request to be picked by the pass it. And usually carries out other of buffers correcting the memory controller cache control for example, there will a queue for cache pieces there will be key for addicted cache blocks. So, there may be any other little bit on that interface the request ultimately be guess queue internal memory controlling input the.


So, this queue alt this request ultimately get transferred to the q k all right and the memory controllers job is to decode the request and take appropriate action. So, you can

think about this memory controller as a very simple process. It essentially pick up request from this q decodes it you gets out the address you can some command time and launch the request to the main memory depending to the request.

(Refer Slide time: 40:04)

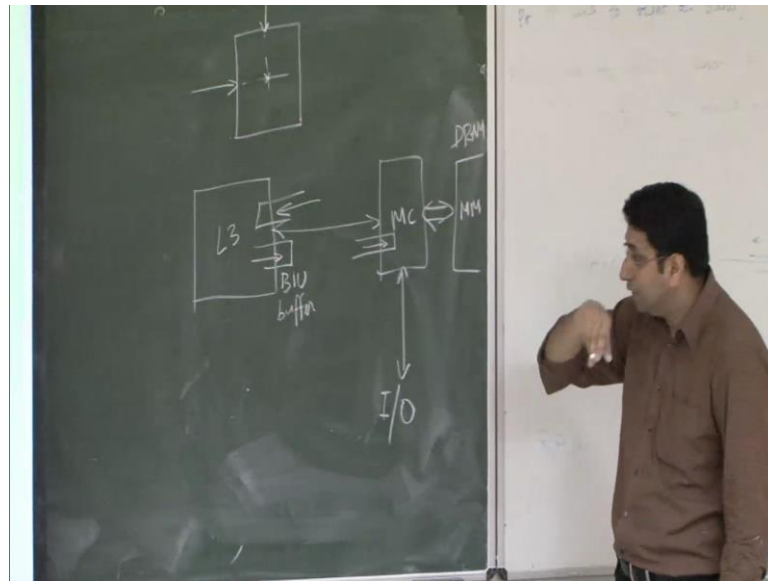
Main memory

- Steps involved in serving a cache miss from the outermost level (beyond miss detection)
 - Queue miss request in BIU buffer
 - How many request types (and hence how many different logical queues)?
 - BIU schedules the request and switches address, control, data (if needed) according to bus protocol
 - Request is fielded at the other end by MC and is put in a queue
 - MC picks requests out of this queue (normally in-order), decodes request type and address, and sends it to DRAM (dynamic random access memory)
 - DRAM access involves decoding row, decoding column, and reading out data (lumped together as access time)
 - Data reply is fielded by MC (DRAM bus connecting MC) and put in a BIU queue for scheduling

43

So, here the steps involve in certain a cache in from level beyond the based. So, you queue the miss request the behind the buffer. So, these the buffer the in buffer.

(Refer Slide time: 40:15)



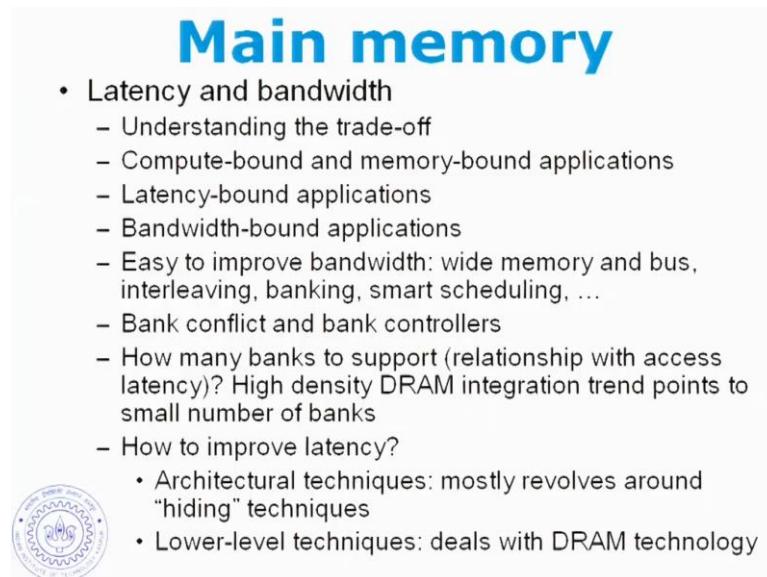
So, how many request tickets and how many different logical queues are there depends on your architecture. But, you can assume that there will be 1 queue 1 request for cache uses and 1 request time for edited caches all right the schedules a request. And, so it the address control and data according to the cache. So, essentially when picks up a request from here, it will launch the address on this box. It will launch whatever punctual that is needed that is command time to the request time it evicted cache block data which will be get back to memory all right.

The request is figure of the other end by the memory controller and a queue. So, this the memory controller fix request out of this queue normal in order there is no out of decodes request types an address and send it to the dram dynamic random access memory. So, this is the dram the dram access involves decoding row decoding column and reading out data lumped together and the as access line. So, dram is essentially arranged as 2 dimensional of ... So, there is a no id there is a column id and take a section point whatever the data is to the access that is what is meant by decoding row the data replies where the memory control.

So, eventually the replies is come back to the memory controller it will take in by memory controller and the scheme for schedule. So, there will be a queue from on this

side also and that will go eventually if we schedule and the processor the cache controller we define that. So, this particular bus is often called a channel that connects a memory controller to the ...

(Refer Slide time: 42:30)



Main memory

- Latency and bandwidth
 - Understanding the trade-off
 - Compute-bound and memory-bound applications
 - Latency-bound applications
 - Bandwidth-bound applications
 - Easy to improve bandwidth: wide memory and bus, interleaving, banking, smart scheduling, ...
 - Bank conflict and bank controllers
 - How many banks to support (relationship with access latency)? High density DRAM integration trend points to small number of banks
 - How to improve latency?
 - Architectural techniques: mostly revolves around "hiding" techniques
 - Lower-level techniques: deals with DRAM technology

So, there are 2 important parameters. So, we talk about latency and bandwidth. So, let and see essentially is time taken the and command send to the dram to the time you never you come to know the data back. So, that is the of own access all right and the bandwidth essentially how fast was the data on the first data come out how fast can you send out the remaining pieces of data so far.

Example for a talk about the cache low it is it is a discussion that the sixty 4 byte cache typically the dram in the space would be 64 bits. So, this particular processors this channel would be for example, 64 bites typically the all right it means to transfer a 64 bytes cache block it require 8 transaction. So, go to past alright. So, the point is that if we had a wider it would actually reduce the transfer number of transfer. That is what the bandwidth once you have rate the data how fast can you transfer to the memory controller.

That is now there are typically 2 types of the applications we will look at the software

program 1 class is called compute were given a particular data point. We do a lot of configuration compute bound the applications and typically in these processors. We do not require much memory the because, the given the data point are the are the program which make a lot of time operating on the data.

So, essentially request the will be a the memory were as in memory bound applications you typically do small amount of or each data point and a large amount of data. So, they will be very much bandwidth bound by the here this is category typically contains a game like: applicable then lot of data to the transfer from the memory to the graphic processor. So, similarly there are other classification like latency bound applications and bandwidth bound applications similarly that the applications. So, these applications would not ask for too much of data, but when they ask it is a always to respond very quick otherwise their performance.

So, these are were variance bound applications where as bandwidth applications to a lot of data in a very shorts band. So, in that case bandwidth is more important. So, CCP improves bandwidth for example, we can have wide memory and bass it could have banking in memory. So, that you can access multiple banks in or it could have the smart scheduler at this queue. It actually figure out we talked about which requires should be schedule all right; if we have memory banks your should be to utilize as many banks. So, you should not be scheduling request which the same band in the same because they cannot be the. So, the question is how many banks support is there a relationship with access?

If I tell you that 1 request and cycles how many bands should I support is there relationship at all.

Yeah.

45 has 20 more than in banks if so that, every cycle I can send the request to banks potentially I have the request for every banks availability alright. So, but this relationship is the is not be use full because usually the access time is very large. You cannot covered that by having because a today points to small number of access and that is what today

there are fourth main banks.

So, we have to deal with this particular problem. So, you have to have a scheduler which should not request. So, it over of the schedule should be to pick up request which means you may not be able to do request and totally depend in bass. So, typically today what happens is that this particular queue will actually get scattered to queue for each bank. So, memory controlling the pick up a request before the request find out the bank number and put it to the corresponding bank queue all right.

Then it is up the bank controller to schedule the request from the bank queue. So, how many of these are architecture techniques mostly around highly techniques. These we are talk to more many of these already for example, all right. There are lower level techniques that, but they deal with dram technology which is not our agency, but there are techniques which is getting into technologies.