CO Earlier Discussion

Today discussion starts at 7:35 pm

For writing here, if you are not able to, click here

Notes

- [ ] Machine instructions and addressing modes.
- [ ] ALU, data-path and control unit.
- [ ] Instruction pipelining.
- [ ] Memory hierarchy: , main memory and secondary storage;
- [ ] I/O interface. (Interrupt and DMA mode)

Cache Memory

Even if you know to solve cache problem, for GATE a basic understanding is required. Most if not all GATE questions are asked with realistic values which are also used in standard books. So following standard helps.

So, why use cache?

Because main memory is slow. Currently accessing main memory takes like 100 clock cycles while CPU performs instructions in 1-2 clock cycles. So, without cache each instruction involving memory can be upto 100 times slow. Cache is a fast memory located near CPU. Being near CPU, it is expensive and small also. (we don't have infinite cache)

How many levels of cache?

Actually no limit. Usually L1 and L2 - currently L3 is also common. Intel architectures use the word “last level cache” for the lowest level of cache. L1 is the nearest to CPU and is about 4-6 clock cycles to access. L2 might take about 15 clock cycles and L3 around 40-50 clock cycles and this varies a lot with architecture.

L1 (Level 1), L2, L3 cache are some specialized memory which work hand in hand to improve computer performance. When a request is made to the system, CPU has some set of instructions to execute, which it fetches from the RAM.

How data is transferred- cache line and cache block?

So, data is not transferred in byte or word size from main memory to cache. It is in blocks- which is some number of bytes- say 64 bytes on Intel 3rd generation.

So, suppose in a C code we write

int a = 5;

this will be converted to a machine instruction

ST A, 5

which stores 5 at memory location A.

So, this is a memory access. We are accessing only 4 bytes (assuming int size as 4) but still the entire cache line size (say 64 bytes) is brought to cache if it is not there. - this may not be true always. Some systems where write through policy is used, for write operation, it may not update the cache (directly update memory only).

can page(os) and block(coincide/conflict with each other?

conflict?

i mean the location where both are stored cannot conflict?Is there any mechanism to

differentiate those?

In VM a page is a unit of memory. PAGE is not a physical concept- it does not really exist- what exist is *physical page frame*- which is a division of main memory. Its usual size is like *4K*. So, we can say, cache block is a smaller division of this. Cache block is residing in cache- but this block data is coming from main memory and hence from some page frame. But page and cache block have no relative significance in anything at least for GATE purpose.

There was a GATE 2010 linked qn asking transfer time between caches. The concept they checked is when different cache levels have different block sizes how data transfer occur?

The data transfer always occur in block size **determined by the requesting** memory level. i.e., if L1 requests data from L2, block size used is that of L1. So, suppose CPU requests a memory from main memory (not currently in cache), and L1, L2, L3 cache block sizes are 8, 16 and 64. How much data will be moved in total? (Ignore the final transfer to CPU)

Answer?

64+16+8 ?

yes, whoever answered this has very good chance in GATE :) such simple logic always helps in GATE questions.

64(MM to l3)+16(l3 to l2) +8(l2 to l1);For time taken we should consider search time also right?And search time and access time are same?

yes.. any question mentioning search time from standard sources? I haven't yet seen them. We can see below after mapping..

okkk.

**** DOUBTS (write @ bottom followed by indexing number):**

2) Why do we need a mapping? - (One reason why set theory is required for CS)

We need a mapping because cache is smaller than main memory. Otherwise there would not be so many gate questions also.

So, cache is like a mini main memory. (From now on we just say one level of cache).

So, suppose main memory is 4GB and cache size is 8MB and cache block size is 64B. (*cache block is same as cache line- so don't get confused*). Now the mapping from main memory blocks to blocks in cache is in m:n, what are m and n?

we have 4GB main memory.

So, let's divide using 64 B blocks, we get 4 * 2^30 / 2^6 = **2^26** blocks in memory.

In cache we have 8MB. So, 8 * 2^20 / 2^6 = 2^17 blocks.

So, the ratio is 2^26 : 2^17 = 2^9: 1

that is, m = 2^9, n = 1. That is the default mapping and is direct mapped cache.

SO 9 bits will be the TAG bit.

this is **important**- use of tag.

2^9 blocks of memory will be map to same cache block...so to identify which block is present we need tag bits

yes. there is possibility of 2^9 = 512 blocks to be present in a single cache block. So, after indexing (that is going to a particular cache block) we need to ensure our current memory request is matching

with the store tag there. This is done using comparators which compares the tag bits. So, ideally we want low tag size as it increases the storage required (we need tag for each block in cache) and also it increases the cost of comparison.

Hi , just one small query , if we have tag bits as 15 and direct mapped cache , so say block offset is 9. if it is asked in question , what is memory space reqd. to store tag info ? will it be 15 * 2^9 bits ?

block offset has no direct relation to tag size. Indirectly it relates because it determines the no. of cache blocks. tag storage size = tag bits * no. of cache blocks.

direct mapped cache ? (sir confusion in naming conventions)

okay..

tag	??? (this denotes which cache line we are going to access) but what it is called - index ????	block offset / or word offset? ???

yes, like i told those bits are used for indexing into cache.

yes sir agree but what it is called - index ?

yes, index bits. set index also. we see below in associativity..

ok sir

For the other, it is "block offset" because we are dealing in units of blocks. I'm not sure if they use word offset also- as we get a word or byte from there. anyway it will be clear in the question. GATE questions rarely have confusing terms- they usually describe situation in detail (which might cause confusion sometimes)

For minimizing block movement between memories .

3) Direct mapped cache

So, what's the issue with direct mapped? is it the best? Is it the fastest?

Direct mapping is the fastest mapping possible. But its issue is if two main memory blocks maps to same cache block location, the older one is removed. i.e., suppose we access memory like

A, D, A, D, A, D …

in this sequence where A and D are memory locations both mapped to same cache block, we will have no hits in cache. And this is not such a rare case. So, any way to avoid this?

use associative mapping which maps any block to any block

yes, the problem with direct map is that each index has “space” for “one block”- why can’t we use 2 blocks there? This will ensure our previous case of

A, D, A, D, ….

will be fully hit in cache and our worst case turned to best case. Putting 2 blocks in a index- now instead of a cache block, we have a set of 2 cache blocks- is called 2-way associative. We can even have 4-way associative and as seen from practice, real applications usually benefit most from 4-way associative cache.

So, what's the extra cost of associativity? Is it someway related to hashing?

hardware complexity increases.

we need mux to find block within set. -

- we need mux to choose tag bits (as there will be many tag bits for a set depending on associativity)
- and then using comparator we compare tag bits

with increase in associativity hardware complexity increases(ie 4 way set associative is more complex than 2 way)

can't we use decoder for MUX?

I thought decoder will be efficient than MUX ,as we need a mux for every bit in the Tag.

@Arjun@pooja

Tag bits increased,so hardware cost(comparator size) increases.

how tag bits increased? In Case of pure Associativity cache.

yes, associativity effectively reduces the cache index entries. So, more main memory blocks can go to a given cache set- so, we need more tag bits.

The indexing process into cache can be seen as a hash function and hence the cache can be seen as a hash table. Now, associativity behaves like a hash table which resolves collision by chaining but restricts the chain length to fixed size (4 for 4-way etc.). So, now all problems of chaining comes here too. But being in hardware, we can use complex hardware to ensure that there is no “sequential search” required as for normal chaining. i.e., in one go all the 4 blocks in a cache set (for 4-way) are compared using their tag bits. This is fast but still not as fast as direct mapped.

In direct mapped ,when cache is full and a conflict occurs then it will be both conflict and capacity miss right?

So, **what's a conflict miss?**
Suppose acces to block m of memory was required and when mapped to cache its cache block no. comes out to be b
Now some other memory block k is already residing in that cache block
so thats a conflict miss for block m

no of blocks in cache be C …… this happens if
m mod C = k mod C = b
(Its called conflict miss because the block is not there in cache so a miss and conflict occurred due to different memory blocks being mapped to same cache block thus Conflict miss)

- And I think this block m is being accessed second time else it would have been compulsory miss

Can a conflict miss occur in a fully associative cache?
Why?

no. because that is how it is defined. It is the miss which happens when we move from a fully associative cache to a direct mapped one. So, to count the no. of such misses we first consider the cache as fully associative and then as direct mapped or set-index mapping as given in question.

Does increasing block size causes increase in conflict miss?
?
No answer? We have to assume cache size is constant here. So, increasing the block size means, less no. of main memory blocks.
yes.Because Ratio(M:N) increases ,so conflict/ miss increase.

yes, that ratio will increase. But we cannot say conflict miss increases. It either remains the same or increases.

I think if cache size is same then no of blocks per set remains same ……
but conflict miss inc or not(i’m not sure)

though probability of getting more data in one block increases
but conflict miss won't increase as data which was suppose in 2 consecutive blocks previously will now be in one block but consecutive blocks won't be creating conflict
or they might

yes, more data in a block. but that doesn't reduce conflict miss. That reduces compulsory miss.

this is a good read.
http://courses.cs.washington.edu/courses/cse378/02sp/sections/section9-2.html

@Arjun But probability of conflict increases right(eg.Hashing)?

yes, probability increases. thats correct..
But conflict depends on set of memory references (is this the reason)?
yes- **sequence** of memory references.. **Ya the document id good**.Thanks for the question.

Capacity miss?
There is no space in cache(lines are occupied) to place a main memory block (we are going to access). We count assuming cache is fully associative- that is for every main memory block it can be present in an cache block.

Compulsory miss/cold start miss ?
miss occurs when a block is accessed for first time. Can this be reduced?
I think no.But can we do prefetching?how??
Bringing lines to the cache before being requested,this Can reduce compulsory and capacity misses .
Reference yes, but prefetching is

outside gate topic..
okk.

yes we can reduce by increasing block size..(assuming cache capacity same)
by increasing block size more words would be there ..and less no of blocks..so less compulsory miss.
But conflict miss/capacity miss will increase.

Doubt- Can capacity miss occur in Direct mapped cache ? (As there will either be conflict or compulsory miss as per me) ?

If we are asked to count the no. of capacity misses in a direct mapped cache, count it assuming fully associative cache. yes, the same miss might be counted twice as capacity and compulsory-but that's fine.

5) Sir, if we invalidate cache after every context switch is made (for protection), so for the next process on CPU, will it have to fill its locality again in cache and as Time Quantum is limited, So before utilizing it, it gets preempted (switched) for which once again the cache is invalidated. So, in this whole process, does Cache State gets saved somewhere ?? If not, then why the performance is not degrading because of low cache usage ?? Or the whole concept of mine is wrong??

yes. Cache is a system thing and shared by all processes. So far we assumed only physical memory address in caching. i.e., we index using physical address bits, and we compared tags using physical address bits. One issue with this is above- on a multi processing system what we do on context switch? Another issue?
Ans:Flush the cache (or) set the invalid bit (or) add process ID as extra parameter.

Context switch frequency in current system are very high- like in milli seconds.. So, flushing the cache is a real bad thing to do. One technique used here is ASID- address space identifier which is like a process id on hardware- and can be used to identify the cache entries for each process.

But the main problem of physically addressed cache is we need to get physical address before looking up in cache. So, in VM system, we need to access page table. Since page tables are in main memory, we go like
MM - Cache
that is we access cache after we look in MM which is not really good.

This problem is mostly solved using TLB- which acts like a cache for MM. Still, we can use VI-virtual indexing for cache and use physical tag bits. This is the common one in use now. This requires cache indexing and TLB lookup to be done in parallel. Once indexing is done, to compare the tag, bits come from TLB.

@Arjun sir consider in VA page number part is used to mapped Frame number simultaneously index part is used to find index (I am asking in context of VIPT) then if page is not present in TLB but offset matches with some index(set) then it should create pb na.becoz we are reading actual frame no from page table ?but sir whole notion of virtual address physically tagged based on these that offset bit of virtual address won't change during translation so we can use it to for fast translation

Why you used "offset" here? Offset bit in VM is used for page offset. It usually is not same as index bits for cache though might include some bits.

yes, that won't change because page and page frame size are same. But page offset bit is like 12 bits (4K page). But we need to address the full main memory for caching. We need higher

bits too- not just the lower 12 bits. If we use a higher associative cache we might be able to fully fit the index bits within offset part, but then also tag bits are there.

And for your question replacing "offset" by "index bits" - that's clearly an issue and in such cases, we have to wait till a MM lookup happens. Fortunately TLB is good and has >99 hit rate usually. So, not a big problem.

Sir can u explain how exactly mapping between index bit and offset takes place??
virtual address consist of virtual page no and offset
so virtual page no given input to tlb now how do we get tag from there?

From TLB- we get physical address rt? And that includes the tag bits- top "x" bits.. Without TLB/page table we have only virtual index.
yes

Using tlb we get physical tag

@arjun do we need cover this virtual address physical indexed.If yes give a good material? which one?

No need to go deep. It is pretty much what we discussed so far. You can see here.
http://cseweb.ucsd.edu/classes/fa10/cse240a/pdf/08/CSE240A-MBT-L18-VirtualMemory.ppt.pdf

6) Explain about independent memory , simultaneous memory organization and hierarchical

A common doubt is during a cache miss should we include cache access time or just main memory time. In such cases we go with standard resources. Here, most standard resources say on cache miss, we have "**miss penalty**". So, if question has this term we are done. But usually it says access time and hence we are not yet done.

Actually for read access, we always go with hierarchical access by default. Reason for this is given here (for pentium architecture)
http://download.intel.com/design/intarch/papers/cache6.pdf

But if question specifies simultaneous access, do that- though this is highly unlikely.

Also, both methods should give almost same value (only these mock tests make an issue here, not GATE)

But sir in simultaneous access loaclity of reference fail.So how can we use cache in that case?As cache is used to employ locality of reference.

Sir please explain briefly how simultaneous memory is accessed.
I mean suppose there are L1,l2,m as memory levels
then simultaneous says searching all at once ?

yes.
Sequential means processor interacts with cache only. Cache gets data from MM if needed. Processor doesnt interact directly with MM.
In simultaneous, processor uses the same bus to request cache and MM. This is the problem of this, bus become busy even when data is in cache. You can see the above link of intel- this is described there.

Yes sir i'll go through that
What do we mean by saying that processor uses same bus for cache and MM?
(If it uses same bus then it will access sequentially as just one bus is there - then how is it simultaneous - I didn't get this point)

I read somewhere a following memory organization-
L1,L2,M then processor sequentially searches like first in L1 if miss then in L2 if miss then in MM
But as soon as data is found in whichever level (L1,L2,MM)
it is directly transferred to cpu - without following the hierarchy of moving the data to upper levels
This type of memory organization comes under which category ?
It mentioned that frequently accessed blocks are kept in upper levels as search is sequential

yes, this is the default in any cache mechanism. Once data is found, it is given to higher level and at the same time to CPU too. -- CPU sniffs the data actually and this technique is called sniffing- its outside gate scope..

(Ok the source from where i read this created the confusion it mentioned it as independent memory organization)
Sir is there anything as independent memory organization ? If yes please explain

There are two ways of handling cache related memory access – look-through cache and look-aside cache. In look-through cache, the cache is checked first and the main memory access starts only when there is a cache miss. In look-aside cache, cache look-up and main memory access is started simultaneously. Once a cache hit occurs, main memory access is cancelled.

A system has the following parameters:

Cache Access Time = 30 ns

Main Memory Access Time = 300 ns

If the system has a hit ratio of 0.95 then the difference between average read access time when look-through cache is used and average read access time look-aside cache is _____(ns).

Solution:

Notations are as follows-

Cache Access Time = TC

Main Memory Access Time = TM

Hit Ratio = h

Average Access Time = TA

For look-through cache:

$TA_1 = TC + (1-h)*TM$

$= 30 + 0.05 * 300$

$= 45$ ns

For look-aside cache:

$TA_2 = h* TC + (1-h)*TM$

$= 30*0.95 + 0.05 * 300$

$= 43.5$ ns

Here look through is sequential/hierarchical and look aside is parallel?And is the description is correct for those?

yes, they are correct. what's confusing here? Look-through was never asked in GATE though..
What is the default we should take?Lookaside?.Ok.

8)**Explain Homonym problem where same virtual address can mapped to different physical address ?yes sir but i didn't understand how same VA mapped to different frame no... so only sol is flushing cache only?**

This happens in virtually addressed cache. Actually needs bit discussion on this, but I have to revise this portion. A question on page coloring was asked recently.
That happens only when 2 processes use the cache on a multi-processing system as we discussed earlier. We could flush the cache on context switch but this is costly.

9)H hit rate
C cache access time
M memory access time
T avg = h*c + (1-h) M
Or
T avg = h*c + (1-h) (M +c)

Can I know which formula to use and where
discussed above- use 2nd in most cases unless told otherwise. Also keep in mind that for disk access when page fault service time is given it includes MM access. i.e., there the formula is similar to the first one above- don't confuse the two.

We have discussed cache a lot, but still some portions like page coloring are left. Also, each problem here must be carefully read and just a bit of imagination as to how stuff work is enough to get answer to most questions correctly. I have seen some people in gateoverflow who gets almost all questions in CO correctly. There isn't much to study here- just dont mugup formula instead just imagine the system and solve questions.

one last thing about time taken for movement.Here
if L1 requests s that of L1. So, suppose CPU requests a memory from main memory (not currently in cache), and L1, L2, L3 cache block sizes are 8, 16 and 64.
Here time of{access time of MM+l3+l2+l1) is it correct?In case of serial organisation.
If parallel then {access time of MM}

that depends on the memory bandwidth. That is if the whole block can be transferred in one go or not. Sometimes this might require multiple requests though practically this is usually 1 access.

It varies depends on organisation also?Like look aside/look through?

yes, but as told earlier look aside is more hypothetical and so unless mentioned we go for hierarchical..
okk ...

Pipelining: (sir plz explain all dependencies and hazards i dont know this)

I guess most of you knows about pipelining and so we can directly go at problems. Also for this topic just solve "PREVIOUS GATE" questions and that is enough and covers all possible questions.

-

Please give a brief idea about these pipelining questions???

In a pipeline system we split an instruction execution to stages. That is a normal instruction might be broken to 5 stages IF, ID, EX, MA, WB. So, why we do this? **Because when the first instruction enters stage 2, we can use stage 1 for next instruction. Each stage produces its output to a buffer and the next stage reads from it.**
@sir this buffer is independent of instructions?

the buffer is tied to the stages.. Each stage has output buffer.

it means if two instructions parallel execute some stage(say) then race through condition will be there?

how can 2 instructions parallelly execute the same stage? Because there is only one unit by default for each stage and hence at a time it takes only 1 instruction. If some pipeline stage has multiple units, then yes we need more buffers or some complex mechanism but this is out of gate scope..

ok :) sir can you give example for this stage buffer . ?means some instructions in which its output is stored nd other instruction coming and then using the stored result?

its like when IF stage completes it puts the output on buffer. I guess it is important from implementation point of view. Just imagine how to transfer data across stages. We cannot just send the data as the **next unit might be busy**. So, we have to hold it and this requires buffer.

okk leace sir you can carry on other points.

splitting into stages increases throughput
yes.. In ideal cases this increase can be upto the no. of stages being used. How?
speedup?
speed (max)=no of stages

yes, because we effectively reduces the instruction execution time to time to finish a stage. But this stage delay depends on the max delay of any stage in the pipeline.

yes

11) Use of pipeline registers ?

If an instruction has finished execution (in IF stage) but can't move to next stage (as its occupied by another instruction) can its data be shifted in buffer and next succeeding instruction can use this stage - IF) ?

No. that's not the reason. Actually that cannot happen with buffer. because suppose IF stage is fast and produces the next output also before the ID is done, it might overwrite the buffer.
We cannot just send the data as the **next unit might be busy**. So, we have to hold it and this requires buffer

Ok sir

12) In IEEE floating point representation (If you can cover this topic in discussion) - biased means like if exponent is 8 bits then is biased fixed to be 127 or it can vary as per ques
if question is on iEEE then they have predefined that it is 127. and it's calculated so with 8 bits 127 is the maximum positive number possible in 2's complement representation. if it is not given that it's iEEE then they will define the bits of exponent and then bias can be calculated as the biggest positive number possible with that many bits in 2's complement like if 5 bits are exponent then the biggest positive number in 2's complement will be 2^(n-1) -1 wich will be 15 . 15 will be the bias . similarly in ieee double precision we have 11 bits for the exponent that's why bias 1023 (2^(11-1)-1)

yes IEEE 754 has fixed them.

13) **DMA Operations**

Consider a disk drive with the following specifications:
16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk is operated in cycle stealing mode whereby whenever one 4 byte word is ready it is sent to memory; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMA cycle. Memory cycle time is 40 nsec. The maximum percentage of time that the CPU gets blocked during DMA operation is_____

Answers given @ http://gateoverflow.in/1393/gateoverflow.in seems to be too much confusing .
whole DMA process:
1)I/O device request DMA,request process by DMA controller
2)DMA controller enable BR(bus request)
3)Processor enable INTR(interrupt) and BG(bus grant)
4)System bus is control by DMA controller from this point onwards
5)I/O device directly transfer data to/from memory directly via DMA controller(System bus)
6)At completion of data transfer disable BR then system buses are restrained back to cpu
Now there are 3 modes in which DMA operates
--Interleaved mode(½ DMA cycle +½ processor cycle)
--Cycle stealing mode(word transfer)
--burst mode(complete I/O transfer)
Now keeping these theory in mind solve above
first we need to calculate how much time it take to transfer 4 byte to memory
3000 Round ---60 sec means 1Disk Round takes 20 msec(1 Disk round is nothing but 1 Track)
therefore,512 sectors ----20 msec(512 sectors in a track)
512*1KB--20 msec-->1B takes 40 msec now we need to transfer 4 bytes so
1B----40msec
4B----160 msec
CPU is blocked only for time period where bus transferring activity is done(i.e all cycles wastage due to transfer system bus to DMA controller ,during actual transfer of data processor is busy by doing some other activity which does not involve system bus) therefore only 40 nsec out of (40+160)nsec is wasted so 40/(200)=20% hope this help….
20% not is in the answer……..
well in sol they have just consider data transfer time and not total of (bus+data) transfer but ideally they should….
Here what is bus transferring activity?
step no 3 constitute bus transfer activity it has many names memory cycle time,transfer time(bus not data)

---

14) Consider a disk with following specification
no of surfaces - 8 , inner dia - 4cm, outer dia - 12 cm , inter track dist - 0.2mm ,,,no of sectors per track - 20

RR = 3600 rpm , sector size = 4KB. Find -

a) capacity of disk
b) data transfer rate .

Answer : http://gateoverflow.in/2650/gateoverflow.in
No. of tracks = record width(which is outer radius - inner radius)/inter track distance
Tot capacity of disk = #surfaces * #tracks * #sectors/track * sector size
Data transfer rate= 4 * 20 * 6 KBps

15) Consider a cache with 64 blocks and block size of 16 bits . the block no of byte address 1600 is ?

I did it like this - 16 bits (so 2 bytes)
block no in memory = 1600 / 2 = 800
so block no in cache will be - 800 % 64
(PLEASE verify if this is correct)

yes, you are correct.

16) what is the difference between access time and transfer time with reference to cache i.e. when a cache hit occurs a word is accessed and transferred to CPU and when a cache miss occurs a block is transferred from higher level memory to cache and then word is transferred to CPU. So is this transfer time equivalent to (cache access+memory access time)..??
In hamacher book there was concept given as while transferring a block the time taken is equal to first word access time of higher memory level + rest of word with access time of cache..it is a bit confusing…please explain the difference??

I guess we discussed this already..

17)In instruction pipeline problems, in what scenario do we use split-phase access?

If any instruction reads a data written by prior instructions we can use split phase access. “we can use” means we have to use unless told otherwise in question. This goes like follows:

In WB stage an instruction writes the data to register file and in ID stage an instruction reads data from register file. With split phase access, the write to register file is done in first half of a clock and

read done in next half. So, effectively we do both operations in one cycle. This does not happen with other stages as those stages might not happen so fast- say EX unit might involve many arithmetic operations, MA unit requires data to/from memory etc.
Thank you sir. ""we can use" means we have to use unless told otherwise in question." this is what I wanted.

Plz don't edit. (a lot of common confusion so plz let it be answered by arjun sir)
from **********

sir regarding pipeline i have a few major doubts.

1-- are raw dependencies are only considered between adjacent instruction ??

i think they should not be . and sir if i have 100 instruction and first is like

r0<- r1+ r2
now all the remaining 99 instruction are reading r0 . does t means there are 99 raw. or i have to consider the cases till it will affect the pipeline.

all are RAW dependencies. Just think in a logical way. If we remove any of them above this instruction which writes value to R0- our result will be wrong. But this is not the case for WAR dependency- it depends only on the last write.
so are we have to consider consecutive instruction for WAR only?ok.
it's not consecutive- just the previous one (it might be even 2 or 3 instruction before) for WAR

2--and sir i can't understand what happens in the memory access phase. ?
if i have a instruction like

R0<- R1 + 4(R2)

what i have read and found that risc only support either the value from register or indexed addressing mode.

what actually will happen . when execution will be over how addition will be done.

I didn't get what you are asking?

q3 -- the consider these 2 instructions ,

I1: R0<- R1 + R2
I2: R5<- 4(R0)+ R6;

sir what i think in this case operand forwarding will not help me because i actually don't need r0 but it is used as indexed addressing mode. i m rght or wrong .
That is wrong. R0 is needed by I2. But..

I guess this is what you asked earlier- since MA is after EX stage in a classic RISC pipeline how this instruction can be execute. - this is not possible. We need a separate load instruction and then do add in next one.

A 5 stage pipelined CPU has the following sequence of stages:

IF — Instruction fetch from instruction memory,

RD — Instruction decode and register read,

EX — Execute: ALU operation for data and address computation,

MA — Data memory access - for write access, the register read

at RD stage is used,

WB — Register write back.

Consider the following sequence of instructions:

I1 : L R0, 1oc1; R0 <= M[1oc1]

I2 : A R0, R0; R0 <= R0 + R0

I3 : A R2, R0; R2 <= R2 - R0

Let each stage take one clock cycle. is the number of clock cycles taken to complete the above sequence of instructions starting from the fetch of I1 ?

(A) 8

(B) 10

(C) 12

(D) 15

Dependences?

RAW: I1 <- I2, I2 <- I3
WAW: I1 <- I2
WAR: I2 <-I3

Assume no operand forwarding:

1	2	3	4	5	6	7	8	9	10	11
IF	ID	EX	MA	WB						
	IF	ID	ID	ID	EX	MA	WB			
				IF	ID	ID	ID	EX	MA	WB

Now, we can use operand forwarding **(by default use operand forwarding)**. Here, as soon as any stage has the data required for the next (or even some following) instruction, it is immediately forwarded.

why cant IF(3rd inst) cant start at 3?And

yes, good question though we dont have a difference in answer. We talked earlier about buffer after each stage rt? So, here suppose IF stage of I3 is done and it completes before the ID stage of I2 is completed, then will overwrite the output of IF stage of I2. To handle this we would need multiple buffers- why use them when there is no practical use.

okk.

sir one question ? when we have to consider the half cycle rule . if the stages are of risc ?? or if it is given that they have used the risc pipeline .

neither. Ideally they should mention it. But even otherwise just always use it for WB-ID stage unless otherwise told so. I haven't seen any GATE question where "RISC" word has any impact on the pipeline calculation. Classic RISC pipeline has well defined 5 stages. I guess there is nothing more to it. Also, in pipelining GATE has enough previous questions- no need to go anywhere else. Must avoid mock test questions.

the instruction decode can start under the writeback ??

That is the split phase. working. You can see above, it is explained.

1	2	3	4	5	6	7	8
IF	ID	EX	**MA**	WB			

	IF	ID	EX	EX	MA	WB	
		IF	-	ID	EX	MA	WB

Why not ID of I3 at 4th cycle? same reason as why IF not at T3 for the non-operand forwarding case. You can see above. Any stage can start only when the next stage of the previous instruction has started to avoid buffer corruption.

ok.Any good material for this?.

All materials I have found are here but previous gate questions in pipeline are the best to see.

http://www.gatecse.in/co-architecture/

okk.I can't understand the part of data corruption,buffers(what you have told).Ok i ll refer those.

data corruption part is just logical - may not be in book.

sir can’t understand what u are saying .

1-- are raw dependencies are only considered between adjacent instruction ??

i think they should not be . and sir if i have 100 instruction and first is like

r0<- r1+ r2
now all the remaining 99 instruction are reading r0 . does t means there are 99 raw. or i have to consider the cases till it will affect the pipeline.

all are RAW dependencies. Just think in a logical way. If we remove any of them above this instruction which writes value to R0- our result will be wrong. But this is not the case for WAR dependency- it depends only on the last write.

so are we have to consider consecutive instruction for WAR only?ok.
its not consecutive- just the previous one (it might be even 2 or 3 instruction before) for WAR

how many will be raw if the above situation happens .

if i have 100 instruction and first is like

r0<- r1+ r2
now all the remaining 99 instruction are reading r0
there will be 99 RAW dependencies.

****

@Arjun, can you please suggest any good resource for learning operand forwarding ?

you have seen the links given in gatecse site ? yes. Actually sometimes it gets confusing, especially when it is not clear from which stage forwarding is done.
unless told in question always assume the best case. I guess theory is not much here, but question they can form any way.
Default is from WB right ?

No, for operand forwarding it is from whichever stage has the operand ready. Say for load instruction it will be MA stage, for ADD instruction EX stage. WB is for register read- which is in case we dont use operand forwarding..

Got it :) This part was most confusing for me !

okay :)

So, finish?
okk.
I'll add page coloring here later.. Not very important but have been asked once.

Q2.Consider the following instructions.

I1 : R1 = 100
I2 : R1 = R2 + R4
I3 : R2 = R4 + 25
I4 : R4 = R1 + R3
I5 : R1 = R1 + 30

Calculate sum of (WAR, RAW and WAW) dependencies the above instructions.

RAW: I2<- I4, I2<-I5

WAR: I4 <- I5, I3 <- I4, I2 <- I3, I2 <- I4

WAW: I1 <- I2, I2 <- I5, I1 <- I5

There was a question on MAL rt? Asked in GATE 2015.

That is the only advanced level question in pipelining ever asked.

http://gateoverflow.in/8560/gate2015-3_51

Sir, how to identify WAW hazard?
according to me, if we are writing to same variable more than once then it is WAW hazard. is it correct?
yes.. so it is easy to identify.

how to avoid WAW hazard? or when we face WAW hazard is it true that we always need to pay penalty?
its not causing problem in pipelining rt? but for parallelization we have to but is taken care off by register renaming. So, practically no penalty. But in high level- it has some problems for compiler optimization and so SSA form is best suited for compiler optimizations. SSA might be asked in compilers this time- was asked last year.

I guess gateoverflow is having load issue- should be fine from tomorrow.. Anyone here has any doubt in above question?

One thing: This year gate syllabus is pretty less- I guess this might be a record at least since 2009. So, each topic in syllabus gets more weightage. In remaining days just ensure you cover each topic in syllabus. Say, if you have no basics in some subject you might well leave it alone now as this is not the time for starting from beginning. But if you know something, just ensure you cover properly by seeing previous gate questions. Only few GATE questions are tough every year- so while solving shouldn't miss them. Year wise solving is recommended from now. Try getting 70+ for all previous papers in 3 hrs and you have no need to worry.. If you are getting less, see which subjects are problem.

NB: Do not really care for mock tests marks. But ideally you should be scoring 50+ for IITs. Mock gate mark and actual gate mark can be about +-15 and so it can go either way. Usually first timers get more than what they get in mock gate, and repeaters get less. But this is not a rule.

okay then. Tomorrow we have compilers but might start late. I'll update..

good evening sir one question if operand forwarding is not to be used then when to schedule I2 if it is dependent on I1 immediately after I1 finishes or after 1 cycle of I1 finishing?