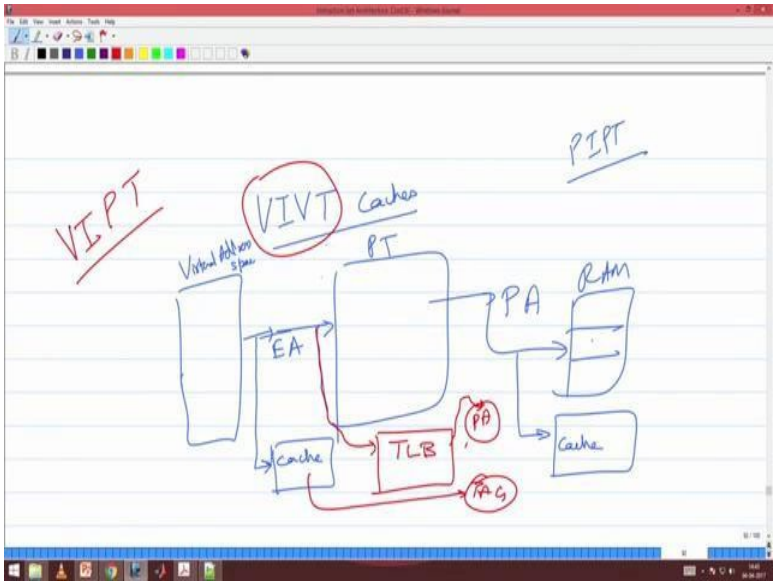
Computer Organization and Architecture
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 38
Part – 01
Virtually Indexed – Virtually Tagged and Physically Tagged Caches

(Refer Slide Time: 00:23)



So, we had been talking about virtually indexed virtually tagged caches, remember right. So, there is so how does the whole system look. So, there is a address space, this is the virtual address space. So, your program has an address that gets added to the segment base and you get an effective address here. This effective address essentially goes through a page translation mechanism, and you get a physical address which you use to basically access your RAM. In this part, there exists a cache memory also, so a copy of this RAM is kept in the cache. So, what we suggested yesterday last class was that we can use this effective address to the cache. We can use this effective address to find out if to see if it is there is a map.

We can index and find out if there is a entry. And by that what will happen if there is a cache hit, I can forget this whole step; if there is a cache hit then I can forget this whole process. If there is a cache miss then I need to basically look into this.

Now; that means, what I am going to store in the cache is the effective address I am using the effective address to index into the cache I am using the effective address to also do what I am using the effective address as the bits of the effective address as the tag bits also. So, essentially this actually comes as a. Now, what would be the disadvantage of this, this is where we stopped last time. So, what is the disadvantage of a virtually indexed virtually tagged cache (Refer Time: 02:42). Are you able to follow my question?

Student: Sir, yes.

Yes, why is that yet so low still low, ok. Tell me, what is the disadvantage of the disadvantage of having a virtually indexed virtually tagged cache. When I want a write back policy I am gone right I have to do a reverse address mapping to find out. So, I need to have a reverse space table in that. So, what is interesting for me is if I have if I can do the indexing if I do completely physically indexed physically tagged cache that means, it will start here. Now, the cache comes. Then it becomes extremely sequential it becomes extremely slow; there is no advantage of having a cache meaning like meaning I will not get that advantage of having a cache. Interestingly what happens is this effective address is also let me take the red pen here is also having a bypass called this TLB which you still remember translation look aside buffer here itself I will get this physical address if there is a TLB.

Now, this is again indexing right, and this is also indexing. So, can I have something like a virtually indexed physically tagged cache? Can I have a virtually indexed physically tagged cache that makes my life much more easier. In the sense, what happens is I get an effective address. When I am indexing into cache to find out whether the entry to which it is hashing to mapping to is a valid entry or not and retrieving the tag bits then I get back the tag bits and only I compare know. So, I index using the effective address and there I retrieve the tag bits, are you getting, I retrieve the tag bits to find out what I said. And when I am retrieving the tag bits at that point of time by that same time what would have happened is if the effective address is also there in the TLB, I would have got a physical address also. So, now, this physical address I can compare with the tag bits that are coming here, and say if it is a cache hit or a miss.

So, what happens is when I want to replace it back, I can reconstruct exactly the physical address. So, I can implement a write back policy here are you able to follow this. So, in a

VIPT, so what happens I am indexing into a TLB, at the same time I am indexing into the cache. So, the two indexing are concurrent. By the time I am getting the tag bits out of my cache, at the same time if there is a TLB hit, I will also get the physical address out of the tag. Now, I can physical the address out of the TLB. Now, I can go on compare the TLB and the cache output of this tag bits. And now go and say if it is a hit or a miss. Are you able to follow, what I am saying right?

So, if I have a VIPT cache then again I am getting the same advantage as a VIVT cache. And I get out of that major disadvantage of not able to implement or having additional circuitry to go and do a write back implement a write back policy, wherein I want to move a cache data back to the memory. So, this is the advantage. So, VIPT is the one that is followed in modern current processes. Many of the processors all of the processors use a VIPT.

Now, why does VIPT work, I told you right for lot of questions in computer architecture there are one-one word answer which will imply the answers. Why does VIPT has a concept of VIPT is a success if I have what? If I have a TLB hit and the cache hit together right, now the way you should answer is why there is a chance of TLB hit and the cache hit to be together. VIPT is a failure if there is a cache hit and a TLB miss, but will it happen at all? Right, it will not happen because whatever address that are stored in the cache is a subset of the pages that are stored in the TLB, we recently used. If you are going to look at replacement cache replacement would be more than the TLB replacement because the granularity there is 4096 bytes the size of a page is much larger than the block of a cache.

So, there is the very, very remote chance probably I will ask as a question for you to think over in your end semester, remind me that can there be a situation where there could be a TLB miss and the cache hit. Most probably it may not right. Are you able to at understand what I am saying? Right, so having a TLB miss and a cache hit is very, very remote right and that is why VIPT has a concept is much more successful. Fair enough?