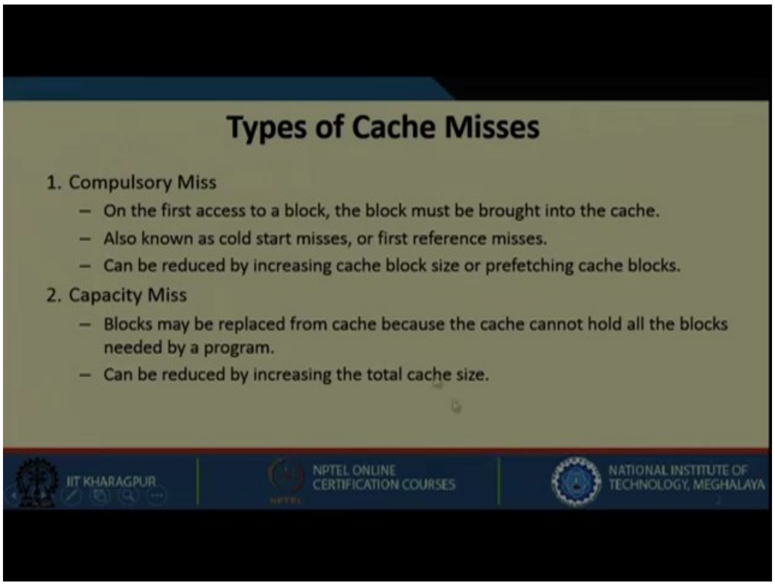
Computer Architecture and Organization
Prof. Kamalika Datta
Department of Computer Science and Engineering
National Institute of Technology, Meghalaya

Lecture - 31
Cache Memory (Part II)

Welcome to the next lecture on Cache Memory.

(Refer Slide Time: 00:24)



Types of Cache Misses

1. Compulsory Miss
   - On the first access to a block, the block must be brought into the cache.
   - Also known as cold start misses, or first reference misses.
   - Can be reduced by increasing cache block size or prefetching cache blocks.
2. Capacity Miss
   - Blocks may be replaced from cache because the cache cannot hold all the blocks needed by a program.
   - Can be reduced by increasing the total cache size.

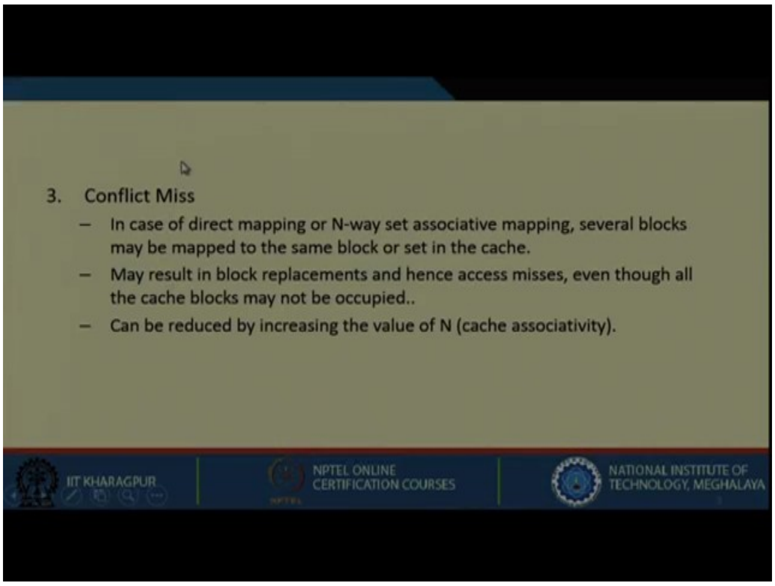
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us now understand the type of cache misses; by cache misses we mean that we want to performs certain operation like read or write, and that particular data that we are trying to access is not present in the cache. So, we have to bring it from some lower level memory into cache. Under that we have many kind of misses, the first one is called compulsory miss. As you know that every time when we access a data or an instruction, first we try to access it from the cache. Initially cache is empty. The programs and data are loaded in main memory, as and when it is required it is brought from main memory to cache memory, and then to the processor.

So, every time there will be a miss initially, and then due to the locality of reference that particular instruction which has been brought will be accessed again and again. So, the first one which is compulsory miss on the first access to a block, the block must be brought into cache. Also this is known as cold start misses or first reference misses. So, every time for all the instruction or data, there will be a first reference miss. How this can be reduced? This can be reduced by increasing the cache block size, or prefetching cache blocks. If you can prefetch a cache block early, then this can be reduced.

Next is capacity miss. Blocks may be replaced from cache because the cache cannot hold all the blocks needed by a program. We have seen that in direct mapping; even if there is space we cannot bring all blocks at the same time. Under such situation what can be done; that is called capacity miss. The blocks may be replaced from cache because the cache cannot hold all the blocks needed by the program. How this can be reduced? This can be reduced by increasing the total cache size.

(Refer Slide Time: 03:30)



3. Conflict Miss
   - In case of direct mapping or N-way set associative mapping, several blocks may be mapped to the same block or set in the cache.
   - May result in block replacements and hence access misses, even though all the cache blocks may not be occupied..
   - Can be reduced by increasing the value of N (cache associativity).

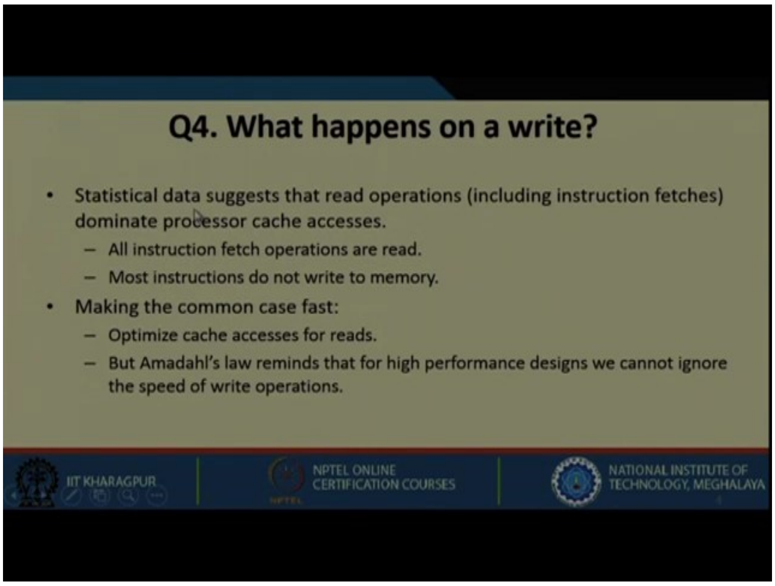
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Another kind of miss is called conflict miss. In case of a direct mapping or N-way set associative mapping we have seen that several blocks may be mapped to the same block or set of the cache.

This may result in block replacement and hence access misses even though all the cache blocks may not be occupied. We say there is a conflict, meaning both for direct mapping and N-way set associative mapping we have seen that the blocks where the data is present may result in block replacement and hence access misses even though all the cache blocks may not be occupied. So, there are some free blocks available, but still we cannot bring the data in those free blocks.

So, in this case there is a restriction that we cannot bring those blocks in any other blocks of the cache memory. This can be reduced by increasing the value of N where we can have the possibility of many main memory blocks to be mapped to one set, and in that set we have to do some searching for getting the particular data. So, this comes under conflict miss.

(Refer Slide Time: 05:12)



Q4. What happens on a write?

- Statistical data suggests that read operations (including instruction fetches) dominate processor cache accesses.
  - All instruction fetch operations are read.
  - Most instructions do not write to memory.
- Making the common case fast:
  - Optimize cache accesses for reads.
  - But Amadahl's law reminds that for high performance designs we cannot ignore the speed of write operations.

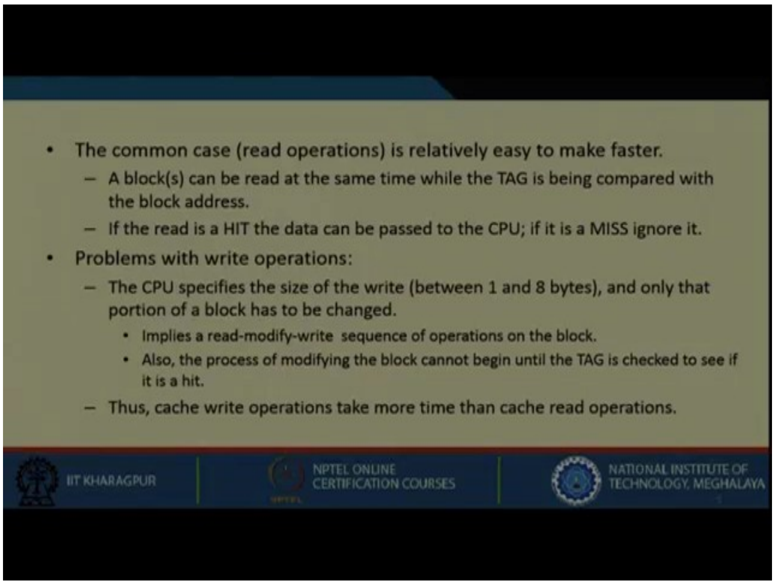
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us now try to answer our fourth and last question that is what happens on a write. Now let us see some statistical analysis. Statistical data suggests that the read operation including instruction fetches dominate processor cache accesses. All instruction fetch operations are read, most instruction do not write to memory; that means, out of the total memory accesses most are reads and less are writes.

We can optimize the read operation of the memory; again think of making the common case fast. So, we optimize cache accesses for reads. For all the read operations the cache can be optimized, but we need to remember from Amdahl's law that we cannot ignore the speed of write operations as well.

So, what we try to do is that we make the common case fast. By doing so, we are making read operations faster, but it does not mean we will leave out the write operations. The write operations are performed with less frequency than read, but still we must take care of that as well. The common case, that is the read operation, is relatively easy to make faster.

(Refer Slide Time: 07:22)



- The common case (read operations) is relatively easy to make faster.
  - A block(s) can be read at the same time while the TAG is being compared with the block address.
  - If the read is a HIT the data can be passed to the CPU; if it is a MISS ignore it.
- Problems with write operations:
  - The CPU specifies the size of the write (between 1 and 8 bytes), and only that portion of a block has to be changed.
    - Implies a read-modify-write sequence of operations on the block.
    - Also, the process of modifying the block cannot begin until the TAG is checked to see if it is a hit.
  - Thus, cache write operations take more time than cache read operations.

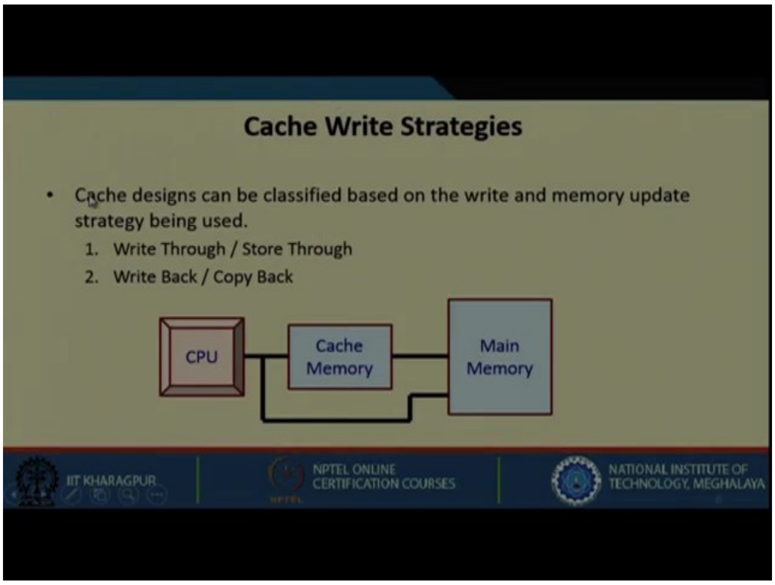
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

We can make this read operation faster; a block or blocks can be read at the same time while the TAG is being compared with the block address. If the read is a hit, the data can be passed to the CPU; if it is a miss we can ignore it.

What problems do we have with write operations? The CPU specifies the size of the write, and only that portion of the block has to be changed. What does this imply? This implies a read-modify-write; that means, we read it, then we modify it, and then again we write it back. Also the process of modifying the block cannot begin until TAG is checked to see if it is a hit.

So, in read operation while we are checking we are simultaneously reading it, but here it is not possible. Unless and otherwise there is a match, we are not doing any operation. Thus we can say that cache write operations take more time than cache read operation.

(Refer Slide Time: 09:45)



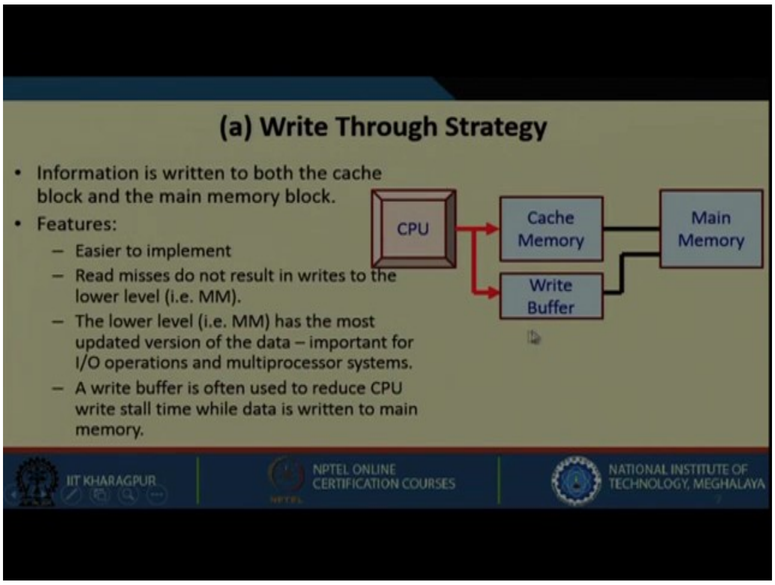
Cache Write Strategies

- Cache designs can be classified based on the write and memory update strategy being used.
  1. Write Through / Store Through
  2. Write Back / Copy Back

CPU
Cache Memory
Main Memory
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Cache design can be classified based on the write and memory update strategy being used. One is write through or store through, another is write back or copy back. We will see both the methods in detail. First let us understand what write through means; we are saying we will be updating directly in main memory. We will write in cache memory and main memory simultaneously, and write back means we will update something in cache memory for the time being and later when that particular block need to be replaced we will update in main memory.

(Refer Slide Time: 10:42)



(a) Write Through Strategy

- Information is written to both the cache block and the main memory block.
- Features:
  - Easier to implement
  - Read misses do not result in writes to the lower level (i.e. MM).
  - The lower level (i.e. MM) has the most updated version of the data – important for I/O operations and multiprocessor systems.
  - A write buffer is often used to reduce CPU write stall time while data is written to main memory.

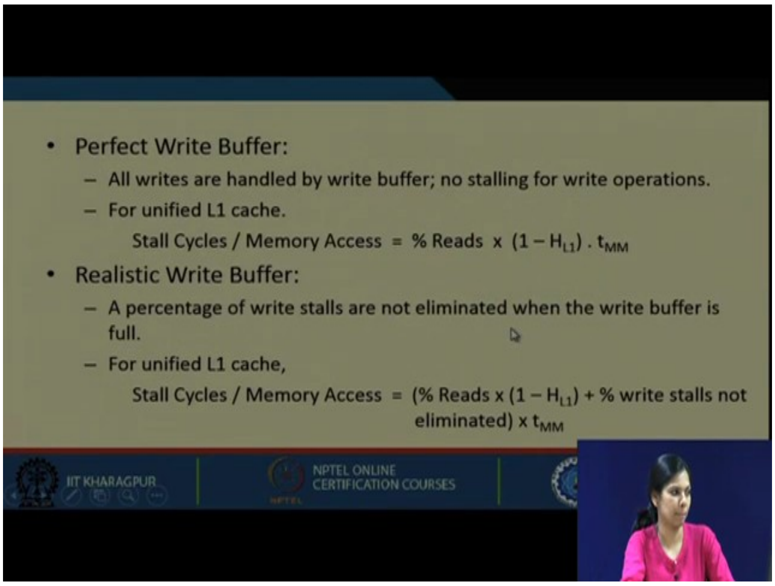
CPU
Cache Memory
Write Buffer
Main Memory
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us see write through strategy. Here the information is written to both the cache block and the main memory block. We are writing the information in both cache memory as well as in main memory. This is easy to implement as we need not have to think of any other details; we are simply updating using write through strategy by updating the cache memory as well as the main memory. The important feature is it is very much easier to implement and read misses do not result in writes to the lower level, whenever there is a read miss it do not result in writes to the lower level of the memory, that is the main memory.

The lower level has the most updated version of the data, and this is also very important both for IO operation and in multiprocessor. Because in case of IO operation as well as for multiprocessor system as we are updating both cache memory and main memory at the same time, so main memory is always updated. A write buffer is often used to reduce CPU write stall time when data is written to main memory.

So, when we write data into main memory we can have a write buffer and what this write buffer will do? Instead of directly writing into main memory, write buffer will be faster. We will be writing into this write buffer, and in turn the write buffer will be putting in to the main memory. So, we can often reduce the CPU write stall time if we have a write buffer in place.

(Refer Slide Time: 12:56)



- Perfect Write Buffer:
  - All writes are handled by write buffer; no stalling for write operations.
  - For unified L1 cache.

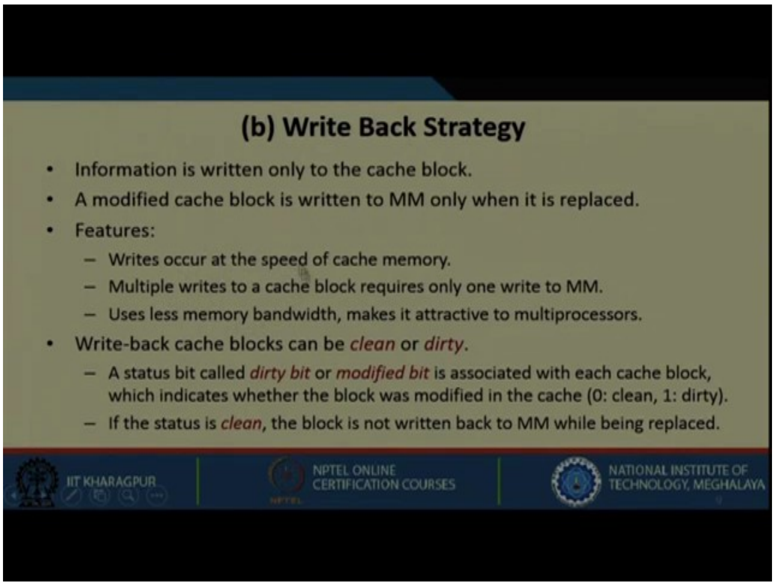
    Stall Cycles / Memory Access = % Reads x $(1 - H_{L1}) . t_{MM}$
- Realistic Write Buffer:
  - A percentage of write stalls are not eliminated when the write buffer is full.
  - For unified L1 cache,

    Stall Cycles / Memory Access = (% Reads x $(1 - H_{L1})$ + % write stalls not eliminated) x t_{MM}

If we have perfect write buffer, all writes are handled by the write buffer; no stalling for write operations is required. But there can be situation where we do not have this perfect write buffer rather we have a realistic write buffer.

In case of realistic write buffer what we generally do, we actually write it at the speed of this buffer; and this buffer size cannot be very much larger. So, a percentage of write stall are not eliminated when the write buffer is full. So, it might happen we are writing more number of words and the write buffer is full in that case we cannot write. Some of the writes can only be performed when this buffer is empty again. So, in that case it cannot be take in care of.

(Refer Slide Time: 14:09)



(b) Write Back Strategy

- Information is written only to the cache block.
- A modified cache block is written to MM only when it is replaced.
- Features:
  - Writes occur at the speed of cache memory.
  - Multiple writes to a cache block requires only one write to MM.
  - Uses less memory bandwidth, makes it attractive to multiprocessors.
- Write-back cache blocks can be *clean* or *dirty*.
  - A status bit called *dirty bit* or *modified bit* is associated with each cache block, which indicates whether the block was modified in the cache (0: clean, 1: dirty).
  - If the status is *clean*, the block is not written back to MM while being replaced.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

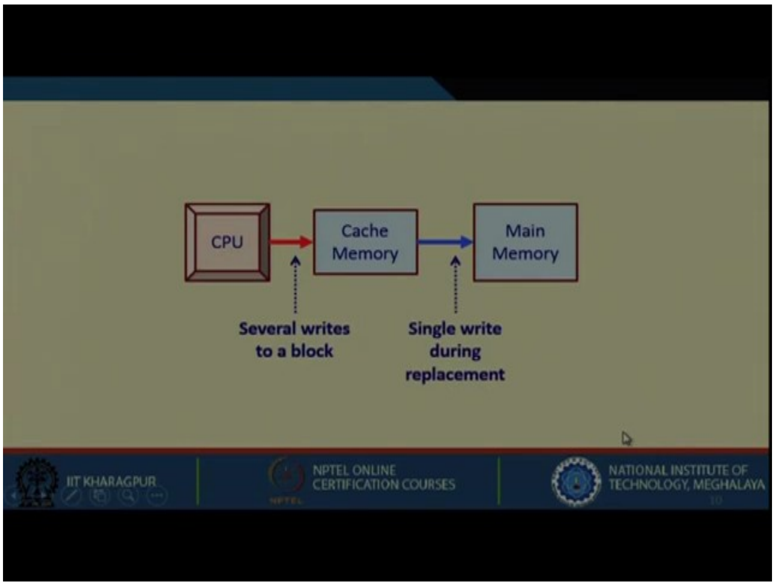
So, there are two things perfect write buffer and realistic write buffer. Now coming to write back strategy, in this case information is written only to the cache block. A modified cache block is written to main memory only when it is replaced. So, the writing is happening at the speed of cache and we are not updating the main memory. So, when we will be updating the main memory? The main memory will get updated when we want to replace that particular block from the cache memory. In that case we need to have some mechanism through which we know that whether this particular block has been updated or it has been not updated.

We can see write occurs at the speed of cache memory; multiple writes to the cache block requires only one write to the main memory. That means, even if we are writing multiple times to the cache block, but at the end only one time it has to be written into the main memory. This is because we are updating not in the main memory, we are updating in the cache block repeatedly; if the block is written repeatedly at end it is required to be written back into the main memory.

So, it uses less memory bandwidth and makes it attractive for multiprocessor. Write back cache blocks can be clean or dirty. What do we mean by clean and dirty? There is a status bit called dirty bit or modified bit with each cache block. It indicates whether the block was modified in the cache or not. If it has been modified then it is marked as 1, that is, it is dirty. If it is not modified it is clean, we mark it as 0. If the status is clean the block is not written back to the main memory while it is being replaced.

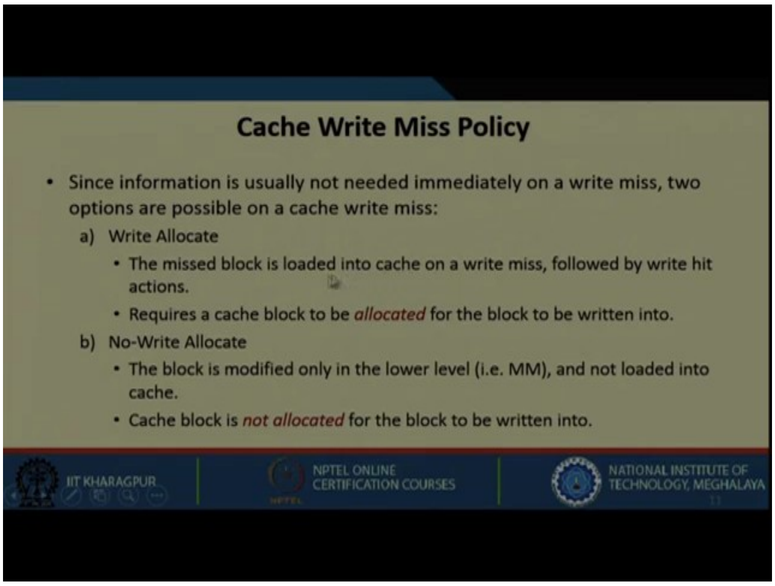
So, in a cache block in write back strategy we update a cache block and we keep a status bit associated with it. If that status bit is 1 meaning that particular block when it is residing in the cache memory has been updated then that change should be reflected in main memory. While it is replaced it has to be changed in main memory. Likewise if it is clean; that means, it need not have to be written back into the main memory, because that block has been brought from main memory to cache memory, but it has not been updated. So, no write back to main memory is required.

(Refer Slide Time: 18:18)


CPU
Cache Memory
Main Memory
Several writes to a block
Single write during replacement
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

What happens between CPU and cache memory? Multiple writes take place between this, but between cache memory and main memory only one time it is written, when the cache block is replaced and new block is brought in.

(Refer Slide Time: 18:45)



Cache Write Miss Policy

- Since information is usually not needed immediately on a write miss, two options are possible on a cache write miss:
  - a) Write Allocate
    - The missed block is loaded into cache on a write miss, followed by write hit actions.
    - Requires a cache block to be *allocated* for the block to be written into.
  - b) No-Write Allocate
    - The block is modified only in the lower level (i.e. MM), and not loaded into cache.
    - Cache block is *not allocated* for the block to be written into.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA
11

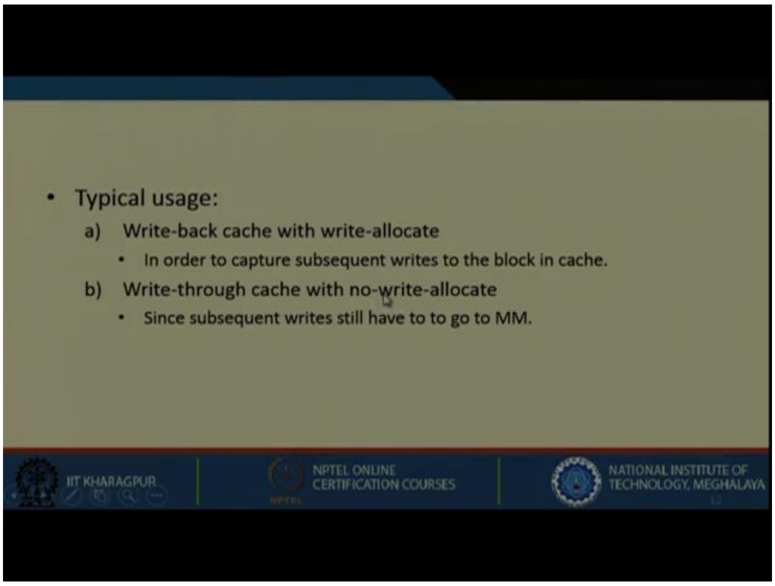
So, let us see some cache write miss policy. When we write into cache what is the cache write miss policy? Since information is usually not needed immediately on a write miss, two option are possible on a cache write miss. When we want to write the cache and a miss occurs what policy can be adopted? First one is write allocate; that means, the missed block is loaded into the cache on a write miss followed by write hit action.

Suppose we write a block into cache, but that block is not present. Write allocate means the missed block is first read and it is brought into the cache, and then it is returned into the cache on a write means followed by a write hit action then write is performed on the that. It requires a cache block to be allocated for the block to be written into. If the cache is not full then it is fine the cache block can be allocated for this purpose. If let us say the cache is full we have to replace one of the blocks from here and put it into the main memory, and then only you can bring the next block and you can go ahead in write allocate.

So, this is what is done. It requires a cache block to be allocated for the block to be written. In such cases if the cache is full then a particular block from the cache needs to be first replaced, and then this block needs to be brought in for write operation. The next one is no write allocate. In no write allocate the block is modified only in the lower level that is main memory, and not loaded into the cache. No write allocate means we are not allocating any space here. No allocation is done whenever there is a write miss we directly writing into the main memory, and we are not bringing it to the cache memory and then writing it back.

So, cache block is not allocated for the block to be written into. In this case a cache block was allocated, but here there is no need for allocation because the block is modified only in the main memory that is the lower level, and not loaded into the cache. In no write allocate whenever a cache miss occurs these are the two ways to which we can perform it.

(Refer Slide Time: 22:04)



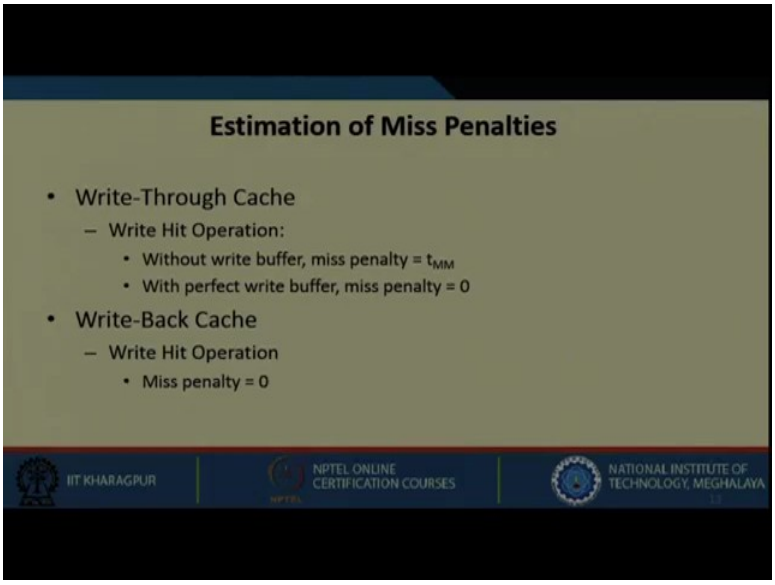
- Typical usage:
  - a) Write-back cache with write-allocate
    - In order to capture subsequent writes to the block in cache.
  - b) Write-through cache with no-write-allocate
    - Since subsequent writes still have to to go to MM.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us see the typical usage of write back cache with write allocate. Write back means, we will write back later we will update most of the time in cache memory and finally, when the block needs to be replaced we will be writing it back to the main memory. So, in write back cache with write allocate.

So, whenever we are doing write through meaning we are updating directly in main memory, and we are not requiring any space. So, no space in the cache is actually required since subsequent write still have to go to main memory. So, all the write has to be in the main memory. Of course, the access time will be more, but the writing is directly on the main memory.

(Refer Slide Time: 23:54)



Estimation of Miss Penalties

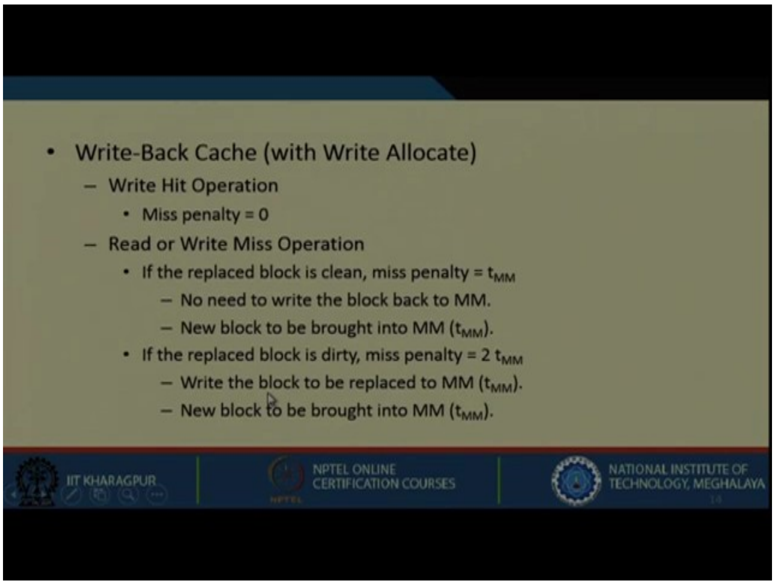
- Write-Through Cache
  - Write Hit Operation:
    - Without write buffer, miss penalty = t_{MM}
    - With perfect write buffer, miss penalty = 0
- Write-Back Cache
  - Write Hit Operation
    - Miss penalty = 0

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Let us now see the estimation of miss penalties for write operation. When it is write through cache, write hit operation without write buffer what will be the miss penalty. Without any write buffer the miss penalty will be access time of main memory only. So, whenever a write hit occurs we are updating directly into the main memory. And with perfect write buffer, there will be no miss penalty because we are writing into the buffer, and the buffer is taking care of that. So, the miss penalty becomes zero here.

In write back cache when write hit operation happens, the miss penalty is zero because we are doing it in the cache itself and the data is present in the cache.

(Refer Slide Time: 25:17)



- Write-Back Cache (with Write Allocate)
  - Write Hit Operation
    - Miss penalty = 0
  - Read or Write Miss Operation
    - If the replaced block is clean, miss penalty = t_{MM}
      - No need to write the block back to MM.
      - New block to be brought into MM (t_{MM}).
    - If the replaced block is dirty, miss penalty = 2 t_{MM}
      - Write the block to be replaced to MM (t_{MM}).
      - New block to be brought into MM (t_{MM}).

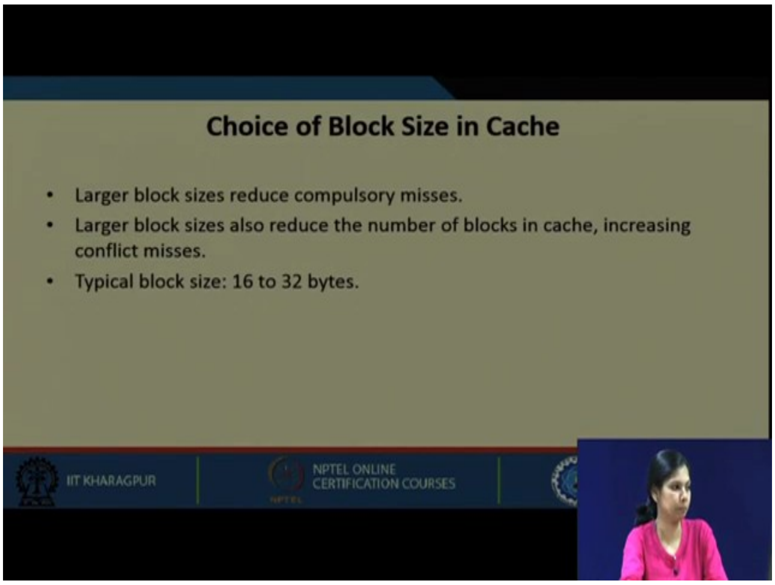
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

Now, let us see write back cache with write allocate. Here when write hit operation happens, the miss penalty will be zero. For read or write miss operation what happens? If the replaced block is clean; miss penalty is tMM; no need to write the block to main memory, and new block to be brought into main memory. So, basically we need not have to write the block into the main memory only a new block will be brought from main memory. So, we are accessing tMM once only.

Now if the replaced block is dirty; miss penalty will be 2 tMM. So, you are writing it to main memory the time will be tMM, a new block to be brought into main memory that will be again tMM. So, twice time of main memory will be required.

So, we have seen so many cases what happens when a write hit occurs, what happens in a write miss occurs, both in case of write through and write back.

(Refer Slide Time: 27:32)



Choice of Block Size in Cache

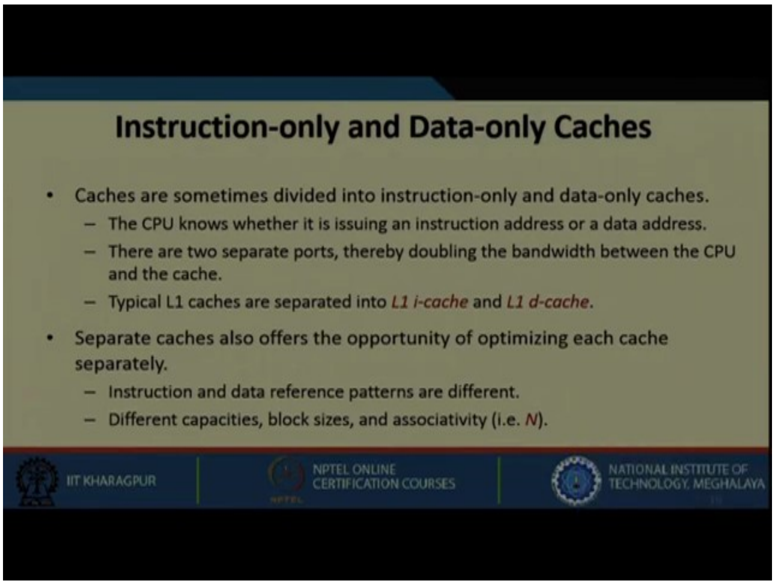
- Larger block sizes reduce compulsory misses.
- Larger block sizes also reduce the number of blocks in cache, increasing conflict misses.
- Typical block size: 16 to 32 bytes.

Regarding choice of block size in cache, larger block size reduced compulsory misses. If you have larger block size then the compulsory misses can be reduced, and larger block size also reduces the number of blocks in cache increasing the conflict misses.

Typical block size is 16 to 32 bytes.

Next is instruction only and data only caches.

(Refer Slide Time: 28:20)



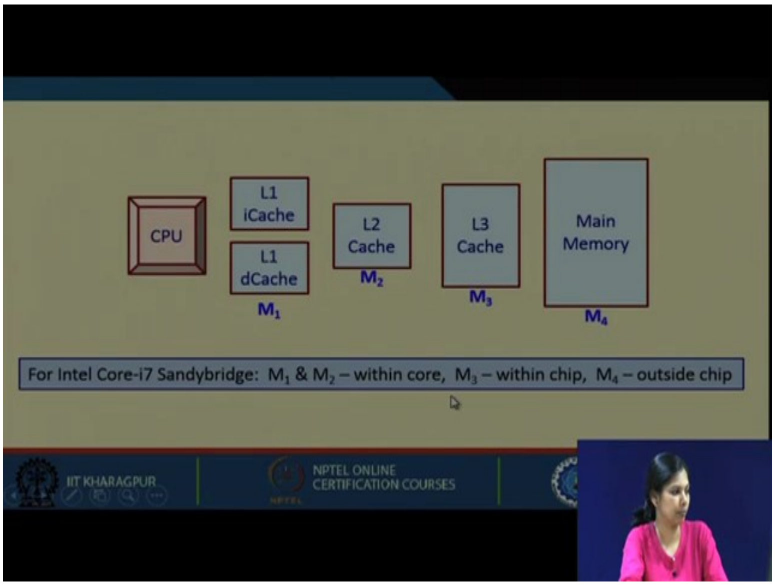
Instruction-only and Data-only Caches

- Caches are sometimes divided into instruction-only and data-only caches.
  - The CPU knows whether it is issuing an instruction address or a data address.
  - There are two separate ports, thereby doubling the bandwidth between the CPU and the cache.
  - Typical L1 caches are separated into *L1 i-cache* and *L1 d-cache*.
- Separate caches also offers the opportunity of optimizing each cache separately.
  - Instruction and data reference patterns are different.
  - Different capacities, block sizes, and associativity (i.e. *N*).

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NATIONAL INSTITUTE OF TECHNOLOGY, MEGHALAYA

We have already seen that for instruction only and data only caches, the CPU knows whether it is issuing an instruction address or a data address; there are two separate ports thereby doubling the bandwidth between the CPU and the cache. Typical L1 caches are separated into L1 I-cache and L1 D-cache. Separate caches also offer the opportunity of optimizing each cache separately, now we know that read operations are more. So, the instruction cache can be improved in a different manner. And data references patterns are also different the way the instruction are different and way the pattern are different. So, different capabilities, block sizes and associativity can be implemented for two different kinds of caches.

(Refer Slide Time: 29:33)



For Intel Core-i7 Sandybridge: M_1 & M_2 – within core, M_3 – within chip, M_4 – outside chip

This is for an Intel core-i7 SandyBridge, where M1 and M2 are within the core, and and this is outside the chip. Main memory is always outside the chip.

So, we come to the end of lecture 31. In the next lecture we will be looking into some of the strategies to improve the cache performance further.

Thank you.