CS147 - Lecture 21

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Handling Cache Miss

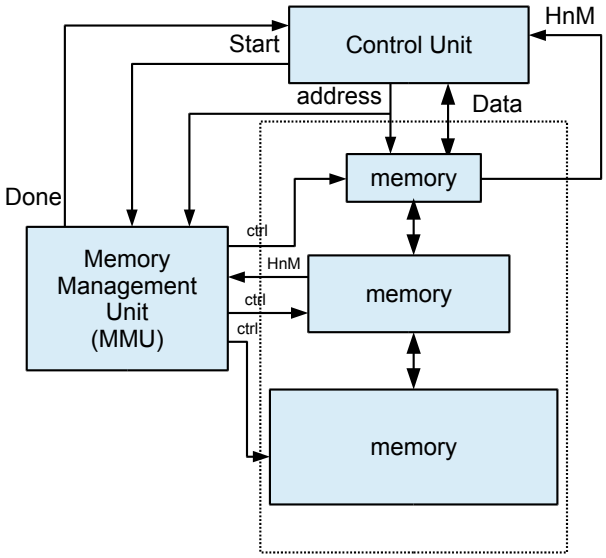
Reference Books / Source:

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*

Handling Cache Miss ...

2

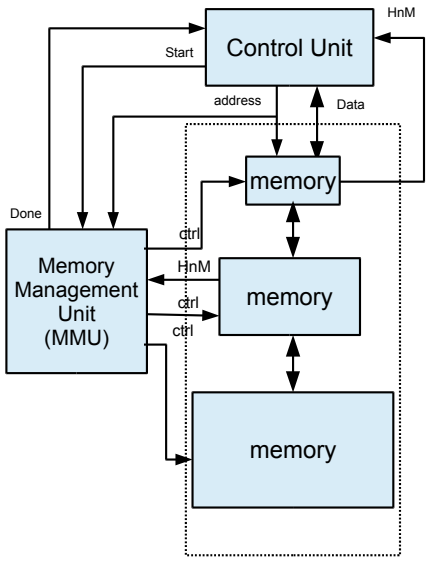
Cache – Memory Management Unit



- Before start with cache miss, let's review the component and process involved in transferring blocks between memory hierarchy. One way is to use interrupt service mechanism (where current process state and register is saved and processor jumps to a interrupt service code to accomplish requested interrupt task, which would be to orchestrate the data transfer between memory hierarchy). However, interrupt mechanism has disadvantage of save/restore mechanism for the process states apart from the fact that we are keeping the processor busy without using its processing power.

- The usual was is to use a separate specialized module memory management unit (MMU) just for the purpose of synchronization of memory content through memory hierarchy. Once processor receives a cache miss signal, it issues a data transfer request to MMU with the given address of the missed information. Once, issued processor can continue with the execution of other instructions in the pipeline till the time it can not any longer avoid a stall. In general, processor faces stall upon a cache miss. The MMU takes the target address for replacing the cache block and initiate the block transfer from lower level memory into higher level memory. If there are multiple level, there may be miss in the intermediate hierarchy level and MMU will receive that miss and start bringing back data from the further lower level memory into the intermediate level. Once the cache line is replaced with required block, MMU issues a done signal which unfreeze the processor to carry on its work.

Handling Cache Miss - Read



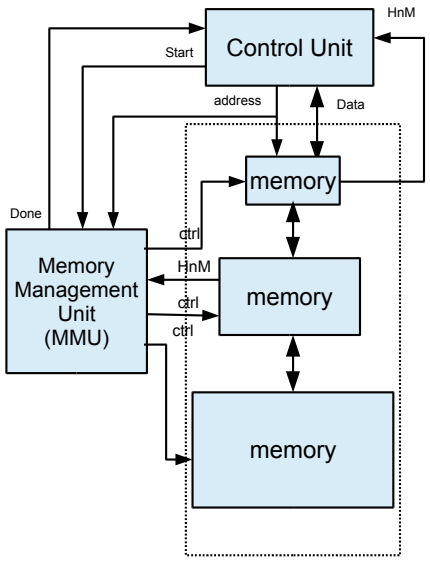
- Let's start with simple scenario
  - On a read miss, bring back the block from lower memory.
- Steps
  - Control unit detect a miss
  - Control unit request MMU to replace the cache line.
  - MMU starts transferring the blocks into higher memory.
    - Control unit is stalled
  - Once done, MMU issues done signal to control unit.
  - Control unit starts again from the memory read operation.

4

- A read operation is simple to handle. If it is a hit there is nothing special to do. Data is back from the cache. If it is a miss, control unit request MMU to transfer data from lower memory to cache and goes to stall (or to execute other instruction in pipe line). Once the block transfer is done, sends a transfer completion status to processor. The processor then resume its execution of the current read operation.

- Hardware threading exploits this performance loop hole (where processor must freeze / stall) and keeps processor going by executing instructions from other hardware threads.

Handling Cache Write



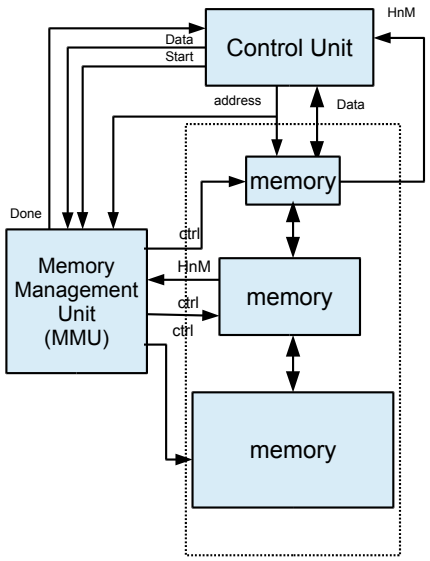
- Write is little bit tricky
  - It incurs memory inconsistency.
- Policies
  - **Write Through**
    - Inconsistency is resolved at write operation itself.
  - **Write Back**
    - Inconsistency is resolved at block replacement time.

5

• Write operation is trickier that read operation. A write causes inconsistency between memory hierarchy. In a memory hierarchy, an address may point to multiple location in the hierarchy. It is very important that each such location contains same information, otherwise operation may give rise to inconsistent result. For example if code i=10 writes value 10 in the corresponding cache line location, it is important to have same information down into the lower level memory. Otherwise, if the corresponding cache line replaced and then again reloaded with variable 'i' the latest value may be lost forever, therefore any further operation involving this variable will produce wrong result.

• There are two strategies to resolve such inconsistency occurred due to write operation. One is 'write through' where any data written in cache also written to corresponding lower level memory at the same time. The other strategy is 'write back' where a cache write marks the cache line for future transfer into lower level memory upon replacement. The idea behind write back is to delay transfer of the block from higher to lower level as much as possible. If the cache line is never replaced, then it is OK to keep the inconsistency – data referenced from that specific cache line will be served right from the cache. Therefor processor will receive right data always. Only if the corresponding cache line needs replacement, we can store the block from cache to lower level memory to maintain the consistency.

Handling Cache Write – Write through

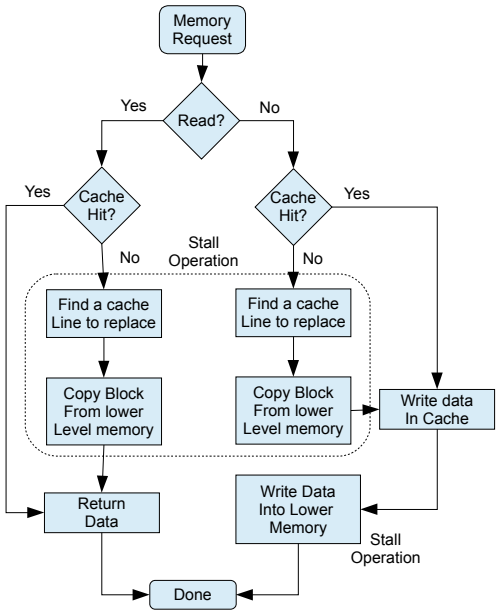


- Steps
  - Control unit detect a miss
  - Control unit request MMU to replace the cache line.
  - MMU starts transferring the blocks into higher memory.
    - Control unit is stalled
  - Once done, MMU issues done signal to control unit.
  - Control unit writes to the cache block
  - Control unit request MMU to write the same data into the lower level memory.
    - Control unit is stalled
  - Once done, MMU issues done signal to control unit.
  - Control unit starts again after memory write stage.

6

- The initial steps in write through strategy is similar to read operation. If there is a miss, corresponding block needs to be brought into the cache. Control unit is usually stalled during this operation. Once the block transfer is done, control unit writes the data into the cache and it requests MMU to write the same data into the lower level memory. Control unit is also stalled during this operation to make sure that the required data is written in the lower level memory. Once the write is done, controller starts the normal operation.

Handling Cache Write – Write through



- Advantage
  - Simple strategy → Simple circuit.
  - No impact on read operation.
- Disadvantage
  - Control unit is stalled twice → high impact on performance.
- Notice
  - Both read and write has cache line replacement.
  - Second stall is coming from expensive write operation at lower memory.

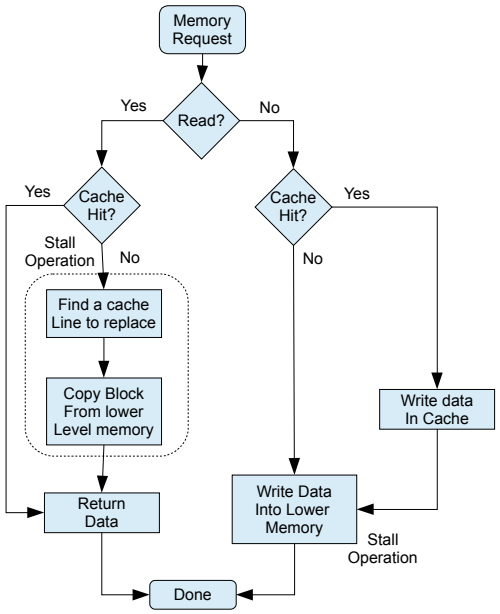
7

• The advantage of write through strategy is that is simple, which will lead into simpler circuit. Also there is no impact on read operation – meaning read operation functionality will remain as it is as we have discussed. The disadvantage is that the control unit is stalled twice which will impact processor performance severely.

• At this point we can put the steps for read and write together into a flow chart. From the flow chart, we can observe two important aspect.

- Both the read and write has the cache line replacement step – which causes stall.
- Second stall in write operation is coming from expensive write operation at the lower level memory.

Handling Cache Write – Write through



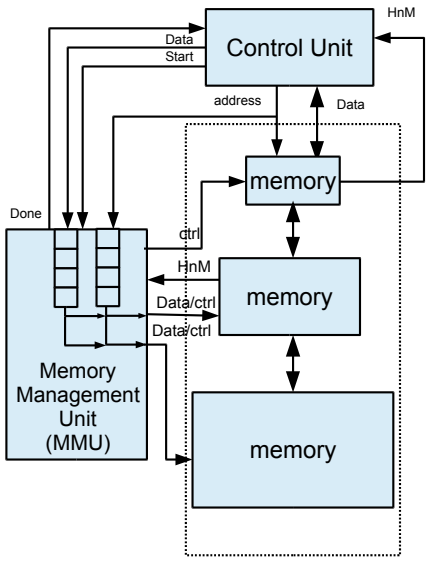
- Improvement
  - Take out the cache line replacement on write miss.
  - Modified block will be loaded on next cache read miss.
- Notice
  - Cache hit on write still incur controller stall.
    - Can we do anything?

8

- We can possibly takeout the cache line replacement step in the write operation with a cache miss and directly write the data into lower level memory. This will not incur any memory inconsistancy, since the cache line is not present incase of a miss. Once a read miss for the corresponding block is occurred on read, it will load the previously modified block (by write operation with miss) into cache line.

- After the above operational modification, we are still having one stall and that too at the write hit condition. In contrast, a read hit does not incur any stall.

Handling Cache Write – Write through



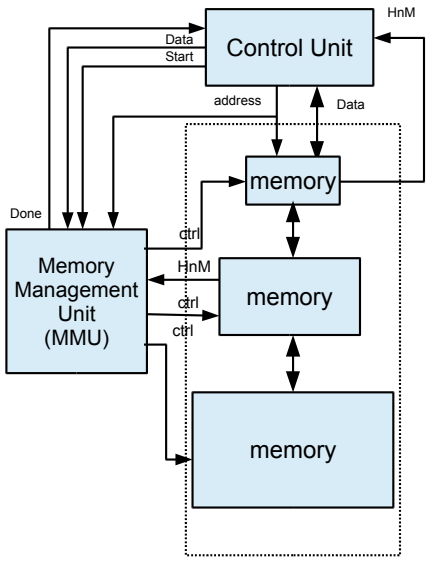
- Addition on write buffers will solve the stall issue.
  - This is a queue structure implemented in hardware.
- Controller deposit the write information (data and address) onto write buffers and continue its work.
  - Only if the write buffer queue is full, stall happens.

9

• Introduction of write buffer into the MMU should solve the problem to some extent. A write buffer is a queue structure (FIFO) implemented in hardware. Control unit can store data and corresponding address into write buffer of MMU. Once written into the write buffer, controller can resume its normal operation. Being a part of the processor, write operation to write buffer by control unit is very fast and thus it can resume its normal operation without any significant stall.

• However, size of write buffer is not unlimited. Therefor if a program has very frequent occurrence of write operation, it may fill up the buffer soon. If the write buffer is full, controller needs to stall the processor.

Handling Cache Write – Write back



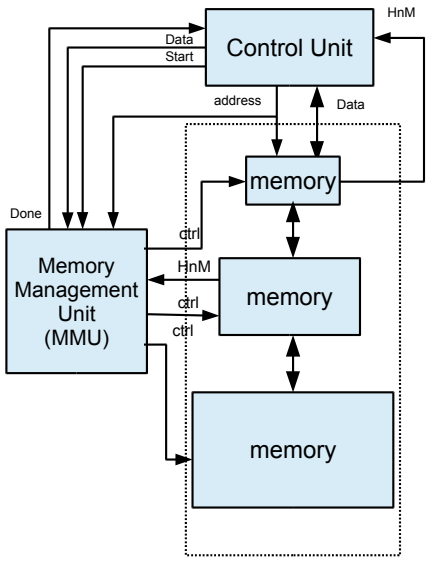
- Steps
  - Control unit detect a miss
  - Control unit request MMU to replace the block
  - MMU starts transferring the blocks into higher memory.
    - Control unit is stalled
  - Once done, MMU issues done signal to control unit.
  - Control unit writes to the cache block
  - The cache line is marked 'dirty' using one of the control bit.
    - 'll be written back to lower memory at replacement.
- Problem
  - What if the cache line is already dirty?

10

• In a write back strategy, if there is a miss, the corresponding block is fetched into cache first (just like a read miss). Once the address is present in the cache, data write happens in the cache, but the lower memory is not written at the same time. This memory inconsistency is kept till it is time to replace the corresponding cache line. At the point of replacing a cache line, if there is new data in that cache line, the cache line is written back into lower level memory first. To mark if there is a new data in a cache line, one more bit is introduced in the cache line's control bit. This control bit is called 'dirty bit'. Upon a write operation in a cache line, the dirty bit is turned on, to mark that this cache line needs to be stored back at replacement time. This strategy of delaying the memory inconsistency resolution works fine because, as long as the dirty cache line is present in the cache and the data is referenced for the read operation, it will return the latest value (because it is present in the cache). Only when the cache line is replaced, the line is copied back into the lower level memory to resolve the memory inconsistency.

• However, this strategy of write back at replacement affects the read operation as well (unlike the write through strategy). Now, if there is an replacement of a dirty cache line, the line must be copied into lower level memory first before another block is copied back into that cache line.

Handling Cache Write – Write back

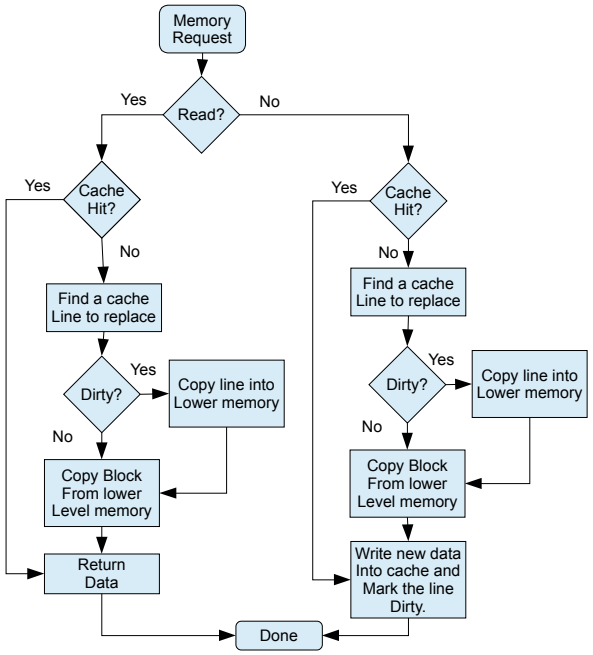


- Steps
  - Control unit detect a miss
  - Control unit request MMU to replace the block
  - **If the cache line is dirty, write it back.**
    - Control unit is stalled.
  - MMU starts transferring the blocks into higher memory.
    - Control unit is stalled
  - Once done, MMU issues done signal to control unit.
  - Control unit writes to the cache block
  - The cache line is marked 'dirty' using one of the control bit.
    - 'll be written back to lower memory at replacement.

11

- Therefore we must add this cache write back step on the cache line replacement step with the write through strategy.

Handling Cache Write – Write back



- Cache line replacement at read also needs to write back the line if the line is dirty.
- Advantage
  - Write operation is faster, especially if write operation occurs faster than write can be handled by lower level memories.
- Disadvantage
  - Complex implementation at digital circuit level.

12

• Write operation is faster with write back strategy, even if a cache miss (for both read and write) will incur two block transfer between adjacent hierarchy (one for write back from higher to lower memory and other for lower to higher memory). Especially when the occurrence of write operation in a program is faster than the write operation that can be handled at lower level memory. The advantage comes from two places – temporal locality and spatial locality. If an address is written, it is likely to be referenced or written in near future (e.g. index variable in a for loop 'for(i=0;i<1000; i++)'. Moreover, if an address is written at one time, possibly its neighboring addresses in the same cache line will be referenced or written in near future.

CS147 - Lecture 21

Kaushik Patra
(kaushik.patra@sjsu.edu)

13

- Handling Cache Miss

Reference Books / Source:

1) *Chapter 4 of 'Computer Organization & Architecture' by Stallings*
2) *Chapter 7 of 'Computer Organization & Design' by Patterson and Hennessy/*