Advanced Computer Architecture

Lecture No. 39

Reading Material

Vincent P. Heuring & Harry F. Jordan	Chapter 7
Computer Systems Design and Architecture	7.4, 7.5

Summary

- Cache Organization and Functions
- Cache Controller Logic
- Cache Strategies

Cache Organization and Functions:

The working of the cache is based on the principle of locality which has two aspects.
Spatial Locality: refers to the fact when a given address has been referenced, the next address is highly probable to be accessed within a short period of time.
Temporal Locality refers to the fact that once a particular data item is accessed, it is likely that it will be referenced again within a short period of time.
To exploit these two concepts, the data is transferred in blocks between cache and the main memory. For a request for data, if the data is available in the cache it results in a cache hit. And if the requested data is not present in the cache, it is called a cache miss. In the given example program segment, spatial locality is shown by the array ALPHA, in which next variable to be accessed is adjacent to the one accessed previously. Temporal locality is shown by the reuse of the loop variable 100 times in For loop instruction.

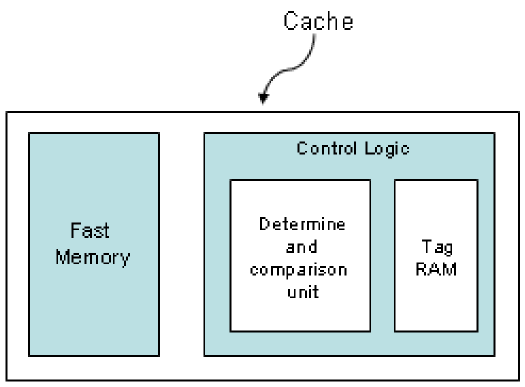
```
Int ALPHA [100], SUM;
SUM=0;
For (i=0; i<100; i++)
{SUM= SUM+ALPHA[i];}
```

Cache Management

To manage the working of the cache, cache control unit is implemented in hardware, which performs all the logic operations on the cache. As data is exchanged in blocks between main memory and cache, four important cache functions need to be defined.

- Block Placement Strategy
- Block Identification
- Block Replacement
- Write Strategy

Block Diagram of a Cache System



In the figure, the block diagram of a system using cache is shown. It consists of two components.

- Fast Memory
- Control Logic Unit

Control logic is further divided into two parts.

Determine and Comparison Unit: For determining and comparisons of the different parts of the address and to evaluate hit or miss.

Tag RAM: Second part consists of tag memory which stores the part of the memory address (called tag) of the information (block) placed in the data cache. It also contains additional bits used by the cache management logic.

Data Cache: is a block of fast memory which stores the copies of data and instructions frequently accessed by the CPU.

Cache Strategies

In the next section we will discuss various cache functions, and strategies used to implement these functions.

Block Placement

Block placement strategy needs to be defined to specify where blocks from main memory will be placed in the cache and how to place the blocks. Now various methods can be used to map main memory blocks onto the cache .One of these methods is the associative mapping explained below.

Associative Mapping:

In this technique, block of data from main memory can be placed at any location in the cache memory. A given block in cache is identified uniquely by its main memory block number, referred to as a tag, which is stored inside a separate tag memory in the cache. To check the validity of the cache blocks, a valid bit is stored for each cache entry, to verify whether the information in the corresponding block is valid or not.

Main memory address references have two fields.

- The word field becomes a “cache address” which specifies where to find the word in the cache.
- The tag field which must be compared against every tag in the tag memory.

Associative Mapping Example

Refer to Book Ch.7 Section (7.5) Figure 7.31(page 350-351) for detailed explanation.

Mechanism of the Associative Cache Operation

For details refer to book Ch.7, Section 7.5, Figure 7.32 (Page 351-352).

Direct Mapping

In this technique, a particular block of data from main memory can be placed in only one location into the cache memory. It relies on principle of locality.

Cache address is composed of two fields:

- Group field
- Word field

Valid bit specifies that the information in the selected block is valid.
For a direct mapping example, refer to the book Ch.7, Section 7.5, Figure 7.33 (page 352 – 353).

Logic Implementation of the Controller for Direct Mapping
Logic design for the direct mapping is simpler as compared to the associative mapping. Only one tag entry needs to be compared with the part of the address called group field.

Tasks Required For Direct Mapping Cache:
For details refer to the book Ch. 7, Section 7.5, Figure 7.34 (Page 353-354).

Cache Design: Direct Mapped Cache
To understand the principles of cache design, we will discuss an example of a direct mapped cache.
The size of the main memory is 1 MB. Therefore 20 address bits needs to be specified. Assume that the block size is 8 bytes. Cache memory is assumed to be 8 KB organized as 1 K lines of cache memory. Cache memory addresses will range from 0 up to 1023. Now we have to specify the number of bits required for the tag memory. The least significant three bits will define the block. The next 10 bits will define the number of bits required for the cache. The remaining 7 bits will be the width of the tag memory.
Main memory is organized in rectangular form in rows and columns. Number of rows would be from 0 up to 1023 defined by 10 bits. Number of rows in the main memory will be the same as number of lines in the cache. Number of columns will correspond to 7 bits address of the tag memory. Total number of columns will be 128 starting from 0 up to 127. With direct mapping, out of any particular row only one block could be mapped into the cache. Total number of cache entries will be 1024 each of 8 bytes.

Advantage:
Simplicity
Disadvantage:
Only a single block from a given group is present in cache at any time. Direct map Cache imposes a considerable amount of rigidity on cache organization.

Set Associative Mapping
In this mapping scheme, a set consisting of more than one block can be placed in the cache memory.
The main memory address is divided into two fields. The Set field is decoded to select the correct group. After that the tags in the selected groups are searched. Two possible places in which a block can reside must be searched associatively. Cache group address is the same as that of the direct-mapped cache.
For details of the Set associative mapping example, refer to the book Ch.7, Section 7.5, Figure 7.35 (Page 354-355).

Replacement Strategy
For a cache miss, we have to replace a cache block with the data coming from main memory. Different methods can be used to select a cache block for replacement.

Always Replacement: For Direct Mapping on a miss, there is only one block which needs replacement called always replacement.
For associative mapping, there are no unique blocks which need replacement .In this case there are two options to decide which block is to be replaced.

- **Random Replacement**: To randomly select the block to be replaced
- **LFU:** Based on the statistical results, the block which has been least used in the recent past, is replaced with a new block.

Write Strategy
When a CPU command to write to a memory data will come into cache, the writing into the cache requires writing into the main memory also.
Write Through: As the data is written into the cache, it is also written into the main memory called Write Through. The advantages are:

- Read misses never result in writes to the lower level.
- Easy to implement than write back

Write Back: Date resides in the cache, till we need to replace a particular block then the data of that particular block will be written into the memory if that needs a write, called write back. The advantages are:

- Write occurs at the speed of the cache
- Multiple writes with in the same block requires only one write to the lower memory.
- This strategy uses less memory bandwidth, since some writes do not go to the lower level; useful when using multi processors.

Cache Coherence
Multiple copies of the same data can exist in memory hierarchy simultaneously. The Cache needs updating mechanism to prevent old data values from being used. This is the problem of cache coherence. Write policy is the method used by the cache to deal with and keep the main memory updated.
Dirty bit is a status bit which indicates whether the block in cache is dirty (it has been modified) or clean (not modified). If a block is clean, it is not written on a miss, since lower level contains the same information as the cache. This reduces the frequency of writing back the blocks on replacement.
Writing the cache is not as easy as reading from it e.g., modifying a block can not begin until the tag has been checked, to see if the address is a hit. Since tag checking can not occur in parallel with the write as is the case in read, therefore write takes longer time.
Write Stalls: For write to complete in Write through, the CPU has to wait. This wait state is called write stall.
Write Buffer: reduces the write stall by permitting the processor to continue as soon as the data has been written into the buffer, thus allowing overlapping of the instruction execution with the memory update.
Write Strategy on a Cache Miss
On a cache miss, there are two options for writing.

Write Allocate: The block is loaded followed by the write. This action is similar to the read miss. It is used in write back caches, since subsequent writes to that particular block will be captured by the cache.
No Write Allocate: The block is modified in the lower level and not loaded into the cache. This method is generally used in write through caches, because subsequent writes to that block still have to go to the lower level.