Computer Organization and Architecture

Cache Memory

Veena Thenkanidiyoor
National Institute of Technology Goa

1

1

Recap

- Memory unit
- Semiconductor memory cells
  - SRAM
  - DRAM
- Internal organization of memory
- SRAM chip, SRAM module
- DRAM, SDRAM, DDR SDRAM

2

2

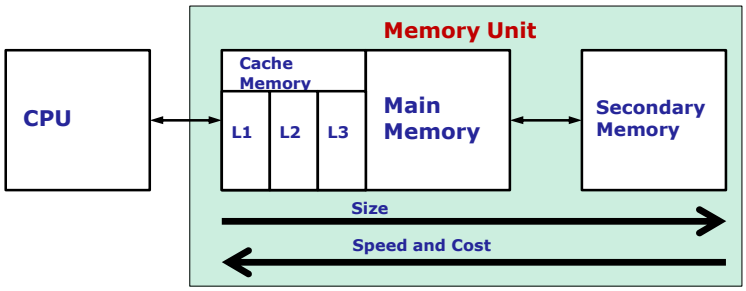
Ideal Memory

- **Fast, large and inexpensive**
- **Fast memory—SRAM**
  - Expensive
  - Cannot build large memory– packing constraints
- **Alternative—DRAM**
  - Simpler cells
  - Less expensive
  - Significantly lower speed
- Not possible to build memory with affordable DRAM for the size expected for handling voluminous data
- **Secondary storage**

3

3

Memory Hierarchy



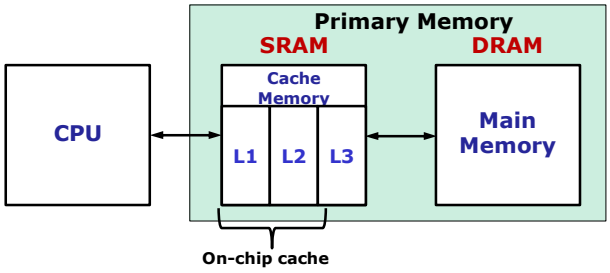
- **Secondary memory—magnetic disks**
  - Possible to build large memory
  - Much slower than semiconductor memory

4

4

Cache Memories

- The speed of main memory is very slow in comparison with the speed of modern processors
- For good performance, processor cannot spend much of its time in waiting to access instructions and data in the main memory
- Scheme is needed to reduce the time to access information
- Efficient solution is the use of cache memory



5

5

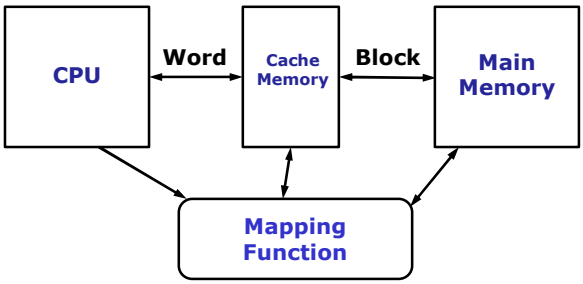
Cache Memory

- The cache memories are designed to exploit the locality of reference in the program
- **Locality of reference**:
  - Many instructions in localized areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently
- Different ways of locality of reference
  1. Temporal locality:
     - Recently executed instruction is likely to execute again very soon
  2. Spatial locality:
     - Instructions in close proximity to a recently executed instruction (with respect to instruction address) are likely to be executed very soon

6

6

Use of a Cache Memory



- Unit of transfer between main memory and cache is block
  - A block is a set of fixed number of words in contiguous address locations
  - Cache block is also called as Cache line
- Mapping function: Correspondence between main memory blocks and those in the cache

7

7

Replacement Algorithm

- Read hit/Write hit [Cache hit]:
  - When processor issues read or write request, cache control circuit determines whether the requested word exits in cache
  - If it exists in cache, the read or write operation is performed on the appropriate cache location
  - This means, read hit or write hit is said to have occurred
- Cache miss:
  - If the desired word is not there in cache during read/write operation, then cache miss is said to have occurred
  - During that time new block need to be brought into cache
- Cache control hardware decide which block to removed to create space for new block that contain referenced word when cache is full
- The collection of rules for making this decision constitutes the replacement algorithms

8

8

Read and Write Operations on Cache

- Read operation:
  - The main memory is not involved
  - Handling read miss:
    - Approach 1:
      - The block of words that contains the requested word is copied into the cache
      - After entire block is loaded into cache, particular requested word is forwarded to processor
    - Approach 2: Load through or early restart
      - A word is sent to processor as soon as it is read from the main memory
      - At the same time, the block of words that contains the requested word is also copied into the cache
      - This approach reduces the processor waiting period, but with the expense of complex circuitry

9

9

Read and Write Operations on Cache

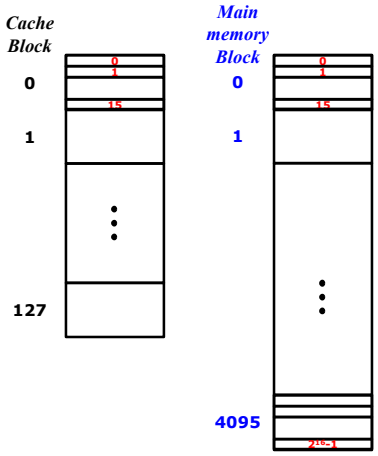
- Write operation:
  - Two techniques for write operation:
    - Write-through protocol
      - The cache location and main memory location are updated simultaneously
    - Write-back (copy-back) protocol
      - Update only the cache location and mark it as updated in a flag bit called dirty bit or modified bit
      - The main memory location is updated later when the block containing that modified word need to be removed from the cache
  - Handling write miss:
    - Write-through protocol:
      - The information is written directly into the main memory
    - Write-back (copy-back) protocol:
      - The block containing the addressed word is first brought into the cache
      - Then the desired word in the cache is overwritten with new information

10

10

Mapping Function

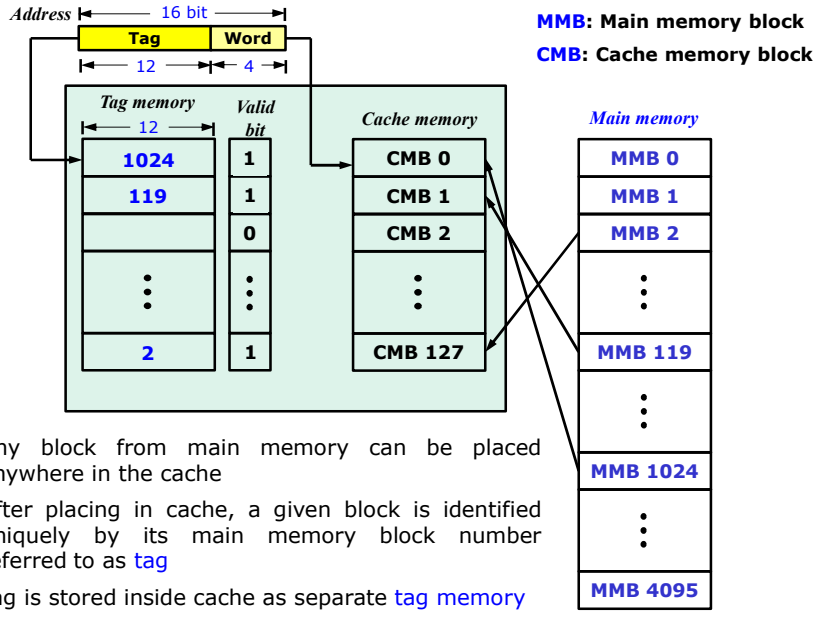
- Specifies where main memory blocks are placed in the cache
- Cache and main memory are viewed as collection of fixed number of blocks
- Example: Consider a cache and main memory with 2K words and 64K words respectively and each blocks are of size 16 words
  - Block size: 16 words
  - Number of blocks in a cache, N = 2K/16 = 128
  - Number of blocks in main memory, M = 64K/16 = 4K = 4096
  - Addressable location in main memory: 2^{16}



11

11

Mapping Function:
Associative Mapping (Associative Mapped Cache)



MMB: Main memory block

CMB: Cache memory block

- Any block from main memory can be placed anywhere in the cache
- After placing in cache, a given block is identified uniquely by its main memory block number referred to as tag
- Tag is stored inside cache as separate tag memory

12

12

Mapping Function:
Associative Mapping (Associative Mapped Cache)

- Any block from main memory can be placed anywhere in the cache
- After placing in cache, a given block is identified uniquely by its main memory block number referred to as tag
- Tag is stored inside cache as separate tag memory
- Cache maintains a control bit called valid bit for each block to indicate whether the block contains valid data
- Main memory address reference is partitioned into two parts: Tag and word
- Advantage:
  - Simple and most flexible
  - Complete use of its capacity
- Disadvantage:
  - Tag memory must be searched entirely for each memory reference: Associative search
  - Expensive

13

13

Mapping Function:
Direct Mapping (Direct Mapped Cache)

- Main memory block is placed in one and only one place in the cache
- Simplest way to determine the cache location
- Let N be the number of blocks in cache
- The jth main memory block is placed (mapped) onto (j mod N)th block of the cache

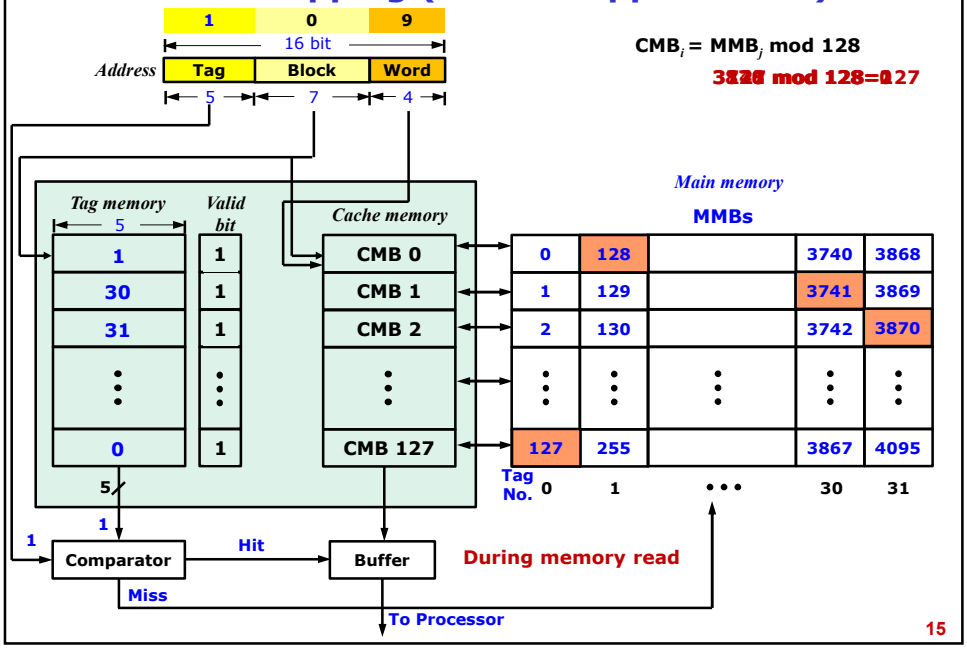
  $$CMB_i = MMB_j \bmod N$$

- Example: For number of blocks in a cache, N = 128 and the number of blocks in main memory, M = 4096
  - The main memory blocks 0 or 128 or 256 etc. are mapped onto cache block 0
  - The main memory blocks 1 or 129 or 257 etc. are mapped onto cache block 1
- For the understanding sake, lets consider the main memory as rectangular array of blocks

14

14

Mapping Function:
Direct Mapping (Direct Mapped Cache)



15

Mapping Function:
Direct Mapping (Direct Mapped Cache)

- Advantage:
  - Simplicity
  - Cost is less compared to fully associative cache
- Disadvantage:
  - Only a single block from a given group can be present in the cache
  - Not flexible
    - Imposes considerable amount of rigidity on the cache organization
- It relies on the principle of locality of reference for its success
- If two blocks of same group are frequently referenced, it leads to trashing

16

16

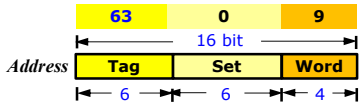
Mapping Function: Set-Associative Mapping

- Combination of the direct and fully associative mapping
- Blocks of the cache are grouped into sets
- Mapping allows block of the main memory to reside in any block of a specific set
- With in a set, it is fully associative
- ***K*-way set associative**:
  - A set may hold *K* number of blocks
  - Number of sets = N/K
  - $CMS_i = MMB_j$ mod (N/K) where CMS = Cache memory set
- Intel P-III and P-IV processors contain 8-way set associative cache
- Example: Let number of blocks in a cache, N = 128 and the number of blocks in main memory, M = 4096

17

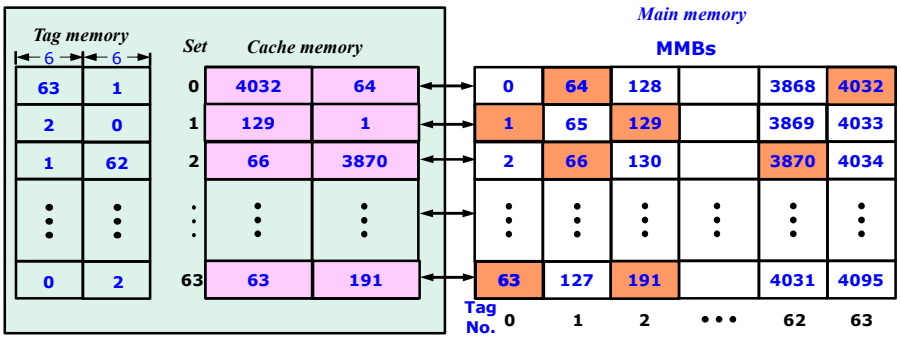
17

Mapping Function: 2-way Set-Associative Mapping



$CMS_i = MMB_j$ mod 128/2

[illegible] mod 64=[illegible]



18

18

Replacement Algorithm

- Direct mapped cache:
  - The position of each block is predetermined
  - No replacement strategy exists
- In associative and set-associative cache, there exists flexibility
  - Replacement strategy needed
- Replacement algorithms helps in deciding which of the old blocks in the cache to be replaced when the cache is full and a new block is brought into the cache
- Objective: To keep the blocks in cache that are likely to be referenced
- Locality of reference in program gives a clue to a reasonable strategy
- It is sensible to replace the block that has gone longest time without being referenced

19

19

Least Recently Used (LRU) Algorithm

- The cache controller need to track references to all blocks
- LRU algorithm is implemented by using counters for each block
- LRU algorithm:
  - **During cache hit**:
    - Counter of the block referenced is set to 0
    - Counter of the empty block locations are unchanged
    - Counter of other occupied block locations are incremented by 1
  - **During cache miss and cache is not full**:
    - Load the main memory block to empty space and set the counter of that block to 0
    - Increment the counter of other block location by 1
  - **During cache miss and cache is full**:
    - Block with largest counter value (LRU) is removed
    - New block is loaded in that emptied location
    - Counter of that location is set to 0
    - Increment the counter of other locations by 1

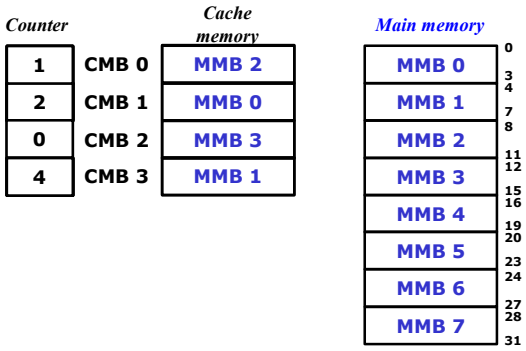
20

20

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3
Read LOC#7
Read LOC#11
Read LOC#2
Read LOC#8
Read LOC#14
Read LOC#16



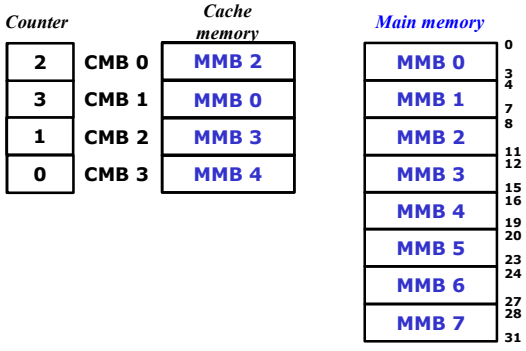
21

21

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3
Read LOC#7
Read LOC#11
Read LOC#2
Read LOC#8
Read LOC#14
Read LOC#16
Read LOC#23



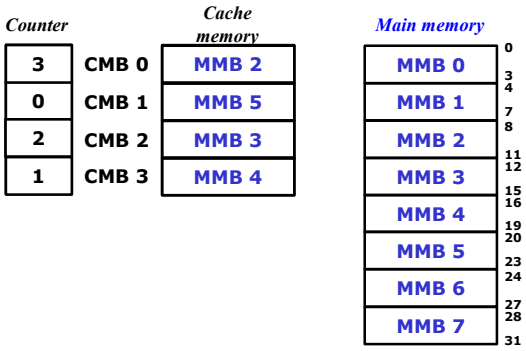
22

22

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3
Read LOC#7
Read LOC#11
Read LOC#2
Read LOC#8
Read LOC#14
Read LOC#16
Read LOC#23
Read LOC#25



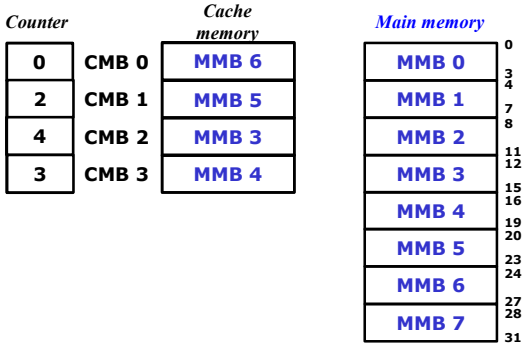
23

23

Illustration: LRU Algorithm

- Consider fully associative cache
  - Let number of blocks in a cache, N = 4 and the number of blocks in main memory, M = 8
  - Let each block is 4 words

Read LOC#3
Read LOC#7
Read LOC#11
Read LOC#2
Read LOC#8
Read LOC#14
Read LOC#16
Read LOC#23
Read LOC#25
Read LOC#27



24

24

Cache Memory Architectures

- Cache memory hierarchy: L1 cache, L2 cache and L3 cache
- Two architectures of cache memories:
  - Harvard architecture cache
    - Separate data and instruction cache
    - Advantages:
      - Prevents conflicts between blocks of instruction and data that might map onto same location
      - Program generally do not modify instructions
      - Instructions take less memory than program data
    - Generally used in L1 cache
  - Unified cache (Princeton architecture cache)
    - Cache contain both instruction and data

25

25

Categories of Cache Misses

- Goal: To reduce number of cache misses
- This requires to know the reasons for cache misses

1. Compulsory miss:
   - Occurs when the cache is first referenced
   - It causes the block to be brought into the cache
2. Capacity miss:
   - Occurs when the amount of data referenced by the program exceeds the capacity of the cache
   - It causes some blocks to be evicted to make room to new data
   - If the evicted data is referenced again by the program, capacity miss occurs
3. Conflict miss:
   - Occurs in associative/set-associative mapping
   - Occurs when program references more blocks of data mapped on to the same set
   - It causes one of the block to be removed
   - If the removed block is referenced again, conflict miss occurs

26

26

Cache Performance Considerations

- Ideally, entire memory hierarchy would appear to the processor as a single memory unit, that has the access time of a cache on processor (L1) and the size of memory disk
- Performance is adversely affected by the actions that must be taken after a miss
- Hit: Successful access to data in a cache
- Hit rate (h): Ratio of number of hits over all attempted access
  - Example: h=0.9 means 90% of the time required block is in cache
- Miss rate: Ratio of number of misses over all attempted access

27

27

Cache Performance Considerations

- Miss penalty:
  - The extra time needed to bring the desired information into the cache
  - It is the time that the processor is stalled during waiting
  - It is also the time needed to bring a block of data from a slower unit in the hierarchy to a faster unit
- Suppose we have a cache and a main memory in the hierarchy
  - Let h be the hit rate
  - t_M be the miss penalty i.e. time to access information in the main memory
  - t_C be the cache hit latency i.e. time to access information in cache
  - The average access time experienced by the processor:

$$t_{avg} = h\,t_C + (1-h)\,t_M$$ i.e. t_{avg} = hit rate * t_C + miss rate * t_M

28

28

Cache Performance Considerations

- Suppose we have L1 & L2 caches and a main memory in the hierarchy
  - h_1 : hit rate of L1 cache
  - h_2 : hit rate of L2 cache
  - t_M be the miss penalty i.e time to access information in the main memory
  - t_{C1}: Time to access L1 cache (L1 cache latency)
  - t_{C2}: Time to access L2 cache (L2 cache latency)
  - The average access time experienced by the processor in the two level cache:

$$t_{avg} = h_1 t_{C1} + (1-h_1) h_2 t_{C2} + (1-h_1)(1-h_2) t_M$$

29

29

Cache Performance Considerations

- A cache has a hit rate of 95%, and a block capacity of 128-byte and a cache hit latency of 5 ns. Each word in a block is 32 bits. The main memory takes 100 ns to return a block.
  - What is the cache block size?
  - What is the average memory access time?
- Ans: 1. Cache block size: 32 words
  2. Average memory access time: 9.75 ns

$$t_{avg} = 0.95 \times 5 + (0.05) \times 100$$
$$= 4.75 + 5$$
$$= 9.75 \text{ ns}$$

30

30

Reference

- **Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", 5th Edition, Tata McGraw Hill, 2002**

31

31

Thank You

32

32