Cache Memory – Part 2

Cache memory

Cache memory is divided into blocks/lines.

These memory blocks can be of different sizes.

Cache Size – C (bytes) = S x A x B

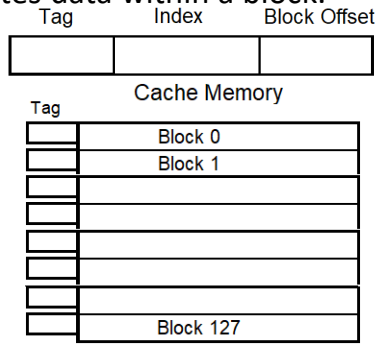
- S = Number of sets or cache rows.
- B = Block size (bytes)
- A = Associativity (determines mapping techniques, to be discussed later)

Cache memory address is generally divided into three fields.

Tag – Most significant bits. Determines which main memory block is mapped to cache memory.

Index – Specifies cache row (set) in which main memory block is copied.

Block Offset – Least significant bits. Locates data within a block.



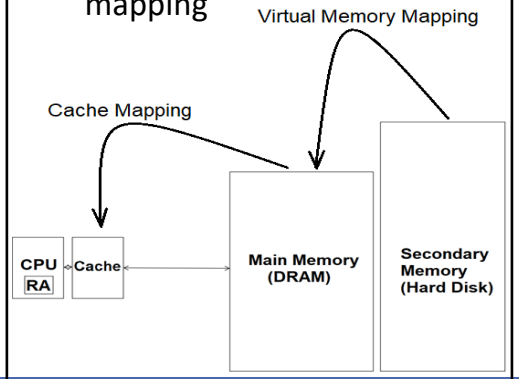
Cache mapping techniques

Cache mapping defines how a memory block is mapped to cache.

- Main memory contains equal size partitions called memory blocks or frames.
- During mapping, some of the main memory blocks are copied to cache memory.
- Remember, Main Memory Size >> Cache Memory Size
- Performance of cache memory mapping function is key to overall speed.

Cache mapping can be performed using following techniques:

- Direct mapping
- Fully associative mapping
- (N-way) Set associative mapping



Direct Mapping

Each memory block maps to a specific cache block, the simplest cache mapping.

Direct Mapping Algorithm

If cache has 2^n blocks, data at memory address i is mapped as: Cache block index $= i \ mod \ 2^n$

Cache memory block # = (Main memory block #) mod (# of cache memory blocks)

0 mod 4 = 0 1 mod 4 = 1 2 mod 4 = 2 3 mod 4 = 3
4 mod 4 = 0 5 mod 4 = 1 6 mod 4 = 2 7 mod 4 = 3
8 mod 4 = 0 9 mod 4 = 1 10 mod 4 = 2 11 mod 4 = 3
12 mod 4 = 0 13 mod 4 = 1 14 mod 4 = 2 15 mod 4 = 3

S = # of sets (rows) = # of cache blocks, Associativity = 1, Block size = B bytes

Advantage:

- Simplest cache mapping technique
- Easy to implement
- No need for replacement policy

Disadvantage:

- Not flexible
- Inefficient usage of cache even when it is not full. 0 mod 128 = 0

128 mod 128 = 0
256 mod 128 = 0

Even if remaining cache is empty, collisions will result in overwriting/evictions.

❑ Poor performance if all three blocks are used frequently.

Consider following scenario…

- Cache memory consists of 128 blocks each containing 16 bytes
  Total cache size = 128 blocks x 16 bytes = 2048 (2K) bytes.
- Main memory is addressable by 16-bit address
  Total size of main memory is 2^{16} = 65536 (64K) bytes.
  Number of 16-byte blocks in main memory = 64K / 16 = 4K
  Cache memory block # = mod 128

Total cache size = 128 blocks x 16 bytes = 2048 (2K) bytes.

Number of 16-byte blocks in main memory = 64K / 16 = 4K

- Memory address is divided into tag, index, block-offset.
- Block size = 16 bytes = 2^4 bytes, Block offset bits = $\log_2 B = 4$ bits.
- Total number of rows = 128 = 2^7 blocks
  Index bits = $\log_2 S = 7$ bits
- Tags = # of main memory blocks / # of cache memory blocks = 4K / 128 = 2^5
  Tag bits = $\log_2 2^5$ = 5 bits