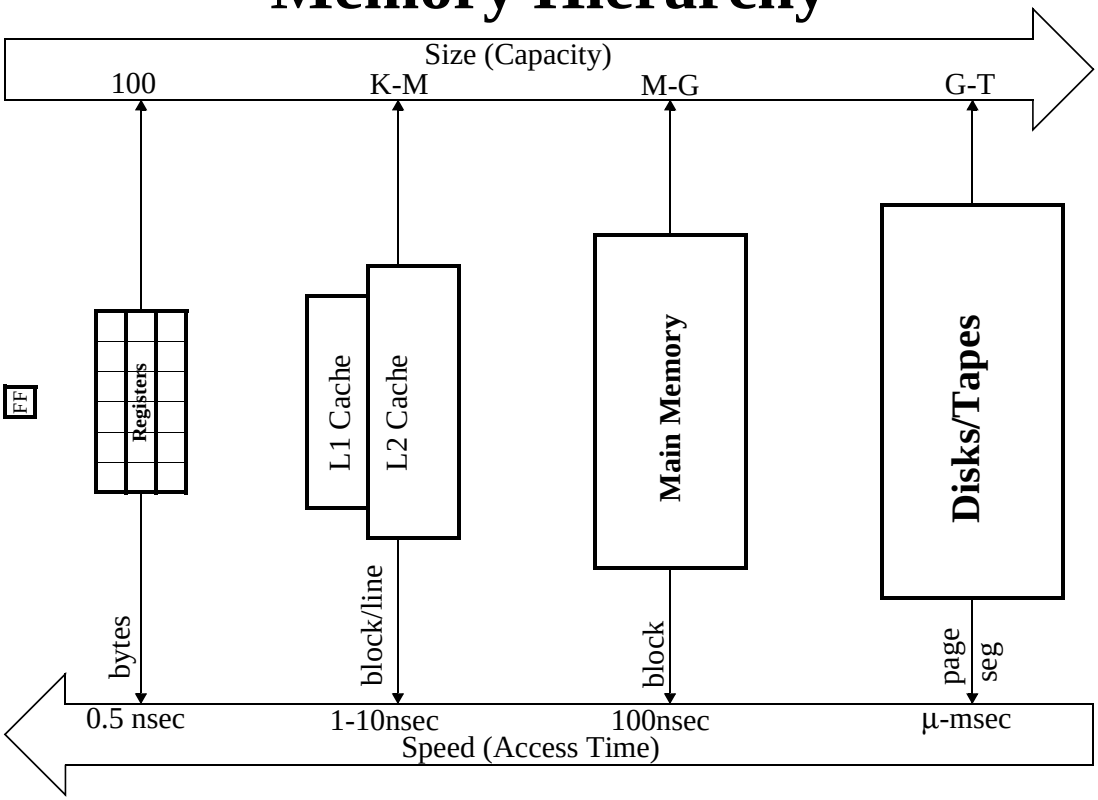
CS650
Computer Architecture

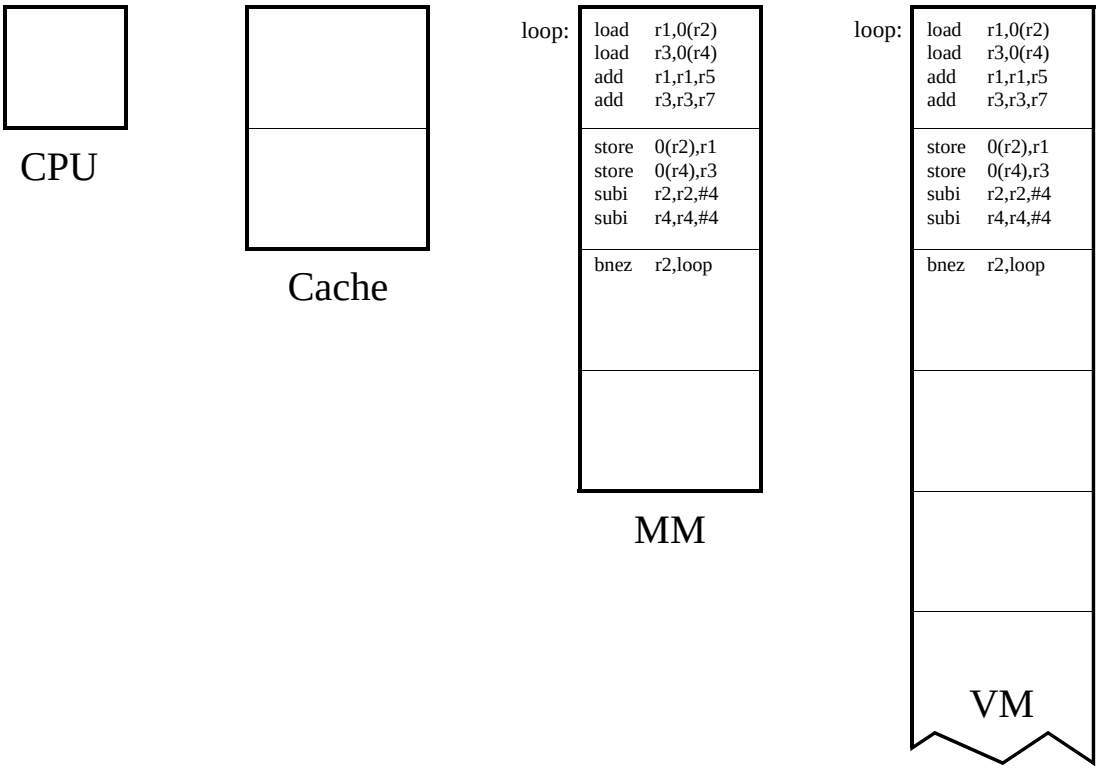
Lecture 8
Memory Hierarchy - Cache Memory

Andrew Sohn
Computer Science Department
New Jersey Institute of Technology

Memory Hierarchy



Issues



Four Issues

1. **Block Placement Policy:** Where to place the newly brought-in block
2. **Identification Policy:** How is the block found if it's in the upper level
3. **Block Replacement Policy:** Which block should be replaced on a miss
4. **Write Policy:** What happens on a write?

Placement Policy

Where to place a MM block in Cache

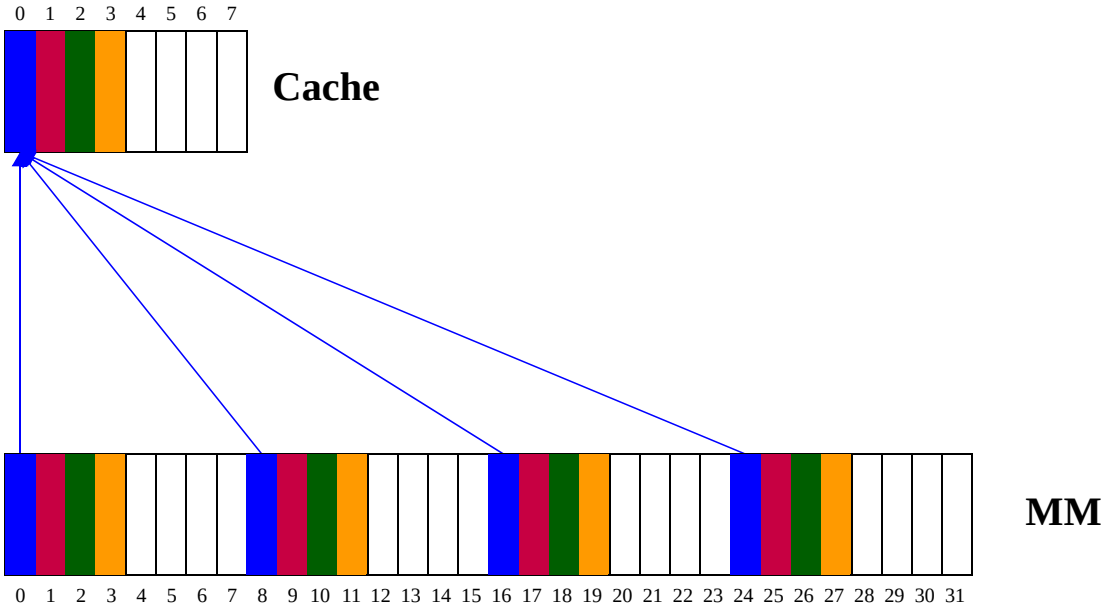
1. Direct Mapping:
Each block of MM has *only one* place it can appear in the cache.
How to determine the cache address? Cache address = (MM block address) % (No of blocks in cache)
For MM block #9, cache address = 9 % 8 = 1. Block #9 can only be placed in block #1 of cache.

2. Fully Associative Mapping:
A block of MM can be anywhere in the cache.
How to determine the cache address??

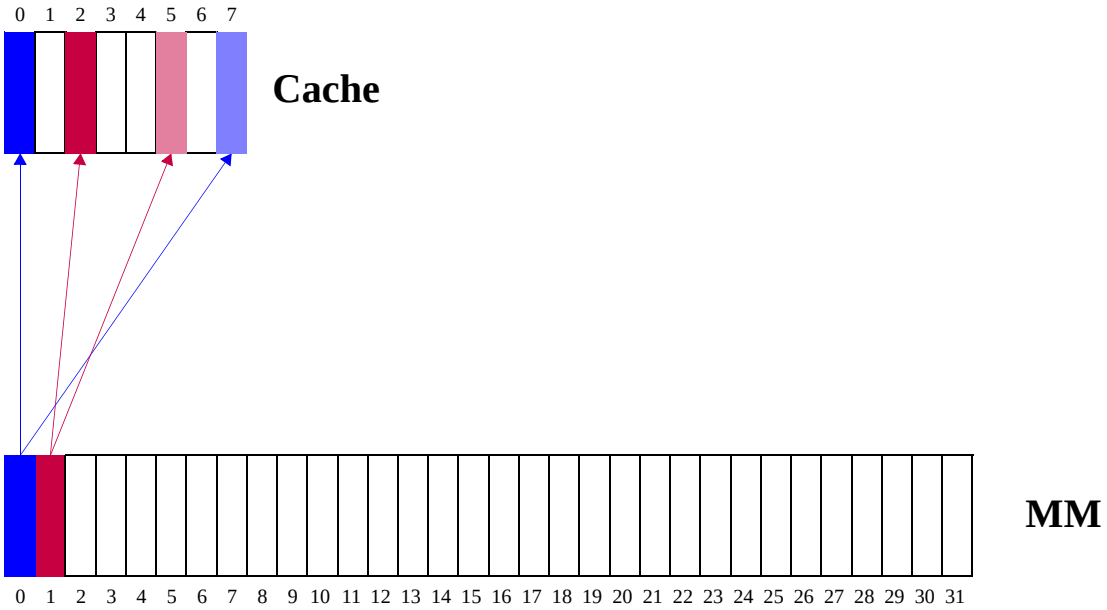
3. Set Associative Mapping:
A block of MM can be placed *anywhere* within the set.
How to determine the cache address? Cache set address =
(MM block address) % (No of sets in the cache)
For MM block #9, set # of cache = 9 % 4 = 1. Block #9 of MM can go anywhere within Set #1.

Cache with 4 sets is called 4-way set associative cache.
Cache with n sets is called n-way set associative cache.

Direct Mapping (1-way)

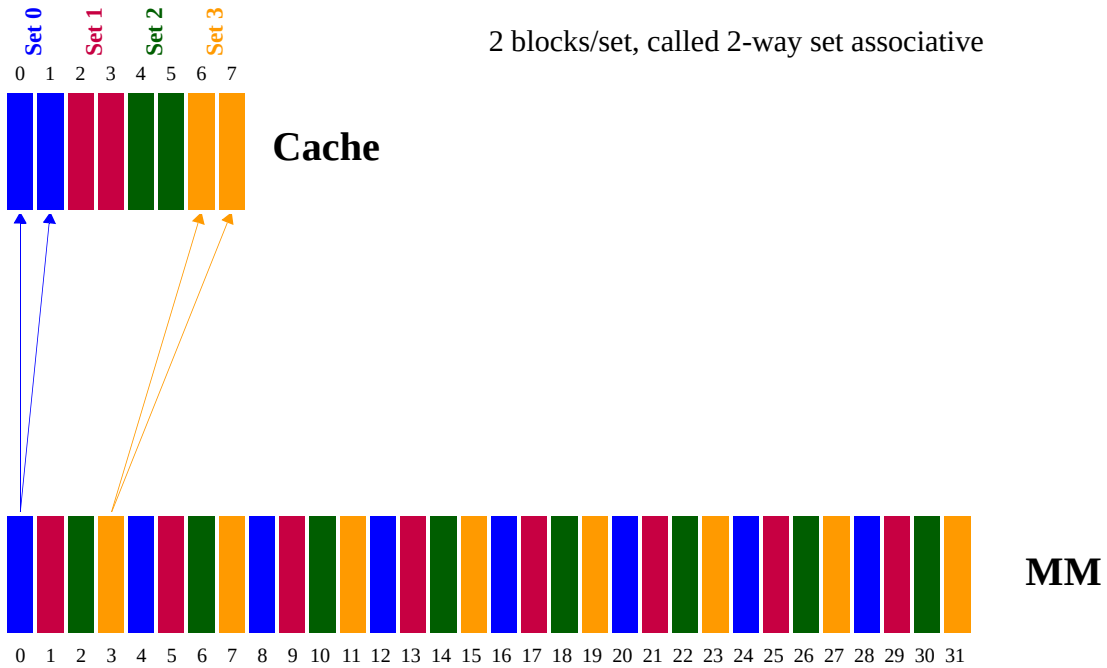


Fully Associative Mapping (n-way)



Set Associative Mapping (k-way)

2 blocks/set, called 2-way set associative



Identification Policy

Finding a MM Block in Cache

Suppose we check if MM block #12 is present in cache. For the comparison purpose, each block of cache has a tag attached to it. How many comparisons are needed for each placement policy?

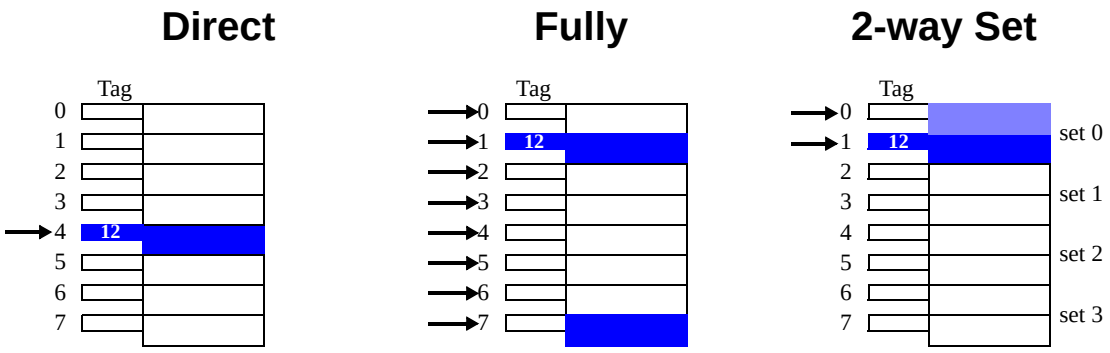
1. Direct Mapping: only 1 comparison.
MM block can appear in only one place in the cache: 12 mod 8 = 4. Cache block 4.
Check the tag of cache block #4 to see if it has 12 in it.

2. Fully Associative Mapping: 8 comparisons.
MM block can appear anywhere in the cache. Need to compare all the 8 tags.

3. Set Associative Mapping: 2 comparisons.
MM block #12 can go anywhere within a particular set that has 2 blocks.
Then, which set to examine? 12 % 4 = 0
Check all the blocks in set 0.

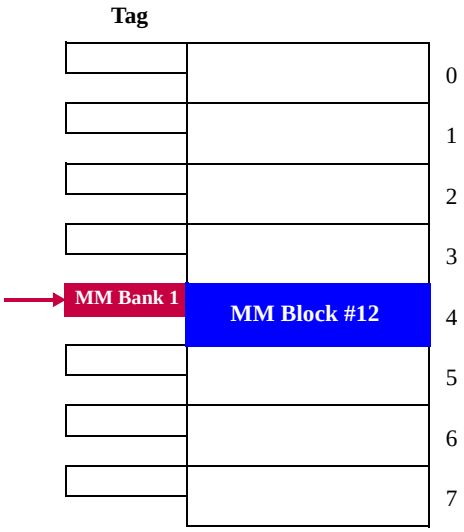
How about n-way set associative mapping? n comparisons since each set has n blocks.

Identification Policy



Direct Mapping

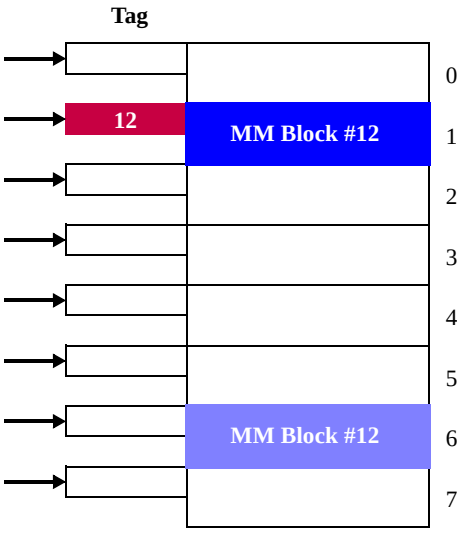
Given cache=8 blocks, MM=32 blocks, 16 bytes/block, check if MM Block #12 is in cache.



- How many comparisons needed? Only 1 since a MM block can appear in one and only one place in the cache.
- Where? 12 % 8 = 4. It can be only in cache block #4.
- Directly go to Tag #4 and compare the content of Tag #4 to see if it has 12 in it.
- But Tag #4 will not contain 12. What is in the tag? Tag contains a bank number of MM, which has Block 12.
- The tag will have 1, indicating Bank 1 of MM since Bank # =12 / 8=1. Block #12 of MM is in MM Bank 1.
- The number of bits required for tag is 2 bits since there are 4 banks in MM.

Fully Associative Mapping

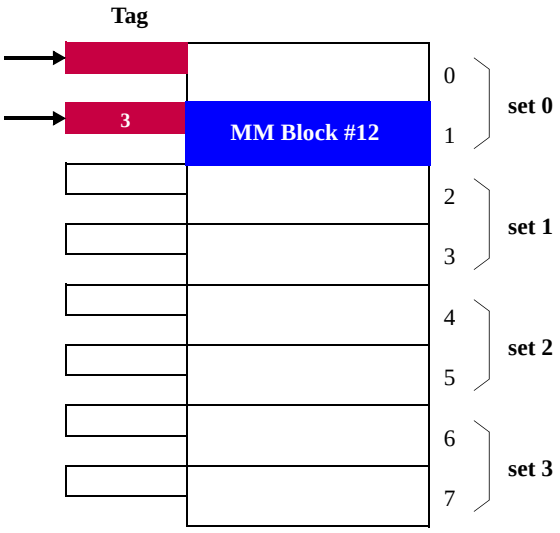
Given cache=8 blocks, MM=32 blocks, 16 bytes/block, check if MM Block #12 is in cache.



- How many comparisons needed? 8. Since MM block can appear anywhere in the cache, we have to compare all 8 tags.
- What is in the tag? Tag contains Block 12.
- The number of bits required for tag is 5 bits since there are 32 blocks in MM for our running example.

2-way Set Associative

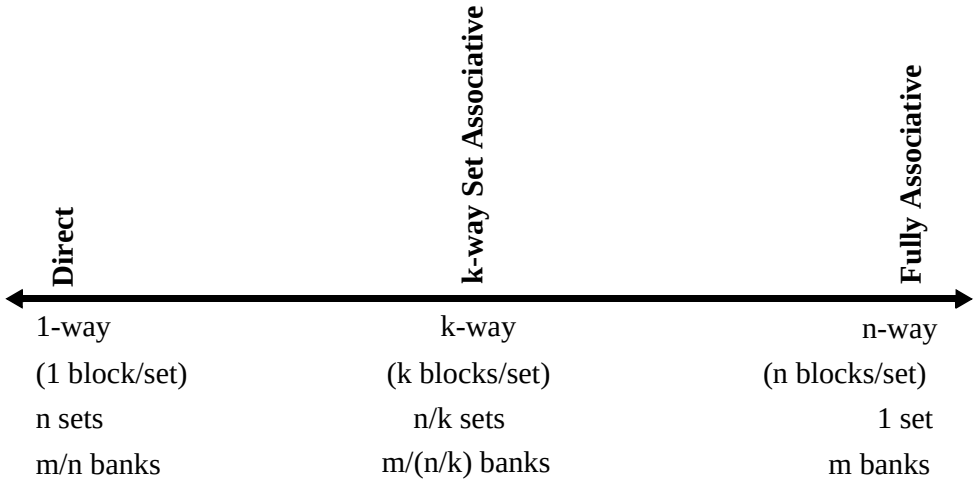
Given cache=8 blocks, MM=32 blocks, 2 blocks/set, 16 bytes/block → Cache=4 sets, MM=32 blocks/4 sets=8 banks. Check if MM Block #12 is in cache.



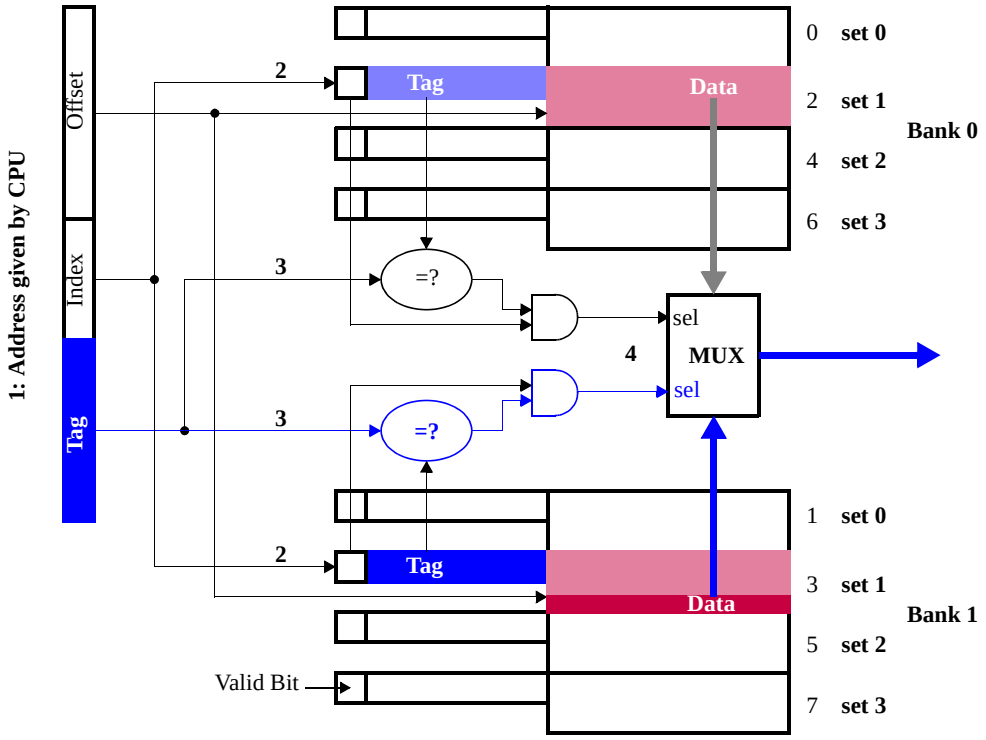
- How many comparisons needed? 2 since MM block #12 can go anywhere within a particular set that has 2 blocks.
- Which set to examine? Set #=12 % 4 = 0
- Go to Set 0 and examine Tag #0 and #1 to see if any of them has 12 in it.
- But Tag #0 or #1 will not contain 12. Tag contains a bank number of MM, which has Block 12.
- Tag 0 or Tag 1 will have 0, indicating Bank 0 of MM since Bank # = 12 div 4 = 3. Block #12 of MM is in Bank 3 of MM. Remember each bank has 4 blocks for 2-way SAM.
- The number of bits required for tag is 3 bits since there are 8 banks in MM.

Associativity

For a cache with n blocks and a MM with m blocks



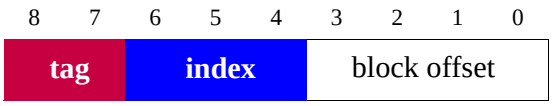
Relatationship of CPUAddress to Cache



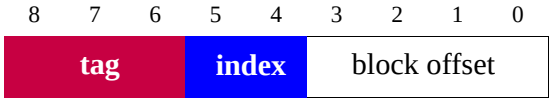
Relationship of CPU Address to Cache

Cache=128 bytes (8 blocks), MM=512 bytes (32 blocks), Block=16 bytes, CPU address formats to cache for four different placement policies:

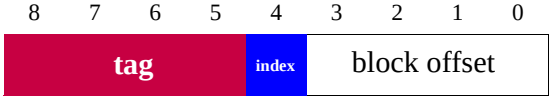
1-way set associative mapping (direct)



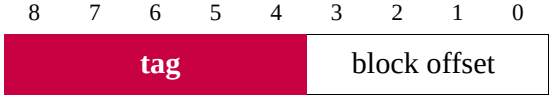
2-way set associative mapping



4-way set associative mapping



8-way set associative mapping (full)



Write Policy

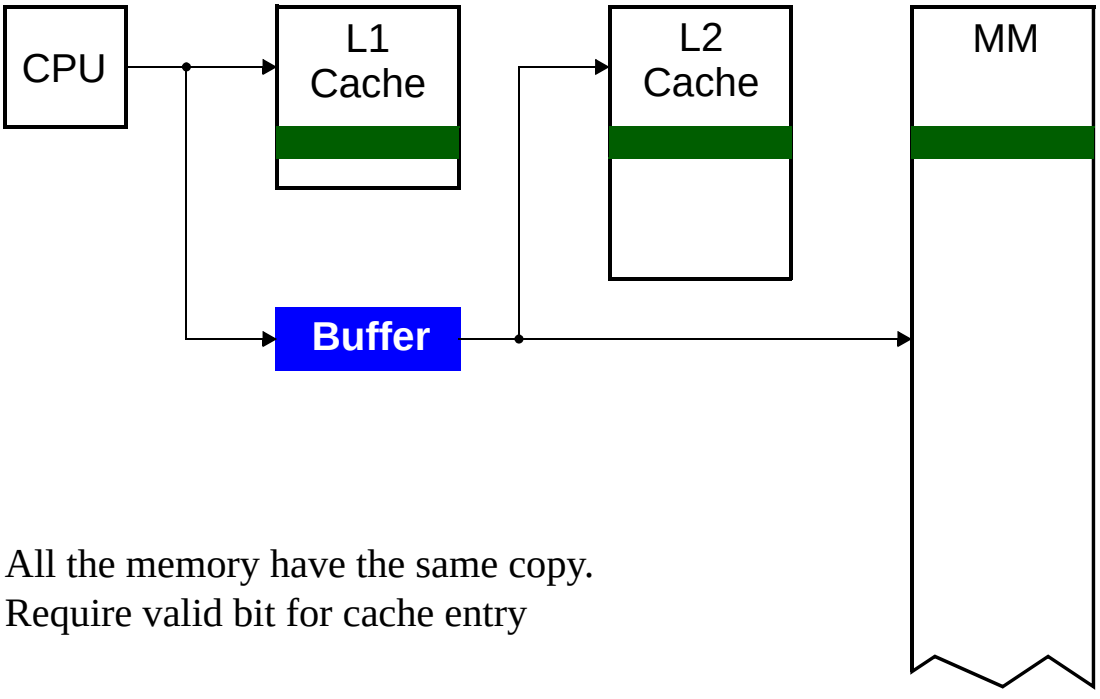
Write Through

- Information is written to both the block in the cache and to the block in the MM.
- If there is a write buffer for MM and it is empty, information is written into cache and write buffer.
- CPU continues working while the write buffer writes the word to memory.
- If the write buffer is full, the cache and the CPU must wait until the ubffer is empty.

Write Back

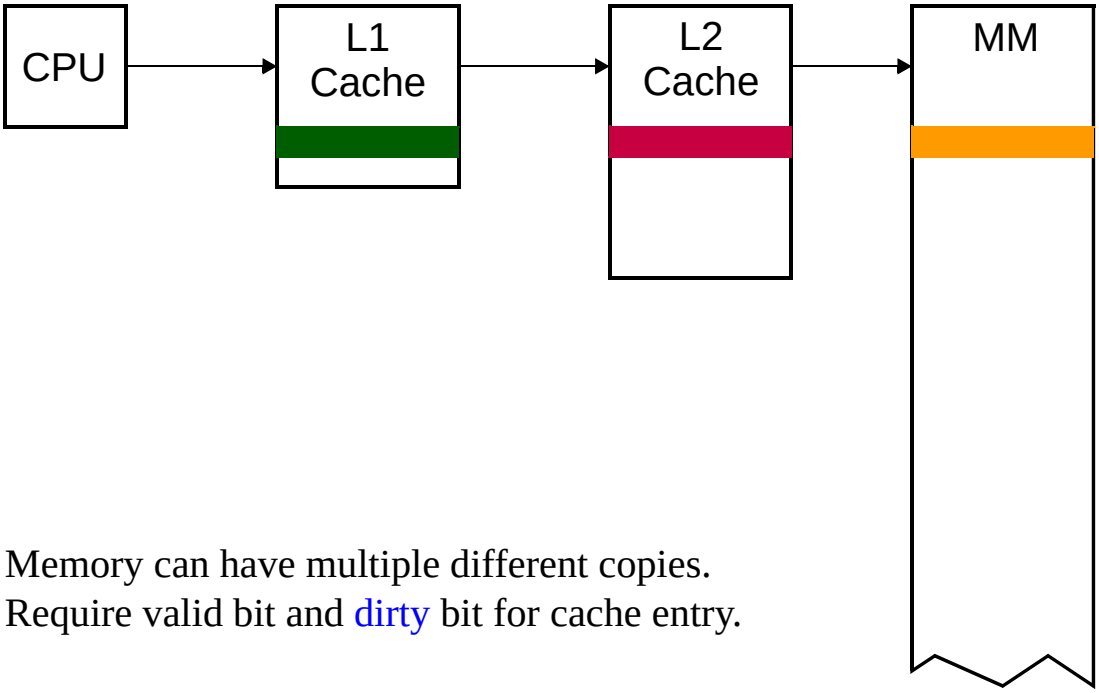
- Information is written only to the block in the cache.
- The modified cache block is written to MM only when it is replaced.
- This requires an additional information (either hardware or software), called dirty bits.
- A dirty bit is attached to each tag of the cache.
- Whenever the information in cache is different from the one in MM, then write back to MM. Hence, write back.

Write Through Policy



All the memory have the same copy.
Require valid bit for cache entry

Write Back Policy



Memory can have multiple different copies.
Require valid bit and dirty bit for cache entry.

What Happens on Write-miss?

Write Allocate

- Can also occur on a read-miss
- The block is loaded into cache then write into the block.

No Write Allocate

- The block is modified in L2/MM and not loaded into the cache.

Improving Cache Performance

1. Reduce cache miss rate (number of cache misses)
2. Reduce cache miss penalty
3. Reduce the time to hit in the cache

Reducing Cache Miss Rate

Reducing the number of cache misses

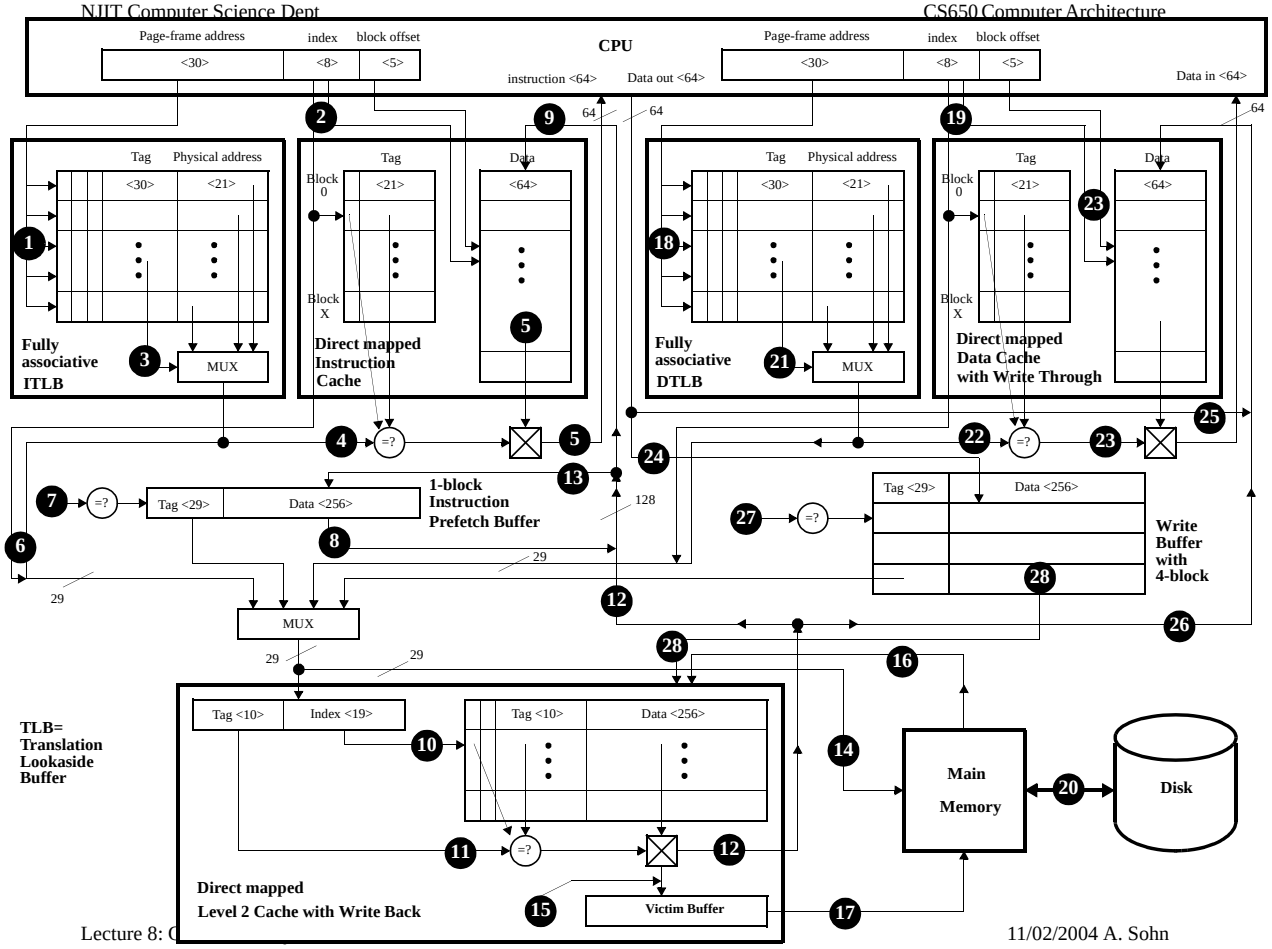
1. Large block size
2. Higher associativity
3. Victim caches
4. Pseudo-associative caches
5. Hardware prefetching of instructions and data
6. Compiler-controlled prefetching
7. Compiler optimization

Reducing Cache Miss Penalty

1. Giving priority to read misses over writes
2. Sub-block placement for reduced miss penalty
3. Early restart and critical word first
4. Nonblocking caches to reduce stalls
5. Second-level caches

Reducing Hit Time

1. Small and simple caches
2. Avoiding address translation during indexing
3. Pipelining writes for fast write hits


CPU
Page-frame address
index
block offset
<30>
<8>
<5>
instruction <64>
Data out <64>
Data in <64>
Tag
Physical address
Block 0
Block X
Data
<21>
<64>
Fully associative ITLB
Direct mapped Instruction Cache
Fully associative DTLB
Direct mapped Data Cache with Write Through
MUX
1-block Instruction Prefetch Buffer
Tag <29>
Data <256>
Write Buffer with 4-block
TLB= Translation Lookaside Buffer
Tag <10>
Index <19>
Direct mapped Level 2 Cache with Write Back
Victim Buffer
Main Memory
Disk
64
128
29