

Cache Memory Mapping Techniques

Continue to read pp. 289-305

Cache Memory Mapping

- Again cache memory is a small and fast memory between CPU and main memory
- A block of words have to be brought in and out of the cache memory continuously
- Performance of the cache memory mapping function is key to the speed
- There are a number of mapping techniques
 - Direct mapping
 - Associative mapping
 - Set associative - mapping

Direct Mapping Technique

- Simplest way of mapping
- Main memory is divided in blocks
- Block j of the main memory is mapped onto block $j \bmod 128$ of the cache – consider a cache of 128 blocks of 16 words each

Cache

tag	Block 0
tag	Block 1
tag	Block 127

- Consider a memory of 64K words divided into 4096 blocks

Where blocks 0, 128, 256, ... 3968 should be mapped to?

Where blocks 126, 254, 382, ... 4094 should be mapped to?

Tag	Block	Word
5	7	4

Main memory address

Direct Mapping Technique (Continued)

- Mapping process
 - Use tag to see if a desired word is in cache
 - If there is no match, the block containing the required word must first be read from the memory
 - For example: MOVE \$A815, DO

10101 00000001 0101
Tag Block # Word

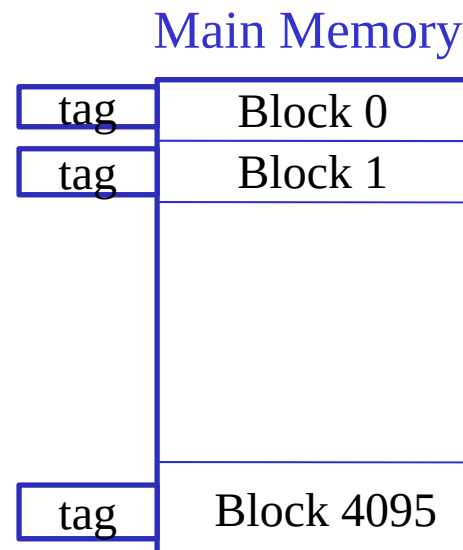
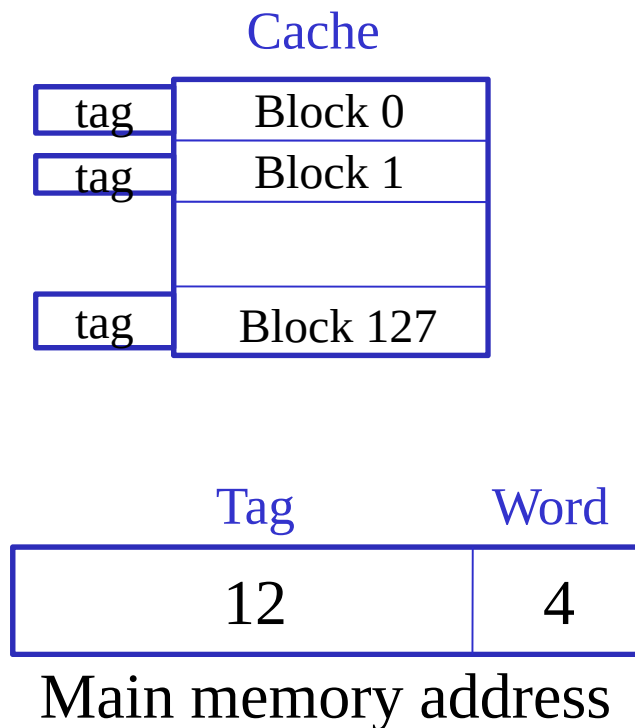
- a. Check if cache has tag 10101 for block 1
match -> hit; different -> miss, load the corresponding block
- b. Access word 5 of the block

Direct Mapping Technique (Continued)

- Advantage
 - simplest replacement algorithm
- Disadvantage
 - not flexible
 - there is contention problem even when cache is not full
 - For example, block 0 and block 128 both take only block 0 of cache:
 - $0 \text{ modulo } 128 = 0$
 - $128 \text{ modulo } 128 = 0$
 - If both blocks 0 and 128 of the main memory are used a lot, it will be very slow

Associative Mapping Technique

- Any block can go anywhere in cache
- 4095 blocks $\rightarrow$ 4095 tag = $2^{12} \rightarrow$ 12 bit tag



Associative Mapping Technique (continued)

- Advantage
 - Any empty block in cache can be used, flexible
 - Must check all tags to check for a hit, expensive (parallel algorithm has been developed to speed up the process)
- What is the next technique?
 - Something between **direct mapping** and **associative mapping**

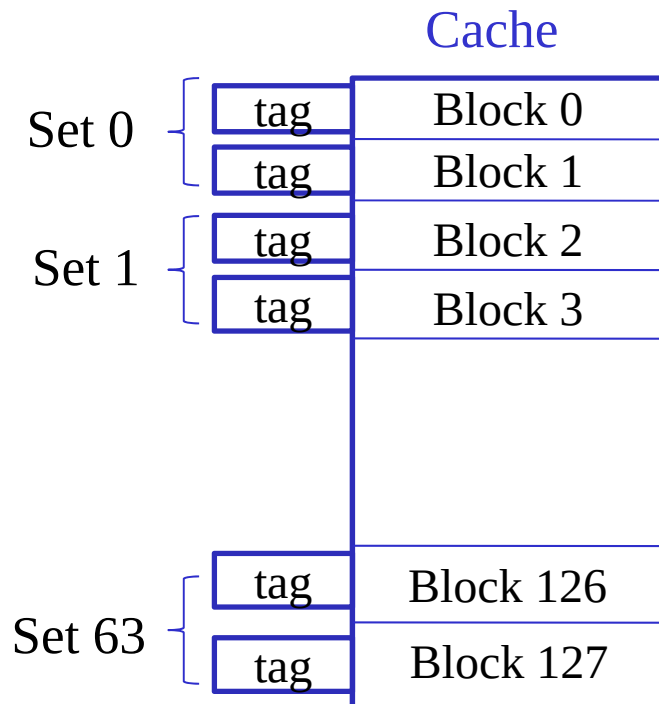
Set Associative Mapping Technique

- Comprise between direct mapping and associative mapping
- Block in main memory maps to a set of blocks in cache – direct mapping
- Can map to any block within the set
- E.g. use 6 bits for tag = $2^6 = 64$ tags
6 bits for set = $2^6 = 64$ sets

Set Associative Mapping Technique (continued)

- Memory Address

Tag	Set	Word
6	6	4

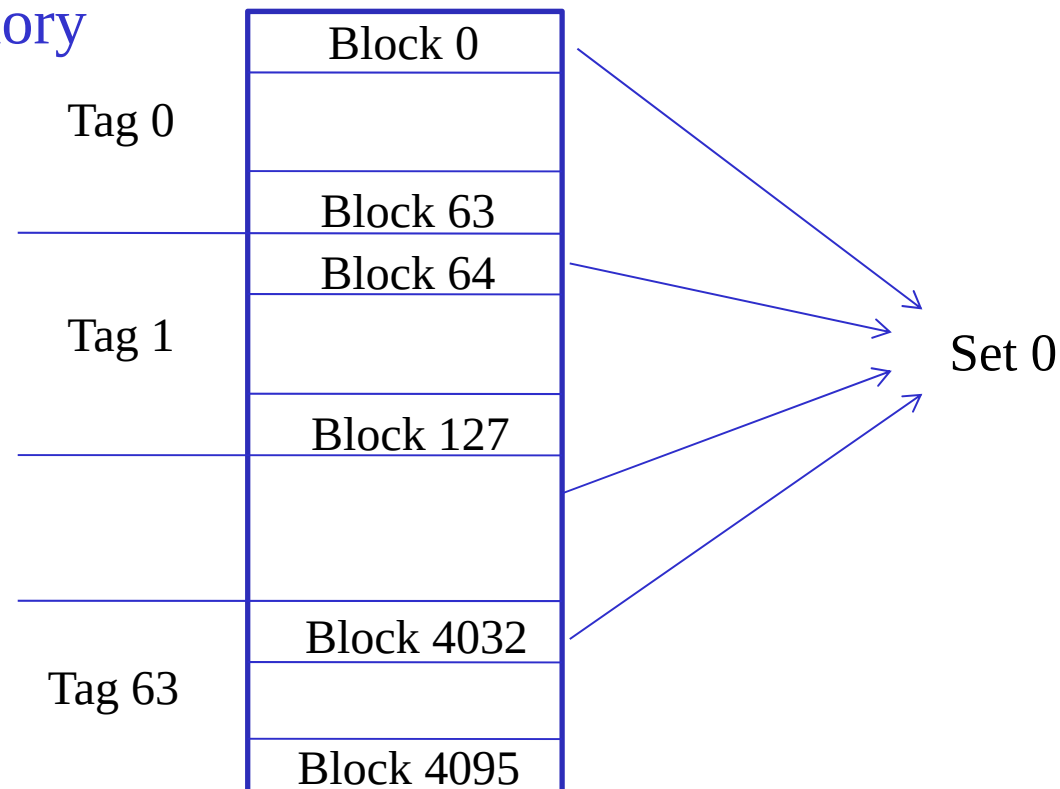


- The blocks in cache are divided into 64 sets and there are two blocks in each set
- How the blocks in the main memory be mapped into cache?
- Main memory blocks 0, 64, 128, 4032 maps to set 0 and can occupy either of the two positions

Set Associative Mapping Technique (continued)

- A set could have **one** block -> **direct mapping**; **128** blocks -> **associative mapping**
- **k** blocks per set is referred to as k-way set-associative mapping

Main memory



Cache Memory Details

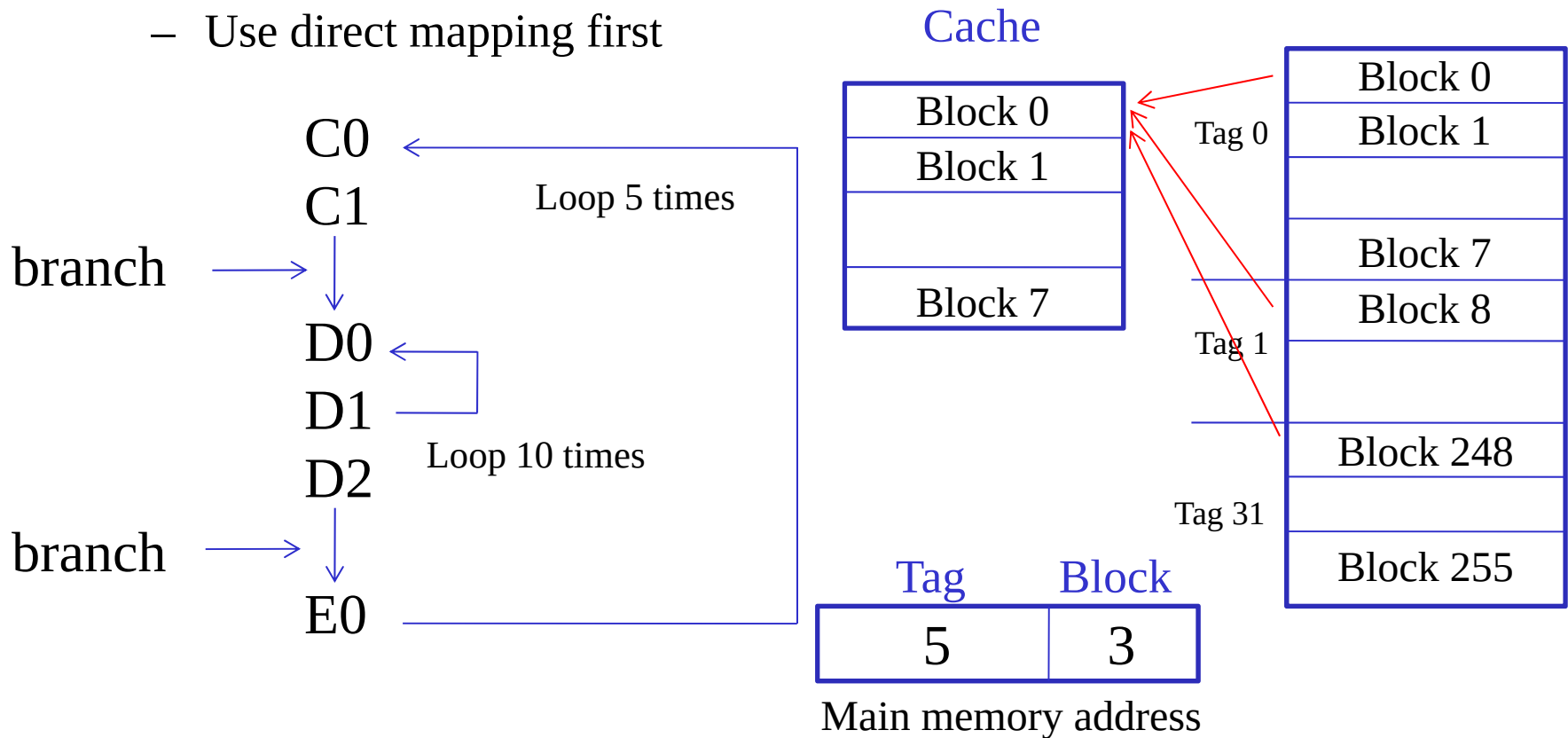
- Block size
 - Depends on how memory is addressed (byte, word, or long word) and accessed (word at a time)
 - 8-16 quite reasonable
 - 68040 – 16 bytes per block
 - Pentium IV – 64 bytes per block
 - Always work with 1 block at a time
 - How many blocks in cache?
 - No of words in cache divided by number of words per block – e.g. 2 k words, 16-word block: $2^{11} / 2^4 = 2^7 = 128$ blocks

Cache Memory Details (continued)

- Replacement Algorithms
 - Replace the one that has gone the longest time without being referenced – Least Recently Used (LRU) – block
- How to know which block of main memory is currently in cache?
 - Look at the tag on data in the block
 - How long is the tag (how many blocks use same block of cache)?
- Study a few examples

Examples

- Small Instruction Cache (read 8.6.3)
 - Cache has 8 blocks, 1 word each
 - Main memory has 256 blocks (words) – 8 bit address
 - Execute the following program
 - Use direct mapping first



Direct Mapping Performance

- How many executions? - $(2 \times 10 + 4) \times 5 = 120$

Cache Block	After C1	After Inner Loop	After E0
0	C0	D0	E0
1	C1	D1	D1
2			D2

Misses	2×5	2×5	First time $2 + 1 \times 4 = 26$
Hits		18×5	$1 \times 4 = 94$

- Hit rate = hits/total = $94/120 = 78.3\%$

Associative Mapping Performance



Cache Block	After C1	After Inner Loop	After E0
0	C0	C0	C0
1	C1	C1	C1
2		D0	D0
3		D1	D1
4			D2
5			E0
6			
7			
Misses	2	2	2 = 6
Hits	next 4 times all hits = 114		

- Hit rate = hits/total = 114/120 = 95%

Set Associative Performance

2 –way -> 4 sets

Tag	Set
6	2

Main memory address

Cache Block		After C1	After Inner Loop	After E0	Second time	
					After C1	After Loop
Set 0 {	0	C0	C0	E0	E0	D0
	1		D0	D0	C0	C0
Set 1 {	0	C1	C1	C1	C1	C1
	1		D1	D1	D1	D1
Set 2 {	0			D2	D2	D2
	1					
Set 3 {	0					
	1					
Misses		2+ 1x4	2 +1x4	2 + 1x4	= 18	
Hits		The rest is all hits = 102				

- Hit rate = hits/total = 102/120 = 85%