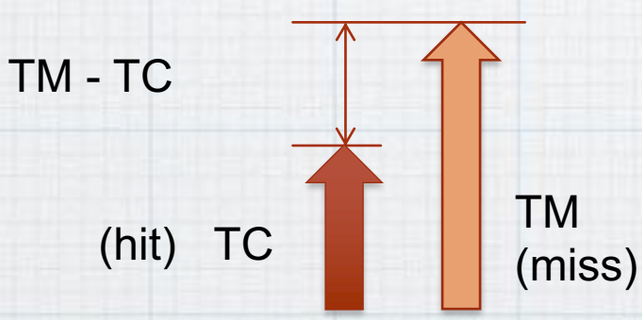
Lookaside Cache

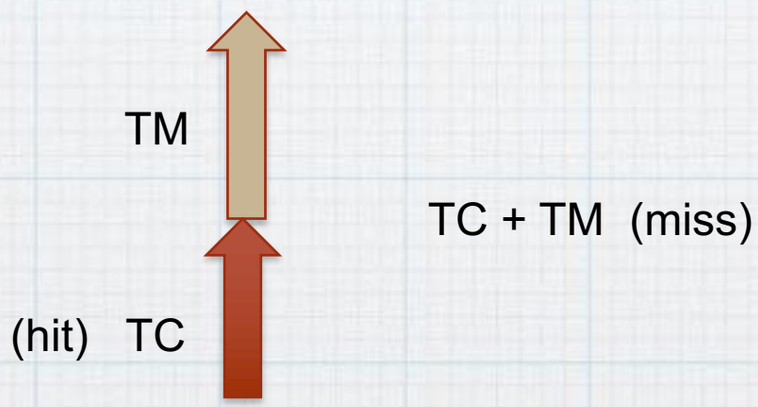
- Accesses to cache and to main memory occur in parallel
- Main memory access is cancelled if hit in cache occurs
- Tends to lower average memory access time
- Increases CPU to memory traffic



TM (memory access time)
TC (cache access time)

Look Through Cache

- First level cache is checked first
- Next level is only checked if miss occurs
- Tends to increase average memory access time
- Avoids unneeded CPU-to-memory traffic
- AMAT = h*TC + (1-h)*(TC+TM) = TC + (1-h)*TM



Consider a system with a main memory access time of 50 ns supported by a cache having a 8 ns access time and a hit rate of 95%.

What is the average memory access time for a look aside cache?

What is the average memory access time for a look through cache?

Consider a system with a main memory access time of 50 ns supported by a cache having a 8 ns access time and a hit rate of 95%.

What is the average memory access time for a look aside cache?

8*0.95 + 50*0.05 = 10.1 ns

What is the average memory access time for a look through cache?

8 + 50*0.05 = 10.5 ns

Consider a system with a main memory access time of 50 ns supported by an L1 cache having a 8 ns access time and a hit rate of 95% and an L2 cache having a 15 ns access time and a hit rate of 90%

What is the average memory access time for a look aside cache?

What is the average memory access time for a look through cache?

Consider a system with a main memory access time of 50 ns supported by an L1 cache having a 8 ns access time and a hit rate of 95% and an L2 cache having a 15 ns access time and a hit rate of 90%

What is the average memory access time for a look aside cache?

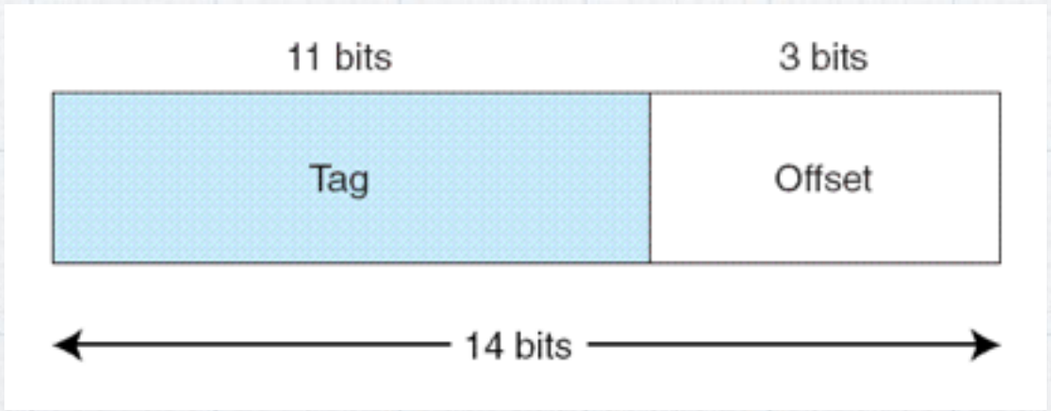
8*0.95 + 0.05*(15*0.9 + 50*0.1) = 7.6 + 0.05*(18.5) = 8.525 ns

What is the average memory access time for a look through cache?

8 + 0.05*(15 + 50*0.1) = 8 + 0.05*(20) = 9 ns

Fully Associative

- Mapping is based on dividing address into two fields
  - Tag field and offset



Example: 14-bit addresses and a cache containing 16 lines. Each line is 8 bytes in size.

A separate tag is stored for each cache line

If a stored tag matches the tag field in the address and the line is valid, there is a hit.

Fully Associative

Tiny example to help you understand the concepts:

Main memory is 2^4 = 16 bytes and is byte addressable. An address is 4 bits A block is 4 bytes. So there are 16 bytes/(4 bytes/block) = 4 blocks

4 blocks = 2^2 blocks so two bits describe the block

4 bytes/block mean the offset is 2 bits.

Our cache is 8 bytes. Since a line = block = 4 bytes, there are 2 lines

Tag Line

0000	
0001	
0010	Block 1
0011	
0100	
0101	Block 2
0110	
0111	
1000	
1001	
1010	Block 3
1011	
1100	
1101	
1110	Block 4
1111	

Fully Associative

A system has a main memory with 2^{30} bytes
It has a fully associative cache that is 2^{10} bytes
The block size is 32 bytes

How many bits are in the address?

How many bits are in the offset? How many in the tag?

How many lines are in the cache?

A program wants to access the data at memory address 0xABCD0123. What tag will be looked for in the cache?

Fully Associative

A system has a main memory with 2^{30} bytes
It has a fully associative cache that is 2^{10} bytes
The block size is 32 bytes

How many bits are in the address? 30

How many bits are in the offset? How many in the tag?

$\log_2(32) = 5$ 30-5 = 25

How many lines are in the cache?

$2^{10} / 2^5 = 2^5 = 32$ lines

A program wants to access the data at memory address 0x2BCD0123. What tag will be looked for in the cache?

30 bit address = 10 1011 1100 1101 0000 0001 0010 0011 Tag = 10 1011 1100 1101 0000 0001 001

Fully Associative

A system has a main memory with 2^{10} bytes
It has a fully associative cache that is 2^5 bytes
The block size is 4 bytes

How many bits are in the address?
How many bits are in the offset? How many in the tag?
How many lines are in the cache?

Fully Associative

A system has a main memory with 2^{10} bytes
It has a fully associative cache that is 2^5 bytes
The block size is 4 bytes

How many bits are in the address? 10
How many bits are in the offset? How many in the tag?

$\log_2(4) = 2$ 10-2 = 8

How many lines are in the cache?

$2^5 / 2^2 = 2^3 = 8$ lines

Fully Associative

A system has a main memory with 2^{10} bytes
It has a fully associative cache that is 2^5 bytes
The block size is 4 bytes

How many bits are in the address?
How many bits are in the offset? How many in the tag? 8
How many lines are in the cache? 8

The cache is initially empty, the access to the following addresses is requested: 101010101010, 1011101100, 1111100010, 1010101000

What is the state of the cache after those accesses?

Line	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Fully Associative

A system has a main memory with 2^{10} bytes
It has a fully associative cache that is 2^5 bytes
The block size is 4 bytes

How many bits are in the address?
How many bits are in the offset? How many in the tag? 8
How many lines are in the cache? 8

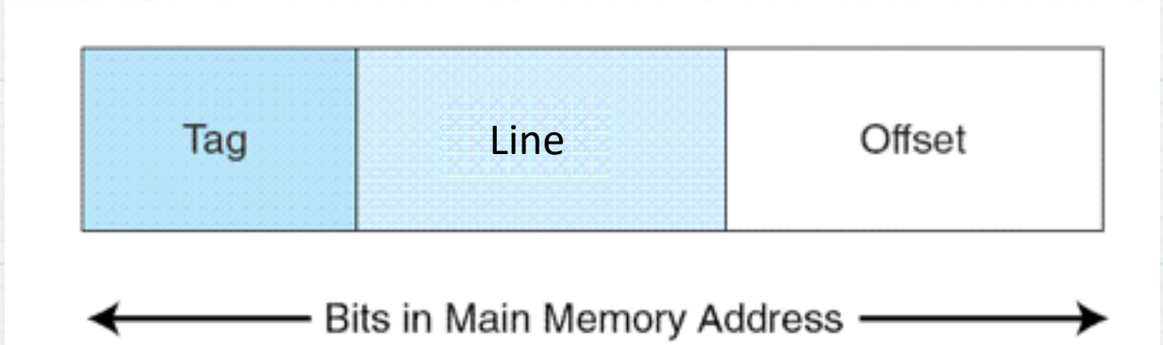
The cache is initially empty, the access to the following addresses is requested: 101010101010, 1011101100, 1111100010, 1010101000

What is the state of the cache after those accesses?

Line	Tag	Data
0	10101010	
1	10111011	
2	11111000	
3		
4		
5		
6		
7		

Direct Mapped

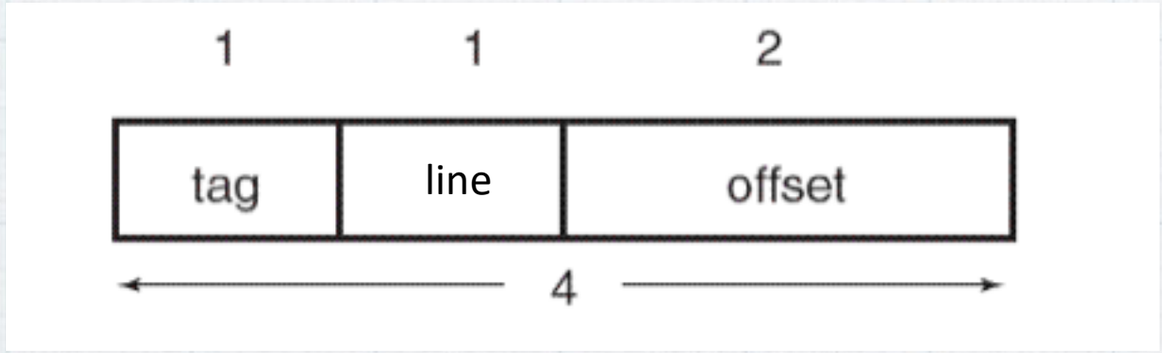
- Address is split into 3 fields for direct mapping
  - The *offset* field identifies location within line
  - The *line* field selects a unique line of cache
  - The *tag* field is whatever is left over.



- Offset width OW = $\log_2$ line size
- Line# width LW = $\log_2$ number of lines
- Tag width = (Bits in address) - LW - OW

Direct Mapped

- EXAMPLE: byte-addressable main memory consisting of 4 blocks with 4-byte block size. Cache contains 2 lines.
- Block 0 and 2 of main memory map to line 0 of cache, and Blocks 1 and 3 of main memory map to line 1 of cache.
- Address format is:

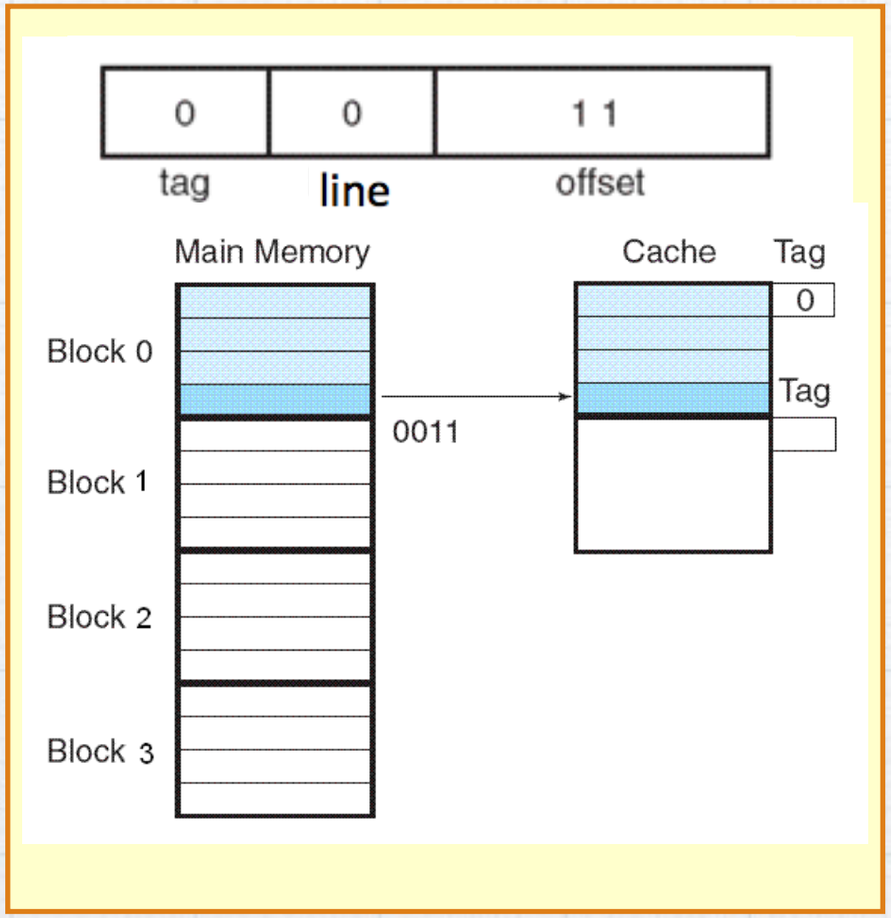


Direct Mapped

Address 3 is in memory block 0 and maps to cache line 0

Block# = 3 / 4 = 0

line# = 0 mod 2 = 0



Cache has 2 lines:
line0 and line 1

Direct Mapped - Sample Problem

A system has a main memory of 2^{32} bytes

It has a direct mapped cache that is 2^{12} bytes, and a block size of 16 bytes

How many bits are in the address?

How many lines are in the cache?

How many bits are in the offset?

How many bits are in the line field?

A program wants to access the data at address 0xABCD0123.
What is the line number?
What is the tag?

Direct Mapped - Sample Problem

A system has a main memory of 2^{32} bytes

It has a direct mapped cache that is 2^{12} bytes and a block size of 16 bytes

How many bits are in the address? 32

How many lines are in the cache? 2^{12} bytes/cache / 2^4 bytes/line = 2^8 lines/cach

How many bits are in the offset? $\log_2(2^4) = 4$

How many bits are in the line field? $\log_2(2^8) = 8$

A program wants to access the data at address 0xABCD0123. What is the line number? What is the tag?

10101011110011010000 00010010 0011