CE/CZ 3001:
Advanced Computer Architecture

Module 5: Memory Systems
- cache design

Asst Prof Liu Weichen
School of Computer Science and Engineering
Nanyang Technological University, Singapore

Overview

- Cache design
- Write and replacement policy
- Instruction vs Data cache
- Cache performance

Cache Design

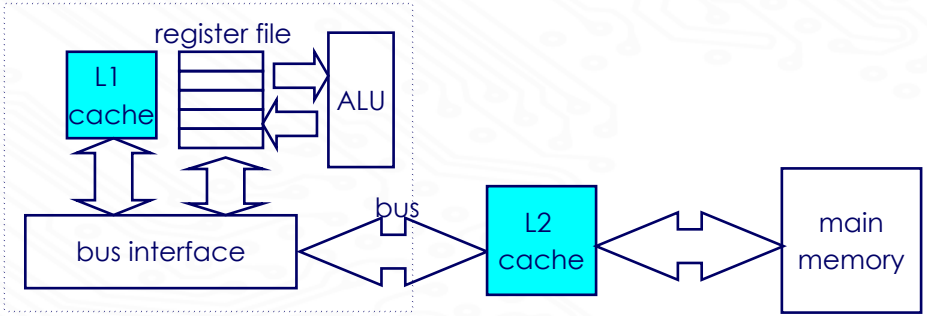
Big picture : memory hierarchy implementation (Part 1/2)

- Memory:
  - Ocean of bits
  - Many technologies are available (SRAM, DRAM, Magnetic disks)
- Key issues:
  - Placement (where bits are stored)
  - Identification (finding the right bits)
  - Replacement (finding space for new bits)
  - Write policy (propagating changes to bits)

Big picture : memory hierarchy implementation (Part 2/2)

- Burning questions:
  - Placement(Where can a block of memory be placed?)
  - Identification(How do I find a block of memory?)
  - Replacement(Which block should I remove if there is a miss?)
  - Write Policy(How do I propagate changes or keep memory updated?)

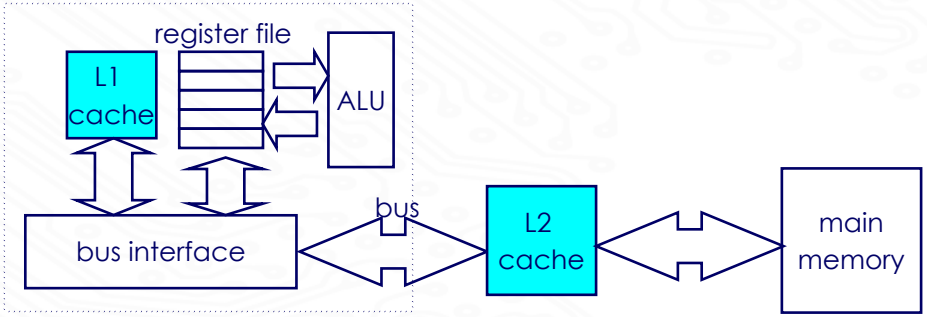
How placement is done in cache? (Part 1/2)



- **Level 1 cache (L1 cache)**
  - Generally very small
  - Typically 8K or 16K
  - Resides in the processor
- **Level 2 cache(L2 cache)**
  - Typically 256K or 512K
  - Resides between CPU and main memory
  - Slower than L1 cache

Modern CPUs normally have built-in L1, L2 and even L3 cache.

How placement is done in cache? (Part 2/2)

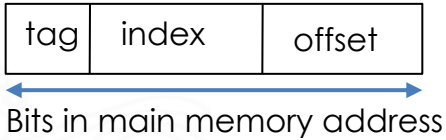


- **What makes cache special?**
  - It is not accessed by address
  - It is accessed by content
  - Content addressable memory (CAM)

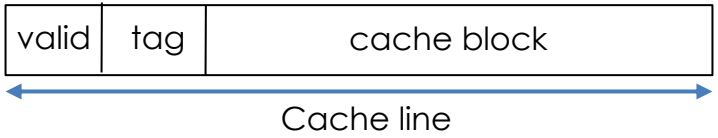
- **Hence cache mapping schemes**
  - The cache entries are checked to see the value requested is stored in cache
  - Needs to convert the main memory address into the cache location

How do you address the cache?

- Main memory and cache are divided into blocks of the same size.
- Each block maps to a location in the cache, determined by the index bits in the main memory address.



- In cache the cache lines are divided into different fields.



- the valid bit pertaining to each cache line tells status of the data in the cache – indicates if the data is valid.

CE/CZ 3001:
Advanced Computer Architecture

Module 5: Memory Systems

- cache organization schemes

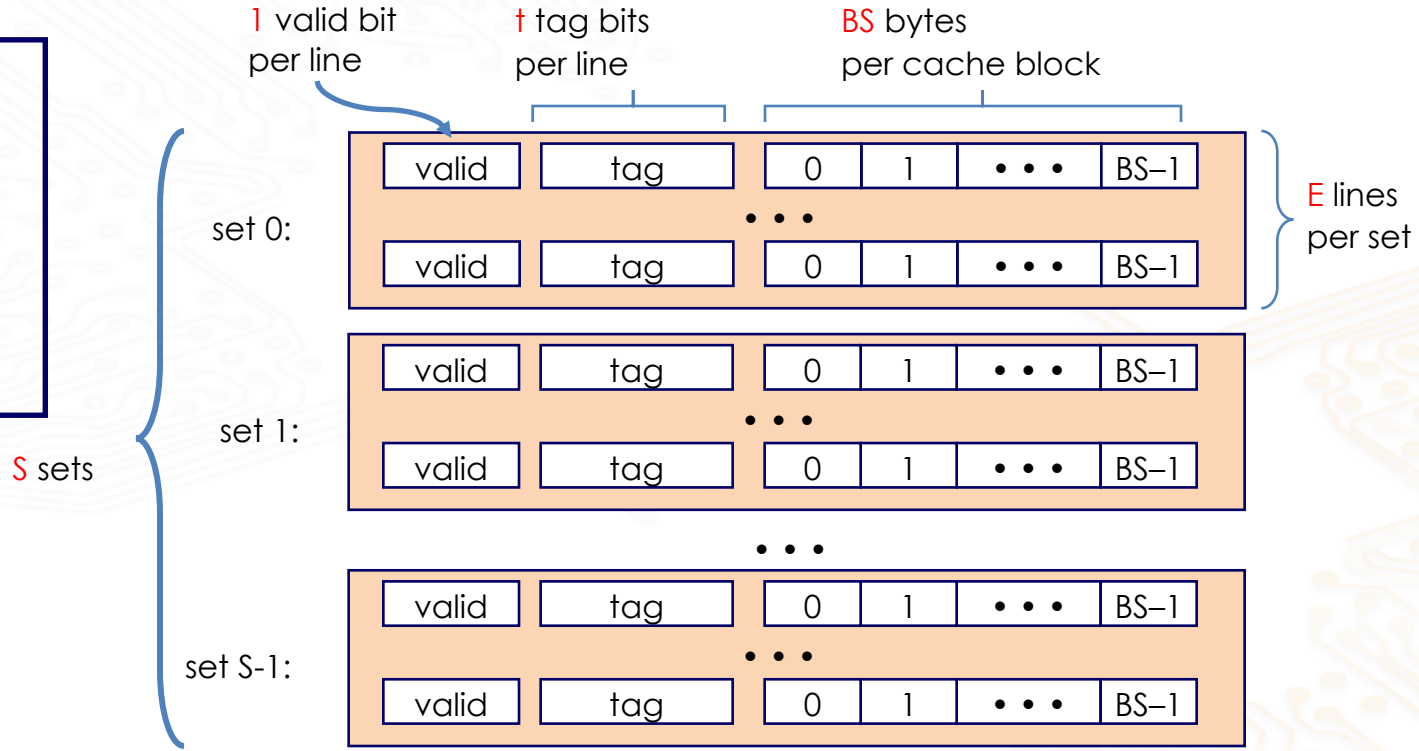
Asst Prof Liu Weichen
School of Computer Science and Engineering
Nanyang Technological University, Singapore

General Organization of a Cache (Part 1/3)

Cache is an array of sets

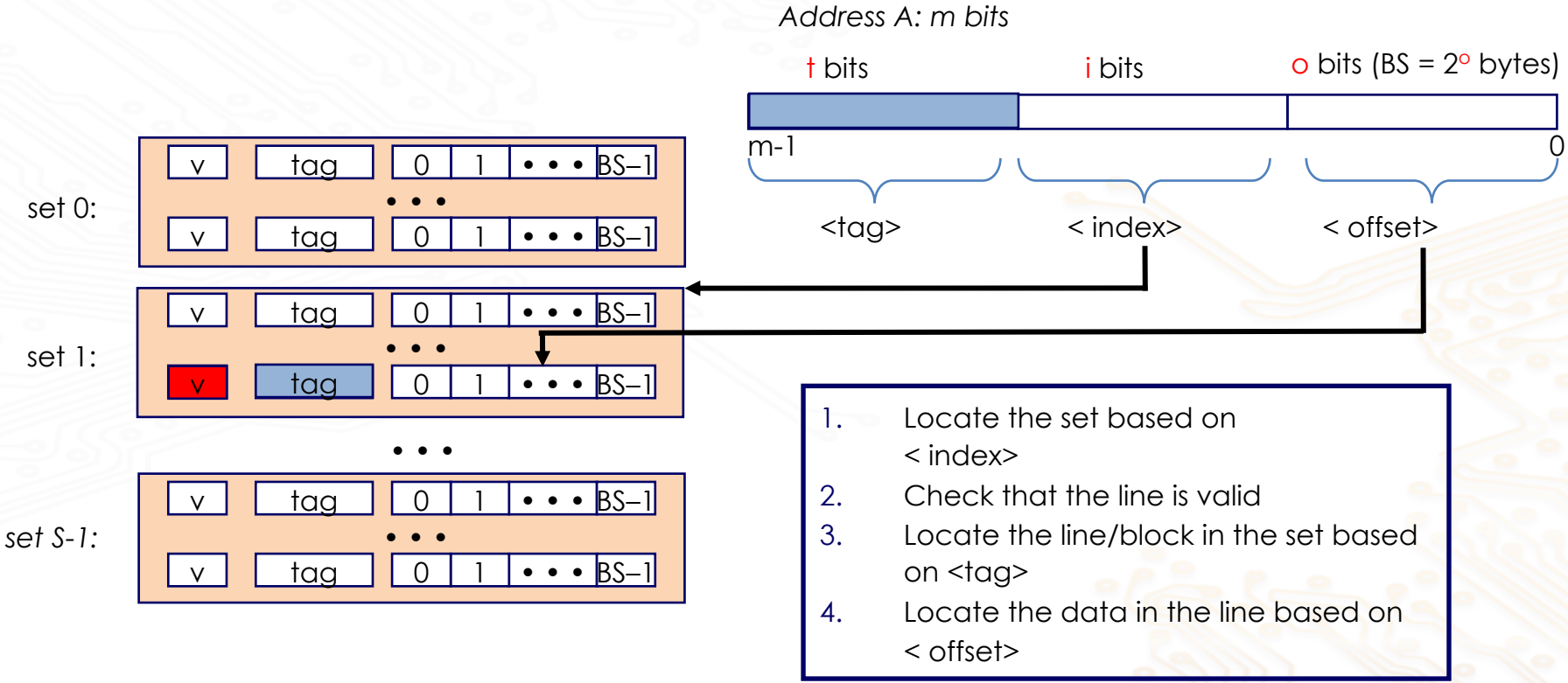
Each set contains one or more lines

Each line holds a block of data



Cache size: C = (1 bit + t bits + BS bytes) x E x S

General Organization of a Cache (Part 2/3)



1. Locate the set based on < index>
2. Check that the line is valid
3. Locate the line/block in the set based on <tag>
4. Locate the data in the line based on < offset>

General Organization of a Cache (Part 3/3)

- How to determine the tag, index and offset bits in main memory address?
- Given
  - address size of main memory (for example, 64 bits for ARMv8)
  - block size (BS), number of sets(S) and number of lines(E)

Consider 64-bit Address: eg: ARMv8

Portion	Length	Purpose
Offset	o=$\log_2$(block size)	Select word within block/line
Index	i=$\log_2$(number of sets)	Select the set
Tag	t=(address size) - o - i	ID block/line within set

Cache design: placement policy (Part 1/2)

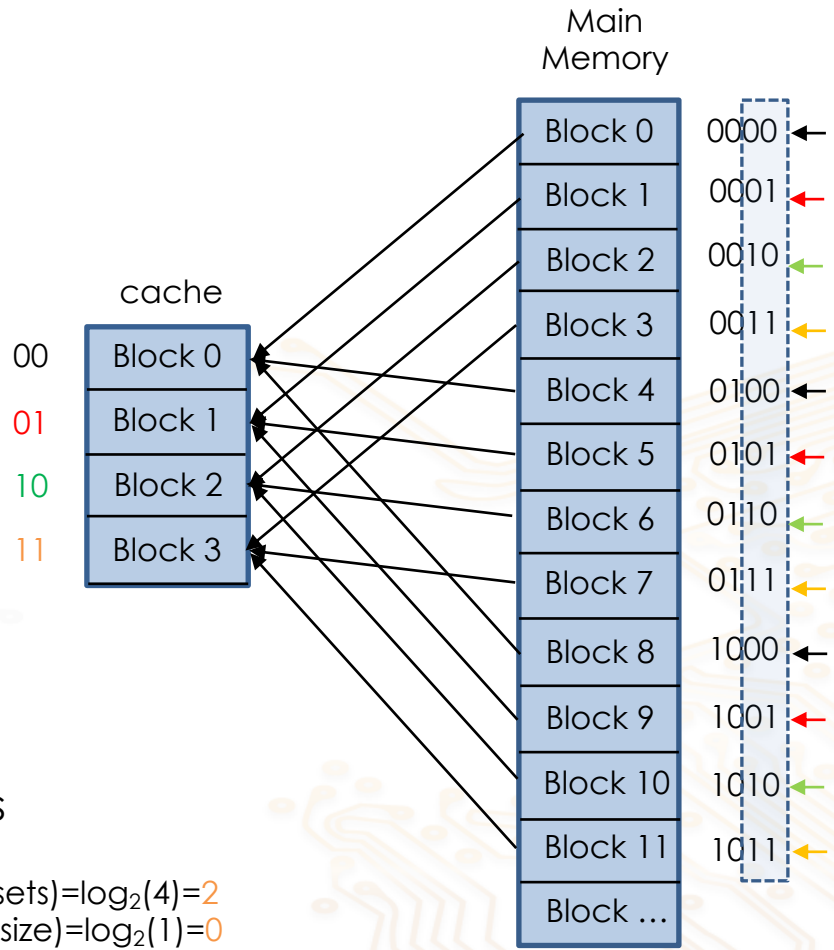
- Mapping of the main memory blocks to the cache lines
  - To decide where in the cache a copy of a selected memory block will reside.
  - The mapping schemes determines where data is placed when it is originally copied into cache.
  - It also provides a method for CPU to find previously copied data when searching cache.

Cache design: placement policy (Part 2/2)

- Direct-mapped cache
  - Simplest and fastest address mapping.
  - A memory block is mapped to a specific cache line only.
- Fully associative cache
  - A memory block can be placed in any of the cache lines.
  - Flexible but expensive.
- Set-associative cache
  - Combines the strategy of fully associative and direct mapped caches.
  - Popularly used.

Direct-mapped cache

- Cache consists of an array of fixed size frames called cache line/blocks.
- Each frame holds a block (consisting of consecutive bytes) of main memory.
- Direct mapping: each memory block is mapped to a specific cache line.
- Set of memory blocks with the same cache-index are mapped to the same cache line.
- Direct mapping: No of lines in a set, E=1
- Cache size (roughly): C = BS x S data bytes



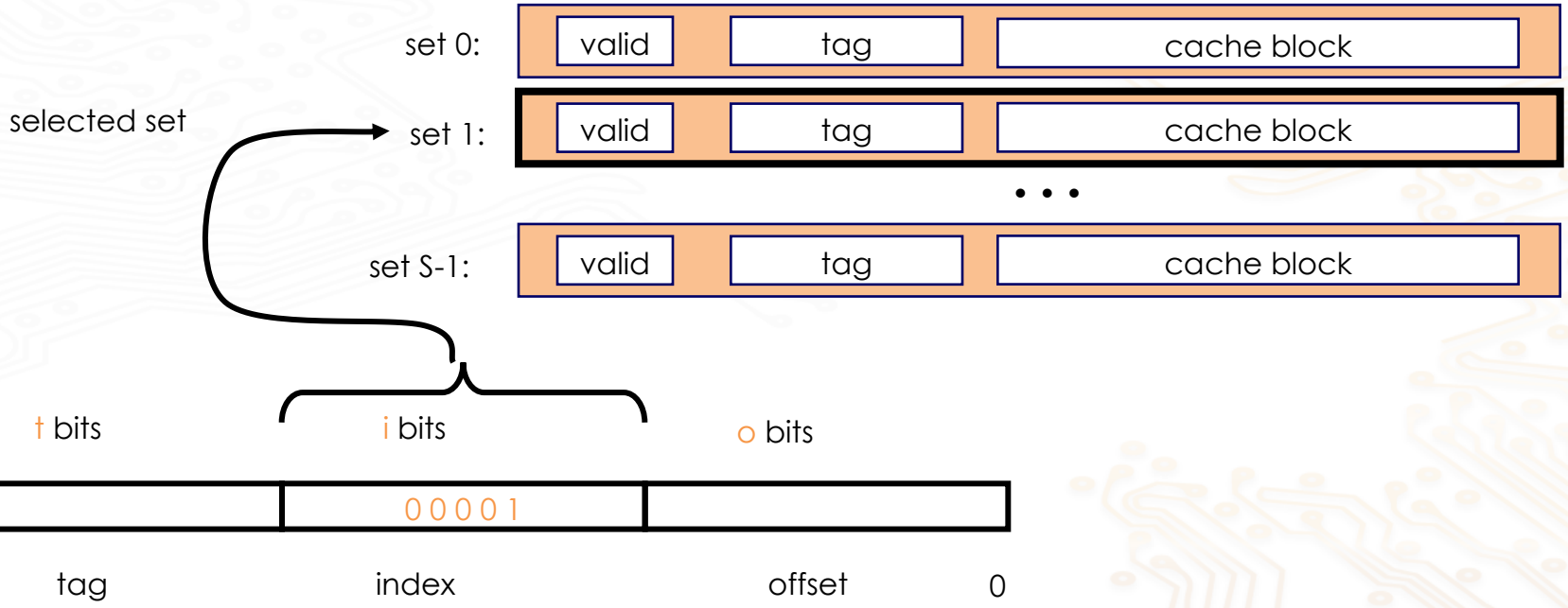
For this example: E=1, Address size= 4 bits, Number of sets (S) =4, block size (BS) =1

Index=$\log_2$(no of sets)=$\log_2(4)=2$
Offset=$\log_2$(block size)=$\log_2(1)=0$
Tag=Address size-(index + offset)=4-(2+0)=2

Accessing direct-mapped cache (Part 1/2)

Set selection

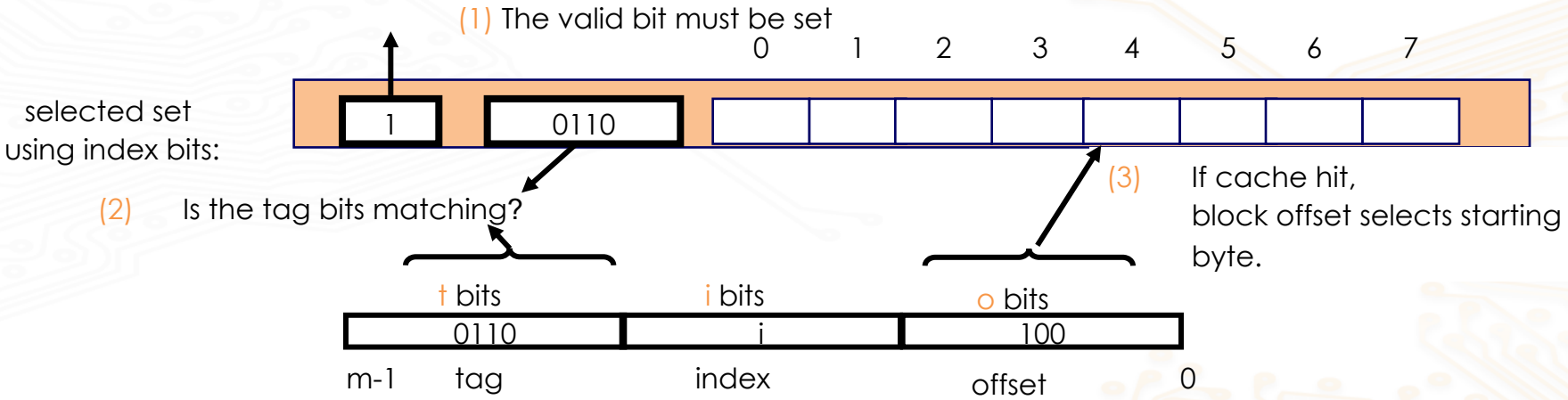
- Use the set index bits to determine the set of interest.



Accessing direct-mapped cache (Part 2/2)

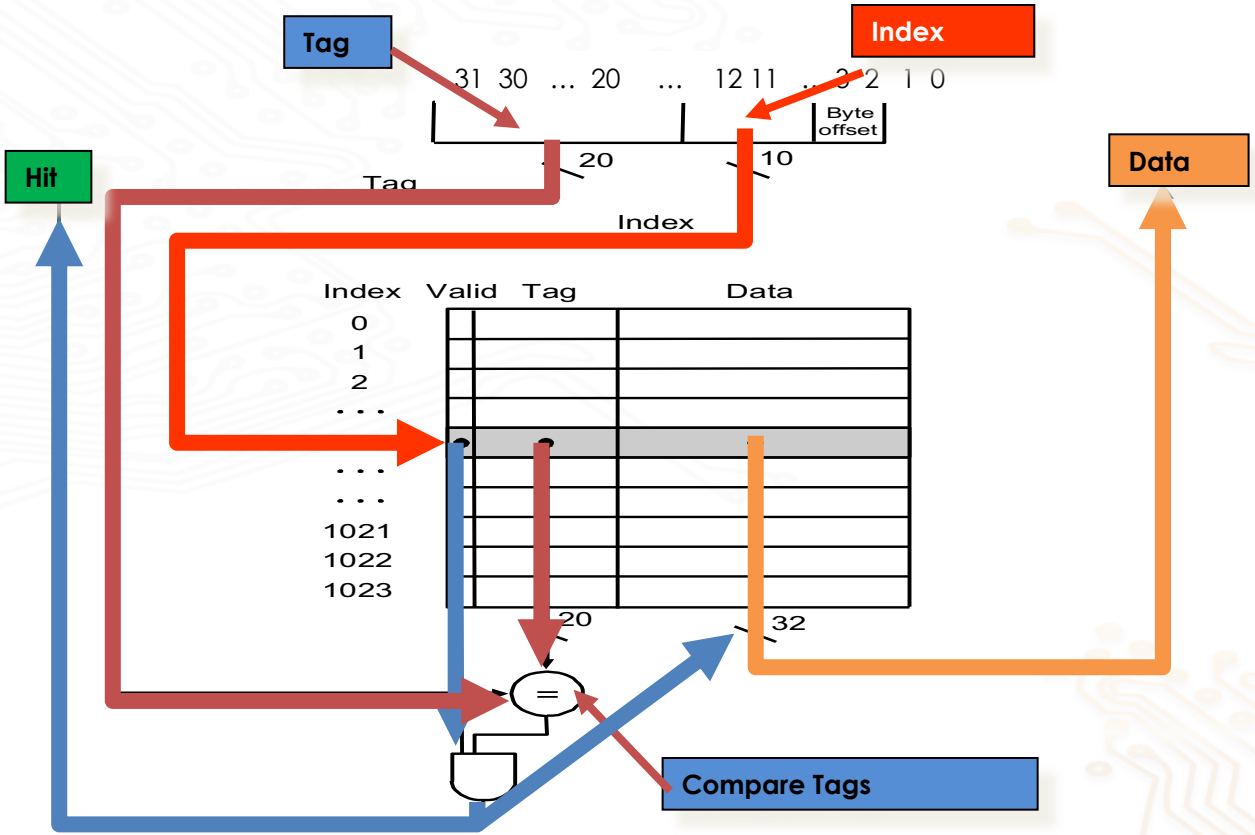
Line matching and word selection

- Line matching: Find a valid line in the selected set with a matching tag.
- Word selection: Then extract the word.



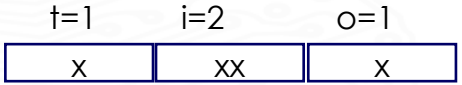
If (1) and (2), then cache hit

Direct-mapped cache: MIPS Example



Accessing direct-mapped cache

Offset=$\log_2$(block size)=1
index=$\log_2$(number of sets)=2
tag=4 - o – i=1



v	tag	data
1	0	M[0-1]
1	0	M[6-7]

M=16 byte main memory,
Block size(BS)=2 bytes/block,
Sets(S)=4 sets,
E=1 entry/set.

Hence the size of address =4 bit.
Total number of cache blocks= E*S=4 blocks

Address trace (reads):

0 **[0000_2],** miss
1 **[0001_2],** hit
7 **[0111_2],** miss
8 **[1000_2],** miss
0 **[0000_2]** miss

Direct-mapped cache

Summary

- Not very expensive as mapping requires no searching
- Hence each main memory block has a unique mapped location in cache
- Direct mapped cache implements a mapping scheme that results in main memory blocks being mapped in a modular fashion
  - How many bits are in main memory address (determined by how many addresses exist in main memory)
  - How many blocks are there in cache(number of sets)
  - How many bytes are there in one block (offset)

Any disadvantages?

Direct-mapped cache :- disadvantages

Offset=$\log_2$(block size)=1
index=$\log_2$(number of sets)=2
tag=4 - o – i=1

t=1	i=2	o=1
x	xx	x

Memory mapping to cache

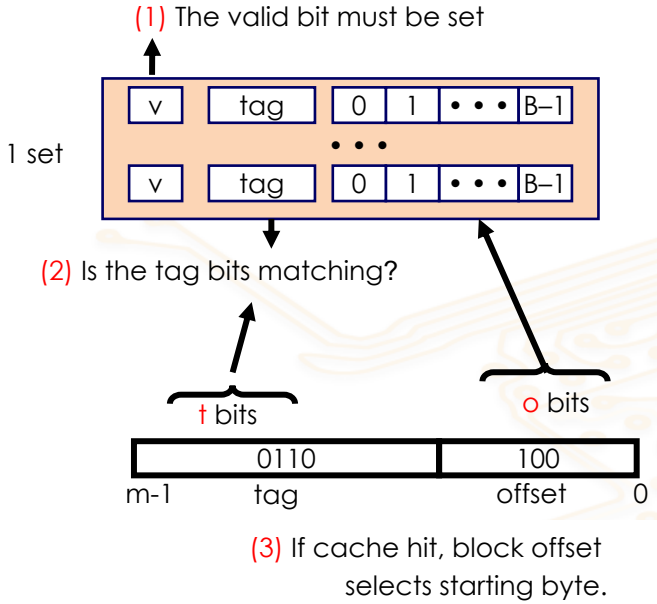
Main memory		Cache
block 0(addresses 0,1)	-	block 0
block 1(addresses 2,3)	-	block 1
block 2(addresses 4,5)	-	block 2
block 3(addresses 6,7)	-	block 3
block 4(addresses 8,9)	-	block 0
block 5(addresses 10,11)	-	block 1
block 6(addresses 12,13)	-	block 2
block 7(addresses 14,15)	-	block 3

M=16 byte addresses,
Block size=2 bytes/block,
Sets=4 sets (4 blocks),
E=1 entry/set.
Hence the size of address =4 bit.
Total Number of cache blocks=S*E=4 blocks

- Sequence of operation:
  Block 0, 4, 0, 4, 0, 4…
- Block 0 and 4 map to same location
- Program will repeatedly throw out 4 and 0 to bring the other
- Can be solved by allowing the block to be placed anywhere in the cache

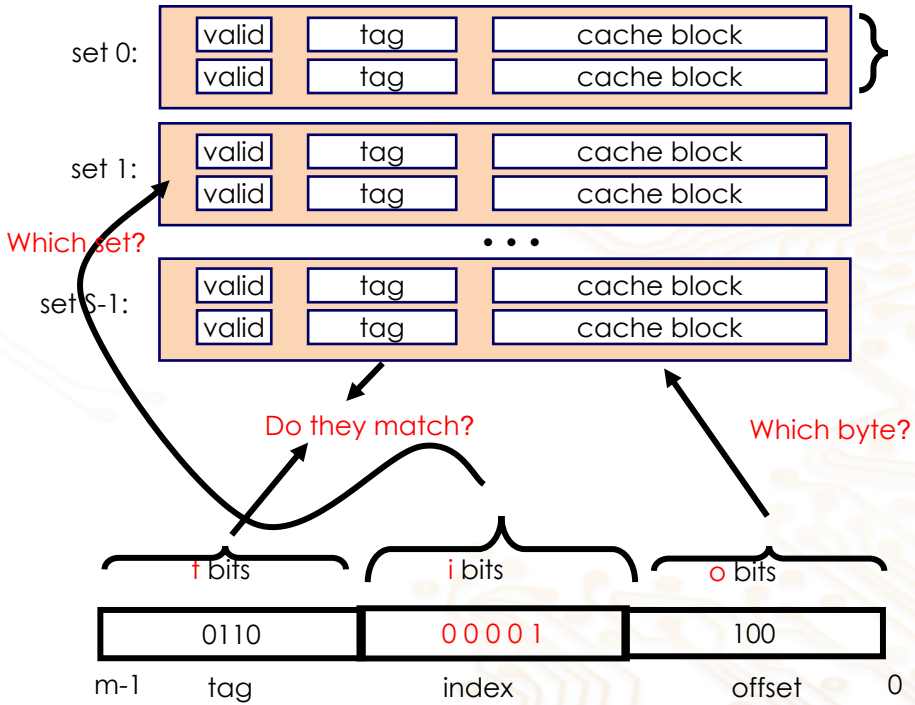
Fully associative cache

- Allows the main memory block to be placed anywhere in cache.
- The only way to find it – search all the cache.
- Cache need to be built from associative memory to search in parallel.
- Search compares the requested tag with all tags in cache to find the desired block.
- Expensive as associated memory needs more hardware.
- Here the main memory needs to be partitioned as tag and offset as there is no need for index(only one set).
- It requires longer tag (disadvantage)

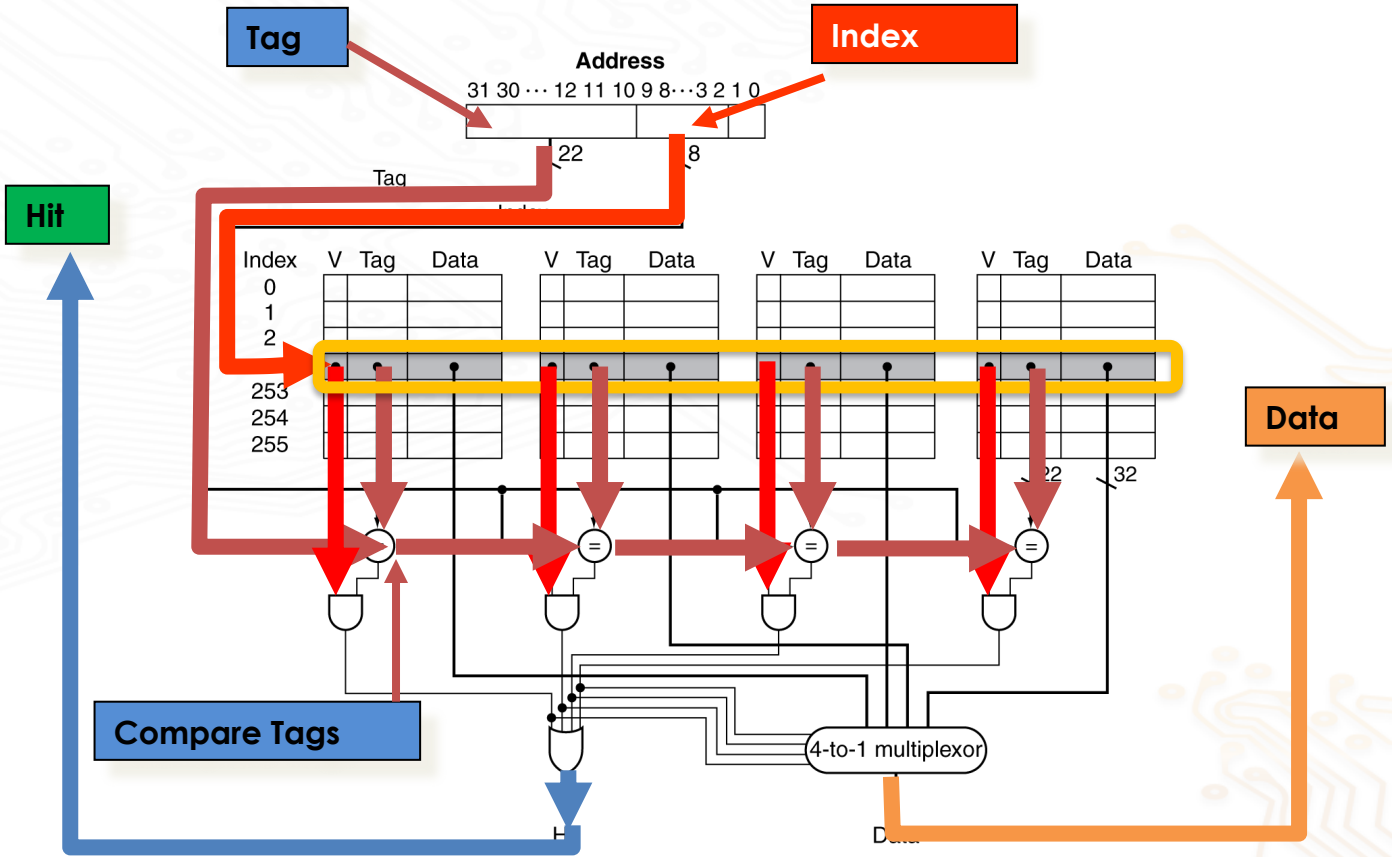


Set associative cache

- Why set associative?
  - Due to speed and complexity fully associative is expensive.
  - Direct mapping is inexpensive but very restricted.
- Better to have a combination of both.
- Similar to direct mapping but more lines per set.
- Needs three fields for main memory address as it has multiple sets and multiple lines per set.



Set associative cache: MIPS Example



Spectrum of associativity

For a cache with 8 entries

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Summary

- Overview of memory hierarchy
- Cache design
  - Direct mapped
  - Set associative
  - Fully associative