

Cache: Direct Mapping Address Structure

Direct Mapping Fundamentals

- Main Memory Structure:
 - Memory is divided into blocks (eg. 64 blocks)
 - Cache is made of lines
- Cache Mapping Formula:
 - A memory block maps to a specific cache line
$$\text{cache line index} = (\text{block \#}) \bmod (\text{\# of lines})$$

Example:

- Main Memory has 32 blocks (0-31), cache has 4 lines (0-3) * Unrealistic Example
- Block 25 maps to line 1 ($25 \bmod 4 = 1$)
- Block 4 and block 16 both map to line 0
- Consequence: If block 4 is in line 0, and block 16 is loaded, then block 4 gets evicted

Tag and Line Number: Address Breakdown

- When we store to cache, we want to store the block # alongside the data data, to better be able to check the cache for hits/misses
- But we don't want to store the entire block # since that would take up too much space

If we have a 4-line cache, any block # that is a multiple of 4 will map to line 0
Likewise, any mem block # that is a (multiple of 4) + 1 will map to cache line 1

	Mem block #	block # binary
line 0:	block 0	0 0 0 0 0
	block 4	0 0 1 0 0
	block 8	0 1 0 0 0
	block 12	0 1 1 0 0
	⋮	⋮

	Mem block #	block # binary
line 1:	block 1	0 0 0 0 1
	block 5	0 0 1 0 1
	block 9	0 1 0 0 1
	block 13	0 1 1 0 1
	⋮	⋮

We notice that the last 2 digits are the same, so we can simply store the first 3 digits and add back the last 2 digits on the fly when we want to access the cache.
This "truncated" block # is called the **tag**

When a block is loaded into a cache line, its identity must be memoized using a **tag**

- Tag = top bits of the address (unique identifier)
- Line Number = lower bits (used to index into the cache)

General Formula

- Address size: S bits (memory size = 2^S)
- Cache size: 2^R lines (line index size = R bits)
- Tag size = $S - R$

Example:

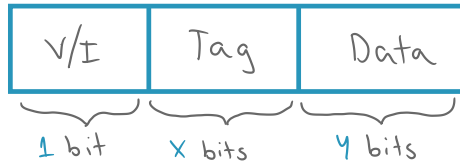
- Memory has 32 blocks = 5 bit address ($S = 5$)
- Cache has 4 lines = 2 bit line # ($R = 2$)
- Tag = $5 - 2 = 3$ bits

Cache Structure

Each cache line contains:

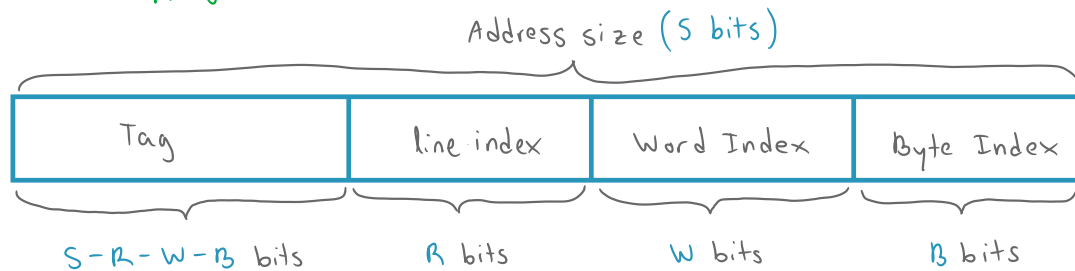
- Valid/invalid bit (V/I): Indicates if the data is valid
- Tag: Used for identifying the block stored
- Data: The contents of the block

Cache Line Format



Tag and data bit width will depend on the system being described

Direct Mapping Address structure



- S = Total address bits
- R = Bits for line index
- W = Bits for word index (If multiple words per block. If only 1 word per block, we don't include a partition for the word index)
- B = Bits for byte index (if the system uses byte addressing, otherwise we don't include this partition for the byte index)

Address Decomposition for Word and Byte Addressing

Scenarios:

- Word Addressing, 1 word per block



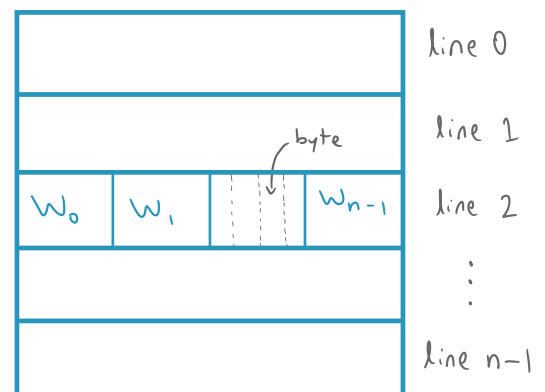
- Byte Addressing, 1 word per block, word size = 4 bytes



- Word Addressing, multiple words per block



- Byte Addressing, multiple words per block



Cache Initialization and Context Switching

- On program startup: all cache lines are marked invalid
- On context switch:
 - Previous process's cache data becomes irrelevant
 - All cache lines are again marked invalid
- Valid bit is set to 1 only when new block data is loaded

Total Cache Size Calculation

Formula:

$$\text{Total cache size} = \# \text{ of lines} \times (1 + \text{Tag bits} + \text{Data bits})$$

- Find the total # of bytes for a direct mapping cache to store 64 KB in 1-word blocks assuming a word size of 32 bits and MIPS addressing

↳ Break down the specifications:

- Word size = 32 bits → 32 bit address
- Block size = 1 word
- Addressing mode: byte addressing (from what we know of MIPS)
- Data: 64 KB

* b = bits

* B = bytes

$$\text{Block size} = 1 \text{ word} = 2^0 \text{ words}$$

↳ We don't need to allocate any bits for word select

$$\text{Word size} = 32 \text{ bits} \rightarrow 4 \text{ B} = 2^2 \text{ B} \rightarrow \text{Allocate } 2 \text{ bits for addressing byte \# (addressing)}$$

$$\text{Data} = 64 \text{ KB} = 64 \times 2^{10} \text{ B} = 2^6 \times 2^{10} \text{ B} = 2^{16} \text{ B}$$

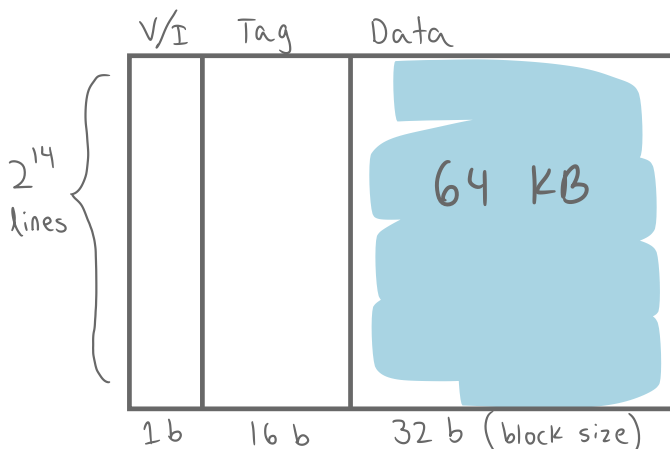
$$\# \text{ of lines} = \frac{\text{Data size}}{\text{block size}} = \frac{2^{16} \text{ B}}{2^2 \text{ B}} = 2^{14} \text{ lines}$$

$$\text{Tag} = 32 - (14 + 2) = 16 \text{ bits}$$

↳ Address Structure

Tag	line #	byte #
16 b	14 b	2 b

↳ Cache Structure



↳ Total cache size

$$\begin{aligned}
 &2^{14} \text{ lines} \times 39 \text{ bits/line} = (39 \cdot 2^{14}) \text{ bits} \\
 &= (39 \cdot 2^4)(2^{10}) \text{ bits} = (39 \cdot 16)(1 \text{ Kb}) = (39 \cdot 16) \text{ Kb} \\
 &= (39 \cdot 2) \text{ Kb} = \boxed{78 \text{ Kb}}
 \end{aligned}$$

→ 39 bits

Cache Initialization and Context Switching

- On program startup: all cache lines are marked invalid
- On context switch:
 - Previous process's cache data becomes irrelevant
 - All cache lines are again marked invalid
- Valid bit is set to 1 only when new block data is loaded

Total Cache Size Calculation

Formula:

$$\text{Total cache size} = \# \text{ of lines} \times (1 + \text{Tag bits} + \text{Data bits})$$

- Find the total # of bytes for a direct mapping cache to store 64 KB in 1-word blocks assuming a word size of 32 bits and MIPS addressing

↳ Break down the specifications:

- Word size = 32 bits → 32 bit address
- Block size = 1 word
- Addressing mode: byte addressing (from what we know of MIPS)
- Data: 64 KB

* b = bits

* B = bytes

$$\text{Block size} = 1 \text{ word} \rightarrow 1 \text{ word} \times 32 \text{ bits/word} = 32 \text{ b} \rightarrow 4 \text{ B} = 2^2 \text{ B}$$

↳ We don't need to allocate any bits for word select, since we have only 1

$$\text{Word size} = 32 \text{ bits} \rightarrow 4 \text{ B} = 2^2 \text{ B} \rightarrow \text{Allocate } 2 \text{ bits for addressing byte \# (byte addressing)}$$

$$\text{Data} = 64 \text{ KB} = 64 \times 2^{10} \text{ B} = 2^6 \times 2^{10} \text{ B} = 2^{16} \text{ B}$$

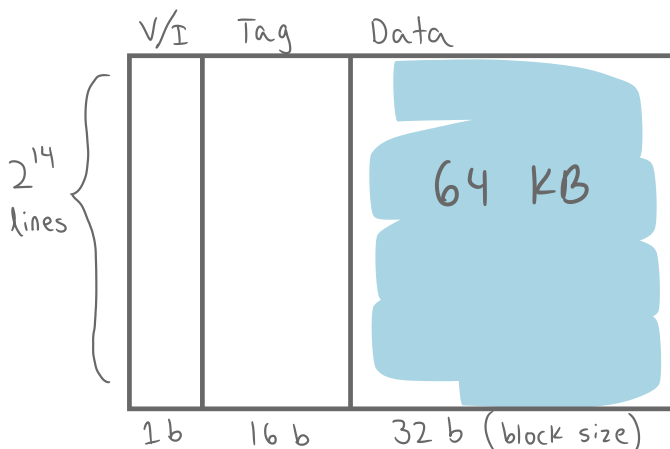
$$\# \text{ of lines} = \frac{\text{Data size}}{\text{block size}} = \frac{2^{16} \text{ B}}{2^2 \text{ B}} = 2^{14} \text{ lines}$$

$$\text{Tag} = 32 - (14 + 2) = 16 \text{ bits}$$

↳ Address Structure

Tag	line #	byte #
16 b	14 b	2 b

↳ Cache Structure



↳ Total cache size

$$\begin{aligned} 2^{14} \text{ lines} \times 39 \text{ bits/line} &= (39 \cdot 2^{14}) \text{ bits} \\ &= (39 \cdot 2^4)(2^{10}) \text{ bits} = (39 \cdot 16)(1 \text{ Kb}) = (39 \cdot 16) \text{ Kb} \\ &= (39 \cdot 2) \text{ Kb} = \boxed{78 \text{ Kb}} \end{aligned}$$