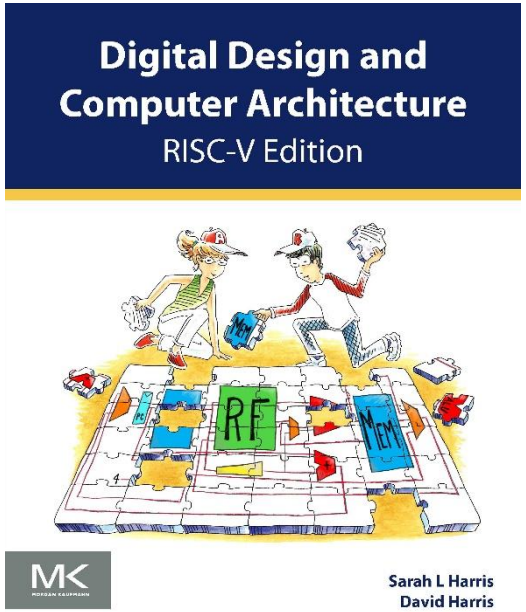
TU WIEN Informatics

Advanced Computer Architecture

C4 - Memory Systems

Johannes Kappes, Daniel Müller-Gritschneder

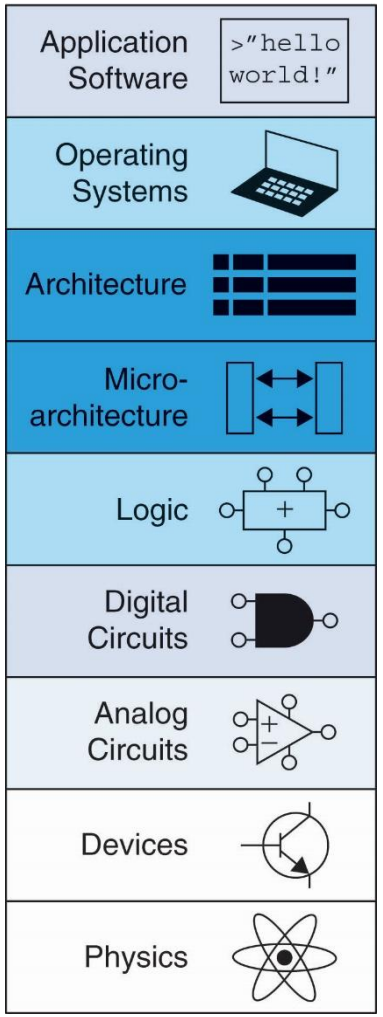
Sources



- Literature: „Digital Design and Computer Architecture: RISC-V Edition“, by Sarah L. Harris and David Harris
  - https://shop.elsevier.com/books/digital-design-and-computer-architecture-risc-v-edition/harris/978-0-12-820064-3
  - https://pages.hmc.edu/harris/ddca/ddcarv.html (Includes resources for students!)
  - They also provide slideshows – the basis for ours! You can investigate extended version at their website.
- Available at TU’s library: https://catalogplus.tuwien.at/permalink/f/qknpf/UTW_alma2113990399000336

Topics

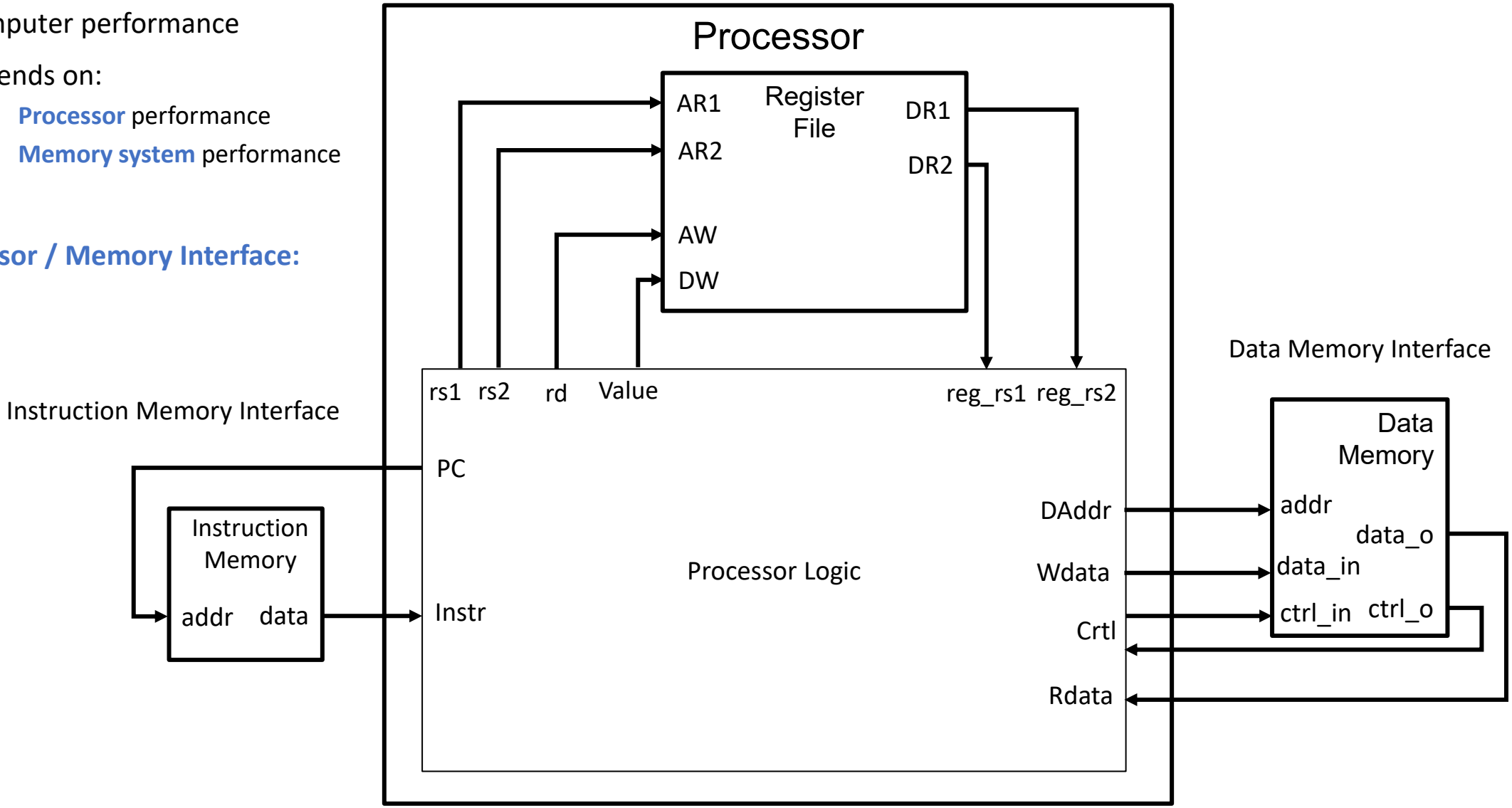
- Introduction
- Memory System Performance Analysis
- Cache Structure
- Cache Replacement Policy
- Cache Write/Read Hit/Miss Policy
- Summary



Introduction

- Computer performance
- depends on:
  - **Processor** performance
  - **Memory system** performance

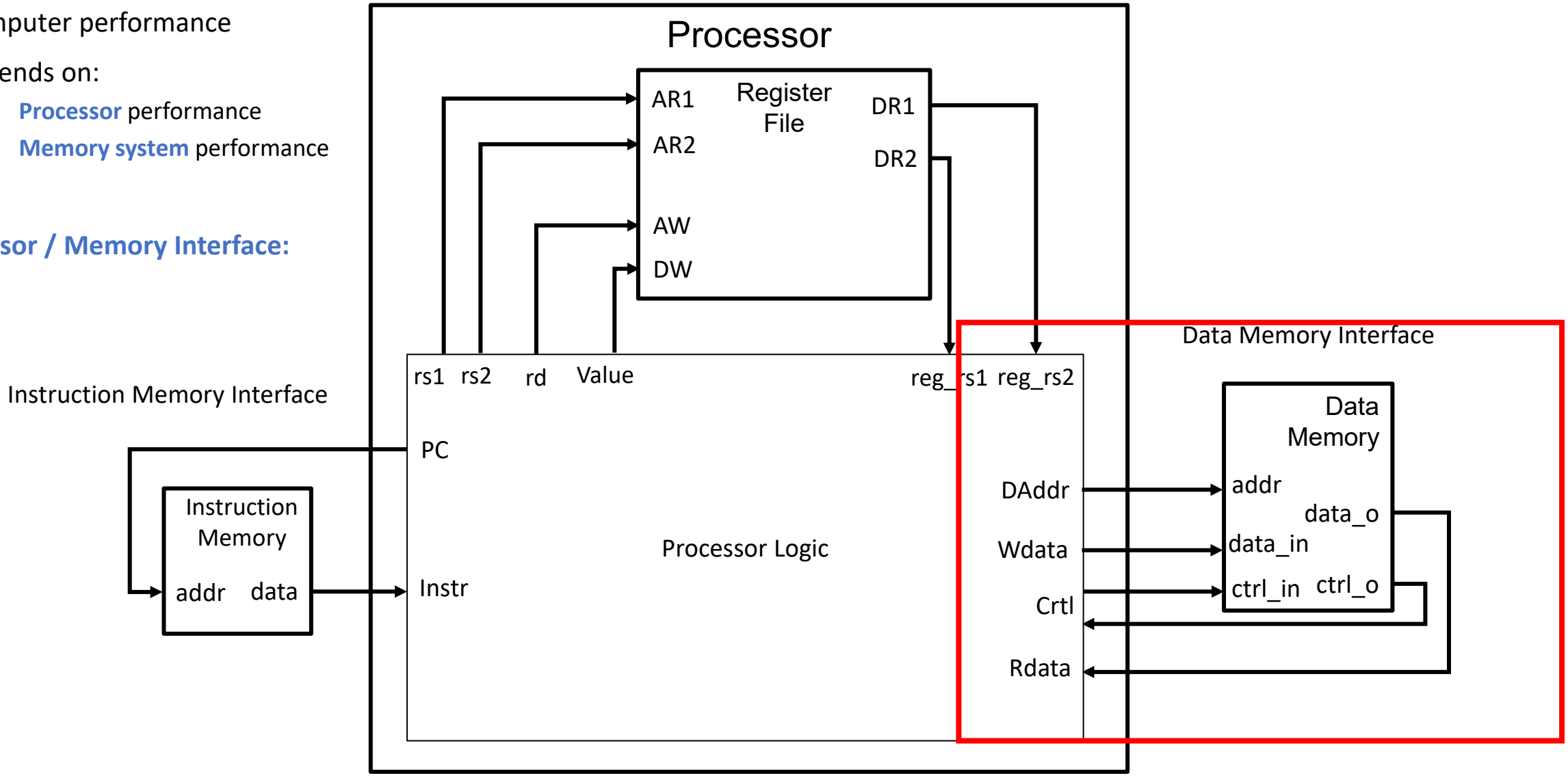
Processor / Memory Interface:



Introduction

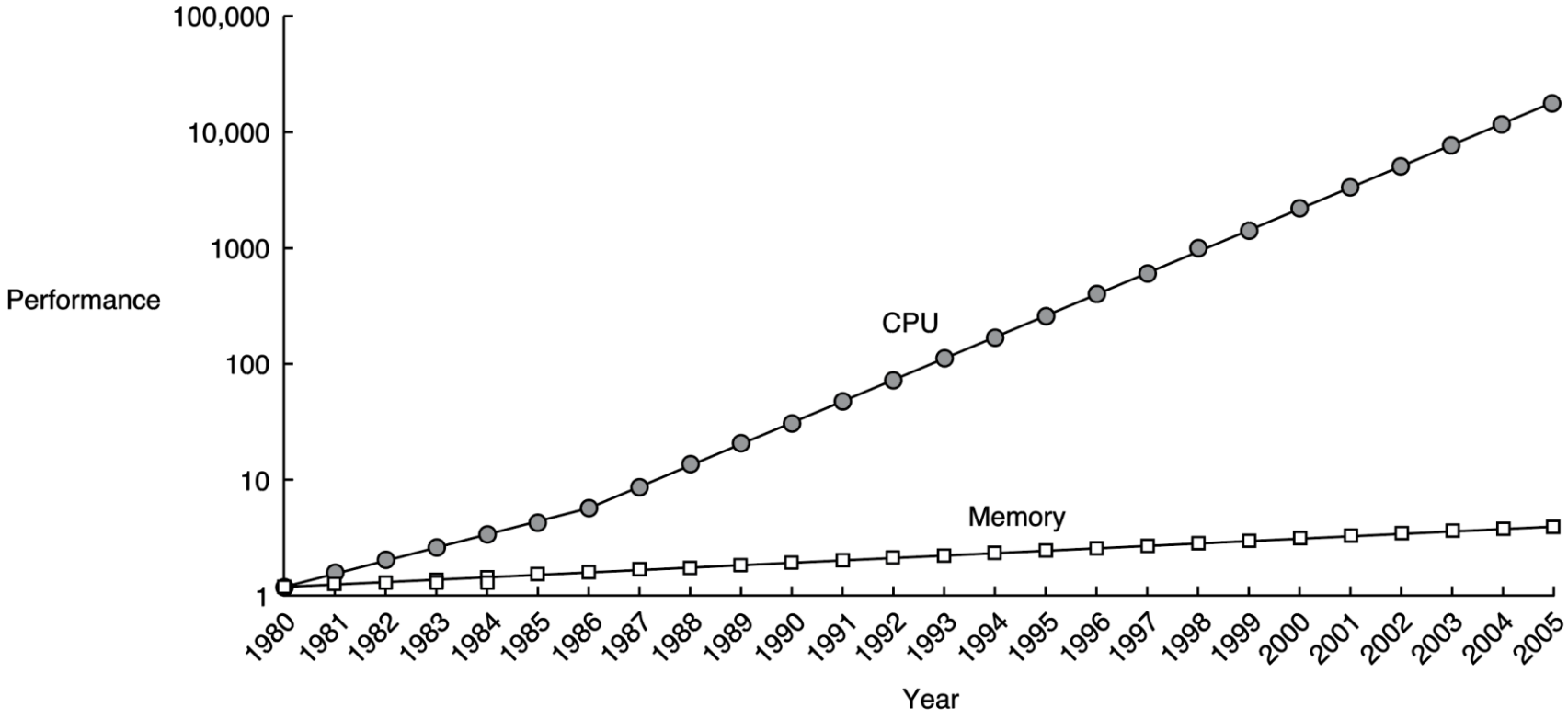
- Computer performance
- depends on:
  - **Processor** performance
  - **Memory system** performance

Processor / Memory Interface:



Processor-Memory Gap

- In prior chapters, we assumed access memory in 1 clock cycle
- This assumption hasn't been true since the 1980's.

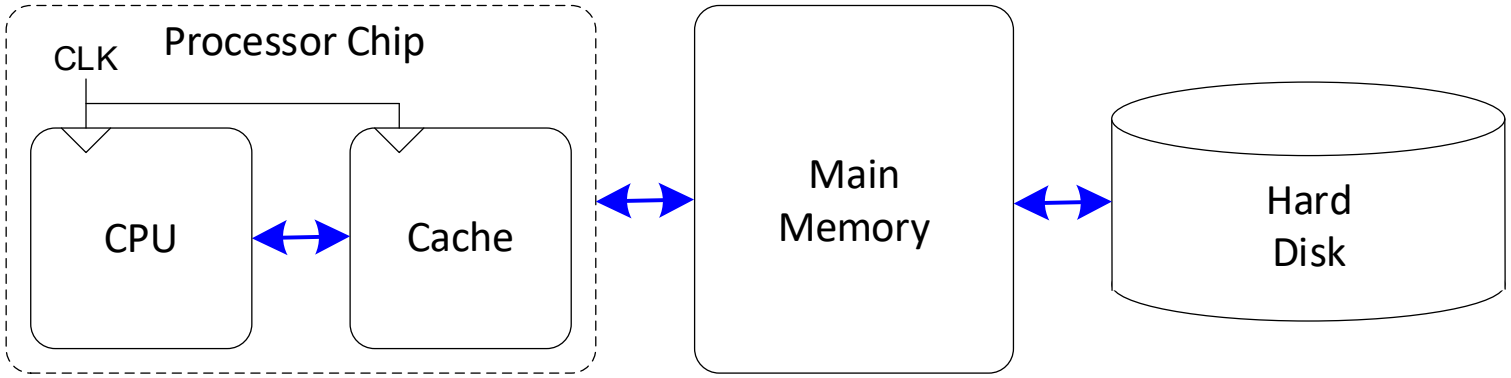


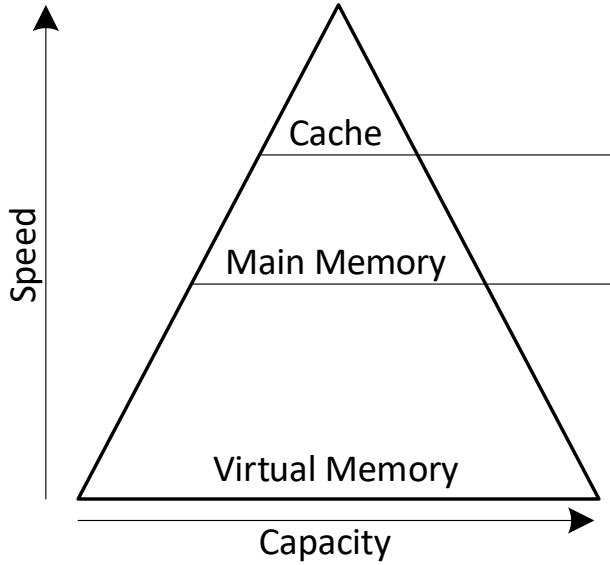
Memory System Challenge

- Make memory system appear as fast as processor
- Use hierarchy of memories
- Ideal memory:
  - **Fast**
  - **Cheap** (inexpensive)
  - **Large** (capacity)

- But we can only choose two!

Memory Hierarchy





Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$100	0.2 - 3	100+
DRAM	$3	10 - 50	30
SSD	$0.10	20,000	0.05 - 3
HDD	$0.03	5,000,000	0.001 - 0.1

Locality

- Exploit locality to make memory accesses fast:
- **Temporal Locality:**
  - Locality in time
  - If data used recently, likely to use it again soon
  - How to exploit: keep recently accessed data in higher levels of memory hierarchy
- **Spatial Locality:**
  - Locality in space
  - If data used recently, likely to use nearby data soon
  - How to exploit: when access data, bring nearby data into higher levels of memory hierarchy too

C4-1 Memory Performance

Memory Performance

- **Hit**: data found in that level of memory hierarchy
- **Miss**: data not found (must go to next level)

$$\boldsymbol{Hit\ Rate} = \frac{\#\ hits}{\#memory\ accesses} = 1\ - Miss\ Rate$$

$$\boldsymbol{Miss\ Rate} = \frac{\#\ misses}{\#memory\ accesses} = 1\ - Hit\ Rate$$

- Average memory access time (AMAT): average time for processor to access data

$$\boldsymbol{AMAT} = t_{cache} + MR_{cache}[t_{MM} + MR_{MM}(t_{VM})]$$

Memory Performance Example 1

- A program has $2{,}000$ loads and stores
- $1{,}250$ of these data values in cache
- Rest supplied by other levels of memory hierarchy
- What are the cache hit and miss rates?

$$Hit\ Rate = \frac{1250}{2000} = 0.625$$

$$Miss\ Rate = \frac{750}{2000} = 0.375 = 1\ - Hit\ Rate$$

Memory Performance Example 2

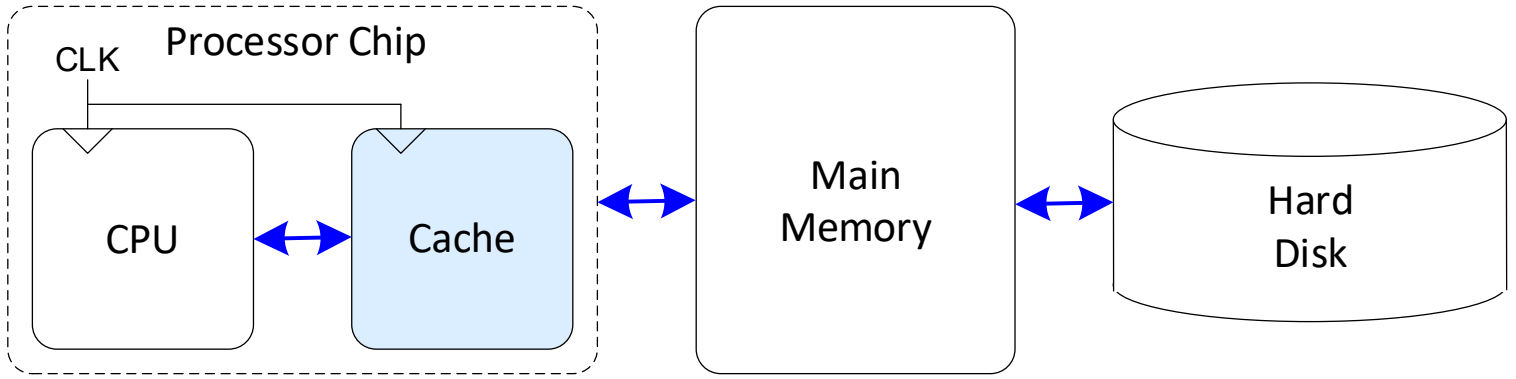
- Suppose processor has 2 levels of hierarchy: cache and main memory
- $t_{cache} = 1\ cycle, t_{MM} = 100\ cycles$
- What is the AMAT (average memory access time) of the program from Example 1?

$$AMAT = t_{cache} + MR_{cache}(t_{MM}) = [1 + 0.375(100)]\ cycles = 38.5\ cycles$$

C4-2 Caches

Cache

- Highest level in memory hierarchy
- Fast (typically ~ 1 cycle access time)
- Ideally supplies most data to processor
- Usually holds most recently accessed data



Cache Design Principles

- What data is held in the cache?
- How is data found?
- What data is replaced?

We focus on data loads, but stores follow the same principles.

What Data is Held in the Cache?

- Ideally, cache anticipates needed data and puts it in cache
- But impossible to predict future
- Use past to predict future – temporal and spatial locality:
  - **Temporal locality**: copy newly accessed data into cache
  - **Spatial locality**: copy neighboring data into cache too

Cache Terminology

- **Capacity (C)**
  - Number of data bytes in cache
- **Block size (b)**
  - Bytes of data brought into cache at once
- **Number of blocks (B)**
  - Number of blocks in cache
  - $B = \frac{C}{b}$
- **Degree of associativity (N)**
  - Number of blocks in a set
- **Number of sets (S)**
  - Each memory address maps to exactly one cache set
  - $S = \frac{B}{N}$

How is Data Found?

- Cache organized into S sets
- Each memory address maps to exactly one set
- Caches categorized by # of blocks in a set:
  - **Direct mapped**: 1 block per set
  - **N-way set associative**: N blocks per set
  - **Fully associative**: all cache blocks in 1 set
- Examine each organization for a cache with:
  - Capacity (C = 8 words)
  - Block size (b = 1 word)
  - So, number of blocks (B = 8)

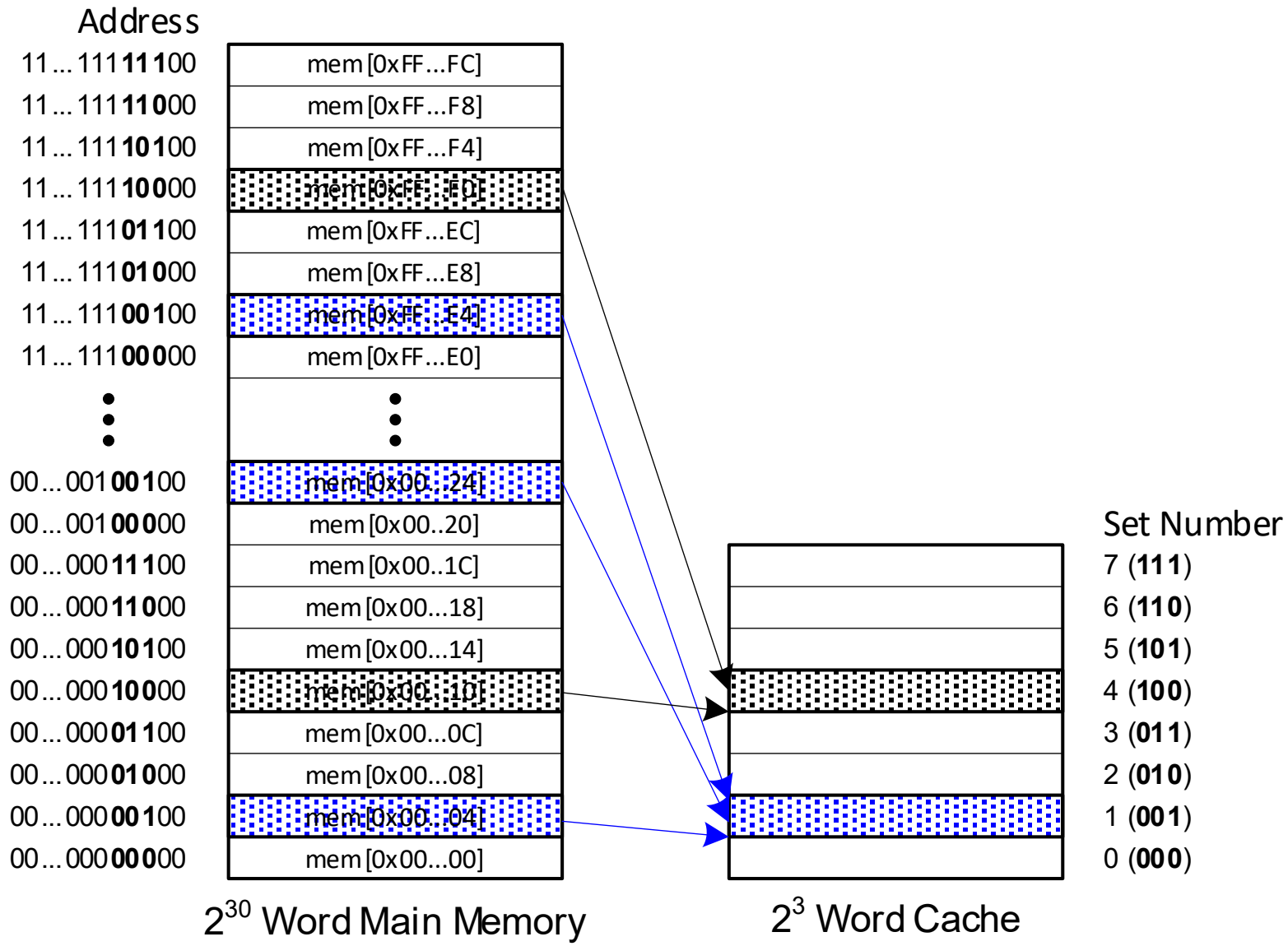
Example of Cache Parameters

- $\boldsymbol{C = 8}$ words (capacity)
- $\boldsymbol{b = 1}$ word (block size)
- So, $\boldsymbol{B = 8}$ (# of blocks)

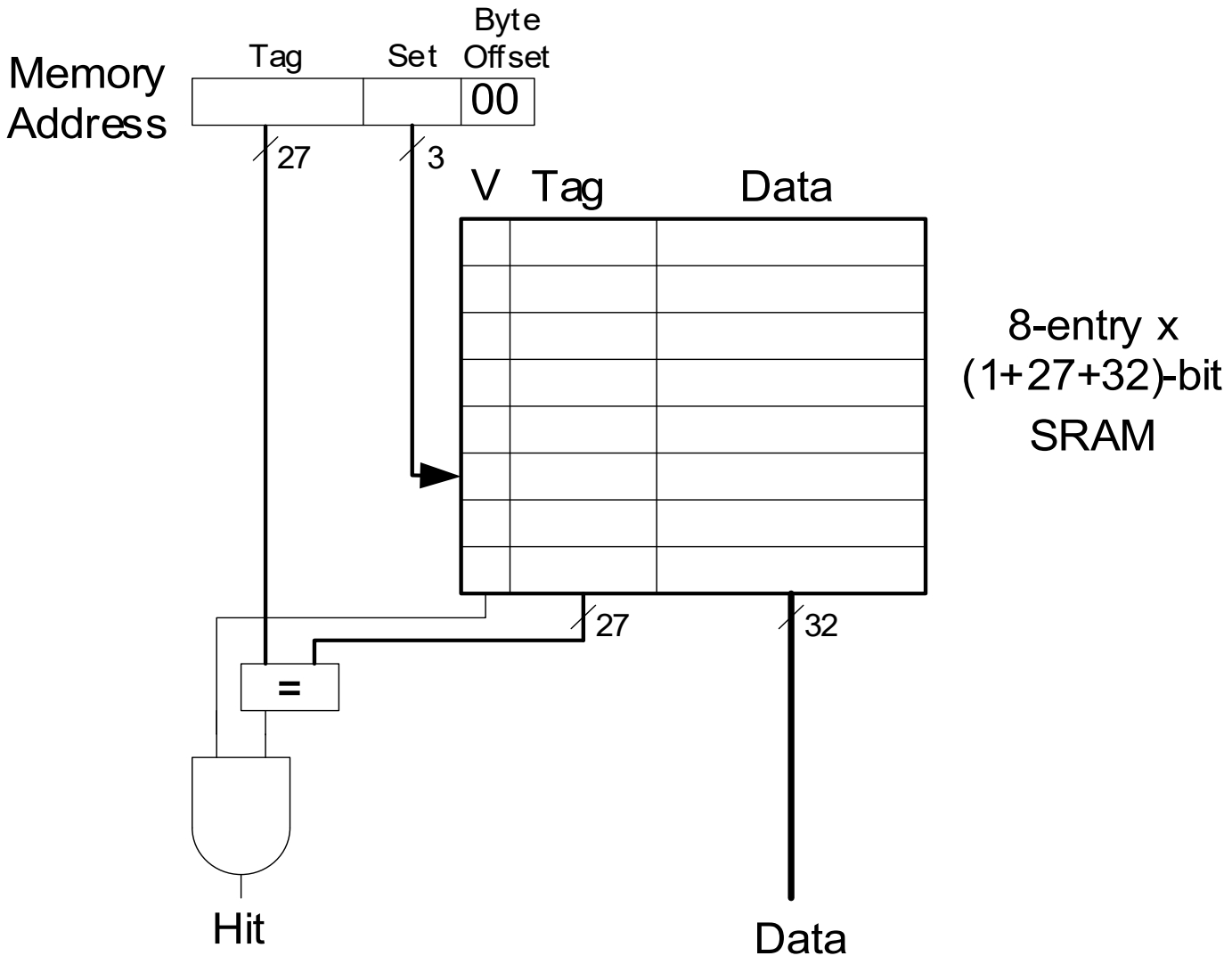
Ridiculously small, but will illustrate organizations

C4-3 Direct-Mapped Caches

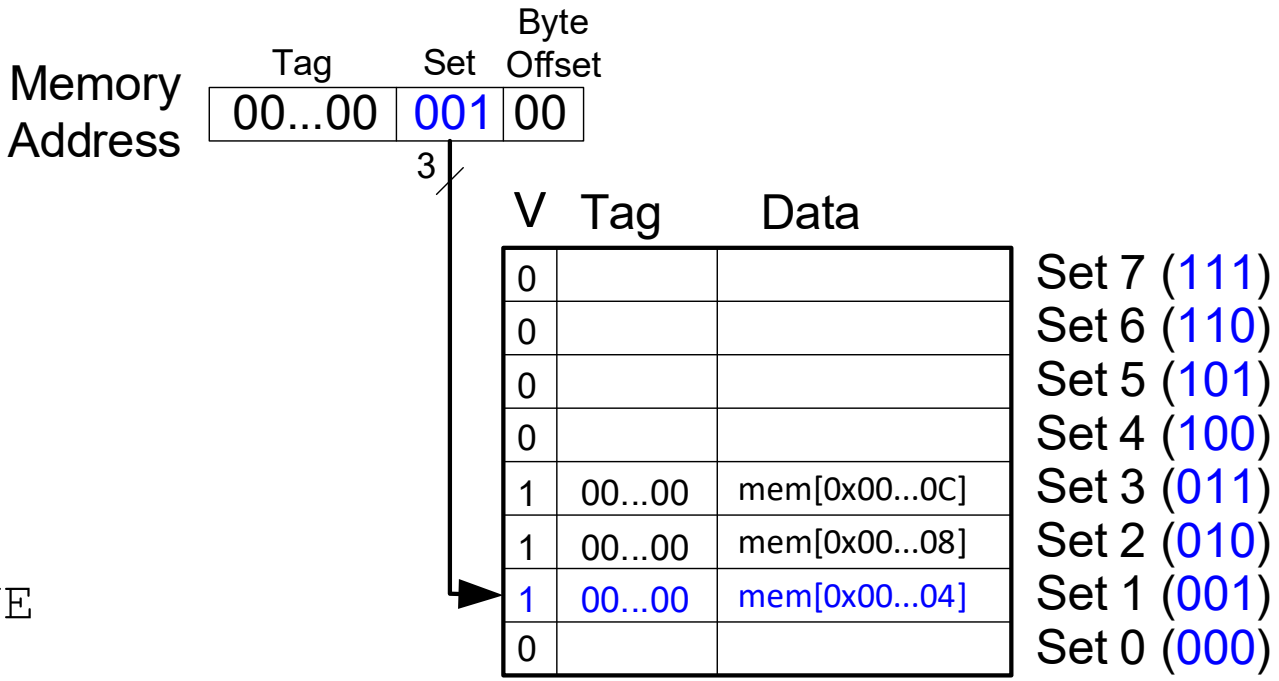
Direct-Mapped Cache



Direct-Mapped Cache Hardware



Direct-Mapped Cache Performance - Compulsory Misses

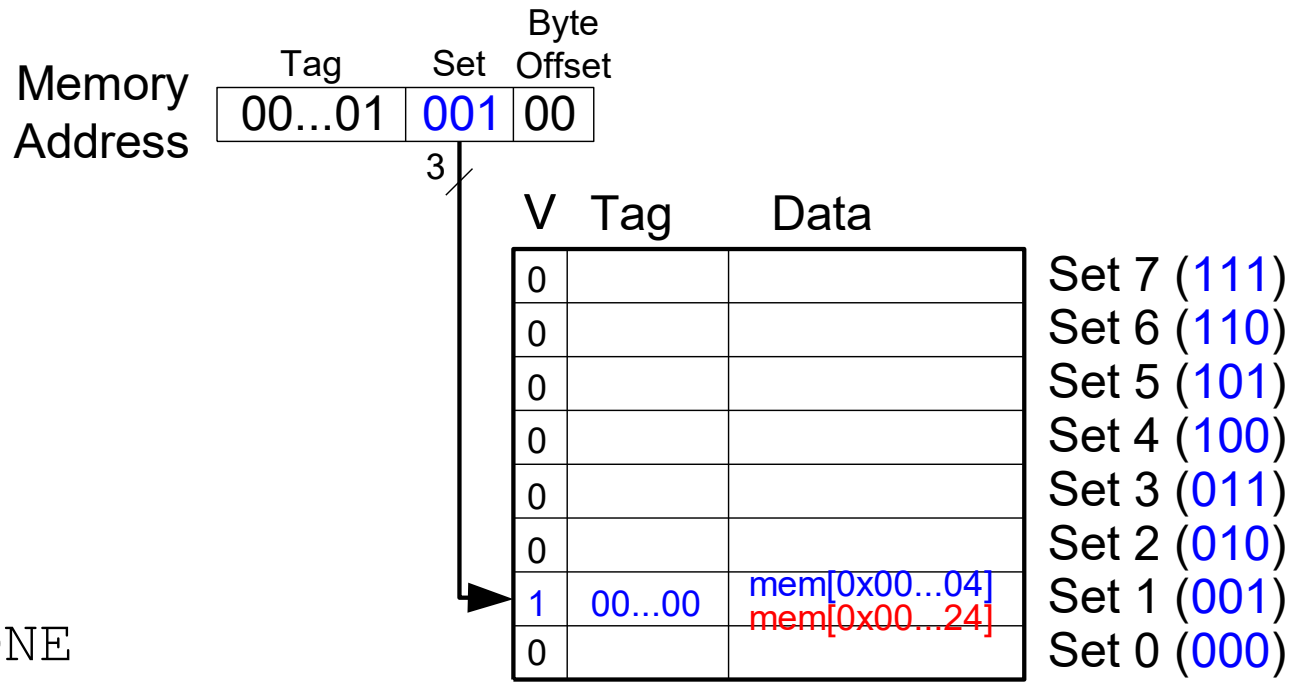



```
# RISC-V assembly code
        addi s0, zero, 5
        addi s1, zero, 0
LOOP:   beq  s0, zero, DONE
        lw   s2, 4(s1)
        lw   s3, 12(s1)
        lw   s4, 8(s1)
        addi s0, s0, -1
        j    LOOP
DONE:
```

$$\textit{Miss Rate} = \frac{3}{15} = 20\%$$

Temporal Locality
Compulsory Misses

Direct-Mapped Cache Performance - Conflict Miss



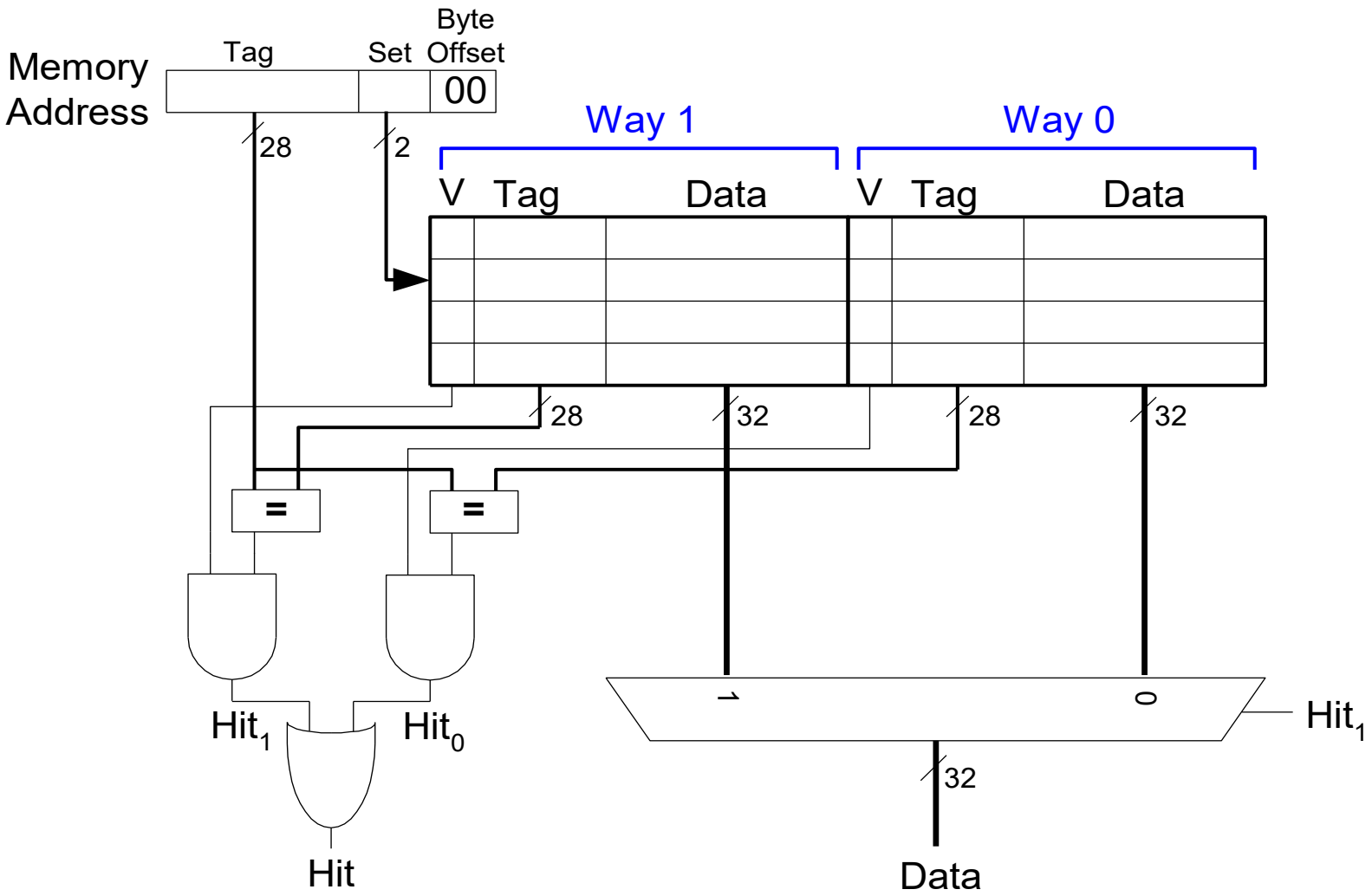

```
# RISC-V assembly code
        addi s0, zero, 5
        addi s1, zero, 0
LOOP:   beq  s0, zero, DONE
        lw   s2, 0x4(s1)
        lw   s4, 0x24(s1)
        addi s0, s0, -1
        j    LOOP
DONE:
```

$$\textit{Miss Rate} = \frac{10}{10} = 100\%$$

Conflict Misses

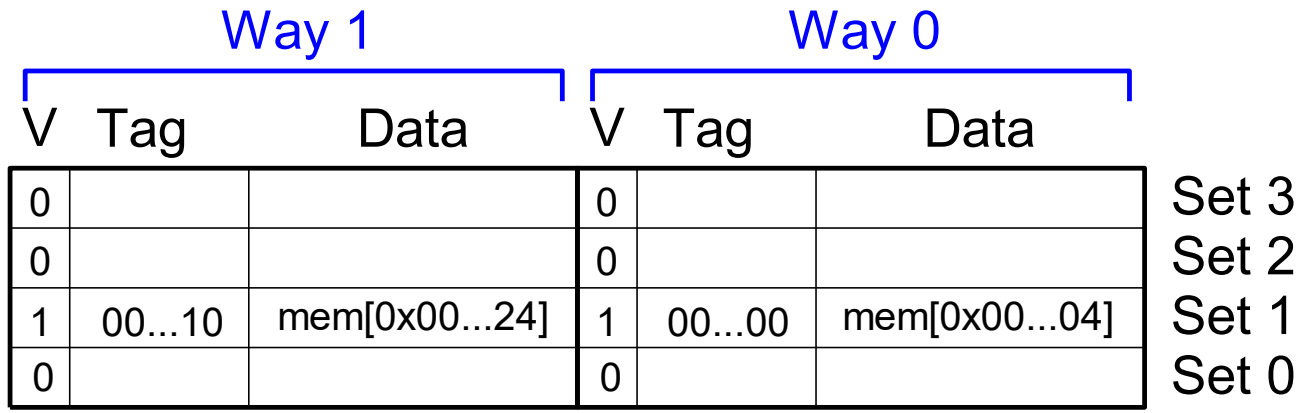
C4-4 Associative Caches

N-Way Set Associative Cache



N-Way Set Associative Cache Performance

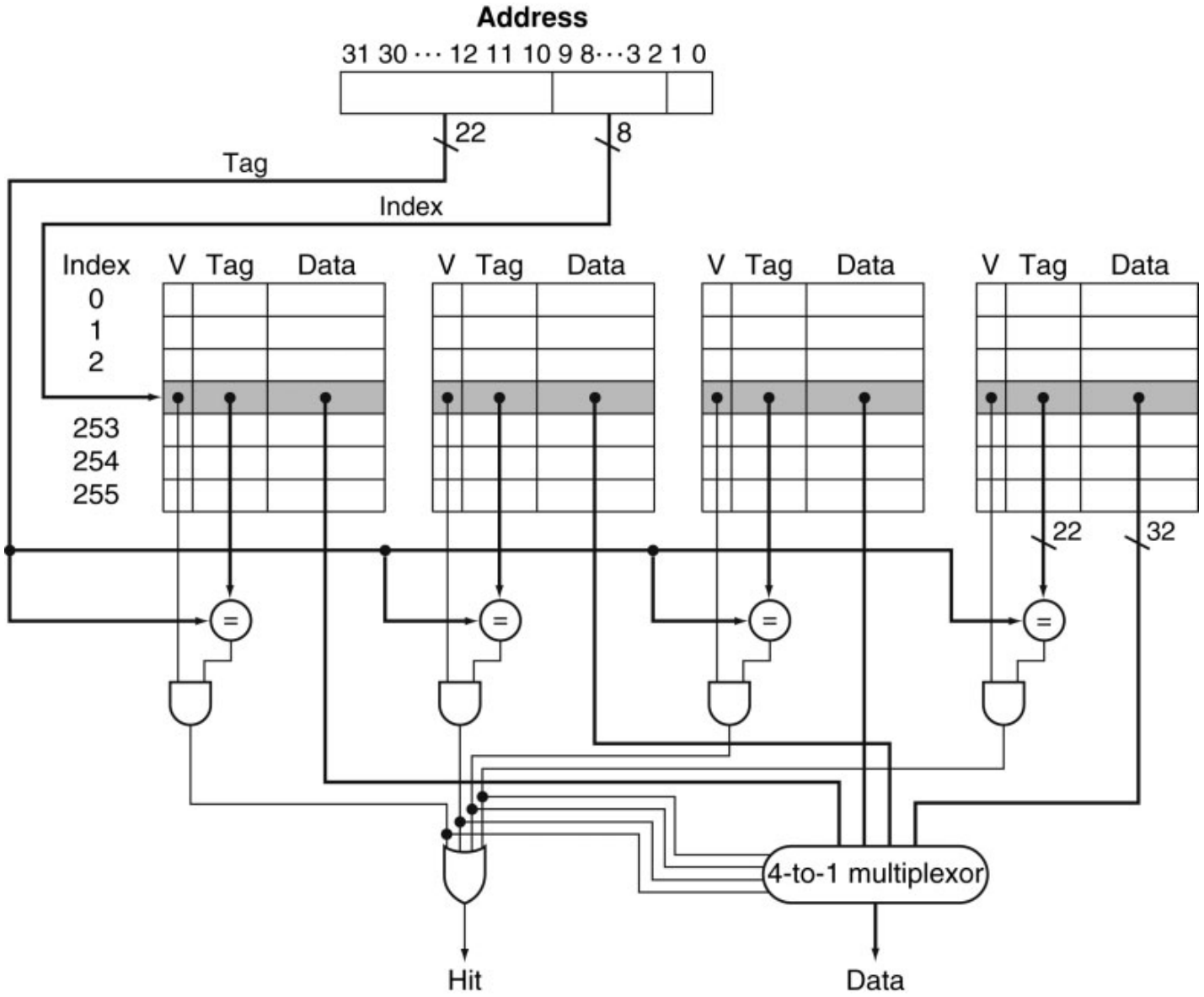
```
# RISC-V assembly code
        addi s0, zero, 5
        addi s1, zero, 0
LOOP:   beq  s0, zero, DONE
        lw   s2, 0x4(s1)
        lw   s4, 0x24(s1)
        addi s0, s0, -1
        j    LOOP
DONE:
```



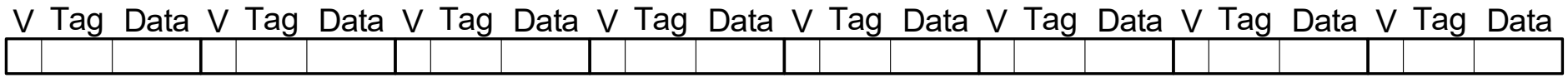
$$Miss\ Rate = \frac{2}{10} = 20\%$$

Associativity reduces Conflict Misses

Set Associative Cache: Aufbau



Fully Associative Cache

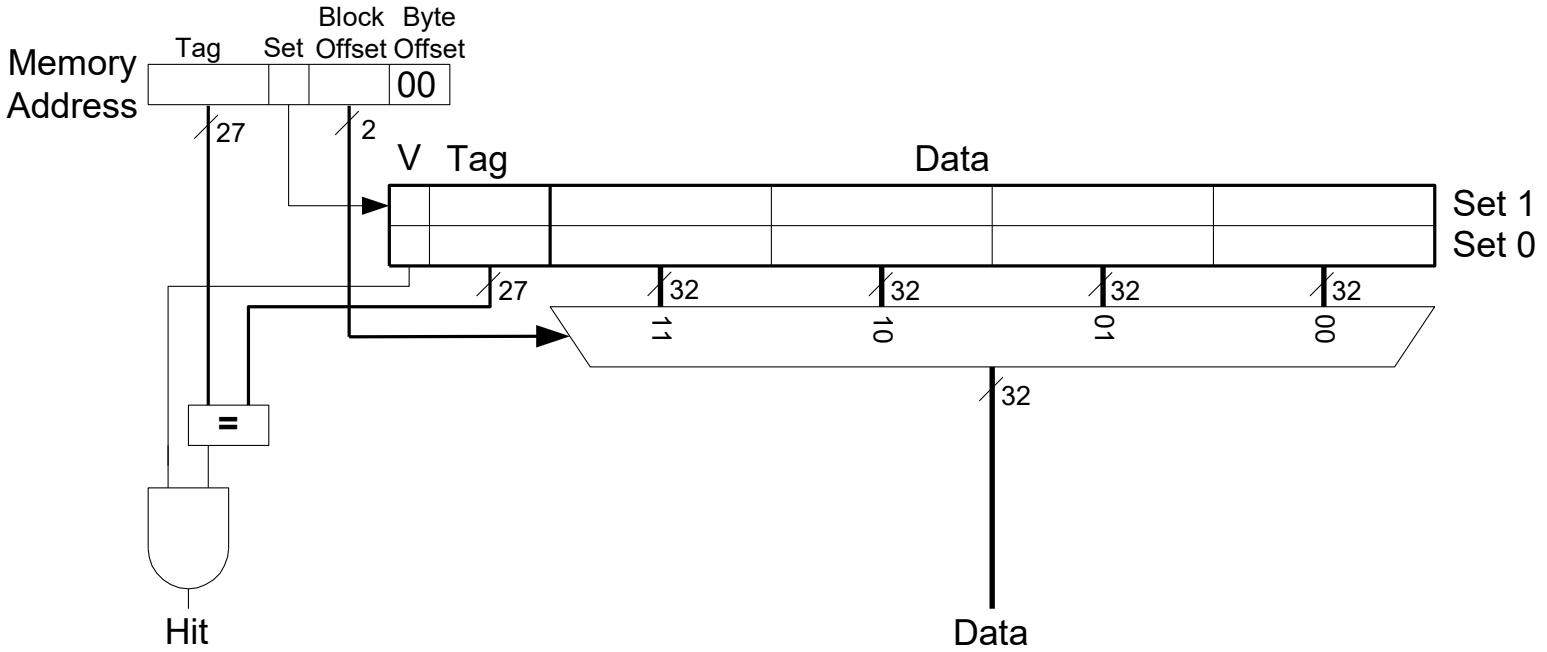


Reduces Conflict Misses but is expensive to build

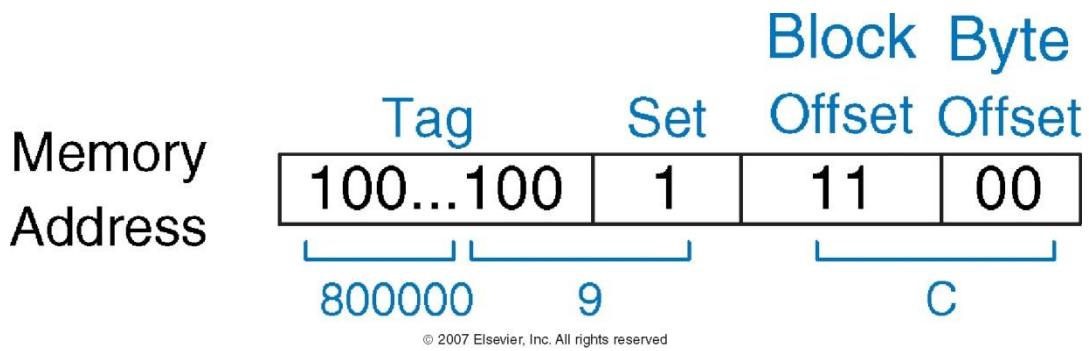
C4-5 Spatial Locality

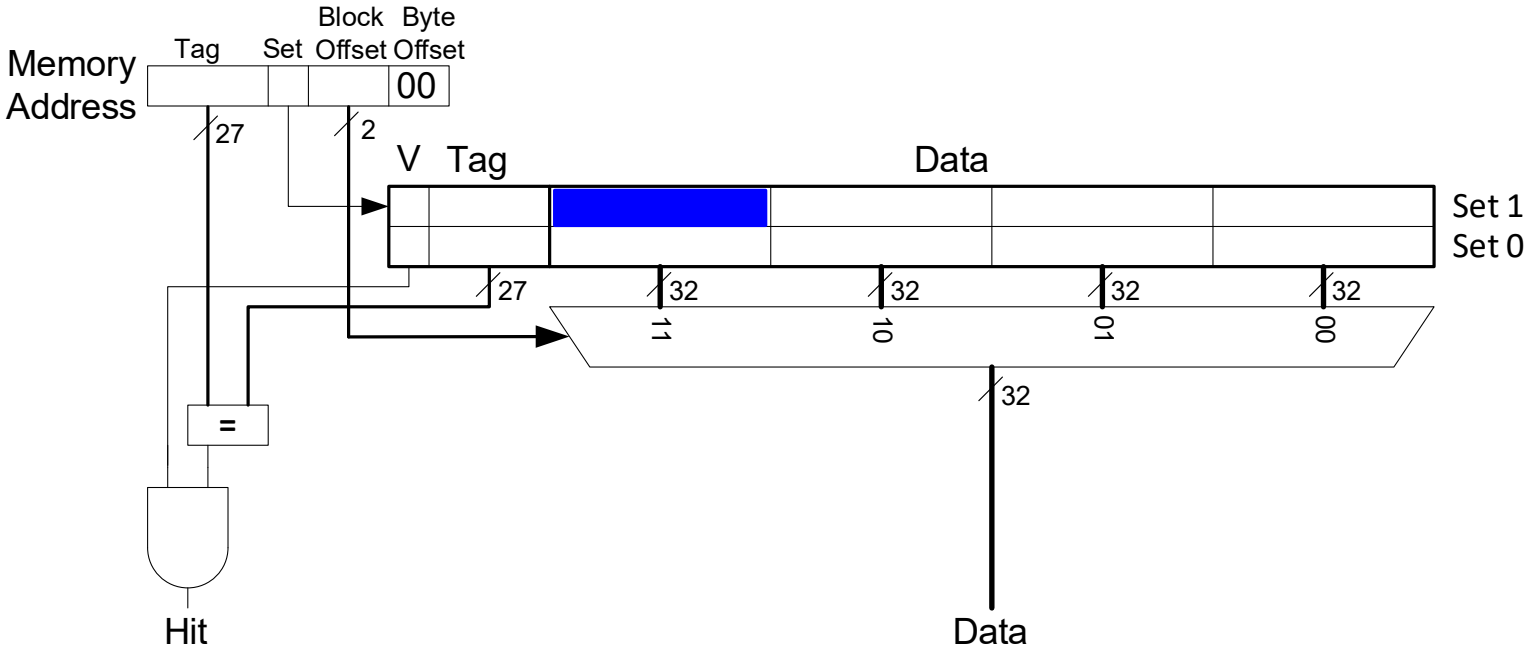
Spatial Locality

- Increase block size:
  - Block size, $b = 4$ words
  - $C = 8$ words
  - Direct mapped (1 block per set)
  - Number of blocks, $B = 2$ $\left(\frac{C}{b} = \frac{8}{4} = 2\right)$



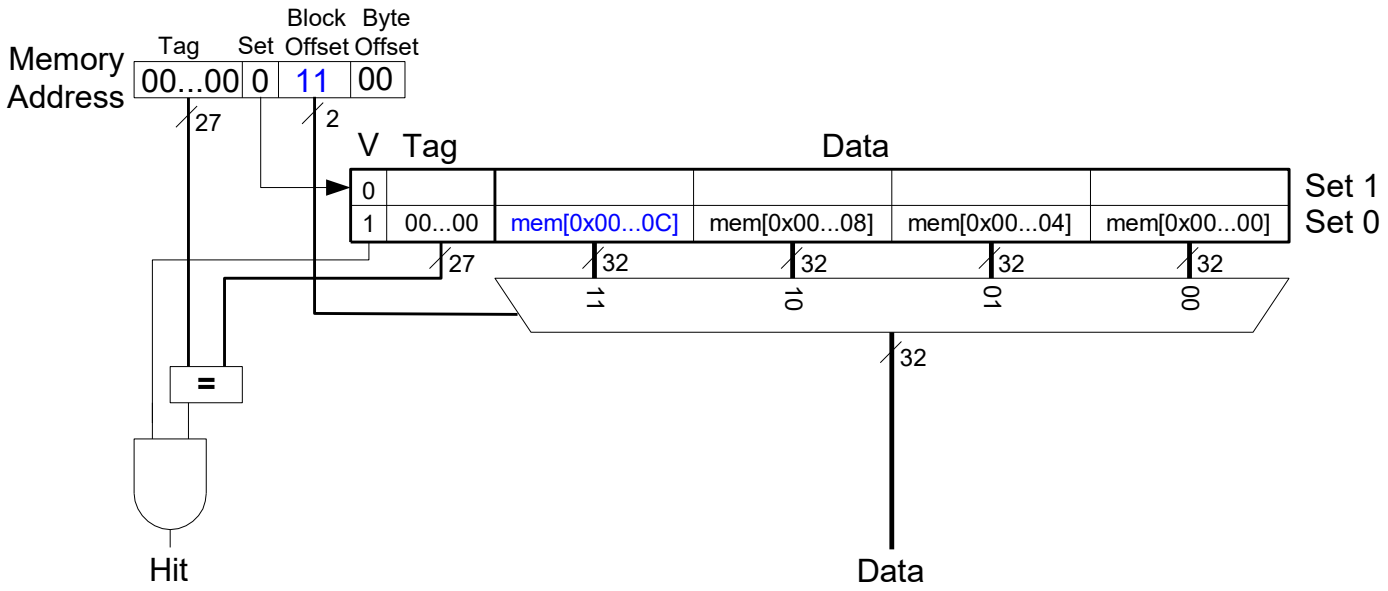
Cache with Larger Block Size





Cache Performance with Spatial Locality

```
# RISC-V assembly code
        addi s0, zero, 5
        addi s1, zero, 0
LOOP:   beq  s0, zero, DONE
        lw   s2, 4(s1)
        lw   s3, 12(s1)
        lw   s4, 8(s1)
        addi s0, s0, -1
        j    LOOP
DONE:
```



$$Miss\ Rate = \frac{1}{15} = 6.67\%$$

Larger blocks reduce compulsory misses through spatial locality

Cache Organization Recap

- Capacity: C
- Block size: b
- Number of blocks in cache: $B = \frac{C}{b}$
- Number of blocks in a set: N
- Number of sets: $S = \frac{B}{N}$

Organization	Number of Ways (N)	Number of Sets (S)
Direct Mapped	1	B
N-Way Set Associative	$1 < N < B$	$\frac{B}{N}$
Fully Associative	B	1

C4-6 Cache Replacement Policy

Types of Misses

- **Compulsory**: first time data accessed
- **Capacity**: cache too small to hold all data of interest
- **Conflict**: data of interest maps to same location in cache

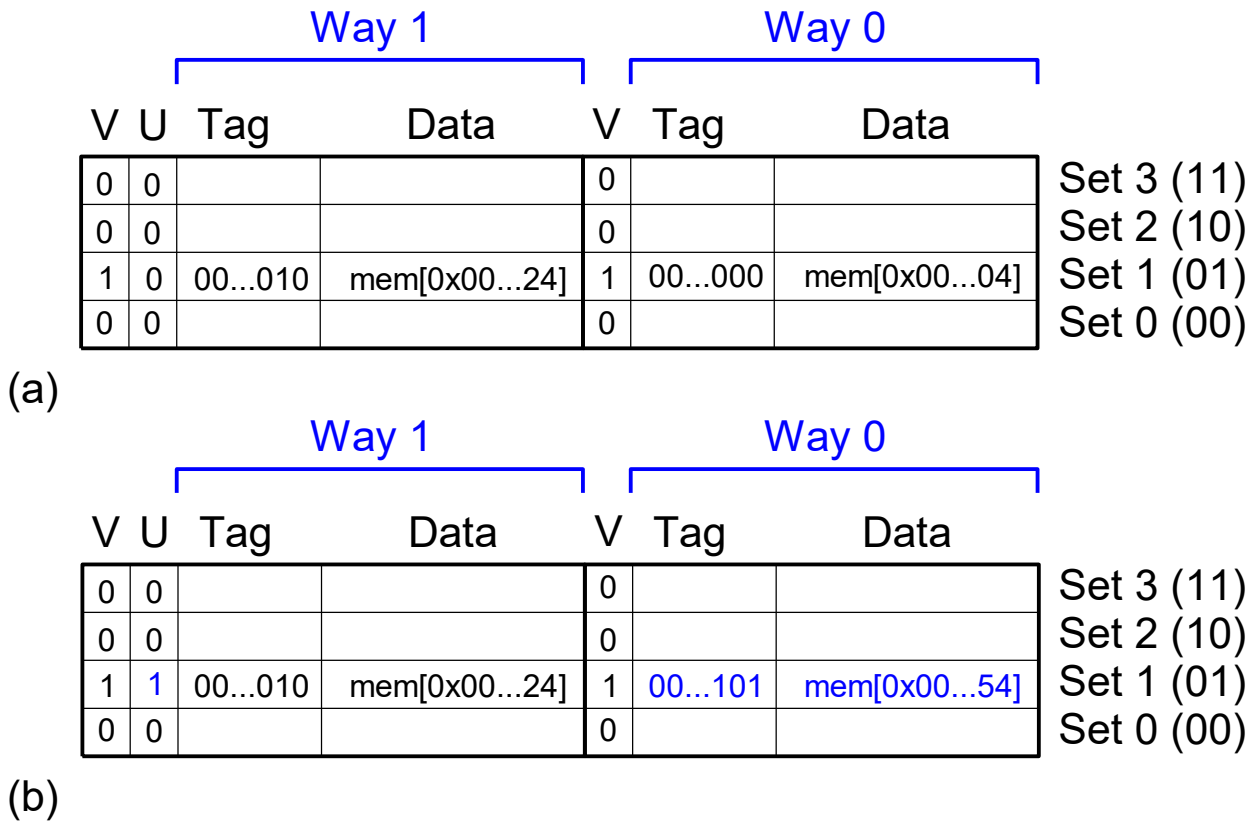
Miss penalty: time it takes to retrieve a block from lower level of hierarchy

Replacement Policy

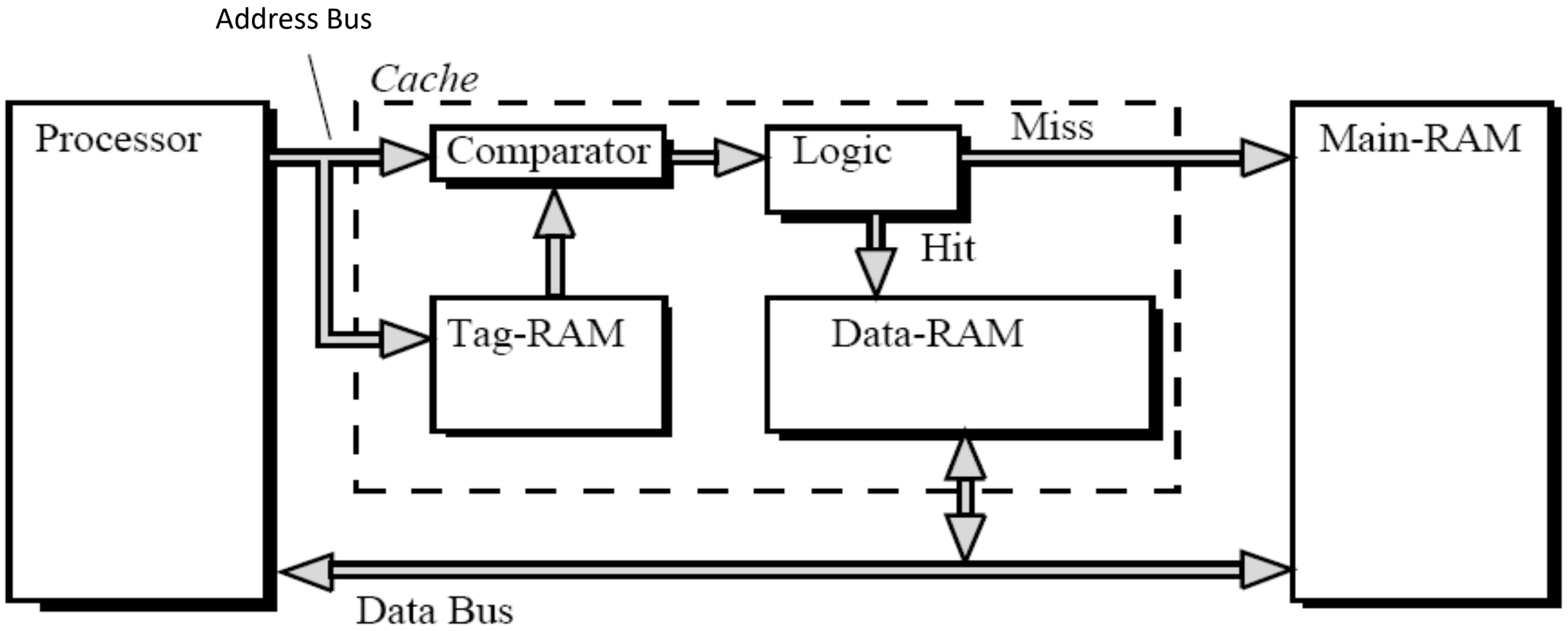
- Cache is too small to hold all data of interest at once
- If cache full: program accesses data X and evicts data Y
- **Capacity miss** when access Y again
- How to choose Y to minimize chance of needing it again?
  - **Least recently used (LRU) replacement**: the least recently used block in a set evicted

LRU Replacement

```
# RISC-V assembly
lw s1, 0x04(zero)
lw s2, 0x24(zero)
lw s3, 0x54(zero)
```



Cache


Address Bus
Cache
Processor
Comparator
Logic
Miss
Main-RAM
Hit
Tag-RAM
Data-RAM
Data Bus

Multilevel Caches

- Larger caches have lower miss rates, longer access times
- Expand memory hierarchy to multiple levels of caches
- Level 1: small and fast (e.g. 16 KB, 1 cycle)
- Level 2: larger and slower (e.g. 256 KB, 2-6 cycles)
- Most modern PCs have L1, L2, and L3 cache

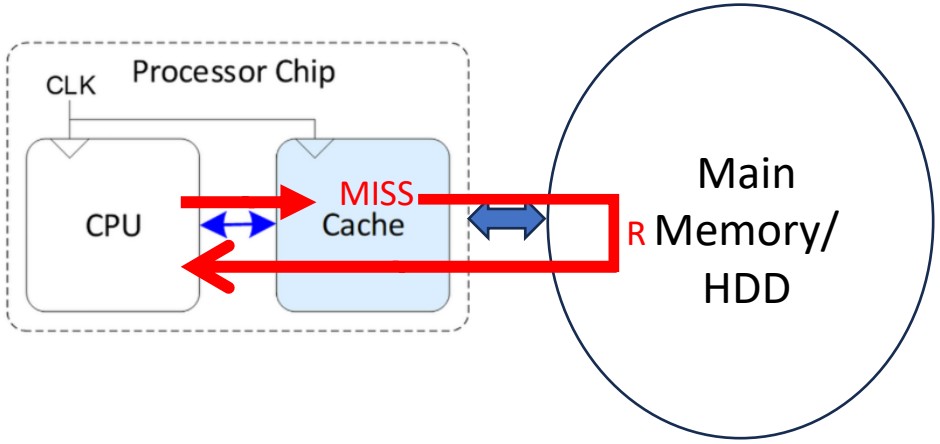
C4-7 Cache Miss/Hit Strategy

Hit & Miss on reading / load

Load

- Load Hit: Valid bit is set and tag maches. Data is found in cache

- Load Miss: Data is not found in cache
  - Pipeline needs to be stalled
  - Slower memory must be asked to deliver the data



Hit & Miss on writing / store

Store

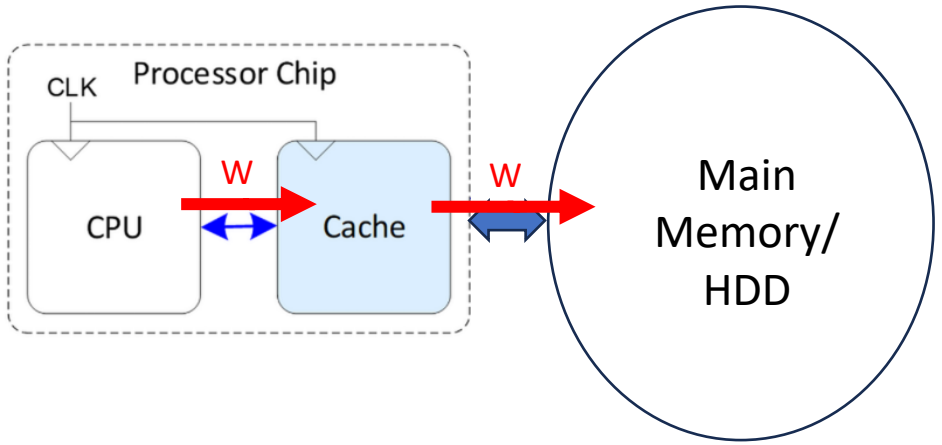
- Write Hit
  - Write-Through
  - Copy-Back
  - Write-Buffer

- Write Miss
  - Write-Around
  - Fetch-on-Write
    - Afterwards: Write Hit

Write-Hit: Write-Through

- update the cache **AND**
- update the main memory immediately

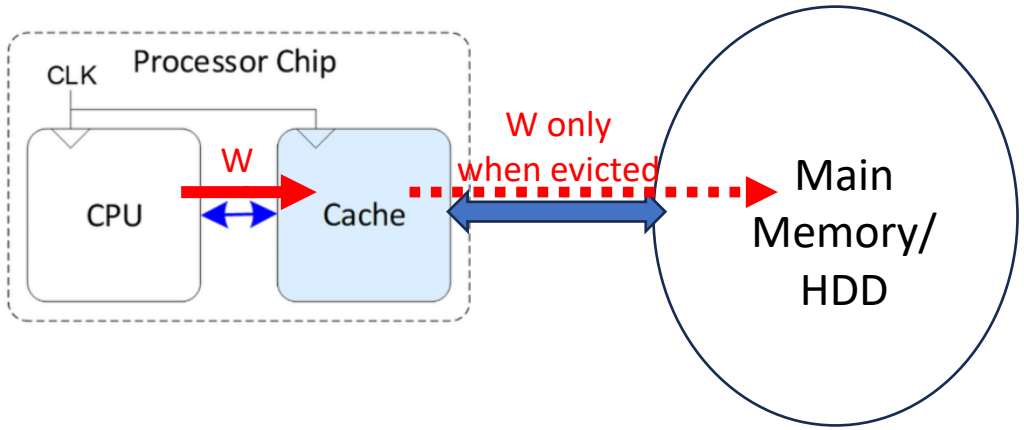
- Pros
  - Data consistency with main memory guaranteed (I/O, multiprocessor)
  - simple
- Cons
  - Frequent accesses to the main memory
  - Loss of performance



Write-Hit: Copy-Back

- refresh the cache **AND** marks the block "**dirty**"
- only update the main memory later when the block is removed from the cache
  - often also referred to as *write-back*

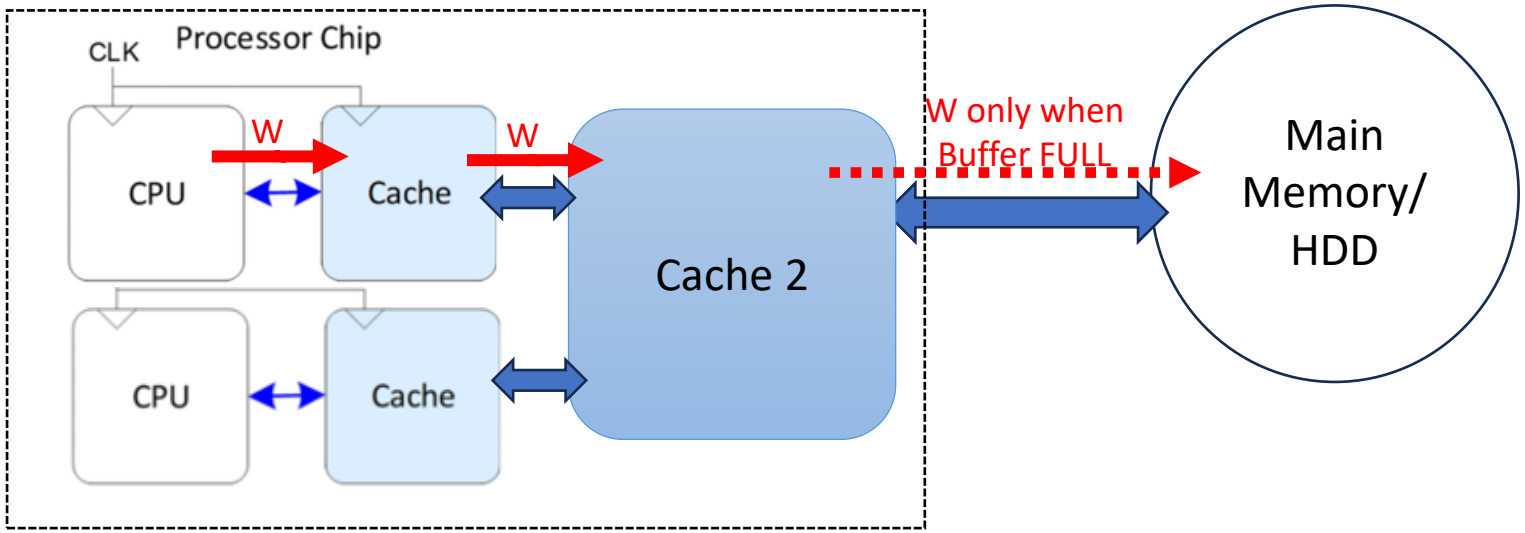
- Pros
  - Write hit is much faster
  - Less frequent accesses to the main memory
- Cons
  - Data inconsistency with the main memory
  - Read miss is slower (due to copy-back)
    - A dirty block needs to be synced before replacing



Write-Hit: Write-Buffer

- for data consistency and fast write operations
  - advantages of Write-Through and Copy-Back

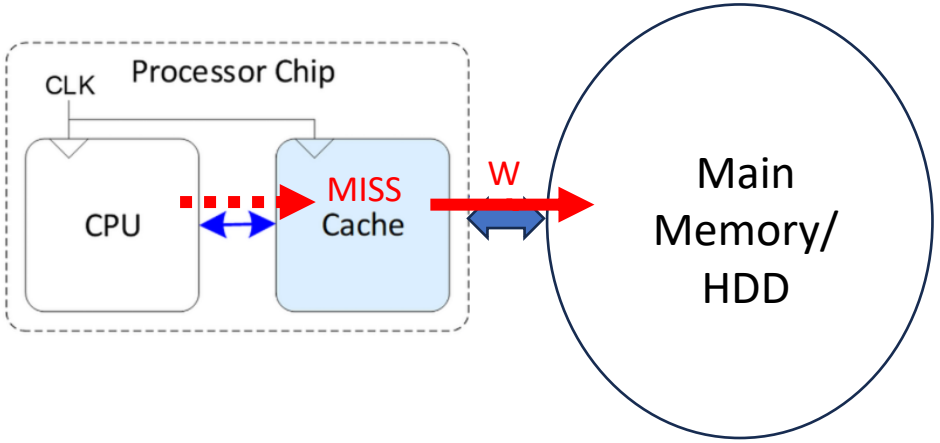
- Write-Buffer (Buffered Write-Through)
  - new value is entered in the cache and second fast cache
  - Processor can continue with further processing
  - if buffer is full, processor must wait



Write-Miss: Write-Around

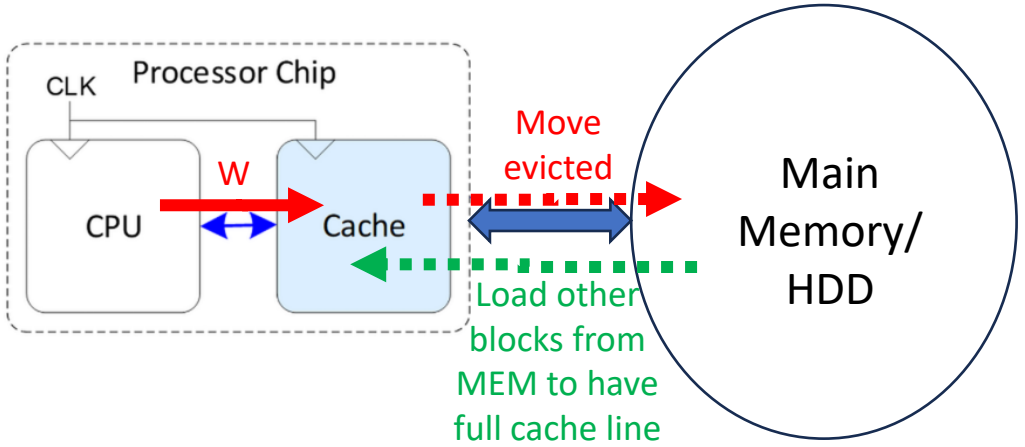
- Ignore the cache AND write directly to memory

- Mostly in combination with Write-Through



Write-Miss: Fetch-on-Write

- Replace the current content of the cache and update Tag
- If block size > 1 word, load the remaining data belonging to the block from the main memory after
  - Read access to the memory and
  - then write hit depending on the strategy
- This is the most frequently used method



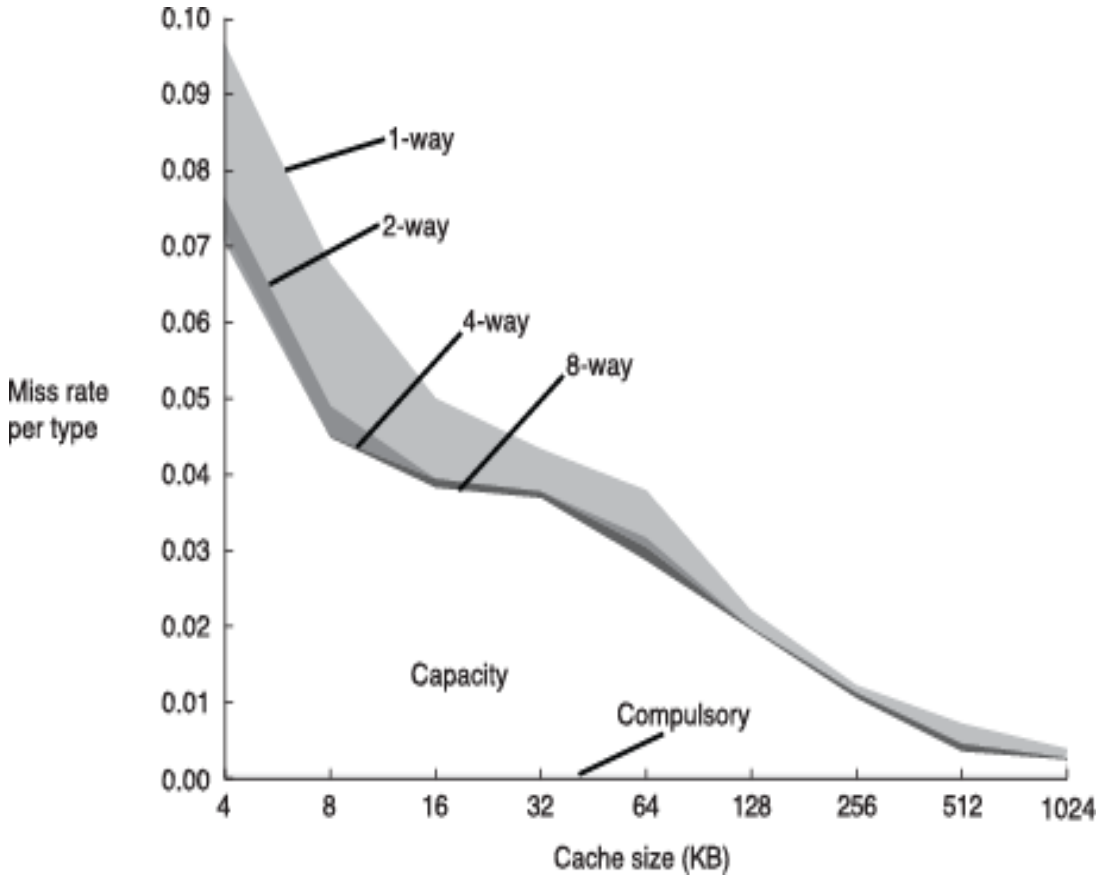
Cache Summary

Cache Summary

- **What data is held in the cache?**
  - Recently used data (temporal locality)
  - Nearby data (spatial locality)
- **How is data found?**
  - Set is determined by address of data
  - Word within block also determined by address
  - In associative caches, data could be in one of several ways
- **What data is replaced?**
  - Least-recently used way in the set

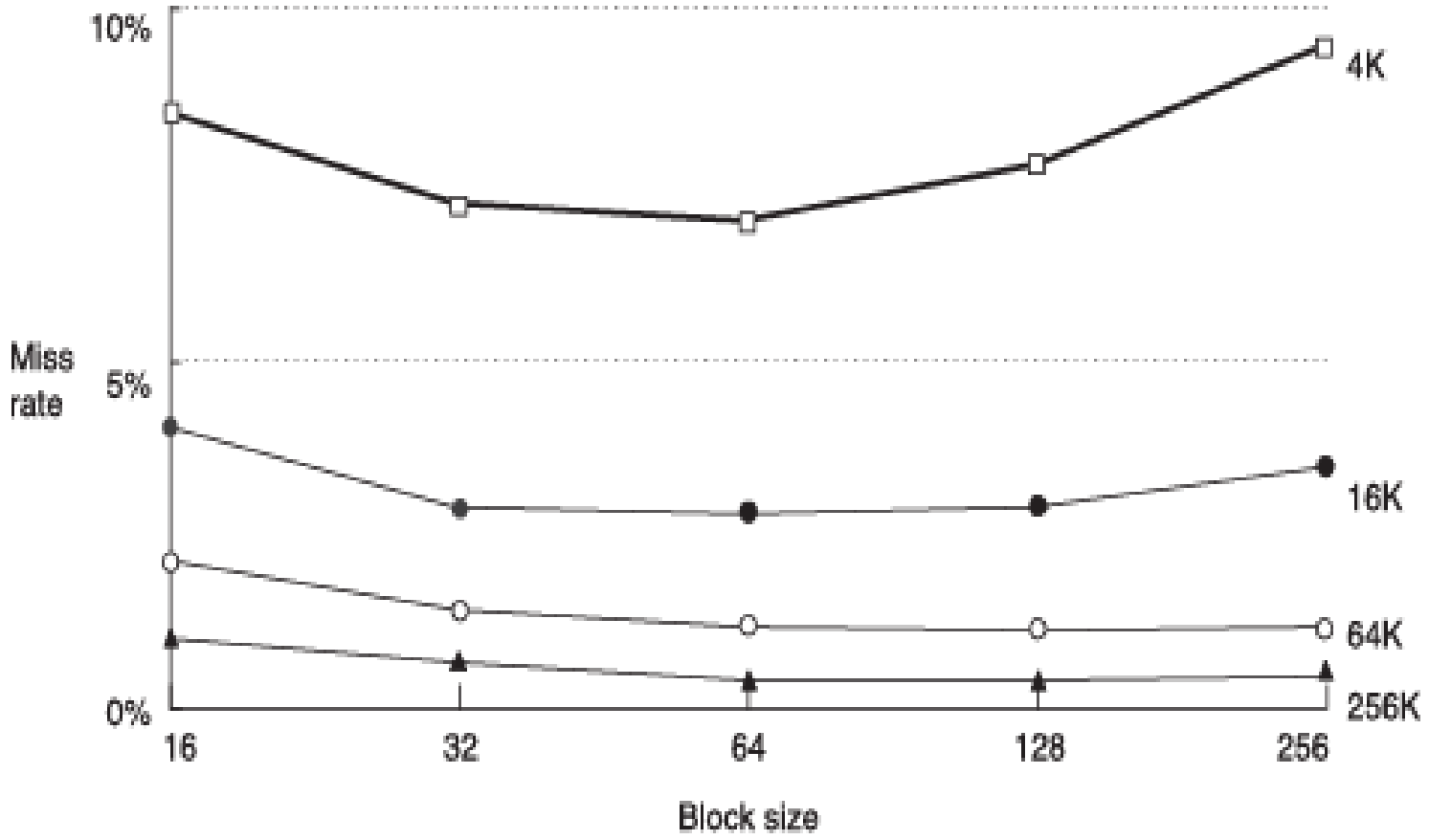
Miss Rate Trends

- **Bigger caches reduce capacity misses**
- **Greater associativity reduces conflict misses**



Adapted from Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*, 2011

Miss Rate Trends



- **Bigger blocks reduce compulsory misses**
- **Bigger blocks increase conflict misses**

Intel Pentium III Die