

Lecture 9

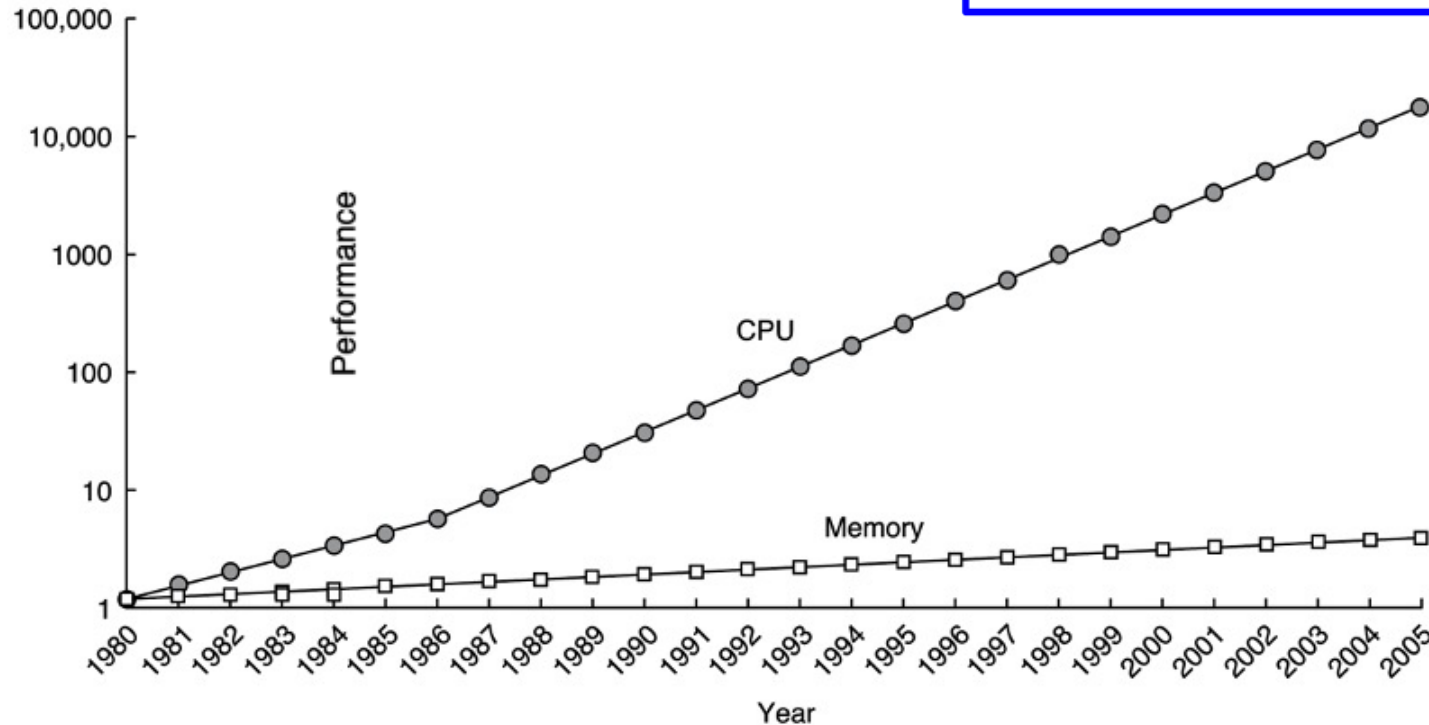
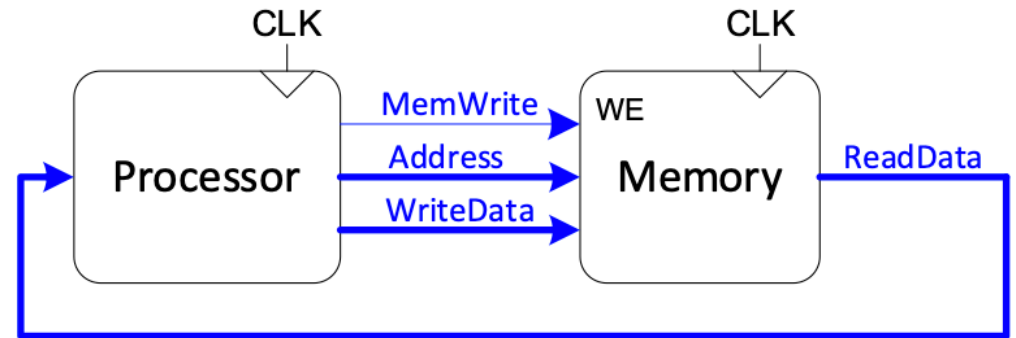
Memory Hierarchy - Cache Memory

Peter Cheung
Imperial College London

URL: www.ee.imperial.ac.uk/pcheung/teaching/EE2_CAS/
E-mail: p.cheung@imperial.ac.uk

Processor to Memory Interface

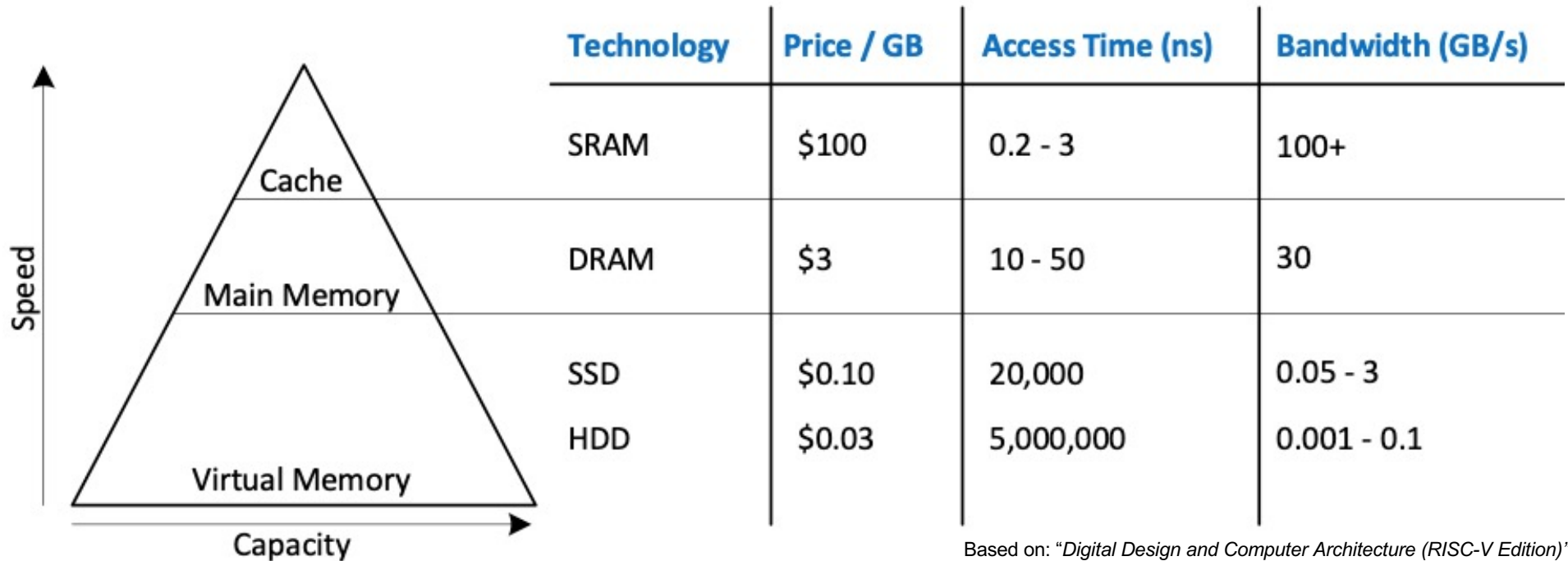
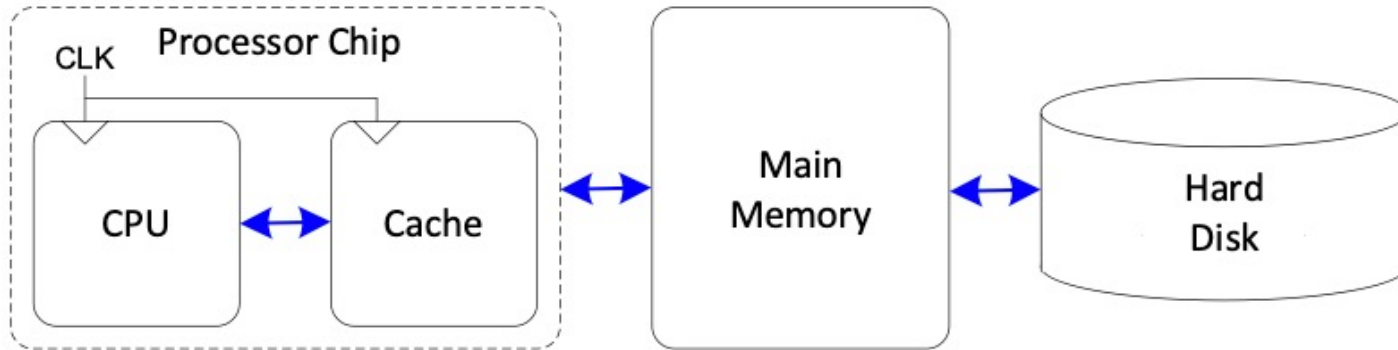
- **Computer performance depends on:**
 - Processor performance
 - Memory system performance



- **Ideal memory**
 - Fast
 - Cheap
 - Large

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Memory Hierarchy



Based on: "Digital Design and Computer Architecture (RISC-V Edition)" by Sarah Harris and David Harris (H&H),

Big idea: Principle of 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

Based on: “*Digital Design and Computer Architecture (RISC-V Edition)*”
by Sarah Harris and David Harris (H&H),

Memory Performance

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

Hit Rate = # hits / # memory accesses
= 1 – Miss Rate

Miss Rate = # misses / # memory accesses
= 1 – Hit Rate

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

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

Based on: “*Digital Design and Computer Architecture (RISC-V Edition)*”
by Sarah Harris and David Harris (H&H),

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 = $1250/2000 = 0.625$

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

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Memory Performance Example 2

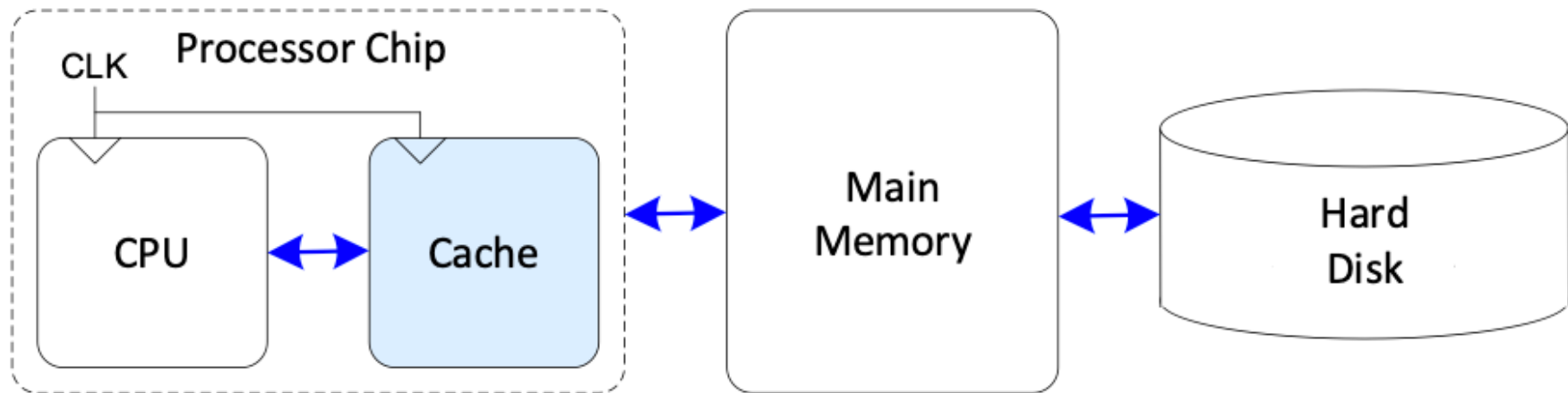
- Suppose processor has 2 levels of hierarchy: cache and main memory
- $t_{\text{cache}} = 1$ cycle, $t_{MM} = 100$ cycles
- What is the **AMAT** (average memory access time) of the program from Example 1?

$$\begin{aligned}\text{AMAT} &= t_{\text{cache}} + MR_{\text{cache}}(t_{MM}) \\ &= [1 + 0.375(100)] \text{ cycles} \\ &= \mathbf{38.5 \text{ cycles}}\end{aligned}$$

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Cache Memory

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



Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Cache Design Questions

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

- 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

Based on: “*Digital Design and Computer Architecture (RISC-V Edition)*”
by Sarah Harris and David Harris (H&H),

Cache Terminology

- **Capacity (C):**
 - number of data bytes in cache
- **Block size (b):**
 - bytes of data brought into cache at once
- **Number of blocks ($B = C/b$):**
 - number of blocks in cache: $B = C/b$
- **Degree of associativity (N):**
 - number of blocks in a set
- **Number of sets ($S = B/N$):**
 - each memory address maps to exactly one cache set

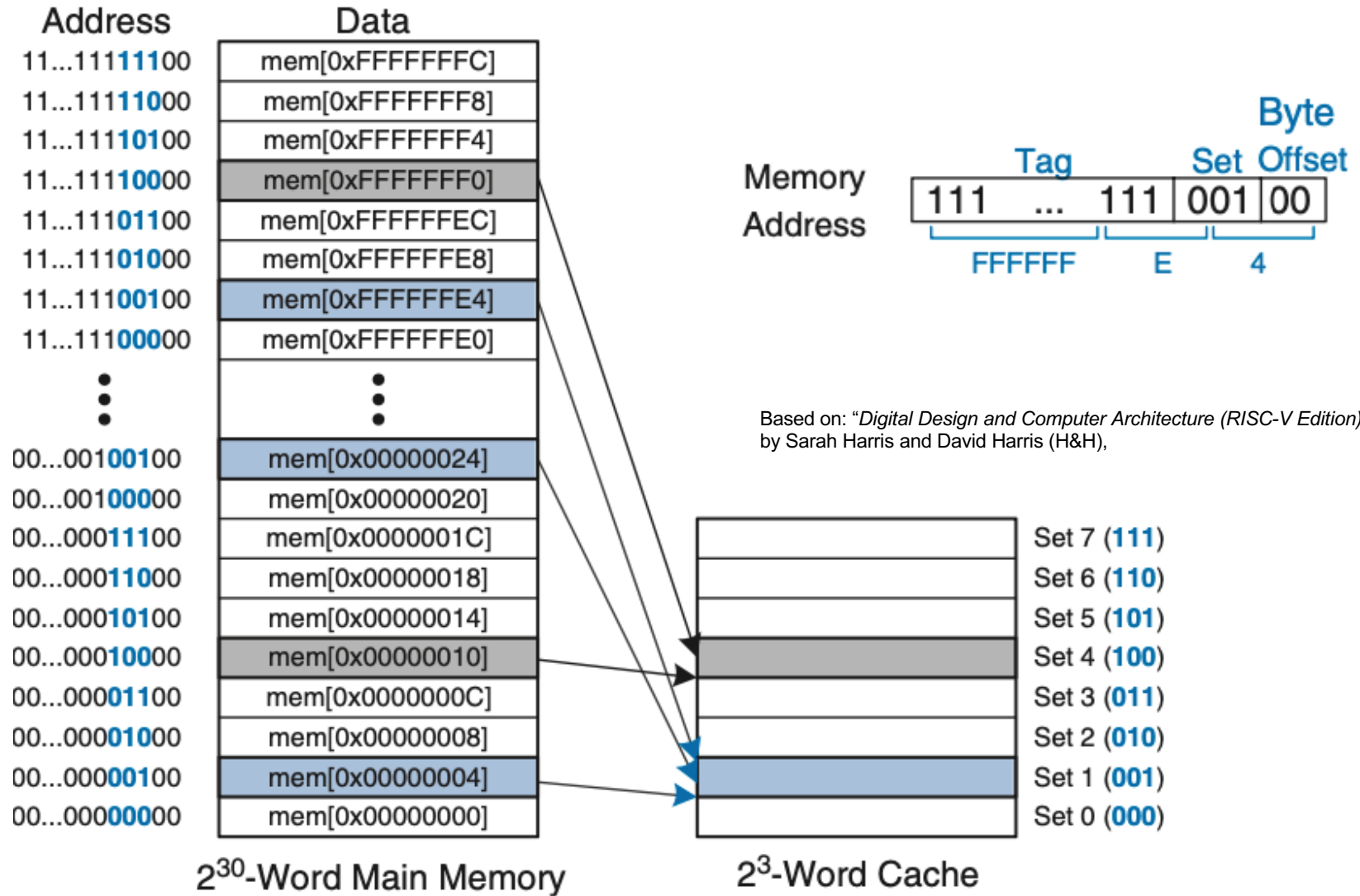
Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

How to find data in cache?

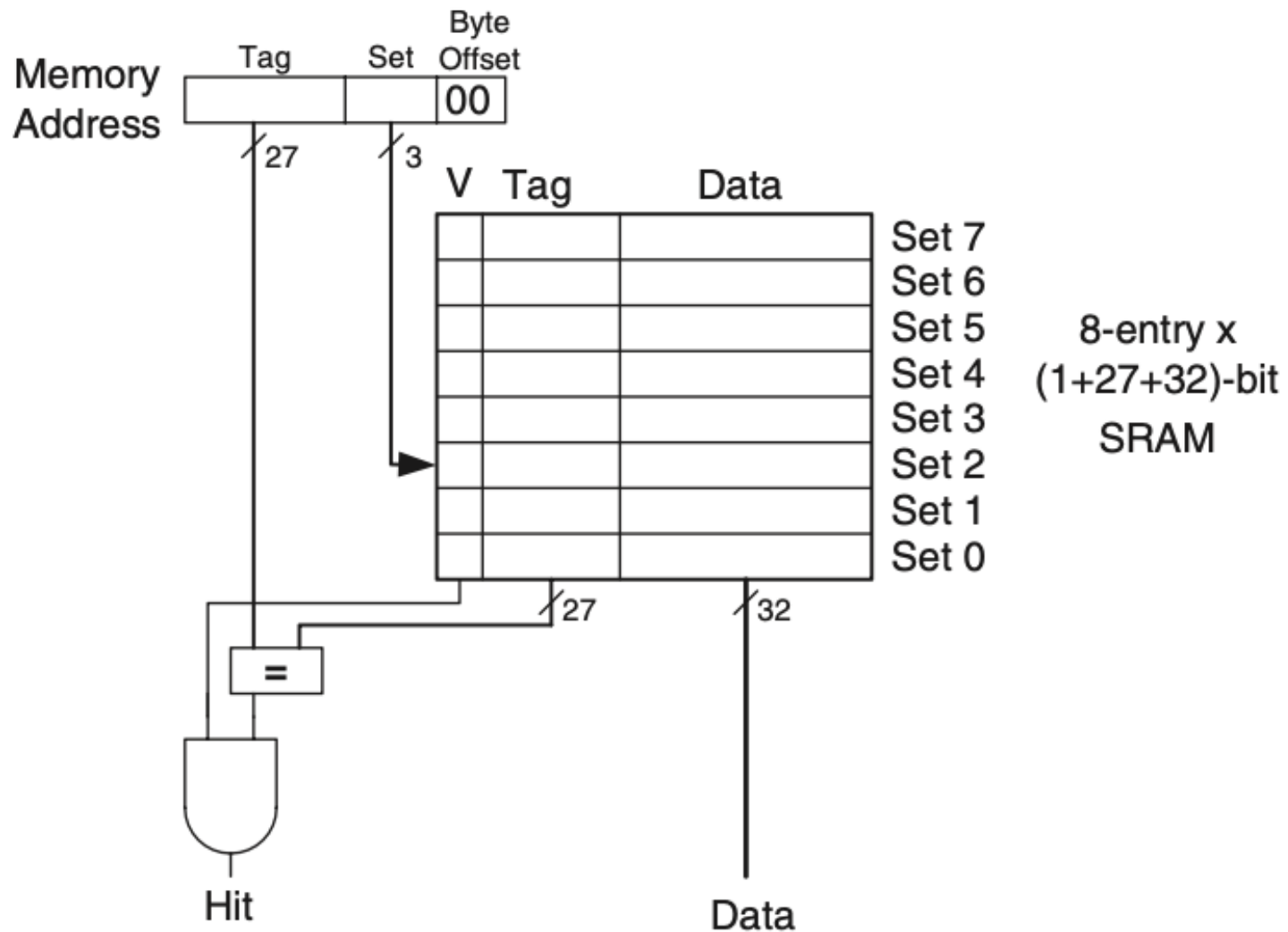
- Cache organized into S sets
- Each memory address maps to exactly one set
- Caches categorized by # of blocks in a set:
 - **Direct mapped:** 1 block per set
 - **N -way set associative:** N blocks per set
 - **Fully associative:** all cache blocks in 1 set
- Examine each organization for a cache with: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

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Direct Mapped Cache



Direct Mapped Cache hardware



Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Direct Mapped Cache Miss Rate Example

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:

```

Memory Address

Tag	Set	Byte Offset
00...00	001	00

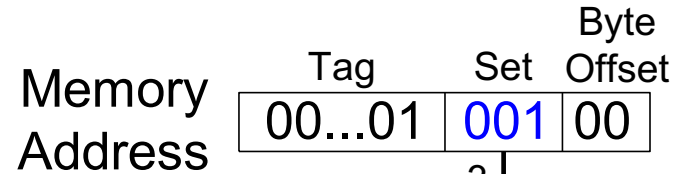
3

V	Tag	Data	
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
1	00...00	mem[0x00...0C]	Set 3 (011)
1	00...00	mem[0x00...08]	Set 2 (010)
1	00...00	mem[0x00...04]	Set 1 (001)
0			Set 0 (000)

Miss Rate = 3/15
= 20%

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
 by Sarah Harris and David Harris (H&H),

Direct Mapped Cache Conflict Miss Example



3

RISC-V assembly code

```

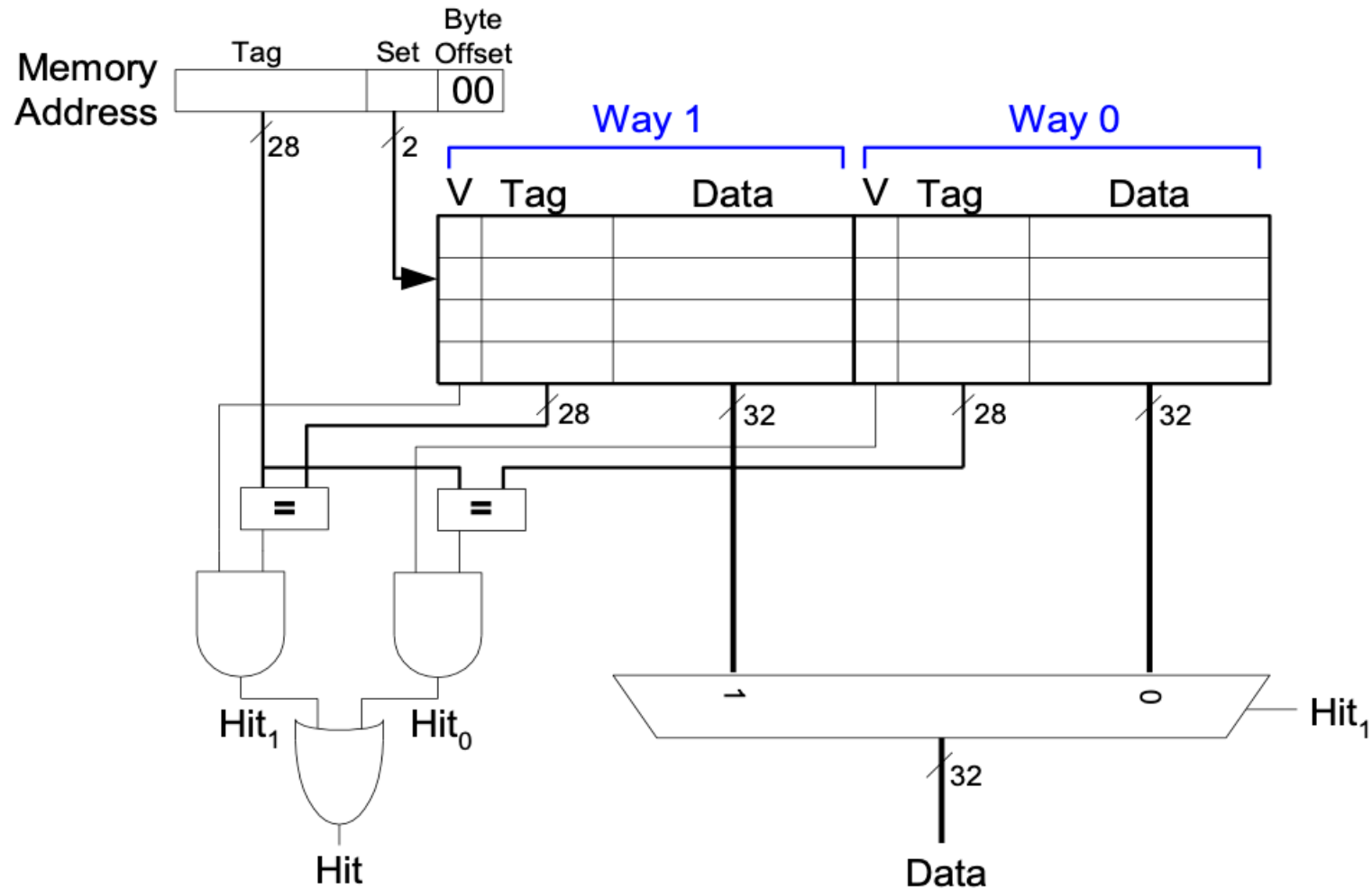
    addi s0, zero, 5
    addi s1, zero, 0
LOOP: beq  s0, zero, DONE
    lw   s2, 0x4(s1)
    lw   s4, 0x24(s1)
    addi s0, s0, -1
    j    LOOP
DONE:
  
```

V	Tag	Data	Set
0			Set 7 (111)
0			Set 6 (110)
0			Set 5 (101)
0			Set 4 (100)
0			Set 3 (011)
0			Set 2 (010)
1	00...00	mem[0x00...04] mem[0x00...24]	Set 1 (001)
0			Set 0 (000)

Miss Rate = 10/10
= 100%

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

2-Way Set Associative Cache



Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

2-Way Set Associative Cache Miss Rate Example

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:

```

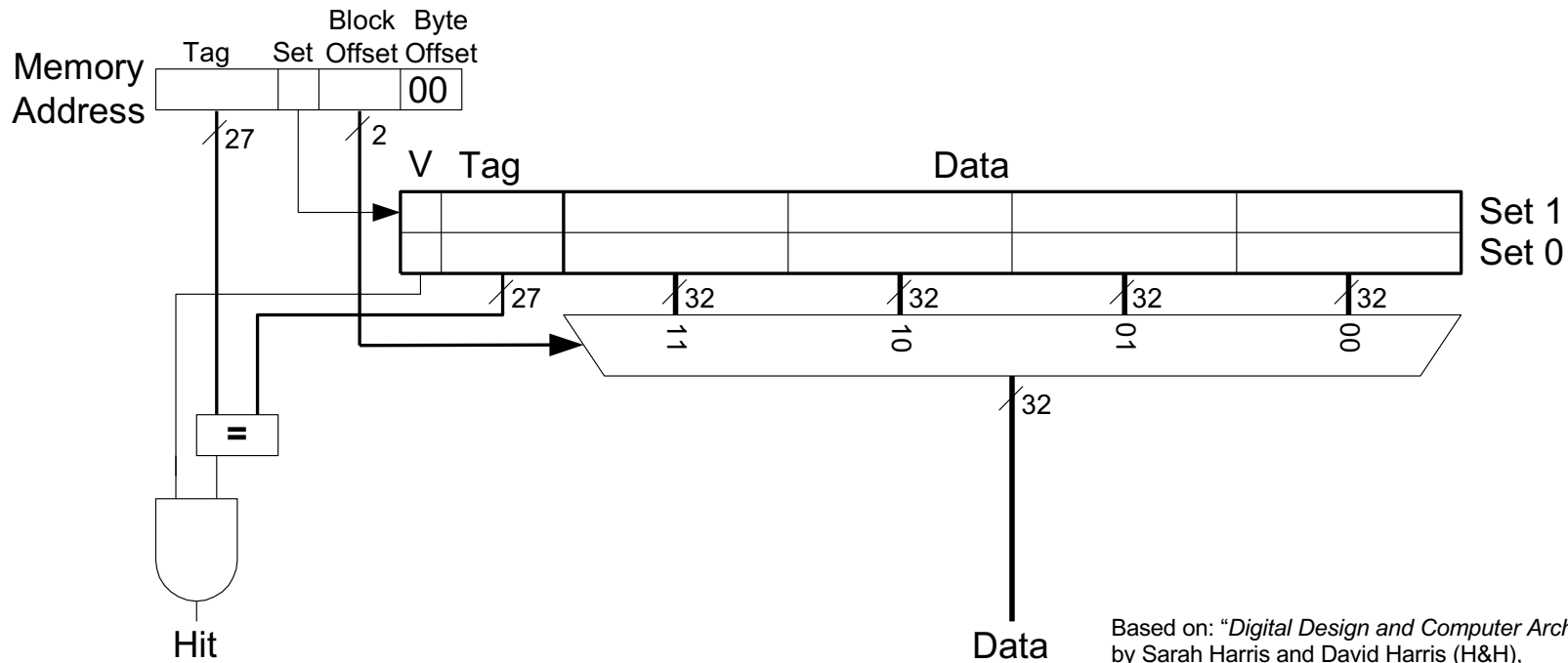
Miss Rate = 2/10
= 20%

Way 1			Way 0			
V	Tag	Data	V	Tag	Data	
0			0			Set 3
0			0			Set 2
1	00...10	mem[0x00...24]	1	00...00	mem[0x00...04]	Set 1
0			0			Set 0

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
 by Sarah Harris and David Harris (H&H),

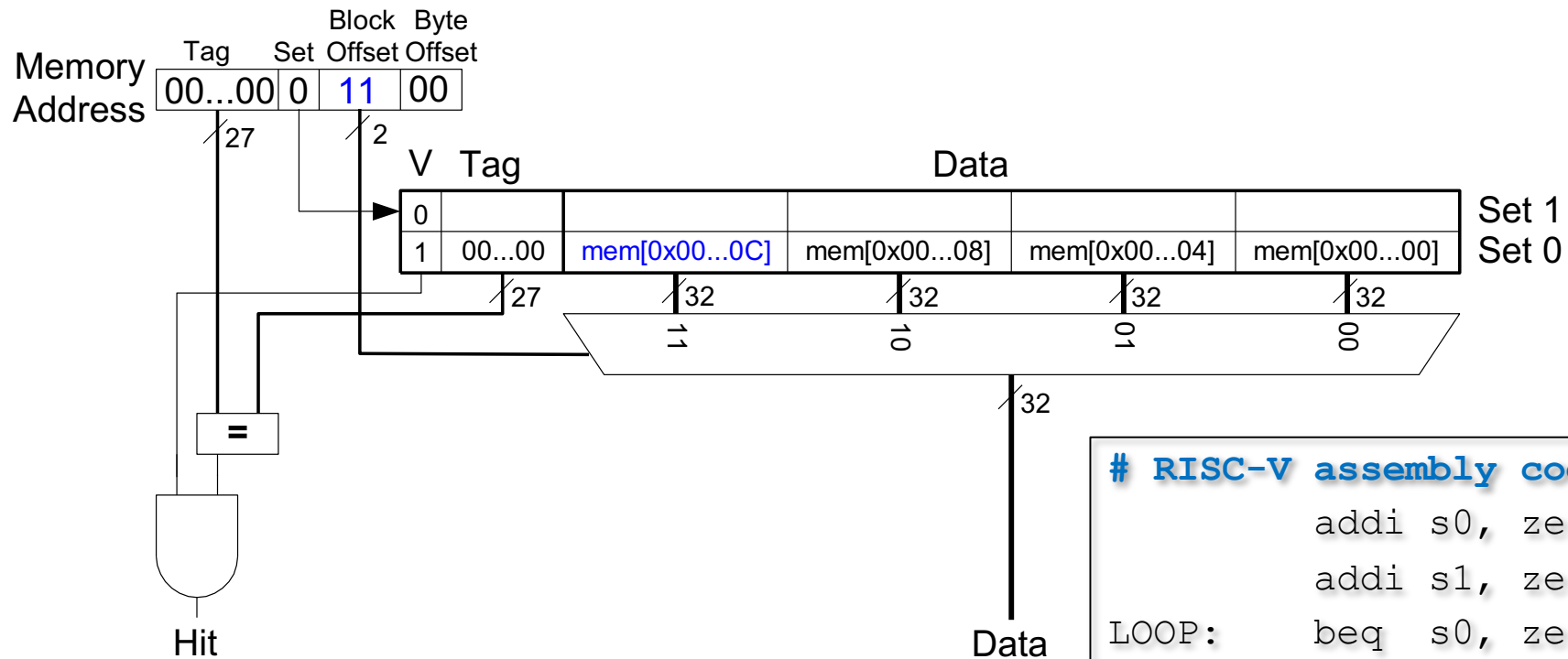
Design for Spatial Locality

- Increase block size:
 - Block size, $b = 4$ words
 - $C = 8$ words
 - Direct mapped (1 block per set)
 - Number of blocks, $B = 2$ ($C/b = 8/4 = 2$)



Based on: "Digital Design and Computer Architecture (RISC-V Edition)" by Sarah Harris and David Harris (H&H),

Cache Performance with $b = 4$



Miss Rate = 1/15
= 6.67%

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:
    
```

Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
 by Sarah Harris and David Harris (H&H),

Cache Organisation Recap

- **Type 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
- **Capacity:** C
- **Block size:** b
- **No. of blocks in cache:** $B = C/b$
- **No. of blocks in a set:** N
- **No. of sets:** $S = B/N$

Organization	Number of Ways (N)	Number of Sets (S)
Direct Mapped	1	B
Set Associative	$1 < N < B$	B/N
Fully Associative	B	1

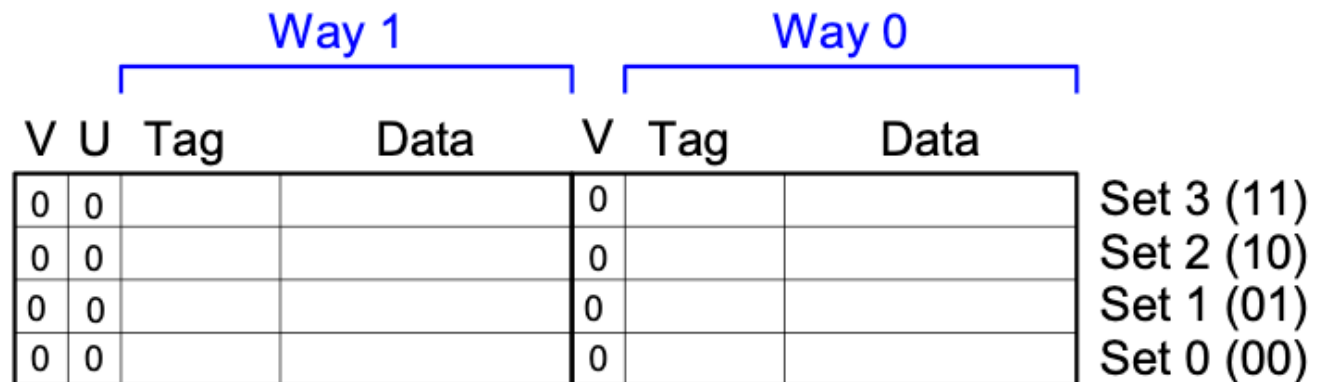
Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

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

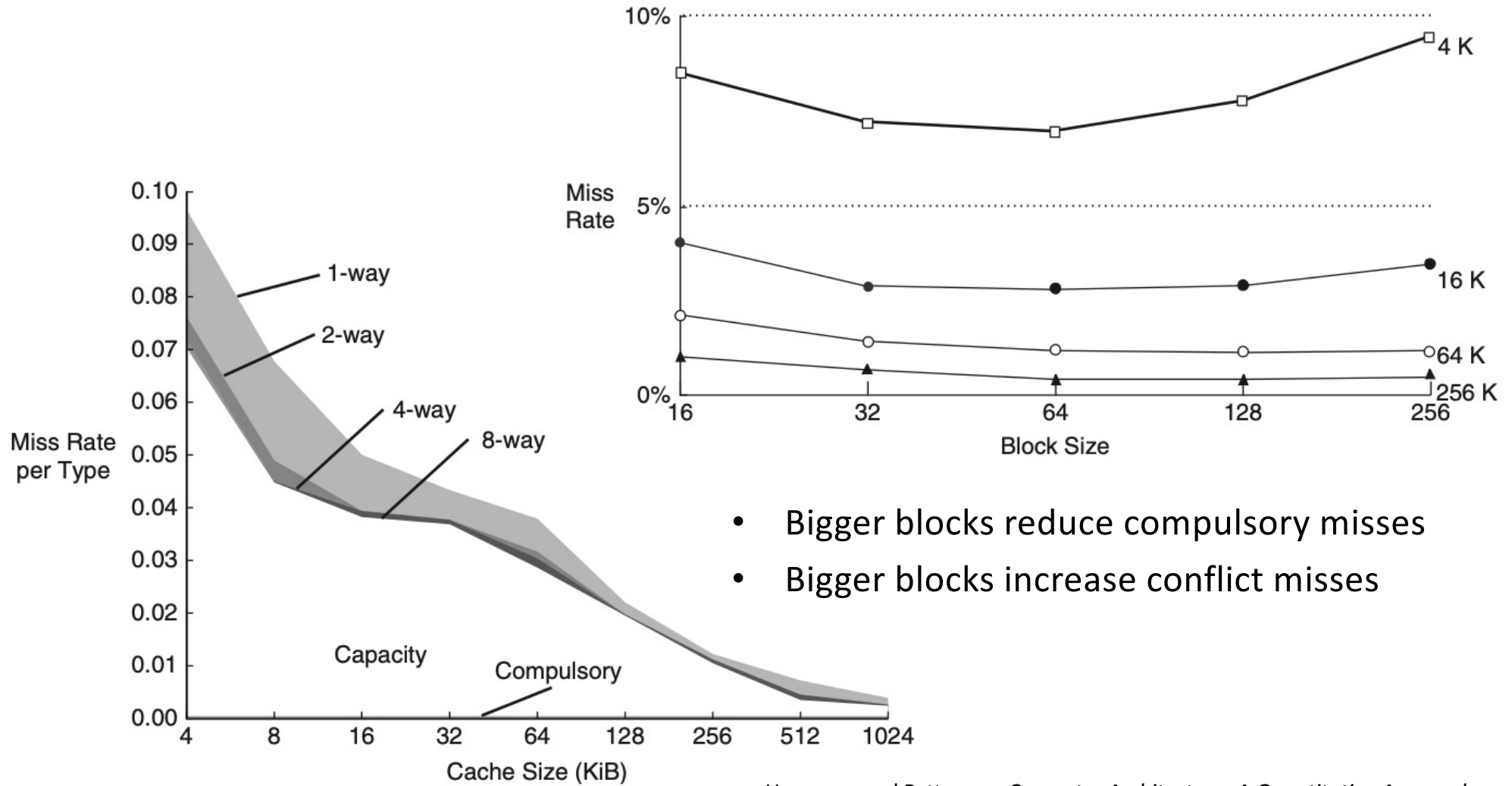
RISC-V assembly

```
lw s1, 0x04(zero)
lw s2, 0x24(zero)
lw s3, 0x54(zero)
```



Based on: "Digital Design and Computer Architecture (RISC-V Edition)"
by Sarah Harris and David Harris (H&H),

Cache Miss Rate vs Sizes



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

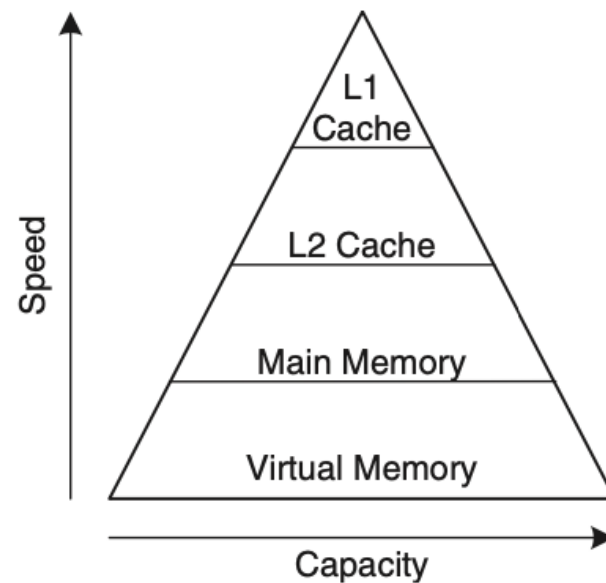
Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed., Morgan Kaufmann, 2012

Write policy for cache memory

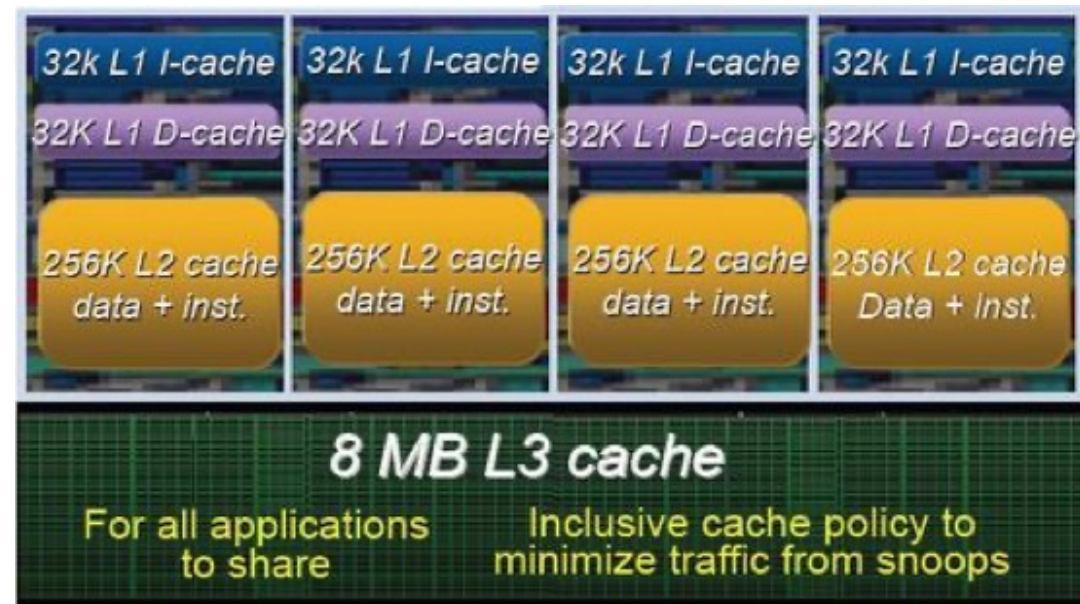
- Same as cache read for hit/miss detection
- If cache misses, cache block is fetched from main memory, the word written to the cache block
- If cache hits, word is simply written to cache block – no need to fetch
- **Problem** – once a word is written, the cache contains data DIFFERENT from the main memory. This is known as **cache coherency** problem.
- There are two ways to handle this:
 - **Write-through cache** – data is written to cache and simultaneously to the main memory
 - **Write-back cache** – data is written to cache and a **dirty bit (D)** associated with the cache block is set. It is written back to main memory ONLY when the block is evicted from the cache.

Multilevel Cache

- 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



Intel i7 Cache Characteristics



Characteristic	L1	L2	L3
Size	32 KB I/32 KB D	256 KB	2 MB per core
Associativity	4-way I/8-way D	8-way	16-way
Access latency	4 cycles, pipelined	10 cycles	35 cycles
Replacement scheme	Pseudo-LRU	Pseudo-LRU	Pseudo-LRU but with an ordered selection algorithm

Summary on Cache

- **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

Based on: “*Digital Design and Computer Architecture (RISC-V Edition)*”
by Sarah Harris and David Harris (H&H),