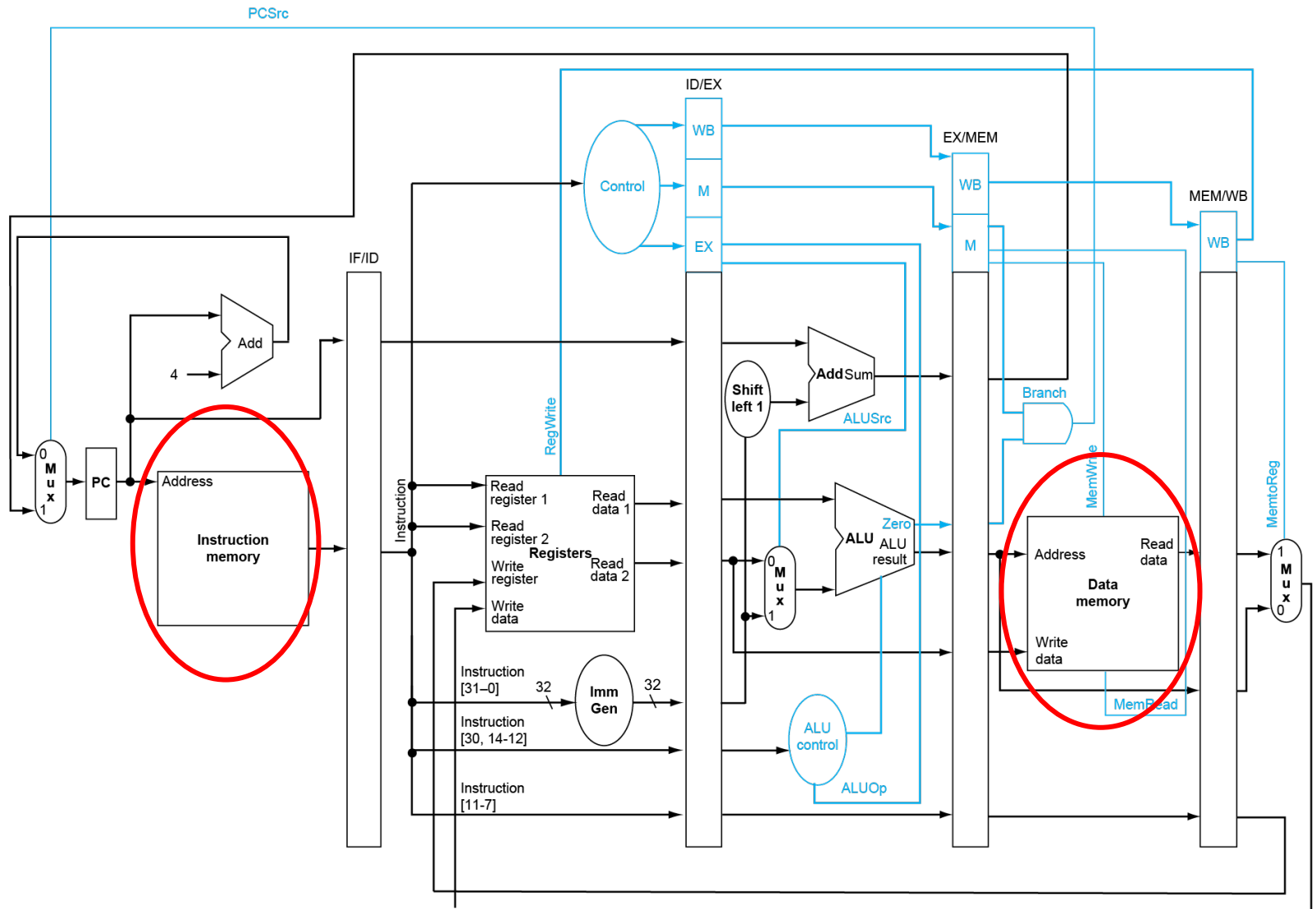


Topic 9

Memory Hierarchy

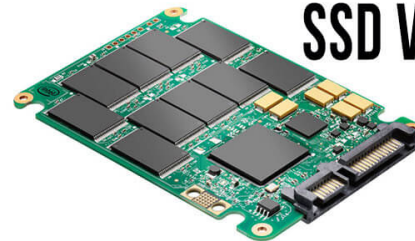
- Cache (1)**

RISC-V Pipeline Architecture



Examples of Memory

- USB
- Disk
- Hard drive
- Flash/SSD
- ...



SSD VS HDD



- FASTER PERFORMANCE
- NO VIBRATIONS OR NOISE
- MORE ENERGY EFFICIENT

- CHEAPER PER GB
- AVAILABLE IN LARGE VERSIONS



Source: Internet

Volatile vs. Non-volatile memory (NVM)

- Volatile memory: stored data is cleared when power is turned off, such as RAM
- Non-volatile memory (NVM): retains stored data after the power is turned off

Examples of Non-Volatile Memory



floppy disk



compact disc



internal hard drive



external hard drive

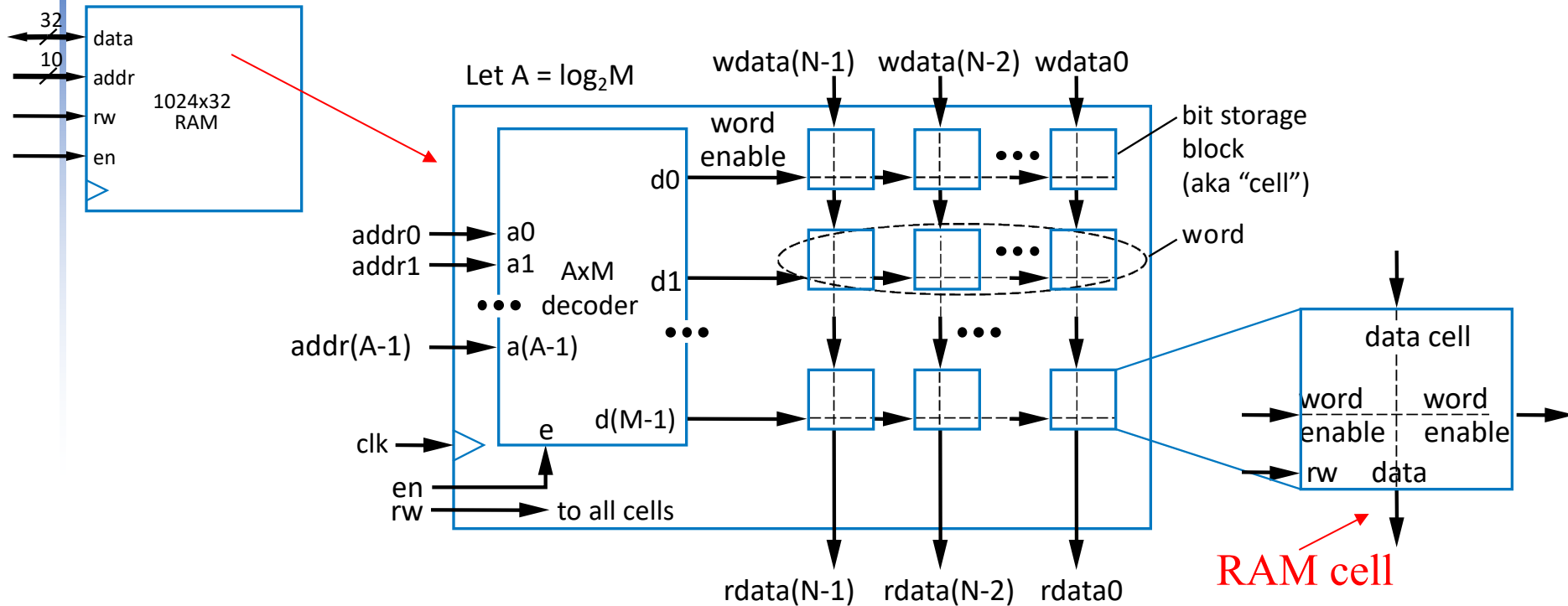


flash drive



SD card

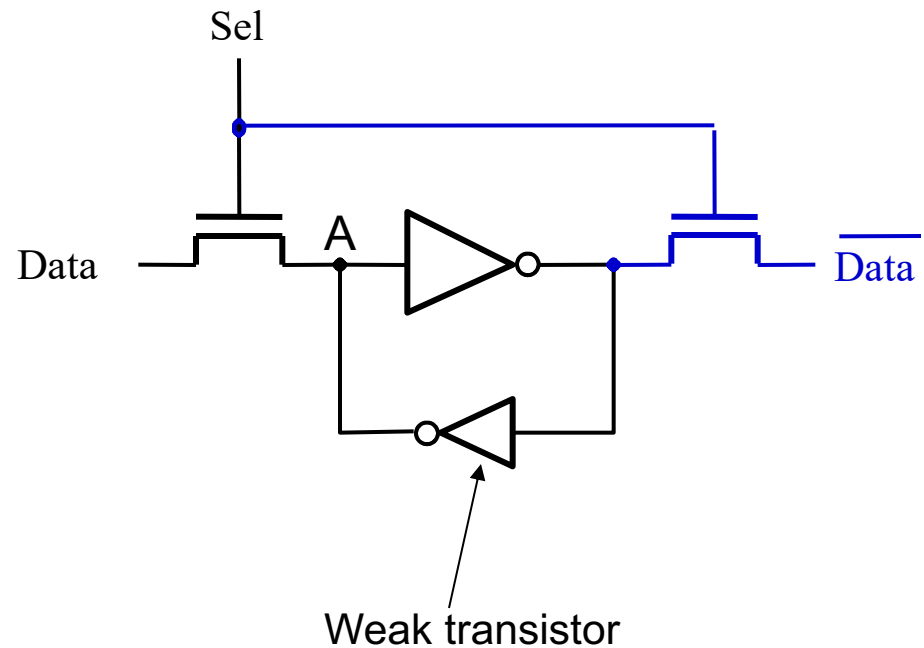
Internal Structure of RAM



- Similar internal structure as register file
 - Decoder enables appropriate word based on address inputs
 - *rw* controls whether cell is written or read

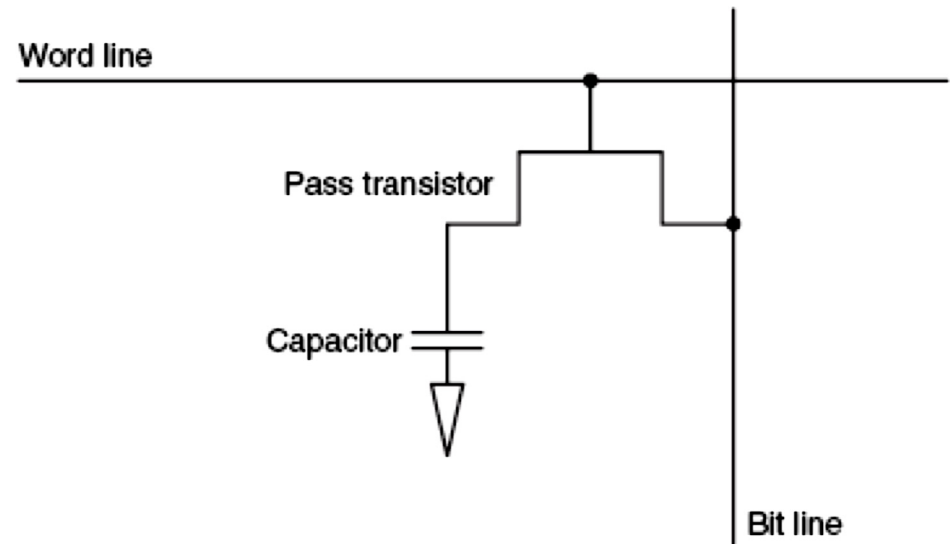
Static RAM (SRAM)

- When Sel = 1, Data is stored and retained in the SRAM cell by the feedback loop
- When Sel = 1, Data can be read out on the same port
- Point A is driven by both the Data transistor and the smaller inverter, but the Data transistor wins because the inverter is implemented using a weak transistor



Dynamic RAM (DRAM)

- Write: turn on word line, charge capacitor through pass transistor by bit line
- Read: charge bit line halfway between high and low, turn on word line, then sense the voltage change on bit line
 - 1 if voltage increases
 - 0 if voltage decreases



Memory Technology

- Static RAM (SRAM)
 - 0.5ns – 2.5ns, \$500 – \$1000 per GB
- Dynamic RAM (DRAM)
 - 50ns – 70ns, \$3 – \$6 per GB
- Magnetic disk (hard drive)
 - 5ms – 20ms, \$0.01 – \$0.02 per GB
- Ideal memory
 - Access time of SRAM
 - Capacity and cost/GB of disk

Wishful Memory

“Ideally one would desire an indefinitely large memory capacity such that any particular word would be immediately available. . .”

- Arthur Burks, Herman Goldstine, and John von Neumann, 1946

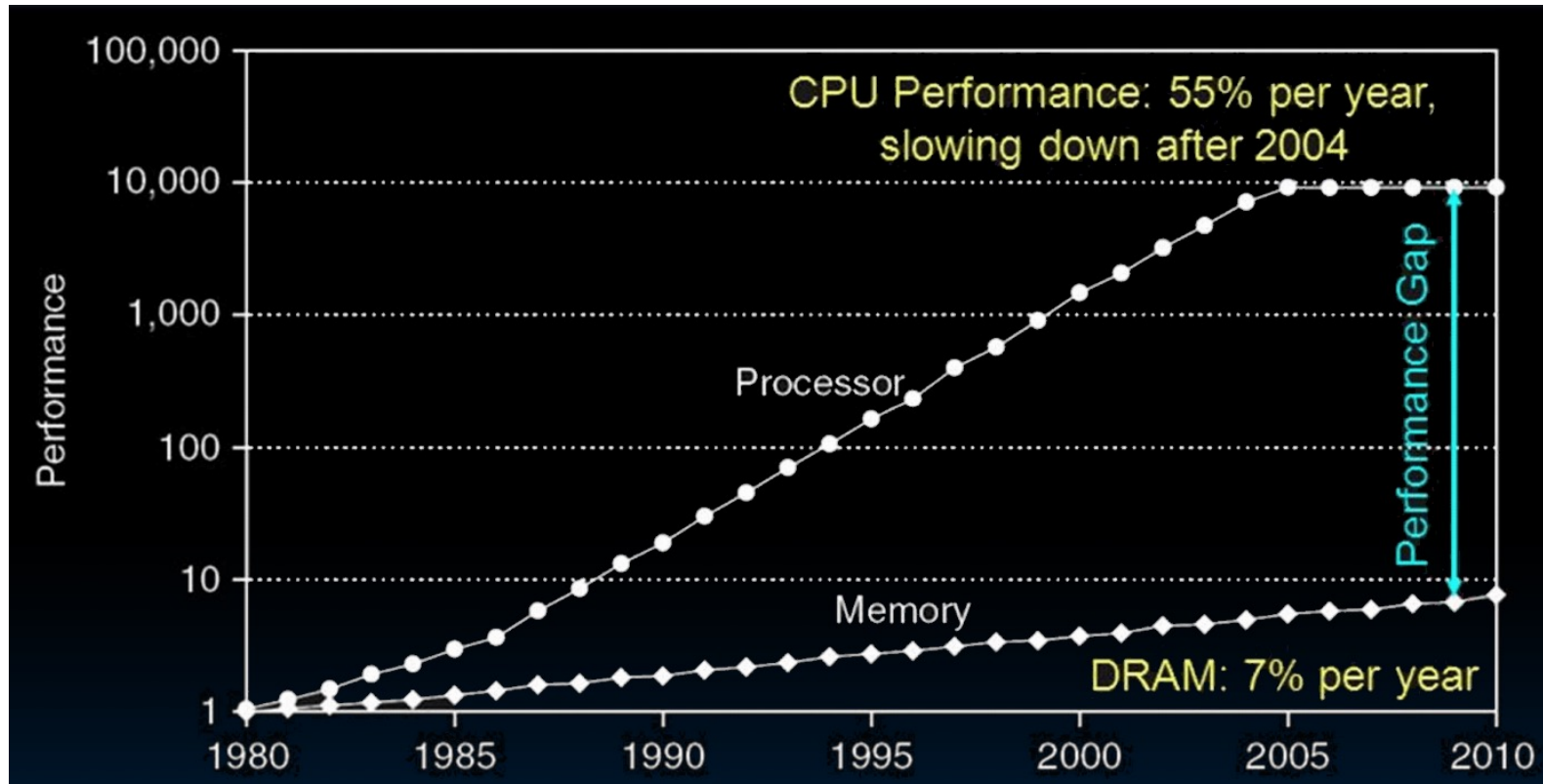


- So far we have **imagined/assumed**
 - A processor with ideal instruction and data memories
 - One cycle per stage...

The Reality

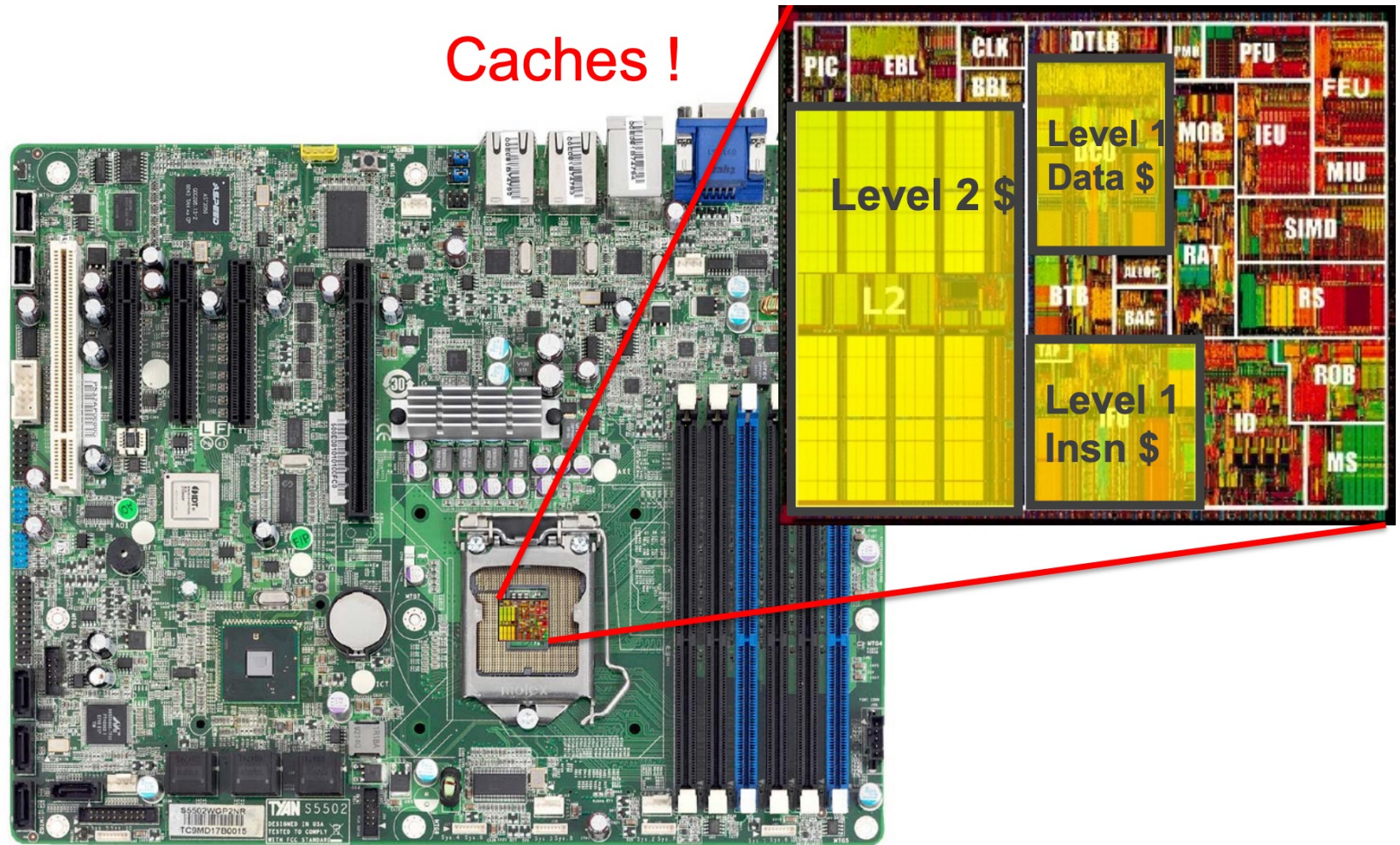
- Hard to find memory technology that is
 - Big
 - Fast
 - Affordable

The problem in reality



- 1980 microprocessor executes ~one instruction in same time as DRAM access
- 2020 microprocessor executes ~1000 instructions in same time as DRAM access
- Slow DRAM access has disastrous impact on CPU performance!

What is the solution here?



Intel Pentium 3, 1999

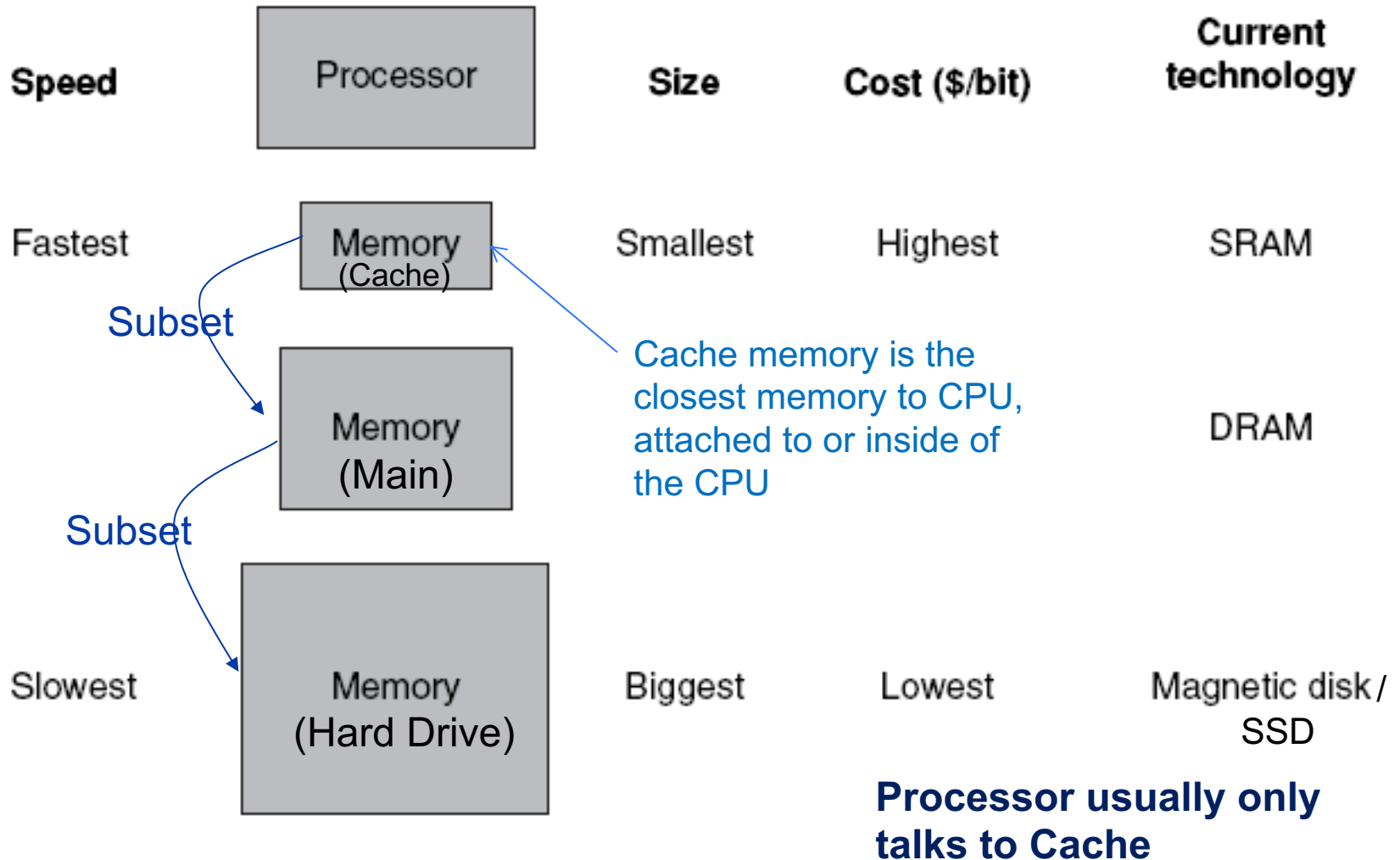
Cache

A CPU cache is a hardware cache used by the CPU of a computer to reduce the average cost (time or energy) to access data from the main memory. A cache is a smaller, faster memory (than main memory), located closer to a processor core, which stores copies of the data from frequently used main memory locations. Most CPUs have a hierarchy of multiple cache levels (L1, L2, often L3, and rarely even L4), with different instruction-specific and data-specific caches at level 1.

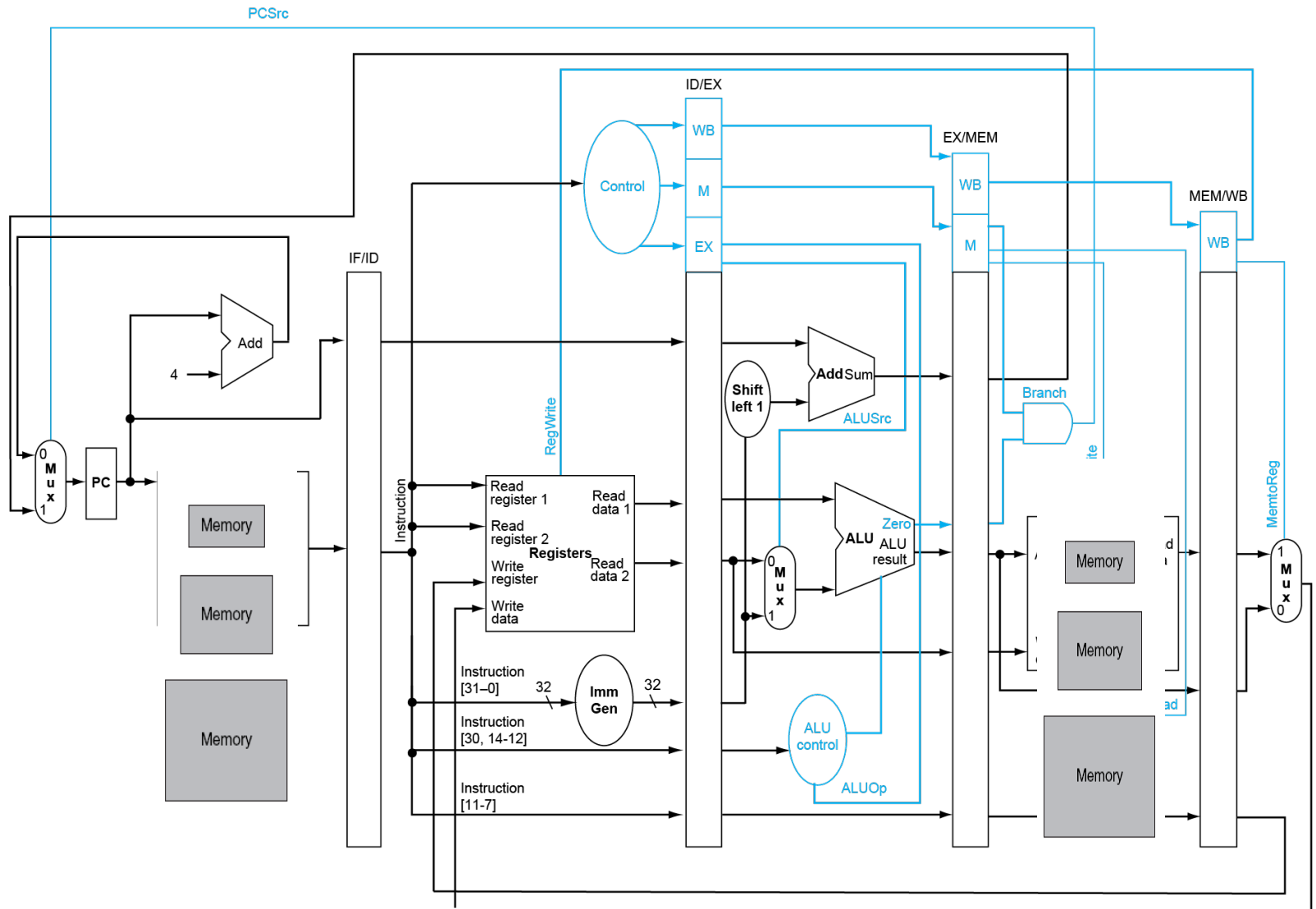
(source: Wikipedia)

Apple M1 chip	
General information	
Launched	November 10, 2020 ^[1]
Designed by	Apple Inc.
Common manufacturer(s)	TSMC
Product code	APL1102 ^[2]
Performance	
Max. CPU clock rate	3.2 GHz ^[1]
Cache	
L1 cache	320 KB per core (performance cores, 192 instructions + 128 data) 192 KB per core (efficient cores, 128 instructions + 64 data)
L2 cache	12 MB (performance cores) 4 MB (efficient cores)
Architecture and classification	
Application	Desktop (Mac Mini, iMac), Notebook (MacBook family), Tablet (iPad Pro)
Min. feature size	5 nm
Microarchitecture	"Firestorm" and "Icestorm" ^[1]
Instruction set	ARMv8.4-A
Physical specifications	
Transistors	16 billion
Cores	8 (4× high-performance + 4× high-efficiency)
GPU(s)	Apple-designed integrated graphics (up to 8 cores)
History	
Predecessor	Intel Core and Apple T2 chip (Mac) Apple A12Z Bionic (iPad Pro)

Memory Hierarchy



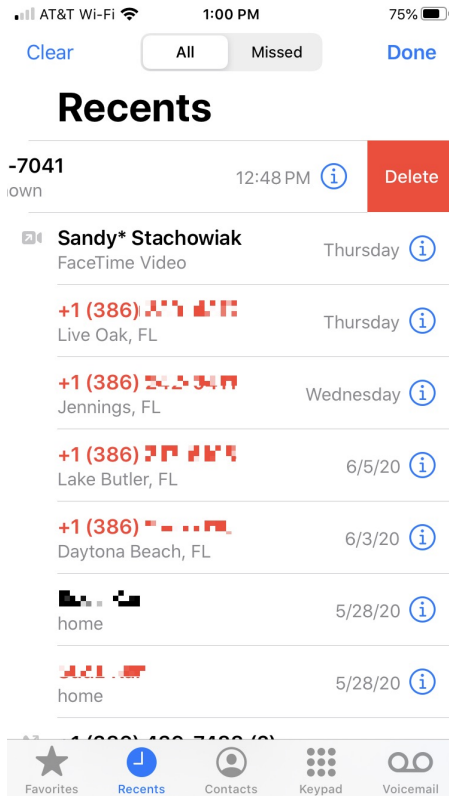
Memory Hierarchy in RISC-V



Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
 - Items that are accessed recently are likely to be accessed again soon
 - e.g., instructions in a loop
- Spatial locality
 - Items near those that are accessed recently are likely to be accessed
 - E.g., sequential instruction access, array data

Life is full of Locality



- Last Called
- Speed Dial
- Favorites Contacts

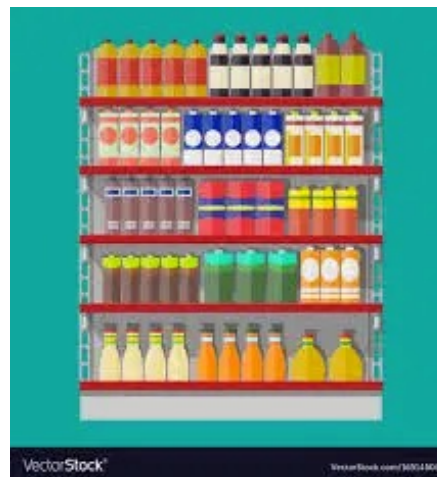
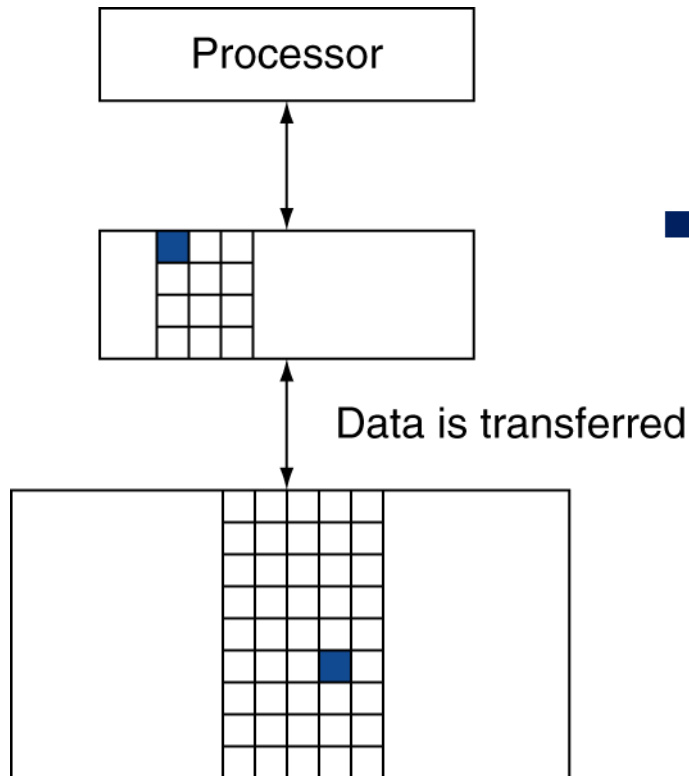


Image: Internet

Principle of Memory Access

- Taking Advantage of Locality
- If a word of data/instruction is referenced
 - Copy this recently accessed word (temporal locality) and nearby items (spatial locality) together as a **block** from lower level memory to higher level memory
 - E.g.: Hard Drive → Main memory, or
Main memory → Cache

Access the Memory Hierarchy



- Memory hierarchy is accessed by CPU through cache
- Unit of Data transfer: **block**
 - aka line
 - Unit of data referencing to take advantage of temporal and spatial locality
 - May be one or multiple words
 - Block also has an address

Block, Word, Byte Addresses

- Byte address vs. word address (assume 16-bit address)
 - Example: one word has 4 bytes

Word number
(simplified word address)

Byte offset

Byte Address	Contents
1111_1111_0000_00 00	Byte 0
1111_1111_0000_00 01	Byte 1
1111_1111_0000_00 10	Byte 2
1111_1111_0000_00 11	Byte 3

14 bits 2 bits

- Word address vs. block address (block number)
 - Example: one block has 2 words (8 bytes)

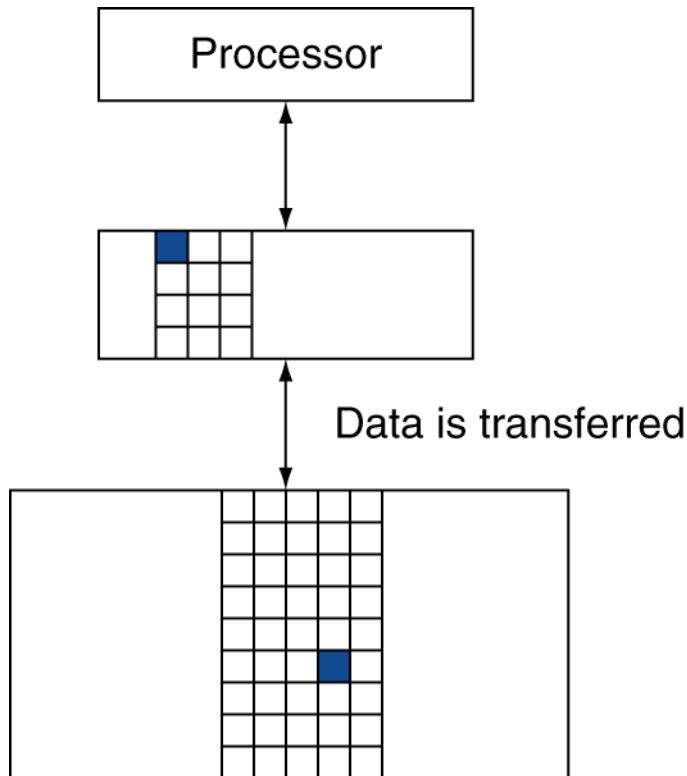
Block number
(Block address)

Word offset

Word Number	Contents
1111_1111_0000_0 0	Word 0
1111_1111_0000_0 1	Word 1

13 bits 1 bit

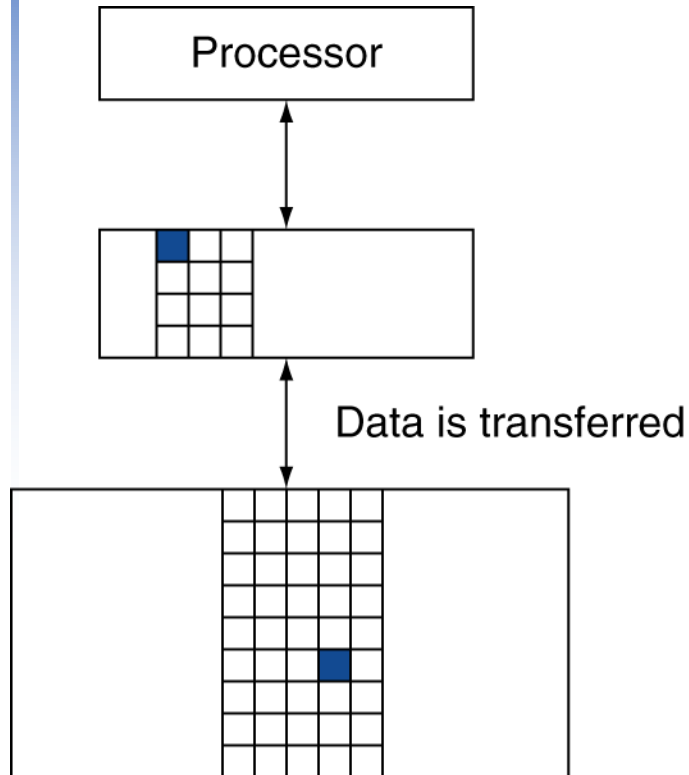
Access the Memory Hierarchy



■ Hit

- If accessed data is present in upper level, access satisfied by upper level
- Hit rate: hits/accesses
- Hit time: time to access a memory including
 - Time to determine whether a hit or miss, and
 - Time to pass block to requestor

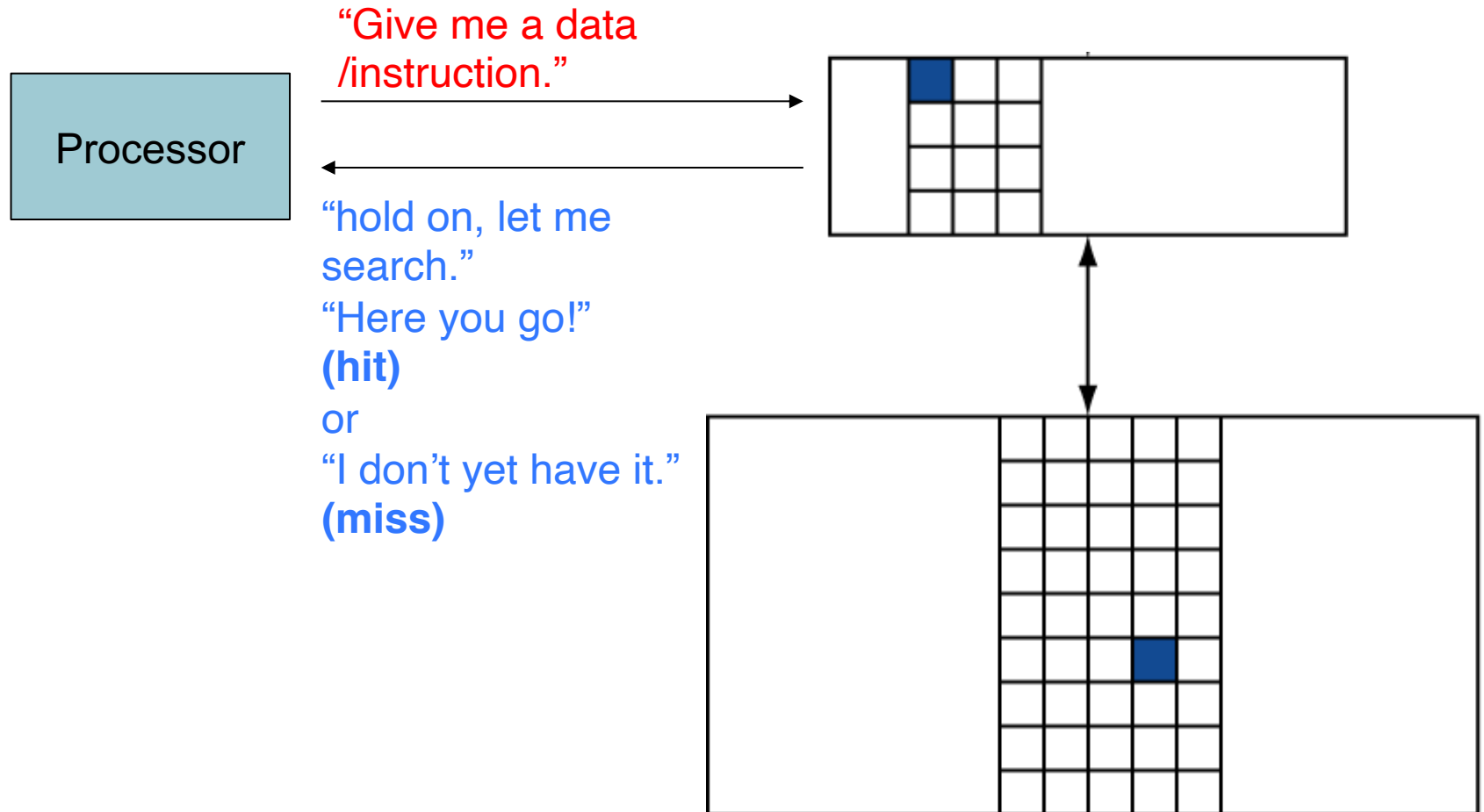
Access the Memory Hierarchy



■ Miss

- If accessed data is absent
 - Block copied from lower to higher level
 - Then accessed data supplied from upper level
- Miss penalty: time taken
- Miss rate: $\text{misses/accesses} = 1 - \text{hit rate}$

Cache Hit or Miss



Miss Penalty

- Miss (time) penalty
 - Time to fetch a block from lower level upon a miss, including
 - Time to access the block
 - Time to transfer it between levels
 - Time to overwrite the higher level block
 - Time to pass block to requestor (CPU)

Cache Miss

- Assume a cache with 1-word blocks X_1, \dots, X_{n-1} .
Now CPU requests X_n

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_3

a. Before the reference to X_n

X_4
X_1
X_{n-2}
X_{n-1}
X_2
X_n
X_3

b. After the reference to X_n

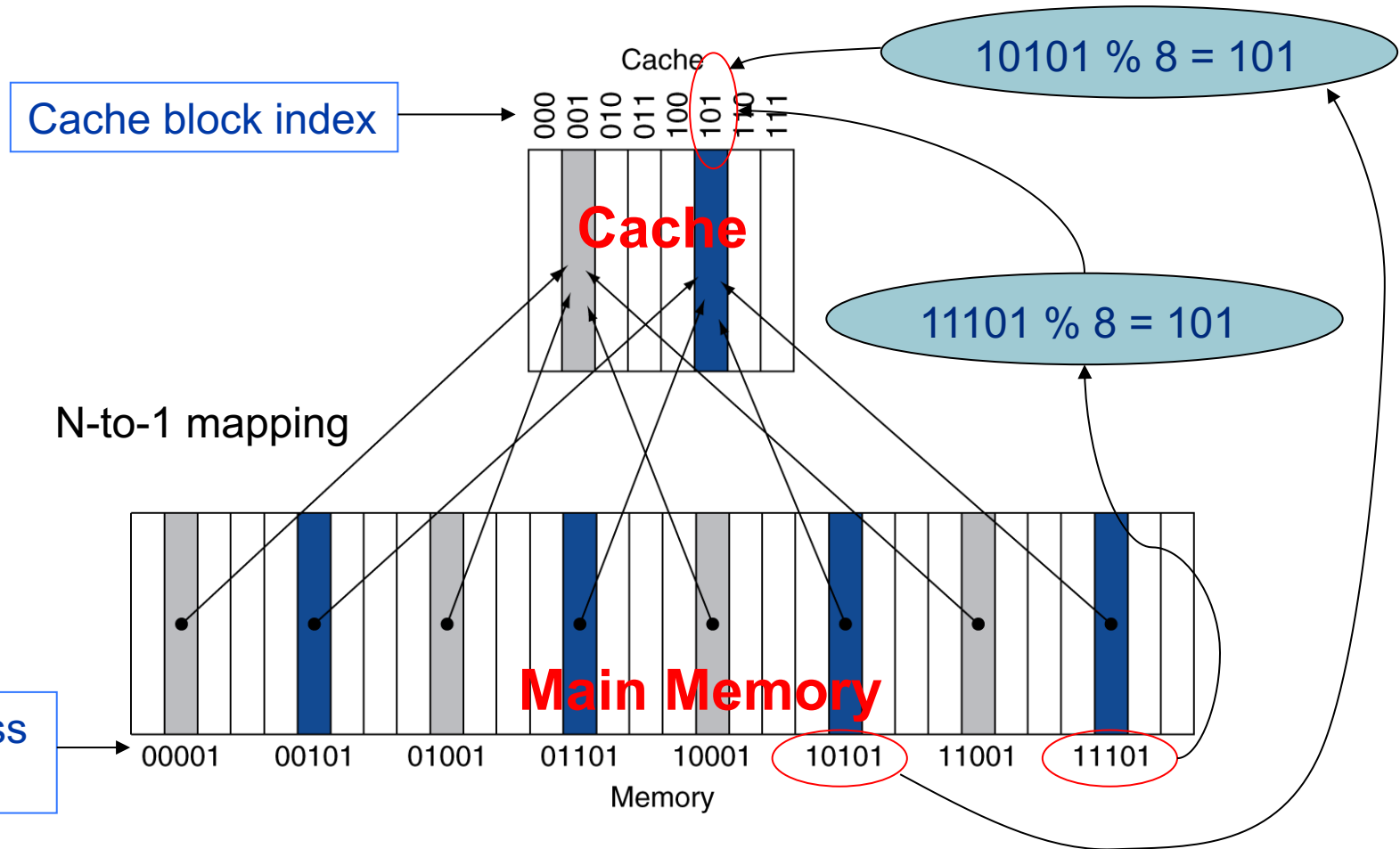
- Miss. Then X_n is brought in from lower level
- But
 - How do we know if there is a hit or miss?
 - Where do we look?

Direct Mapped Cache

- Direct mapped cache: each (main) memory location corresponds to **one** choice in cache
- **Multiple** memory locations correspond to **one** block in cache: n-to-1 mapping
- How to map?
- Location of a block in cache is determined by address of the requested word
 - Given word address, we can get block address (block number)
 - Cache location (or cache block index) =
(Block address in memory) % (Number of blocks in cache) =
lower M bits of block address in memory
 $M = \log_2(\text{Number of blocks in cache})$

Direct Mapped Cache

- Number of blocks in a cache is a power of 2
- Low-order bits of block address in memory = cache index



Tags in Cache

- How do we know which block of data is present in cache since it's n-to-1 mapping?
 - Store part of block address together with the data when the data is copied to higher level
 - Only need to store the high-order bits of memory block address, excluding the **lower M bits of block address** ($M = \log_2(\text{Number of blocks in cache})$)
 - Called **tag**

Direct Mapped Cache Example

- 8-block cache
- 1 word per block (word address = block address)

Cache Contents

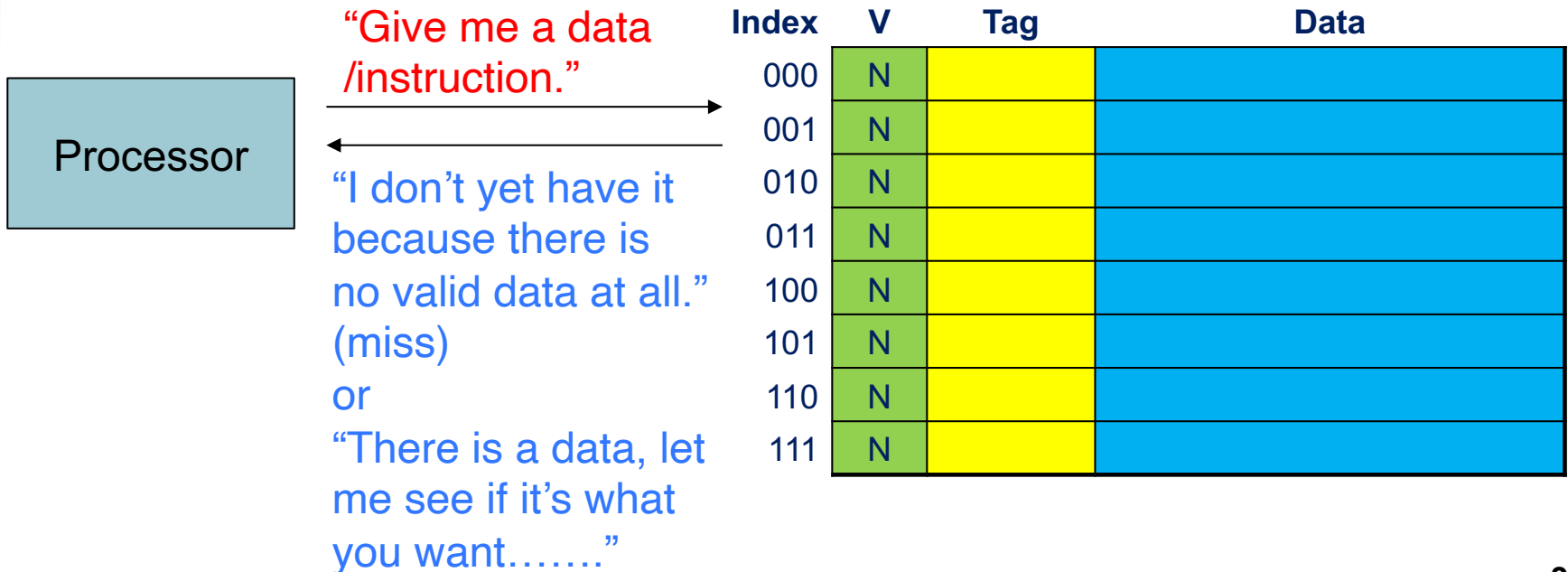
1 word per block

Index	Tag	Data
000		
001		
010		
011		
100		
101		
110		
111		

8 blocks

Search Data in Cache

- Waste searching time if there is no valid data in a location
 - **Valid bit**: 1 = has data, 0 = empty
 - Initially 0 (N), to improve search speed



Cache Example 1

- 8-block cache
- 1 word per block (word address = block address)
- direct mapped
- Assuming 7-bit byte addresses sent by CPU

1 word per block

	Index	V	Tag	Data
8 blocks	000	N		
	001	N		
	010	N		
	011	N		
	100	N		
	101	N		
	110	N		
	111	N		

Cache Example 1 (cont.)

Simplified word address (decimal) without byte offset 00

lw R1 ← mem[22]			
Requested mem addr	Block (word) Addr	Hit/miss	Cache block
10110 00	10 110	Miss	110

Simplified word address (22)

Byte offset

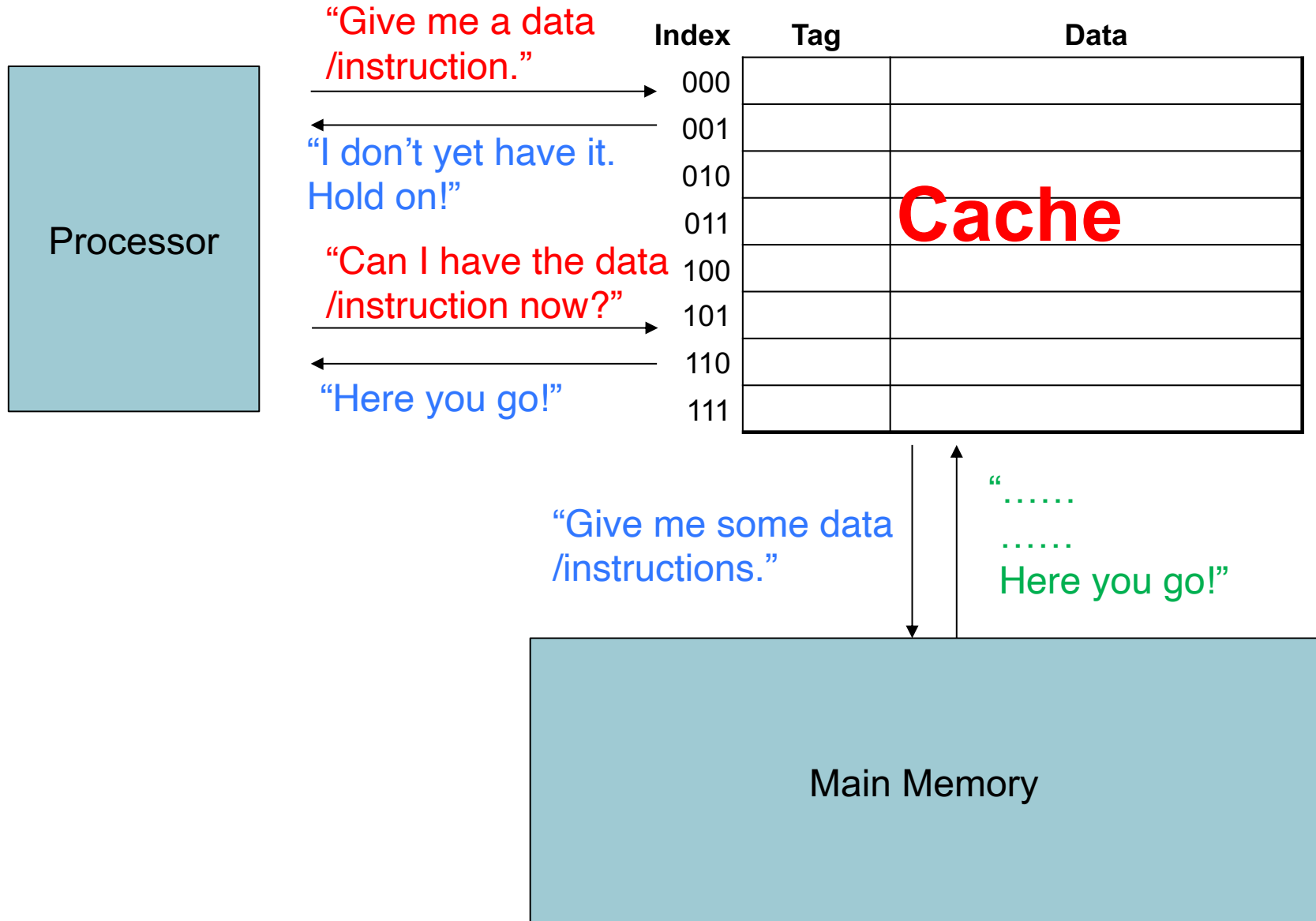
Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Cache Miss

- On cache hit, pipeline proceeds normally
- On cache miss
 - Stall the CPU pipeline
 - Using processor control unit and a cache controller
 - Freeze registers and wait for memory
 - Fetch block from next level of hierarchy

Cache Miss



Cache Example 1 (cont.)

lw R1 ← mem[22]

Requested mem addr	Block (word) Addr	Hit/miss	Cache block
10110 00	10 110	Miss	110

byte offset 00
are omitted

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	0x05ACF011
111	N		

Cache memory

Word Addr	Data
00000(0)	0x81230431
00001(1)	0xABCD3305
...	...
10101(21)	0x12345678
10110(22)	0x05ACF011
...	...
11110(30)	0x00000000
11111(31)	0x00000000

Main memory

Cache Example 1 (cont.)

lw R2 ← mem[26]			
Requested mem addr	Block (word) Addr	Hit/miss	Cache block
11010 00	11 010	Miss	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	0x5DC60007
011	N		
100	N		
101	N		
110	Y	10	0x05ACF011
111	N		

Cache memory

Word Addr	Data
00000 (0)	0x81230431
00001 (1)	0xABCD3305
...	...
10101 (21)	0x12345678
10110 (22)	0x05ACF011
...	...
11010 (26)	0x5DC60007
...	...
11110 (30)	0x00000000
11111 (31)	0x00000000

Main memory

Cache Example 1 (cont.)

	Requested mem addr	Block (word) Addr	Hit/miss	Cache block
lb R3 ← mem[22]byte1	10110 01	10 110	Hit	110
sw R4 → mem[26]	11010 00	11 010	Hit	010

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	(R4)
011	N		
100	N		
101	N		
110	Y	10	0x05ACF011
111	N		

Hit due to temporal locality

R3 FFFFFFFF0

Cache Example 1 (cont.)

Requested mem Addr	Block (word) Addr	Hit/miss	Cache block
1000000(lw)	10 000	Miss	000
0001100(sw R5)	00 011	Miss	011
1000000(lw)	10 000	Hit	000

Index	V	Tag	Data
000	Y	10	0x8765ABCD
001	N		
010	Y	11	(R4)
011	Y	00	0xFFFF0002 → (R5)
100	N		
101	N		
110	Y	10	0x05ACF011
111	N		

Word Addr	Data
00000(0)	0x81230431
00001(1)	0xABCD3305
00010(2)	0xFFFF0001
00011(3)	0xFFFF0002
...	...
10000(16)	0x8765ABCD
...	...
10101(21)	0x12345678
10110(22)	0x05ACF011
...	...
11010(26)	0x5DC60007
...	...
11110(30)	0x00000000
11111(31)	0x00000000

Cache Example 1 (cont.)

Requested mem Addr	Block (word) Addr	Hit/miss	Cache block
1001000(lw)	10 010	Miss	010

Currently occupied by Mem[26]

Index	V	Tag	Data
000	Y	10	0x8765ABCD
001	N		
010	Y	10	(R4)→0x0000FF00
011	Y	00	(R5)
100	N		
101	N		
110	Y	10	0x05ACF011
111	N		

Word Addr	Data
00000(0)	0x81230431
00001(1)	0xABCD3305
00010(2)	0xFFFF0001
00011(3)	0xFFFF0002
...	...
10000(16)	0x8765ABCD
10001(17)	0x12345678
10010(18)	0x0000FF00
...	...
11010(26)	0x5DC60007
...	...
11110(30)	0x00000000
11111(31)	0x00000000