

CPS101
Computer Organization and Programming
Lecture 13: The Memory System

Robert Wagner

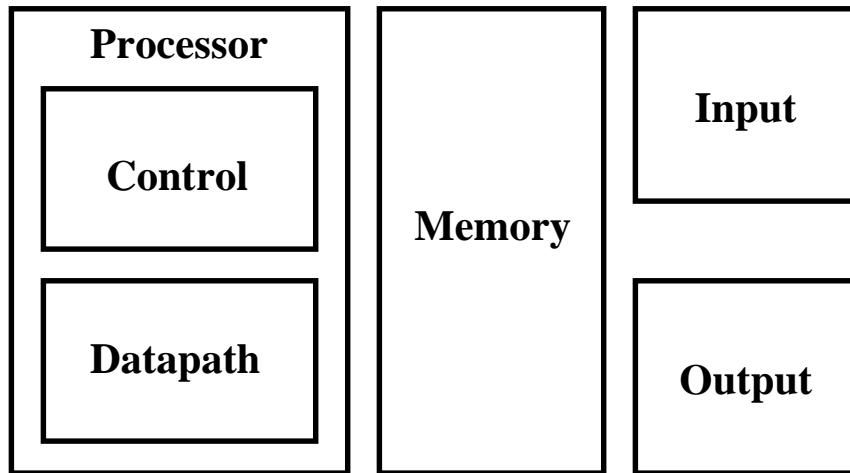
Outline of Today's Lecture

- **Memory System: the BIG Picture?**
- **Memory Technology: SRAM**
- **Memory Technology: DRAM**
- **A Real Life Example: SPARCstation 20's Memory System**
- **Summary**
- **Memory Hierarchy**
- **Direct Mapped Cache**

Chapter 7, Appendix B, pages 26-33

The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



- Today's Topic: Memory System

Memory Technologies

- **Random Access:**
 - “Random” is good: access time is the same for all locations
 - **DRAM: Dynamic Random Access Memory**
 - High density, low power, cheap, slow
 - Dynamic: content must be “refreshed” regularly
 - **SRAM: Static Random Access Memory**
 - Low density, high power, expensive, fast
 - Static: content will last “forever” (until lose power)

- **“Not-so-random” Access Technology:**
 - Access time varies from location to location and from time to time
 - Examples: Disk, CDROM

- **Sequential Access Technology: access time linear in location (e.g., Tape)**

Memory Hierarchy

- The various technologies listed have varying speed and cost

	Registers	SRAM	DRAM	Disk
Nsec/access	3	5-25	50-80	5-10 million
Cost per MByte	\$250,000	\$100-250	\$5-10	\$0.10-0.20

- Programs are known to spend most of their time accessing a small part of their address space -- Principle of Locality
- Motivates designing memory as a HIERARCHY, with small, fast levels holding recently accessed data (which you hope is likely to be accessed again soon), while larger, slower-access levels hold larger and larger pieces of the total memory the program needs -- Disk holding it all
- Scheme gives the ILLUSION that all program memory can be accessed equally quickly, by having OS and hardware move data items from level to level, based on the access pattern of the program
- TAPE is generally so slow that it is reserved only for daily or monthly backups, to protect users from disk drive failures and viruses -- some applications may need it to store huge data bases, also.

Random Access Memory (RAM) Technology

- **Why do computer professionals need to know about RAM technology?**
 - **Processor performance is usually limited by memory latency and bandwidth.**
 - **Latency: The time it takes to access a word in memory.**
 - **Bandwidth: The average speed of access to memory (Words/Sec).**
 - **As IC densities increase, lots of memory will fit on processor chip**
 - **Tailor on-chip memory to specific needs.**
 - **Instruction cache**
 - **Data cache**
 - **Write buffer**

- **What makes RAM different from a bunch of flip-flops?**
 - **Density: RAM is much denser**
 - **Speed: RAM access is slower than flip-flop (register) access.**

Technology Trends

Capacity Speed

Logic: 2x in 3 years 2x in 3 years

DRAM: 4x in 3 years 1.4x in 10 years

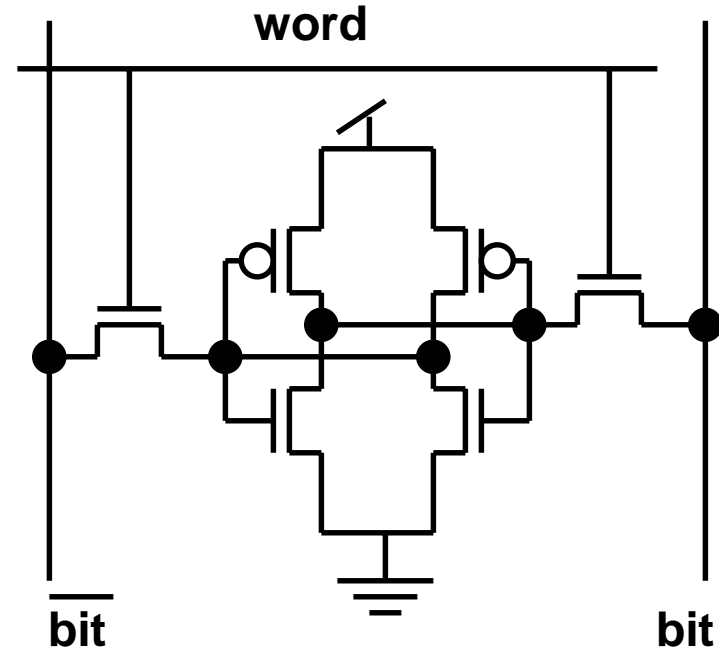
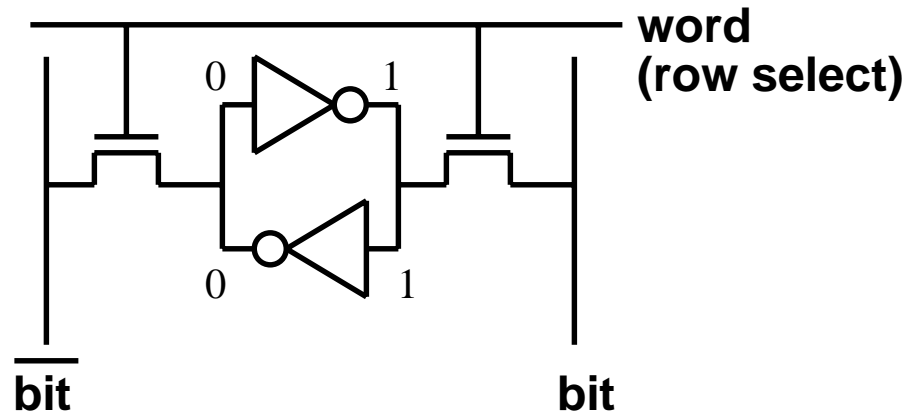
Disk: 2x in 3 years 1.4x in 10 years

<u>Year</u>	<u>Size</u>	<u>Cycle Time</u>
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1! (Size growth from 1980 to 1995)
2:1! (Cycle Time reduction from 1980 to 1995)

Static RAM Cell

6-Transistor SRAM Cell



◦ Write:

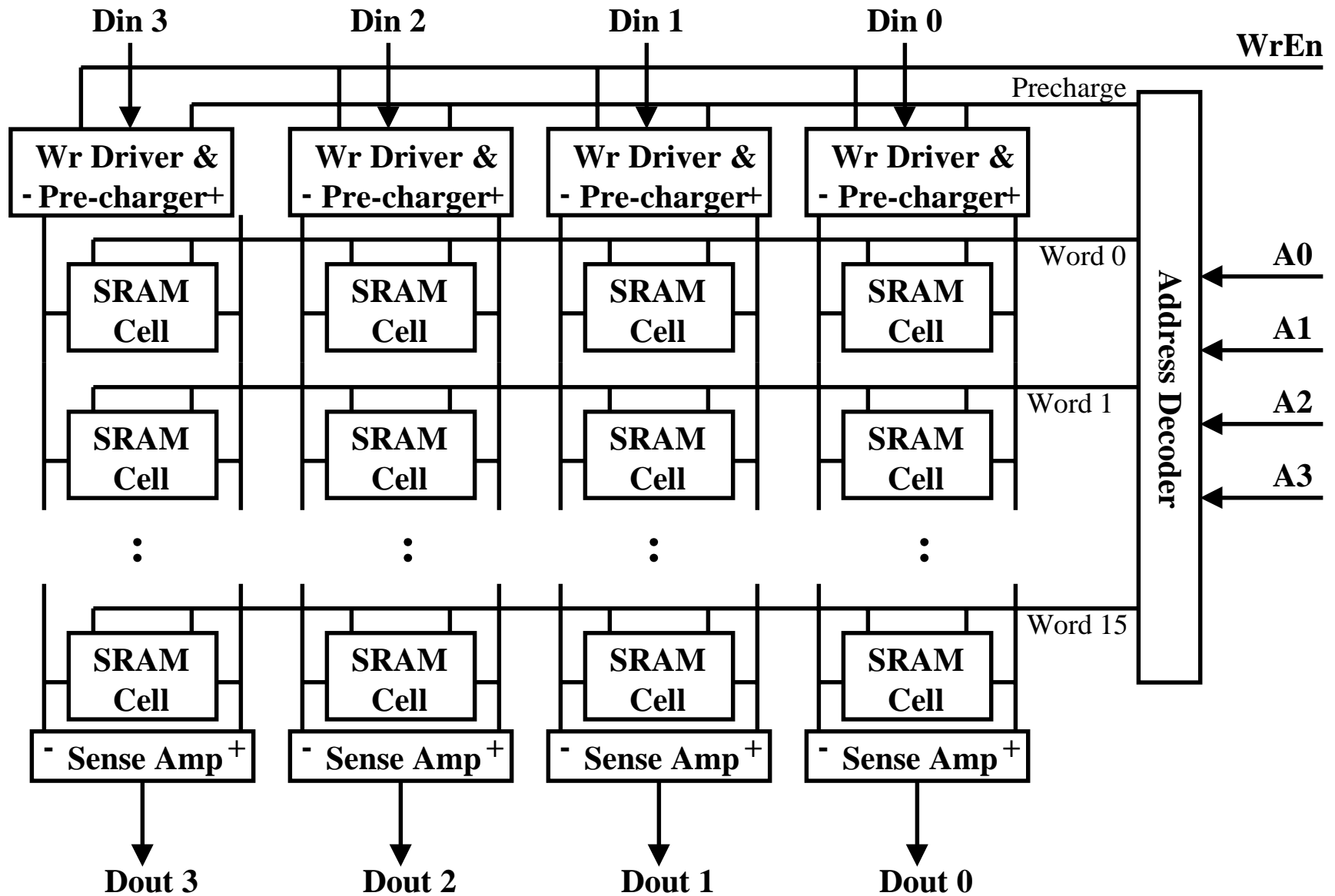
1. Drive bit lines ($\overline{\text{bit}}=1, \text{bit}=0$)
- 2.. Select row
3. Strength of BIT line signal over-rides latch

◦ Read:

1. Precharge bit and $\overline{\text{bit}}$ to Vdd
- 2.. Select row
3. Cell pulls one line low

4. Sense amp on column detects difference between bit and $\overline{\text{bit}}$

Typical SRAM Organization: 16-word x 4-bit



1-Transistor Memory Cell (DRAM)

◦ Write:

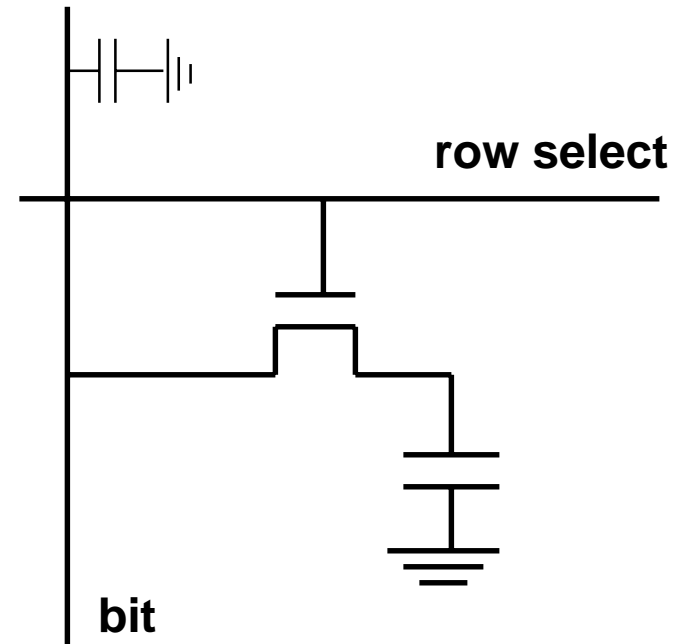
- 1. Drive bit line
- 2.. Select row

◦ Read:

- 1. Precharge bit line to Vdd
- 2.. Select row
- 3. Cell and bit line share charges
 - Very small voltage changes on the bit line
- 4. Sense (fancy sense amp)
 - Can detect changes of ~1 million electrons
- 5. Write: restore the value

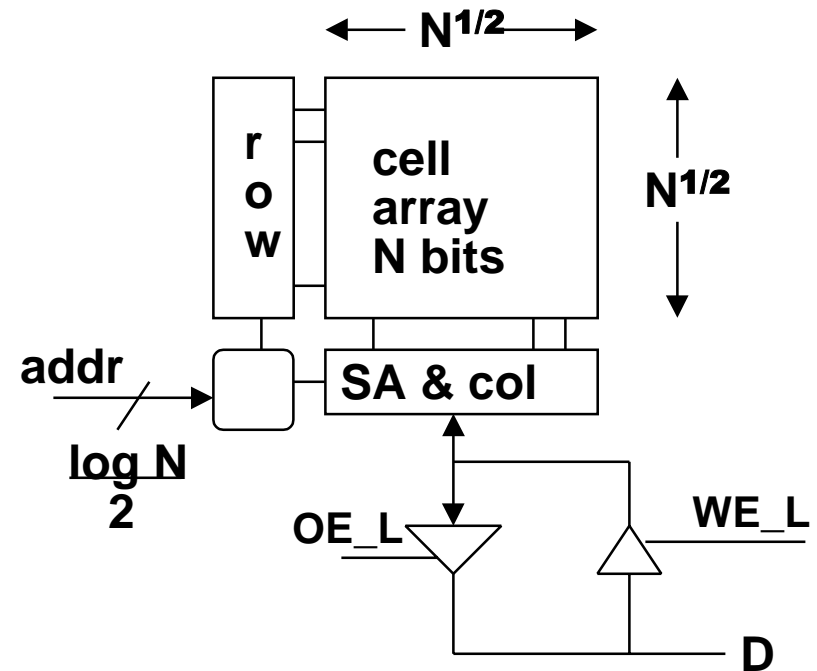
◦ Refresh

- 1. Just do a dummy read to every cell.

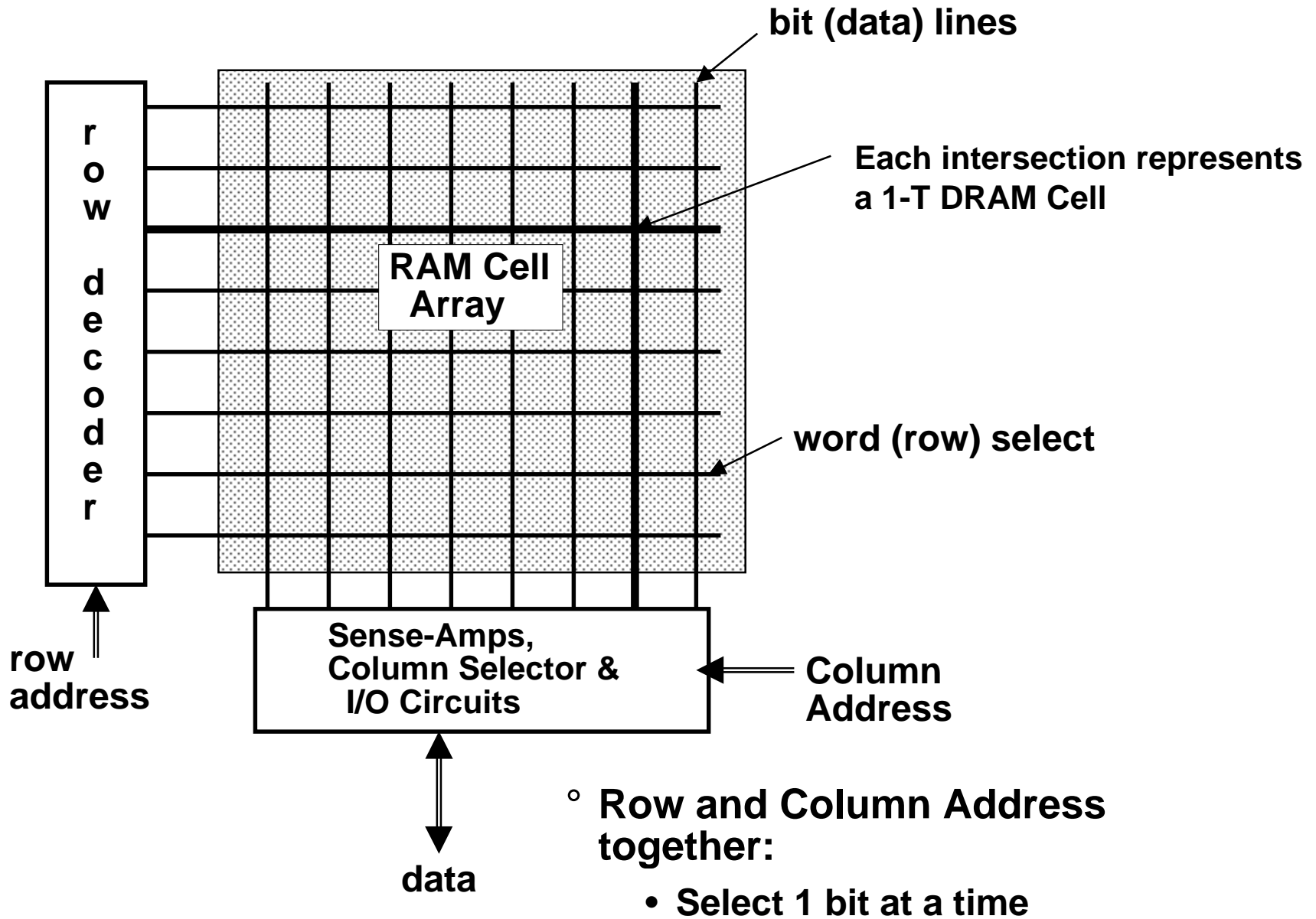


Introduction to DRAM

- **Dynamic RAM (DRAM):**
 - Refresh required
 - Very high density
 - Low power (.1 - .5 W active, .25 - 10 mW standby)
 - Low cost per bit
 - Pin sensitive (few pins):
 - Output Enable (OE_L)
 - Write Enable (WE_L)
 - Row address strobe (ras)
 - Col address strobe (cas)

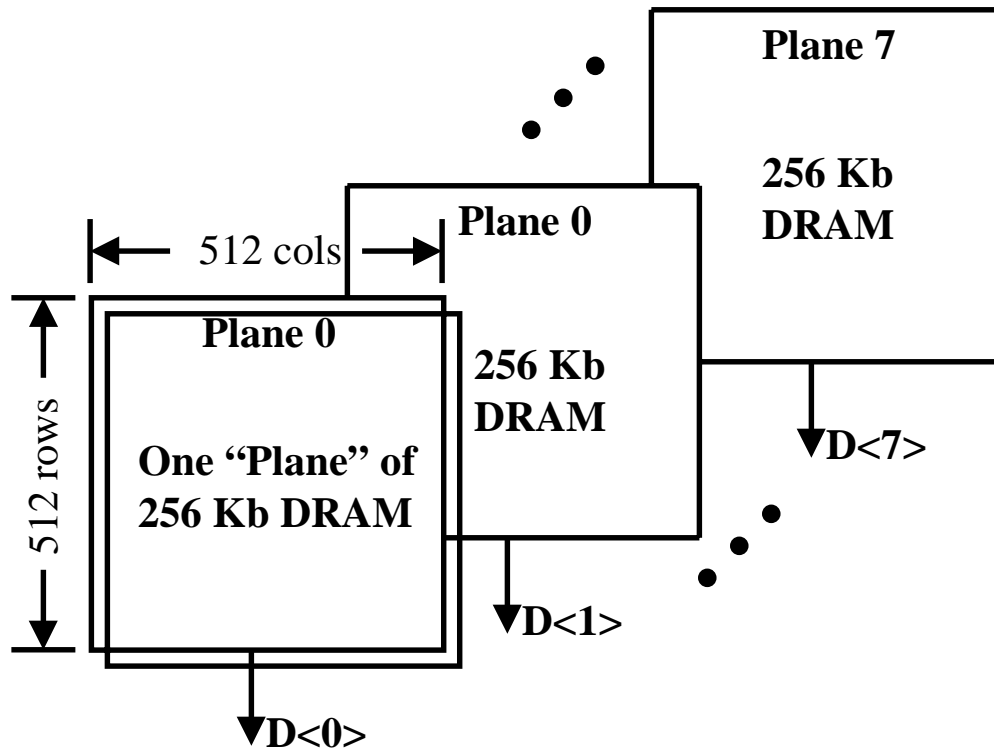


Classical DRAM Organization (square)



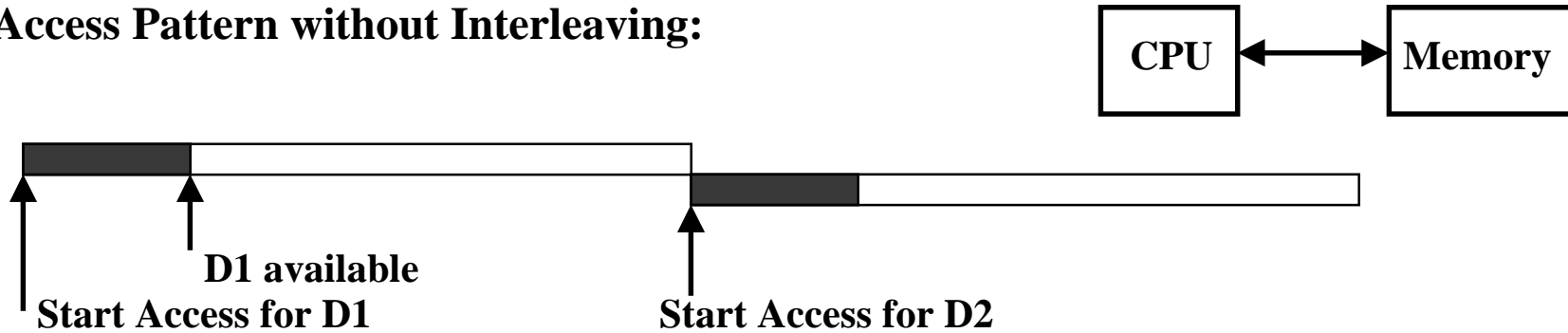
Typical DRAM Organization

- Typical DRAMs: access multiple bits in parallel
 - Example: 2 Mb DRAM = 256K x 8 = 512 rows x 512 cols x 8 bits
 - Row and column addresses are applied to all 8 planes in parallel

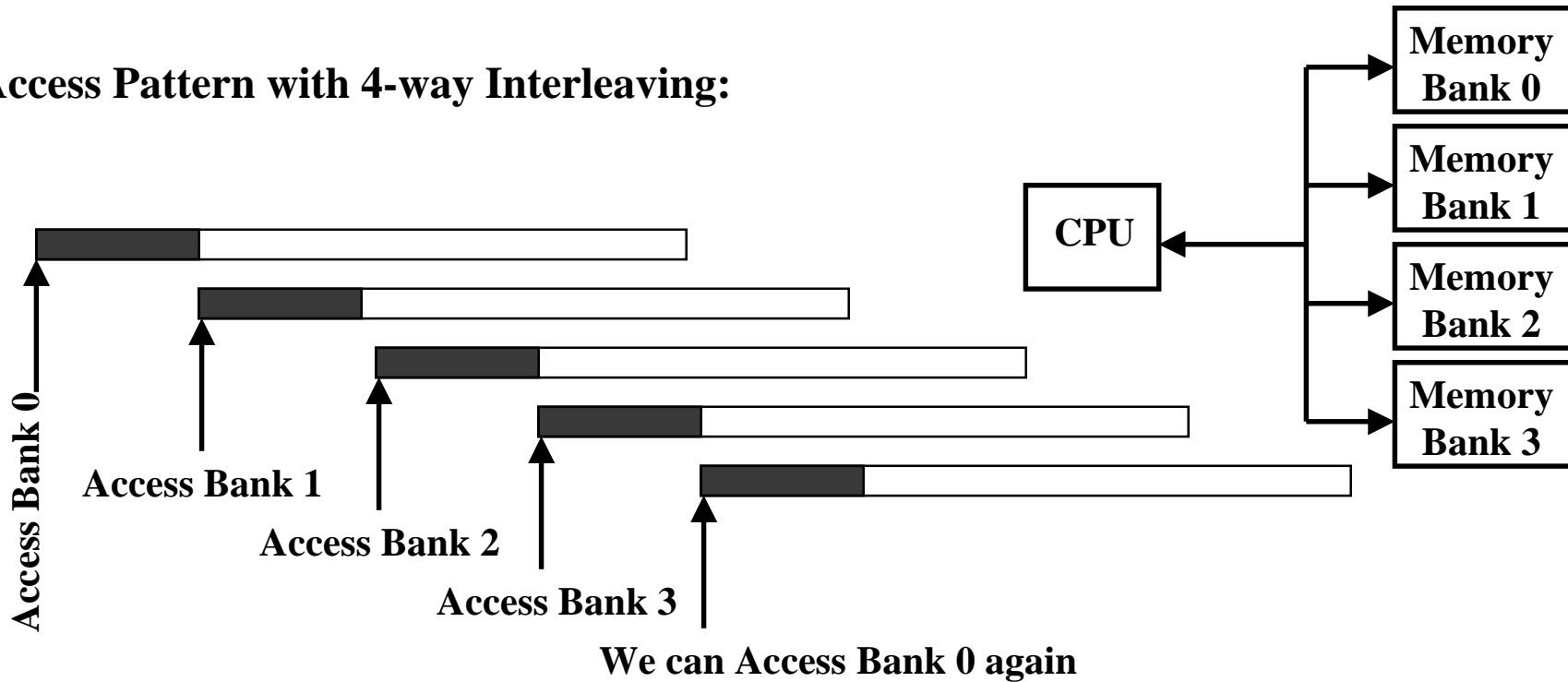


Increasing Bandwidth - Interleaving

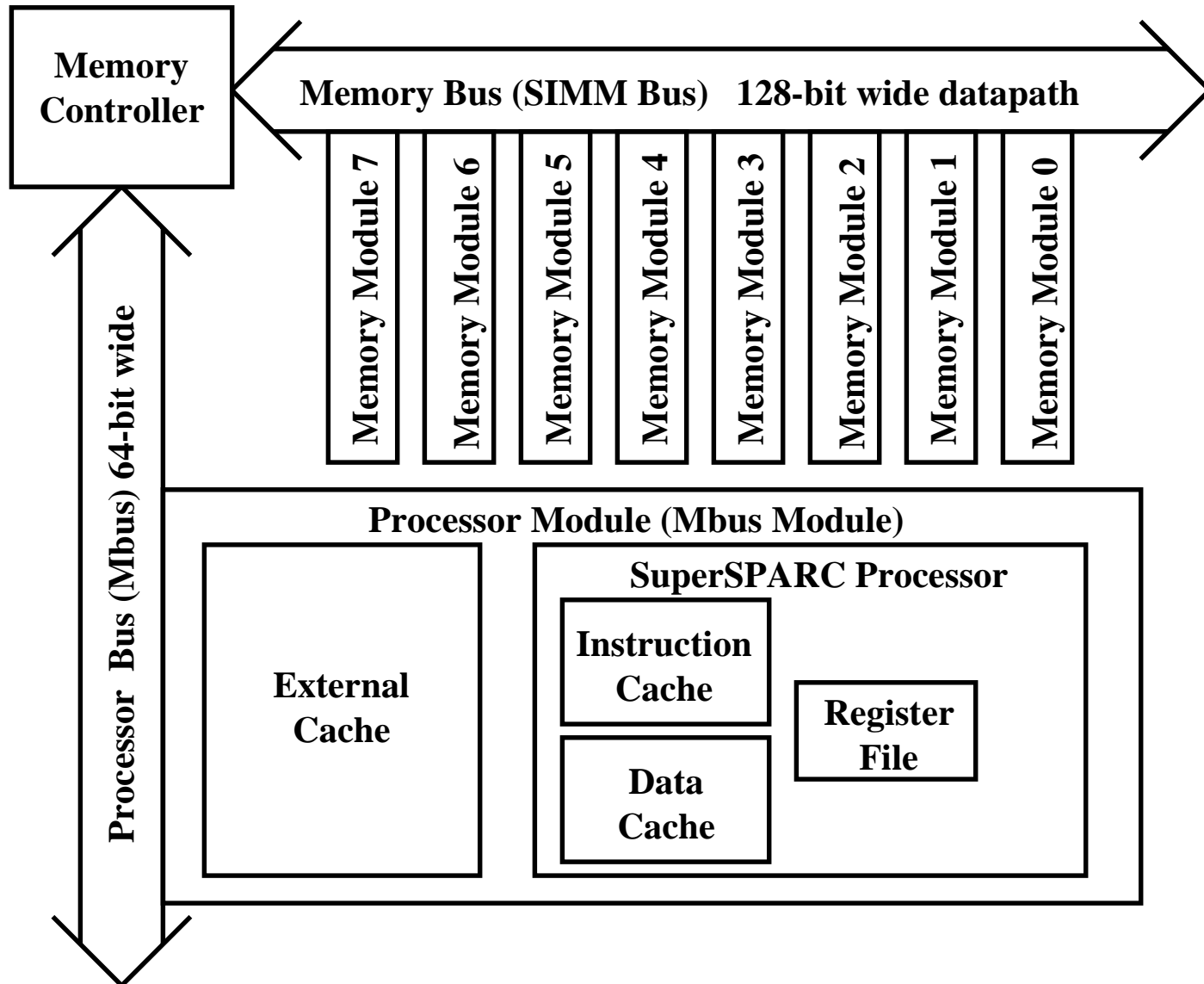
Access Pattern without Interleaving:



Access Pattern with 4-way Interleaving:



SPARCstation 20's Memory System Overview



Principle of Locality: A Summary

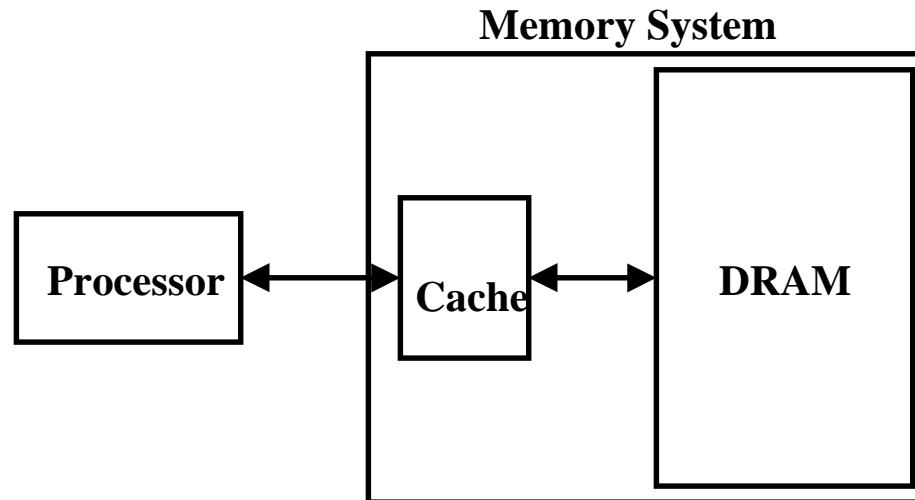
- **Two Different Types of Locality are observed:**
 - **Temporal Locality (Locality in Time):** If an item is referenced, it will tend to be referenced again soon.
 - **Spatial Locality (Locality in Space):** If an item is referenced, items whose addresses are close by tend to be referenced soon.

- **By taking advantage of the principle of locality:**
 - **Present the user with as much memory as is available in the cheapest technology.**
 - **Provide access at the speed offered by the fastest technology.**

- **DRAM is slow but cheap and dense:**
 - **Good choice for presenting the user with a BIG memory system**

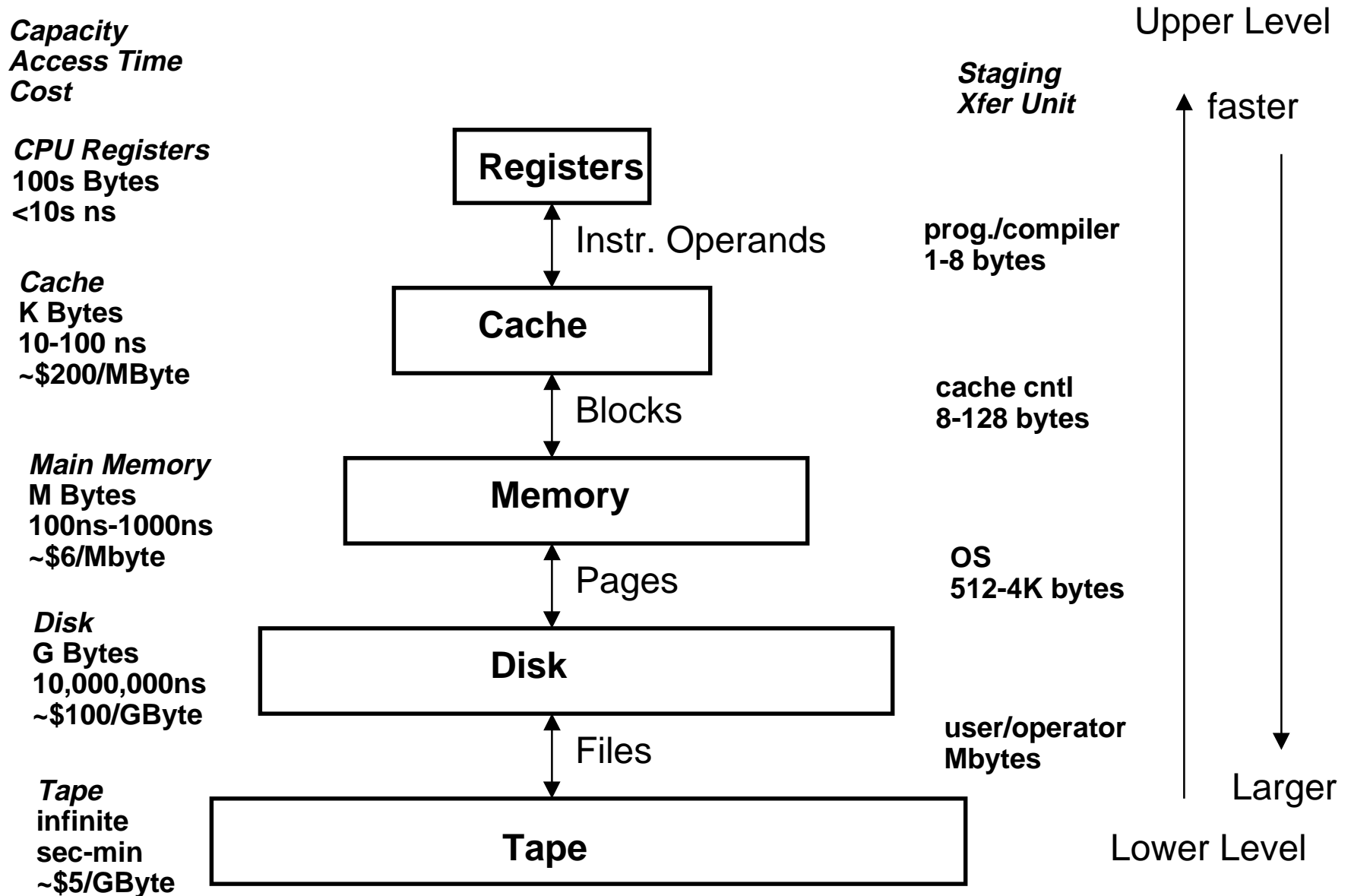
- **SRAM is fast but expensive and not very dense:**
 - **Good choice for providing the user FAST access time.**

The Motivation for Caches



- **Motivation:**
 - Large memories (DRAM) are slow
 - Small memories (SRAM) are fast
- **Make the *average access time* small by:**
 - Servicing most accesses from a small, fast memory.
- **Reduce the *bandwidth* required of the large memory**

Levels of the Memory Hierarchy



The Principle of Locality



- **The Principle of Locality:**

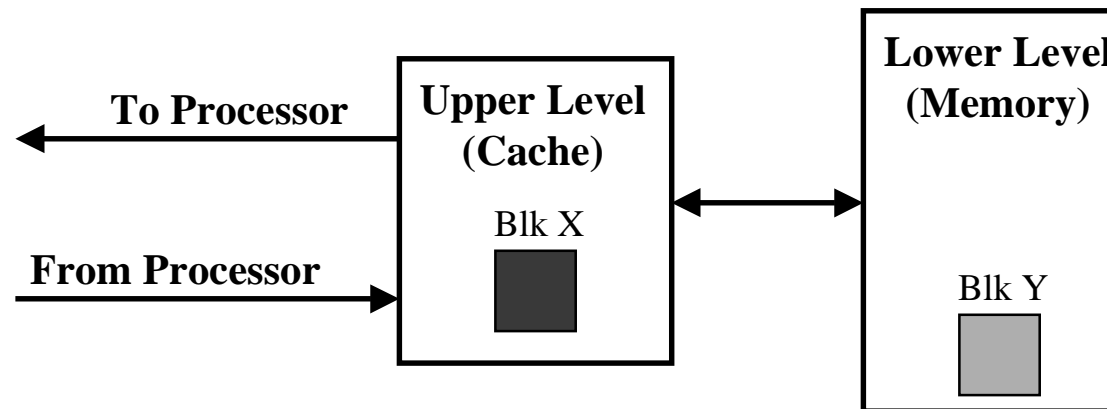
- Programs access a relatively small portion of their address space during any period of a few hundred instructions.
- Example: 90% of time in 10% of the code

- **Two Different Types of Locality:**

- **Temporal Locality (Locality in Time):** If an item is referenced, it will tend to be referenced again soon.
- **Spatial Locality (Locality in Space):** If an item is referenced, items whose addresses are close by tend to be referenced soon.

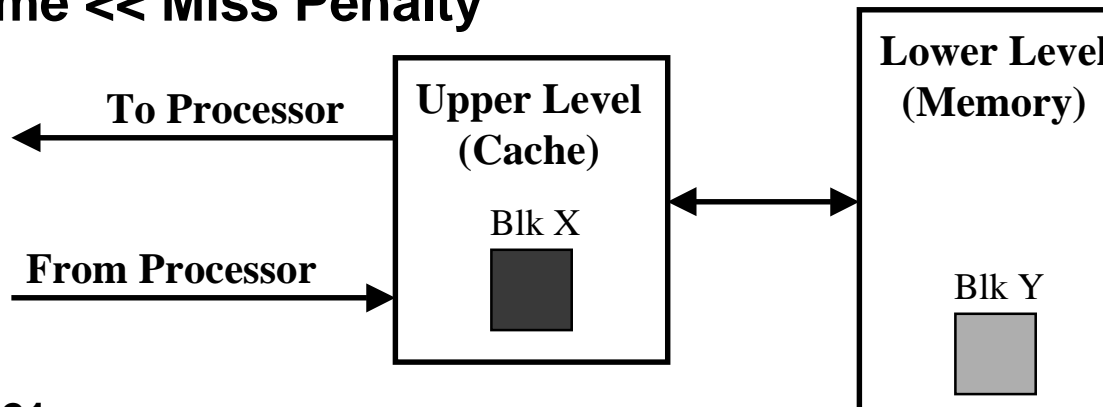
Memory Hierarchy: Principles of Operation

- At any given time, data is copied between only 2 adjacent levels:
 - Upper Level (Cache) : the one closer to the processor
 - Smaller, faster, and uses more expensive technology
 - Lower Level (Memory): the one further away from the processor
 - Bigger, slower, and uses less expensive technology
- **Block:**
 - The minimum unit of information that can either be present or not present in the top level of the hierarchy



Memory Hierarchy: Terminology

- **Hit: data appears in some block in the upper level (example: Block X)**
 - **Hit Rate:** the fraction of memory accesses found in the upper level
 - **Hit Time:** Time to access the upper level which consists of
RAM access time + Time to determine hit/miss
- **Miss: data needs to be retrieved from a block in the lower level (Block Y)**
 - **Miss Rate** = $1 - (\text{Hit Rate})$
 - **Miss Penalty** = Time to replace a block in the upper level +
Time to deliver the block to the processor
- **Hit Time \ll Miss Penalty**

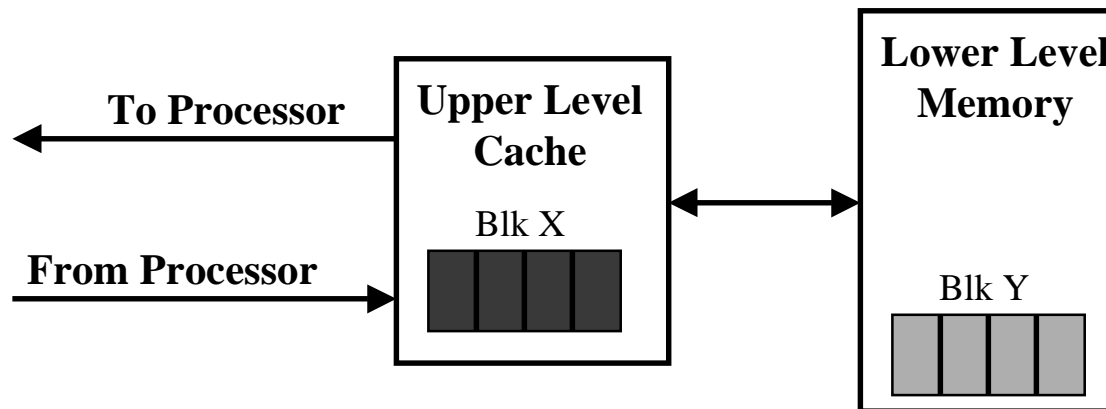


Basic Terminology: Typical Values

Typical Values	
Block (line) size	16 - 128 bytes
Hit time	1 - 4 cycles
Miss penalty	10 - 100 cycles (and increasing)
(access time)	(10-100 cycles)
(transfer time)	(5 - 22 cycles)
Miss rate	1% - 20%
Cache Size	8 KB - 4 MB

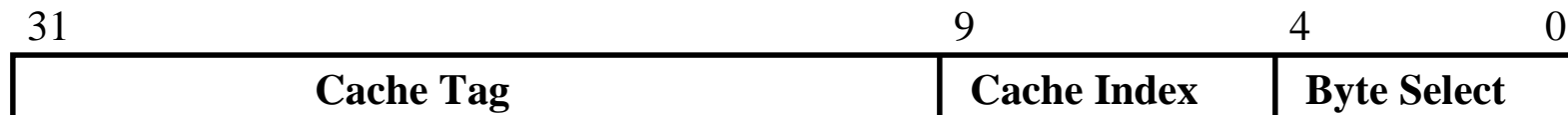
How Does Cache Work?

- **Temporal Locality (Locality in Time):** If an item is referenced, it will tend to be referenced again soon.
 - Keep more recently accessed data items closer to the processor
- **Spatial Locality (Locality in Space):** If an item is referenced, items whose addresses are close by tend to be referenced soon.
 - Move blocks consisting of contiguous words to the cache



Cache Implementation Principles

- A memory address must be examined quickly, to determine if the addressed location **IS** or **IS NOT** in the cache
- Cache = “look-up table”
 - **BUT:** Hardware has time for only **ONE** reference to the **SRAM** memory which implements the cache
- Hardware approximation: Use **SOME** bits of the address to determine **WHERE** in cache data at that address **MUST** be stored
 - Called “cache index bits”
- Since many addresses have **SAME** index bits, cache also stores the rest of the address
 - Called “cache tag bits”
- **Lowest order bits of address are used to select BYTE within a cache BLOCK**



Direct Mapped Cache

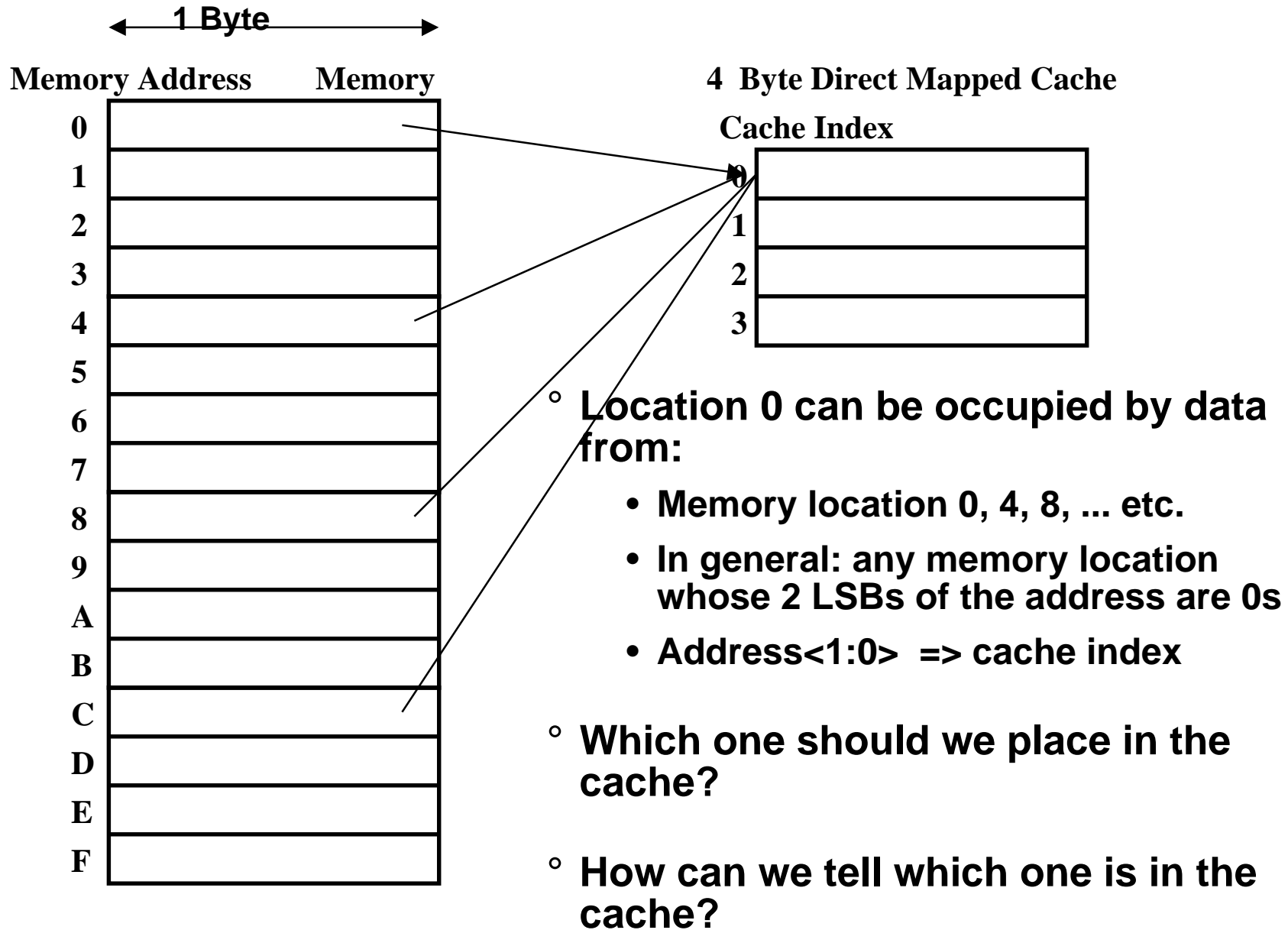
- **Direct Mapped cache is an array of fixed size blocks. Each block holds consecutive bytes of main memory data.**
- **The Tag Array holds the Block Memory Address.**
- **A valid bit associated with each cache block tells if the data is valid. (The data is not valid if it hasn't been loaded from memory.)**
 - **Cache Index: The location of a block (and it's tag) in the cache.**
 - **Byte Select: The byte location within the cache block.**

Cache-Index = (<Address> Mod (Cache_Size)) / Block_Size

Byte-Select = <Address> Mod (Block_Size)

Tag = <Address> / (Cache_Size)

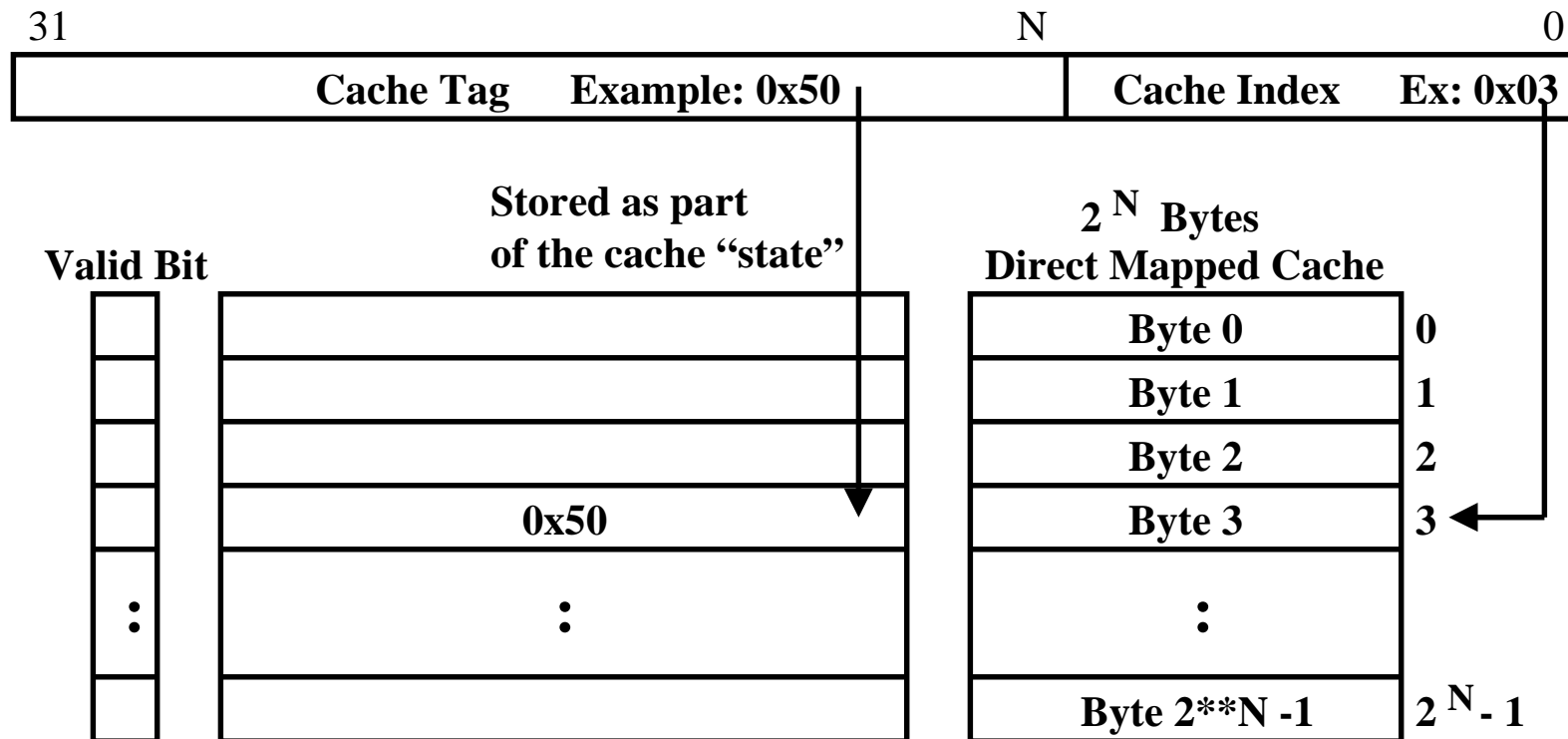
The Simplest Cache: Direct Mapped Cache



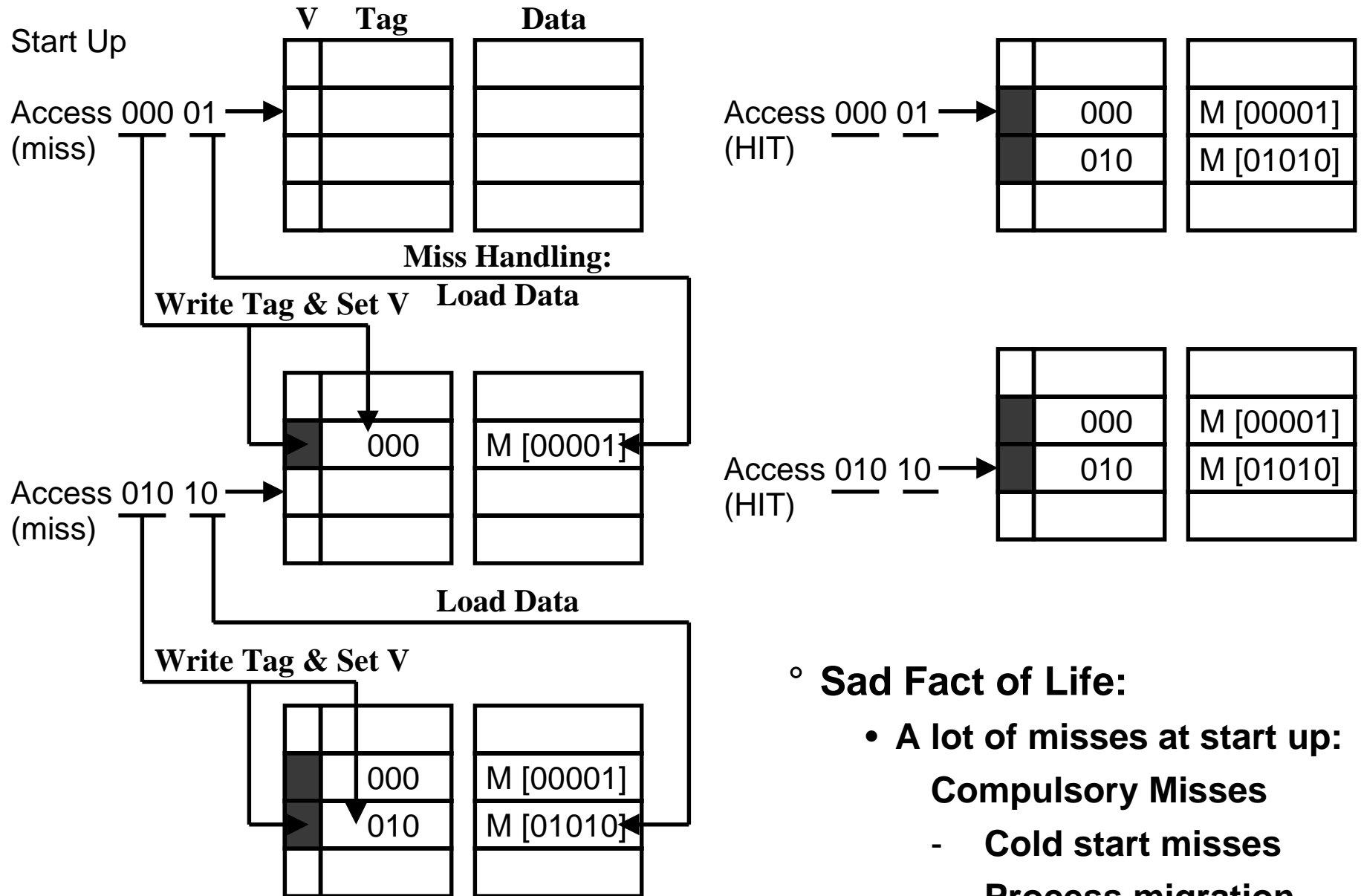
Cache Tag and Cache Index

- Assume a 32-bit memory (byte) address:
 - A 2^N byte direct mapped cache with 1-Byte blocks:
 - Cache Index: The lower N bits of the memory address
 - Cache Tag: The upper $(32 - N)$ bits of the memory address

Example address: 0x5003. Cache holds $2^8 = 256$ bytes



Cache Access Example



◦ **Sad Fact of Life:**

- **A lot of misses at start up:**

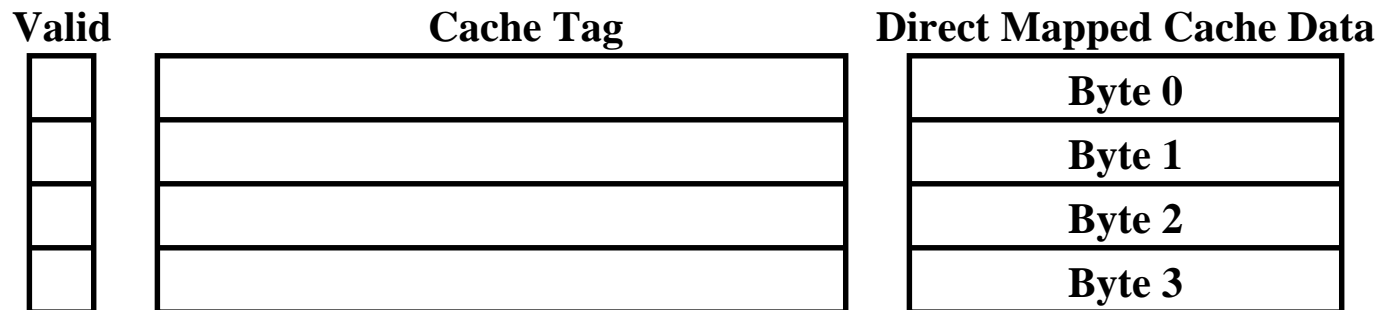
Compulsory Misses

- **Cold start misses**

- **Process migration**

Definition of a Cache Block

- **Cache Block:** the cache data that has its own cache tag
- **Our previous “extreme” example:**
 - 4-byte Direct Mapped cache: Block Size = 1 Byte
 - Take advantage of Temporal Locality: If a byte is referenced, it will tend to be referenced soon.
 - Did not take advantage of Spatial Locality: If a byte is referenced, its adjacent bytes will be referenced soon.
- **In order to take advantage of Spatial Locality: increase the block size**



Block Size Tradeoff

◦ In general, larger block sizes take advantage of spatial locality **BUT:**

- Larger block size means larger miss penalty:
 - Takes longer time to fill up the block
- If block size is too big relative to cache size, miss rate will go up
 - Too few cache blocks

◦ In general, Average Access Time:

- = Hit Time x (1 - Miss Rate) + Miss Penalty x Miss Rate

