

CS211

Computer Architecture

The Memory Hierarchy

- Topics
 - Storage technologies and trends
 - Locality of reference
 - Caching in the memory hierarchy

Memory until now...

- We've relied on a very simple model of memory for most this class
 - Main Memory is a linear array of bytes that can be accessed given a memory address
 - Also used registers to store values
- Reality is more complex. There is an entire **memory system**.
 - Different memories exist at different levels of the computer
 - Each vary in their speed, size, and cost

Random-Access Memory (RAM)

- Key features
 - **RAM** is packaged as a chip.
 - Basic storage unit is a **cell** (one bit per cell).
 - Multiple RAM chips form a memory.
- Static RAM (**SRAM**)
 - Each cell stores bit with a six-transistor circuit.
 - Retains value indefinitely, as long as it is kept powered.
 - Relatively insensitive to disturbances such as electrical noise.
 - Faster and more expensive than DRAM.
- Dynamic RAM (**DRAM**)
 - Each cell stores bit with a capacitor and transistor.
 - Value must be refreshed every 10-100 ms.
 - Sensitive to disturbances.
 - Slower and cheaper than SRAM.

Memory speeds

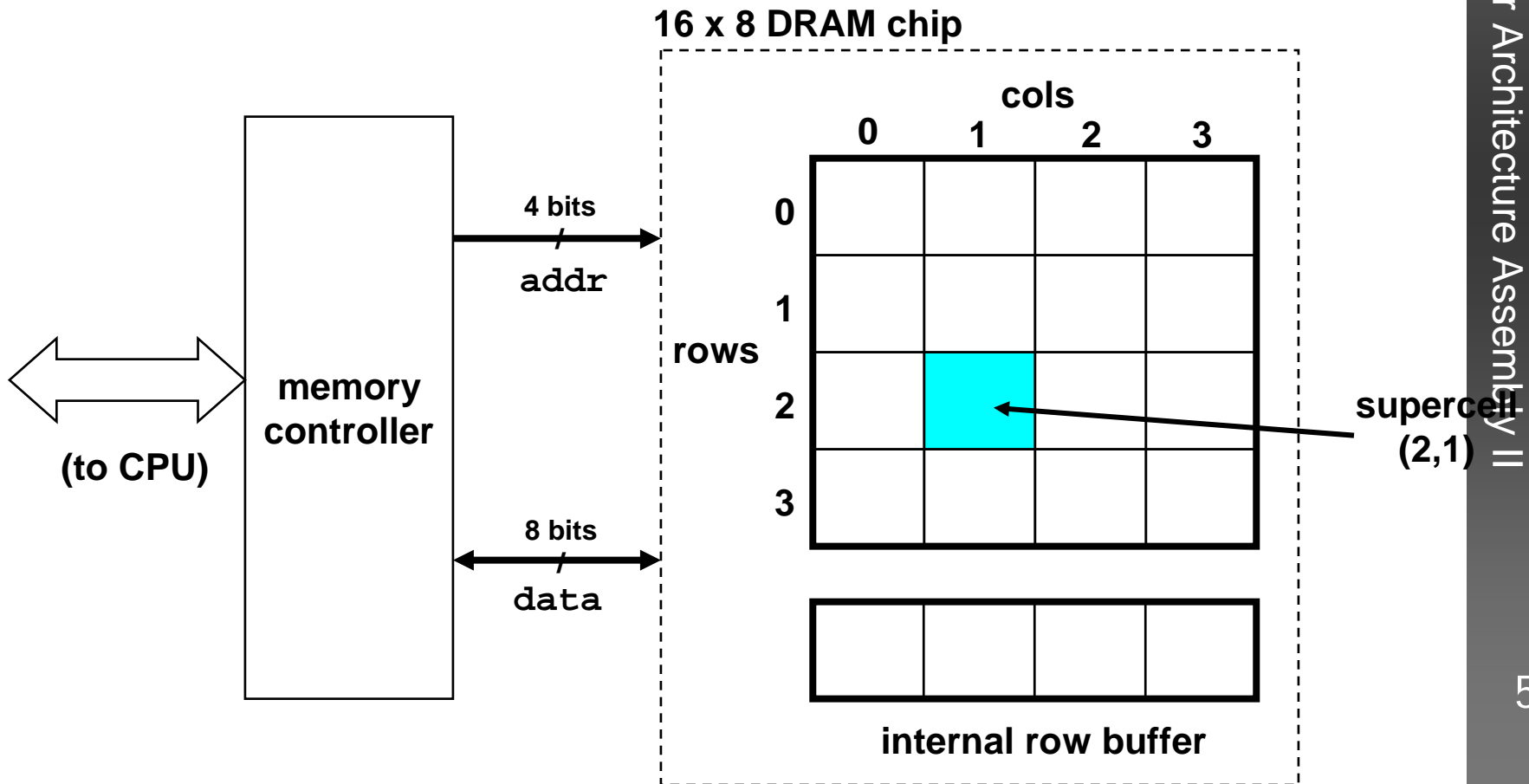
- Processor Speeds : 1 GHz processor speed is 1 nsec cycle time.
- Memory Speeds (50 nsec)

DIMM Module Chip Type	Clock Speed[MHz]	Bus Speed[MHz]	Transfer Rate [MB/s]
PC1600 DDR200	100	200	1,600
PC2100 DDR266	133	266	2,133
PC2400 DDR300	150	300	2,400

- Access Speed gap
 - Instructions that store or load from memory

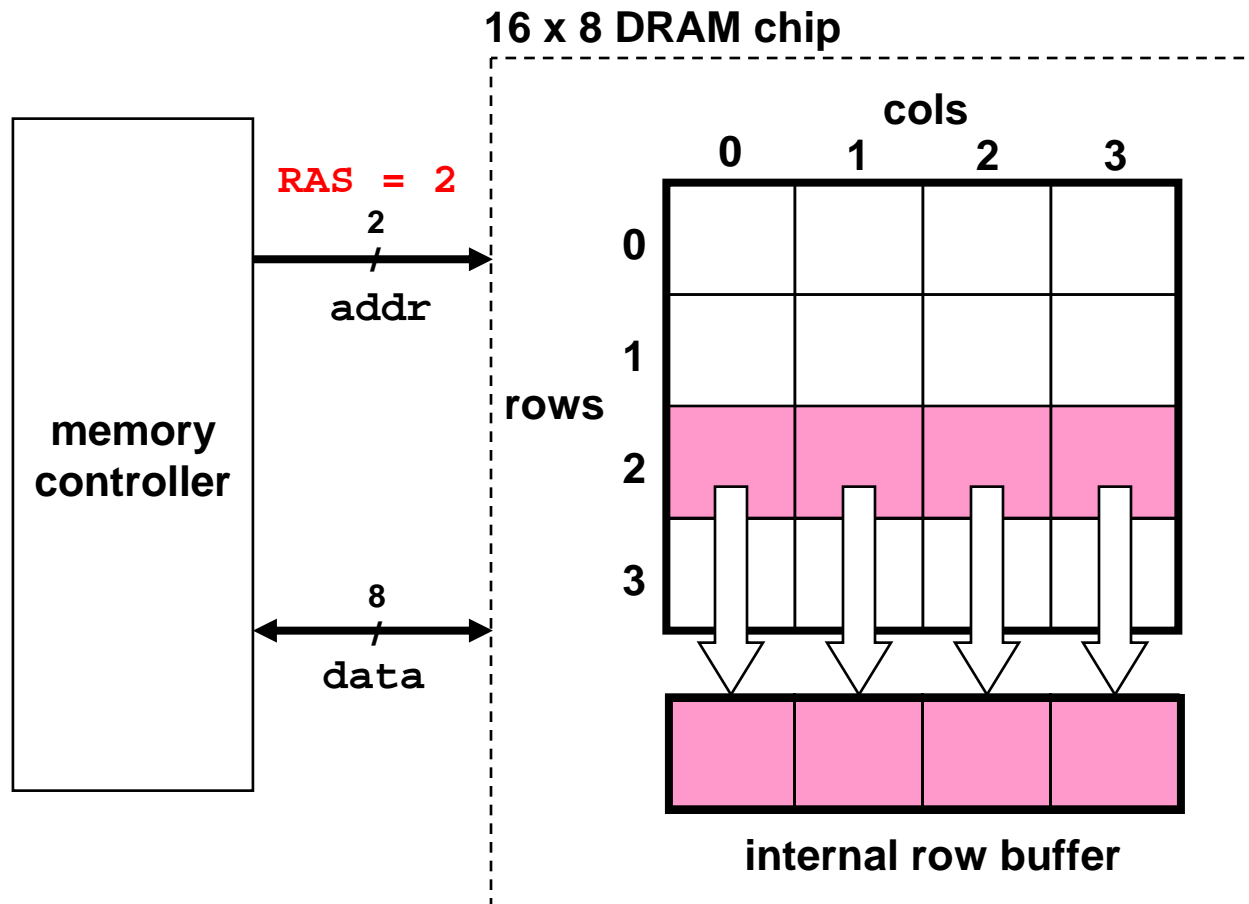
Conventional DRAM Organization

- $d \times w$ DRAM:
 - dw total bits organized as d **supercells** of size w bits



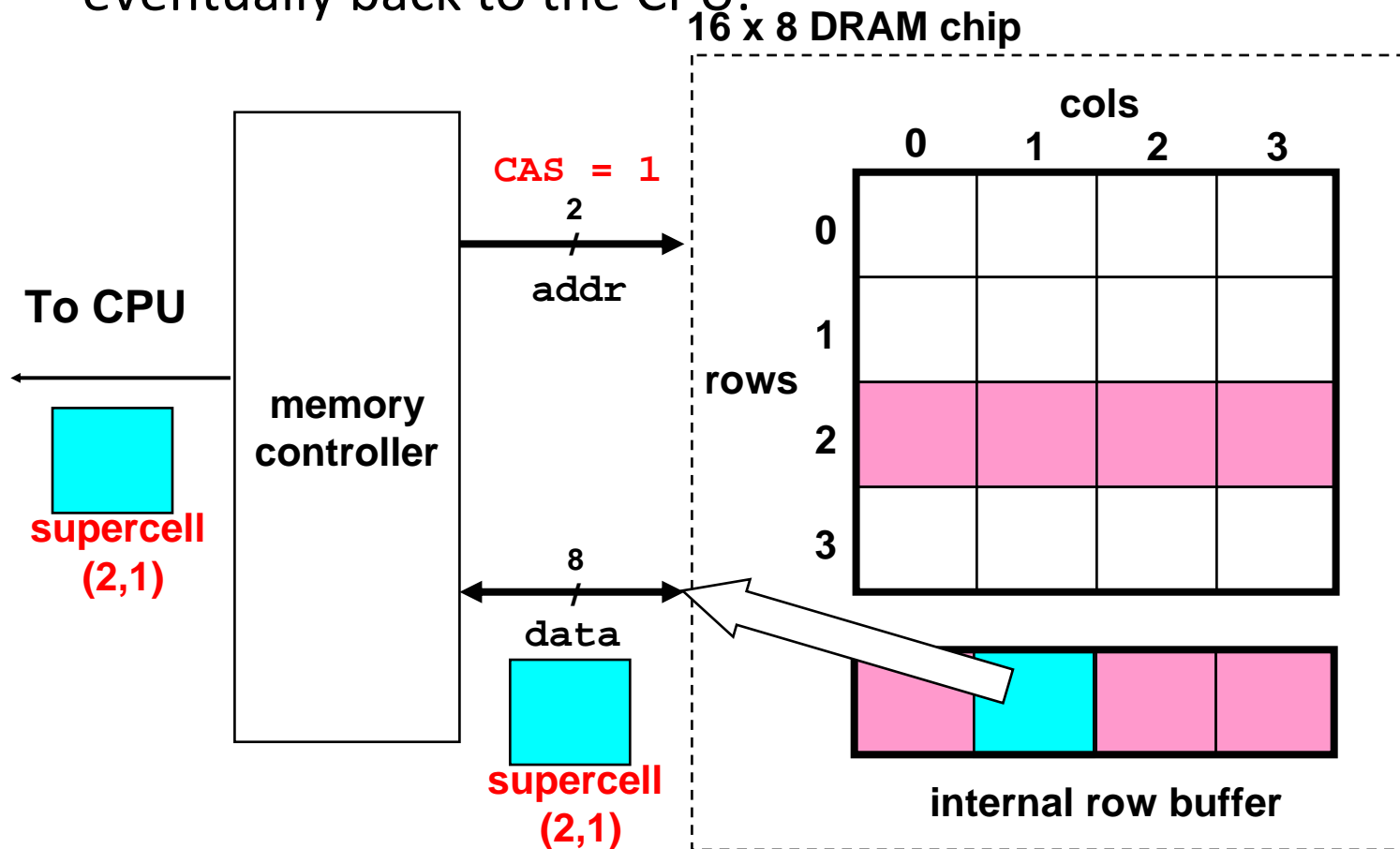
Reading DRAM Supercell (2,1)

- Step 1(a): Row access strobe (**RAS**) selects row 2.
- Step 1(b): Row 2 copied from DRAM array to row buffer.



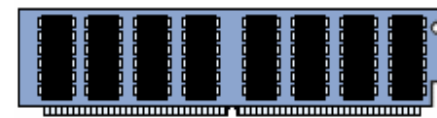
Reading DRAM Supercell (2,1)

- Step 2(a): Column access strobe (**CAS**) selects column 1.
- Step 2(b): Supercell (2,1) copied from buffer to data lines, and eventually back to the CPU.

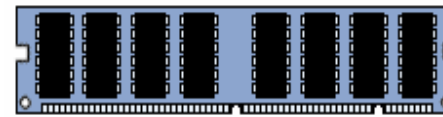


Memory Modules... real life

- We've only been discussing single DRAM Chips
- Several DRAM chips are bundled into **Memory Modules**
 - SIMMS - Single Inline Memory Module
 - DIMMS - Dual Inline Memory Module



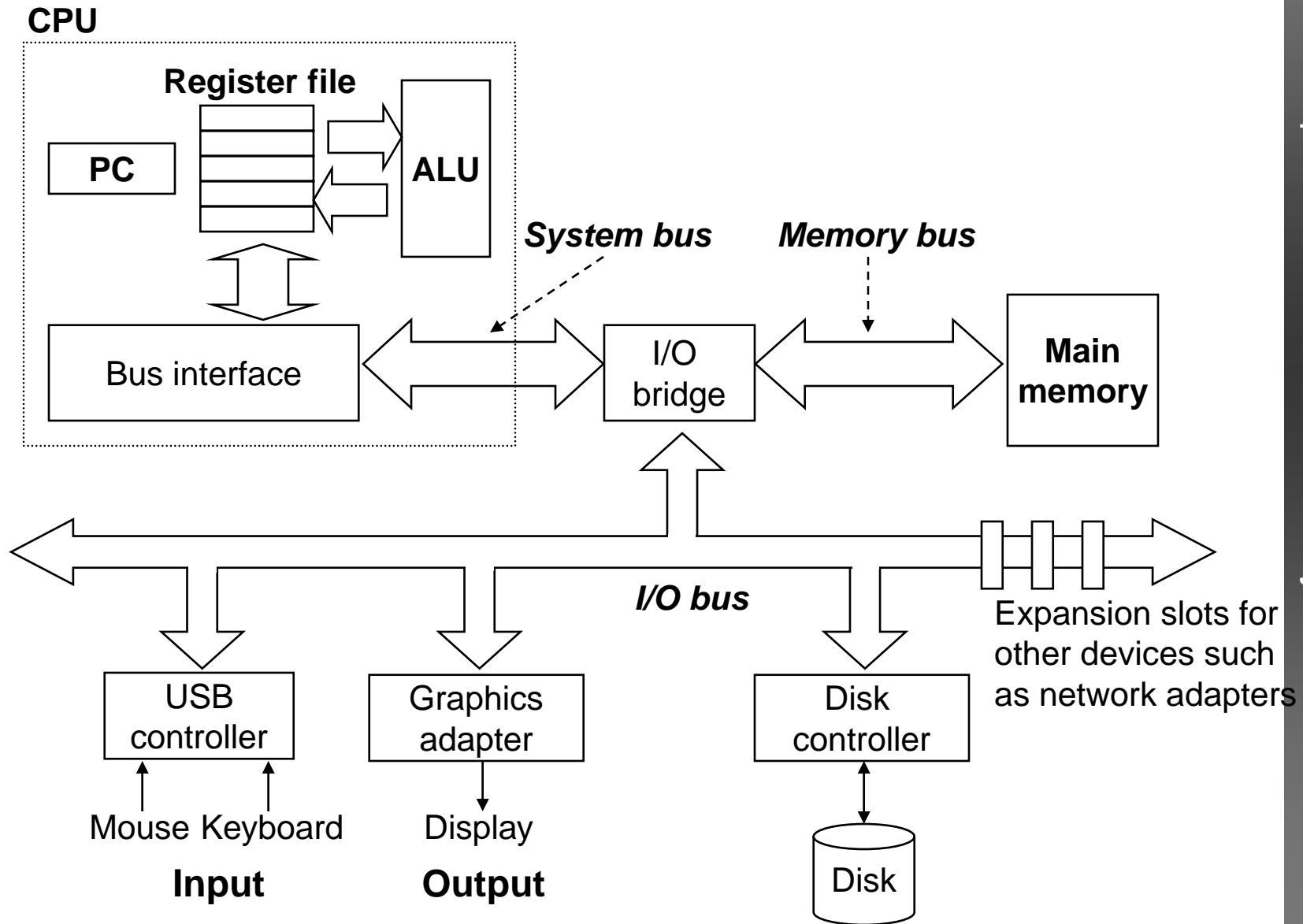
SIMM memory - 72 pin



DIMM memory

Source for Pictures: <http://en.kioskea.net/contents/pc/ram.php3>

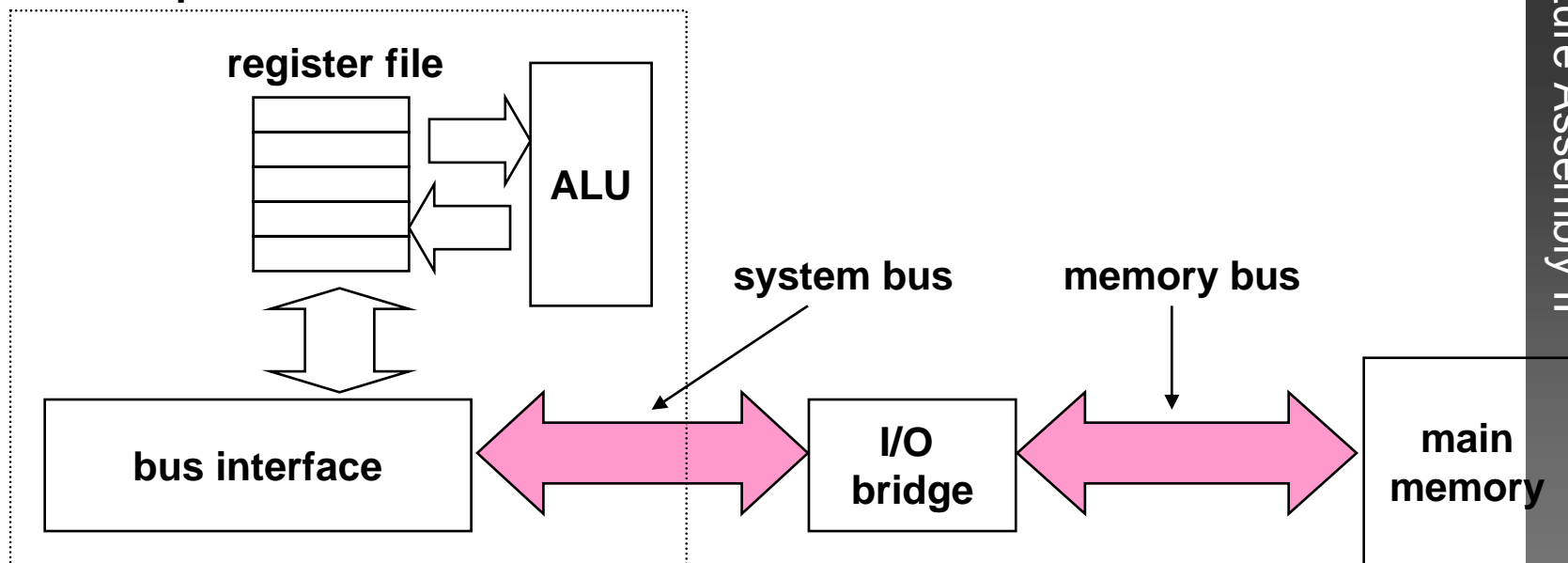
Von Neumann



Typical Bus Structure Connecting CPU and Memory

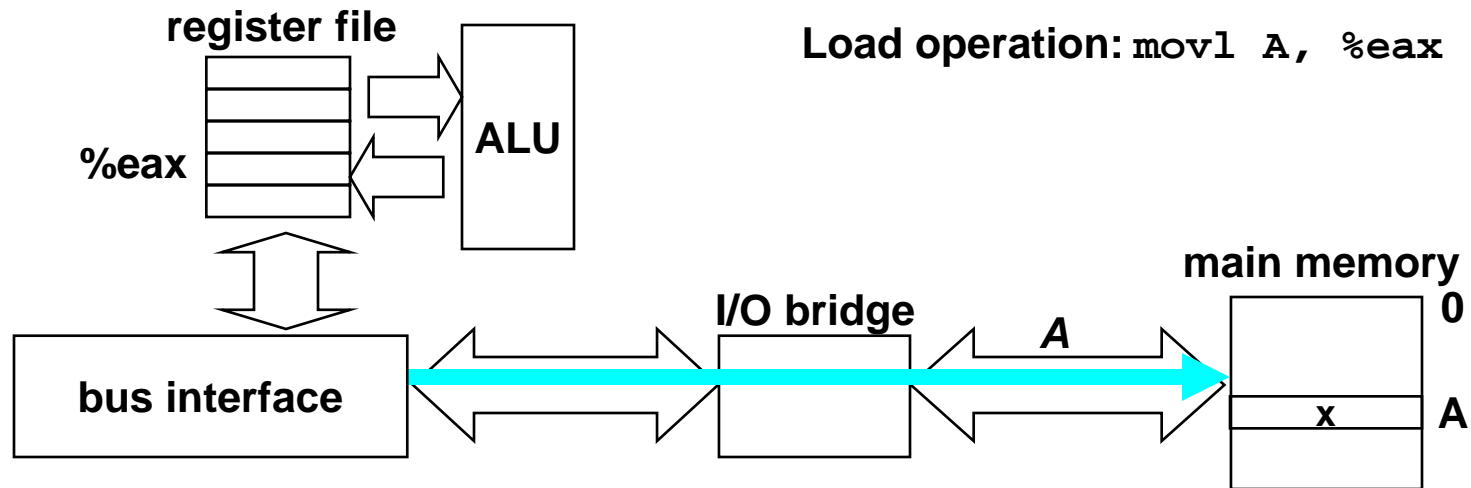
- A **bus** is a collection of parallel wires that carry address, data, and control signals.
- Buses are typically shared by multiple devices.
- Information passed through **transactions**.

CPU chip



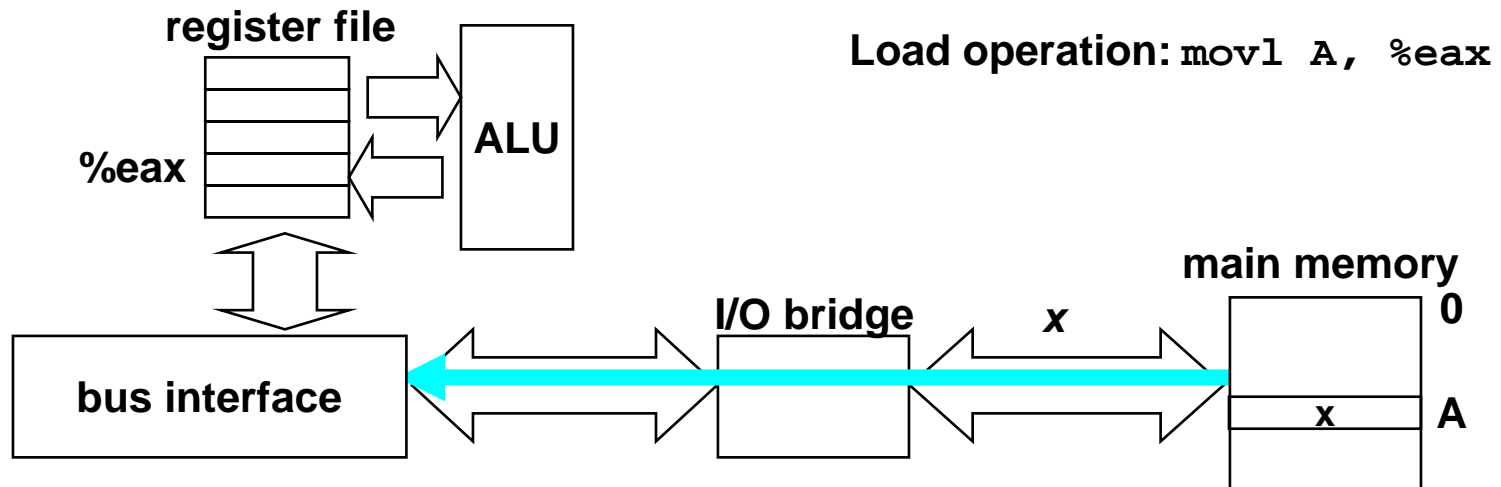
Memory Read Transaction (1)

- CPU places address A on the memory bus.



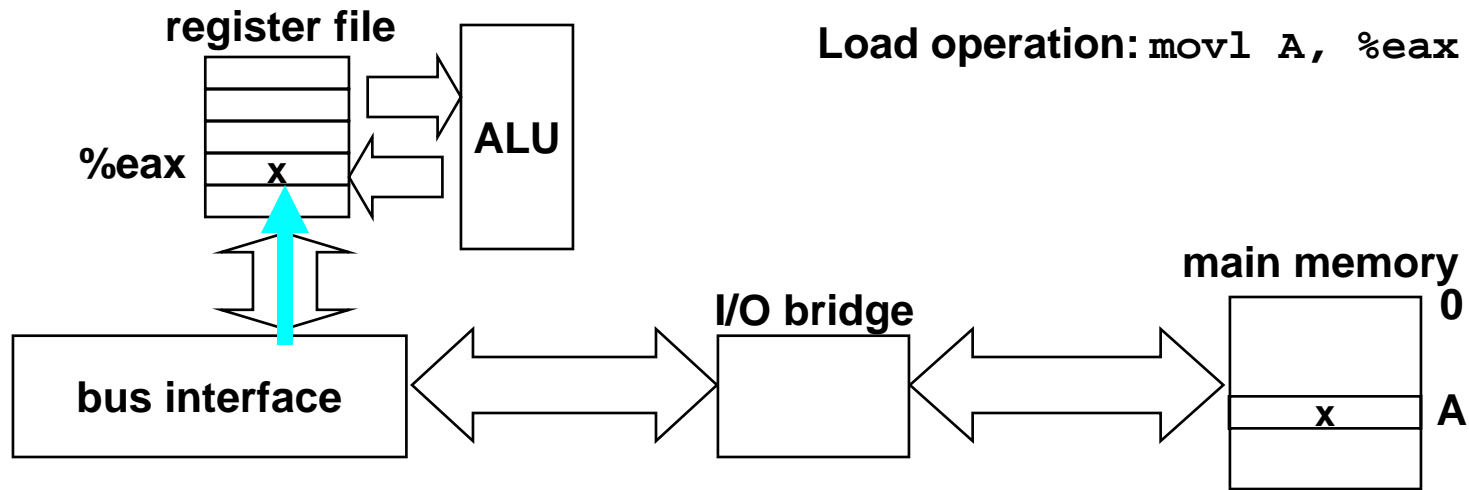
Memory Read Transaction (2)

- Main memory reads A from the memory bus, retrieves word x, and places it on the bus.



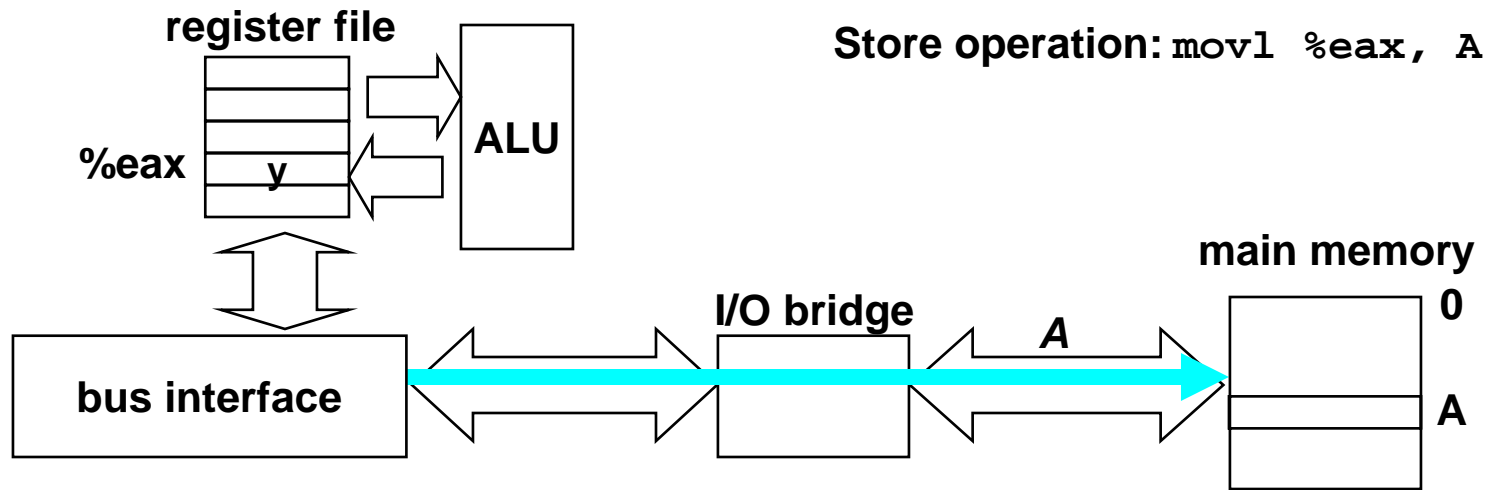
Memory Read Transaction (3)

- CPU read word x from the bus and copies it into register `%eax`.



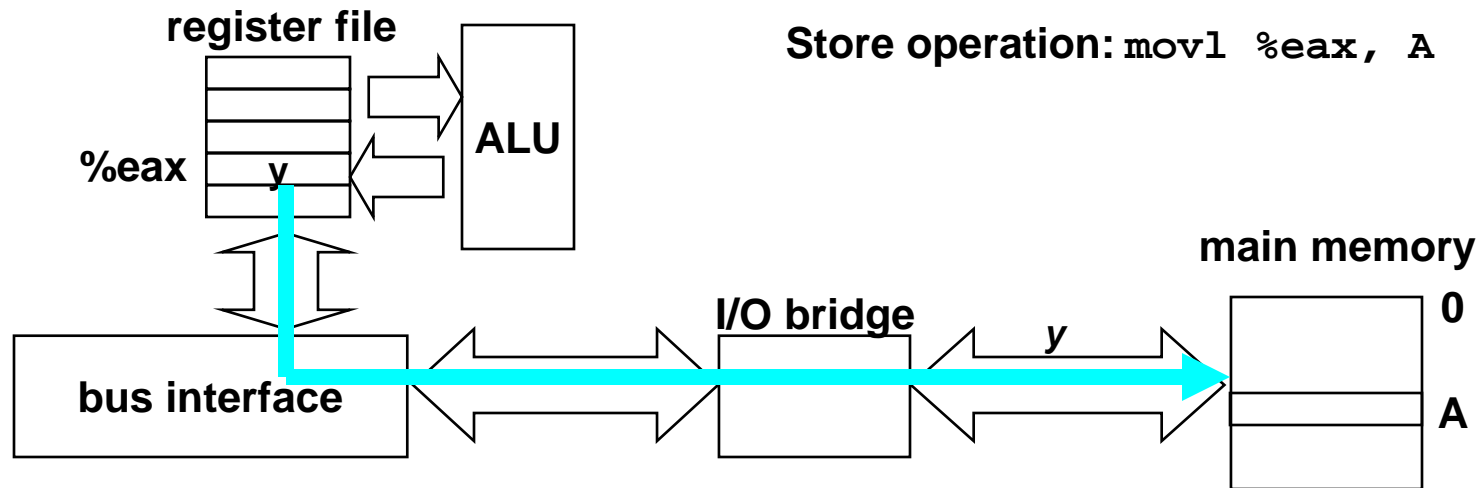
Memory Write Transaction (1)

- CPU places address A on bus. Main memory reads it and waits for the corresponding data word to arrive.



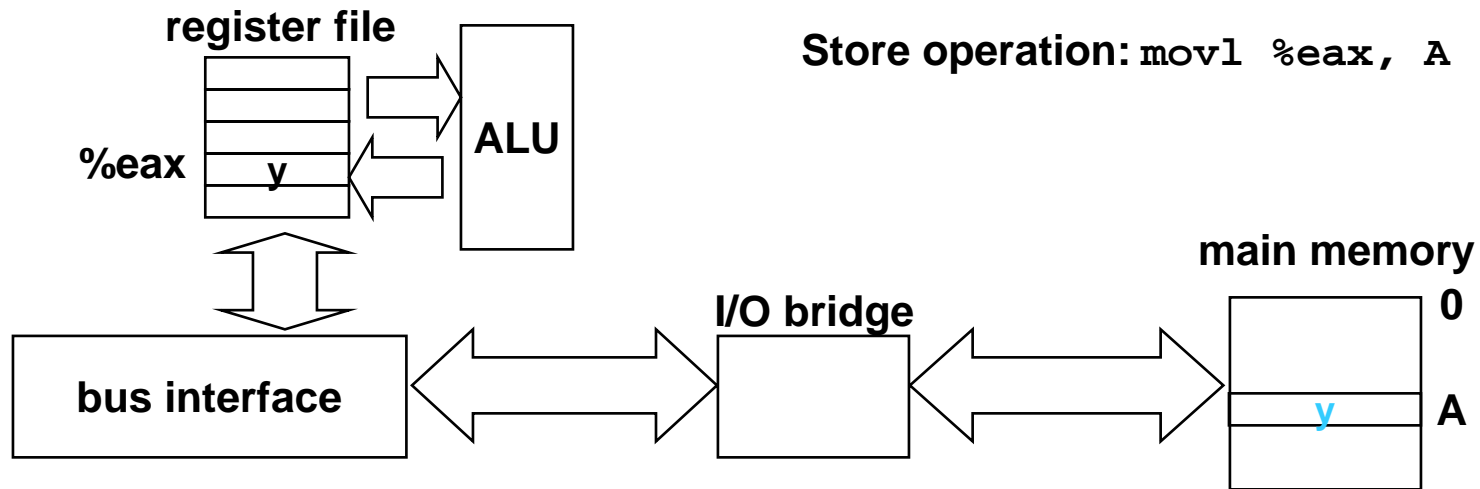
Memory Write Transaction (2)

- CPU places data word y on the bus.

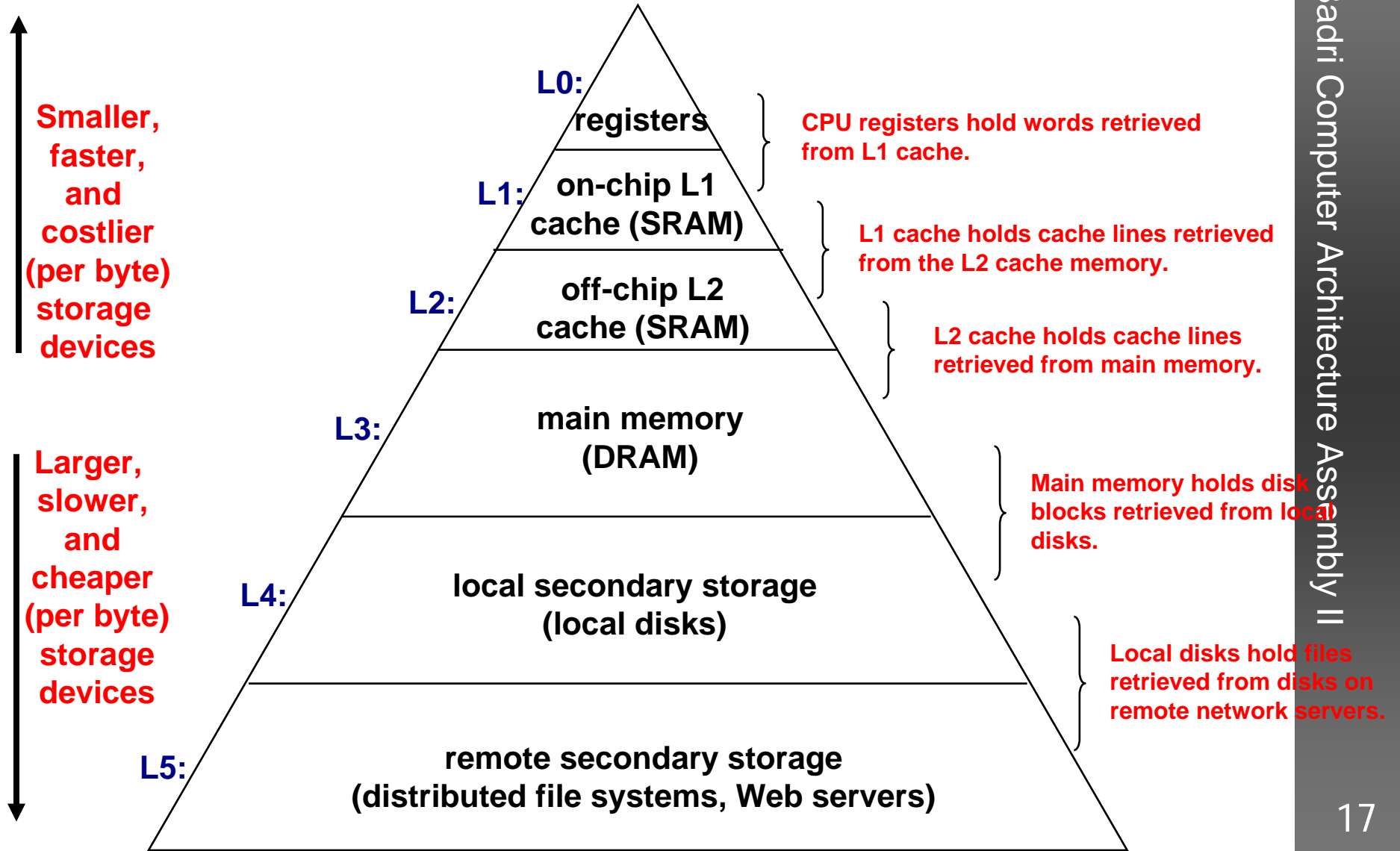


Memory Write Transaction (3)

- Main memory read data word y from the bus and stores it at address A .

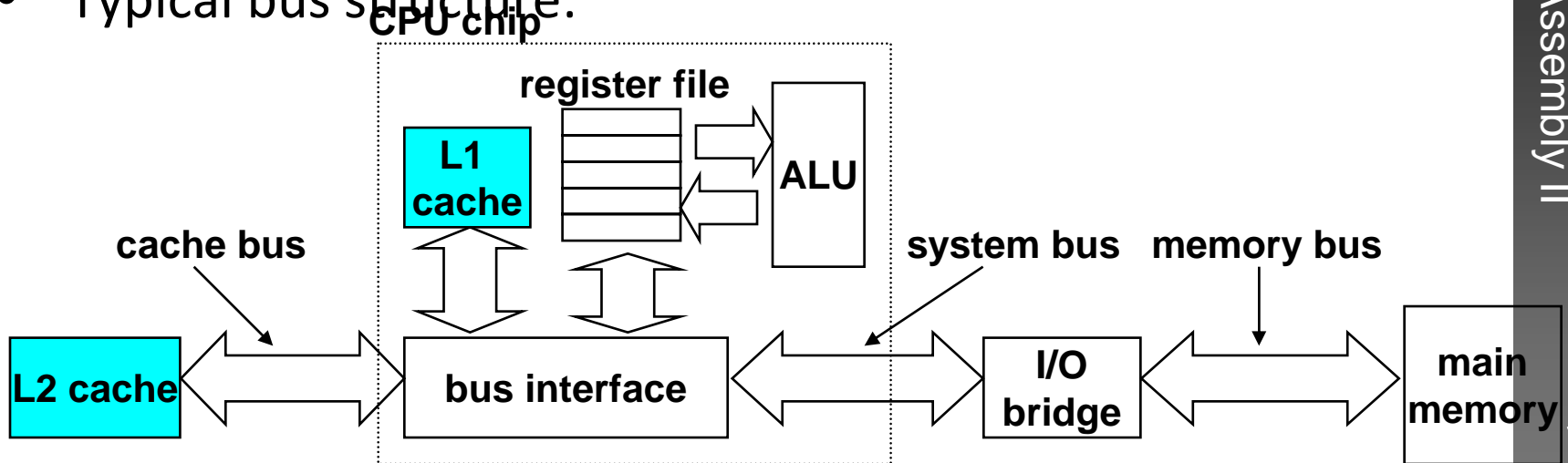


Memory Hierarchy (Review)



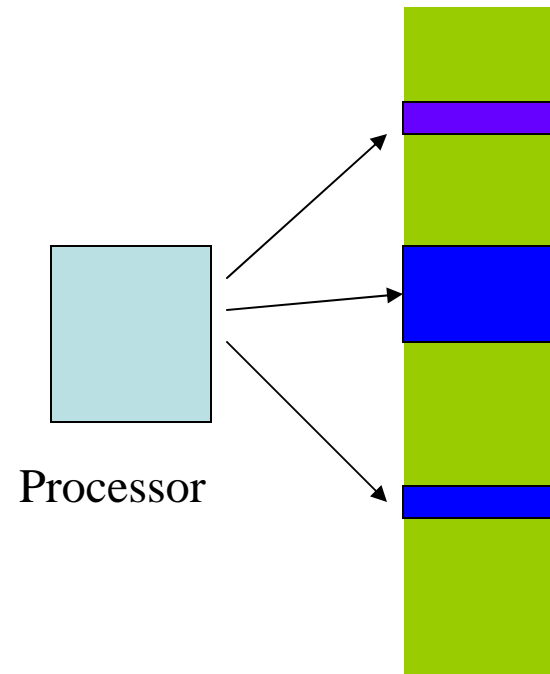
Cache Memories

- Cache memories are small, fast SRAM-based memories managed automatically in hardware.
 - Hold frequently accessed blocks of main memory
- CPU looks first for data in L1, then in L2, then in main memory.
- Typical bus structure:



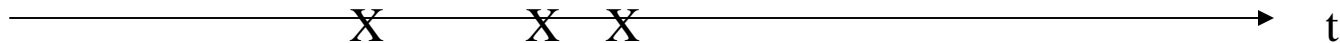
How to Exploit memory hierarchy

- Availability of memory
 - Cost, size, speed
- Principle of locality
- Memory references are bunched together
 - A small portion of address space is accessed at any given time
- This space in high speed memory
- Problem: not all of it may fit



Types of locality

- Temporal locality
 - Tendency to access locations recently referenced



- Spatial locality
 - Tendency to reference locations around recently referenced
 - Location x , then others will be $x-k$ or $x+k$



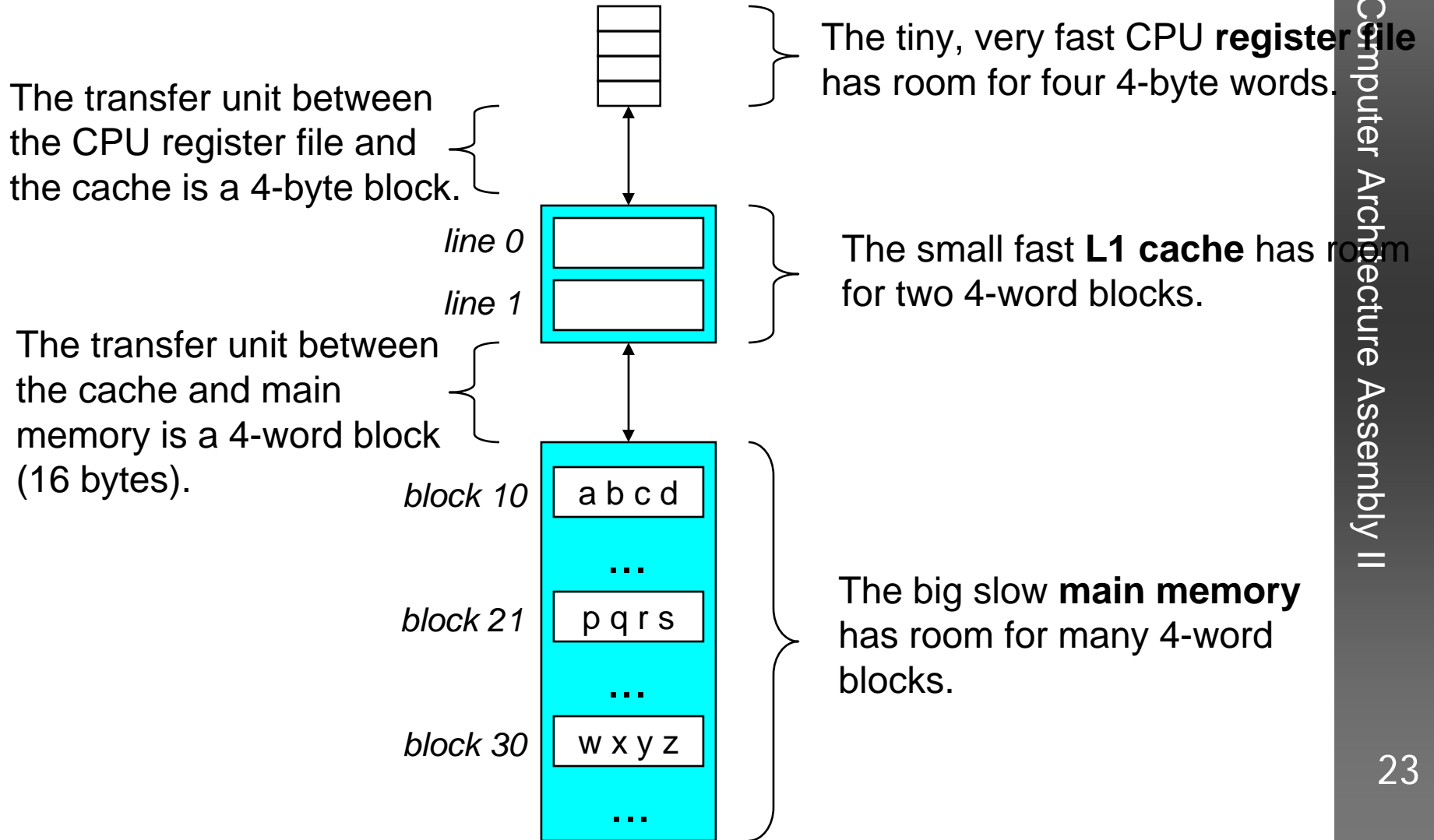
Sources of locality

- Temporal locality
 - Code within a loop
 - Same instructions fetched repeatedly
- Spatial locality
 - Data arrays
 - Local variables in stack
 - Data allocated in chunks (contiguous bytes)

What does locality buy?

- Address the gap between CPU speed and RAM speed
- Spatial and temporal locality implies a subset of instructions can fit in high speed memory from time to time
- CPU can access instructions and data from this high speed memory
- Small high speed memory can make computer faster and cheaper
- Speed of 1-20 nsec at cost of \$50 to \$100 per Mbyte
- This is Caching!!

Inserting an L1 Cache Between the CPU and Main Memory

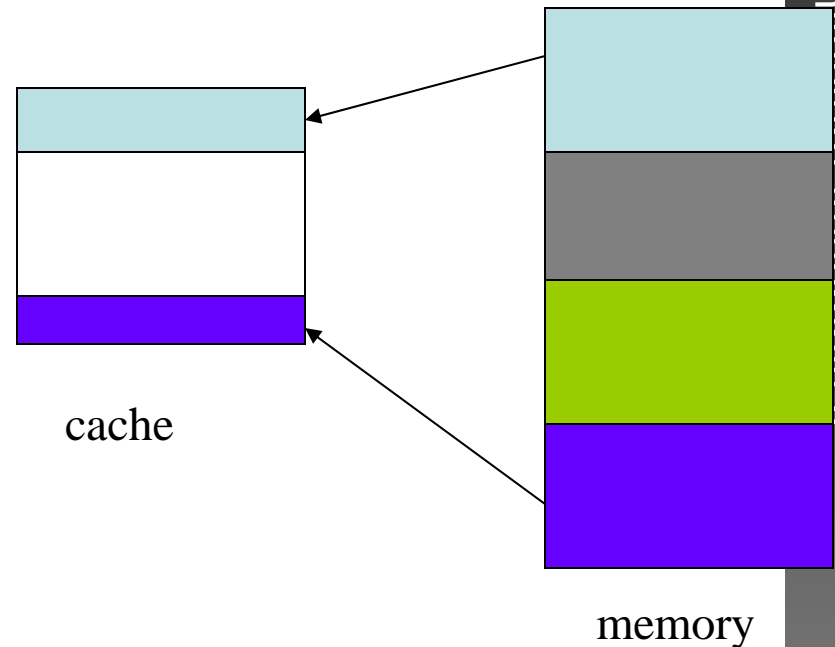


What info. does a cache need

- **Cache**: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.
- You're essentially allowing a **smaller** region of memory to hold data from a **larger** region. Not a 1-1 mapping.
- What kind of information do we need to keep:
 - The actual data
 - Where the data actually comes from
 - If data is even considered valid

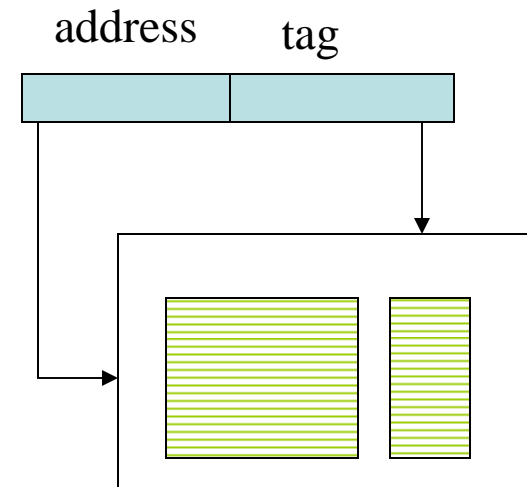
Cache Organization

- Map each region of memory to a smaller region of cache
- Discard address bits
 - Discard lower order bits (a)
 - Discard higher order bits (b)
- Cache address size is 4 bits
- Memory address size is 8 bits
- In case of a)
 - 0000xxxx is mapped to 0000 in cache
- In case of b)
 - xxxx0001 is mapped to 0001 in cache



Finding data in cache

- Part of memory address applied to cache
- Remaining is stored as tag in cache
- Lower order bits discarded
- Need to check if 00010011
 - Cache index is 0001
 - Tag is 0011
- If tag matches, hit, use data
- No match, miss, fetch data from memory



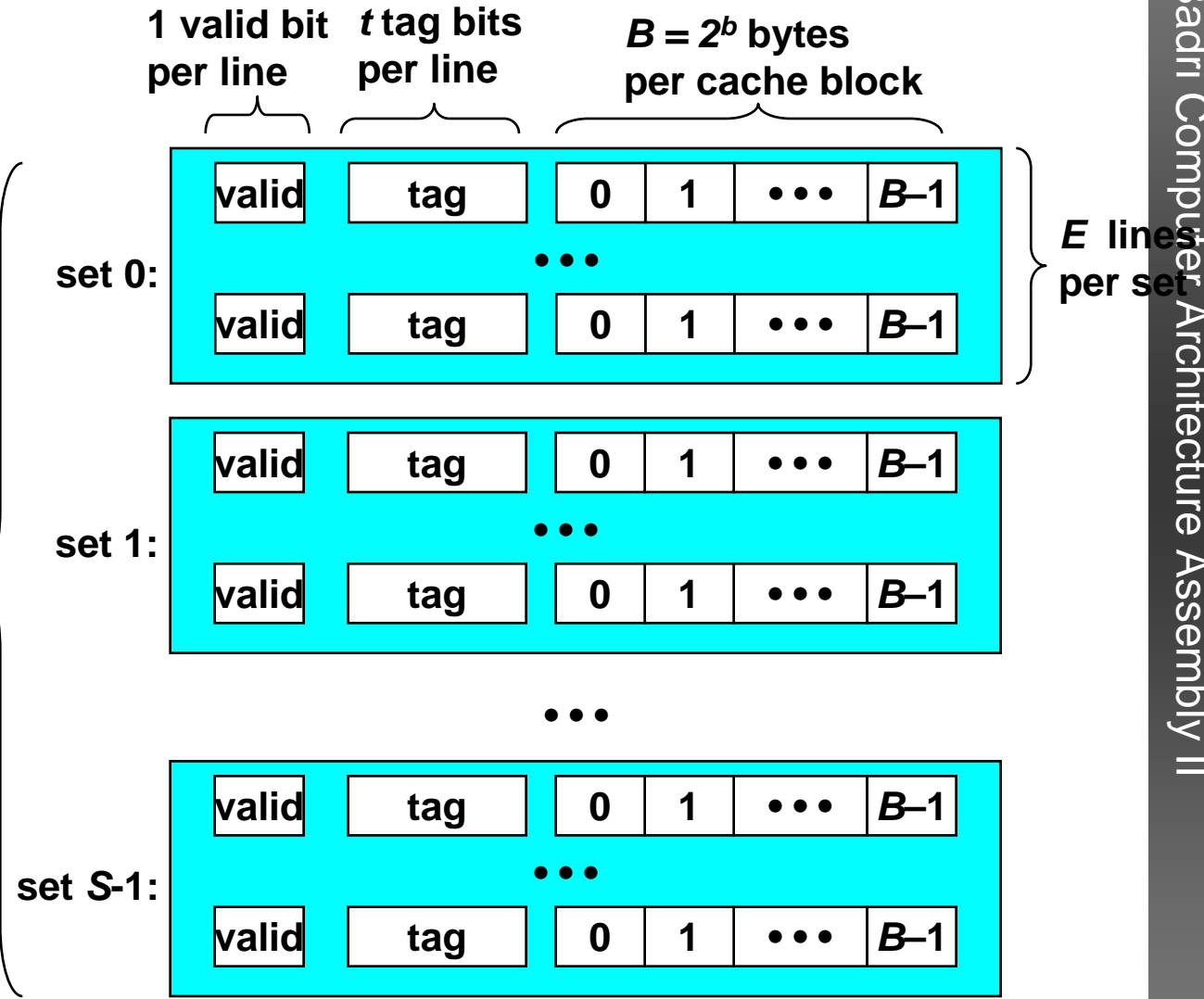
General Org of a Cache Memory

Cache is an array of sets.

Each set contains one or more lines.

Each line holds a block of data.

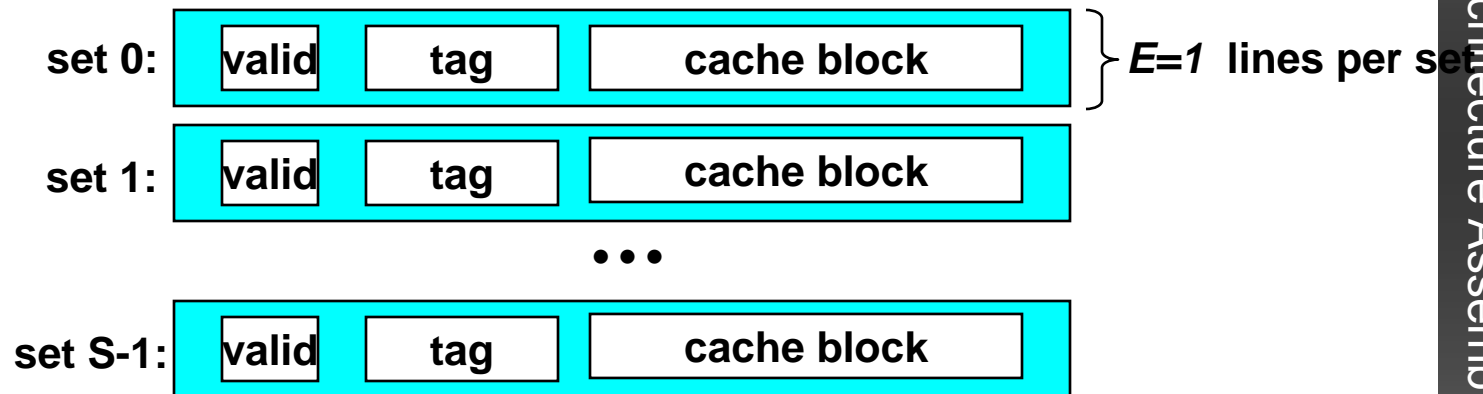
$S = 2^s$ sets



Cache size: $C = B \times E \times S$ data bytes

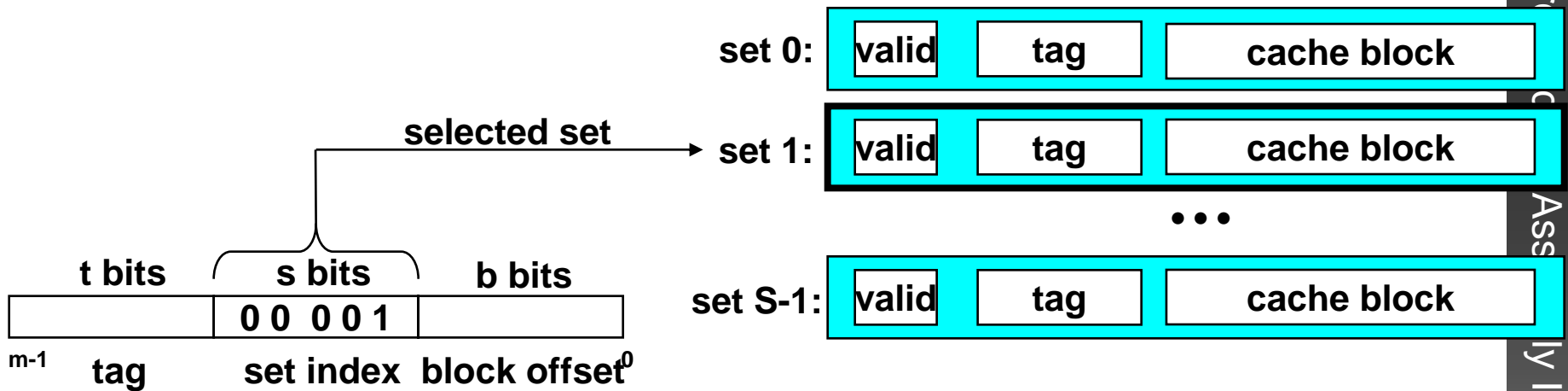
Direct-Mapped Cache

- Simplest kind of cache
- Characterized by exactly one line per set.



Accessing Direct-Mapped Caches

- Set selection
 - Use the set index bits to determine the set of interest.



Accessing Direct-Mapped Caches

- Line matching and word selection
 - Line matching:** Find a valid line in the selected set with a matching tag
 - Word selection:** Then extract the word

