CMSC 313 Lecture 26

- DigSim Assignment 3
- Cache Memory
- Virtual Memory + Cache Memory
- I/O Architecture

DigSim Assignment 3: Finite State Machine Simplifications

Due: Tuesday, December 14, 2004

Objective

The objective is to design and simplify a moderately complex a finite state machine.

Assignment

You have already completed the design and simplification steps in Homework 5. Your assignment here is to implement the finite state machine you designed in Question #5 of Homework 5. You should use the state assignment and flip-flop selection that uses the smallest total number of gates. Note that the gates used for the output z might be different for different state assignments.

What to submit

1) If the finite state machine you implemented differs significantly from the one you turned in for Homework 5, then submit the transition diagram, truth tables and formulas you used to implement this finite state machine in class on Tuesday, December 14.

2) Save your circuit as you did in DigSim Assignment 1. Submit the circuit file using the Unix submit command as in previous assignments. The submission name for this assignment is: `digsim3`. `The` UNIX command to do this should look something like:

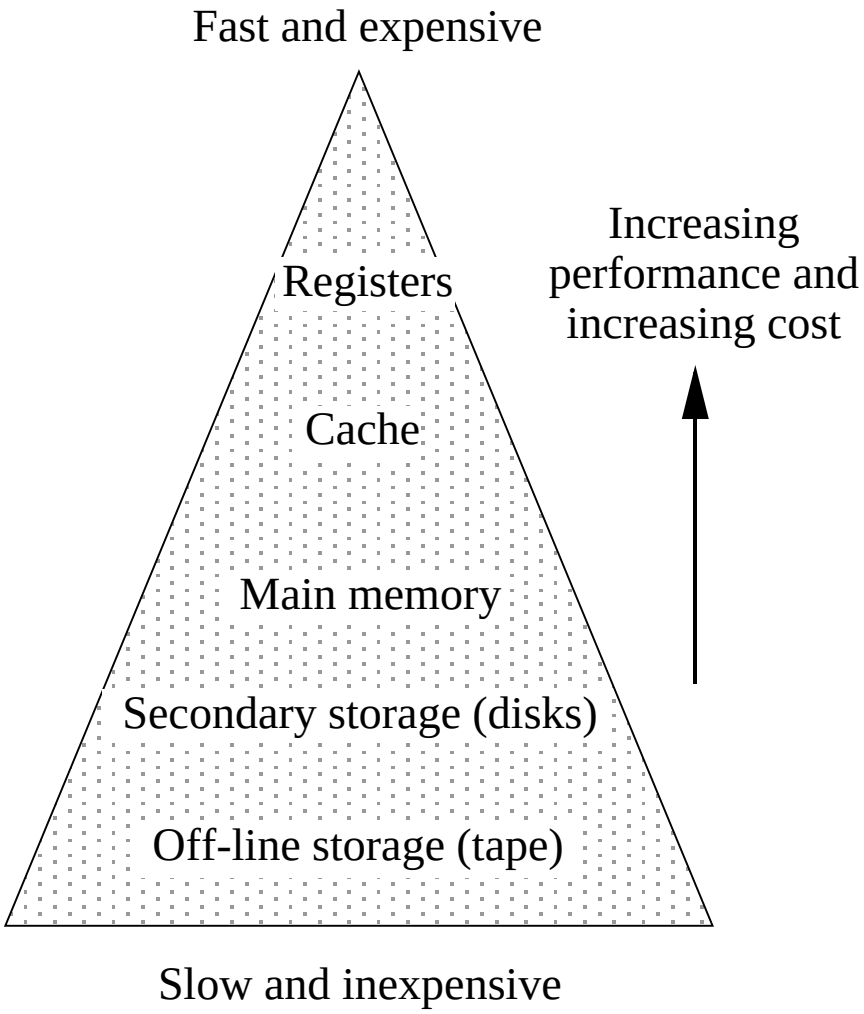
```
submit cs313_0101 digsim3 d3.sim
```

Last Time

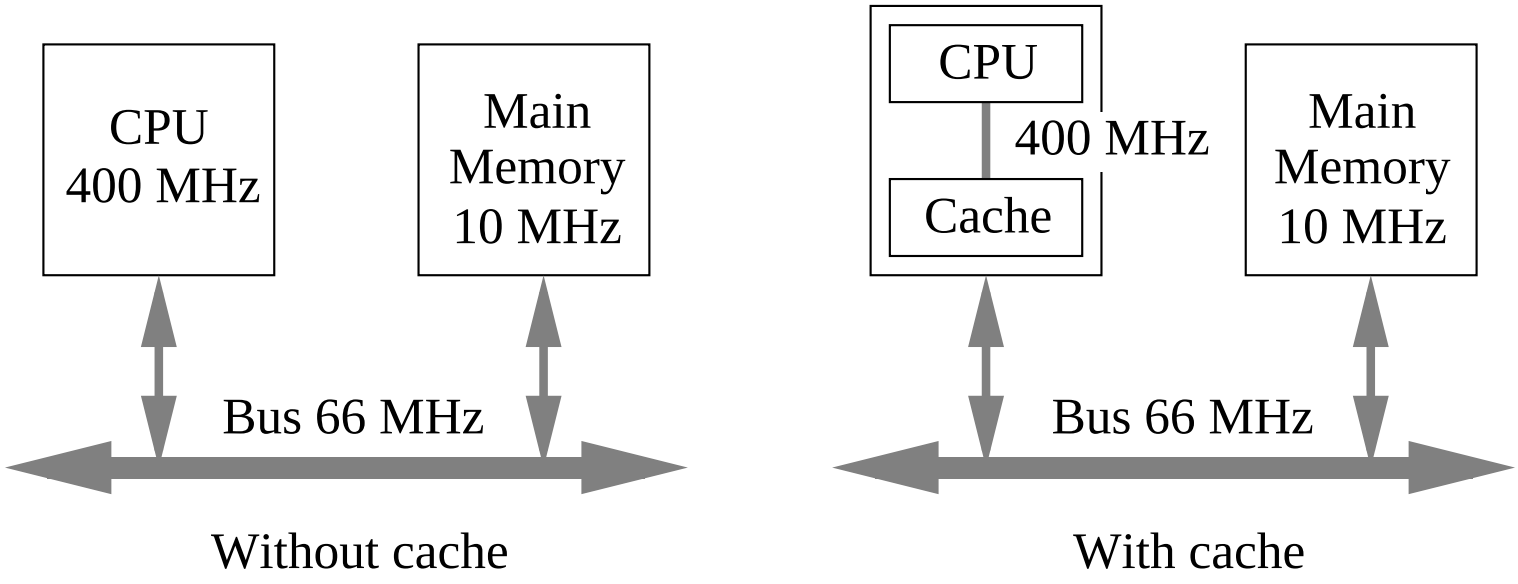
- Memory Organization
- Technological advances in DRAM
- SDRAM allows quick access to successive memory locations in the same row.

Why is this helpful??

The Memory Hierarchy

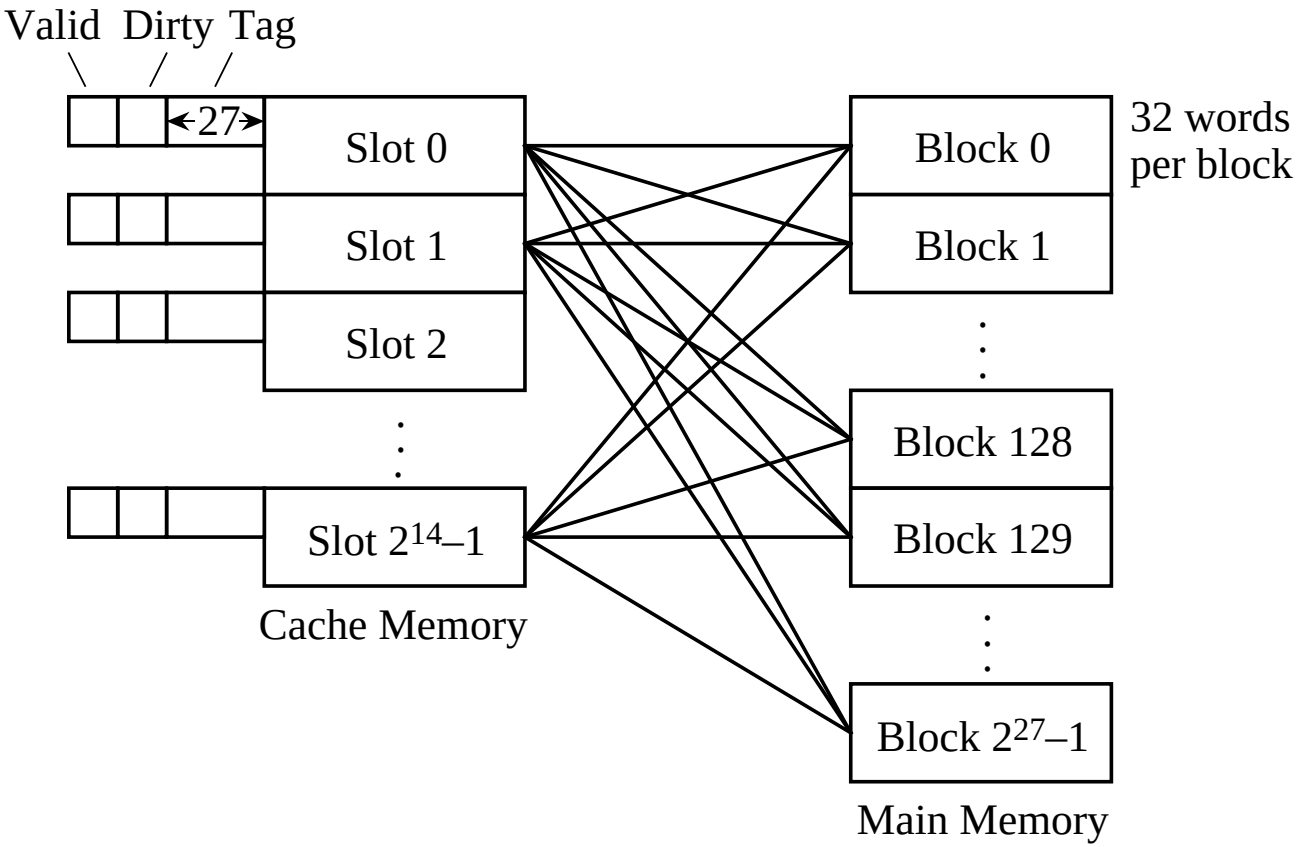


Placement of Cache in a Computer System



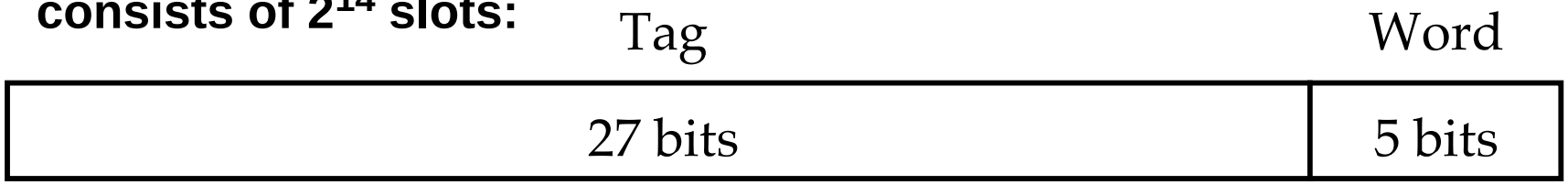
- **The *locality principle*: a recently referenced memory location is likely to be referenced again (*temporal locality*); a neighbor of a recently referenced memory location is likely to be referenced (*spatial locality*).**

An Associative Mapping Scheme for a Cache Memory

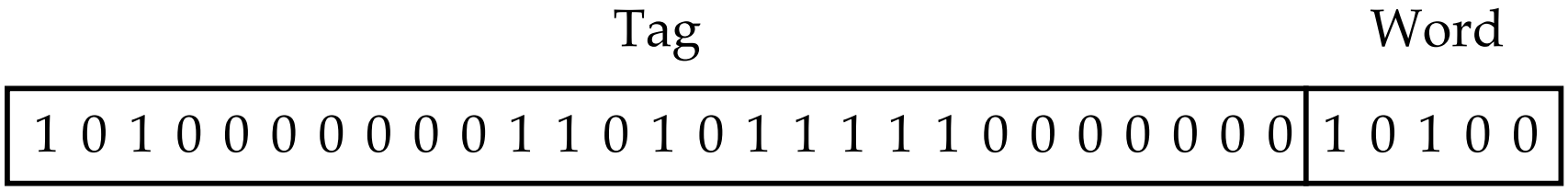


Associative Mapping Example

- Consider how an access to memory location $(A035F014)_{16}$ is mapped to the cache for a 2^{32} word memory. The memory is divided into 2^{27} blocks of $2^5 = 32$ words per block, and the cache consists of 2^{14} slots:



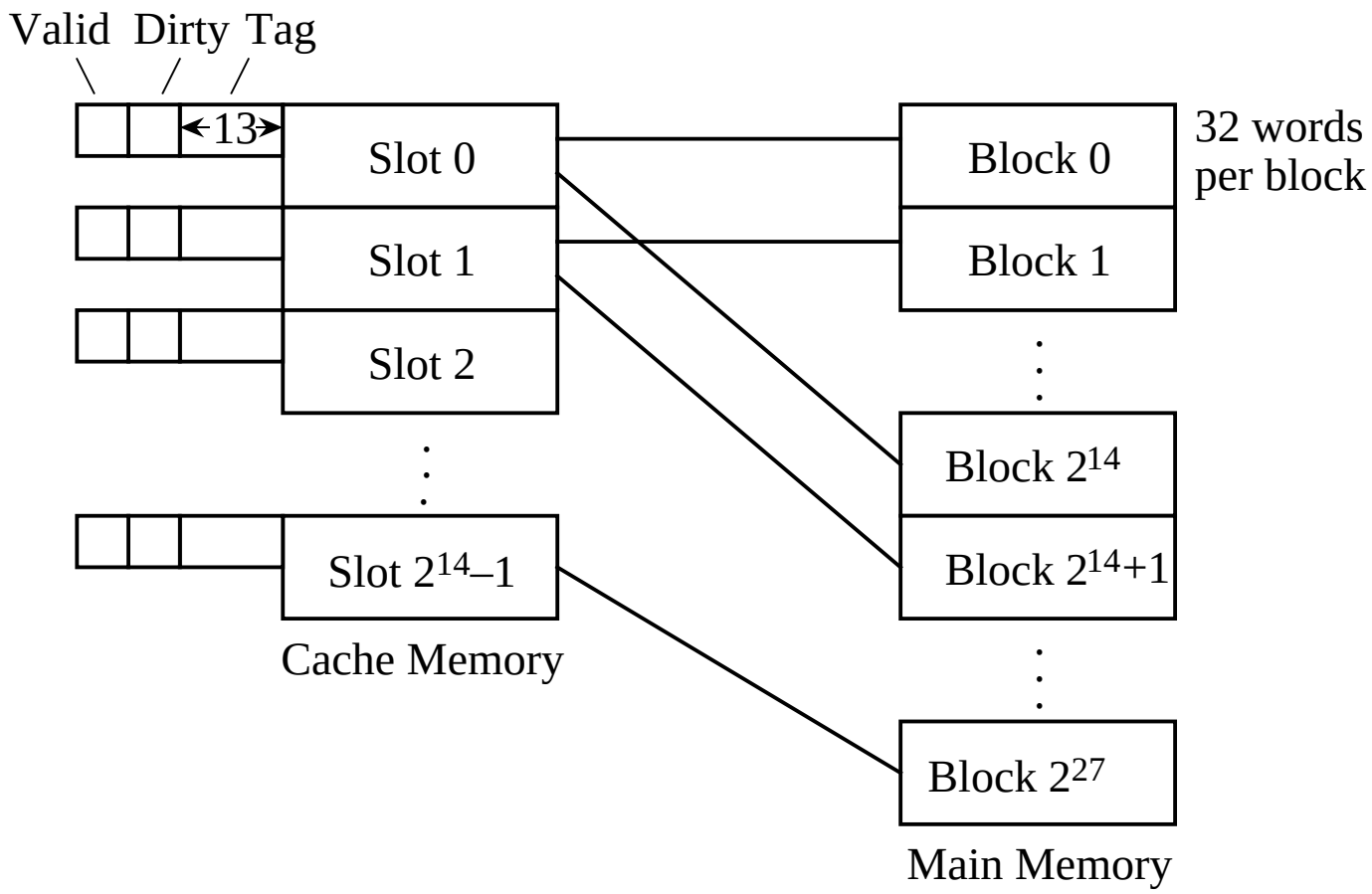
- If the addressed word is in the cache, it will be found in word $(14)_{16}$ of a slot that has tag $(501AF80)_{16}$, which is made up of the 27 most significant bits of the address. If the addressed word is not in the cache, then the block corresponding to tag field $(501AF80)_{16}$ is brought into an available slot in the cache from the main memory, and the memory reference is then satisfied from the cache.



Replacement Policies

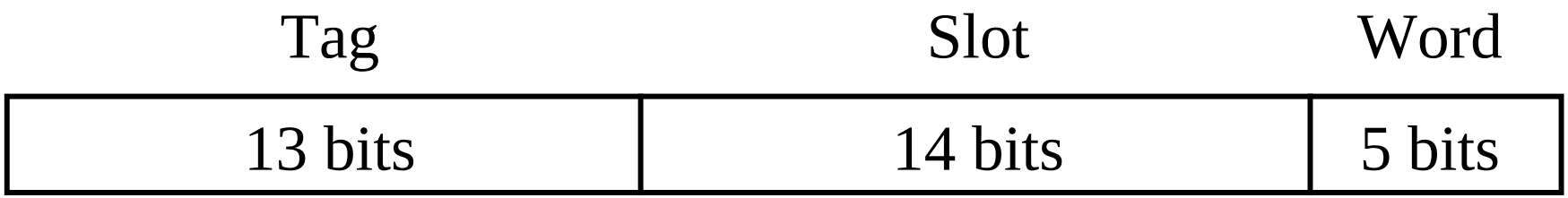
- When there are no available slots in which to place a block, a *replacement policy* is implemented. The replacement policy governs the choice of which slot is freed up for the new block.
- Replacement policies are used for associative and set-associative mapping schemes, and also for virtual memory.
- Least recently used (LRU)
- First-in/first-out (FIFO)
- Least frequently used (LFU)
- Random
- Optimal (used for analysis only – look backward in time and reverse-engineer the best possible strategy for a particular sequence of memory references.)

A Direct Mapping Scheme for Cache Memory

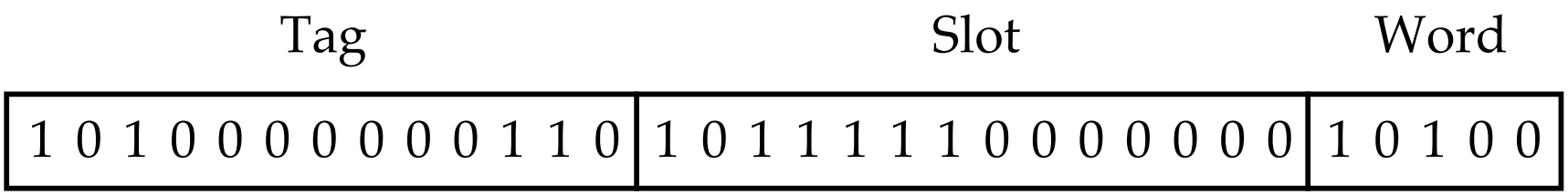


Direct Mapping Example

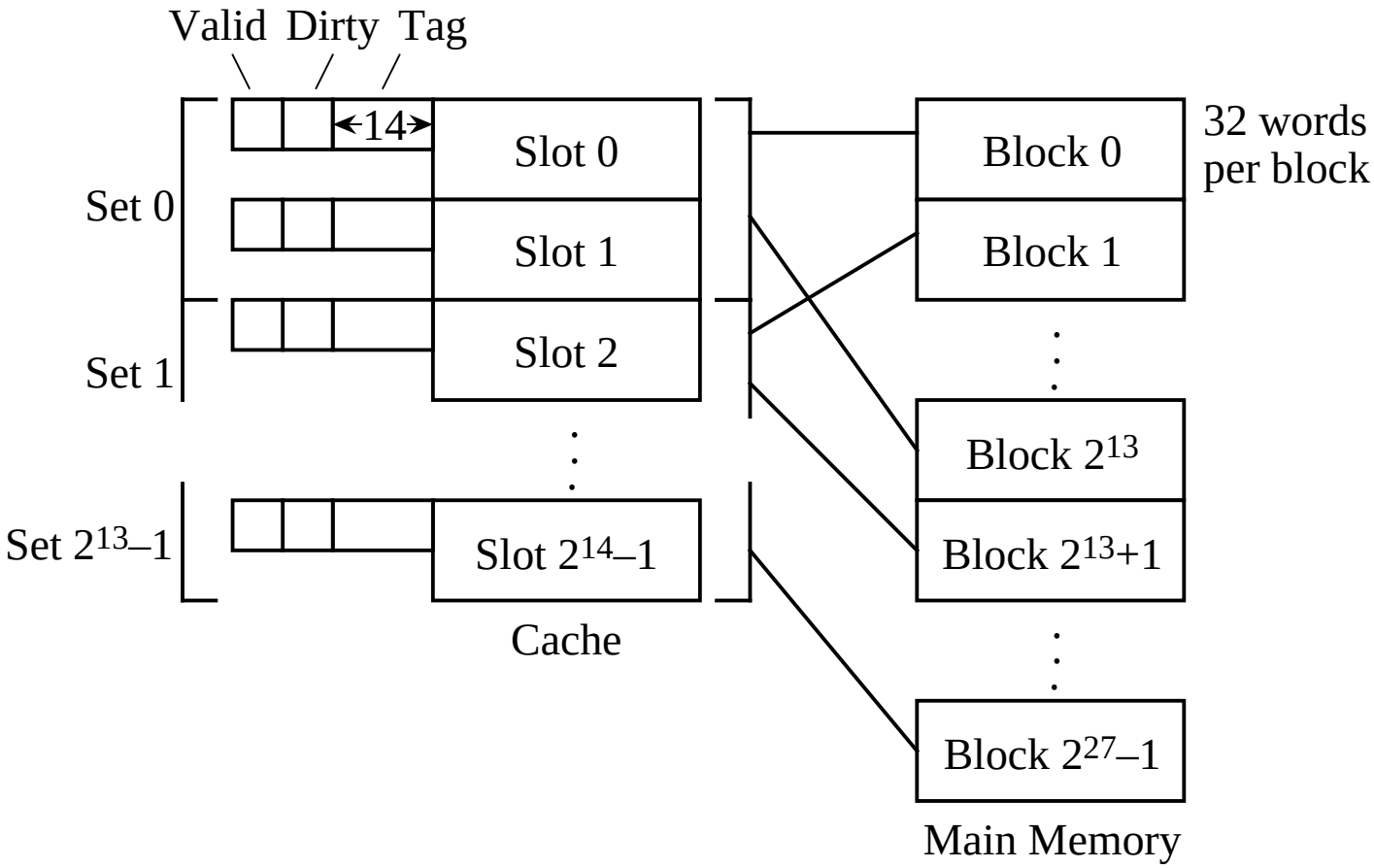
- **For a direct mapped cache, each main memory block can be mapped to only one slot, but each slot can receive more than one block. Consider how an access to memory location $(A035F014)_{16}$ is mapped to the cache for a 2^{32} word memory. The memory is divided into 2^{27} blocks of $2^5 = 32$ words per block, and the cache consists of 2^{14} slots:**



- **If the addressed word is in the cache, it will be found in word $(14)_{16}$ of slot $(2F80)_{16}$, which will have a tag of $(1406)_{16}$.**

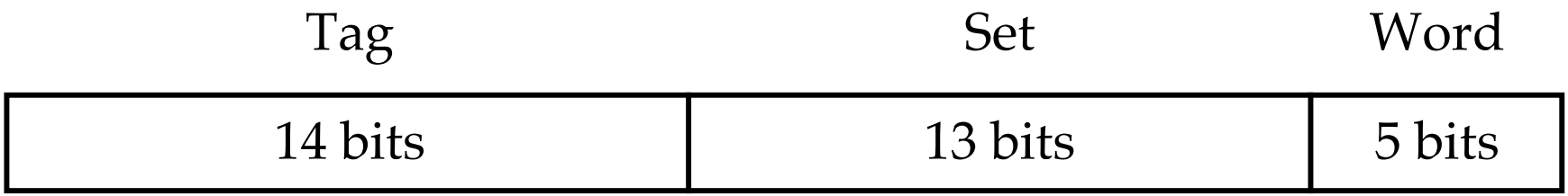


A Set Associative Mapping Scheme for a Cache Memory

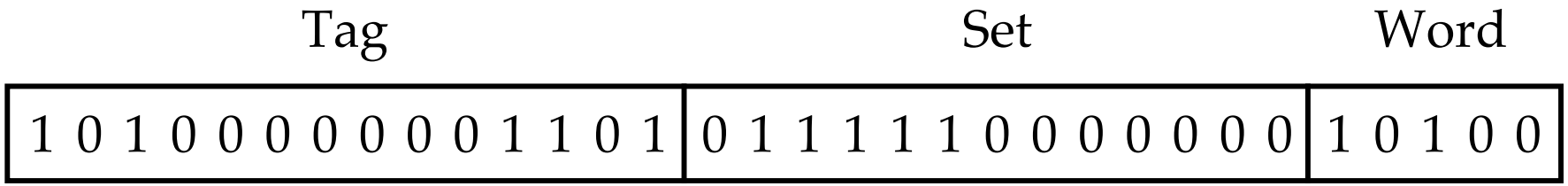


Set-Associative Mapping Example

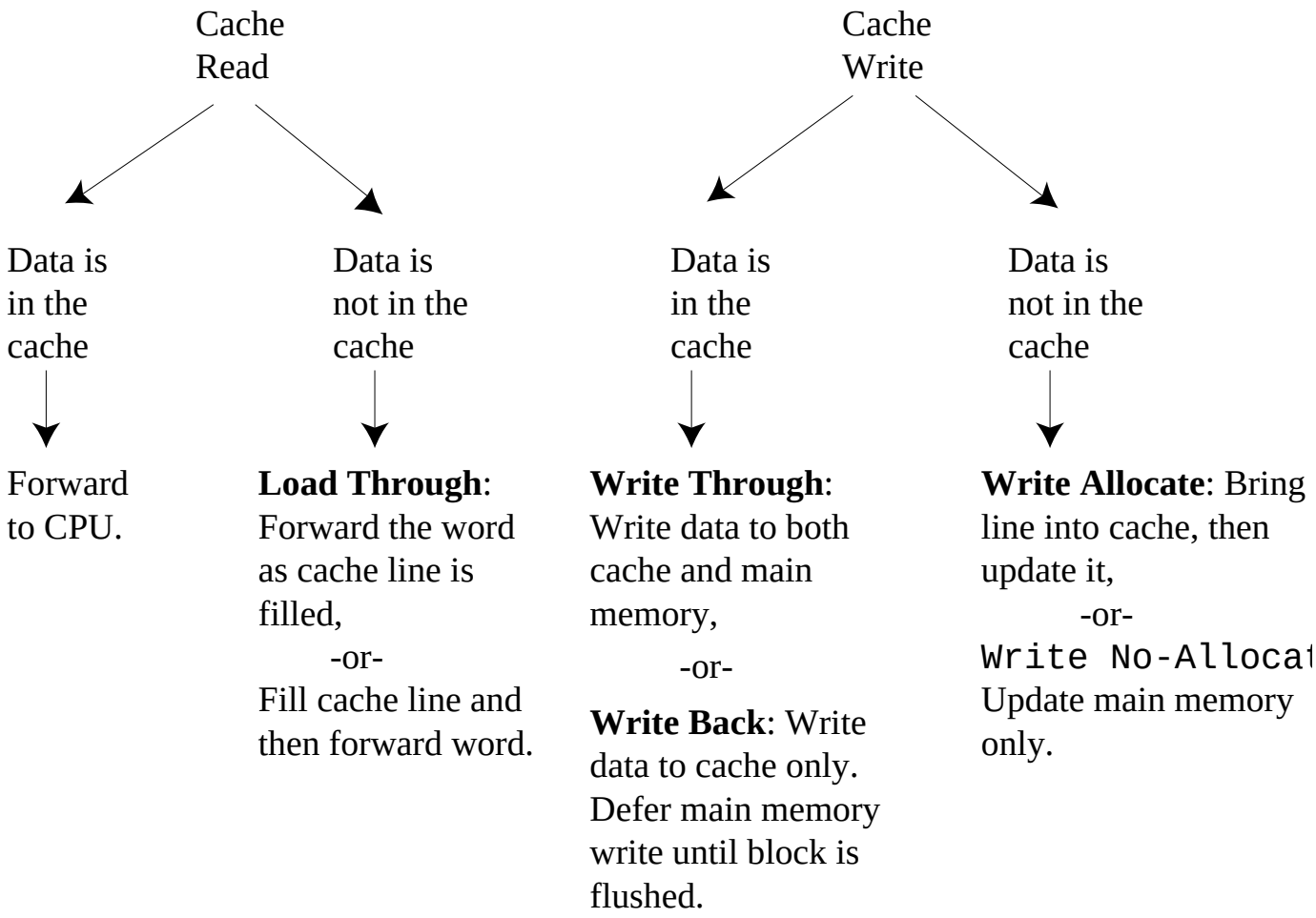
- **Consider how an access to memory location $(A035F014)_{16}$ is mapped to the cache for a 2^{32} word memory. The memory is divided into 2^{27} blocks of $2^5 = 32$ words per block, there are two blocks per set, and the cache consists of 2^{14} slots:**



- **The leftmost 14 bits form the tag field, followed by 13 bits for the set field, followed by five bits for the word field:**



Cache Read and Write Policies



Virtual Memory + Cache Memory

1. CPU instruction generates a virtual address.
2. Translation Lookaside Buffer (TLB) used to locate recently accessed page frames.
3. In case of a TLB miss, use page table(s) to find page frame for virtual address. TLB updated.

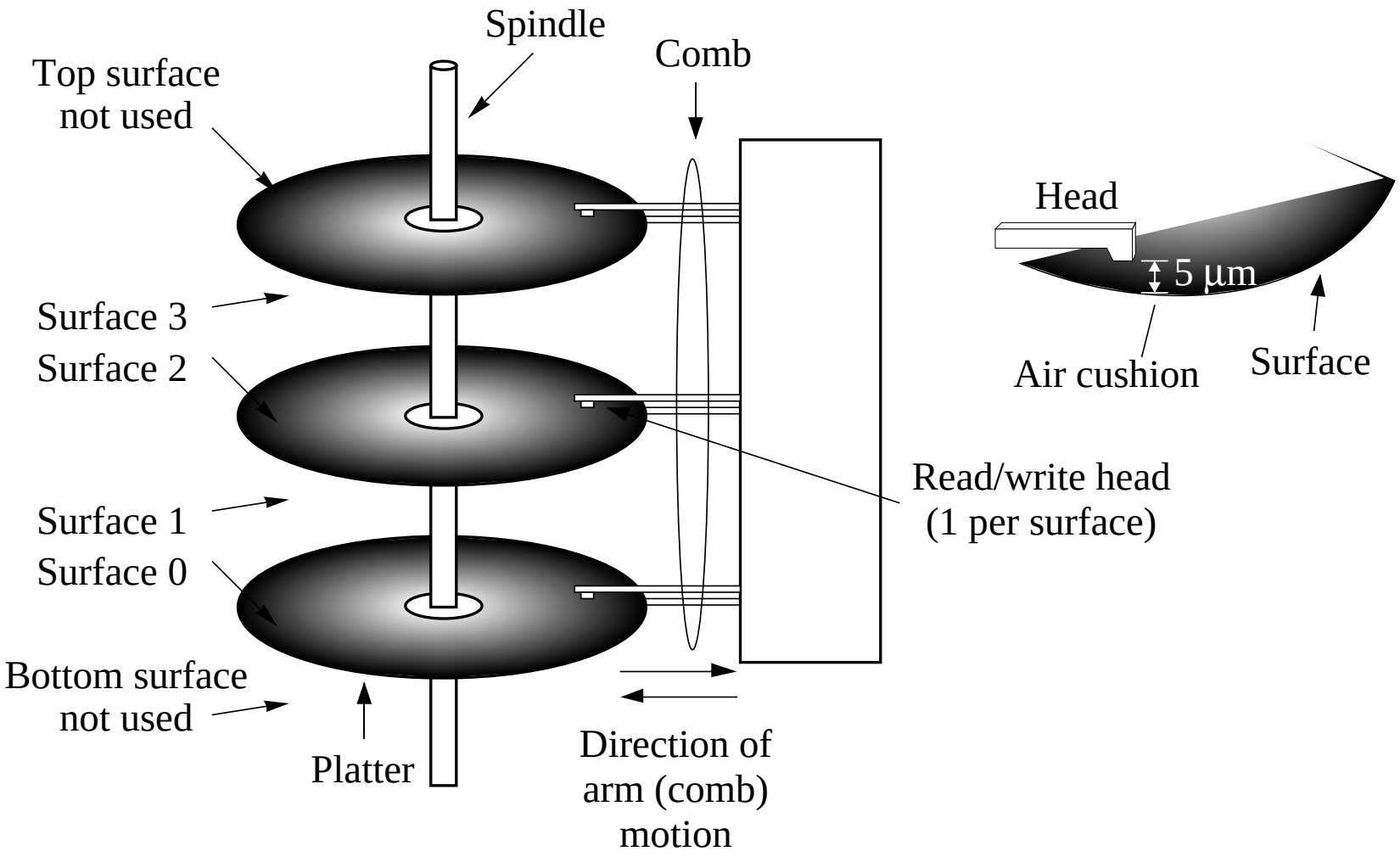
   Note: page frame may have been swapped to disk!
4. Virtual address is now converted into a physical address.
5. Search cache for recently accessed physical addresses.
6. In case of a cache miss, use main memory. Cache updated.

Input/Output

Basic Issues in I/O

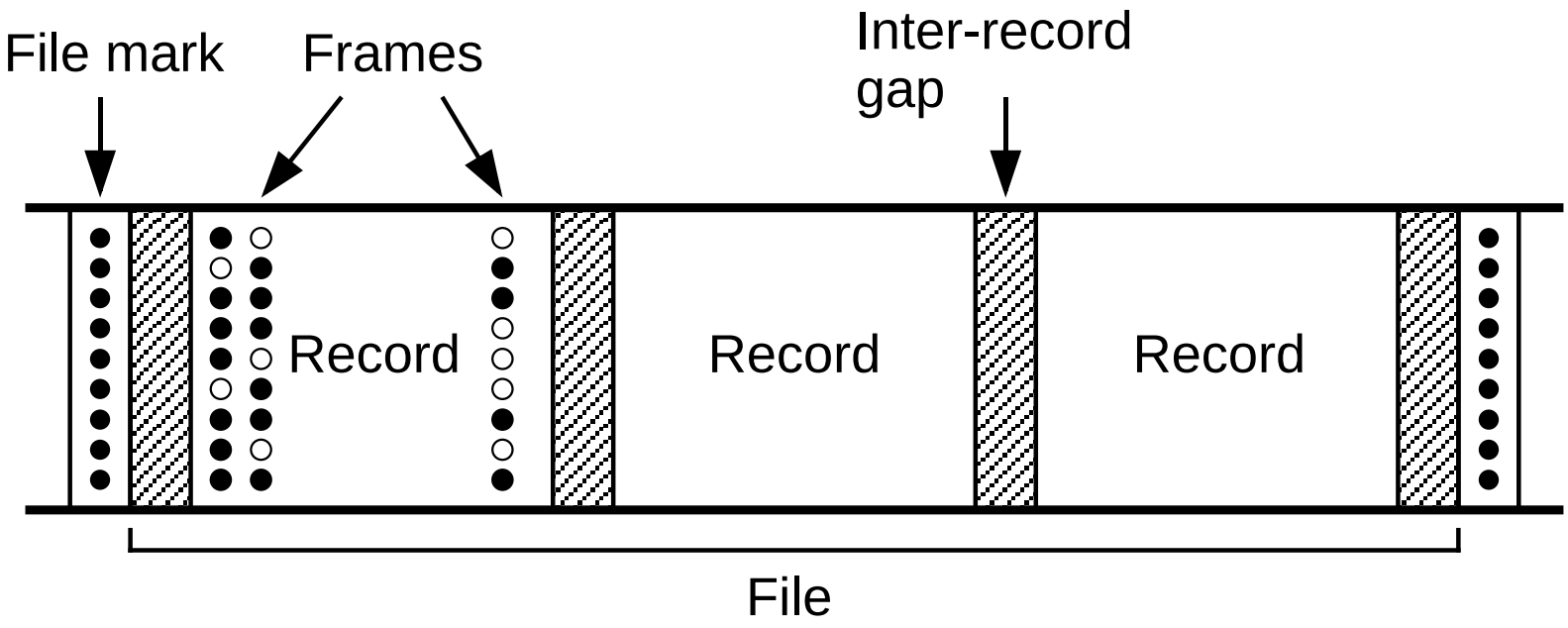
- Allow many components to communicate.
- Do not want to have $n^2/2$ connections to connect n components together.
- Devices communicate a different speeds.
- Some devices transfer lots of data, others very few.

A Magnetic Disk with Three Platters



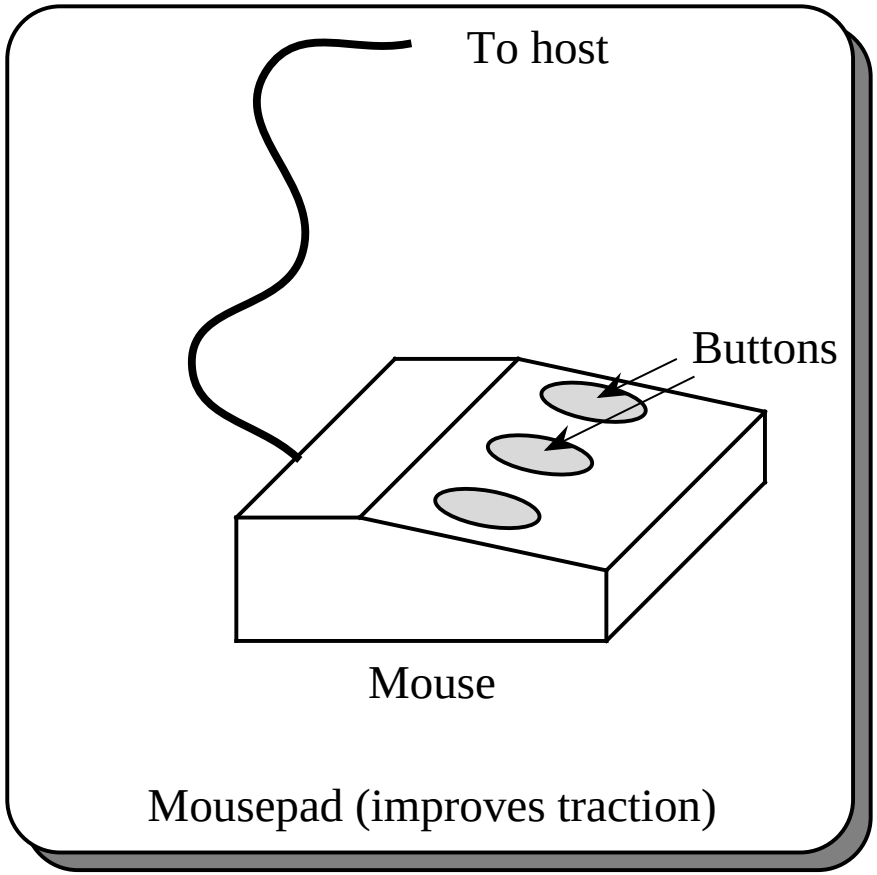
Magnetic Tape

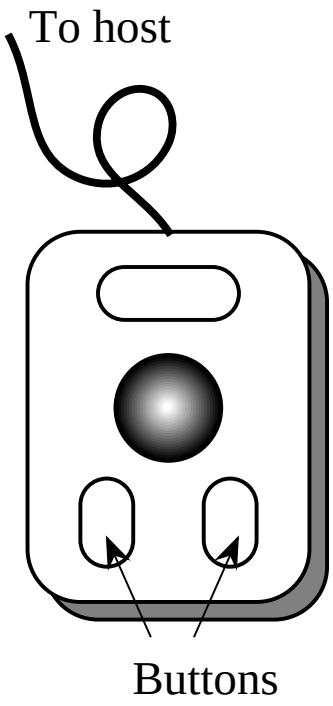
- **A portion of a magnetic tape (adapted from [Hamacher, 1990]).**



Mouse and Trackball

- **A three-button mouse (left) and a three-button trackball (right).**



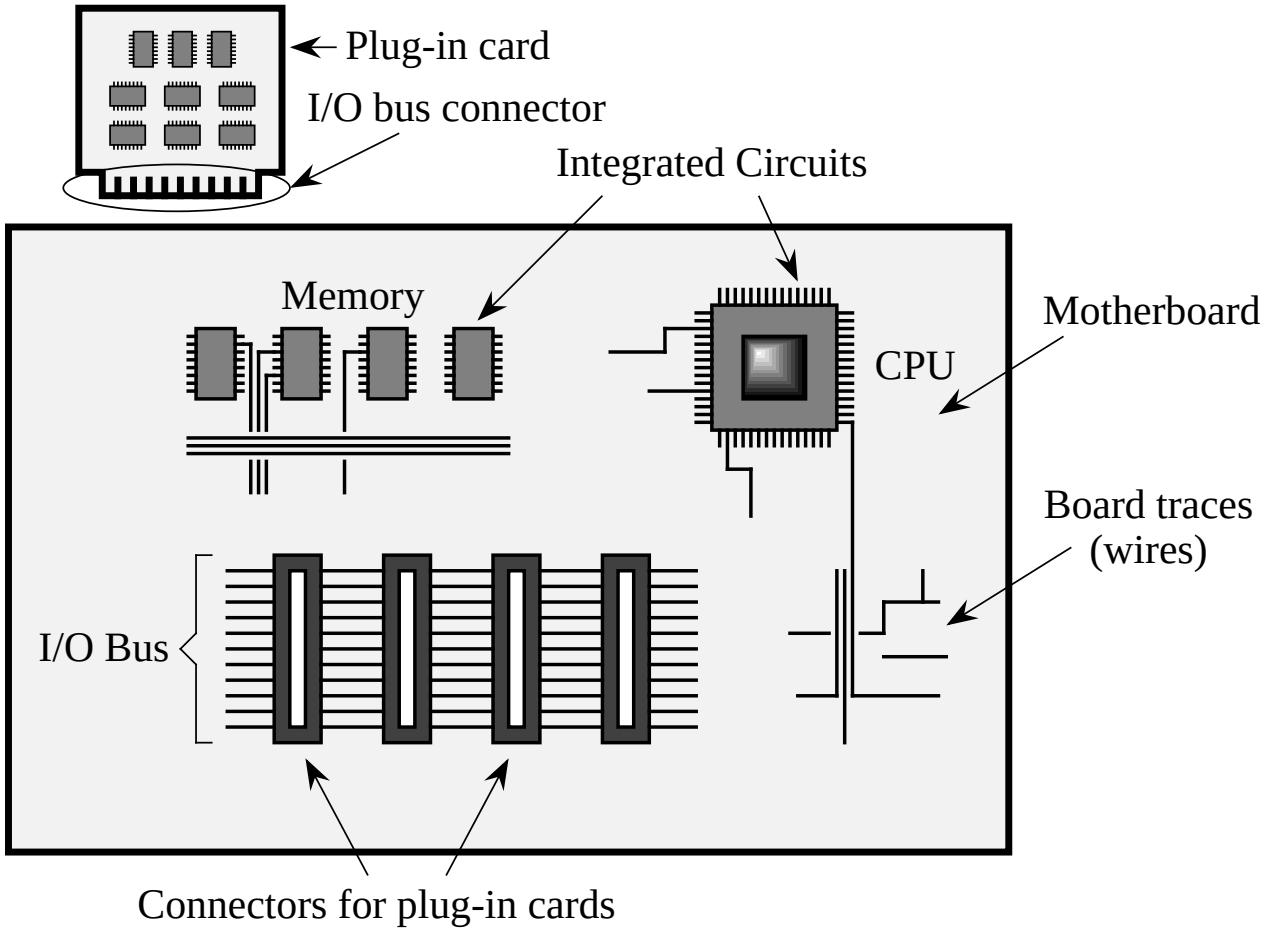


System Bus Issues

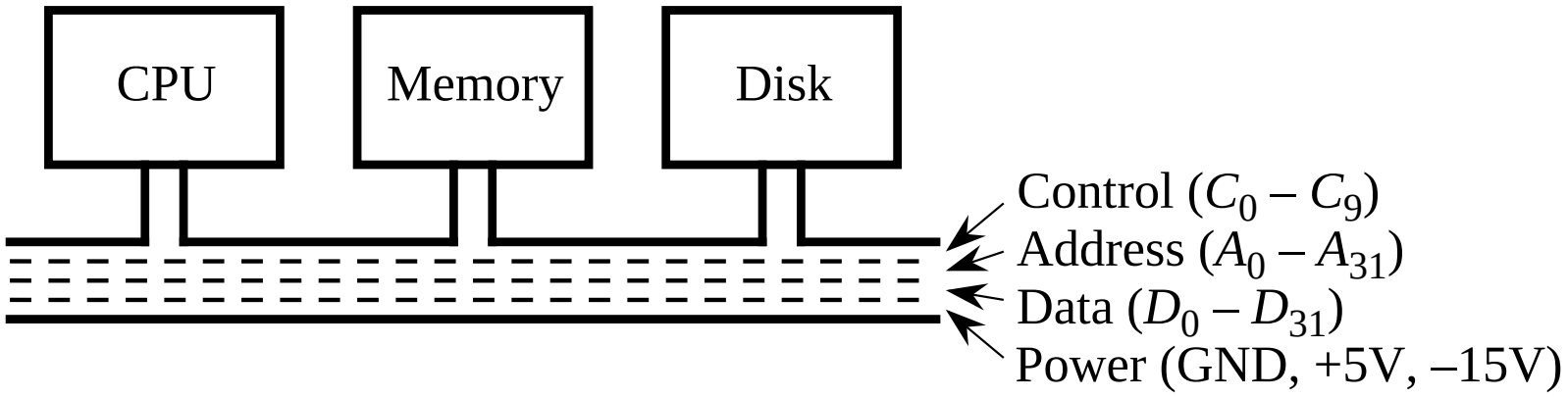
- Only n connections needed for n components.
- Bus can become a performance bottleneck.
- Synchronous vs asynchronous
- Bus arbitration.

Simple Bus Architecture

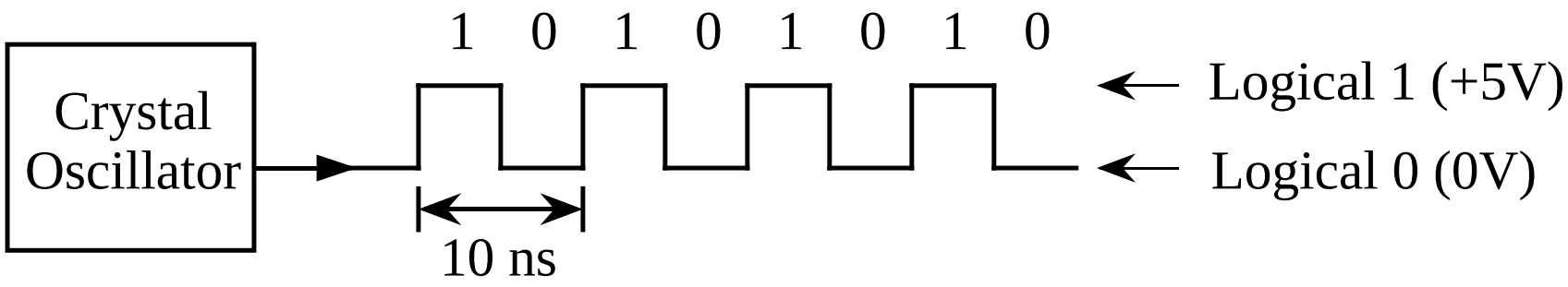
- **A simplified motherboard of a personal computer (top view):**



Simplified Illustration of a Bus

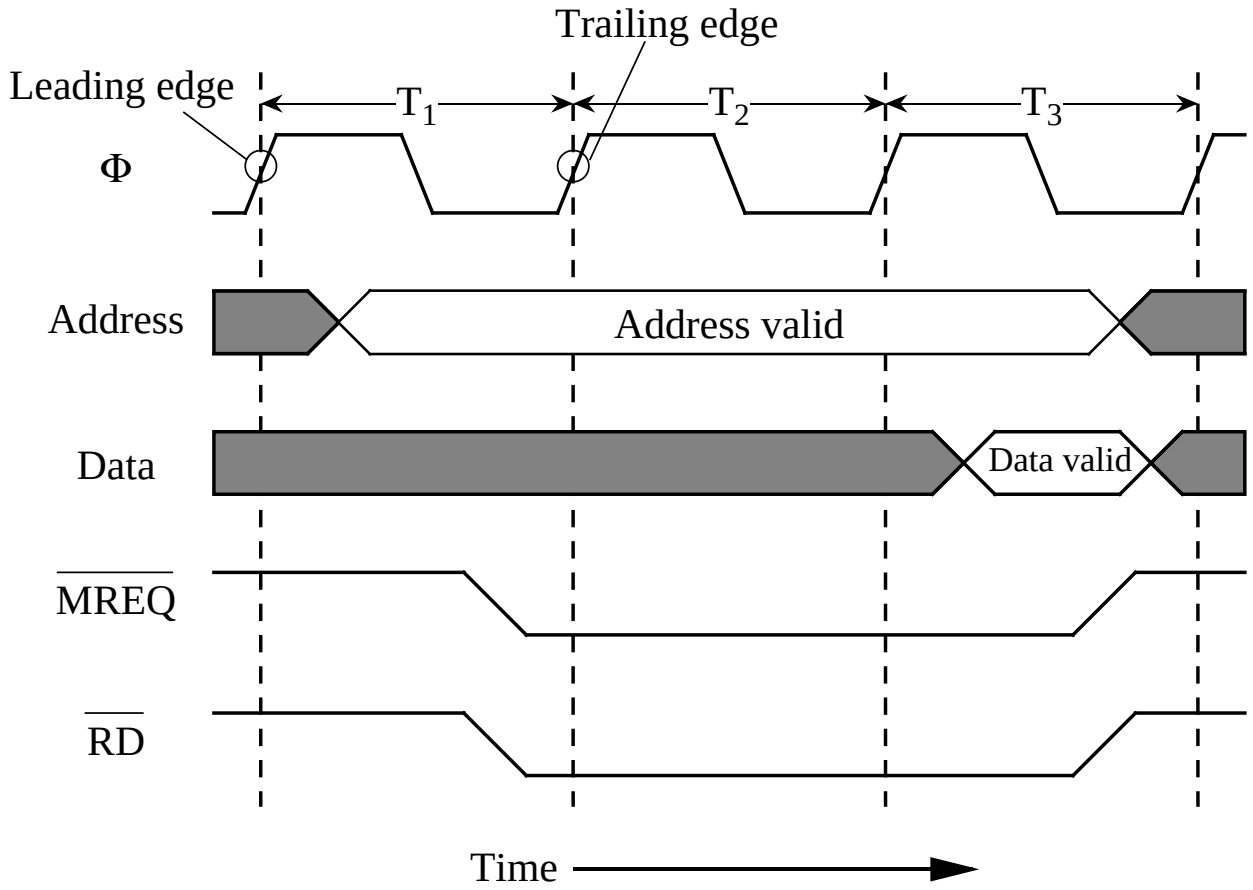


100 MHz Bus Clock



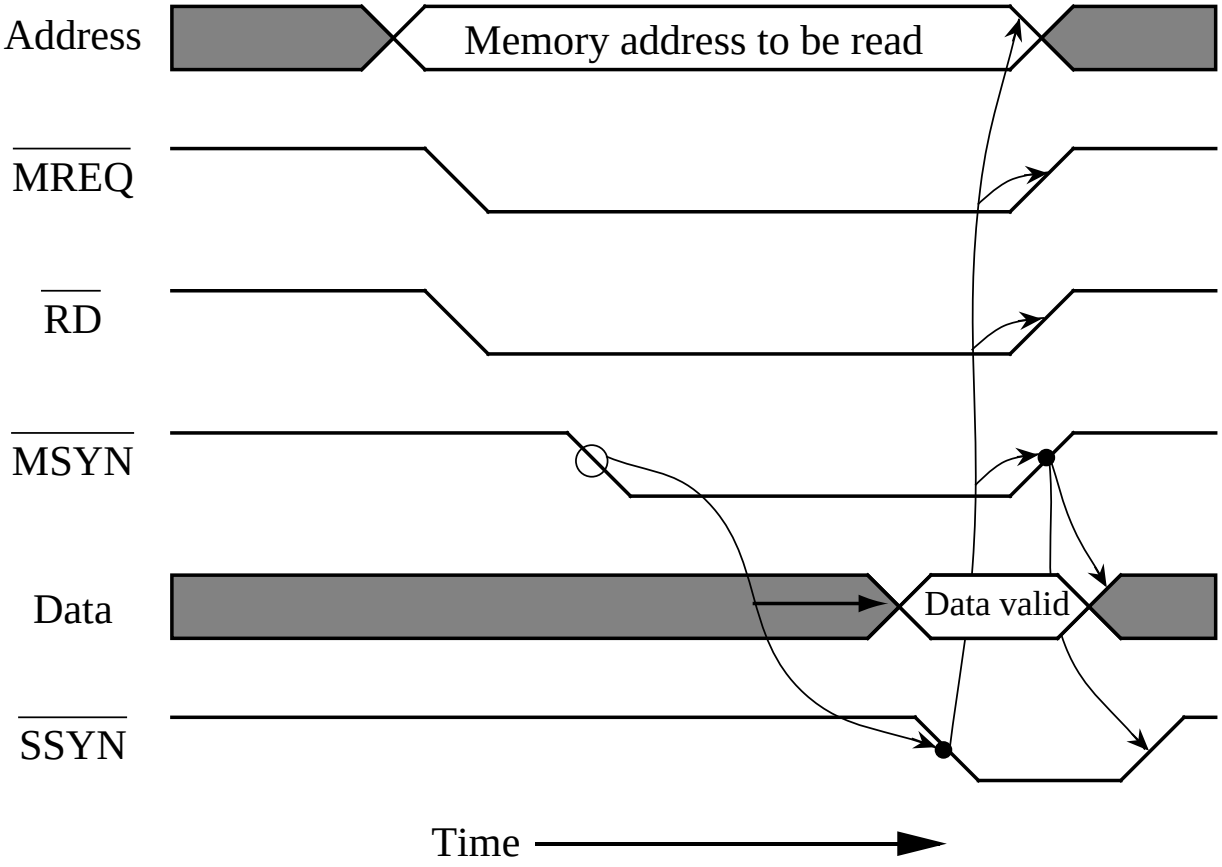
The Synchronous Bus

- **Timing diagram for a synchronous memory read (adapted from [Tanenbaum, 1999]).**



The Asynchronous Bus

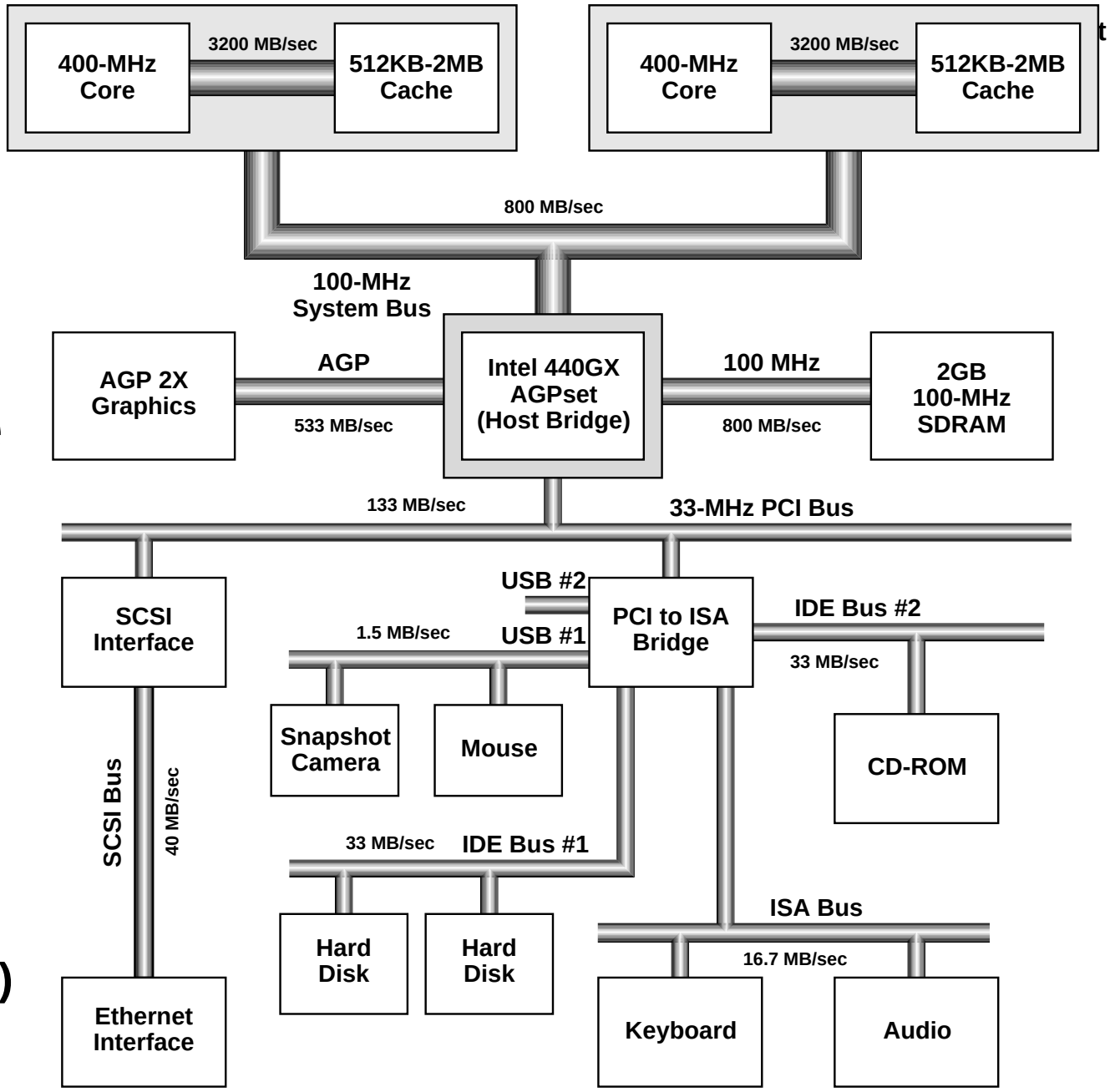
- **Timing diagram for asynchronous memory read (adapted from [Tanenbaum, 1999]).**



Bridge Based Bus Architecture

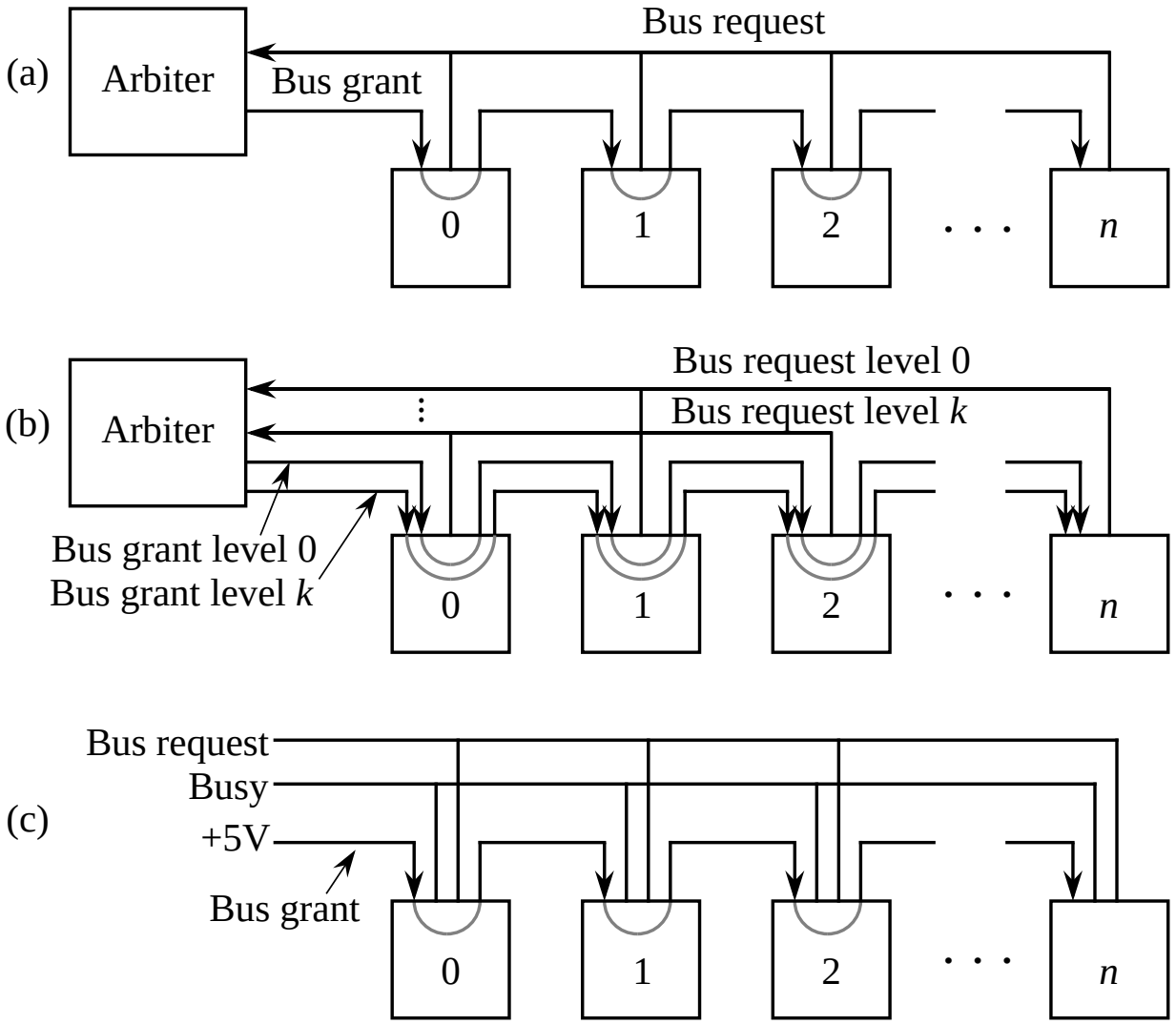
- Bridging with dual Pentium II Xeon processors on Slot 2.

(Source: http://www.intel.com.)



Bus Arbitration

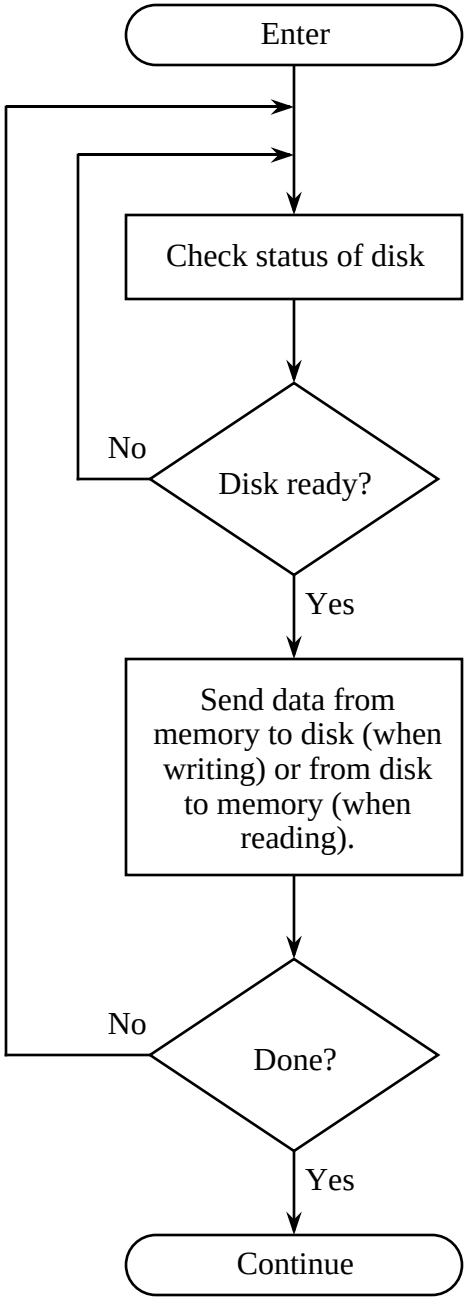
- **(a)Simple centralized bus arbitration; (b) centralized arbitration with priority levels; (c) decentralized bus arbitration. (Adapted from [Tanenbaum, 1999]).**



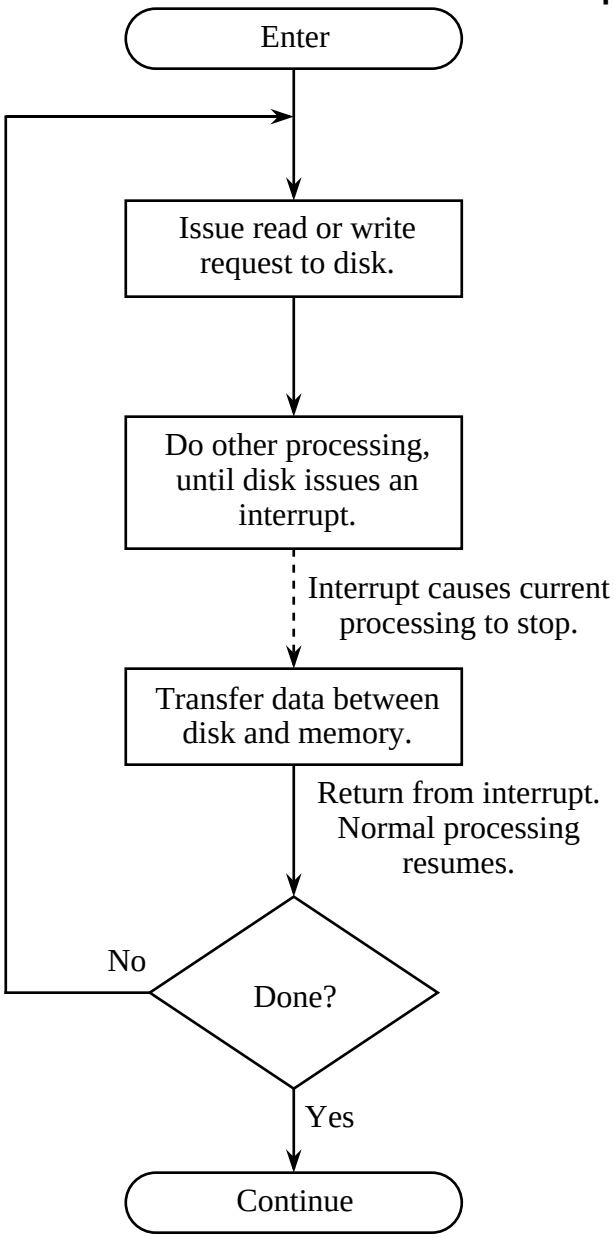
Types of I/O Control

- Programmed I/O = polling
- Interrupt driven.
- DMA = Direct Memory Access.
- Channel I/O: uses dedicated I/O processors.

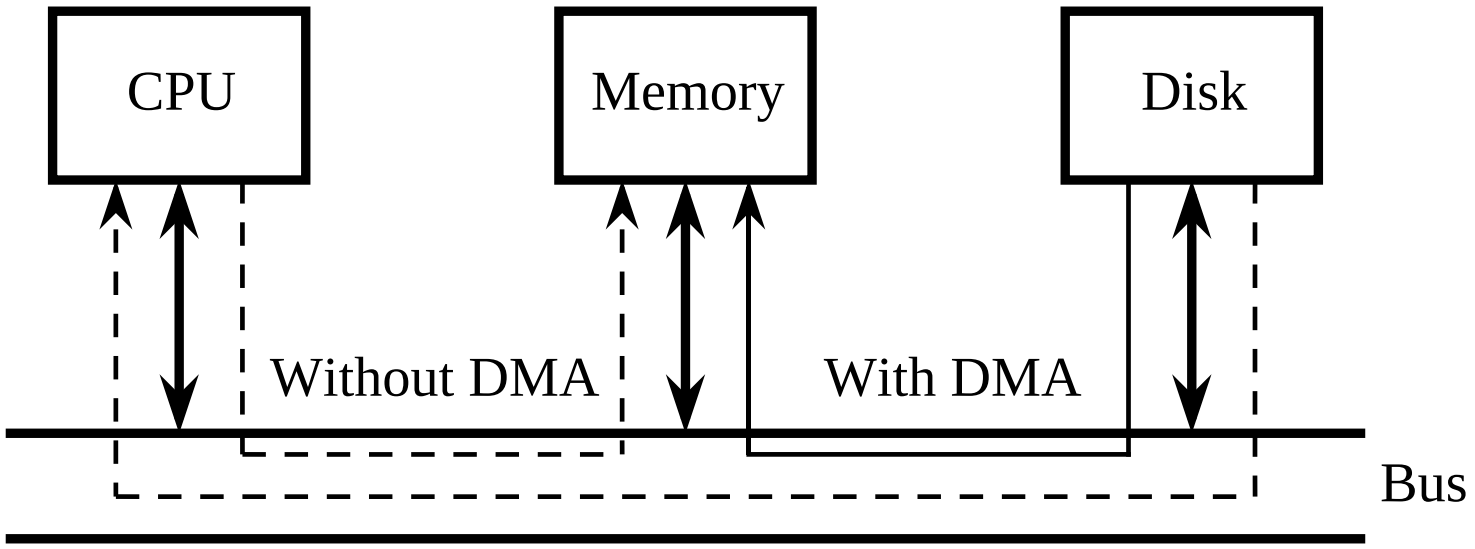
Programmed I/O Flowchart for a Disk Transfer



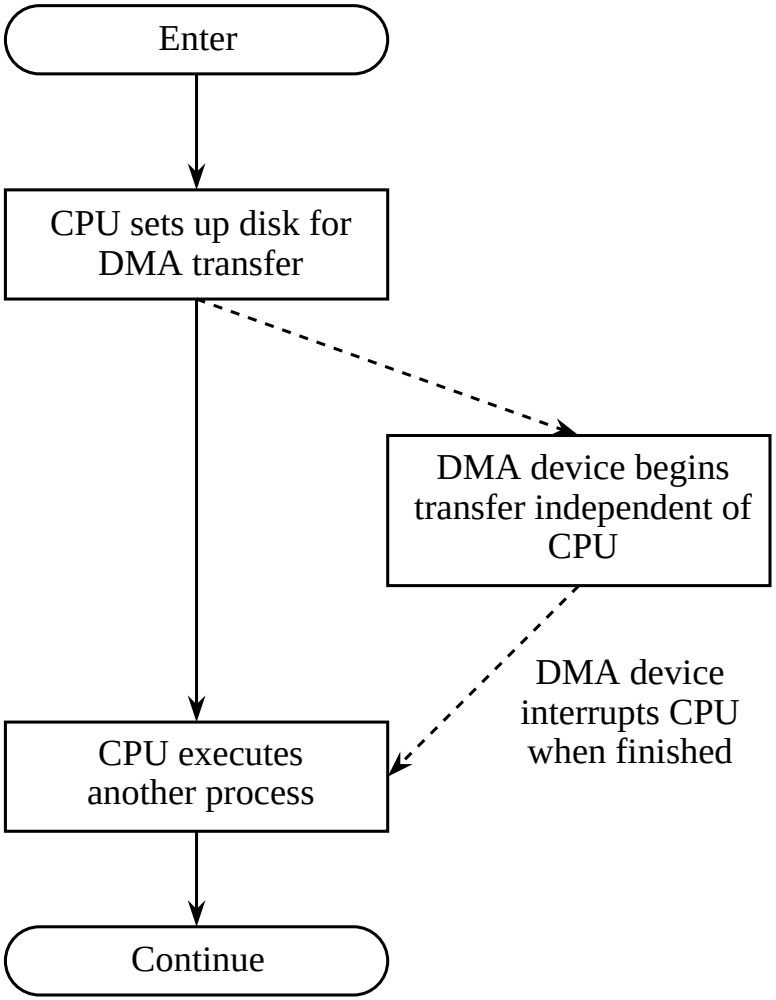
Interrupt Driven I/O Flowchart for a Disk Transfer



DMA Transfer from Disk to Memory Bypasses the CPU



DMA Flowchart for a Disk Transfer



Buffering

- Many I/O devices have on-board memory.
- Communication can be done at memory-to-memory speeds.
- More expensive, requires local controller.
- I/O becomes more like networking.

Next Time

- Performance Metrics
- Trends