COMPUTER HARDWARE

Input- Output and Communication

Memory Systems

Computer I/O

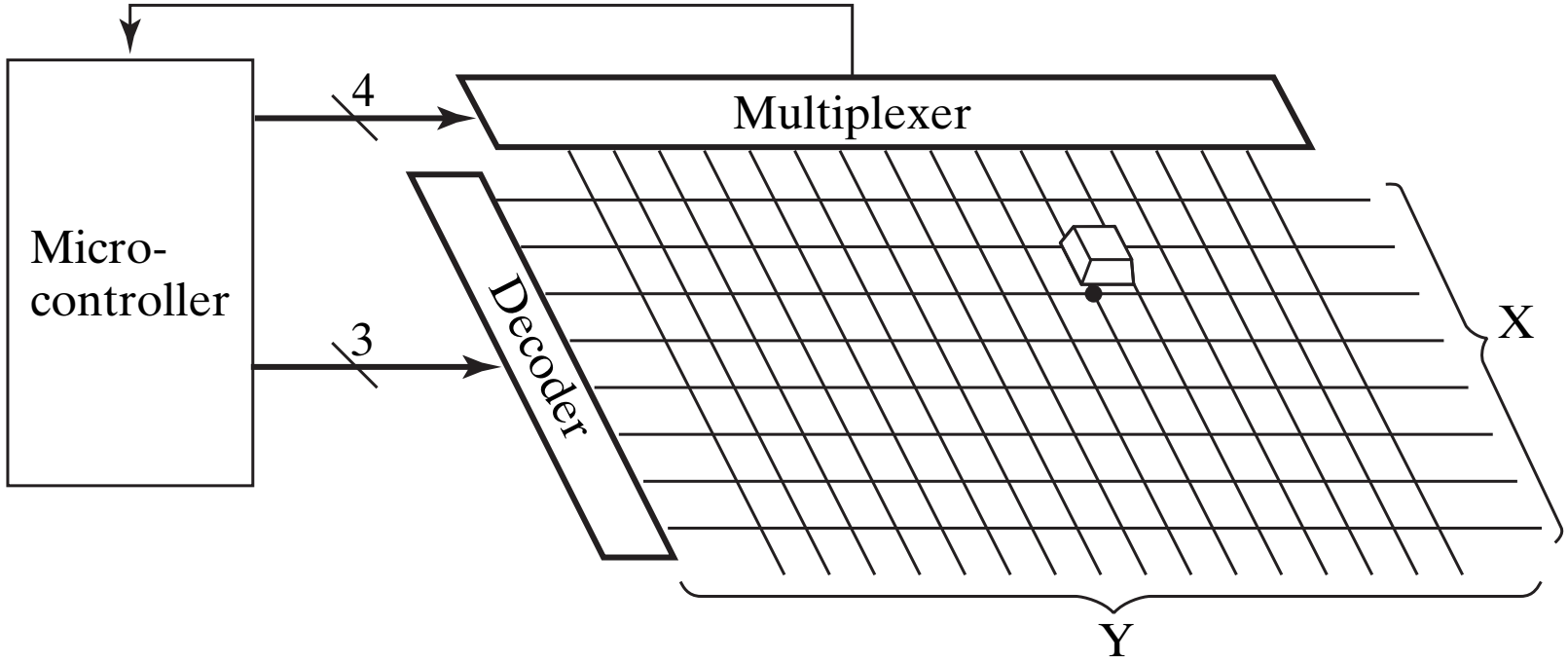
- I/O devices commonly found in Computer systems
  - Keyboards
  - Displays
  - Printers
  - Magnetic Drives
  - Compact disk read only memory (CD-ROM)
- Others
  - Modems or other communications devices
  - Scanners
  - Sound cards with speakers and microphones
- Significant numbers of computers, such as those used in automobiles
  - Analog-to-digital converters
  - Digital-to-analog converters
  - Data acquisition and control components

Sample Peripherals

- Devices that the CPU controls directly are said to be connected **online**.
- Input or output devices attached to the computer online are called **peripherals**
  - Keyboard
  - Hard Drive
  - Liquid Crystal Display Screen

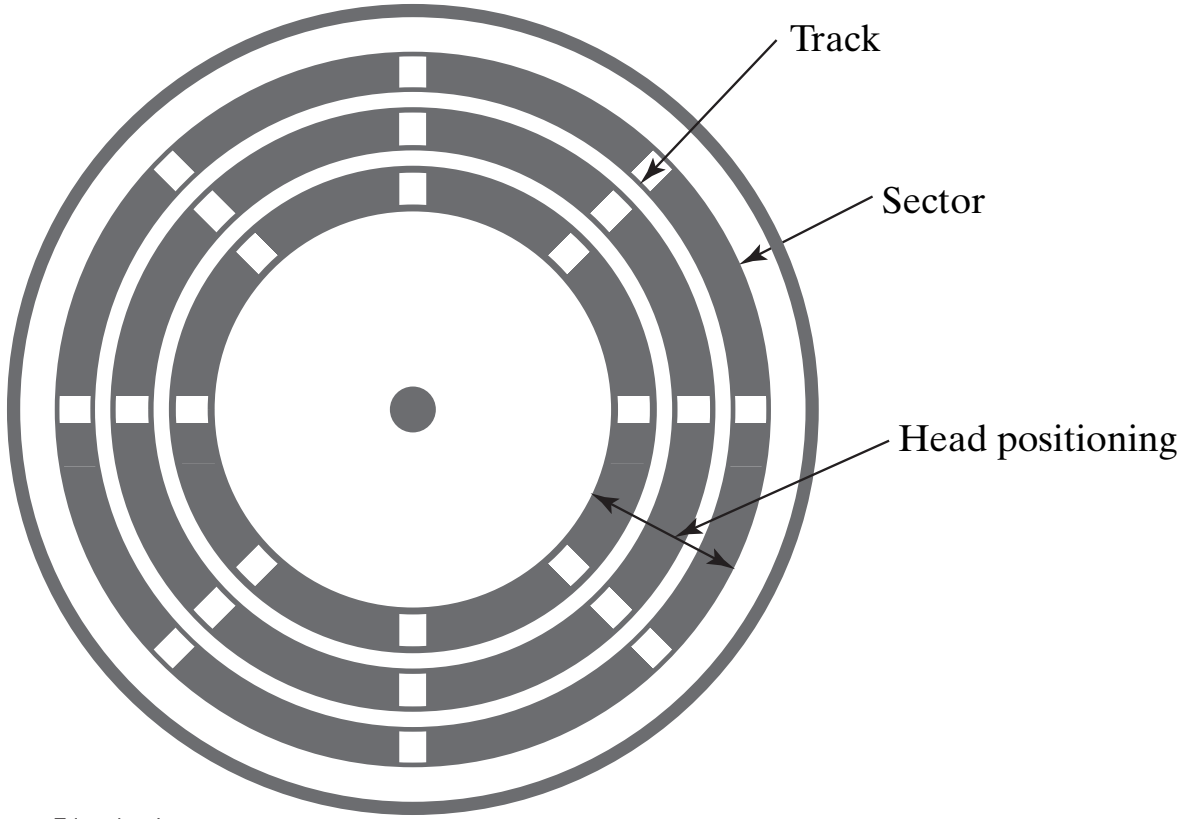
Keyboard

- **Scan matrix** lies beneath the keys.
- Whether a key is depressed and released, the control code at the time of the event is sensed and is translated by the microcontroller into **K-scan code**.
- When a key is depressed, a **make code** is produced, when a key is released a **break code** is produced.



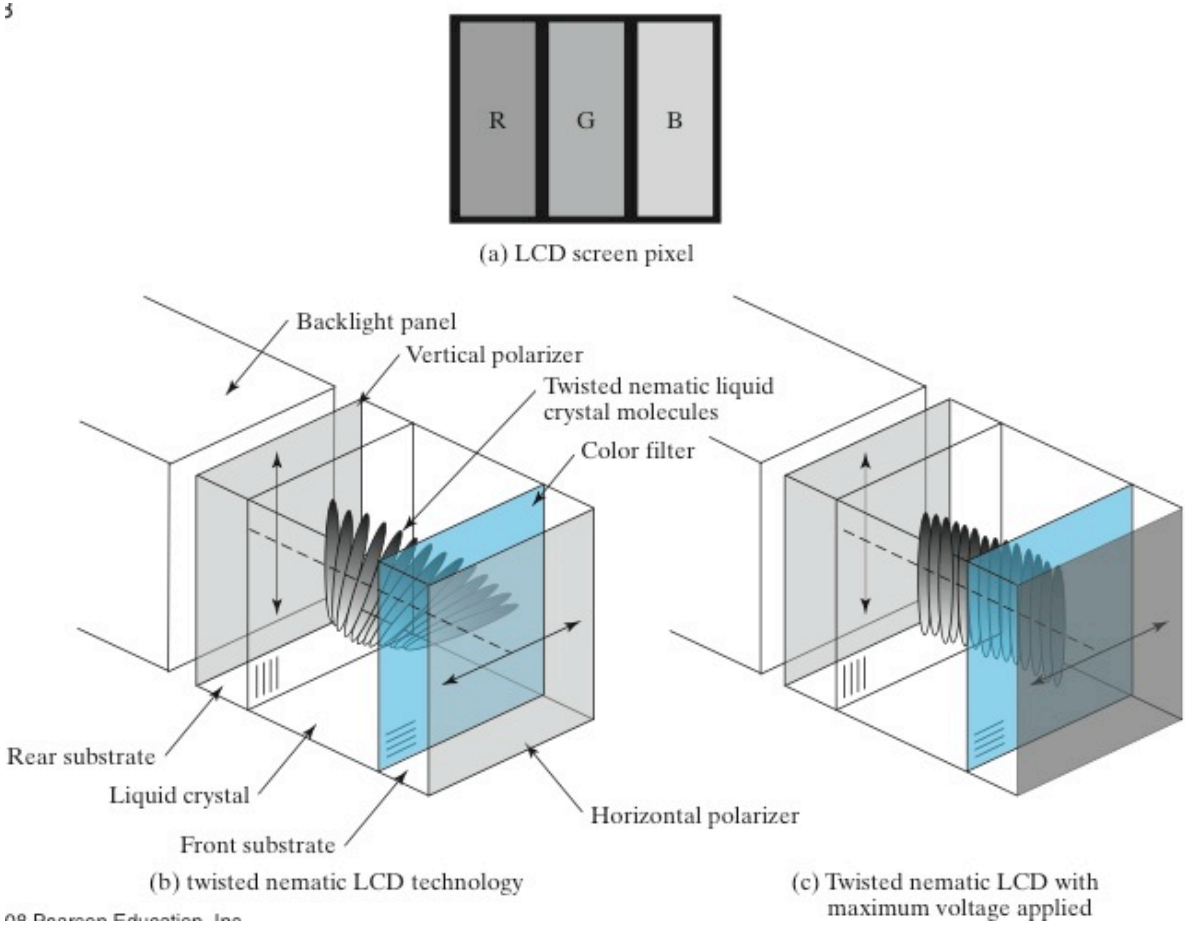
Hard Drive

- The hard drive is the primary intermediate-speed, nonvolatile, writable storage medium.
- Revolutions Per Minute (rpm) : rotational speed of a disk
- Disk **access** time and **transfer** time



Liquid Crystal Display (LCD) Screen

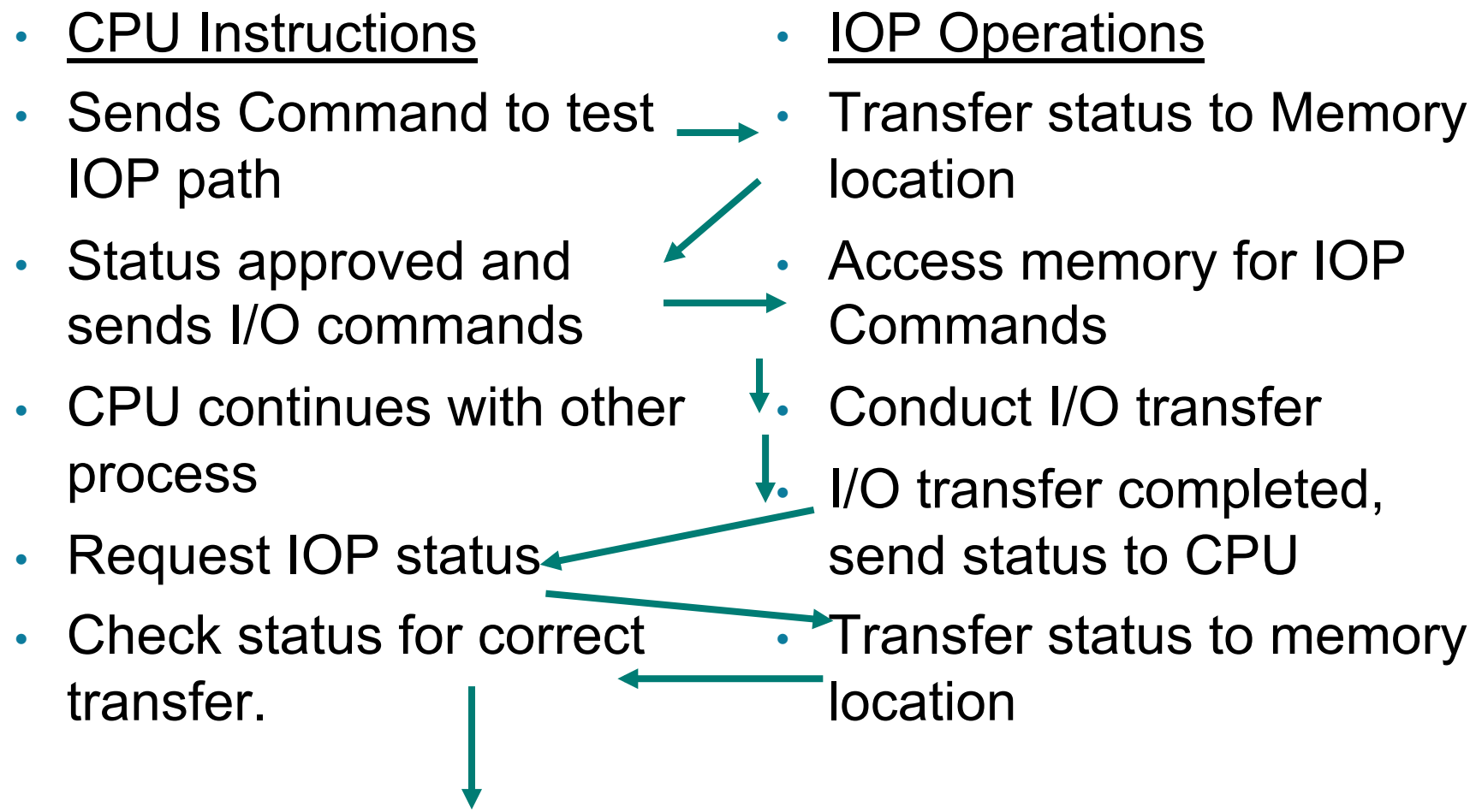
- The display screen is defined in terms of picture elements called **pixels**
- The color display has three subpixels correspond to the primary colors red, green and blue (RGB)



I/O Processors

- I/O Processors handles all of the interactions between the I/O devices and the CPU.
- I/O Processors communicates with input and output devices through separate address, data, and control lines.
  - This provides an independent pathway for the transfer of information between external devices and internal memory.
- Relieves the CPU of ‘I/O device chores’
- Input-Output Processor (IOP)
  - Classified as a processor with direct memory access capability.
  - IOP fetches and execute its own instructions
    - Independent of the CPU
    - CPU initiating the IOP program
  - CPU is the master processor. IOP is considered the slave processors
  - There can be more than one or more IOP’s

The Process of the IOP

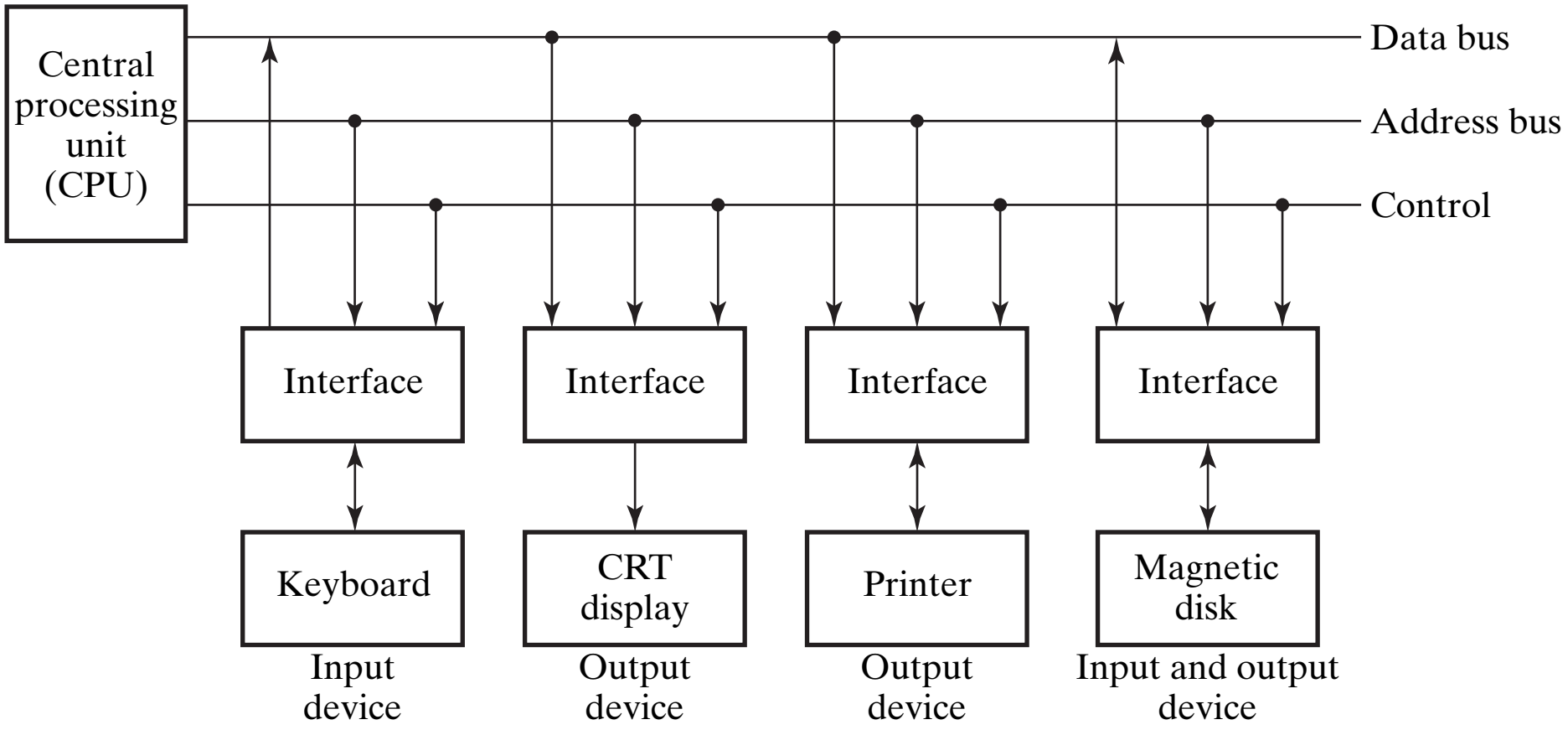


I/O Interfaces

- The differences with CPU
  - Peripherals are often electromechanical devices whose manner of operation is different. A conversion of signal values may be required.
  - The data transfer rare is different from the clock rate of the CPU. A synchronization mechanism may be needed.
  - Data codes and formats differ from the word format in CPU.
  - The operating modes od peripherals differ from each other.

I/O Bus and Interface Unit

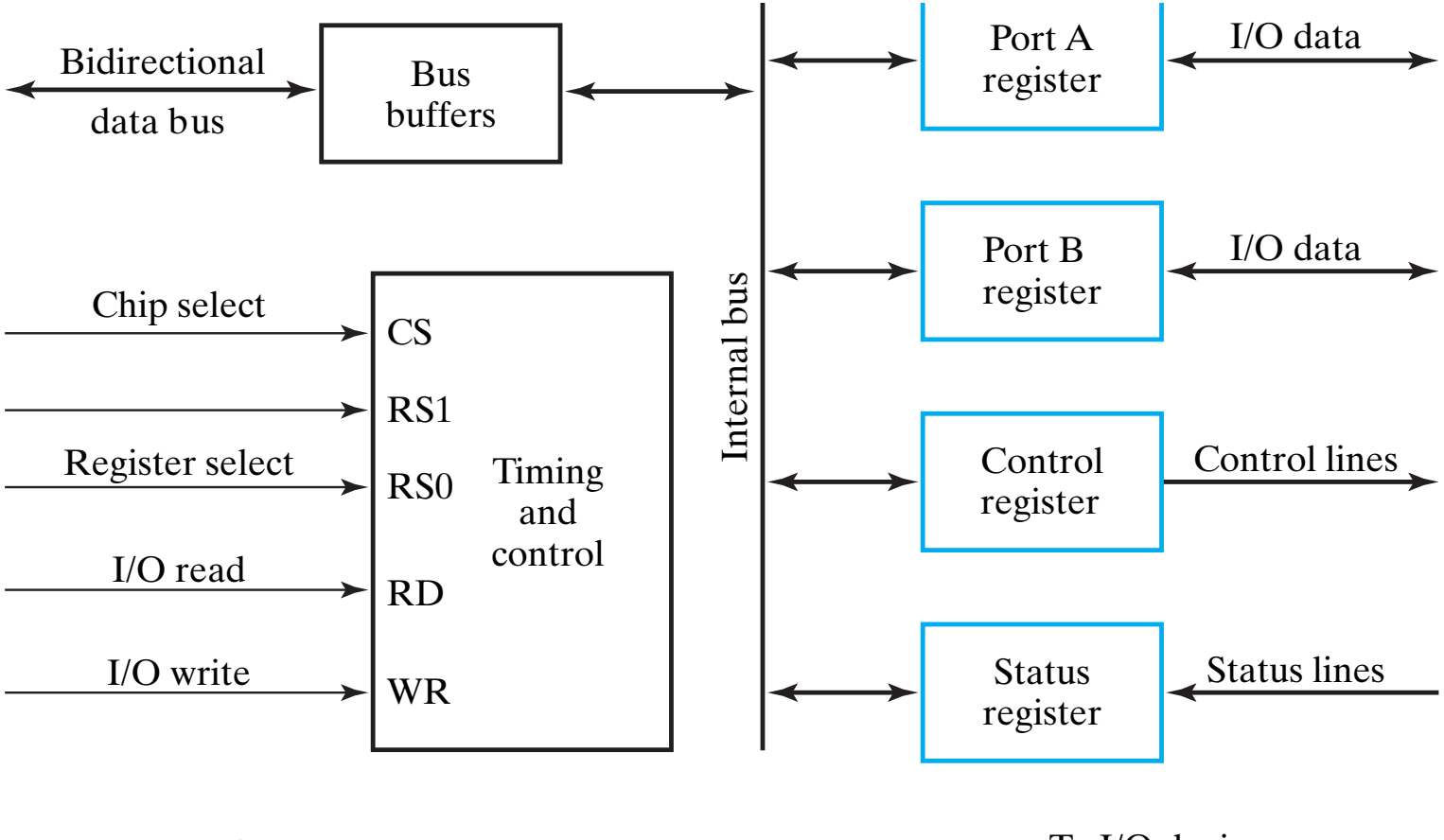
- A typical communication structure between the CPU and several peripherals.



I/O Configurations

- The I/O configurations
  - Memory-mapped: Use common data, address, and control busses for both memory and I/O
  - Isolated: Share a common address and data but, but use different control lines.

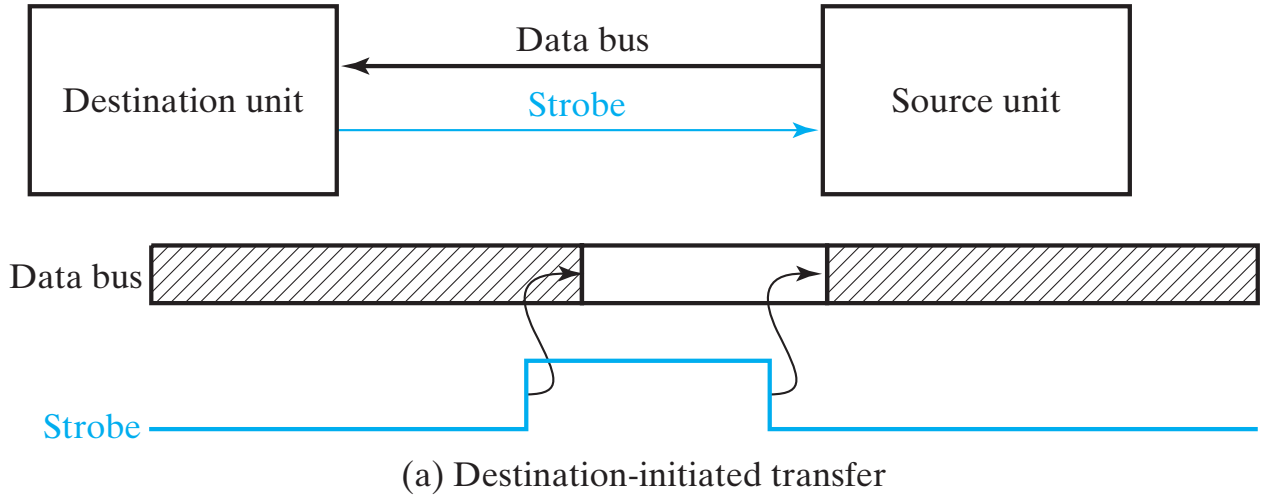
Example of I/O Interface



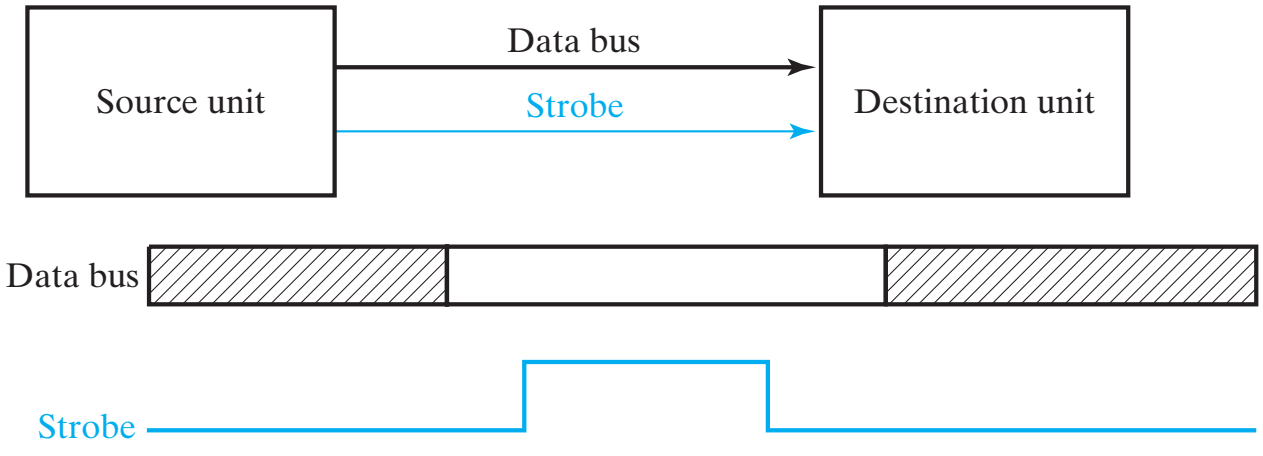
CS	RS1	RS0	Register selected
0	X	X	None: data bus in high-impedance state
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Strobing

- The data bus between the two units is assumed to be made bidirectional by the use of three state buffers.



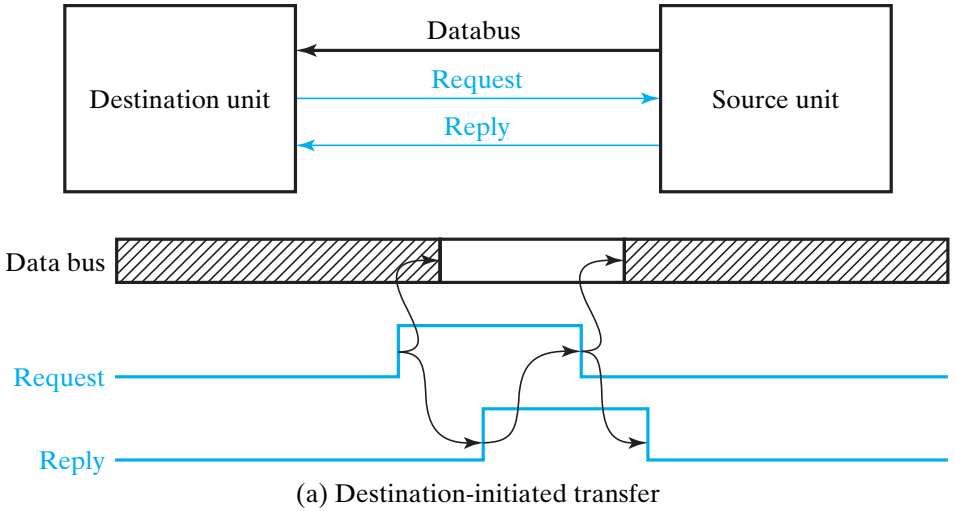
(a) Destination-initiated transfer



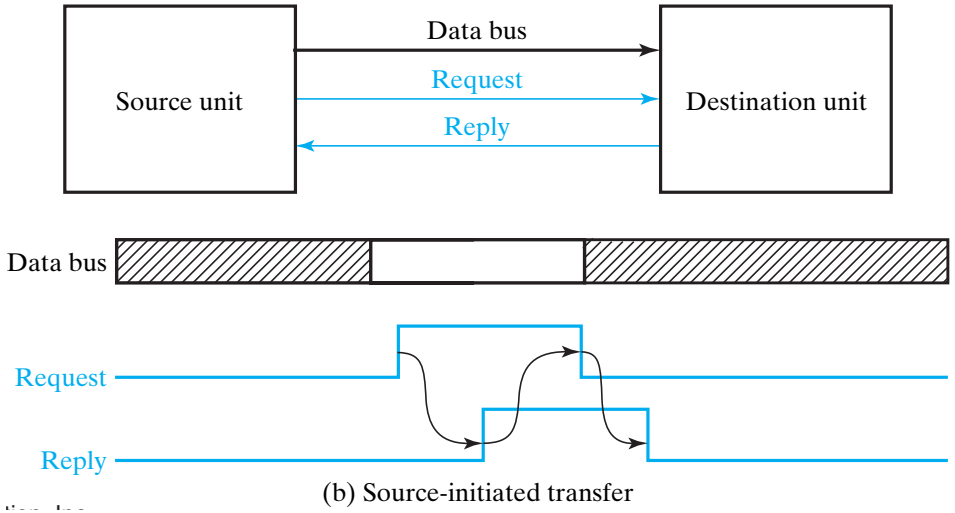
(b) Source-initiated transfer

Handshaking

- Use two control signals to deal with the timing of transfers



(a) Destination-initiated transfer



(b) Source-initiated transfer

Serial Communication

- Two forms of communication
  - Parallel communication
    - Transfers more than one bit of data at a given time
    - N-bits transmitted at the same time through n- wires
    - Faster but requires many wires and is used in short distances
    - EX: Input/output devices, DMA controllers, and I/O processors
  - Serial Communication
    - Serial communication refers to devices that cannot handle more than one bit of data at any given time by design.
    - Requires one wire and is slower.
    - Usually CPU use Parallel communication, if the device is serial, then the data is converted to use Parallel communication
    - EX: Modems

Serial Communication

- Serial transmission is slower but less expensive and easy.
- The devices that do the conversion to digital signals are called data sets or modems.
- Three different modes
  - **Simplex**: Carry information in one direction only.
  - **Half-duplex**: capable of transmitting in both direction but in only one direction at a time.
  - **Full-duplex**: send and receive data in both directions simultaneously.
- Device must agree on number of data bits per data transmission.
  - Parity Bits: Error checking
  - Stop Bits: End of transmission

Two types of Serial Communication

- Asynchronous Serial Communication
  - Interacts with devices outside of the computer
    - Ex: modem connecting to another computer
  - Transmit individual bytes instead of large blocks
  - Do not share a common clock.
- Synchronous Serial transmission
  - Transmits block of data in frames.
    - Frames are had head in front of the data and a tail at the end of the data.
  - The head and tail contain information that allows the two computers to synchronize their clocks

Synchronous Transmission

- Instead of transmitting a start and stop bit for each data value, Synchronous transmission strings together several data values into a data block called a ‘frame.’
- There are several layers in the frame, similar to a data packet.
- There is a leading information, address of where the data is going, control ensure correct destination, the data it self, Cyclic redundancy check (CRC) to check there is no error and the trailing information

Asynchronous Serial Communication

- Each byte is transmitted separate entity.
  - The Device must be able to recognize:
    - When transmission is occurring
    - When to read a bit of data
    - When the transmission ends
    - When the transmission is idle (no data being transmitted)
- Steps:
  - Device 1 transmission will output a ‘start bit’
    - A line of transmission is used to describe the communication.
  - Device 2 receives and confirms the bit.
  - Device 2 begins to read a data bit off the line
  - Then the process repeats however many data bits are on the line.
  - The least significant bit is sent first and most important significant bit is sent last.
  - Then Device 2 receives and confirms ‘stop bit’

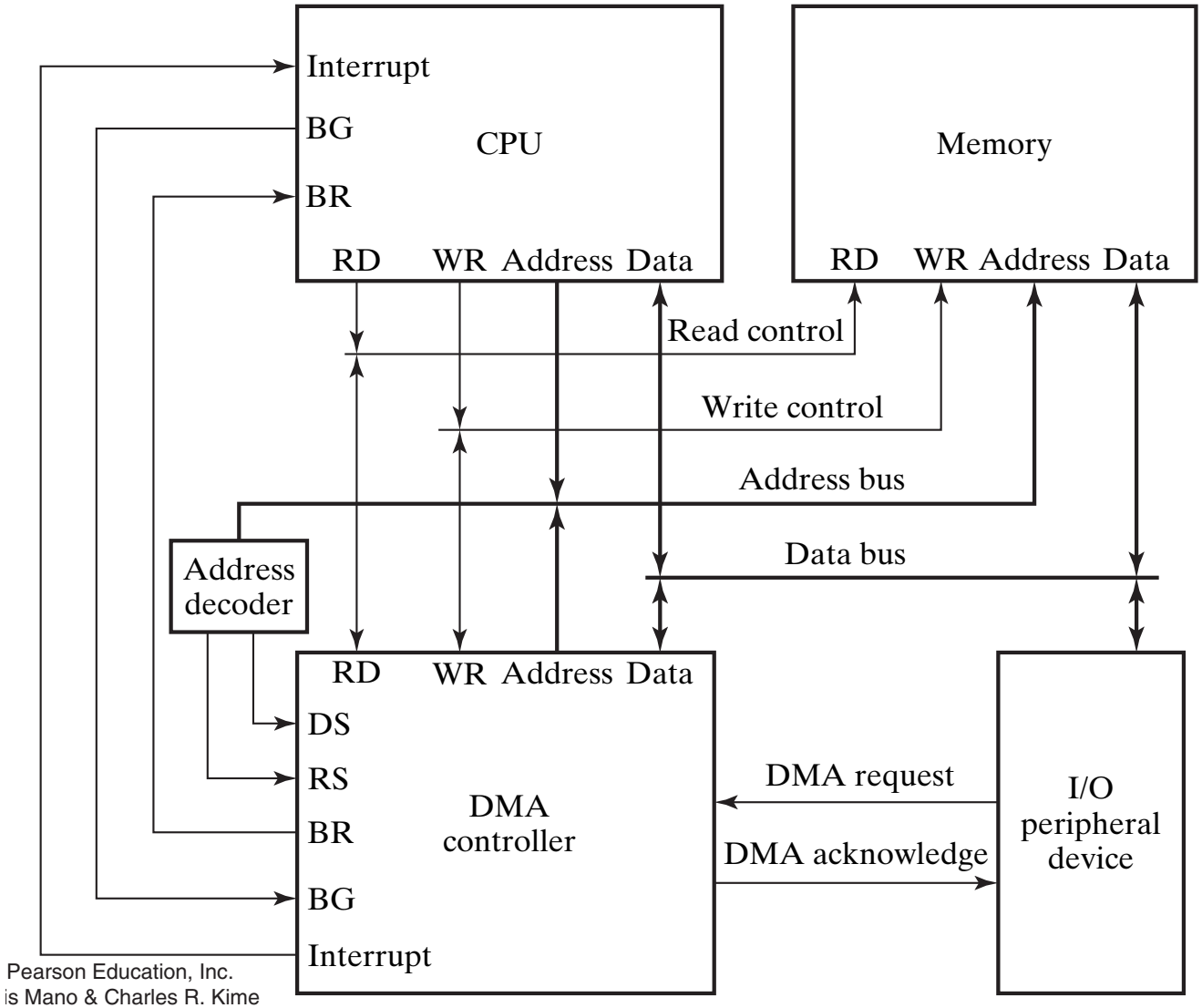
Universal Asynchronous Receiver/Transmitters (UARTs)

- Asynchronous serial communication is a popular function. Manufacturers have designed special chip to deal with Asynchronous serial communication. This relieves the CPU of this task
- It is used to convert serial communication to parallel communication when receiving and convert parallel communication to serial when sending
- CPU sends command to the UARTs control register to determine the number of data bits, parity, and the number of stop bits to be used in transmission

Modes of Transfer

- Data transfer to and from peripherals may be handled in one of three possible modes.
  - Data transfer under program control
  - Interrupt-initiated data transfer
  - Direct memory access (DMA) transfer.
- DMA
  - DMA improves system performance by speeding up data transfer between memory and I/O System
  - Bypass CPU, allow CPU to be use in another process.
  - DMA controllers must manipulates each data transfer from I/O devices, and can only read

DMA Transfer

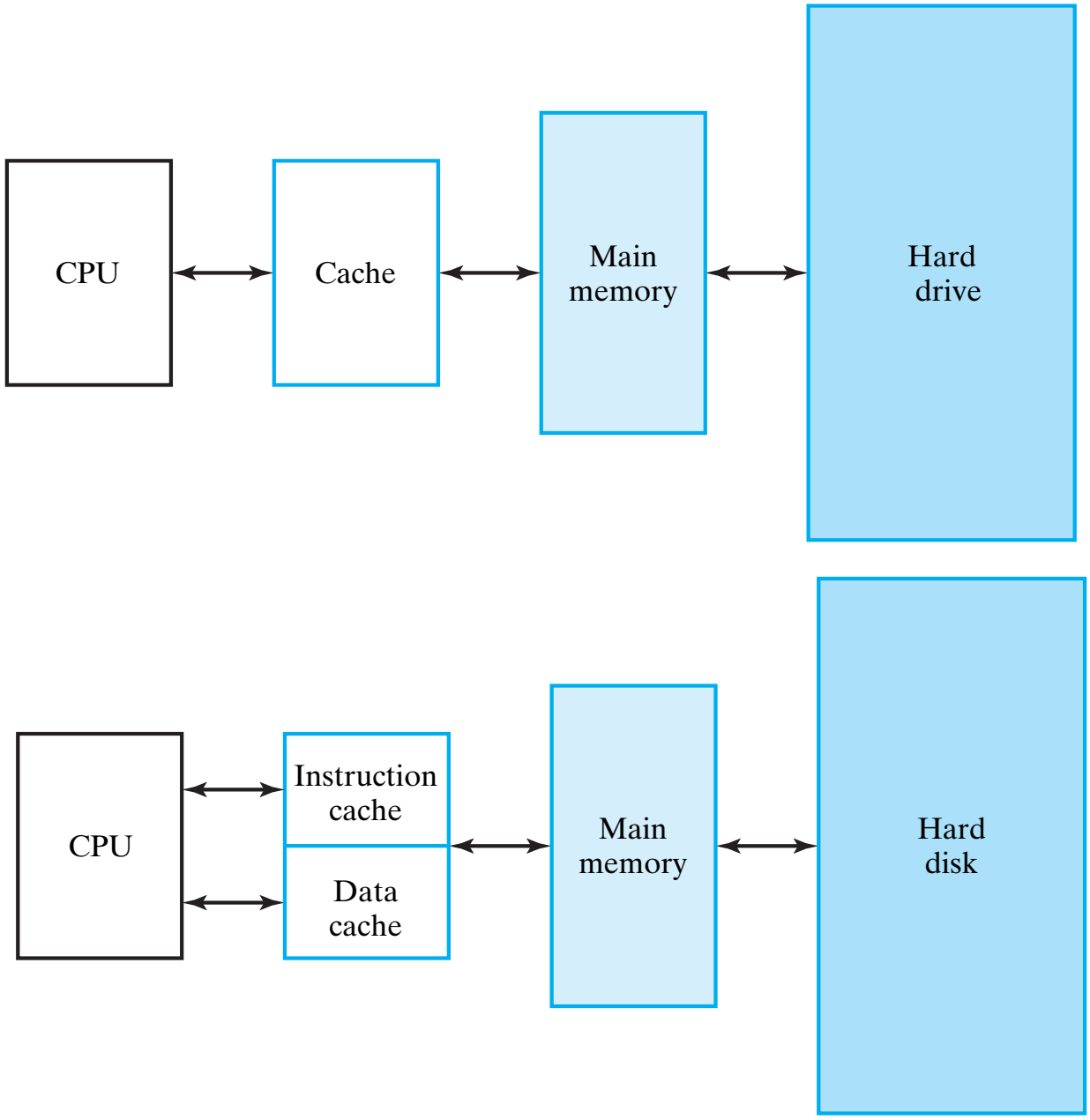


The Memory System: Memory Hierarchy

A **Memory System** is normally comprised of a hierarchy of memories:

- **Cache** - very fast (1 or 2 cycle access), but small (e.g. 32 KB-64 KB)
  - built with SRAM on-board the processor chip
  - designed as two separate caches (to improve bandwidth) - one for **instructions** and one for **data**
- **Main Memory** - larger (typically 32 MB – 512 MB) and slower (50 ns access) than cache
  - built with DRAM chips on separate modules/card
- **Virtual Memory** - very large (say 2 GB - 16 GB), but also very slow (15 - 20 ms access)
  - built with magnetic (hard) disk
- Ideally, we would like the memory system to always appear as very large and very fast!!

The Memory System: Memory Hierarchy



Memory Systems: Hierarchy

- Concept of an **infinite cache**:
  - fetches by the CPU for instructions or data normally come from cache (say 95% of time)
  - if instructions or operands are not in cache, a "miss" occurs and CPU waits while MMU (memory management unit) goes to main memory for the missing instruction or operand
  - on the very rare occasion that the operand or instruction is not in main memory, the CPU must go to the hard disk to find it (while the processor either waits idle or branches)
    - most of the time the instructions/data are available in cache giving the appearance of a large, fast memory!
- Memory addressing: 32 bit address can access 4 GB of data/ instructions
- Speed & Cost of 4GB DRAM Main Memory:
    - if all memory were only main memory (DRAM), 4 GB would cost $24,000 at $6/MB
    - access time would be only 50 ns, rather than the 2-3 ns obtainable with on-board cache
  - ***Memory hierarchy is essential to achieving high speed, large memory, & low cost!!!***

The Ideal Memory System

- Assume the memory hierarchy on the previous chart is comprised of the following:
  - 2 on-chip 32 MB caches running at 3 ns cycle time (single cycle cache)
    - perform instruction fetch from I-cache while concurrently fetching operands or writing data in D-cache
    - assume we find instructions and data in the caches (hit) 95% of the time
  - 64 MB DRAM main memory operating at 50 ns cycle time
    - assume of the 5% cache misses, we find the data in main memory 99.9999% of the time, i.e. we have to go to the hard disk drive for the data or instructions only 0.000005% of the time
  - 4 GB hard disk (virtual memory) with a 16 ms latency (random access time)
- Apparent access time:
  - (0.95 x 3 ns) + (0.04999995 x 50 ns) + (5E-8 x 16 ms) = 6.15 ns
- System cost:
  - SRAM cache adds ~$70 to cost of processor chip
  - DRAM cost = ~$380 at $6/MB
  - 4 GB Hard disk cost = ~$200
  - Total cost to system = ~$650
- By all appearances, a very large, very fast, and cheap memory system!

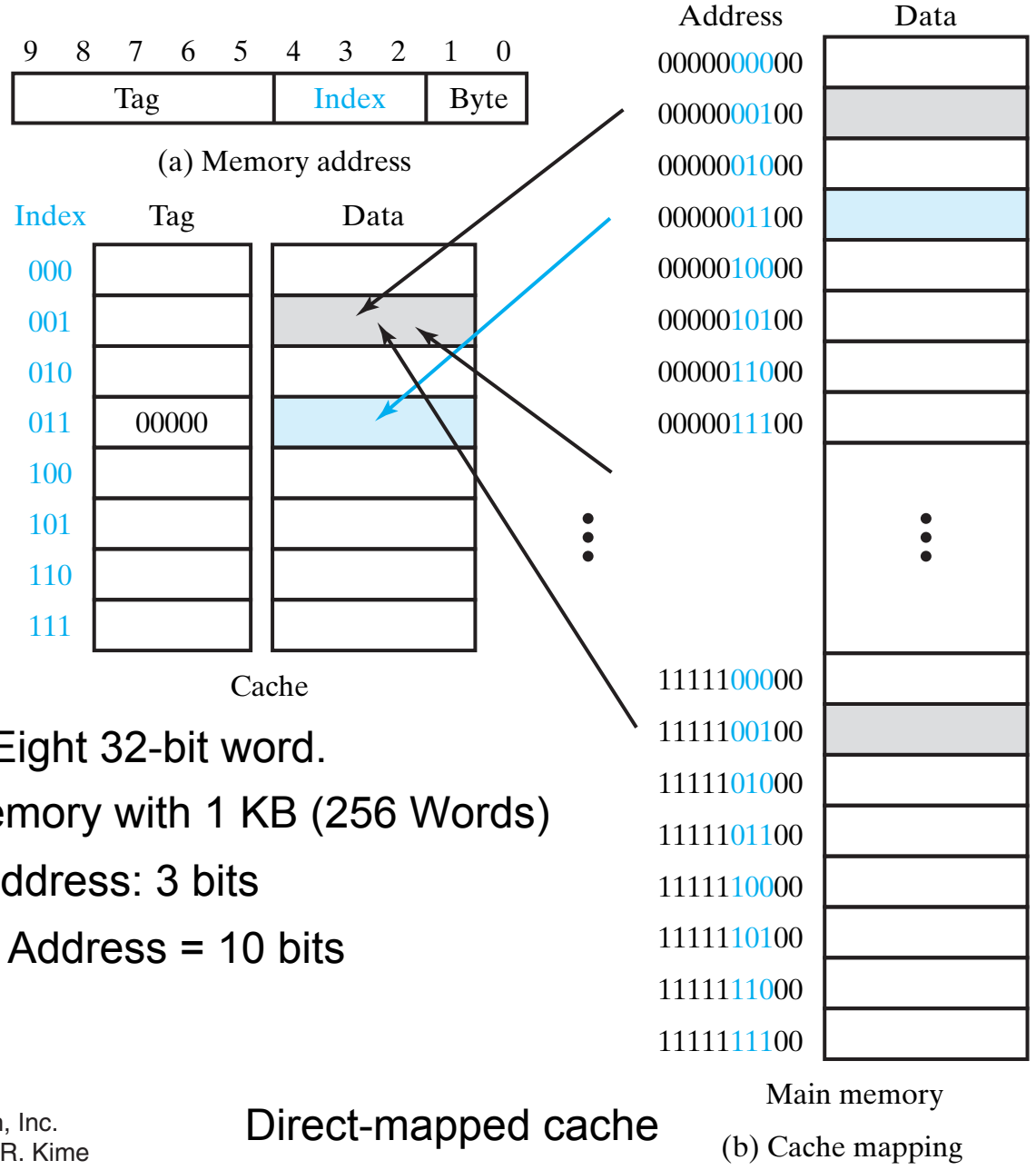
Why it works: Locality of Reference

- **temporal locality**
  - programs tend to contain loops (often nested loops) where an instruction and/or data are accessed many times in sequence
- **spatial locality**
  - instructions and/or data that are stored in contiguous (neighboring) locations are often repeatedly accessed for reading or writing in a typical program
- memory hierarchy makes use of temporal and spatial locality by transferring at one time a group of instructions/data into cache or into main memory
  - A group of instructions or data transferred from main memory into cache is called a ***line*** of data (say 32 bytes)
  - A group of instructions or data transferred from disk storage into main memory is called a ***page*** of data (say 4K bytes)
    - **Virtual memory** = the appearance that all 4GB addressable memory resides in main memory
- Studies of the execution of computer programs have demonstrated the importance of locality of reference in designing a hierarchical memory system.
  - Temporal and spatial locality allow us to achieve a **near infinite cache** in practice for the operation of most computer programs!
- **thrashing** = phenomenon of frequent disk accesses due to a particular program perhaps accessing a database which does not fit entirely into main memory
  - Solution: need a larger main memory!

Cache Memory Organization

- Cache organization schemes:
  - **direct mapped**
  - **fully associative**
  - **set-associative**
- **Line**: A block of data transferred into cache at a given time (4B in text illustrations)
  - the memory address is comprised of 5 bit tag, 3 bit index, and 2 bit byte fields
  - the cache stores both the data (line) as well as the main memory address (tag) of the data
- Hit and Miss
  - When CPU requests data from cache, the address of requested data is compared with addresses of data in cache. If both tag and index addresses match (called a **cache hit**), the requested data is present in cache
    - data word (or byte) is transferred to CPU
  - If the address of requested data does not match tag plus index address of data present in cache, the cache signals the CPU that a **cache miss** has occurred.
    - Main memory transfers a new line (containing the requested data word) into the cache and also sends the requested word (or byte) along to the CPU
  - When a cache miss occurs and a new line of data is to be transferred in from main memory, the cache is likely already full of existing data lines so that one of them needs to be replaced with the new data line. If the line to be replaced has been changed since it was brought in from main memory, it must be written back into main memory first.

Cache Memory Organization



- Cache: Eight 32-bit word.
- Main memory with 1 KB (256 Words)
- Cache address: 3 bits
- Memory Address = 10 bits

Direct-mapped cache

Virtual Memory Concept and Implementation

- Virtual Memory is large memory storage addressable by 32 bit (or higher) memory address but beyond the size capability of the physical address space in main memory
  - desire that virtual memory appears to the CPU to be main memory
    - addressability
    - average latency not to exceed main memory access time by very much
  - each program sees a memory space equal to the virtual address space
    - the hardware must map each virtual address into a physical address in main memory
  - Virtual address space from different programs may map to the same physical address space
    - Allows code and data to be shared across multiple programs
  - normally implemented with hard disk (or tape drive) storage
- **Pages**: Blocks of addresses in virtual memory which map into physical page frames of identical size in main memory
    - analogous to "line" in main memory/cache hierarchy