Operating Systems!

Input/Output (I/O):
An Overview of the OS and I/O
(Part 1)

Prof. Travis Peters

Montana State University

CS 460 - Operating Systems

Fall 2020

https://www.cs.montana.edu/cs460

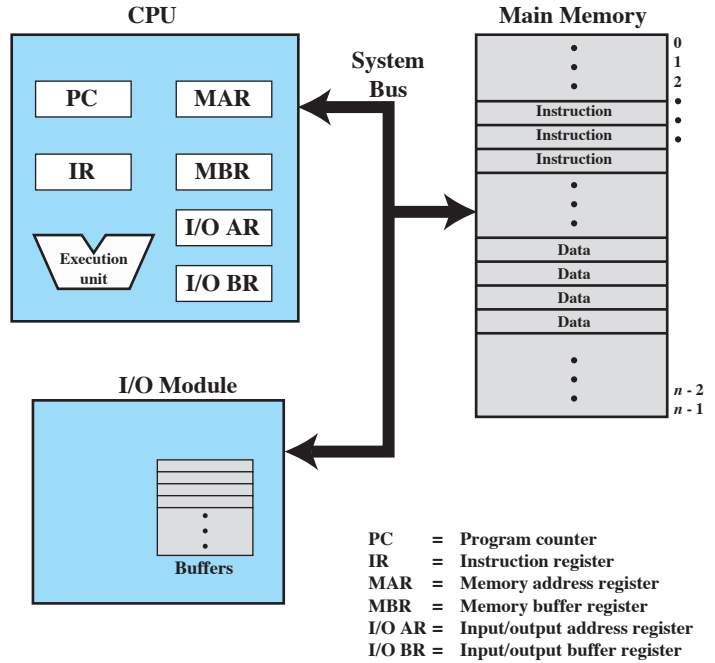
Some diagrams and notes used in this slide deck have been adapted from Sean Smith's OS courses @ Dartmouth. Thanks, Sean!

Today

- Announcements
  - Congrats on submitting your proposals! ☺
- Upcoming Deadlines
  - **Project Submission (HARD DEADLINE)**
    Sunday [11/15/2020] @ 11:59 PM (MST)
  - **Project Evaluations (HARD DEADLINE)**
    Wednesday [11/18/2020] @ 11:59 PM (MST)

Today *(cont.)*

- Agenda
  - Key concepts behind I/O
    - *I/O Categories. Memory access. What is I/O? What is a file? What is a file system? etc.*
  - Files
    - Operations
    - Implementations
  - File System
    - Virtual File Systems
    - Implementations
  - Secondary Storage
    - Emphasis on **disks** and **disk scheduling**
  - Security



What is I/O?

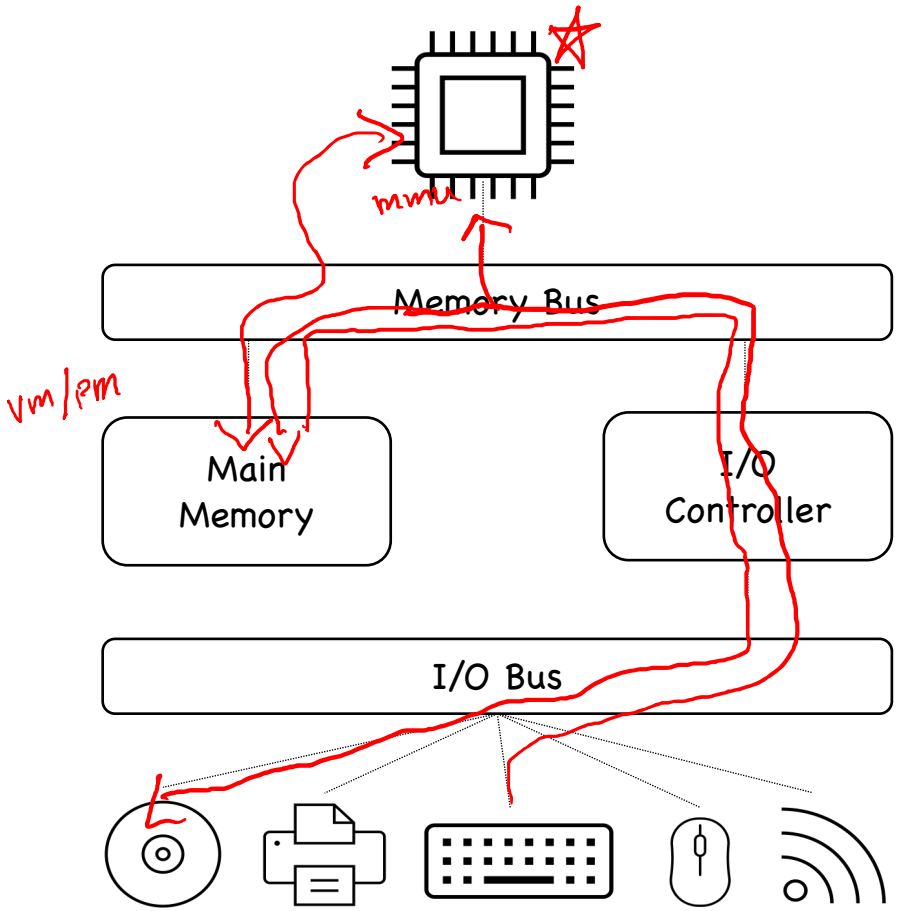
Categories of I/O Devices

I/O is how the system talks to the outside world!

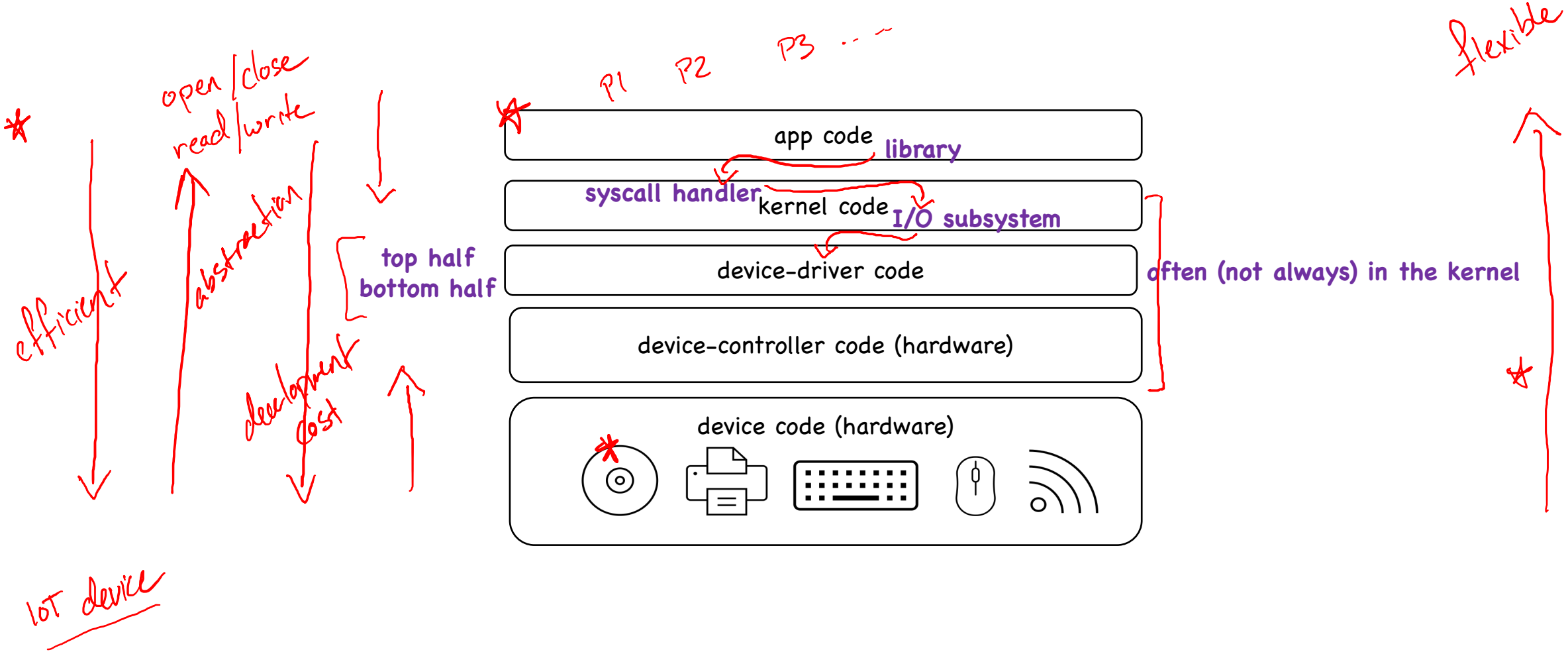
- **Human Readable**
  Computer-User Interactions
  (e.g., printers, terminals, keyboards, mice)

- **Machine Readable**
  Computer-Device Interactions
  (e.g., disk drives, controllers, sensors, actuators)

- **Communication**
  Computer-Computer Interactions
  (e.g., network devices, modems)



The Big Picture



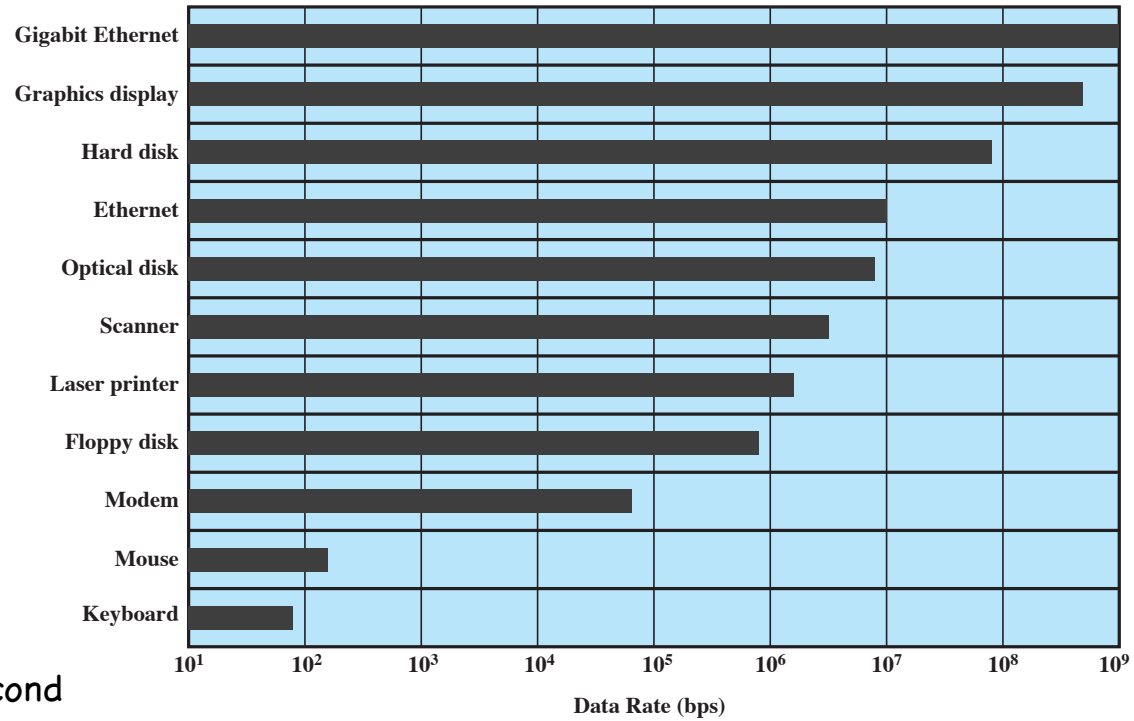
I/O: Overarching Objectives

Objective: *Efficiency*

Modern CPUs: many BILLIONS of instructions per second.
Modern memory systems: 2-4 GB/sec bandwidth

I/O bottleneck

ex: 10s-100s Mbits/sec



ex: 9-10 keystrokes per second

Figure 11.1 Typical I/O Device Data Rates

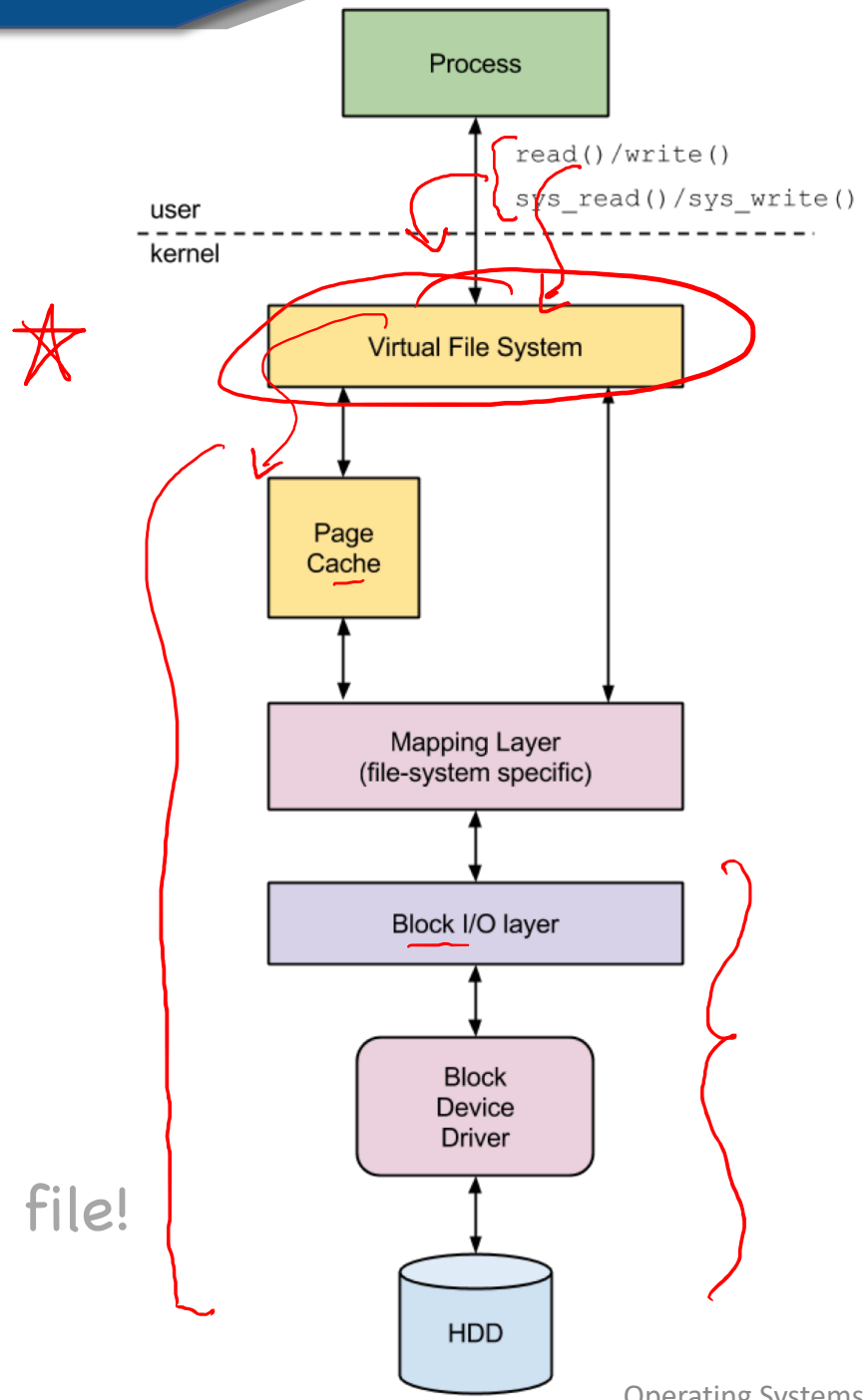
Objective: *Generality*

want:

- simplicity
- correctness
- support for diverse range of devices

-> Lots of abstractions (read/write, open/close, etc.)

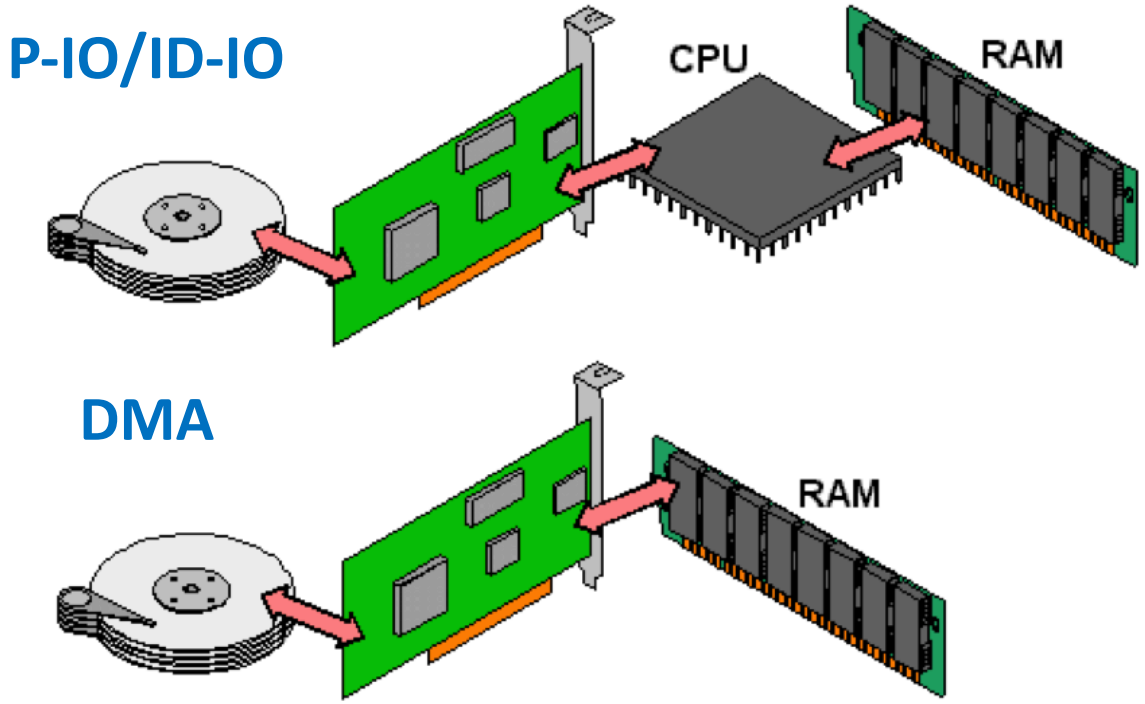
NOTE:
In UNIX, each I/O device is associated with a special file managed by the FS, which enables programs/users to interact with devices like any other file!



I/O: How?

Recall: Techniques for I/O & Memory Access

- Programmed I/O
  (*active*; processor polls I/O device)
- Interrupt Driven I/O
  (*assist*; I/O device does work; interrupt processor for help)
- DMA
  (*delegate*; DMA module give OP, DEV, ADDR, #WORDS; send interrupt when done)



Direct Memory Access

Role of DMA: transfer data to/from memory over system bus

< op, dev, addr, n >

-> most efficient to have all I/O done via DMA on dedicated I/O bus

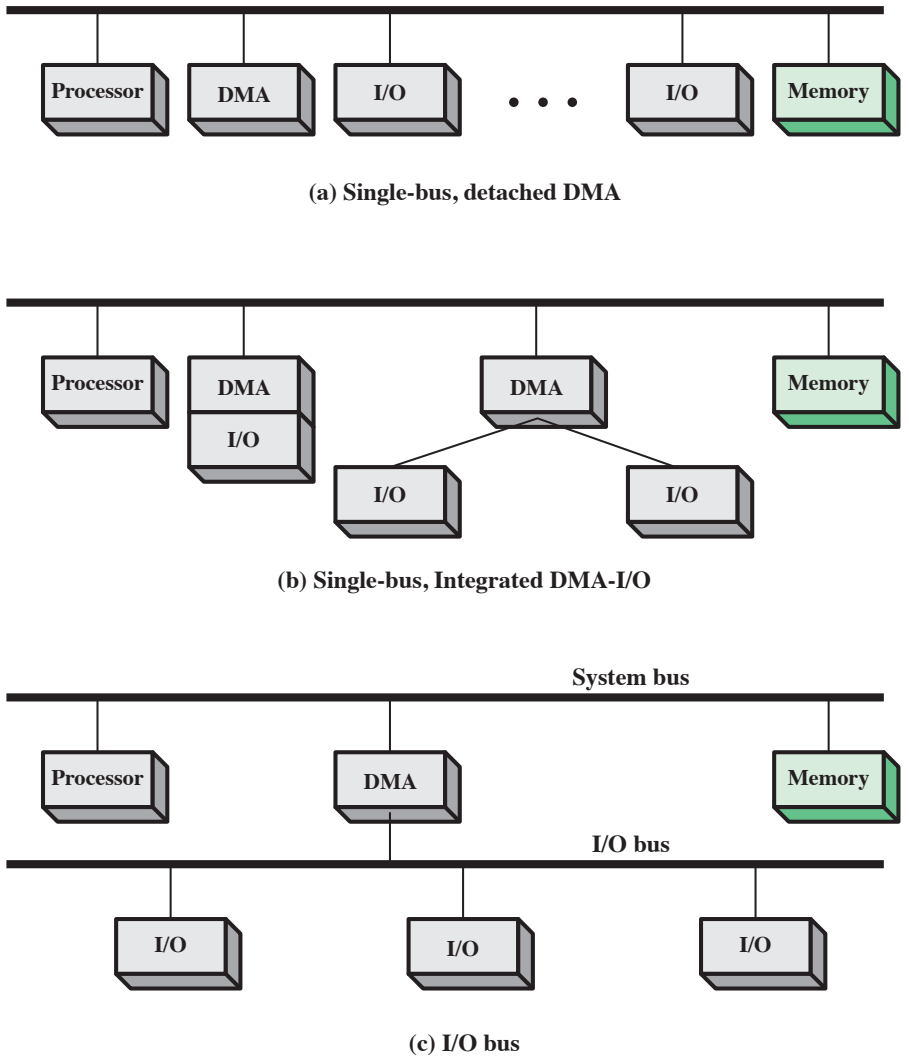


Figure 11.3 Alternative DMA Configurations