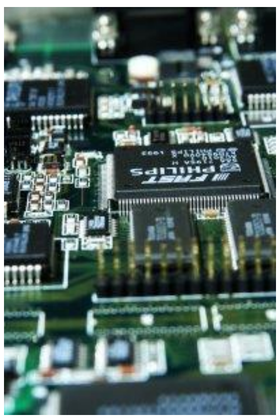
CSE 321b

Computer Organization (2)

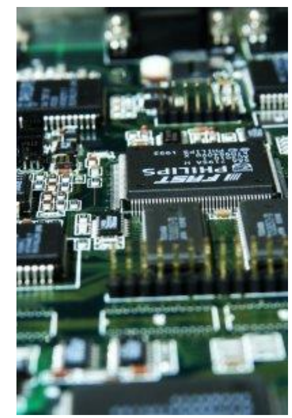
تنظيم الحاسب (2)



3rd year, Computer Engineering

Winter 2016

Lecture #7



Dr. Hazem Ibrahim Shehata

Dept. of Computer & Systems Engineering

Credits to Dr. Ahmed Abdul-Monem Ahmed for the slides

Adminstrivia

- Assignment #2:
  - To be released later on this weekend.
- Midterm:
  - Date: Thursday, April 21, 2014
  - Time: 10:30am – 12:00pm
  - Location: classrooms #27321 (قاعة 4د) & #27309
  - Coverage: lectures #1 ➔ #7

Website: http://hshehata.github.io/courses/zu/cse321b/
Office hours: Sunday 11:30am – 12:30pm

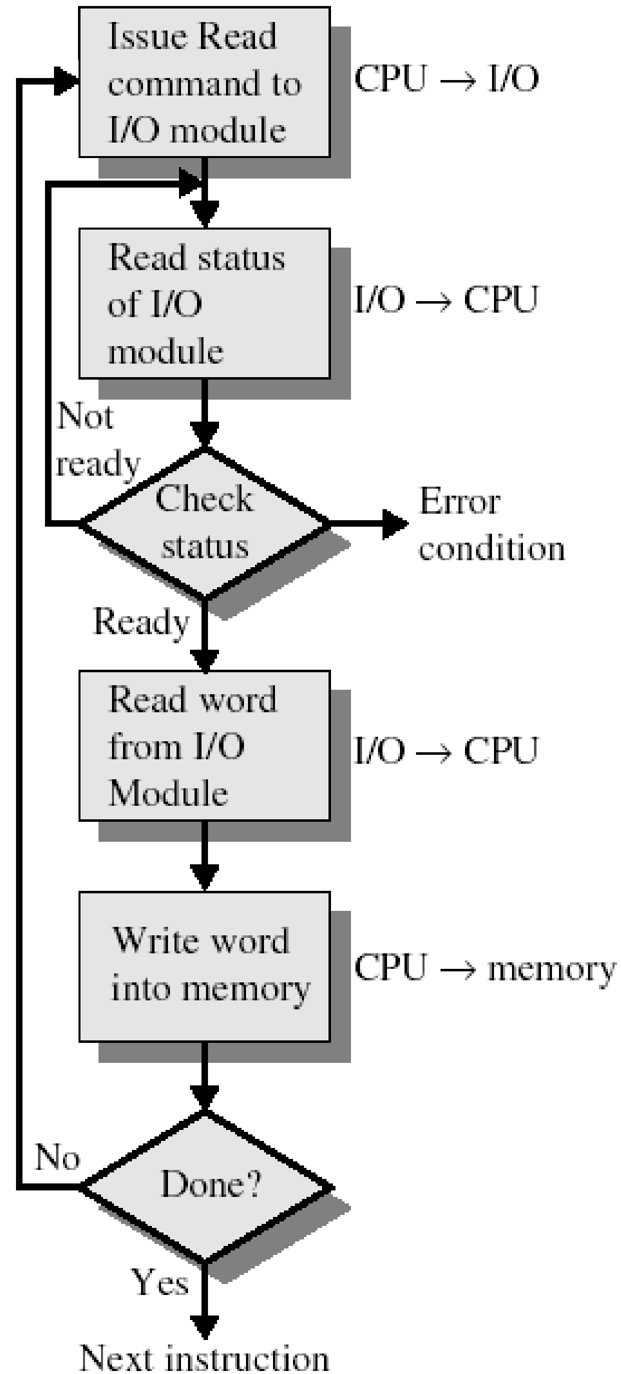
Chapter 7. Input / Output (*Cont.*)

Outline

- External Devices
  - Types
  - Structure
- I/O Modules
  - Function
  - Structure
- I/O Techniques
  - Programmed I/O
  - Interrupt-Driven I/O
  - Direct Memory Access
- I/O Channels & Processors

Programmed I/O

- CPU issues read/write command to I/O module.
- CPU periodically checks status bits.
  - —CPU waits for command to complete.
  - —I/O module does not interrupt CPU.
- CPU reads data from device/memory and writes it to memory/device.

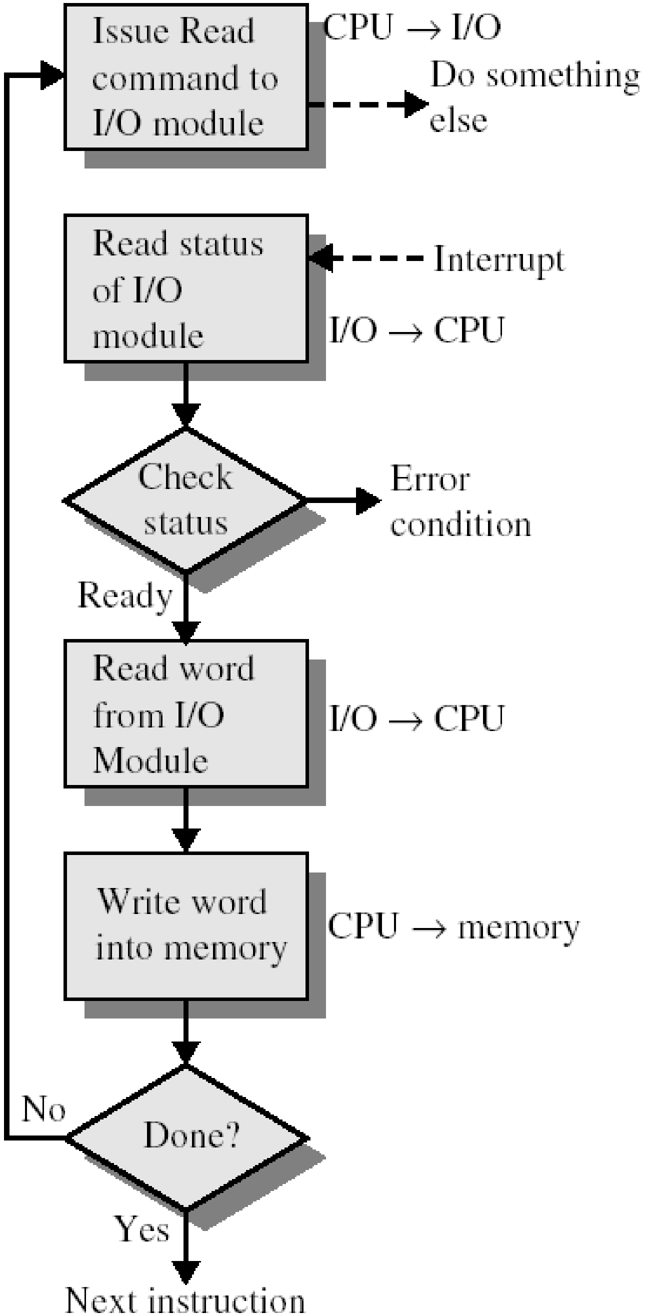


Interrupt-Driven I/O

- Purpose: To overcome CPU waiting.
- No repeated CPU checking of device.
- CPU issues command and moves on to do other useful work.
- I/O module interrupts CPU when ready.

Interrupt-Driven I/O

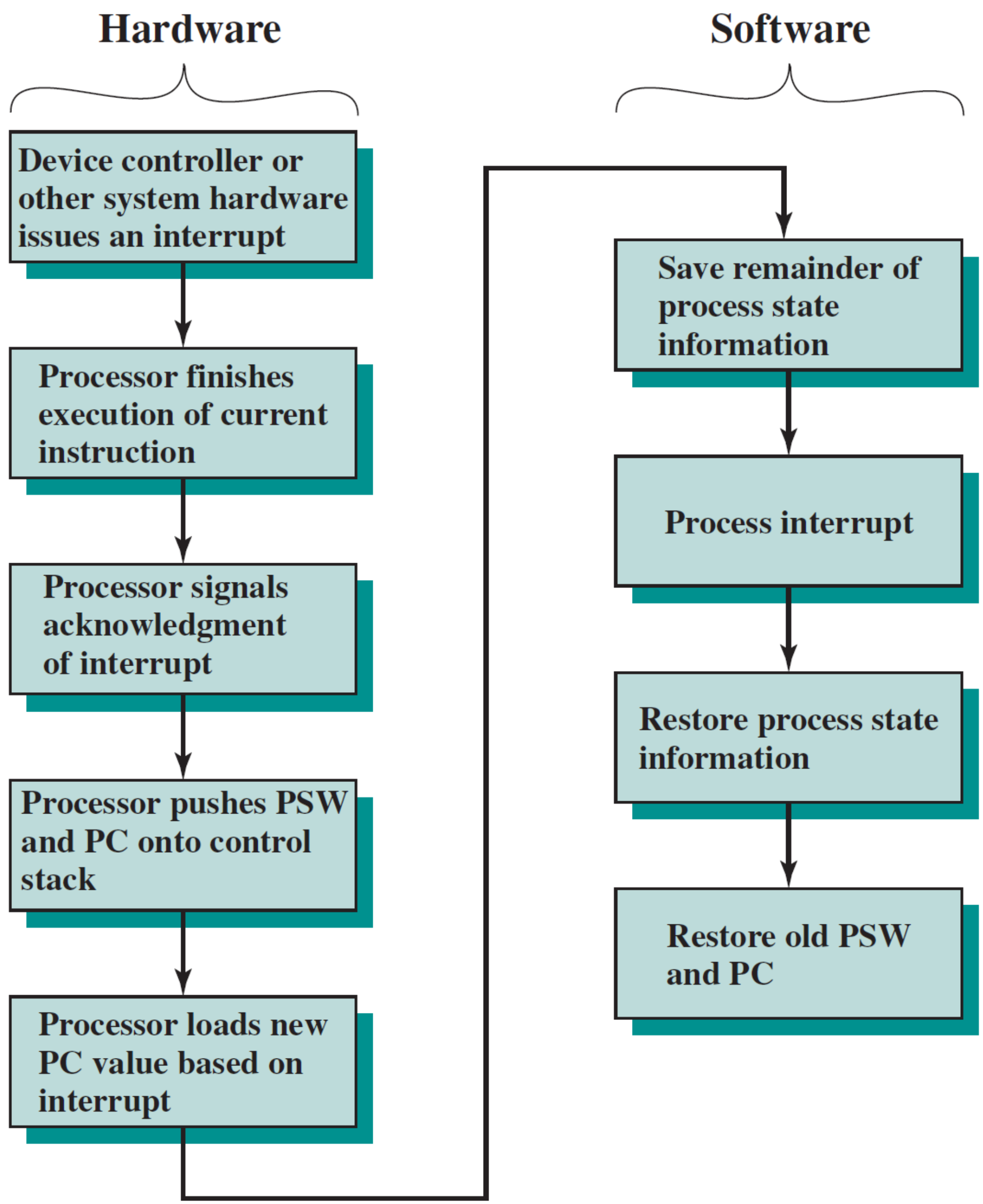
- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work.
- I/O module interrupts CPU.
- CPU reads data from I/O module.
- CPU writes data to memory.



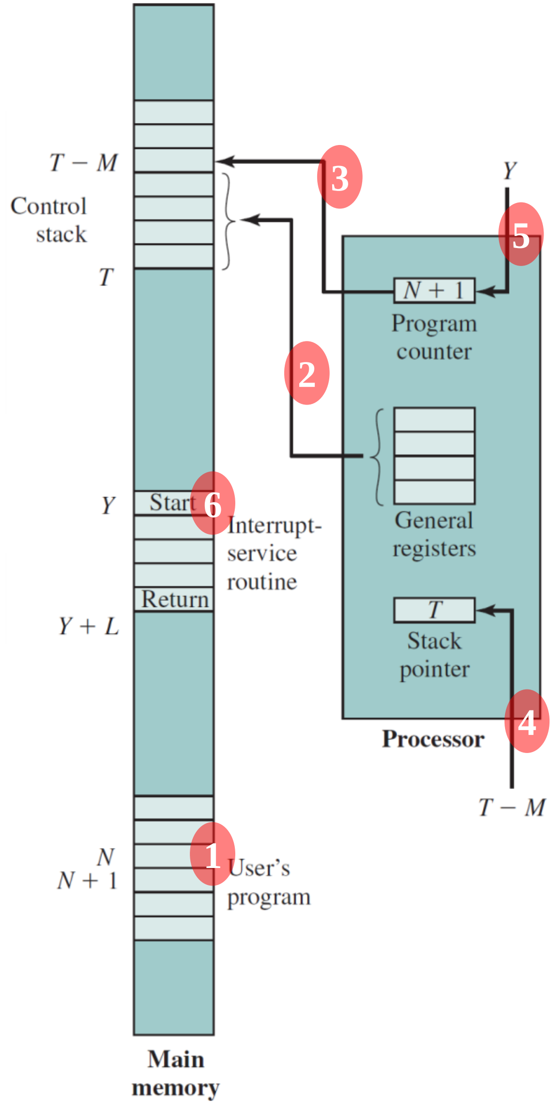
CPU Viewpoint

- Issue read command.
- Do other work.
- Check for interrupt at end of each instruction cycle.
- If interrupted:
    - Save context (registers).
    - Process interrupt.
        - Fetch data (from module) & store (to memory)

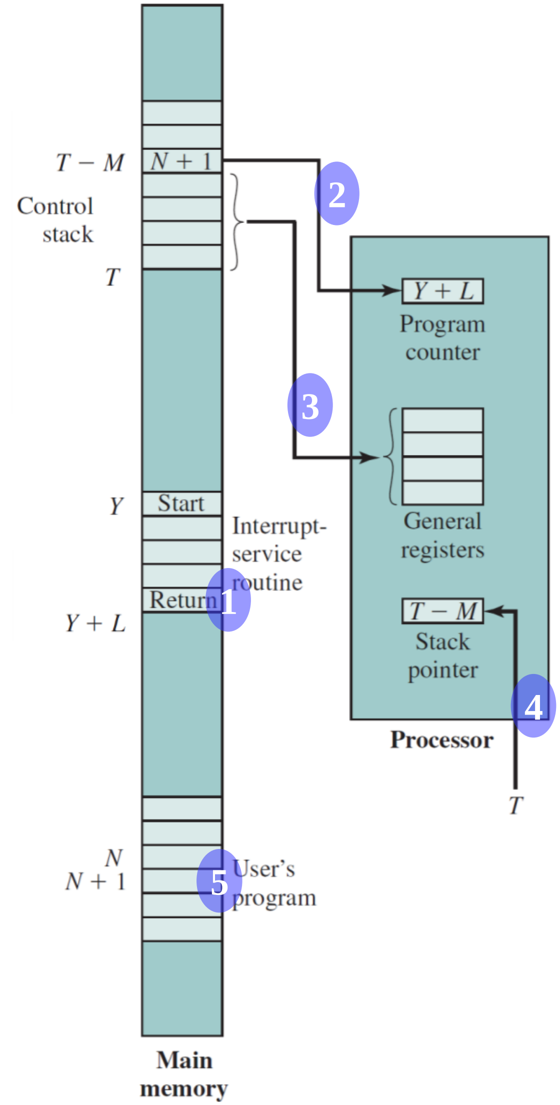
Simple Interrupt Processing



Changes in Mem. & Reg.'s upon Interrupt



(a) Interrupt occurs after instruction at location N



(b) Return from interrupt

Identifying Interrupting Module

- How to identify the module issuing the interrupt?
  1. Different line for each module
     - Limits number of devices.
  2. Software poll (single line)
     - CPU asks each module in turn ➔ time consuming!!
     - e.g., Send TESTI/O ➔ Set address lines ➔ Check status reg.
  3. Daisy chain or Hardware poll (single line)
     - All modules share a single interrupt request line.
     - Interrupt acknowledge sent down a chain.
     - Module (that issued interrupt signal) places its vector on bus.
     - CPU uses vector to identify handler routine.
  4. Bus Master (single line)
     - Module must claim the bus before it can raise interrupt.
     - e.g., PCI & SCSI.

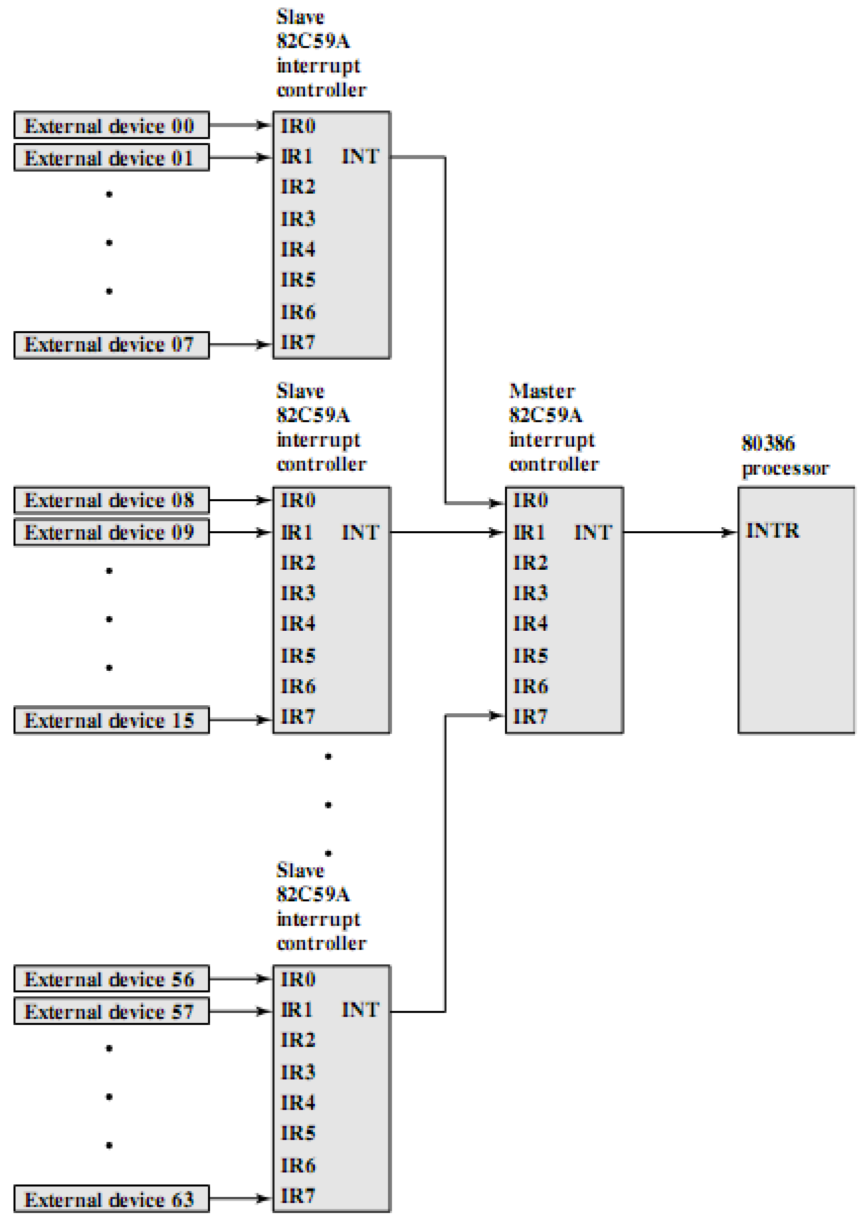
Multiple Interrupts & Priorities

- How to deal with simultaneous interrupts?
- Solution: prioritize interrupts!

1. With multiple lines:
   - Each interrupt line has a priority.
   - Higher priority lines can interrupt lower priority lines.
2. With software polling:
   - Priority determined by order in which modules are polled.
3. With daisy chain:
   - Priority determined by order of modules on chain.
   - Closer modules have higher priority.
4. With bus arbitration:
   - Only current master can interrupt.
   - Priority is defined by the bus arbitration protocol.

Example - PC Bus

- 80386 has one interrupt line!
- To handle more interrupts, connect 1 (or more) interrupt arbiter ➔ Intel 8259.
- Intel 8259 has 8 intrpt. lines.
- Sequence of events:
  - 8259 accepts interrupts.
  - 8259 determines priority.
  - 8259 signals 8086 (raises INTR line).
  - CPU Acknowledges.
  - 8259 puts correct vector on data bus.
  - CPU processes interrupt.

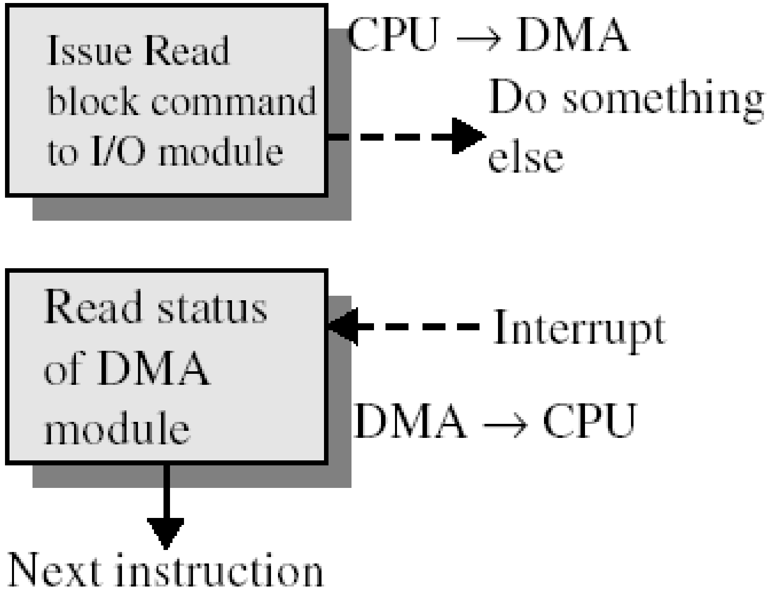


Direct Memory Access (DMA)

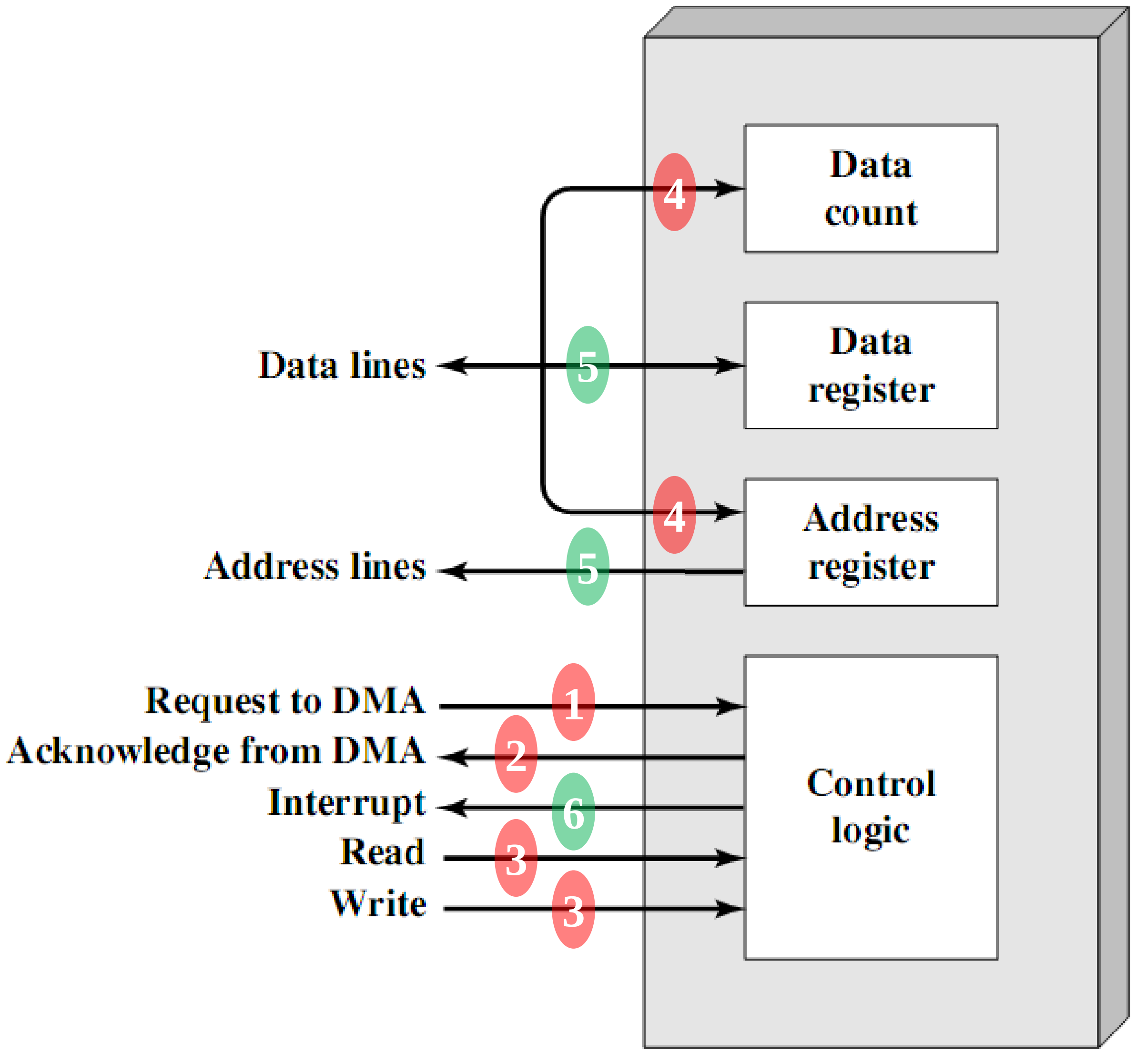
- Interrupt driven and programmed I/O require active CPU intervention.
  - CPU tests and services a device.
    - Transfer rate is limited (depending on CPU availability)!!
  - Many instructions are executed for every I/O.
    - CPU is tied up in managing an I/O transfer!!
- DMA is a more efficient technique (when transferring large volumes of data, i.e., blocks).
  - Additional module on the system bus ➔ DMA controller.
  - DMA controller mimics CPU and takes over the bus to transfer the data with no CPU intervention!

DMA Operation

- CPU tells DMA controller:
  - —Type of Operation (Rd/Wr).
  - —Address of device.
  - —Starting address of a data block in memory.
  - —Amount of data to be transferred.
- CPU carries on with other work.
- DMA controller performs the transfer.
- DMA controller sends interrupt when finished.



Typical DMA Controller

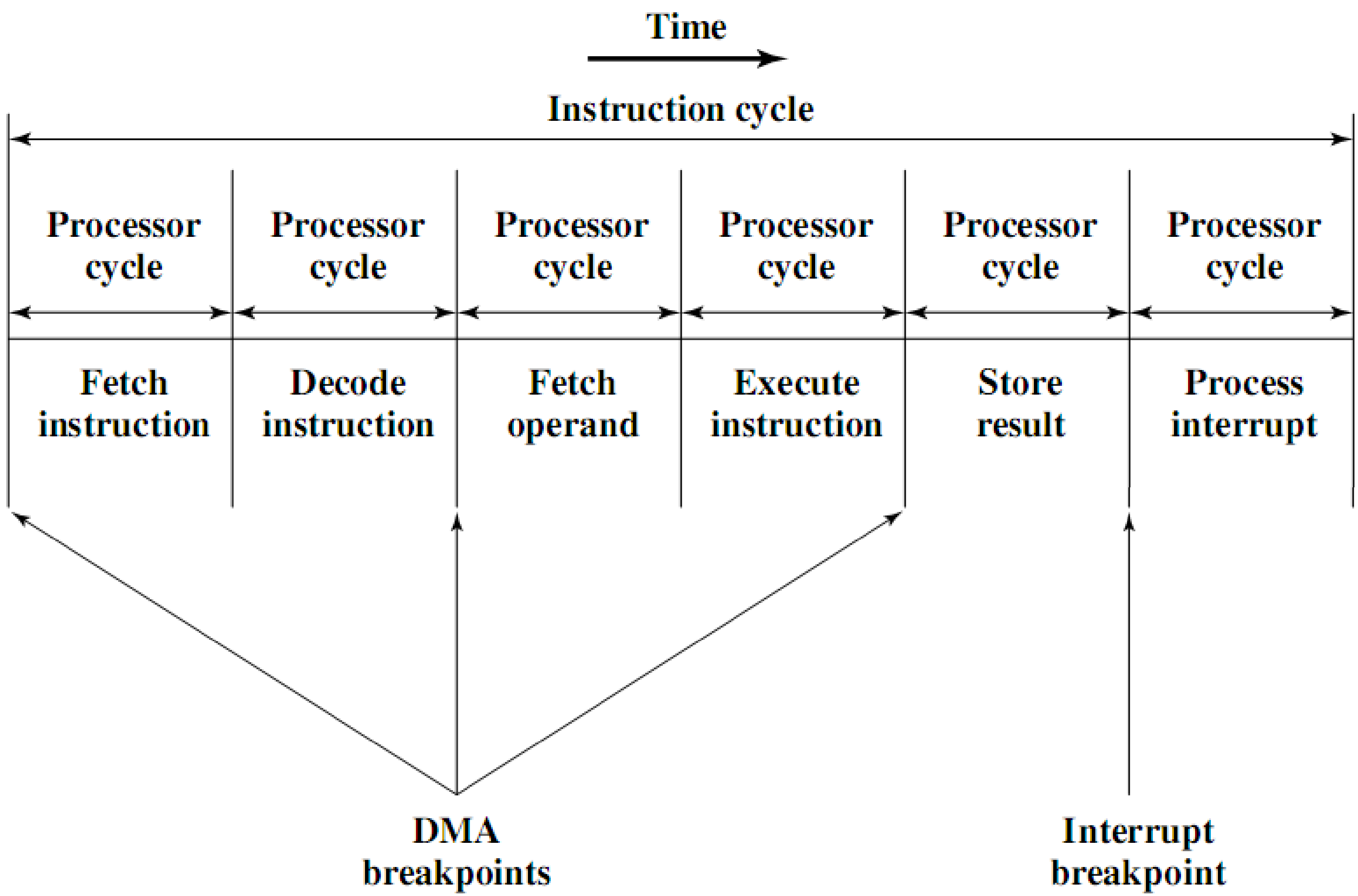


DMA Transfer modes

- DMA controller transfers data from/to MM over the system bus.
- DMA controller takes over bus for a bus cycle to transfer data by one of the following techniques:
  - —Use bus only when CPU not using it: transparent mode.
  - —Force CPU to suspend operation temporarily ➔ DMA steals bus cycles from CPU: cycle stealing mode.
- Notice this is not an interrupt!
  - —CPU does not switch context.
- CPU gets suspended just before it accesses bus.
  - —i.e. before an operand or data fetch or a data write.

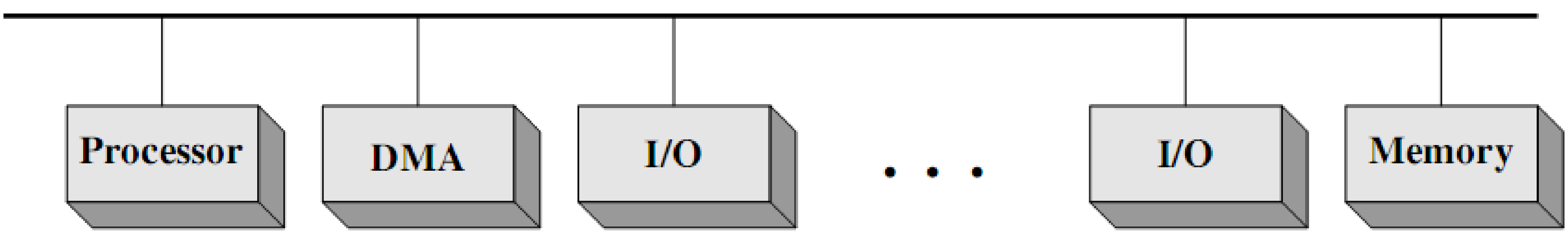
- Slows down CPU. Faster than CPU doing transfer!

DMA and Interrupt Breakpoints During an Instruction Cycle



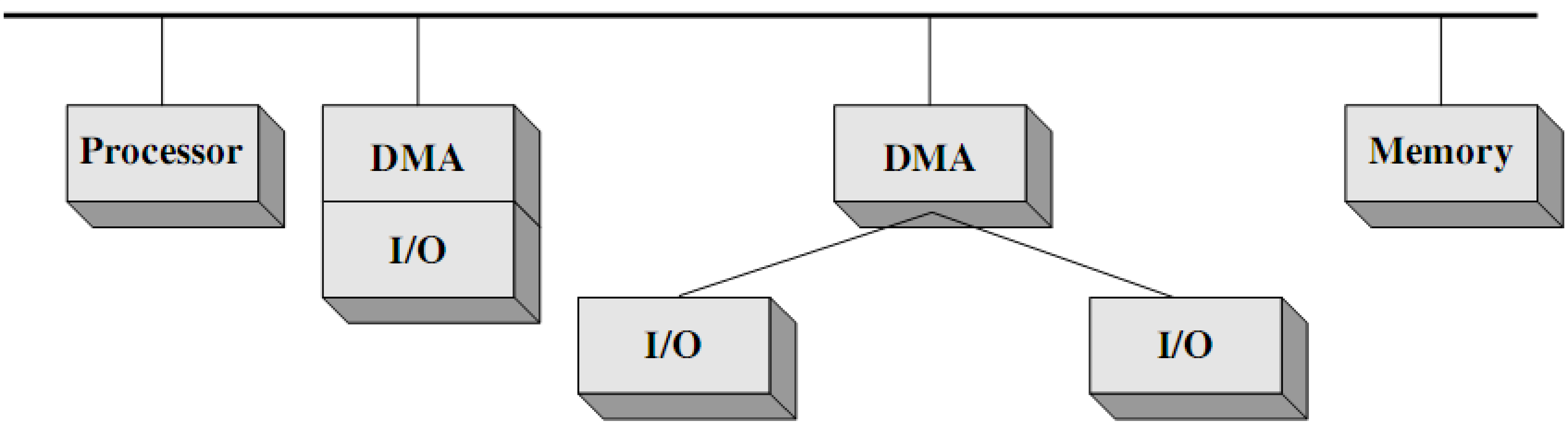
DMA Configurations (1)

- Features: single-bus, detached DMA controller.
- DMA module acts as a surrogate processor.
- DMA module uses programmed I/O.
- Each transfer uses system bus twice.
  - —I/O device → DMA controller → memory.
- CPU is suspended twice per transfer.
- DMA controller has no I/O interfaces.
- Inexpensive yet inefficient!



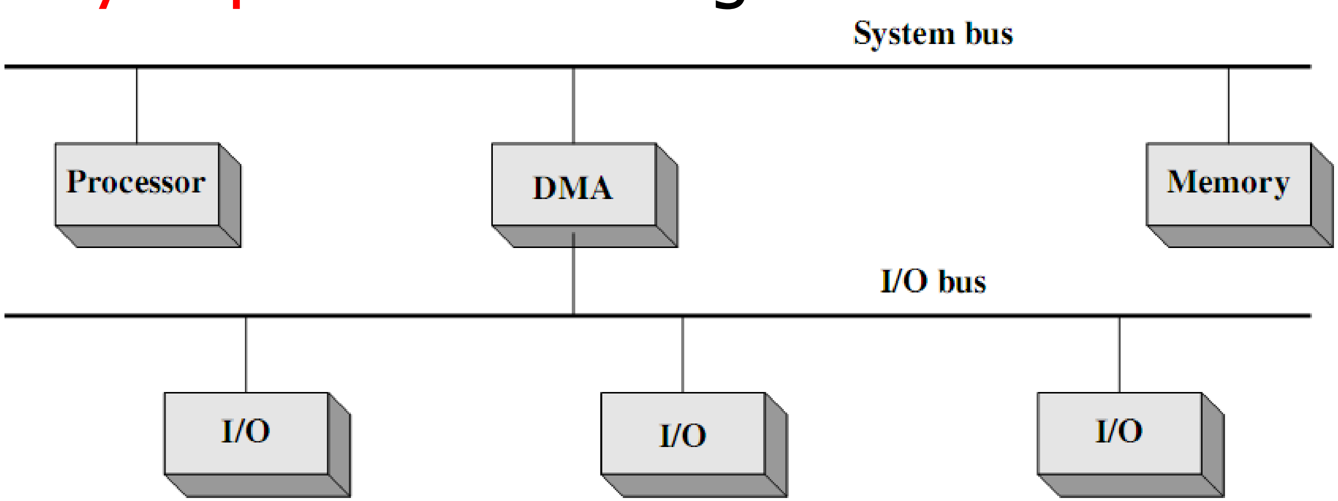
DMA Configurations (2)

- Features: single-bus, integrated DMA controller.
- Controller may support more than one device.
- Each transfer uses system bus once.
  - —DMA controller ➔ memory.
- CPU is suspended once per transfer.
- DMA controller has one or more I/O interfaces.
- Efficient yet expensive!



DMA Configurations (3)

- Features: separate I/O bus.
  - —Connecting all DMA-capable devices.
- Each transfer uses system bus once.
  - —DMA controller ➔ memory
- CPU is suspended once.
- DMA controller has only one I/O interface.
- Easily expandable configuration.



Evolution of I/O function

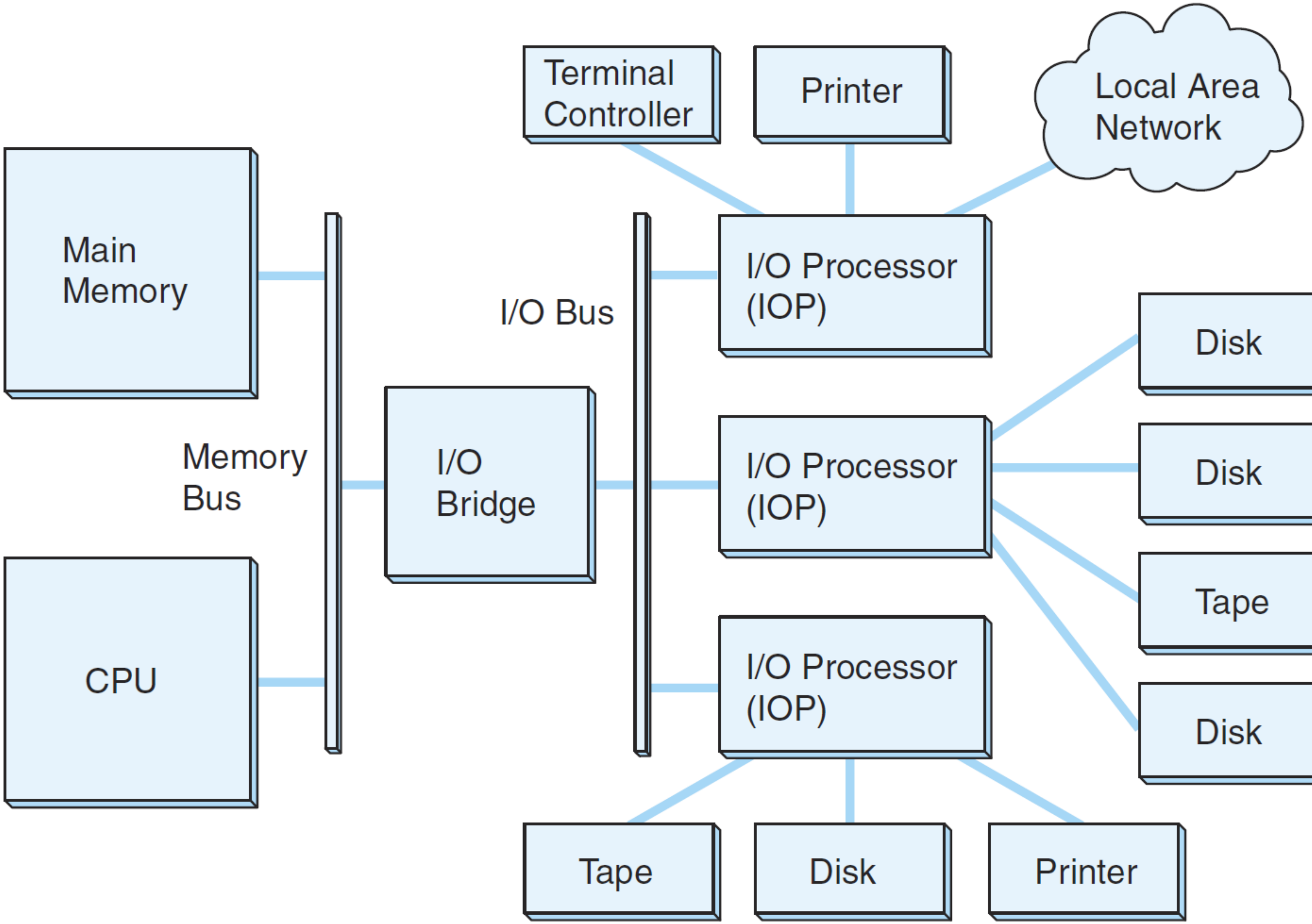
1. No I/O module.
   - CPU directly controls i/o device.
2. I/O module responding to CPU.
   - Programmed I/O.
3. I/O module interrupting CPU.
   - Interrupt-driven I/O.
4. I/O module accessing memory.
   - Direct-memory access (DMA).
5. I/O module executing program.
   - I/O channel.
6. I/O module executing program from local memory.
   - I/O processor, .e.g., GPU.

- **NOTE**: On occasions, no distinction is made between the terms: I/O channel and I/O processer!!!

I/O Channels

- I/O channels:
  - an extension to the DMA concept
  - able to execute I/O instructions
- I/O channel equipped with special-purpose processor, referred to as I/O processor (IOP)!!!!
- CPU instructs I/O channel to do the transfer
  - I/O processor fetches and executes I/O program from memory.
- I/O channel does the entire transfer.
- Improves speed
  - Takes load off CPU.
  - Dedicated processor is faster.

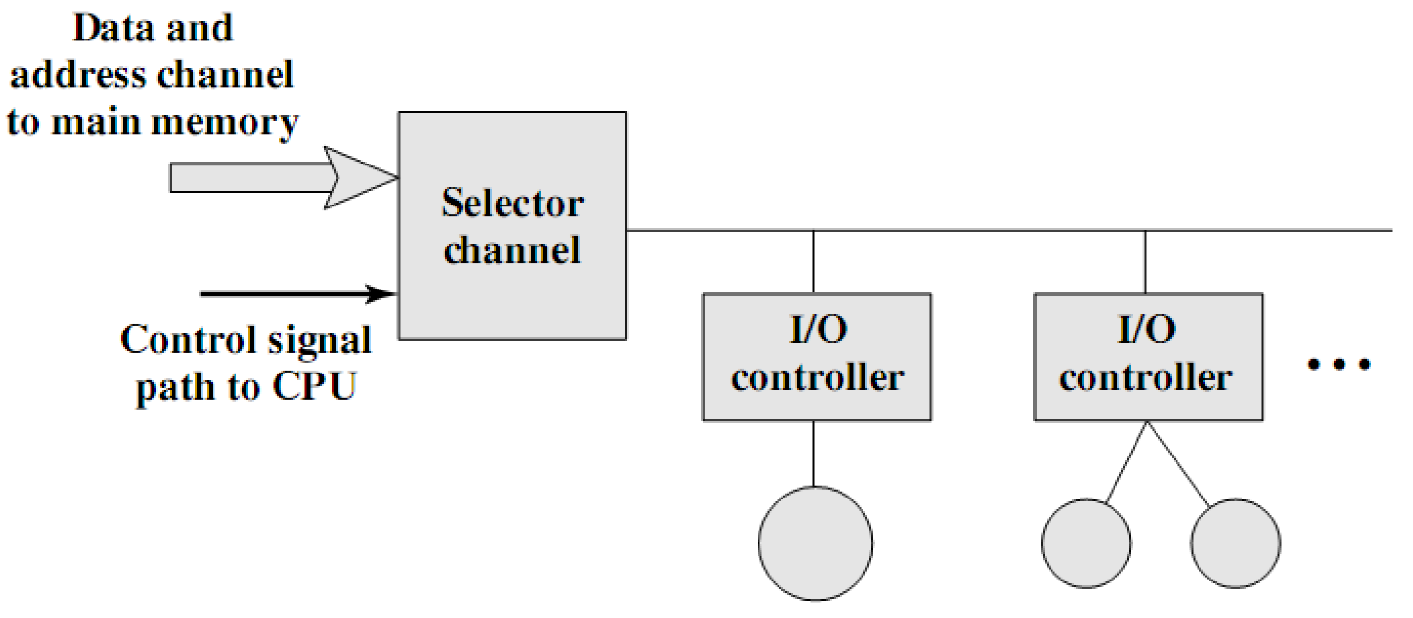
A Channel I/O Configuration



• "The Essentials of Computer Organization and Architecture", Null and Lobur

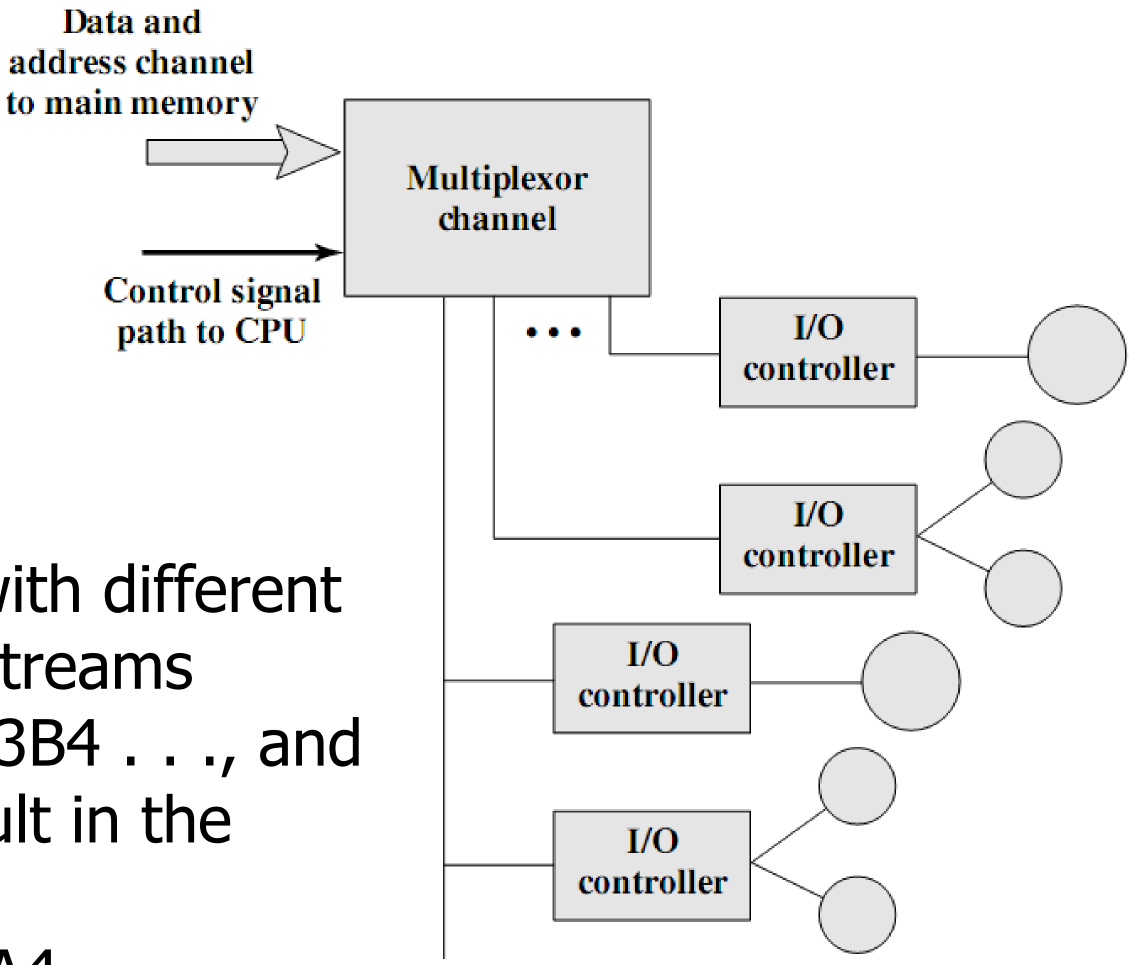
I/O Channel Architecture – Selector

- Channel controls multiple high-speed devices.
- At any given time, channel is dedicated to data transfer with only one of these devices.
- Each device/set of devices is/are handled by a controller (I/O module).



I/O Channel Architecture – Multiplexor

- Handles multiple low-speed devices at the same time.
- A byte multiplexor accepts or transmits characters as fast as possible to multiple devices.
- Example: 3 devices with different rates and individual streams A1A2A3A4 ..., B1B2B3B4 . . ., and C1C2C3C4 might result in the character stream A1B1C1A2C2A3B2C3A4 ...



Reading Material

- Stallings, Chapter 7:
  - Pages 232-237
  - Pages 240-243
  - Pages 246-248