Input / Output

Overview

- I/O module is the third key element of a computer system. (others are CPU and Memory)
- All computer systems must have efficient means to receive input and deliver output
- We will look at
  - I/O module and their interface to the system
  - I/O mechanisms
  - Example interfaces

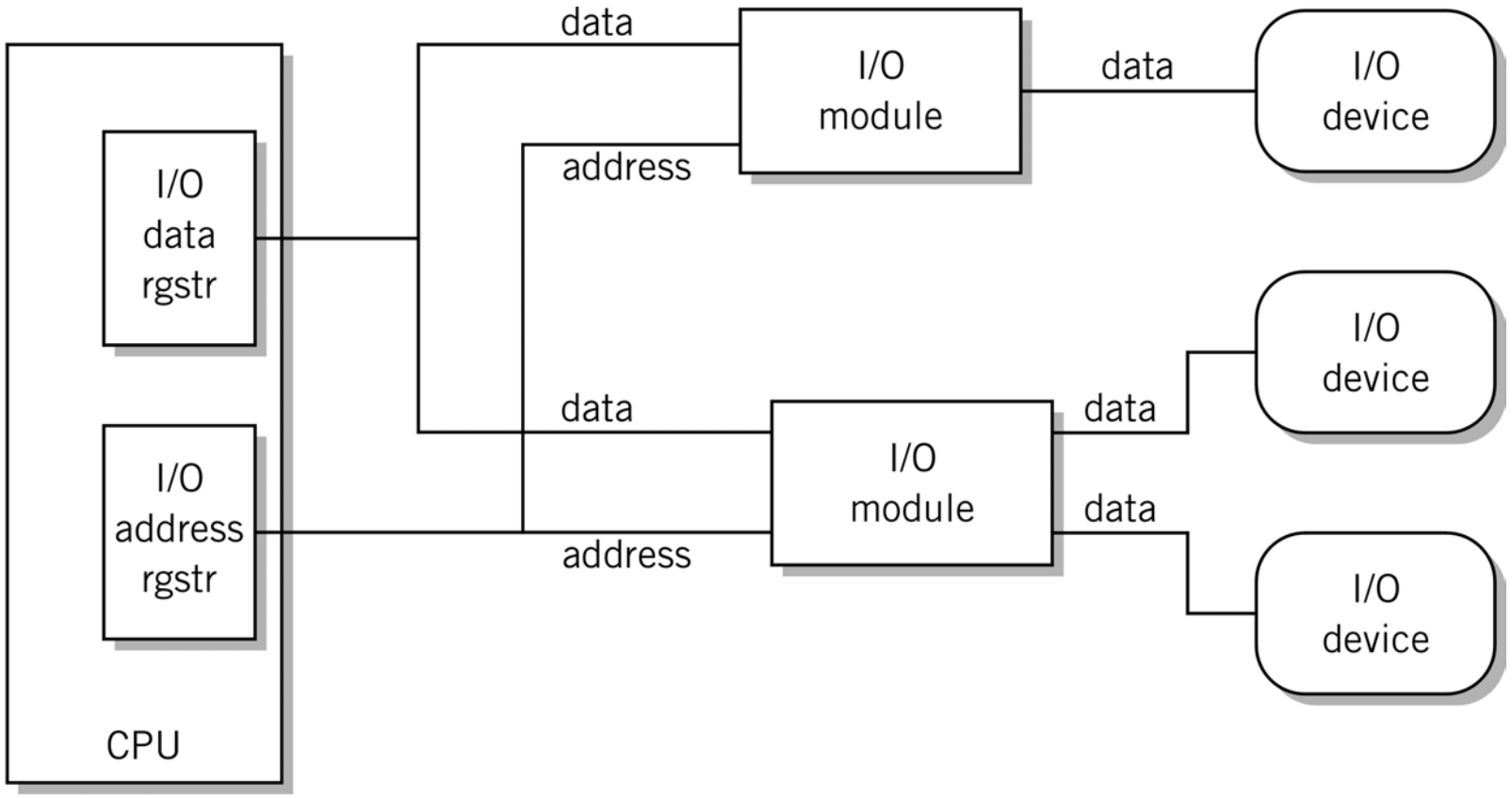
I/O modules

- Each I/O module interfaces to the system bus and controls one or more peripheral devices.

- External devices are generally not connected directly to the bus structure of the computer systems -

  - Wide variety of devices require different logic interfaces - impractical to expect the CPU to know how to control each device
  - Mismatch of data rates
  - Different data representation

I/O modules



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-03

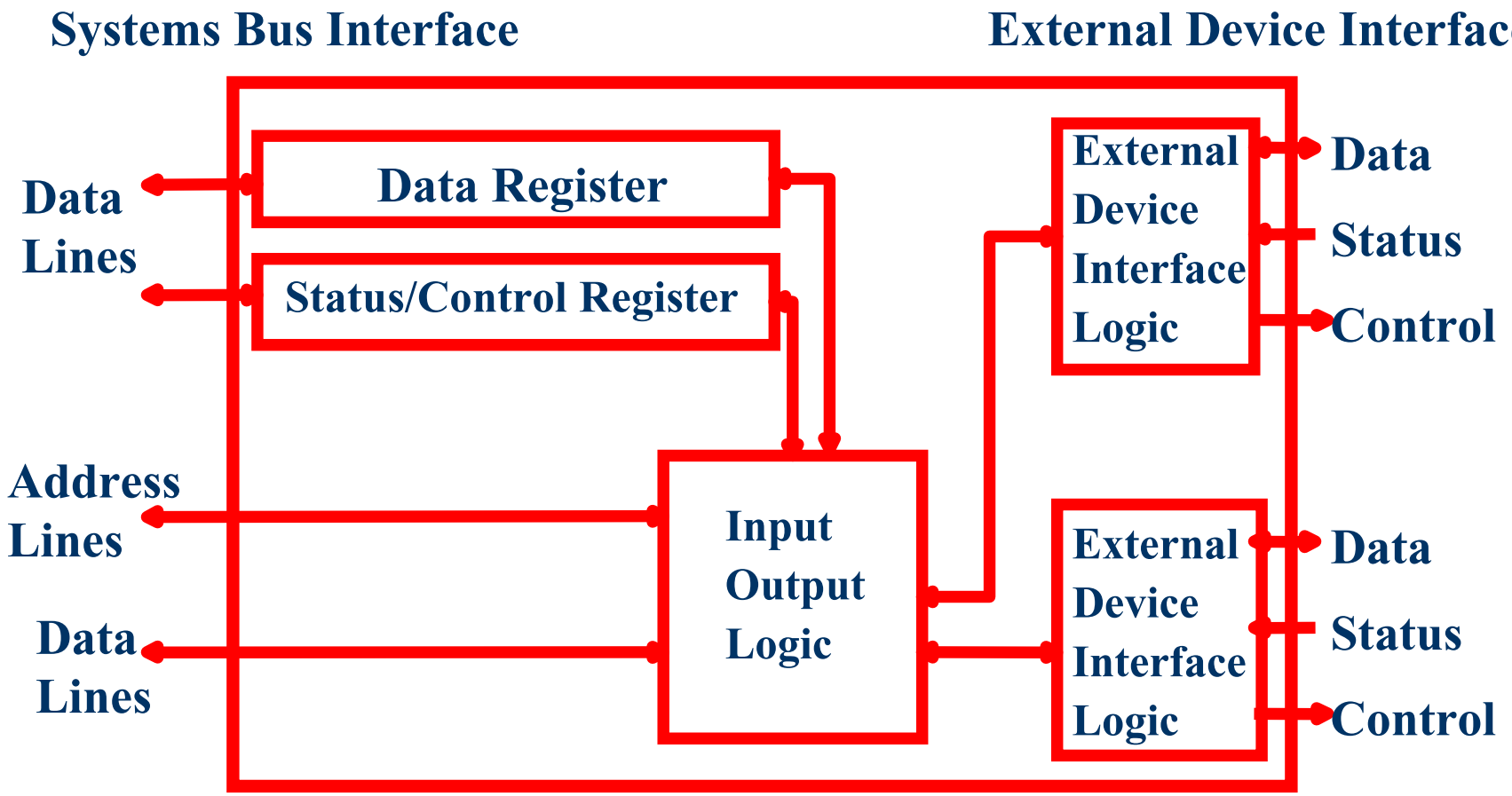
I/O modules

- The I/O modules

  - Not just simple mechanical connectors
  - Contain "intelligence" - logic for performing communication functions between the peripherals and the bus.
  - Provide standard interfaces to the CPU and the bus
  - Tailored to specific I/O devices and their interfaces requirement
  - Relieve CPU of the the management of I/O devices
  - Interfaces consist of

    - Control
    - Status and
    - Data signals

I/O Module Diagram



Input Output Techniques: Programmed

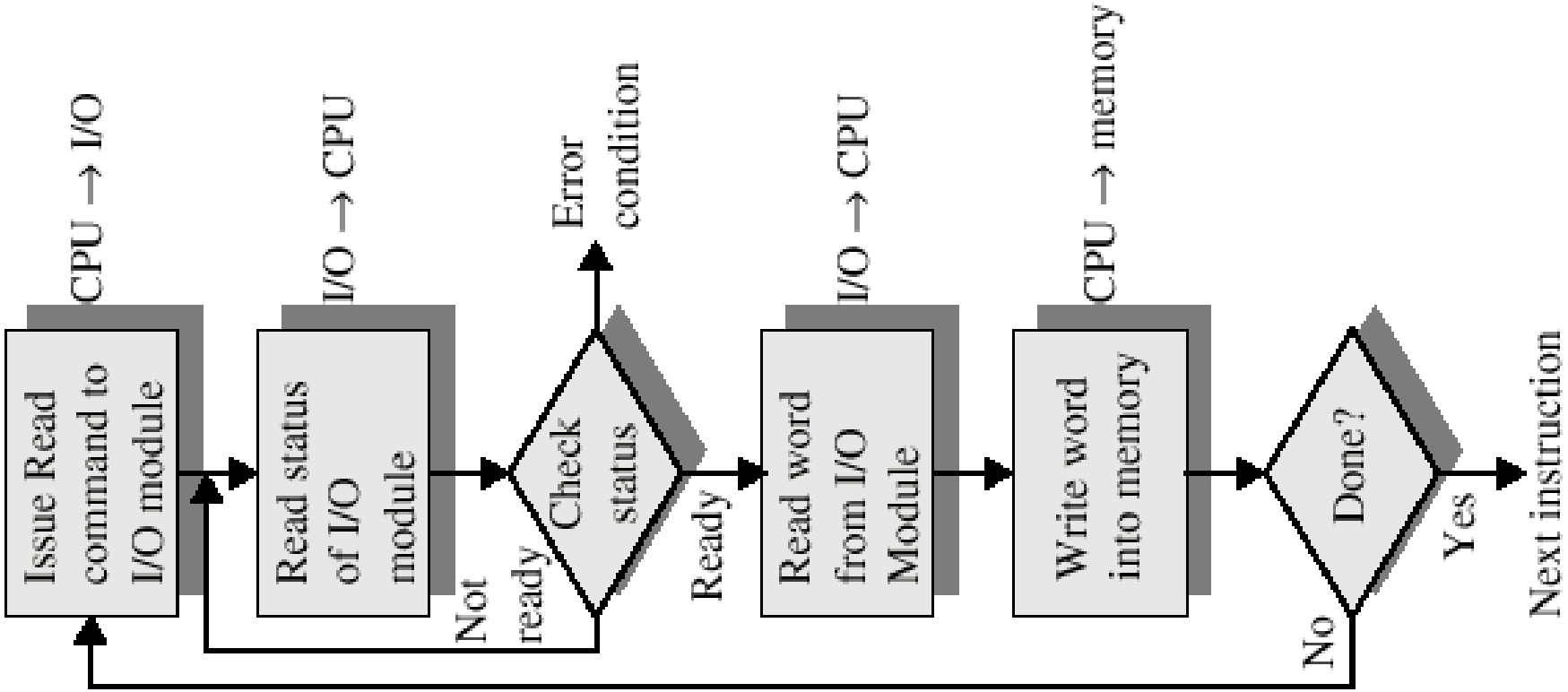
- I/O operation in which the CPU issues the I/O command to the I/O module

- CPU is in direct control of the operation
  - Sensing status,
  - Read/write commands,
  - Transferring data

Input Output Techniques: Programmed

- CPU waits until the I/O operation is completed before it can perform other tasks
- Completion indicated by a change in the status bits
- CPU must periodically poll the module to check its status bits
- The speed of the CPU and peripherals can differ by orders of magnitude, programmed I/O waste huge amount of CPU power
- Very inefficient
- CPU slowed to the speed of peripherals

Input Output Techniques: Programmed



¶Programmed I/O Operation

¶Simple to implement. Requires very little special software or hardware

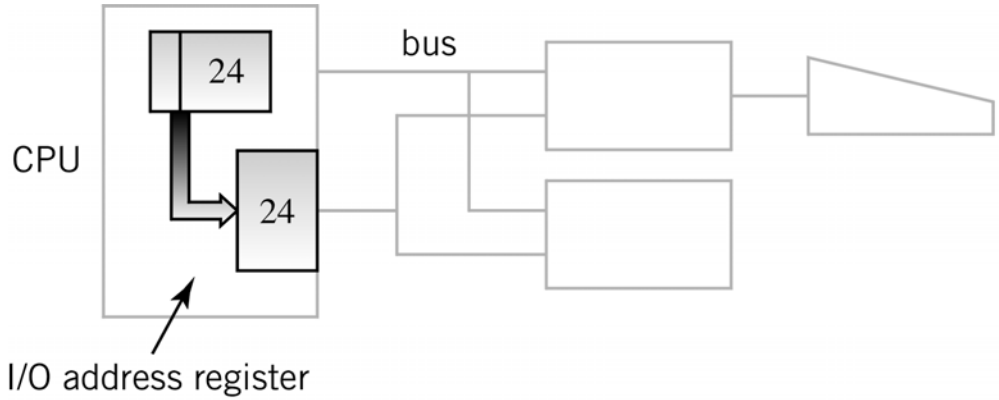
Input Output Techniques: Programmed

- Addressing I/O Devices
  - Under programmed I/O data transfer is very like memory access (CPU viewpoint)
  - Each device given unique identifier
  - CPU commands contain identifier (address)

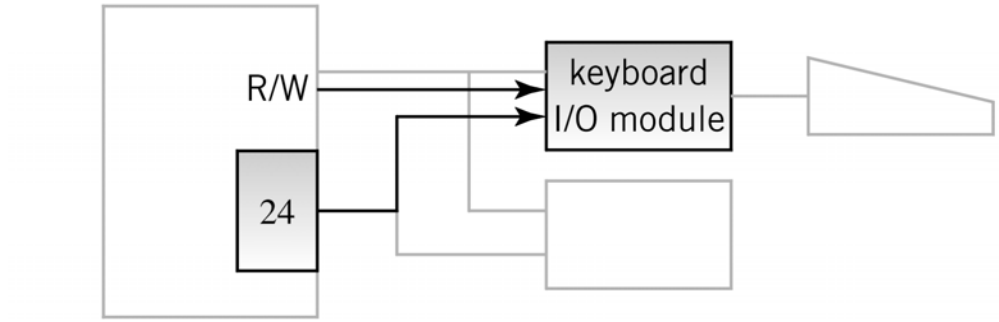
Input Output Techniques: Programmed

- Memory mapped I/O
  - Devices and memory share an address space
  - I/O looks just like memory read/write
  - No special commands for I/O
    - Large selection of memory access commands available
- Isolated I/O
  - Separate address spaces
  - Need I/O or memory select lines
  - Special commands for I/O
    - Limited set

Input Output Techniques: Programmed



1. CPU executes INPUT 24 instruction. Address 24 is copied to the I/O address register.

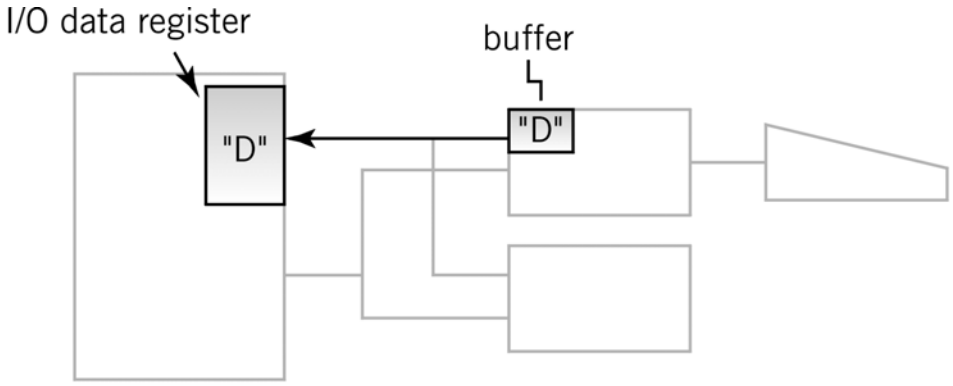


2. Address 24 is recognized by the keyboard I/O module. A read/write control line indicates that the instruction is an INPUT.

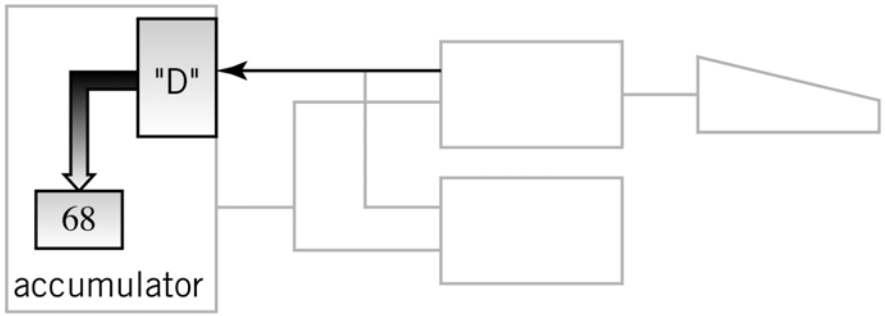
(Figure continues on next slide)

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-04

Input Output Techniques: Programmed



3. A buffer in the I/O module holds a keystroke, in this case ASCII 68, the letter "D". The data is transferred to the I/O data register.



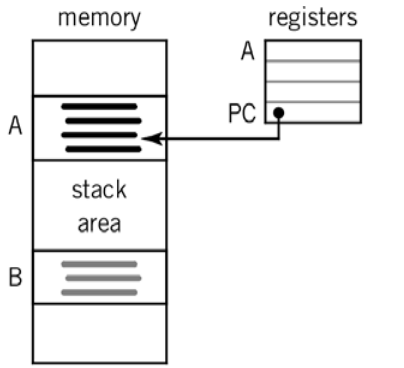
4. From there it is copied to the appropriate accumulator or general-purpose register, completing the operation.

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-04 continued

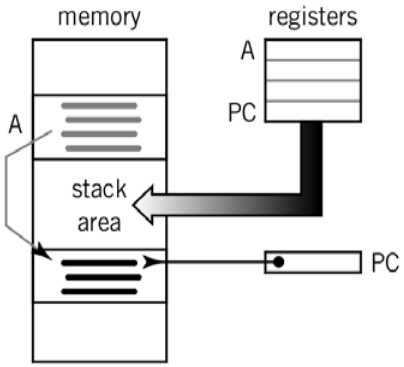
Interrupts

- Mechanism by which other modules (e.g. I/O) may interrupt normal sequence of processing
- Program
  e.g. overflow, division by zero
- Timer
  Generated by internal processor timer
- I/O
  from I/O controller
- Hardware failure
  e.g. memory parity error

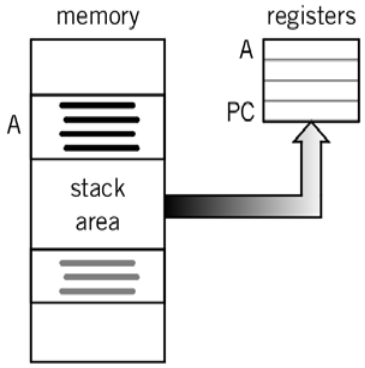
Interrupt Program Flow



1. Before interrupt arrives, program A is executing. The program counter points to the current instruction.



2. When the interrupt is received by the CPU, the current instruction is completed, all the registers are saved in the stack area (or in a special area known as a process control block). The PC is loaded with the starting location of program B, the interrupt handler program. This causes a jump to program B, which becomes the executing program.



3. When the interrupt routine is complete the registers are restored, including the program counter, and the original program resumes exactly where it left off.

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-05

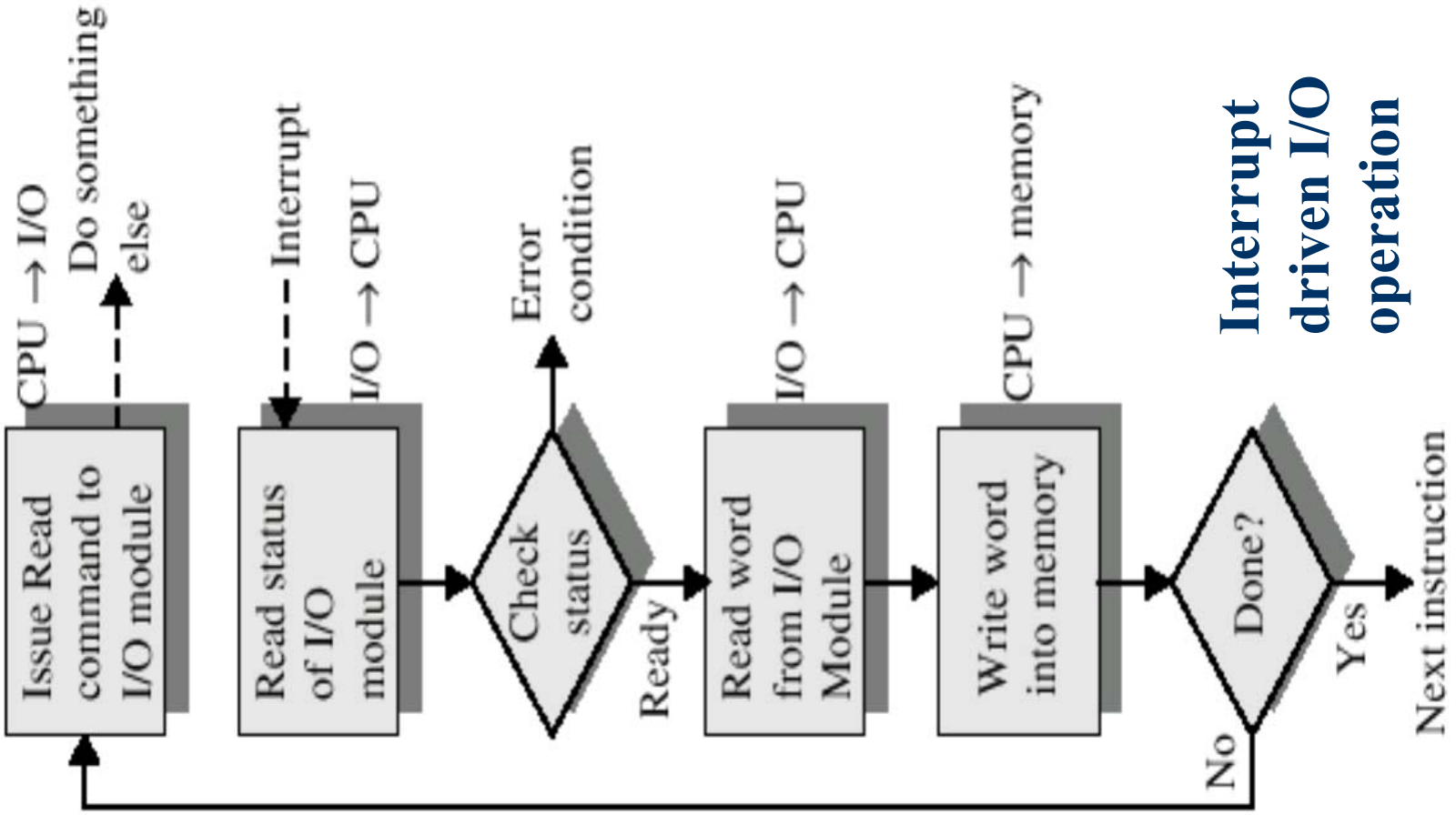
Interrupt Cycle

- Processor checks for interrupt
  - Indicated by an interrupt signal (a control signal)
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program

Input Output Techniques: Interrupt Driven

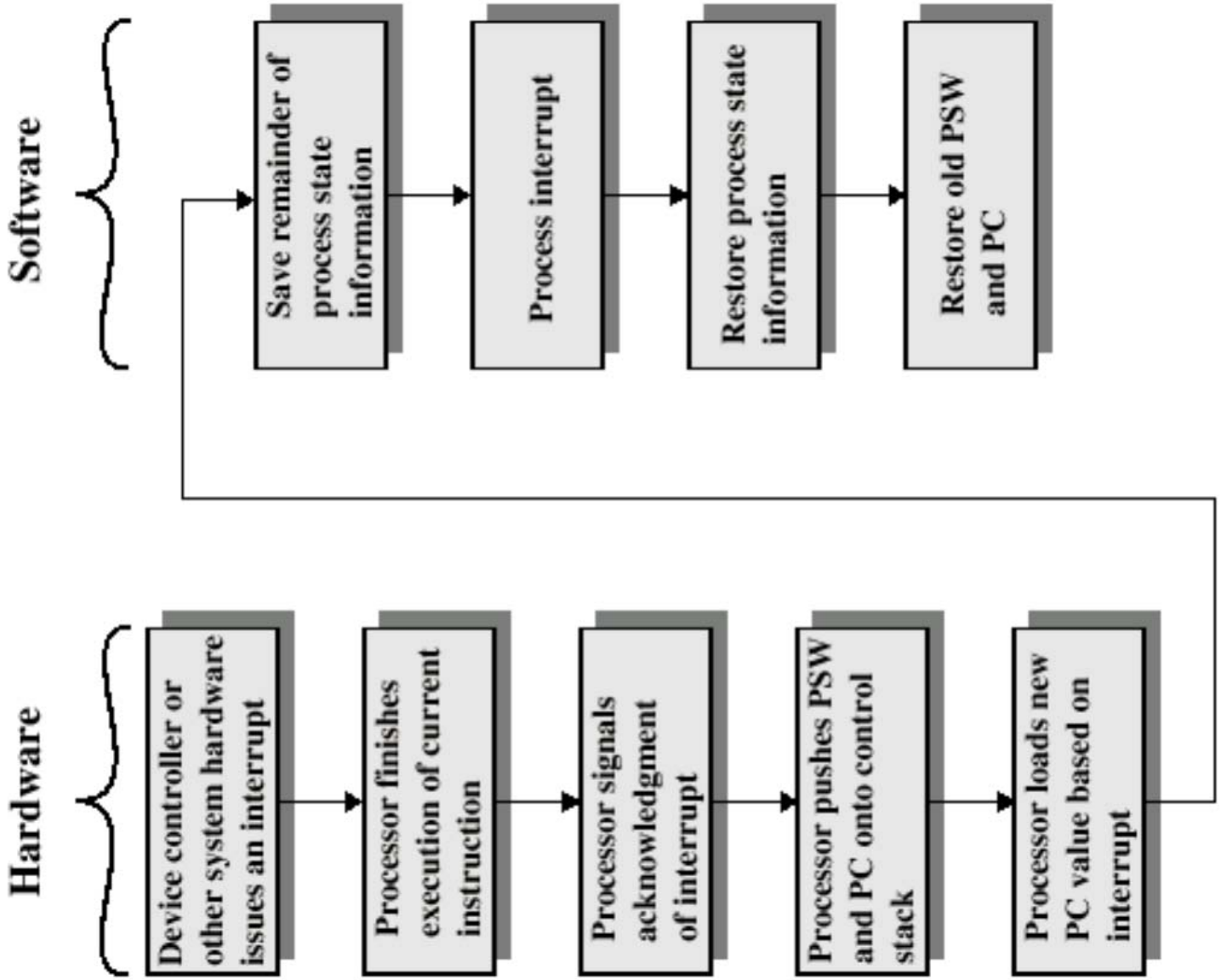
- To reduce the time spent on I/O operation, the CPU can use an interrupt-driven approach
  - CPU issues I/O command to the module
  - CPU continues with its other tasks while the module performs its task
  - Module signals the CPU when the I/O operation is finished (the interrupt)
  - CPU responds to the interrupt by executing an interrupt service routine and then continues on with its primary task
- CPU recognizes and responds to interrupts at the end of an instruction execution cycle
- A wide variety devices use interrupt for I/O

Input Output Techniques: Interrupt Driven

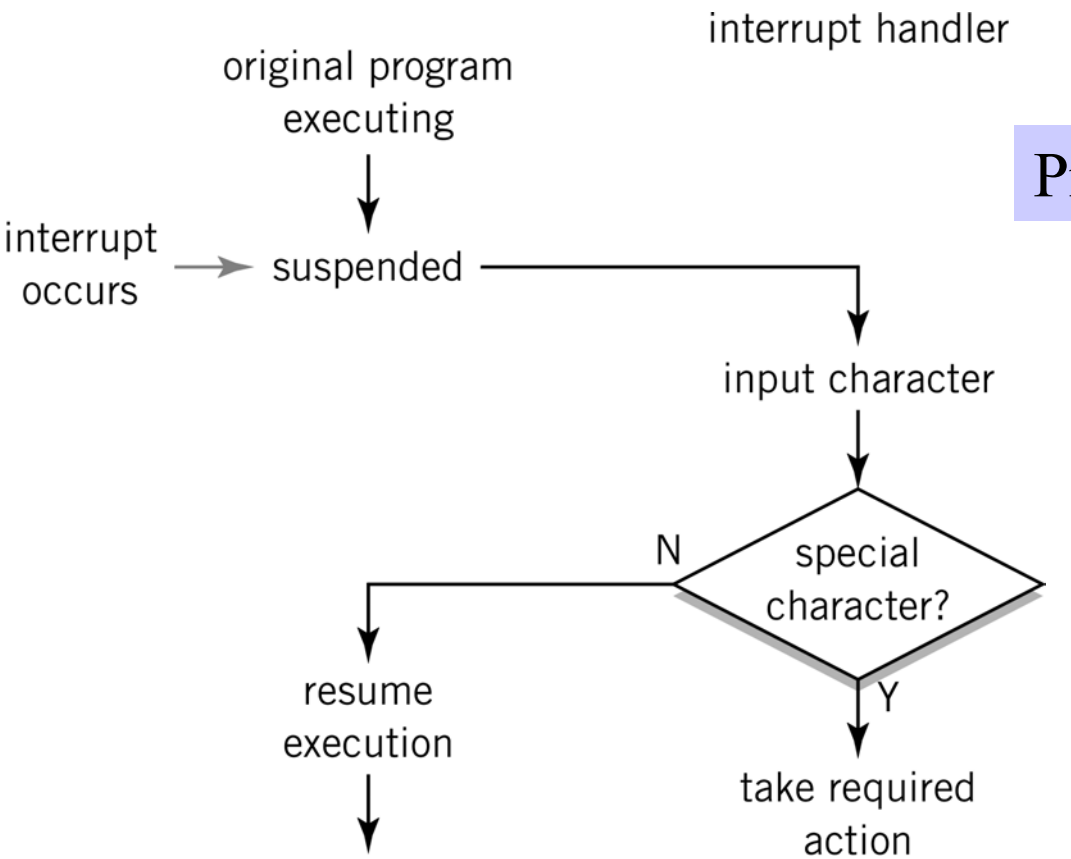


Input Output Techniques: Interrupt Driven

CPU's Response to an Interrupt



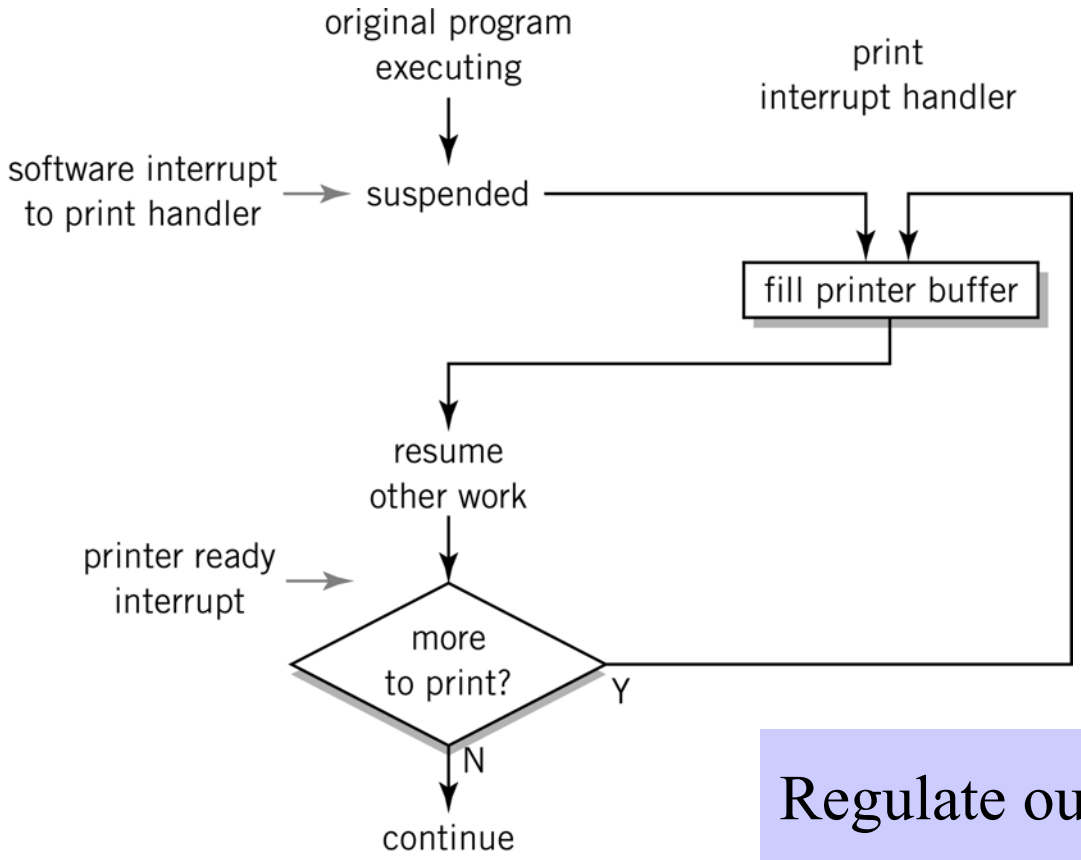
Input Output Techniques: Interrupt Driven



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-06

Process keyboard Input

Input Output Techniques: Interrupt Driven



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-07

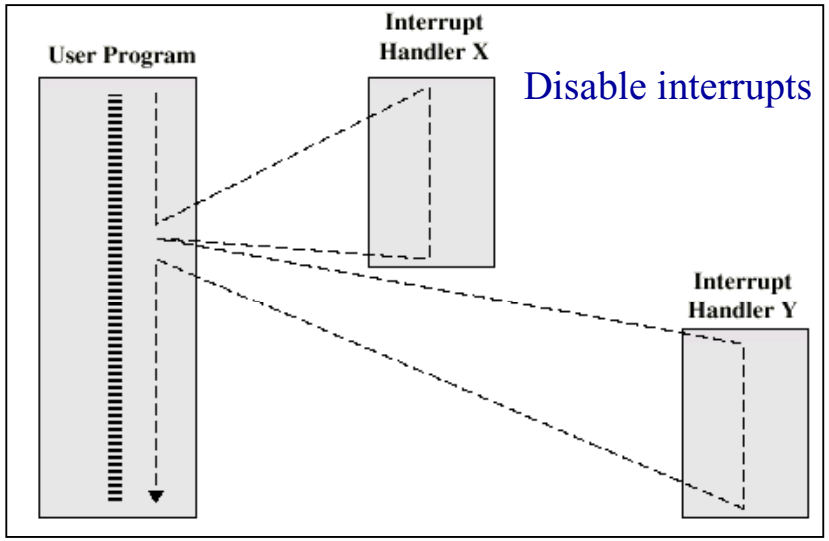
Regulate output flow:

Using a print handler interrupt

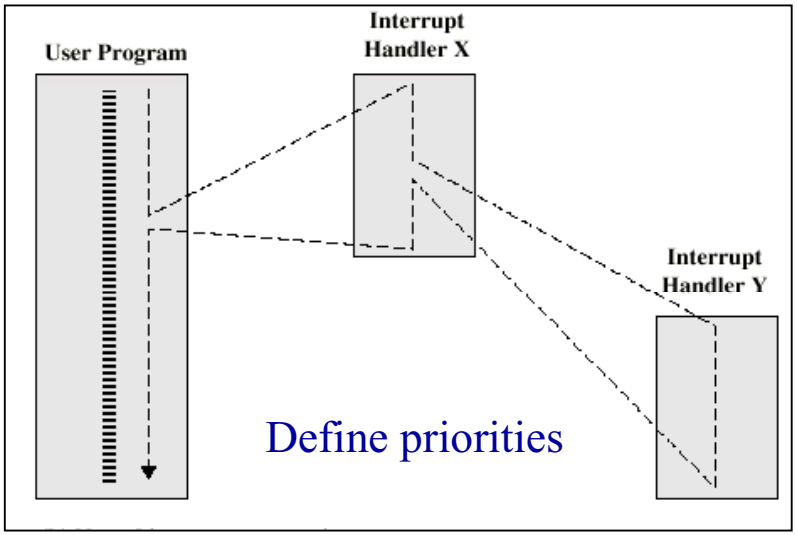
Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Multiple Interrupts



Disable interrupts



Define priorities

Input Output Techniques: DMA

- DMA - Direct Memory Access
- Both programmed and interrupt driven I/O require the continue involvement of the CPU in on going I/O operation
- DMA take the CPU out of the task except for the initialization of the process
- Large amount of data can be transferred without severely impacting CPU performance

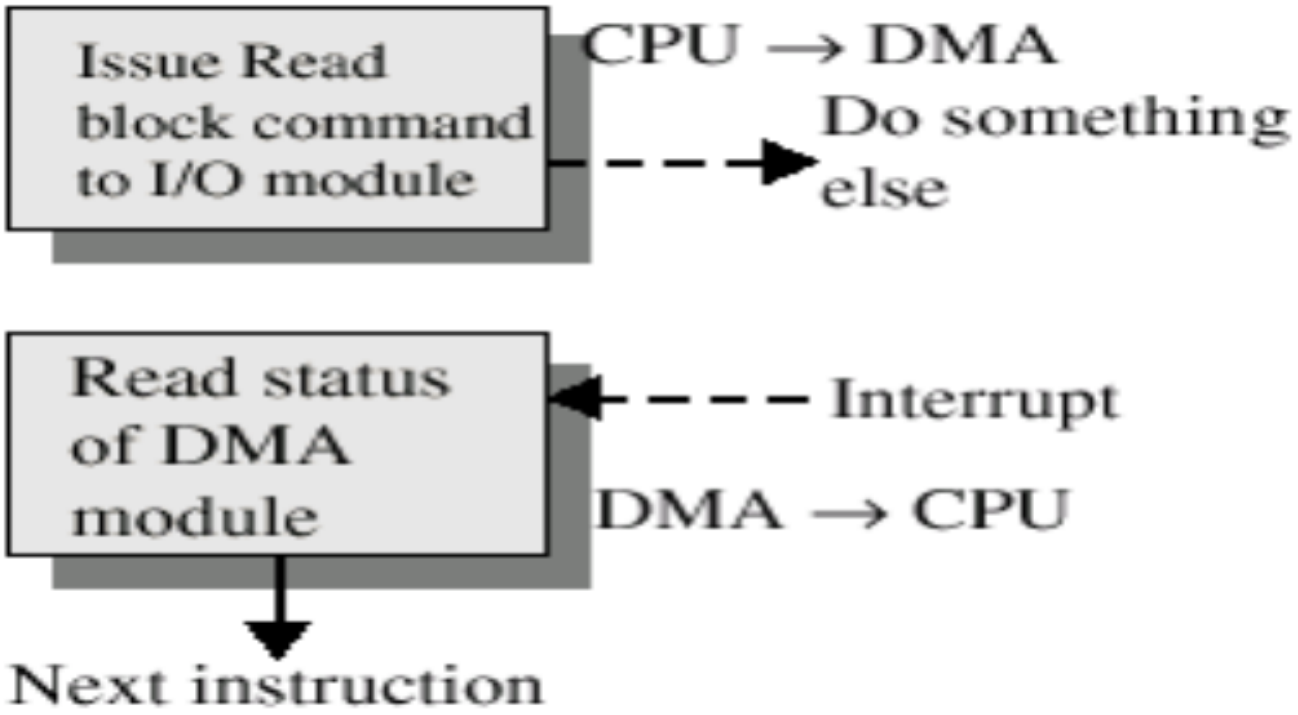
Input Output Techniques: DMA

- DMA operation require additional hardware - DMA Controller module

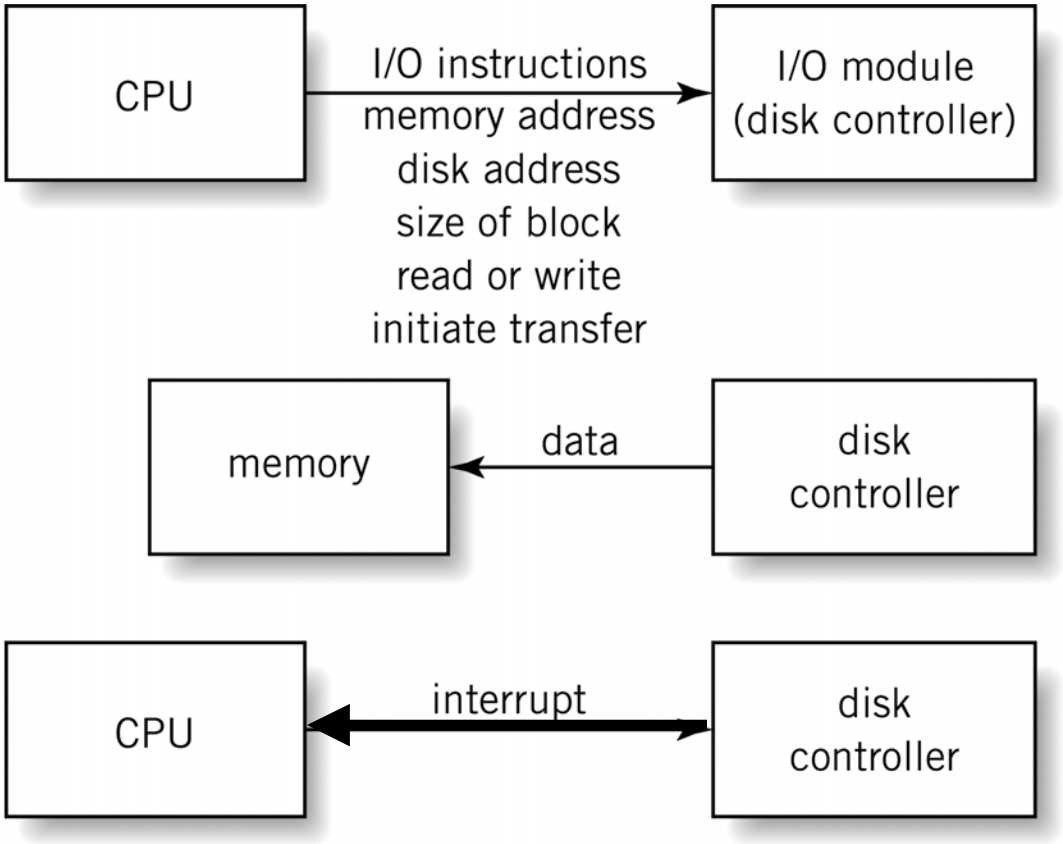
- DMA process
  - CPU initializes DMA module
    - Define read or write operation
    - I/O device involved
    - Start address of memory block
    - Number of words to be transferred
  - CPU then continues with other work

- In practice, DMA uses the bus when the CPU is not using it.
  - No impact on the CPU performance

Input Output Techniques: DMA



Input Output Techniques: DMA



1. Programmed I/O used to prepare I/O module for transfer by providing required information and initiating transfer.

2. DMA transfer. In this case data is transferred from disk to memory.

3. Upon completion, disk controller sends **completion** interrupt to CPU.

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 8, Figure 08-14

Transfer a block of data from memory to disk