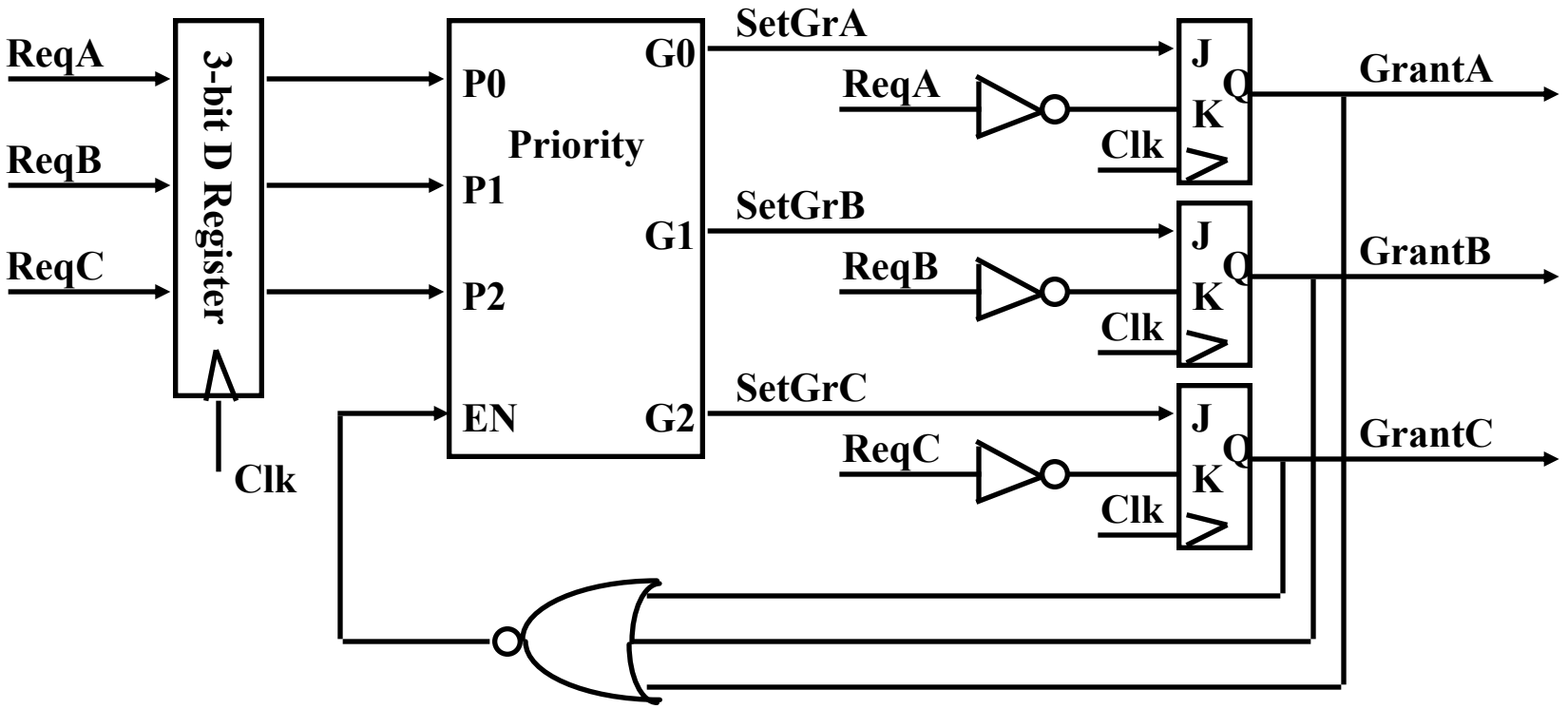
Simple Implementation of a Bus Arbiter



Responsibilities of the Operating System

- The operating system acts as the interface between:
  - The I/O hardware and the program that requests I/O
- Three characteristics of the I/O systems:
  - The I/O system is shared by multiple program using the processor
  - I/O systems often use interrupts (external generated exceptions) to communicate information about I/O operations.
    - Interrupts must be handled by the OS because they cause a transfer to supervisor mode
  - The low-level control of an I/O device is complex:
    - Managing a set of concurrent events
    - The requirements for correct device control are very detailed

Operating System Requirements

- Provide protection to shared I/O resources
  - Guarantees that a user's program can only access the portions of an I/O device to which the user has rights
- Provides abstraction for accessing devices:
  - Supply routines that handle low-level device operation
- Handles the interrupts generated by I/O devices
- Provide equitable access to the shared I/O resources
  - All user programs must have equal access to the I/O resources
- Schedule accesses in order to enhance system throughput

OS and I/O Systems Communication Requirements

- The Operating System must be able to prevent:
  - The user program from communicating with the I/O device directly
- If user programs could perform I/O directly:
  - Protection to the shared I/O resources could not be provided
- Three types of communication are required:
  - The OS must be able to give commands to the I/O devices
  - The I/O device must be able to notify the OS when the I/O device has completed an operation or has encountered an error
  - Data must be transferred between memory and an I/O device

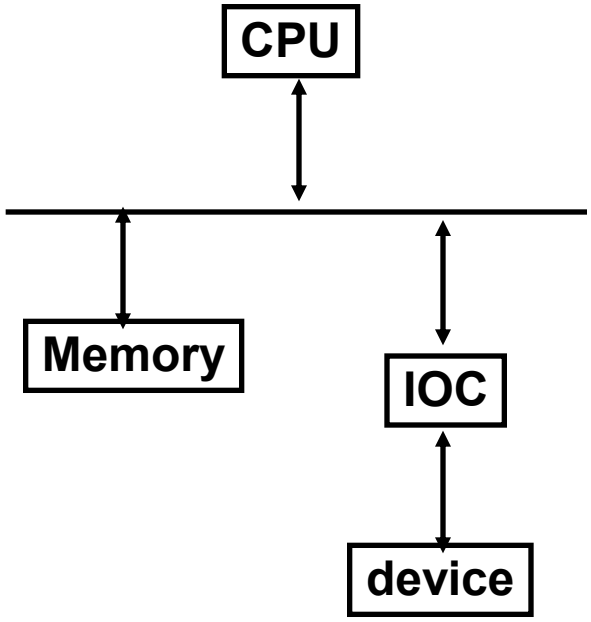
Giving Commands to I/O Devices

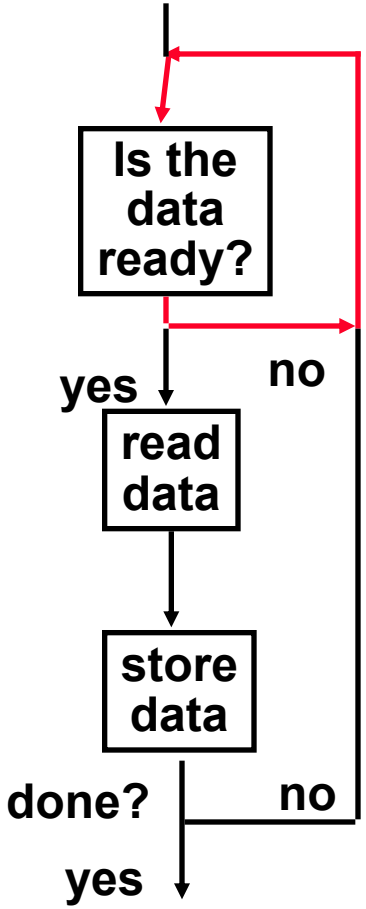
- Two methods are used to address the device:
  - Special I/O instructions
  - Memory-mapped I/O
- Special I/O instructions specify:
  - Both the device number and the command word
    - Device number: the processor communicates this via a set of wires normally included as part of the I/O bus
    - Command word: this is usually send on the bus's data lines
- Memory-mapped I/O:
  - Portions of the address space are assigned to I/O device
  - Read and writes to those addresses are interpreted as commands to the I/O devices
  - User programs are prevented from issuing I/O operations directly:
    - The I/O address space is protected by the address translation

I/O Device Notifying the OS

- The OS needs to know when:
  - The I/O device has completed an operation
  - The I/O operation has encountered an error
- This can be accomplished in two different ways:
  - Polling:
    - The I/O device put information in a status register
    - The OS periodically check the status register
  - I/O Interrupt:
    - Whenever an I/O device needs attention from the processor, it interrupts the processor from what it is currently doing.

Polling: Programmed I/O



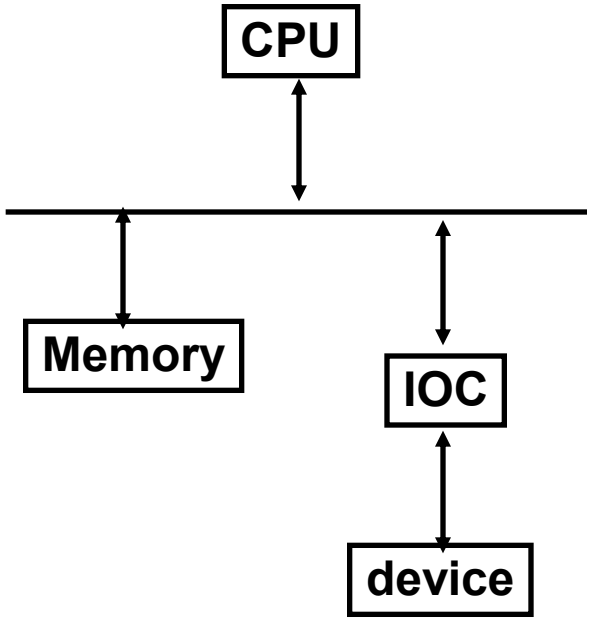


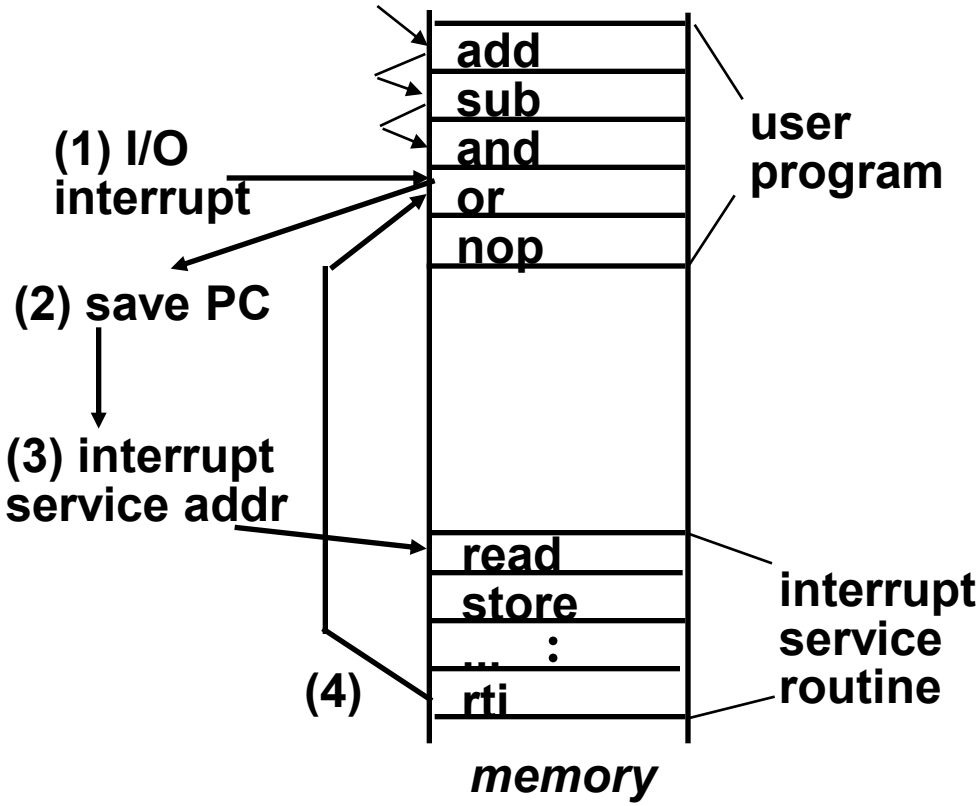
busy wait loop not an efficient way to use the CPU unless the device is very fast!

but checks for I/O completion can be dispersed among computation intensive code

- Advantage:
  - Simple: the processor is totally in control and does all the work
- Disadvantage:
  - Polling overhead can consume a lot of CPU time

Interrupt Driven Data Transfer





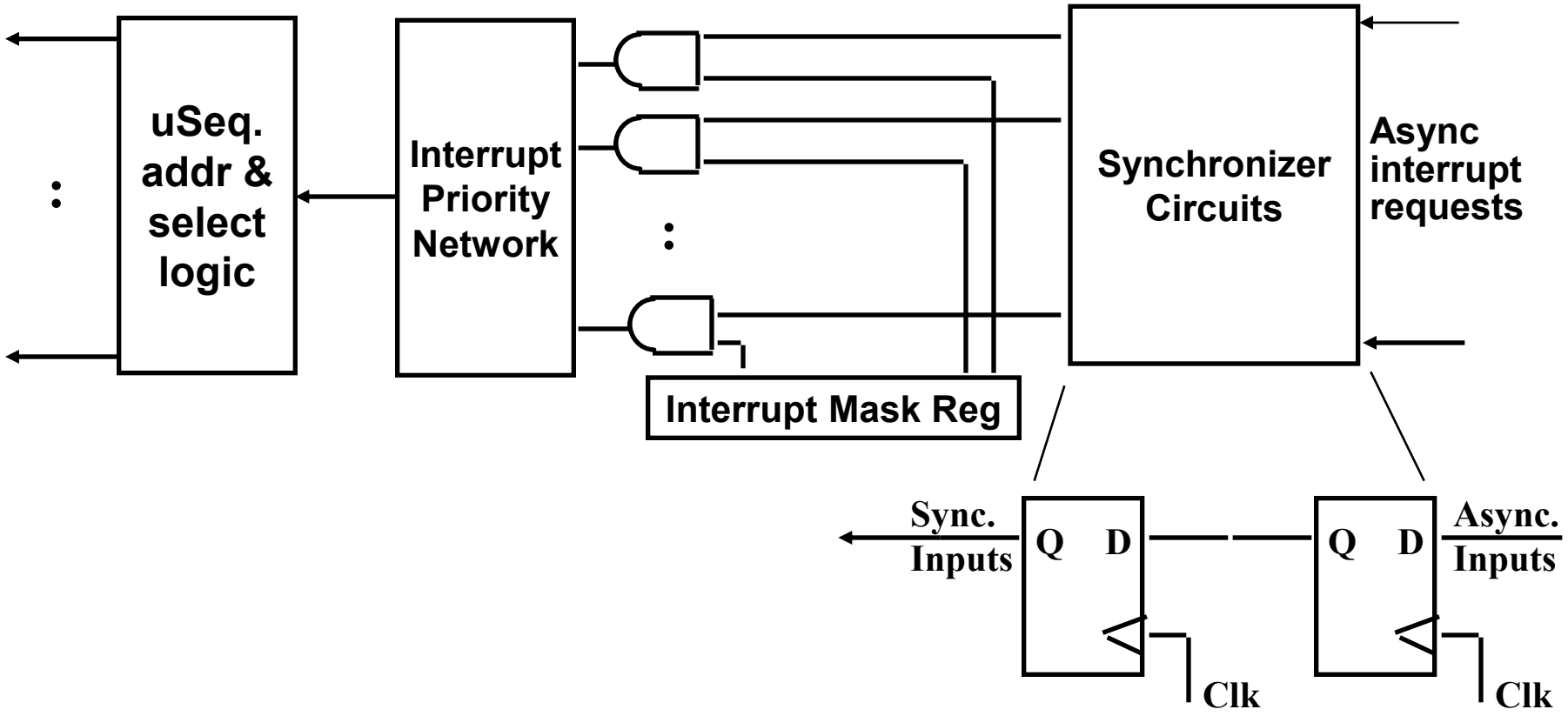
- Advantage:
  - User program progress is only halted during actual transfer
- Disadvantage, special hardware is needed to:
  - Cause an interrupt (I/O device)
  - Detect an interrupt (processor)
  - Save the proper states to resume after the interrupt (processor)

I/O Interrupt

- An I/O interrupt is just like the exceptions except:
  - An I/O interrupt is asynchronous
  - Further information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction
  - I/O interrupt does not prevent any instruction from completion
    - You can pick your own convenient point to take an interrupt
- I/O interrupt is more complicated than exception:
  - Needs to convey the identity of the device generating the interrupt
  - Interrupt requests can have different urgencies:
    - Interrupt request needs to be prioritized

Interrupt Logic

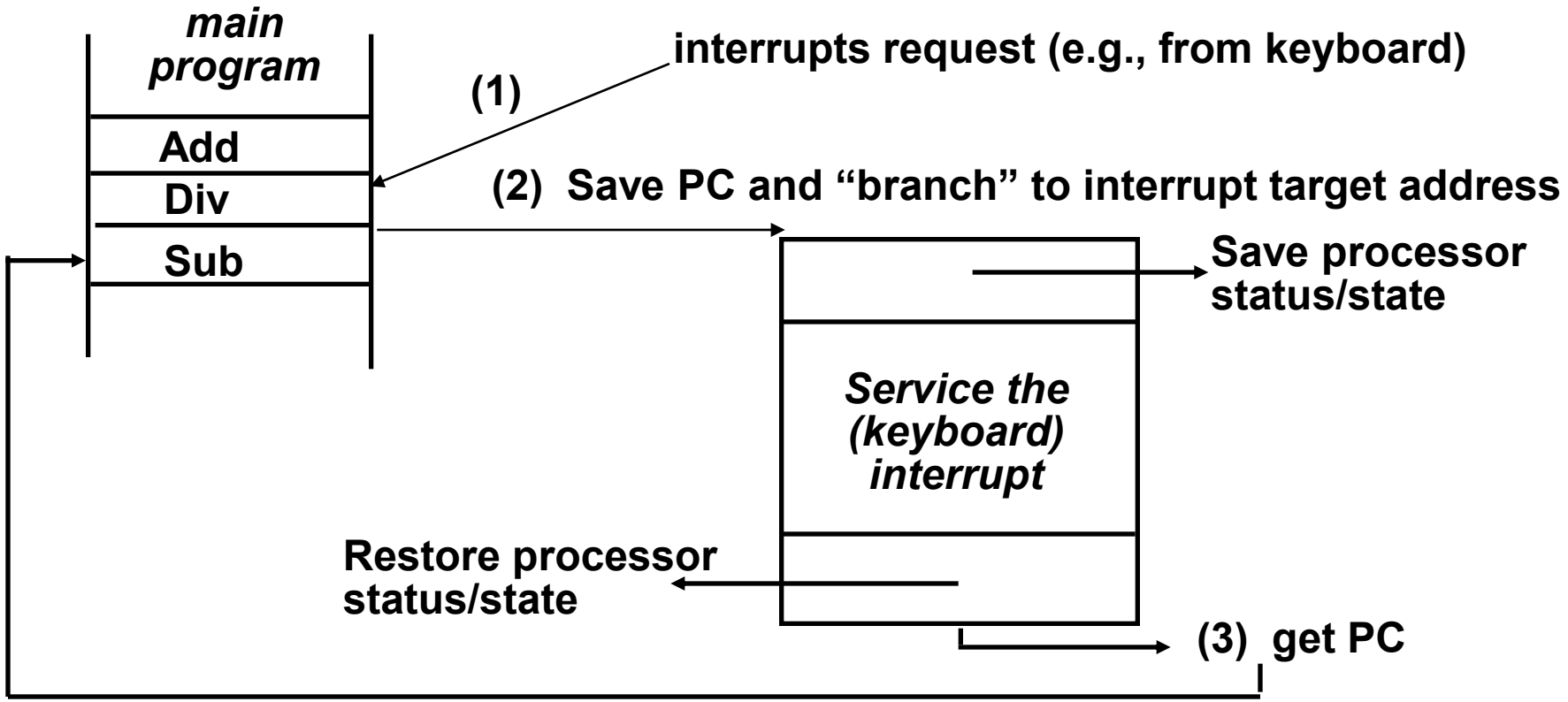
- Detect and synchronize interrupt requests
  - Ignore interrupts that are disabled (masked off)
  - Rank the pending interrupt requests
  - Create interrupt microsequence address
  - Provide select signals for interrupt microsequence



Program Interrupt/Exception Hardware

- Hardware interrupt services:
  - Save the PC (or PCs in a pipelined machine)
  - Inhibit the interrupt that is being handled
  - Branch to interrupt service routine
  - Options:
    - Save status, save registers, save interrupt information
    - Change status, change operating modes, get interrupt info.
- A “good thing” about interrupt:
  - Asynchronous: not associated with a particular instruction
  - Pick the most convenient place in the pipeline to handle it

Programmer's View

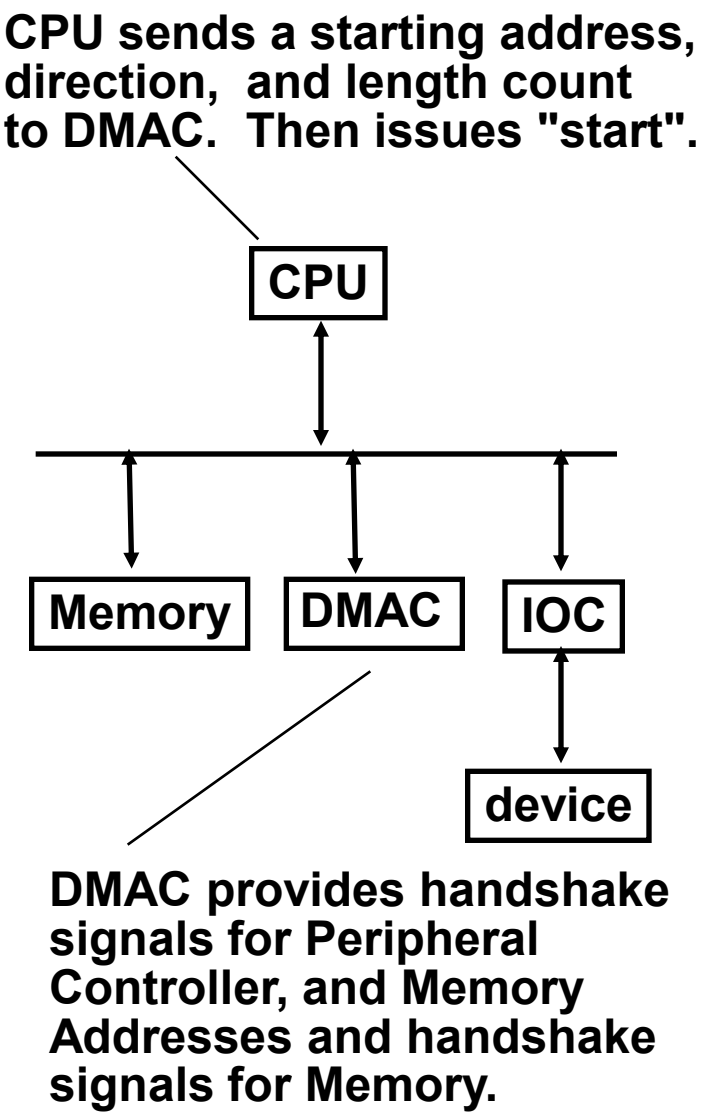


° Interrupt target address options:

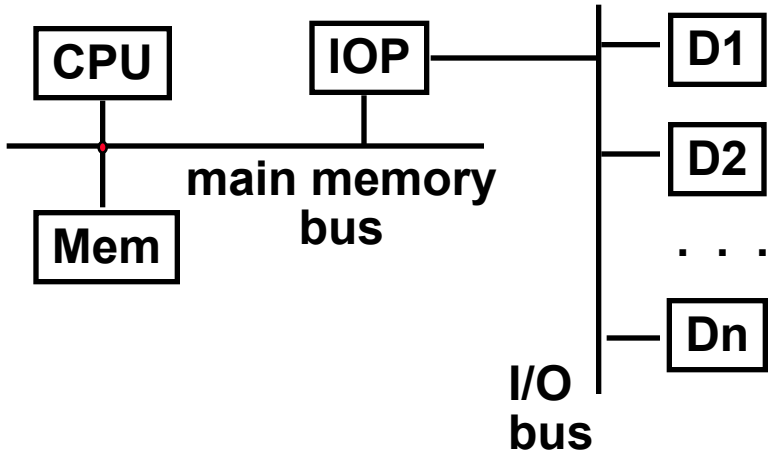
- General: Branch to a common address for all interrupts
  Software then decode the cause and figure out what to do
- Specific: Automatically branch to different addresses based on interrupt type and/or level--vectored interrupt

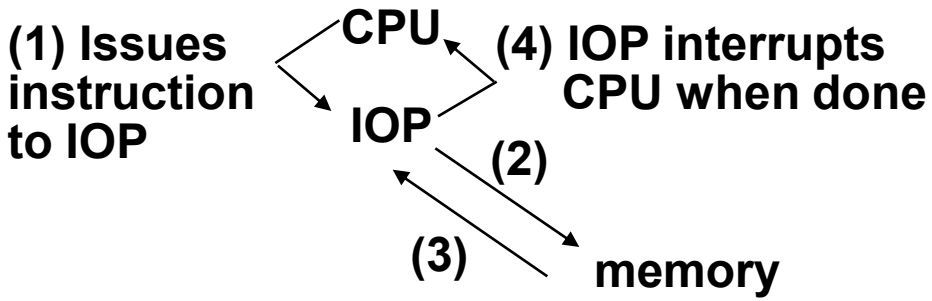
Delegating I/O Responsibility from the CPU: DMA

- Direct Memory Access (DMA):
  - External to the CPU
  - Act as a maser on the bus
  - Transfer blocks of data to or from memory without CPU intervention



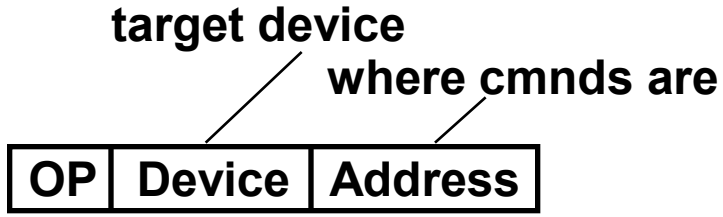
Delegating I/O Responsibility from the CPU: IOP



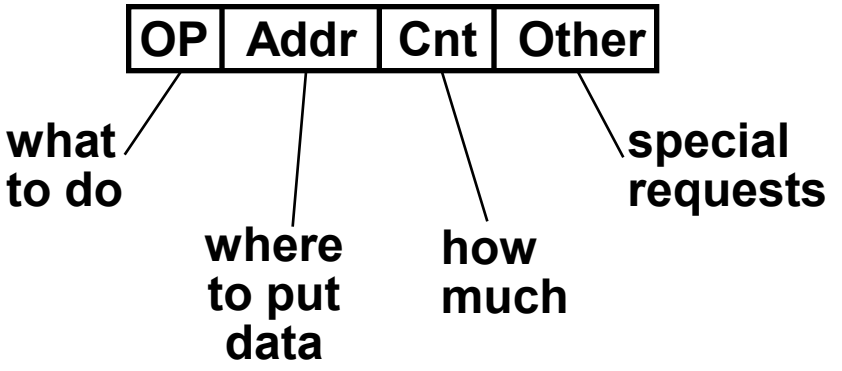


Device to/from memory transfers are controlled by the IOP directly.

IOP steals memory cycles.



IOP looks in memory for commands



Summary:

- Three types of buses:
  - Processor-memory buses
  - I/O buses
  - Backplane buses
- Bus arbitration schemes:
  - Daisy chain arbitration: it cannot assure fairness
  - Centralized parallel arbitration: requires a central arbiter
- I/O device notifying the operating system:
  - Polling: it can waste a lot of processor time
  - I/O interrupt: similar to exception except it is asynchronous
- Delegating I/O responsibility from the CPU
  - Direct memory access (DMA)
  - I/O processor (IOP)