CS152
Computer Architecture and Engineering
Lecture 20: Busses and OS's Responsibilities

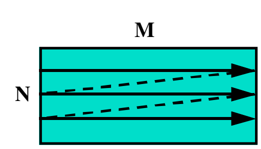
April 7, 1995

Dave Patterson (patterson@cs) and
Shing Kong (shing.kong@eng.sun.com)

Slides available on http://http.cs.berkeley.edu/~patterson

Recap: IO Benchmarks and I/O Devices

- Disk I/O Benchmarks:
  - Supercomputer Application: main concern is data rate
  - Transaction Processing: main concern is I/O rate
  - File System: main concern is file access
- Three Components of Disk Access Time:
  - Seek Time: advertised to be 12 to 20ms. May be lower in real life.
  - Rotational Latency: 5.6 ms at 5400 RPM and 8.3 ms at 3600 RPM
  - Transfer Time: 2 to 4 MB per second



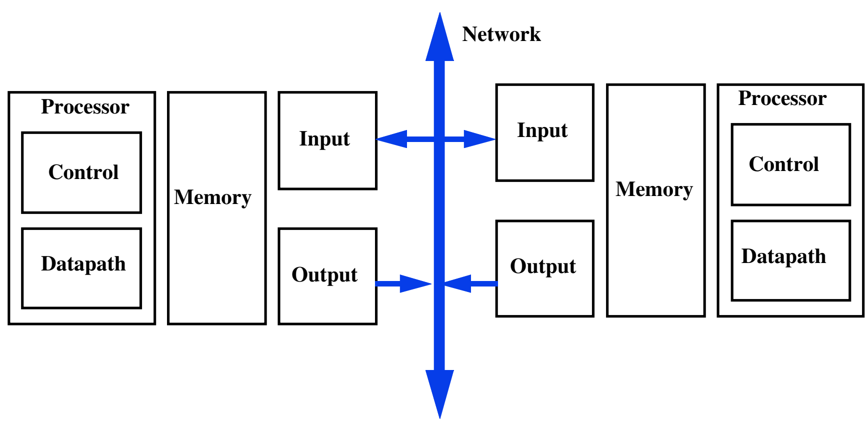
- Graphic Display:
  - Resolution: (M pixels) x (N scan lines)
  - Frame Buffer size and bandwidth requirement can be reduced by placing a Color Map between the Frame Buffer and CRT display
  - VRAM: a DRAM core with a high speed shift register

Outline of Today's Lecture

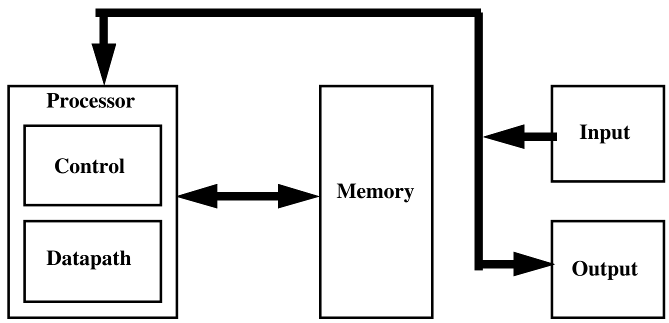
- Recap and Introduction (5 minutes)
- Introduction to Buses (15 minutes)
- Questions and Administrative Matters (5 minutes)
- Bus Types and Bus Operation (10 minutes)
- Bus Arbitration and How to Design a Bus Arbiter (15 minutes)
- Break (5 minutes)
- Operating System's Role (15 minutes)
- Delegating I/O Responsibility from the CPU (5 minutes)
- Summary (5 minutes)

The Big Picture: Where are We Now?

- Today's Topic: How to connect I/O to the rest of the computer?
- Next Wednesday Topic: Network

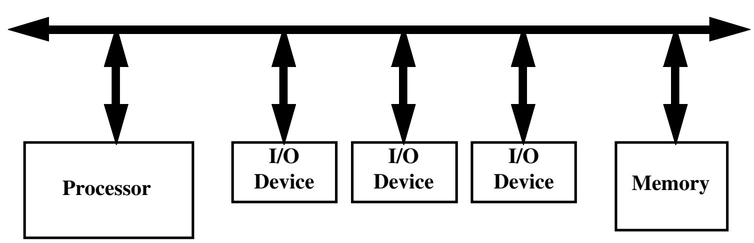


Buses: Connecting I/O to Processor and Memory



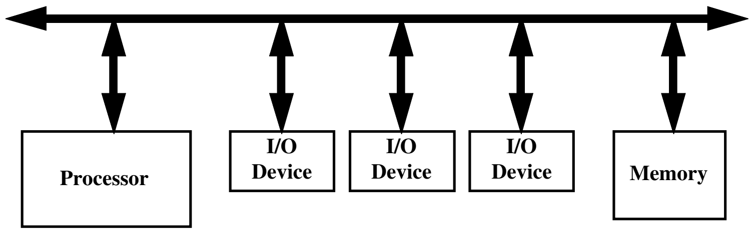
- A bus is a shared communication link
- It uses one set of wires to connect multiple subsystems

Advantages of Buses



- Versatility:
  - New devices can be added easily
  - Peripherals can be moved between computer systems that use the same bus standard
- Low Cost:
  - A single set of wires is shared in multiple ways

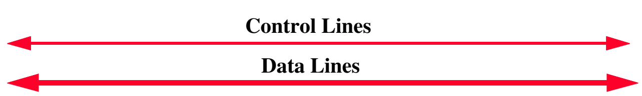
Disadvantages of Buses



° It creates a communication bottleneck
  • The bandwidth of that bus can limit the maximum I/O throughput

° The maximum bus speed is largely limited by:
  • The length of the bus
  • The number of devices on the bus
  • The need to support a range of devices with:
    - Widely varying latencies
    - Widely varying data transfer rates

The General Organization of a Bus

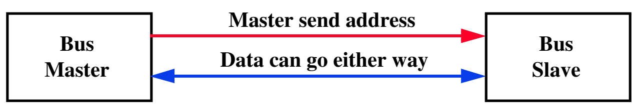


° Control lines:
  • Signal requests and acknowledgments
  • Indicate what type of information is on the data lines

° Data lines carry information between the source and the destination:
  • Data and Addresses
  • Complex commands

° A bus transaction includes two parts:
  • Sending the address
  • Receiving or sending the data

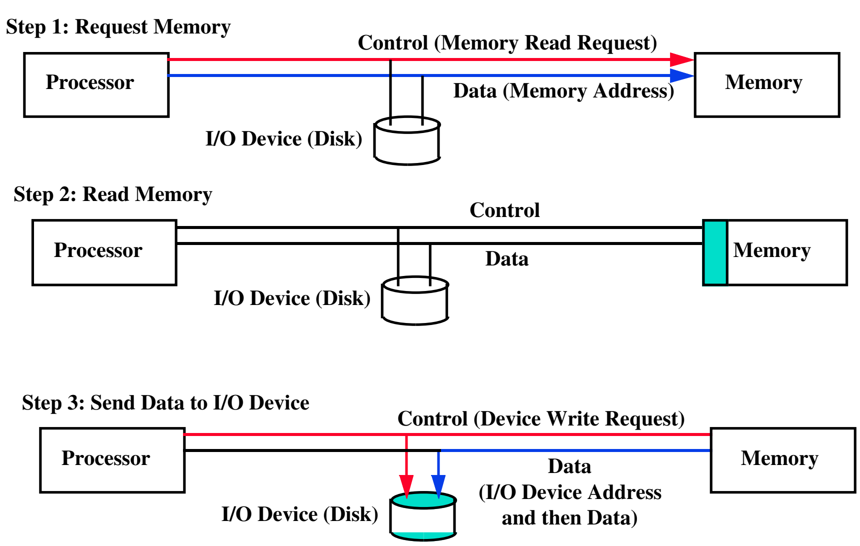
Master versus Slave



- A bus transaction includes two parts:
  - Sending the address
  - Receiving or sending the data
- Master is the one who starts the bus transaction by:
  - Sending the address
- Salve is the one who responds to the address by:
  - Sending data to the master if the master ask for data
  - Receiving data from the master if the master wants to send data

Output Operation

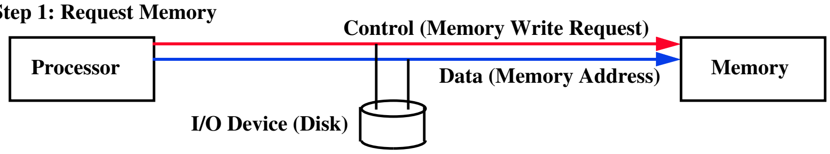
- Output is defined as the Processor sending data to the I/O device:



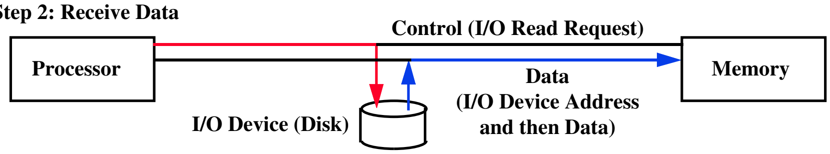
Input Operation

◦ **Input is defined as the Processor receiving data from the I/O device:**

Step 1: Request Memory



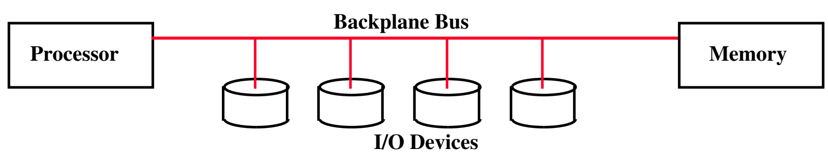
Step 2: Receive Data



Types of Buses

- ◦ **Processor-Memory Bus (design specific)**
  - Short and high speed
  - Only need to match the memory system
    - Maximize memory-to-processor bandwidth
  - Connects directly to the processor
- ◦ **I/O Bus (industry standard)**
  - Usually is lengthy and slower
  - Need to match a wide range of I/O devices
  - Connects to the processor-memory bus or backplane bus
- ◦ **Backplane Bus (industry standard)**
  - Backplane: an interconnection structure within the chassis
  - Allow processors, memory, and I/O devices to coexist
  - Cost advantage: one single bus for all components

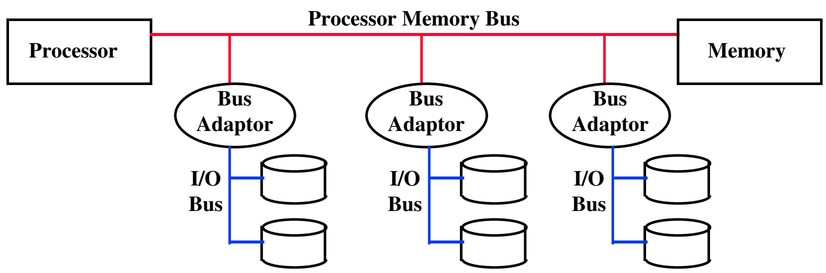
A Computer System with One Bus: Backplane Bus



- A single bus (the backplane bus) is used for:
  - Processor to memory communication
  - Communication between I/O devices and memory
- Advantages: Simple and low cost
- Disadvantages: slow and the bus can become a major bottleneck
- Example: IBM PC

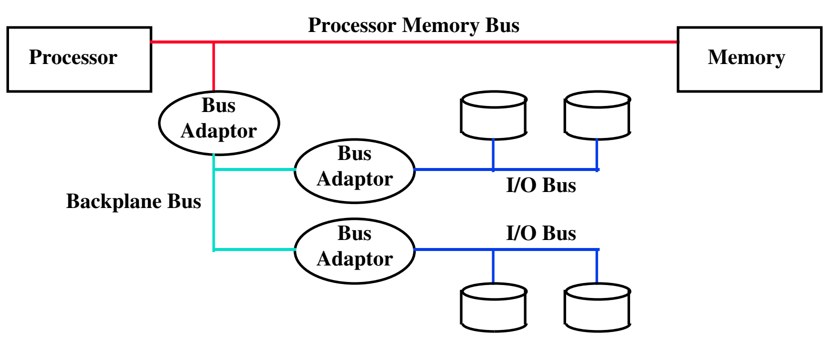
Questions and Administrative Matters (5 Minutes)

A Two-Bus System



- I/O buses tap into the processor-memory bus via bus adaptors:
  - Processor-memory bus: mainly for processor-memory traffic
  - I/O buses: provide expansion slots for I/O devices
- Apple Macintosh-II
  - NuBus: Processor, memory, and a few selected I/O devices
  - SCCI Bus: the rest of the I/O devices

A Three-Bus System



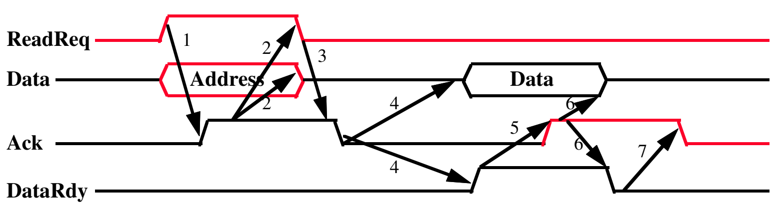
- A small number of backplane buses tap into the processor-memory bus
  - Processor-memory bus is used for processor memory traffic
  - I/O buses are connected to the backplane bus
- Advantage: loading on the processor bus is greatly reduced

Synchronous and Asynchronous Bus

° Synchronous Bus:
  • Includes a clock in the control lines
  • A fixed protocol for communication that is relative to the clock
  • Advantage: involves very little logic and can run very fast
  • Disadvantages:
    - Every device on the bus must run at the same clock rate
    - To avoid clock skew, they cannot be long if they are fast

° Asynchronous Bus:
  • It is not clocked
  • It can accommodate a wide range of devices
  • It can be lengthened without worrying about clock skew
  • It requires a handshaking protocol

A Handshaking Protocol

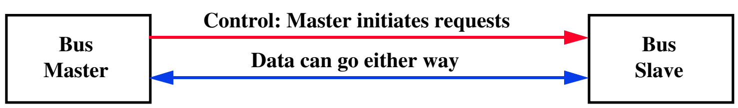


° Three control lines
  • ReadReq: indicate a read request for memory
    Address is put on the data lines at the same line
  • DataRdy: indicate the data word is now ready on the data lines
    Data is put on the data lines at the same time
  • Ack: acknowledge the ReadReq or the DataRdy of the other party

Increasing the Bus Bandwidth

- Separate versus multiplexed address and data lines:
  - Address and data can be transmitted in one bus cycle if separate address and data lines are available
  - Cost: (a) more bus lines, (b) increased complexity
- Data bus width:
  - By increasing the width of the data bus, transfers of multiple words require fewer bus cycles
  - Example: SPARCstation 20's memory bus is 128 bit wide
  - Cost: more bus lines
- Block transfers:
  - Allow the bus to transfer multiple words in back-to-back bus cycles
  - Only one address needs to be sent at the beginning
  - The bus is not released until the last word is transferred
  - Cost: (a) increased complexity
    (b) decreased response time for request

Obtaining Access to the Bus



- One of the most important issues in bus design:
  - How is the bus reserved by a devices that wishes to use it?
- Chaos is avoided by a master-slave arrangement:
  - Only the bus master can control access to the bus:
    It initiates and controls all bus requests
  - A slave responds to read and write requests
- The simplest system:
  - Processor is the only bus master
  - All bus requests must be controlled by the processor
  - Major drawback: the processor is involved in every transaction

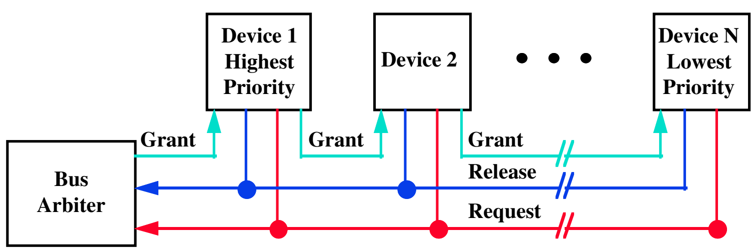
Multiple Potential Bus Masters: the Need for Arbitration

° Bus arbitration scheme:
  - A bus master wanting to use the bus asserts the bus request
  - A bus master cannot use the bus until its request is granted
  - A bus master must signal to the arbiter after finish using the bus

° Bus arbitration schemes usually try to balance two factors:
  - Bus priority: the highest priority device should be serviced first
  - Fairness: Even the lowest priority device should never be completely locked out from the bus

° Bus arbitration schemes can be divided into four broad classes:
  - Distributed arbitration by self-selection: each device wanting the bus places a code indicating its identity on the bus.
  - Distributed arbitration by collision detection: Ethernet uses this.
  - Daisy chain arbitration: see next slide.
  - Centralized, parallel arbitration: see next-next slide

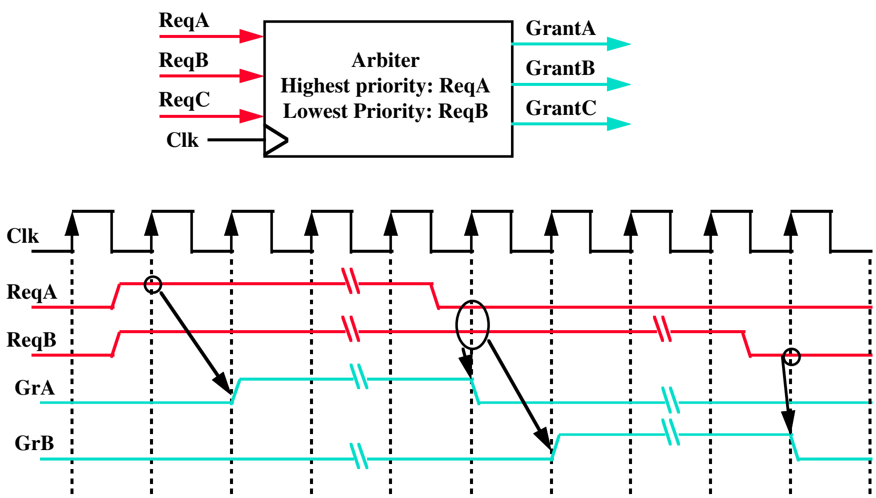
The Daisy Chain Bus Arbitrations Scheme



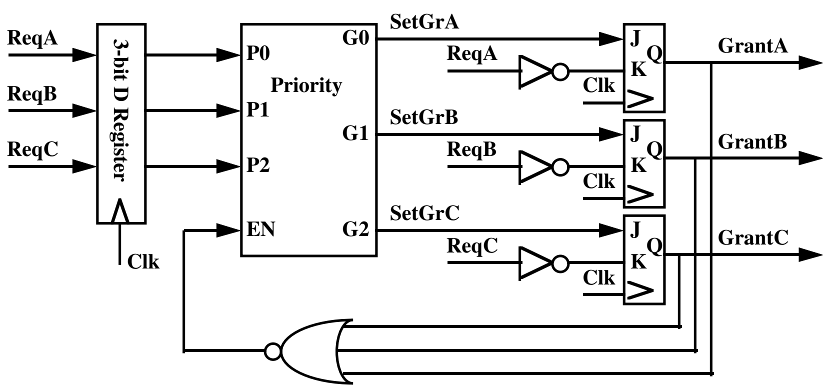
° Advantage: simple

° Disadvantages:
  - Cannot assure fairness:
    A low-priority device may be locked out indefinitely
  - The use of the daisy chain grant signal also limits the bus speed

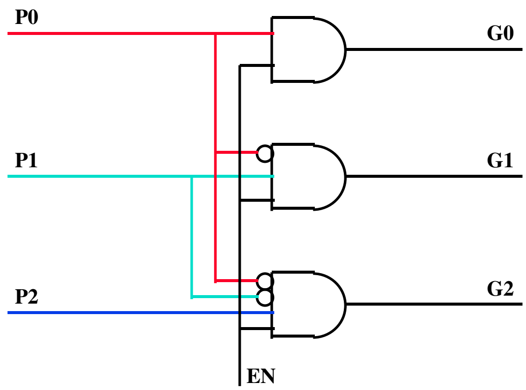
Centralized Arbitration with a Bus Arbiter



Simple Implementation of a Bus Arbiter



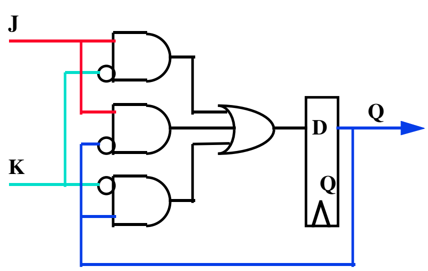
Priority Logic



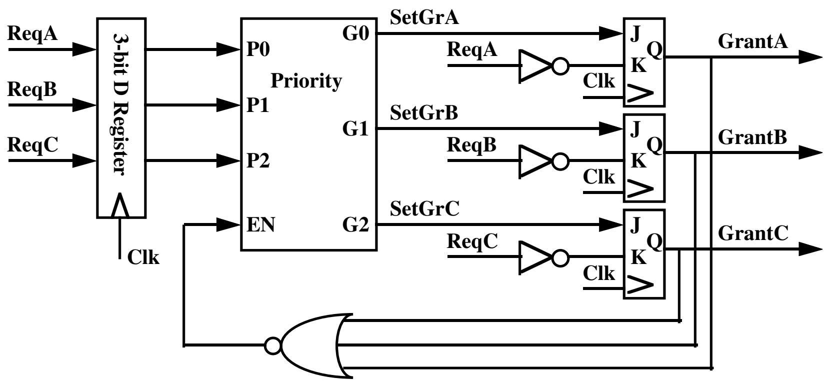
JK Flip Flop

° JK Flip Flop can be implemented with a D-Flip Flop

J	K	Q(t-1)	Q(t)
0	0	0	0
0	0	1	1
0	1	x	0
1	0	x	1
1	1	0	1
1	1	1	0



Simple Implementation of a Bus Arbiter



Break (5 Minutes)

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

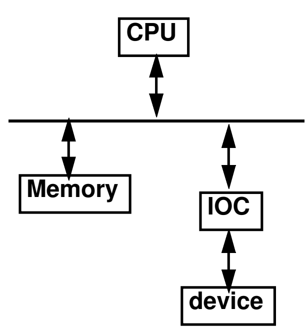
Giving Commands to I/O Devices

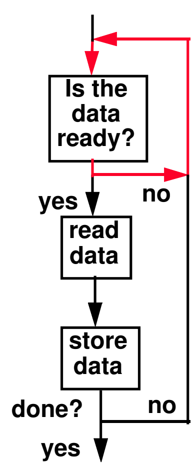
- Two methods are used to address the device:
  - Special I/O instructions
  - Memory-mapped I/O
- Special I/O instructions specify:
  - Both the device number and the command word
    - Device number: the processor communicates this via a set of wires normally included as part of the I/O bus
    - Command 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



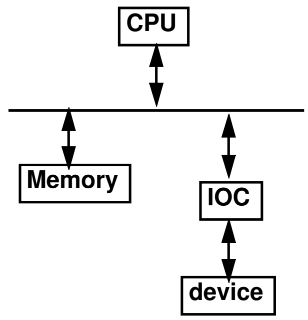


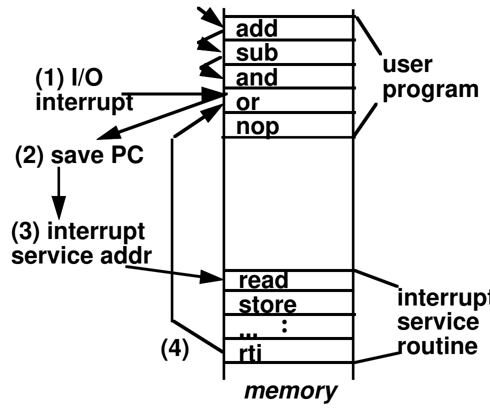
busy wait loop not an efficient way to use the CPU unless the device is very fast!

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:**

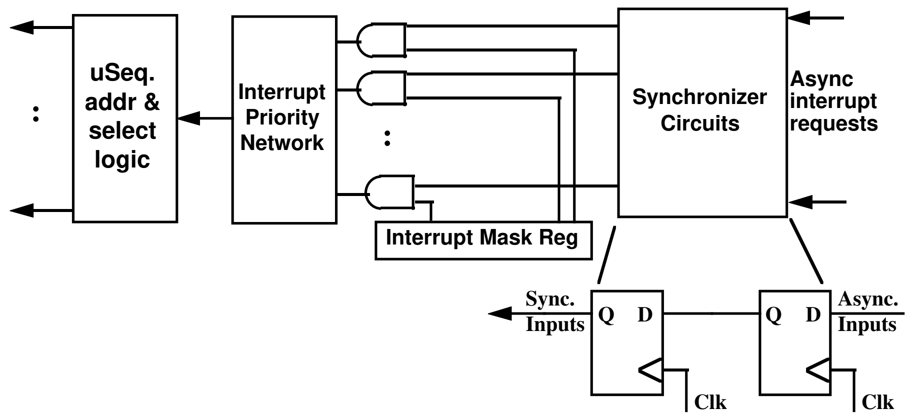
- **I/O interrupt is not associated with any instruction**
- **I/O interrupt does not prevent any instruction from completion**
  - **You can pick your own convenient point to take an interrupt**

° **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

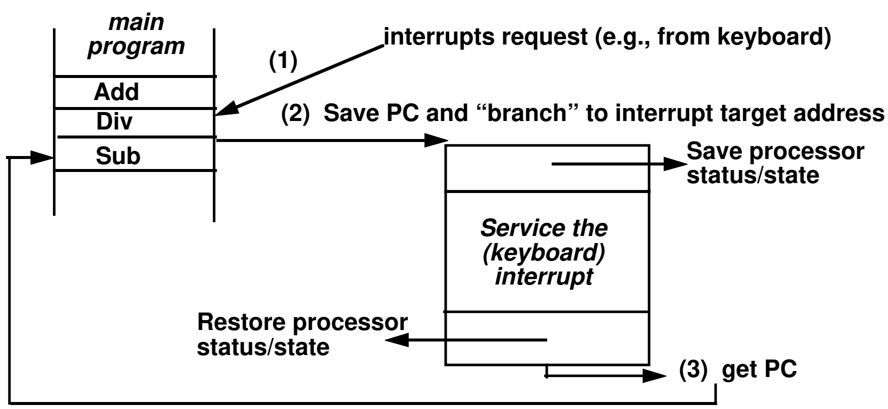


Program Interrupt/Exception Hardware

° 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

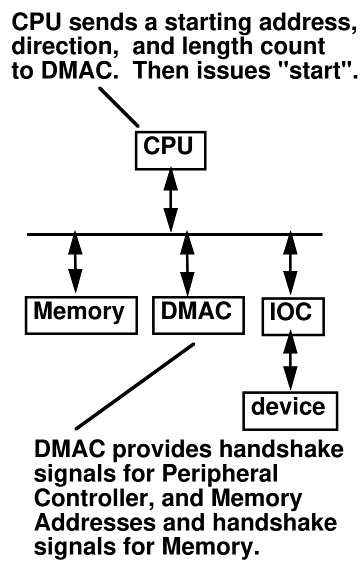
Programmer's View



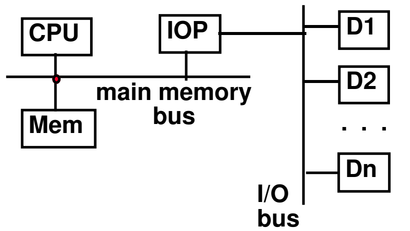
- Interrupt target address options:
  - General: Branch to a common address for all interrupts
    Software then decode the cause and figure out what to do
  - Specific: Automatically branch to different addresses based on
    interrupt type and/or level--vectored interrupt

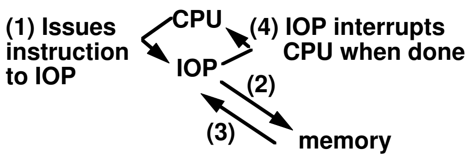
Delegating I/O Responsibility from the CPU: DMA

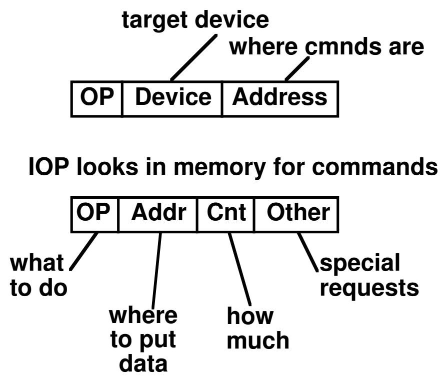
- Direct Memory Access (DMA):
  - External to the CPU
  - Act as a maser on the bus
  - Transfer blocks of data to or from memory without CPU intervention



Delegating I/O Responsibility from the CPU: IOP







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

IOP steals memory cycles.

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)

Where to get more information?

- Happy trail ...

Shing Kong (Kong) shing.kong@eng.sun.com
1047 Noel Dr., Apt. #4
Menlo Park, CA 94025
415-321-2270 415-786-6377 (w)

- Until we meet again :-)