Computer Organization and Operating System

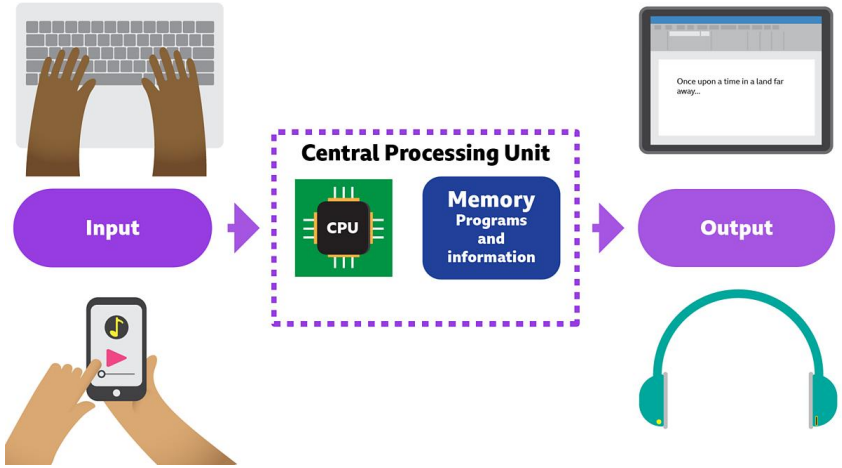
Input and Output

Akharin Khunkitti

KMITL

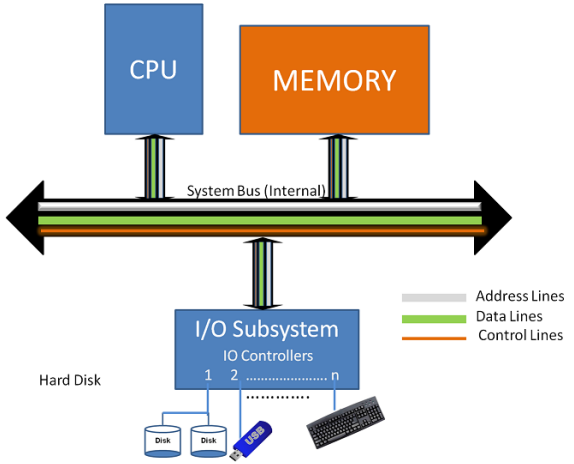
Topic

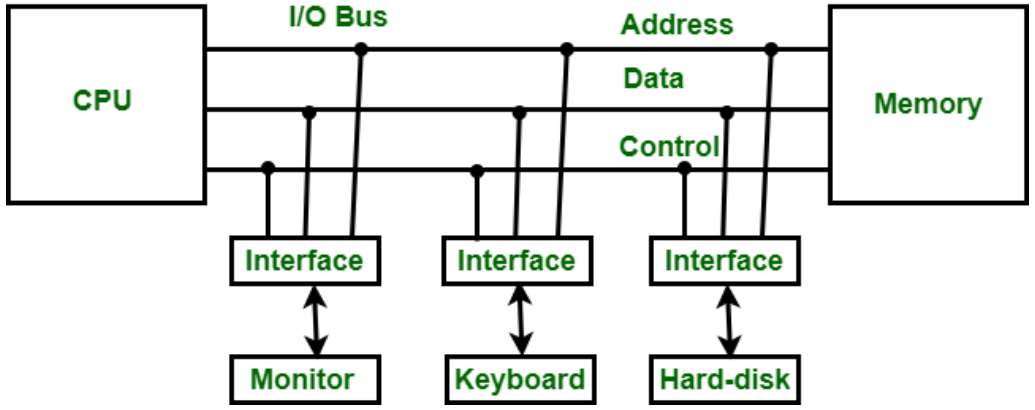
- Input / Output Overview
- I/O Techniques
  - Programmed, Interrupt, DMA
- DMA – Direct Memory Access
- I/O Interfacing
- Conclusion



Input / Output Overview

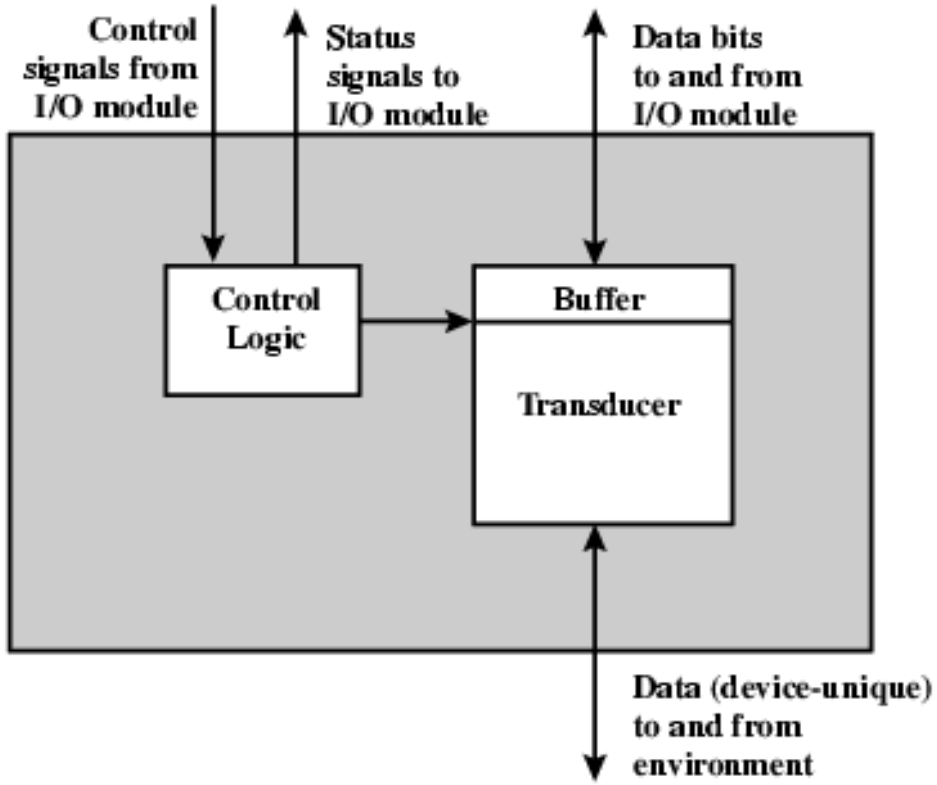
- I/O Functions
  - Move/Transfer Data between computer and “External Devices”
  - Input - Move/Transfer Data From Outside to CPU/Memory
  - Output – Move/Transfer Data From CPU/Memory to Outside
- Internal Locations
  - CPU – Direct to Registers
    - Small number of Registers
  - Memory – Temporary places, wait for CPU processing
    - Can be big amount of data
- Normally Transfer between Main Memory
  - Input => Memory
  - Memory => Output





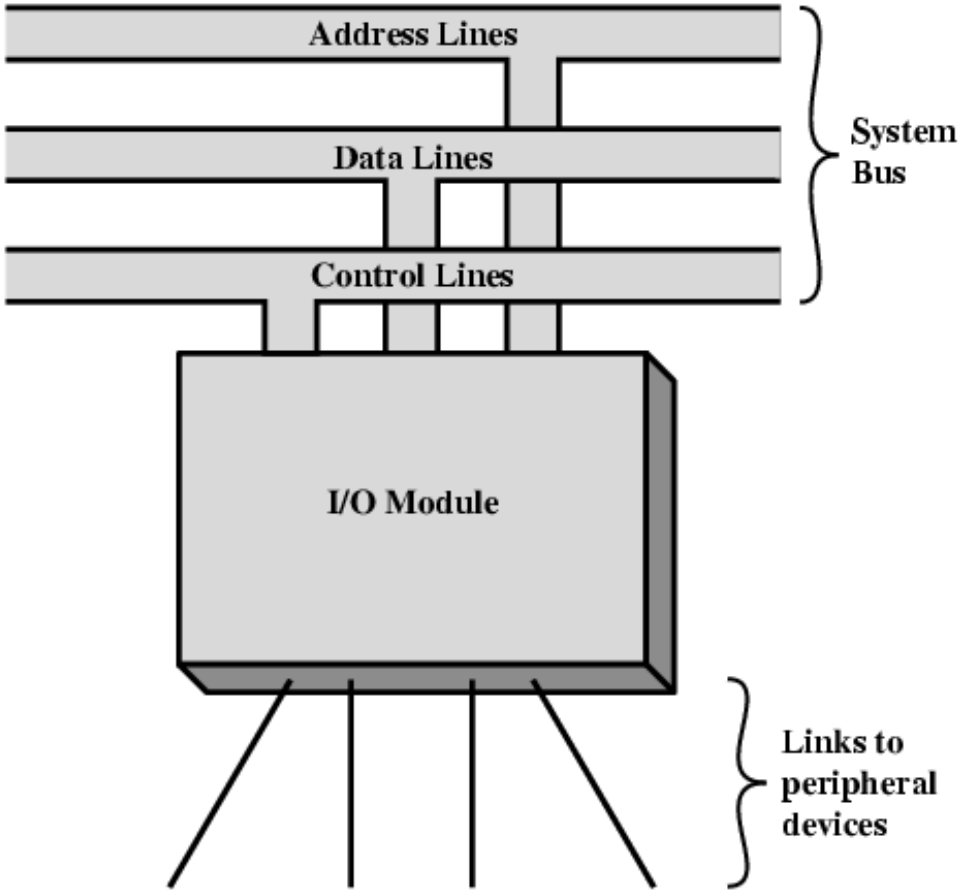
External Devices

- Human readable
  - Screen, printer, keyboard
- Machine readable
  - Monitoring and control
- Communication
  - Modem
  - Network Interface Card (NIC)



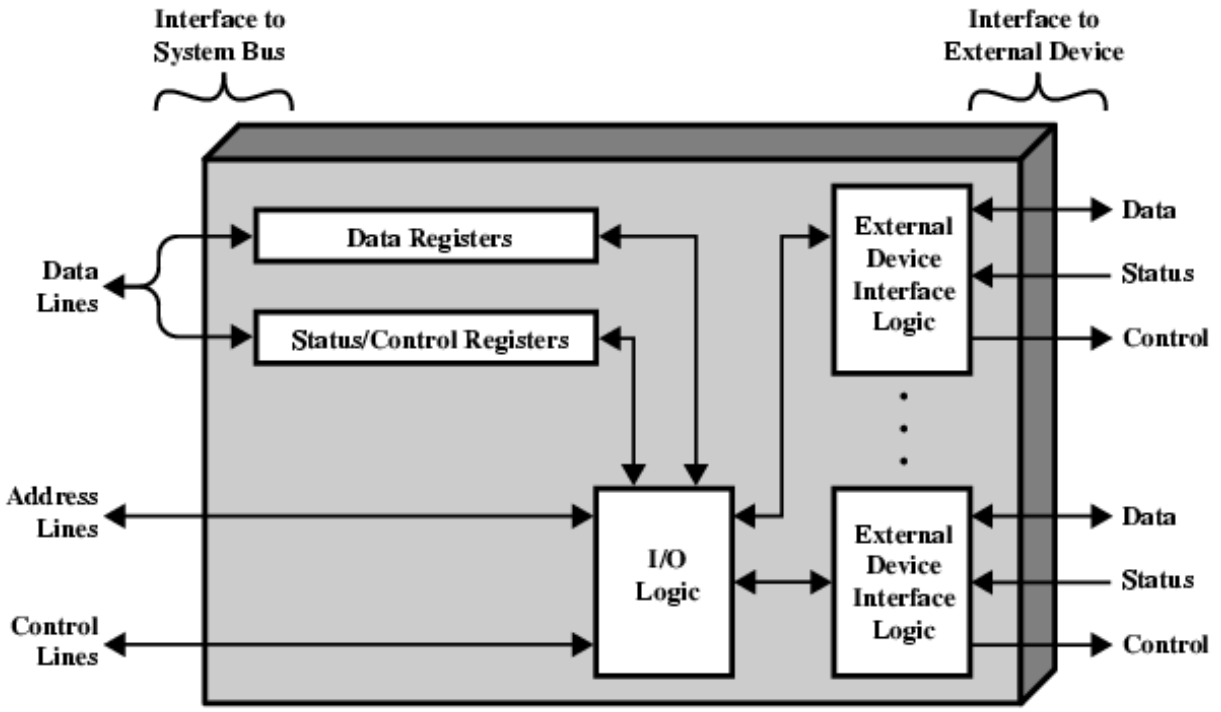
Input / Output Modules

- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and Memory (RAM)
- Need I/O modules
- Interface to CPU and Memory
- Interface to one or more peripherals



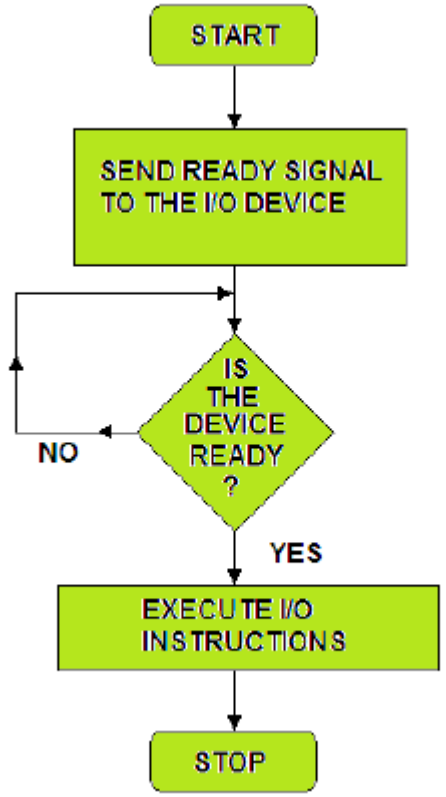
I/O Module Functions

- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection



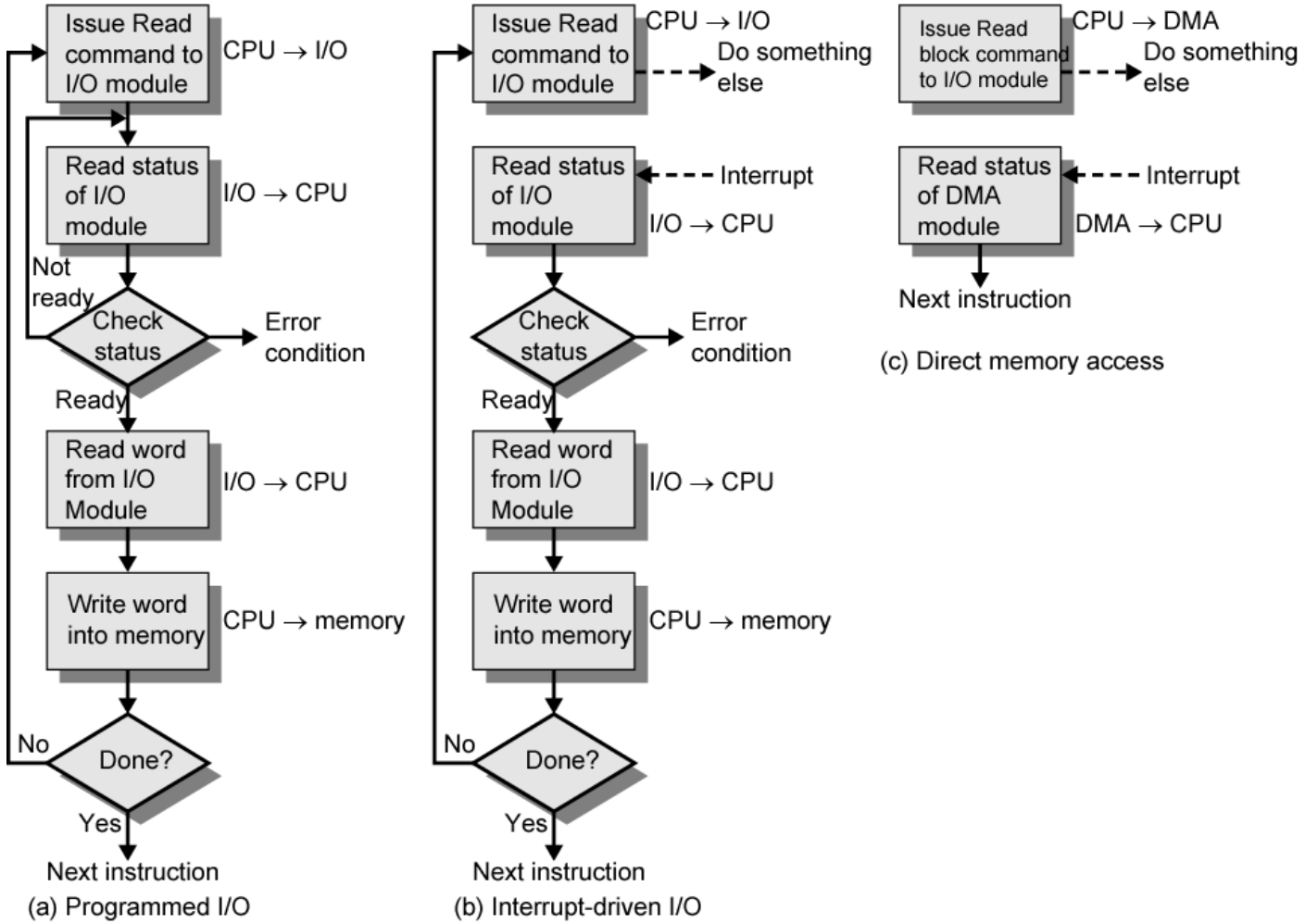
I/O Transfer Steps

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.



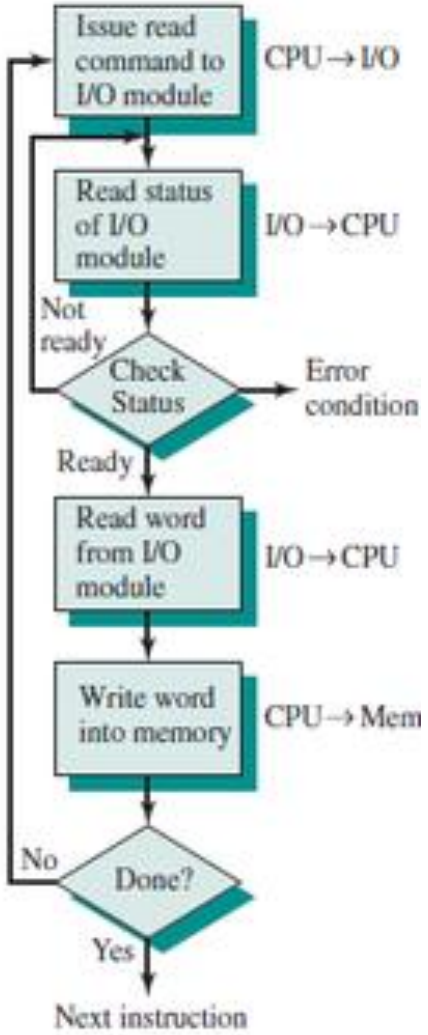
Input / Output Techniques

- Programmed I/O
- Interrupt I/O
- Direct Memory Access (DMA)



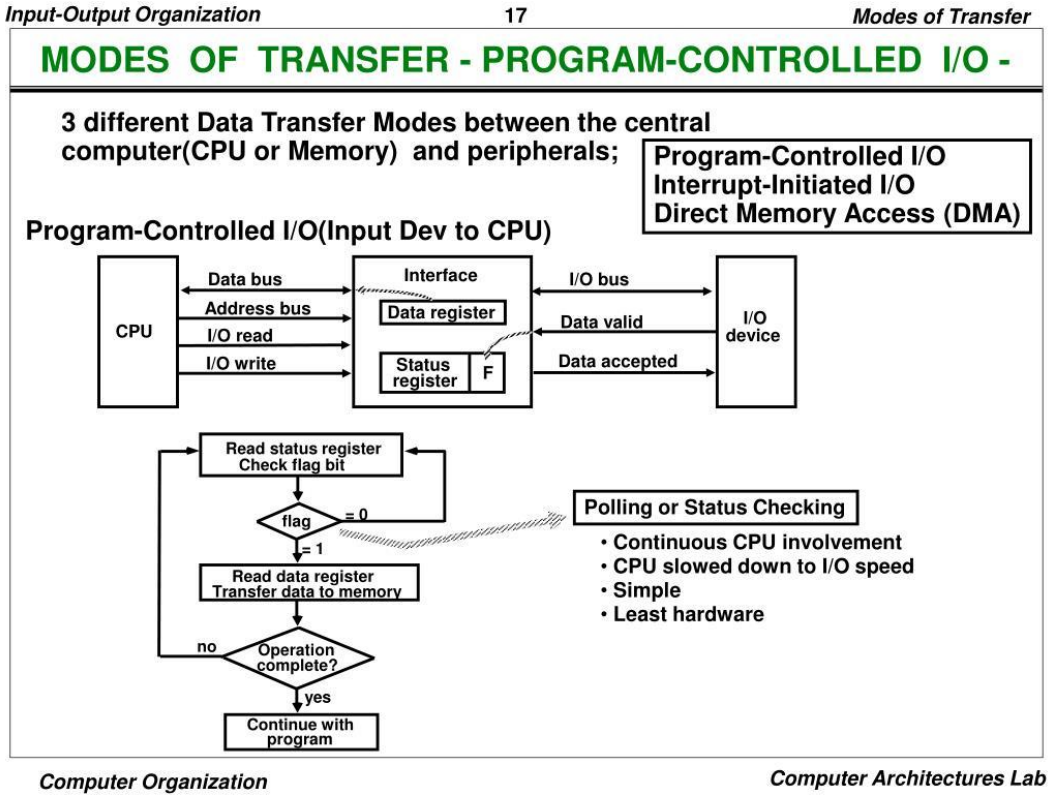
Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time
- Steps
  - CPU requests I/O operation
  - I/O module performs operation
  - I/O module sets status bits
  - CPU checks status bits periodically
  - I/O module does not inform CPU directly
  - I/O module does not interrupt CPU
  - CPU may wait or come back later



I/O Operations

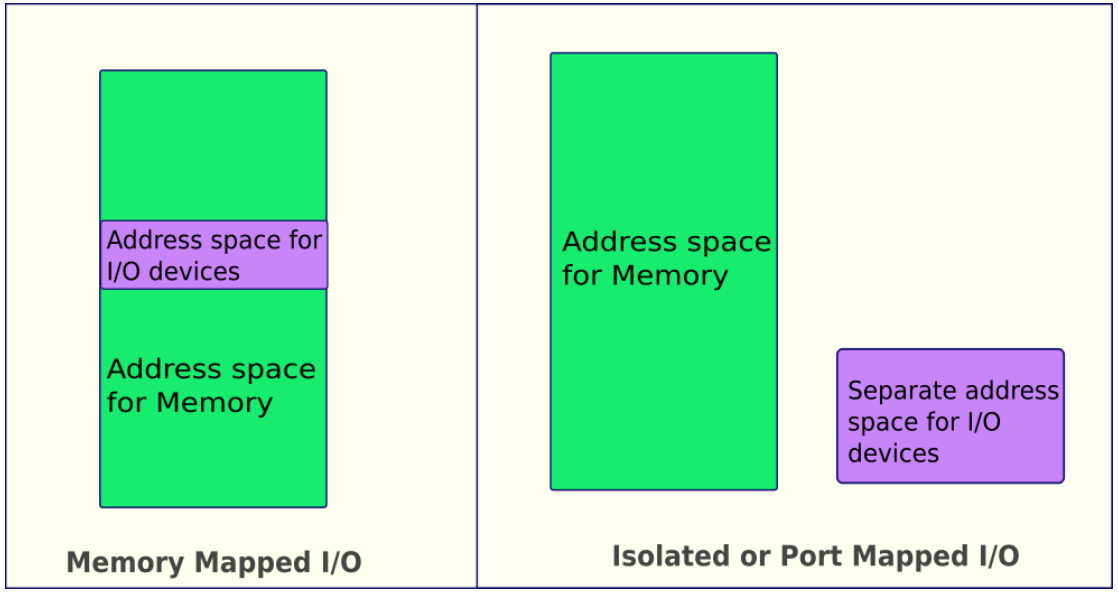
- I/O Commands
  - CPU issues address
    - Identifies module (& device if >1 per module)
  - CPU issues command
    - Control - telling module what to do
      - e.g. spin up disk
    - Test - check status
      - e.g. power? Error?
    - Read/Write
      - Module transfers data via buffer from/to device



- 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)

I/O Mapping

- 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



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

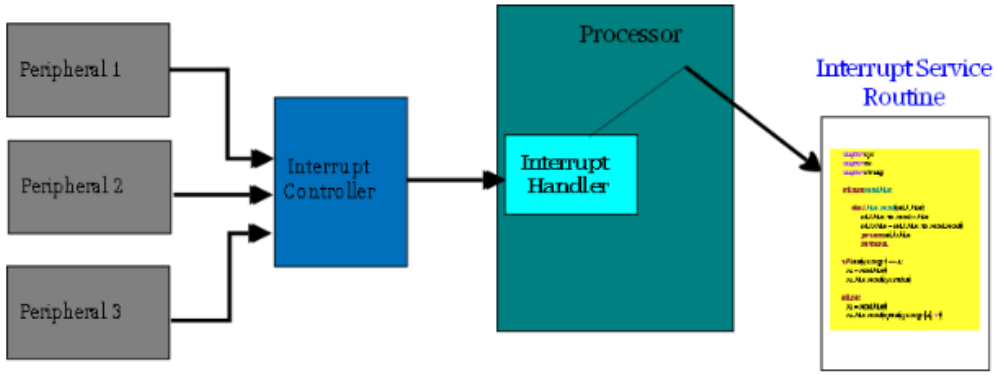
(a) Memory-mapped I/O

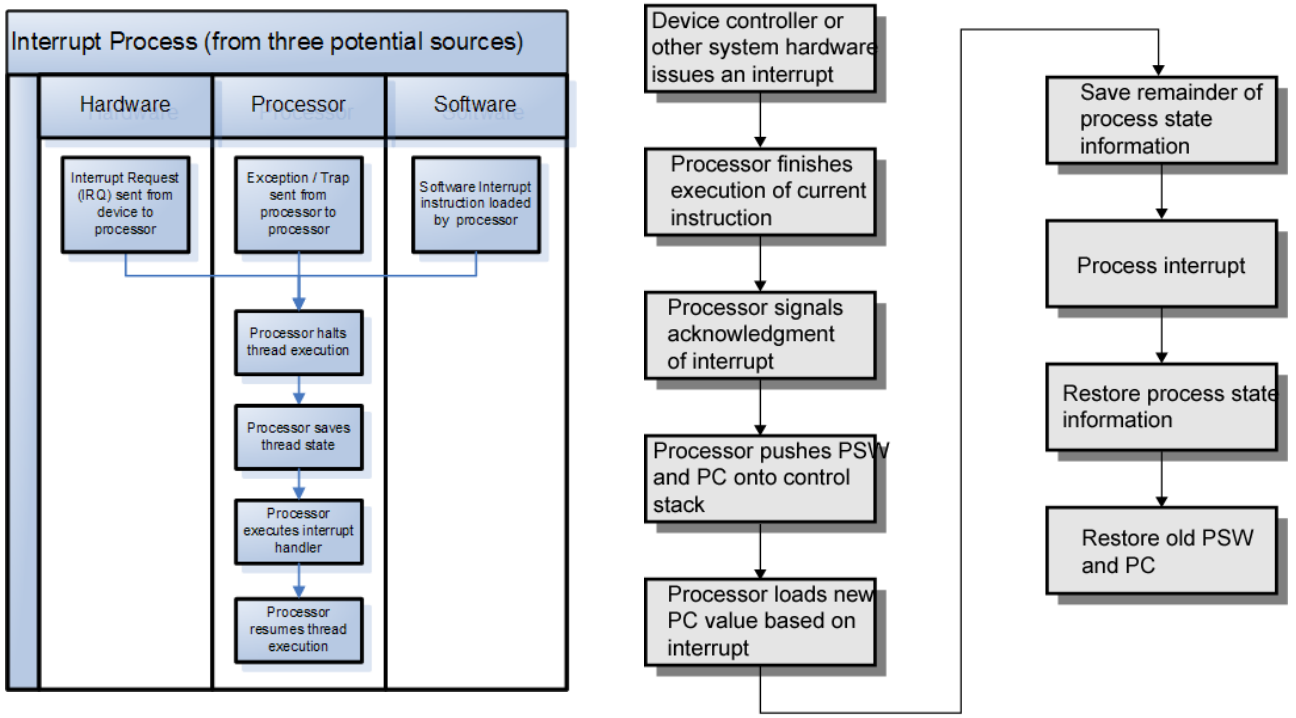
ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

Interrupt Driven I/O

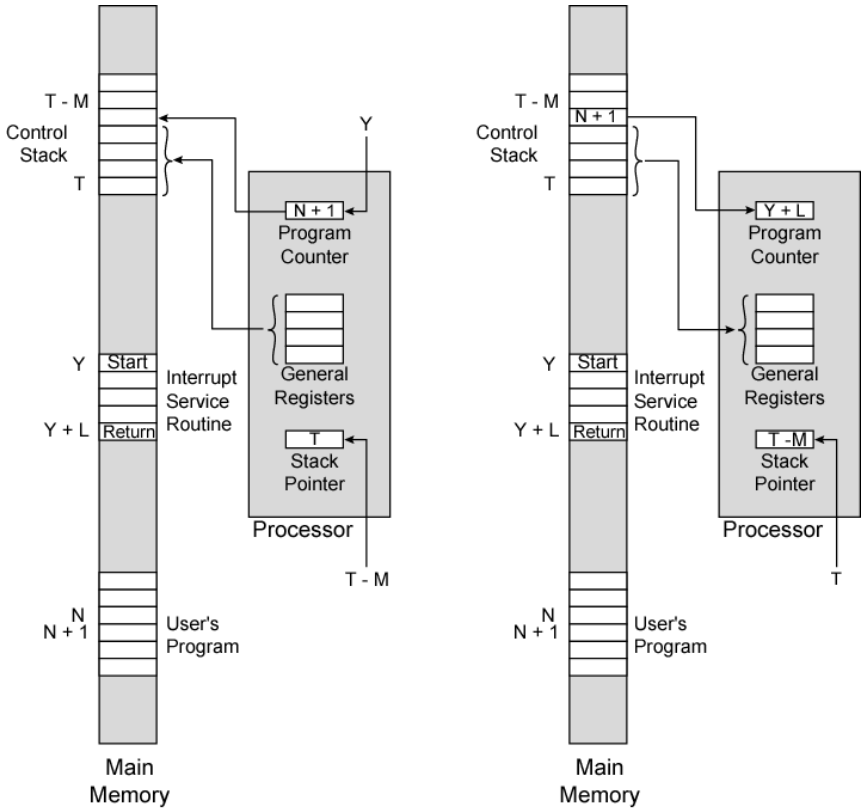
- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready
- Basic Steps
  - CPU issues read command
  - I/O module gets data from peripheral
    - while CPU does other work
  - I/O module interrupts CPU
  - CPU requests data
  - I/O module transfers data





CPU Viewpoint for Interrupt

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

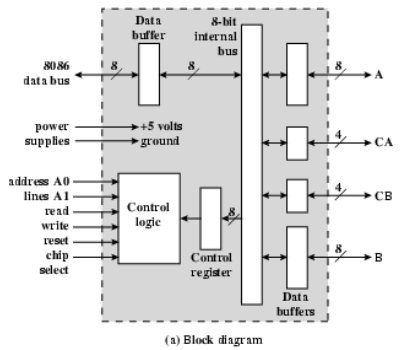


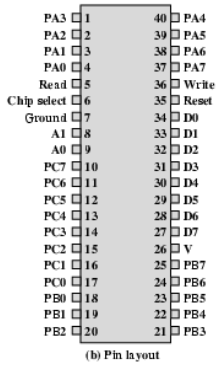
(a) Interrupt occurs after instruction at location N

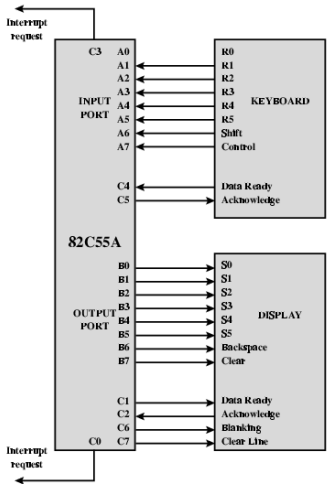
(b) Return from interrupt

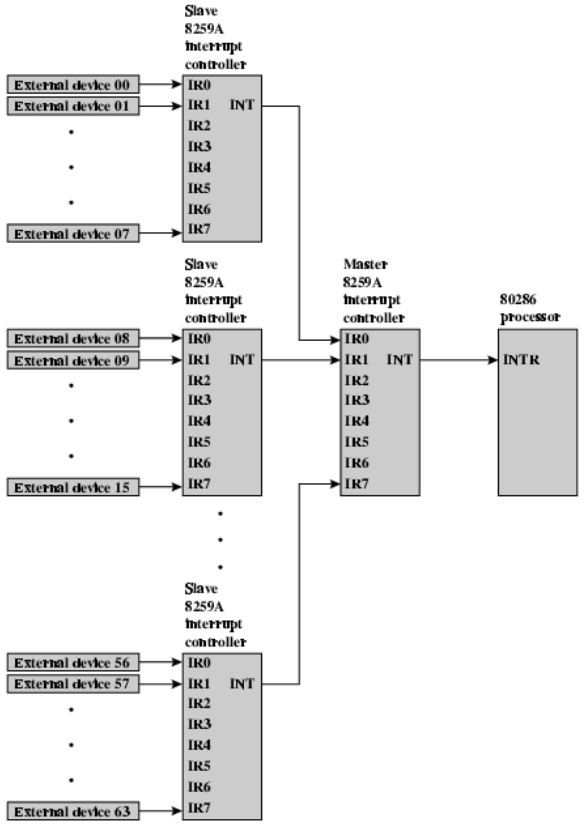
Interrupt I/O

- Interrupt Module
- Different line for each module
  - PC
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow
- Multiple interrupts
  - i.e. an interrupt handler being interrupted
  - Each interrupt line has a priority
  - Higher priority lines can interrupt lower priority lines



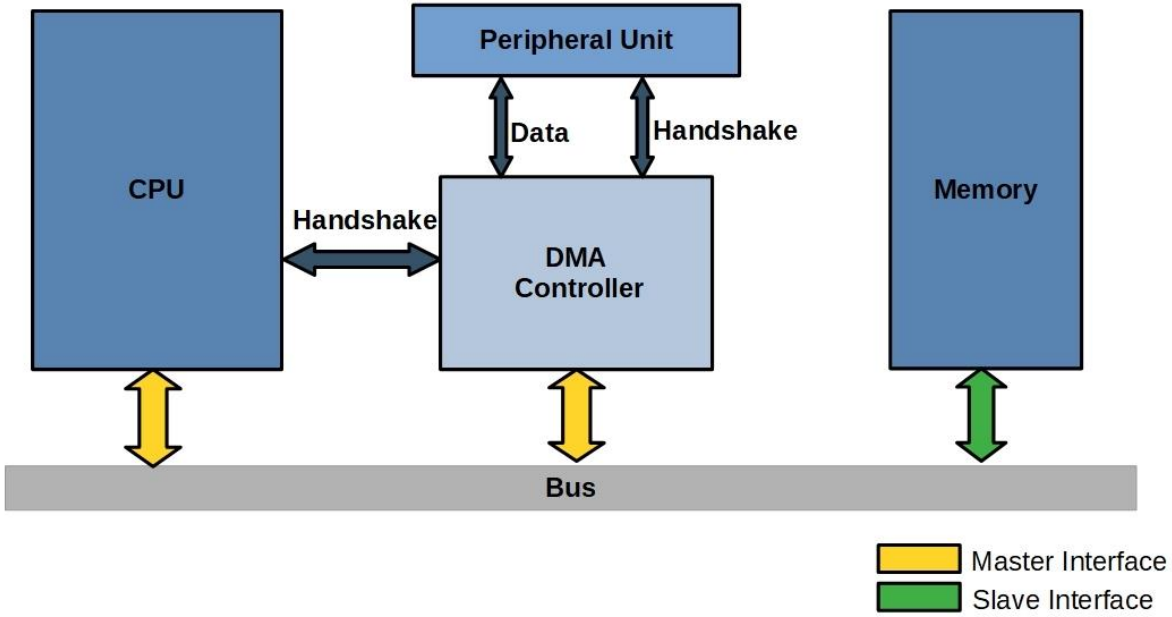


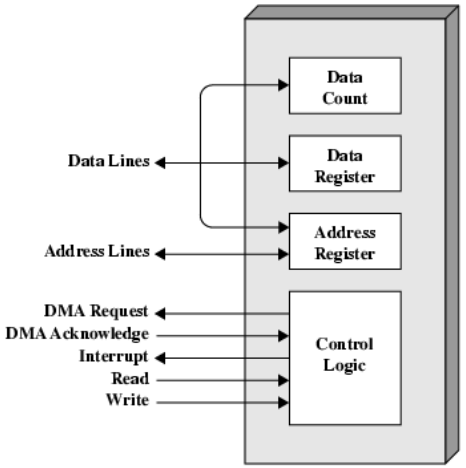




DMA - Direct Memory Access

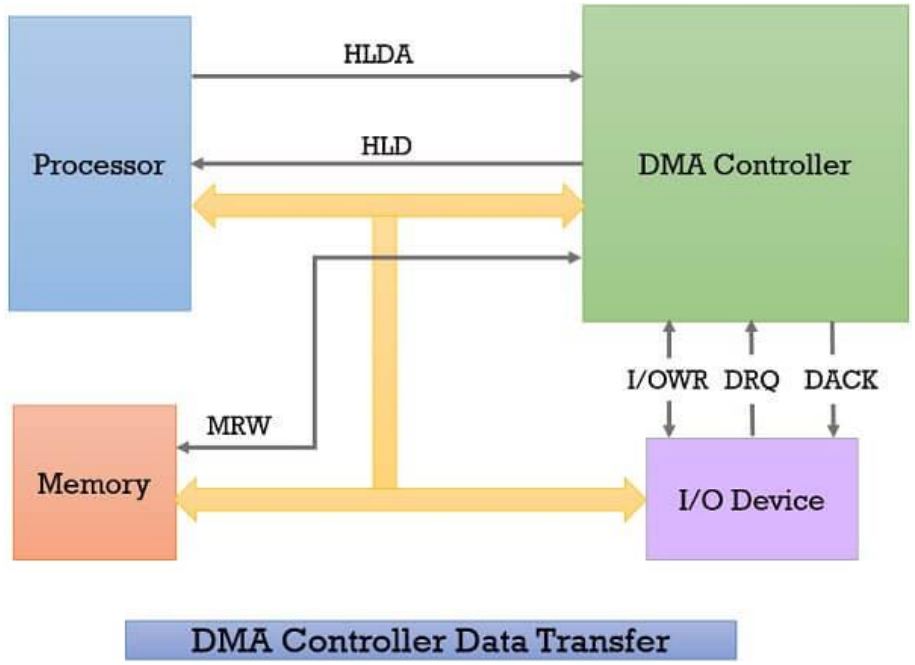
- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA is the answer
- DMA Functions
  - Additional Module (hardware) on bus
    - DMA Controller
  - DMA controller takes over from CPU for I/O





DMA Operation

- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU starts DMA Controller
  - CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished



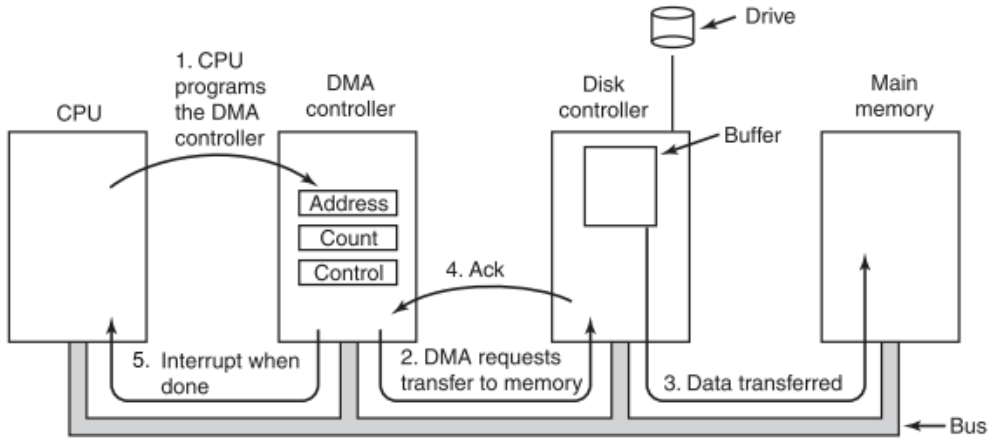
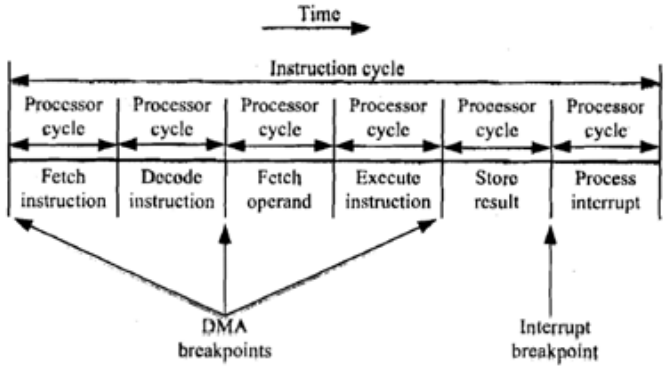
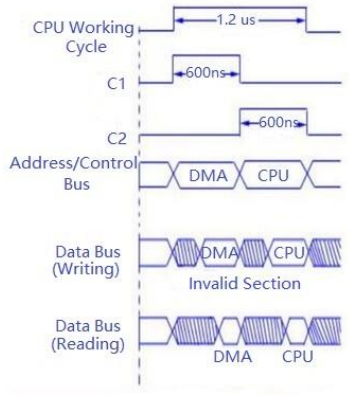


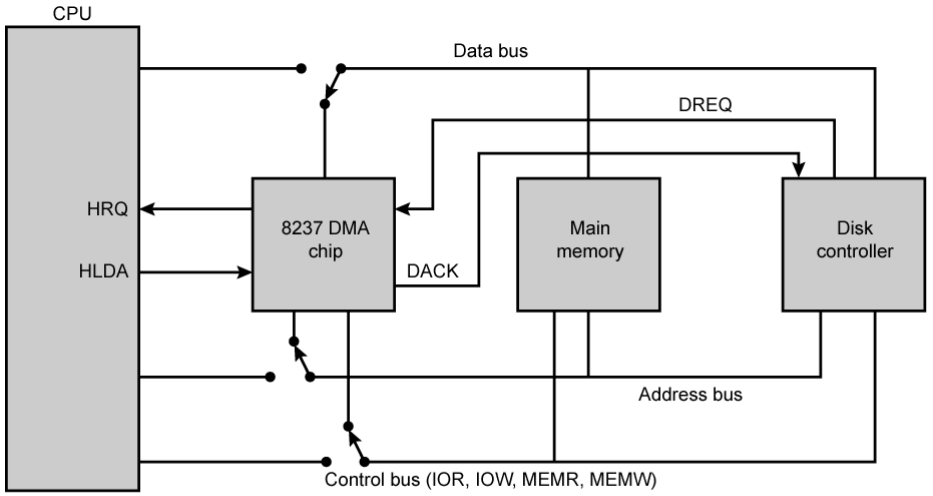
Figure 5-4. Operation of a DMA transfer.

DMA Transfer Cycle – Bus Stealing

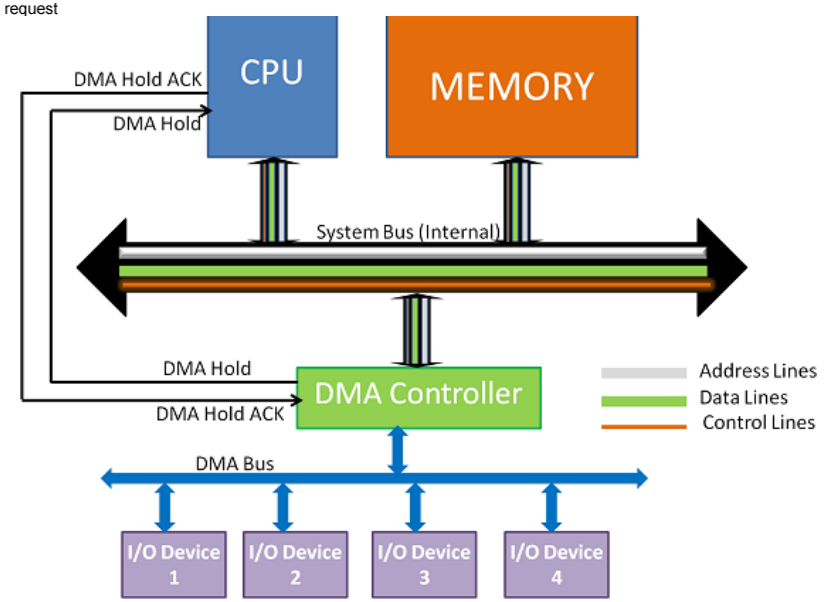
- DMA controller takes over bus for a cycle
  - DMA Bus request to CPU
  - CPU sends DMA Bus Acknowledge to DMA Controller
    - When CPU not uses bus
  - DMA Controller can control the bus
  - Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer





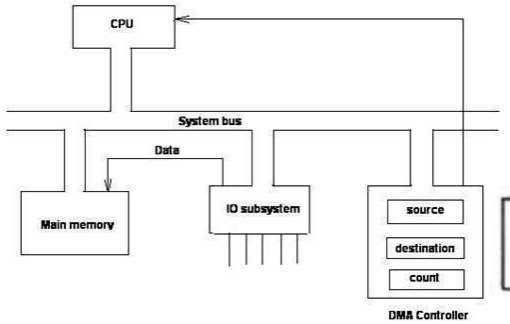


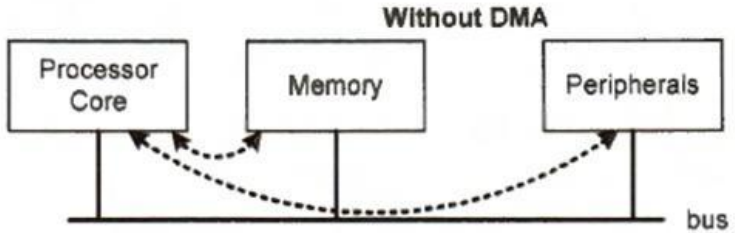
DACK = DMA acknowledge
DREQ = DMA request
HLDA = HOLD acknowledge
HRQ = HOLD request

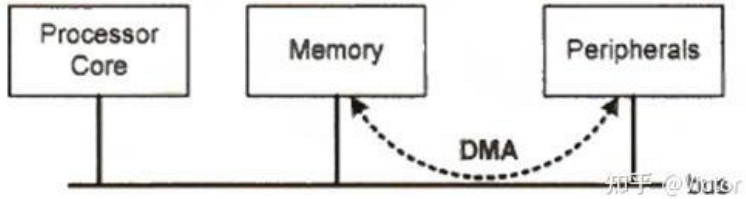


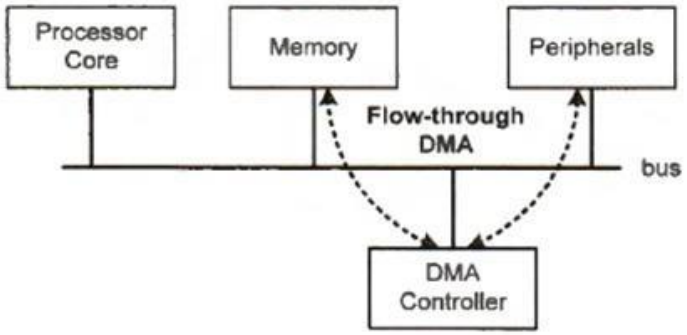
DMA Fly-By

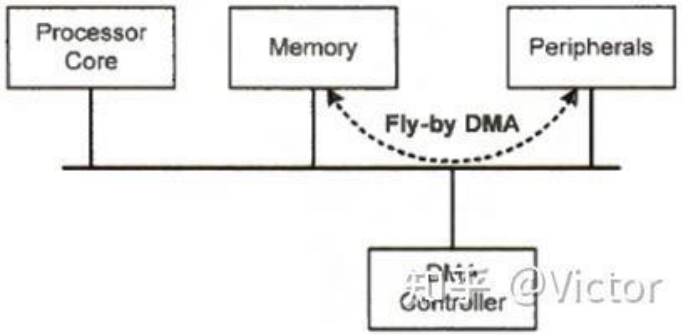
- While DMA using buses processor idle
- Processor using bus, DMA idle
  - Known as fly-by DMA controller
- Data does not pass through and is not stored in DMA chip
  - DMA only between I/O port and memory
  - Not between two I/O ports or two memory locations
- Can do memory to memory via register
- DMA Controller may contain multiple DMA channels
  - Programmed independently
  - Any one active
  - Numbered 0, 1, 2, 3, and …

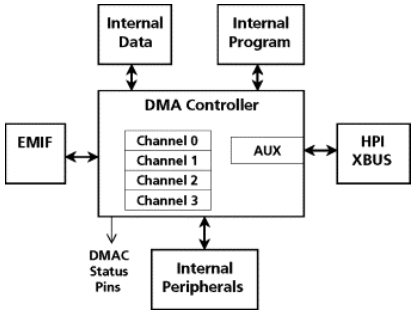






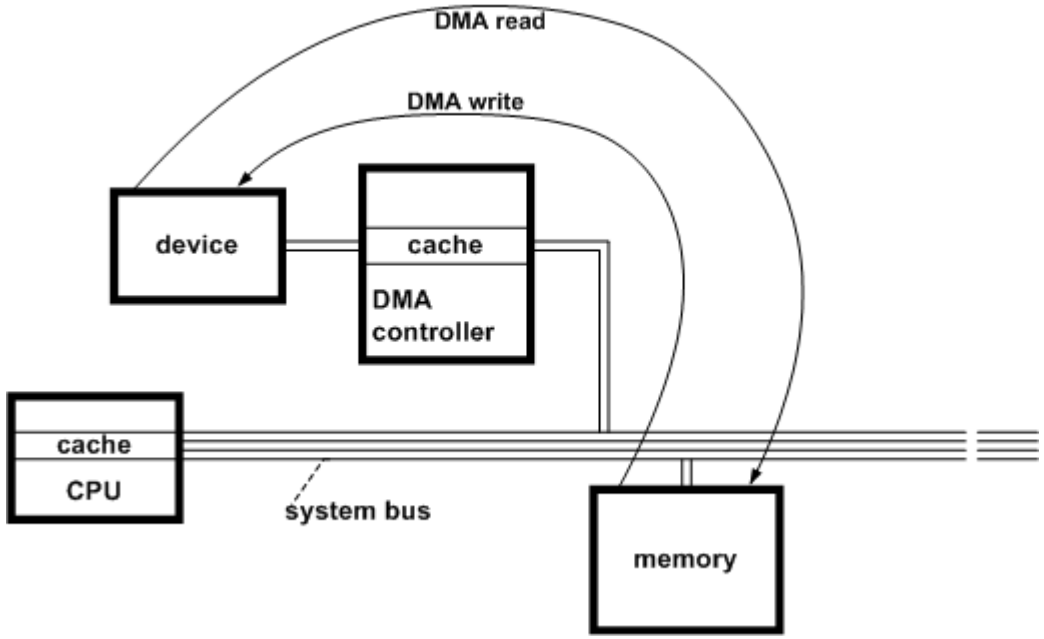






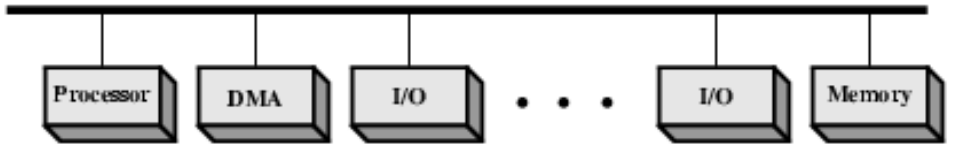
DMA Aside

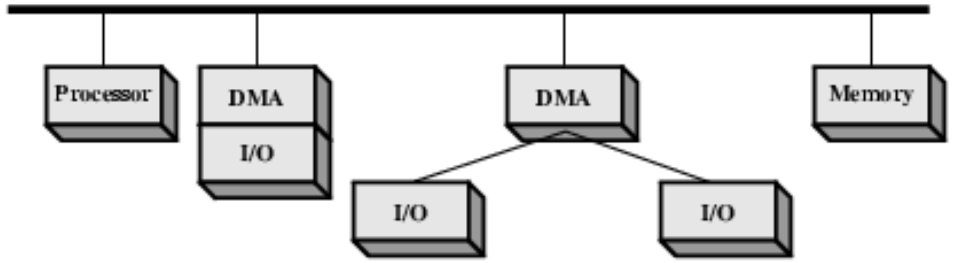
- What effect does caching memory have on DMA?
  - Cache is a copy of memory
- What about on board cache?
- Hint:
  - How much are the system buses available?
  - cache controller support DMA



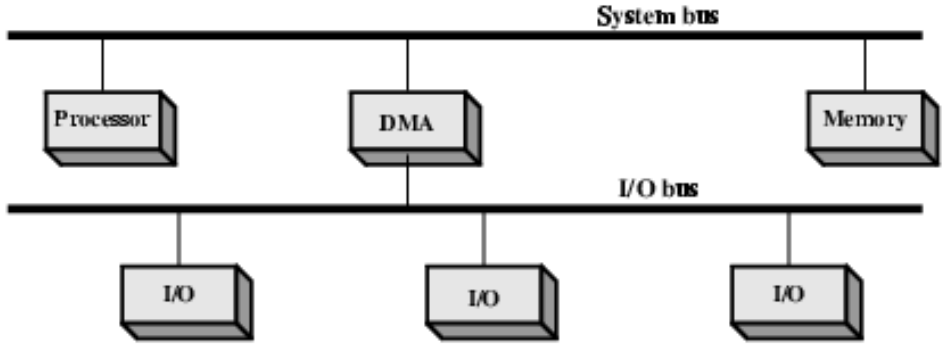
DMA Configurations

- Single Bus, Detached DMA controller
  - Each transfer uses bus twice
  - I/O to DMA then DMA to memory
  - CPU is suspended twice
- Single Bus, Integrated DMA controller
  - Controller may support >1 device
  - Each transfer uses bus once
    - DMA to memory
  - CPU is suspended once
- Separate I/O Bus
  - Bus supports all DMA enabled devices
  - Each transfer uses bus once
    - DMA to memory
  - CPU is suspended once





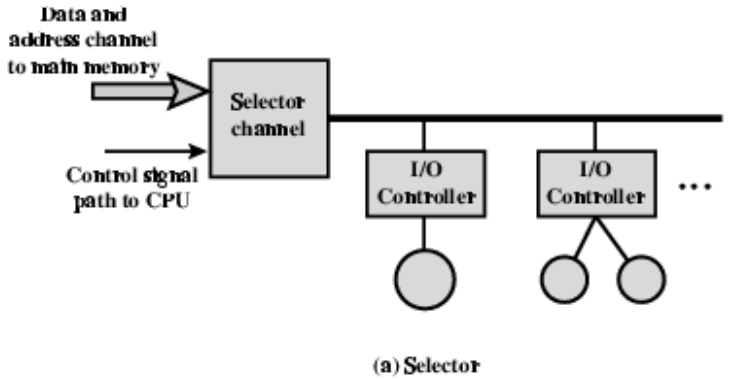
(b) Single-bus, Integrated DMA-I/O



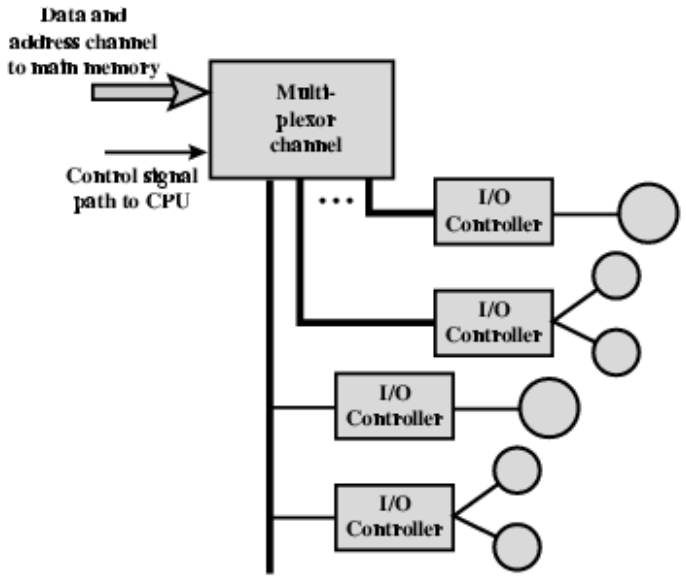
(c) I/O bus

I/O Channels

- I/O devices getting more sophisticated
  - e.g. 3D graphics cards
- CPU instructs I/O controller to do transfer
- I/O controller does entire transfer
- Improves speed
  - Takes load off CPU
  - Dedicated processor is faster



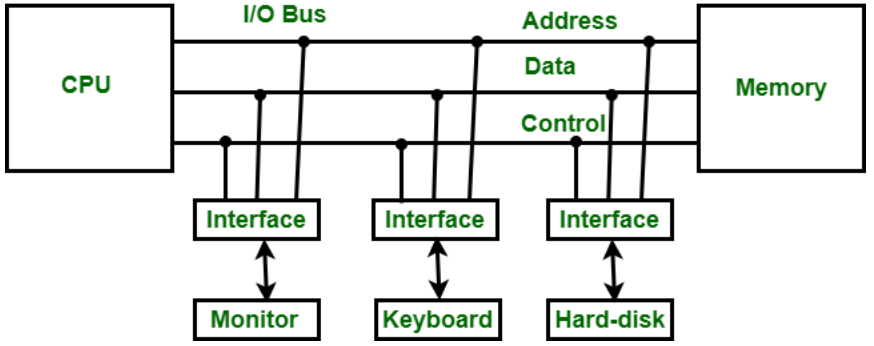
(a) Selector

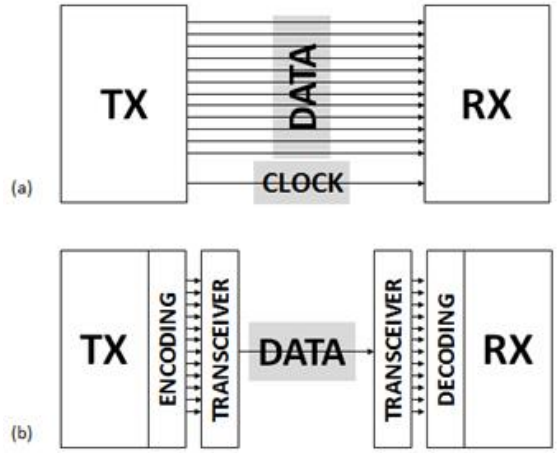


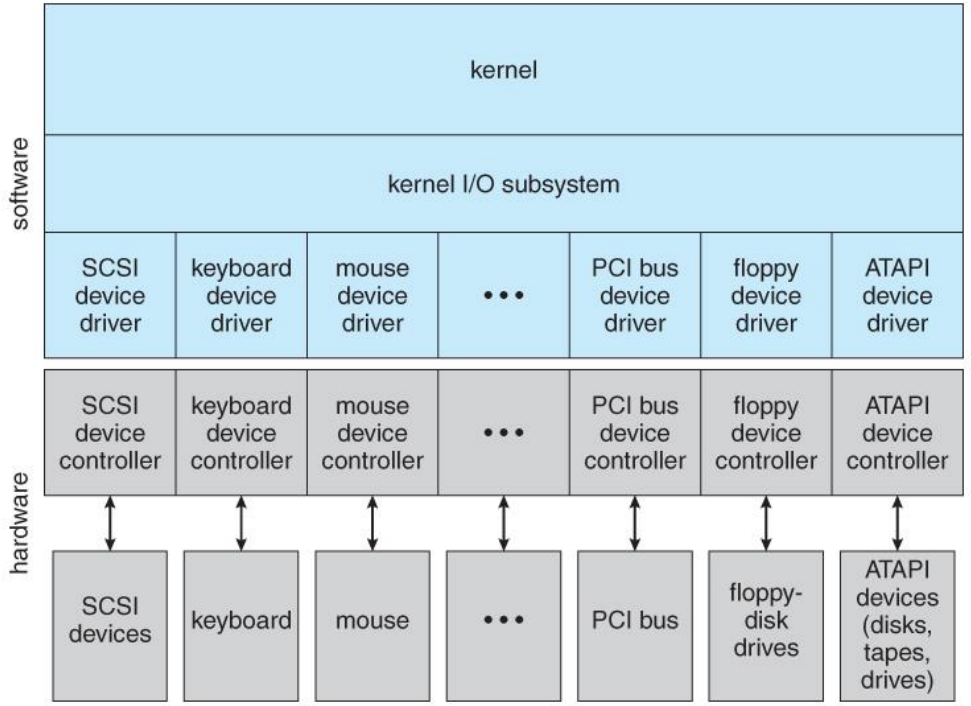
(b) Multiplexor

I/O Interfacing

- Connecting devices together
- Bit of wire?
- Dedicated processor/memory/buses?
- Hardware with Software
- I/O Interface Classification
  - Parallel vs Serial
  - Functions
    - Disk
    - Display
    - Keyboard/Mouse
    - Universal
    - Etc.
  - Internal vs External
  - Distances vs Device Number



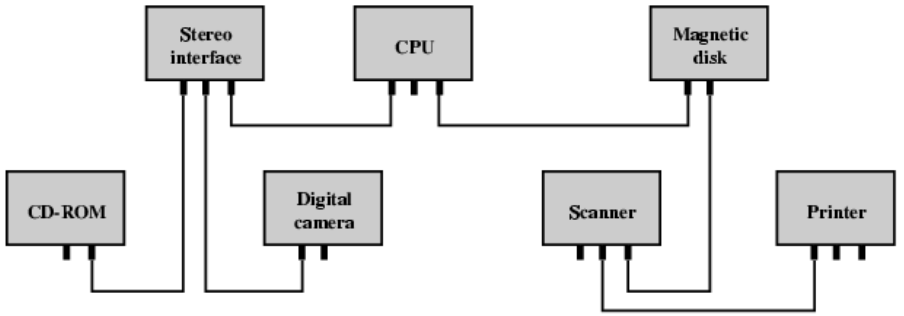


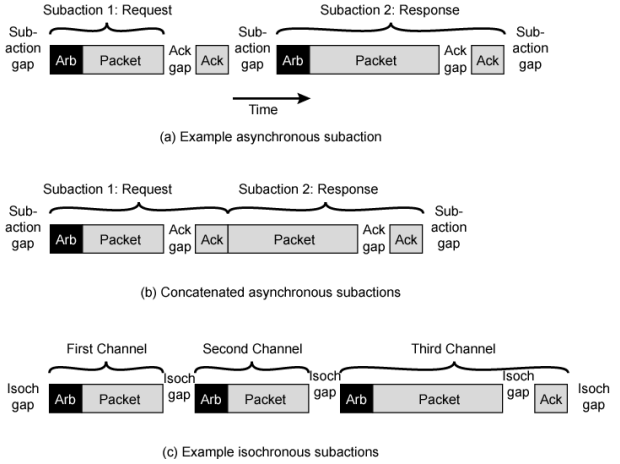


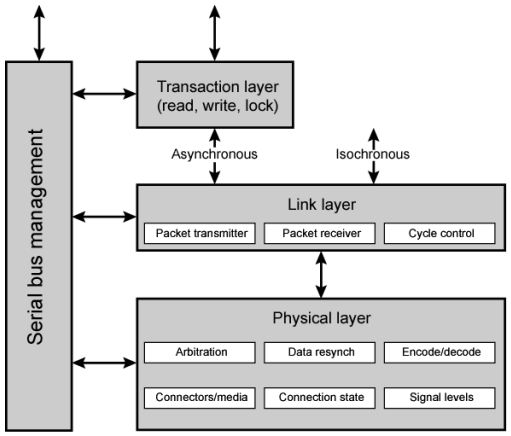
Example: IEEE 1394 FireWire

- High performance serial bus
- Fast
- Low cost
- Easy to implement
- Also being used in digital cameras, VCRs and TV
- Configuration
  - Daisy chain
  - Up to 63 devices on single port
    - Really 64 of which one is the interface itself
  - Up to 1022 buses can be connected with bridges
  - Automatic configuration
  - No bus terminators
  - May be tree structure

Simple FireWire Configuration

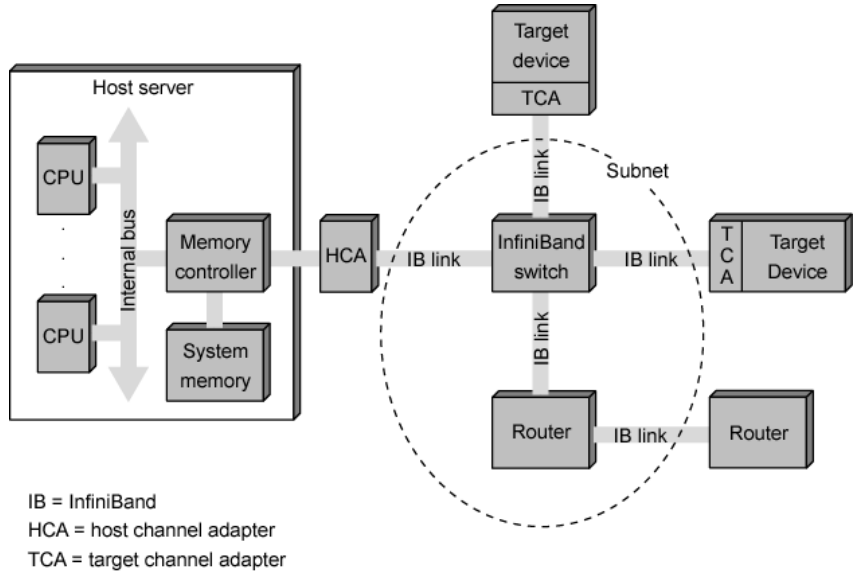


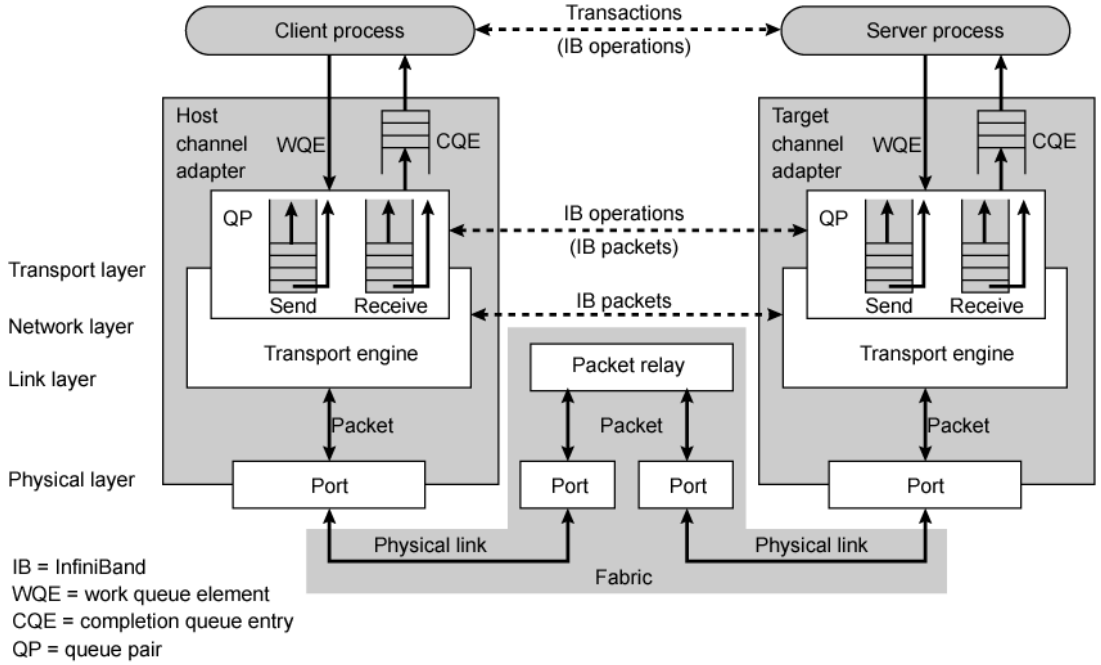




Example: InfiniBand

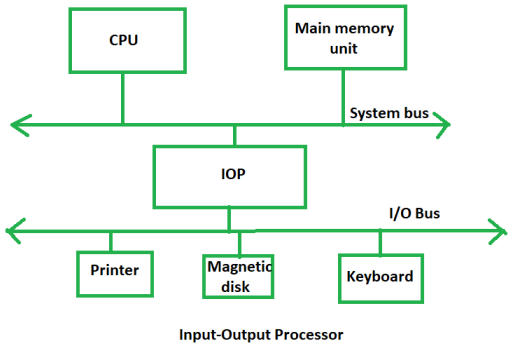
- I/O specification aimed at high end servers
  - Merger of Future I/O (Cisco, HP, Compaq, IBM) and Next Generation I/O (Intel)
- Version 1 released early 2001
- Architecture and spec. for data flow between processor and intelligent I/O devices
- Intended to replace PCI in servers
- Increased capacity, expandability, flexibility
- Architecture
  - Remote storage, networking and connection between servers
  - Attach servers, remote storage, network devices to central fabric of switches and links
  - Greater server density
  - Scalable data centre
  - Independent nodes added as required
  - I/O distance from server up to
    - 17m using copper
    - 300m multimode fibre optic
    - 10km single mode fibre
  - Up to 30Gbps

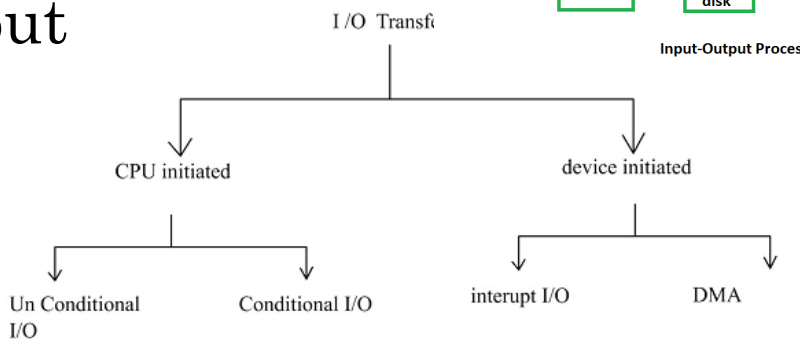


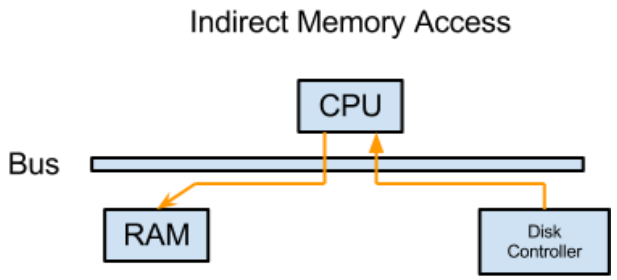


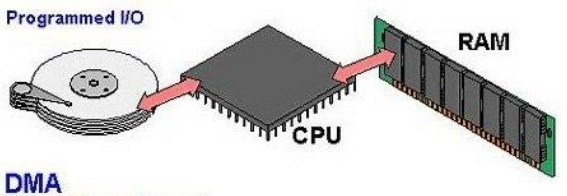
Conclusion

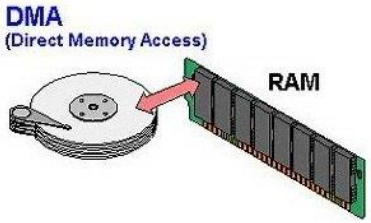
- Input / Output Transfer Data Into or out from Computer Systems => Memory
- **Input / Output Techniques**
  - Programmed I/O – CPU Intensive - Slow
  - Interrupt I/O – CPU Faster
  - DMA – Fastest with HW Controller
- I/O Interfaces connected I/O Modules to computer system
- Many I/O Interfacing Standards

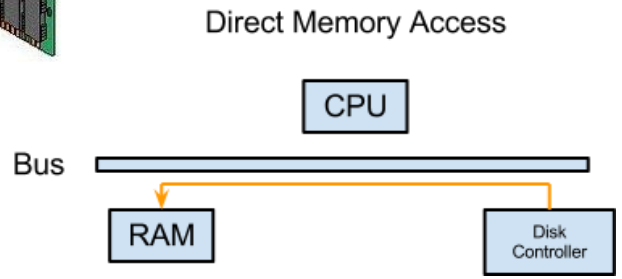












END

Questions?