电子科技大学
格拉斯哥学院
Glasgow College, UESTC

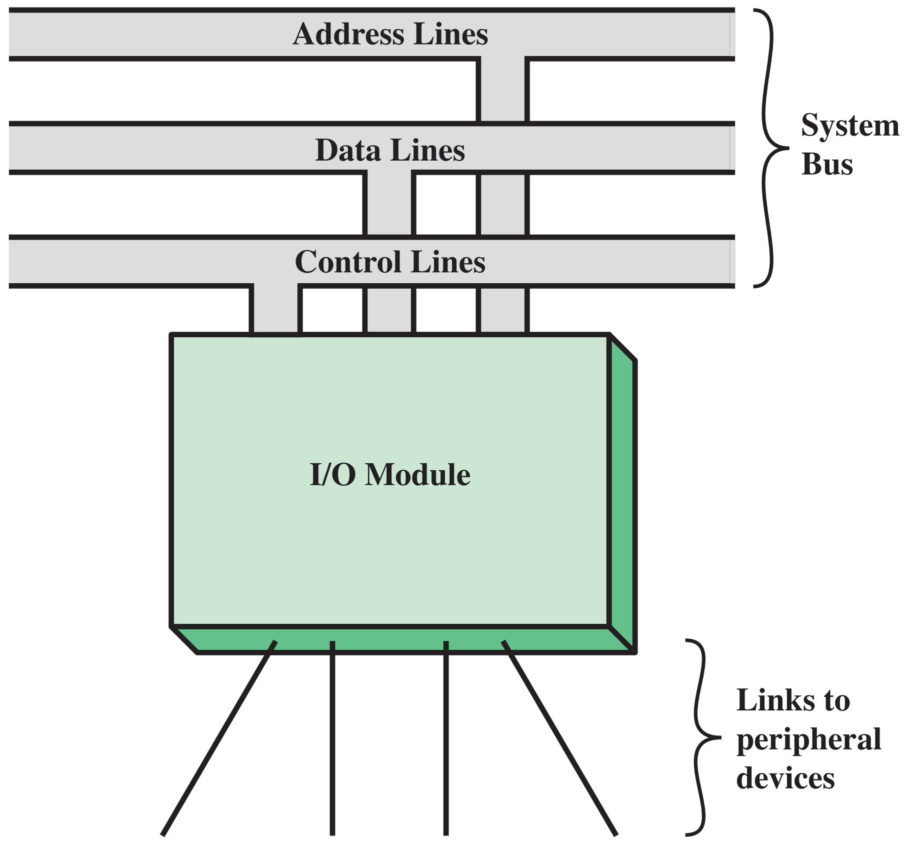
UESTC4019: Real-Time Computer Systems and Architecture

Lecture 10

Input and Output Module

Generic Model of an I/O Module

- An I/O module has two major functions:
  - Interface to the processor and memory via the system bus or central switch
  - Interface to one or more peripheral devices by tailored data links
- In this session, we will examine:
  - The various ways in which the I/O function can be performed in cooperation with the processor and memory: the internal I/O interface
  - The external I/O interface, between the I/O module and the outside world



External Devices

- Provide a means of exchanging data between the external environment and the computer

- Attach to the computer by a link to an I/O module
  - The link is used to exchange control, status, and data between the I/O module and the external device

- Peripheral device
  - An external device connected to an I/O module

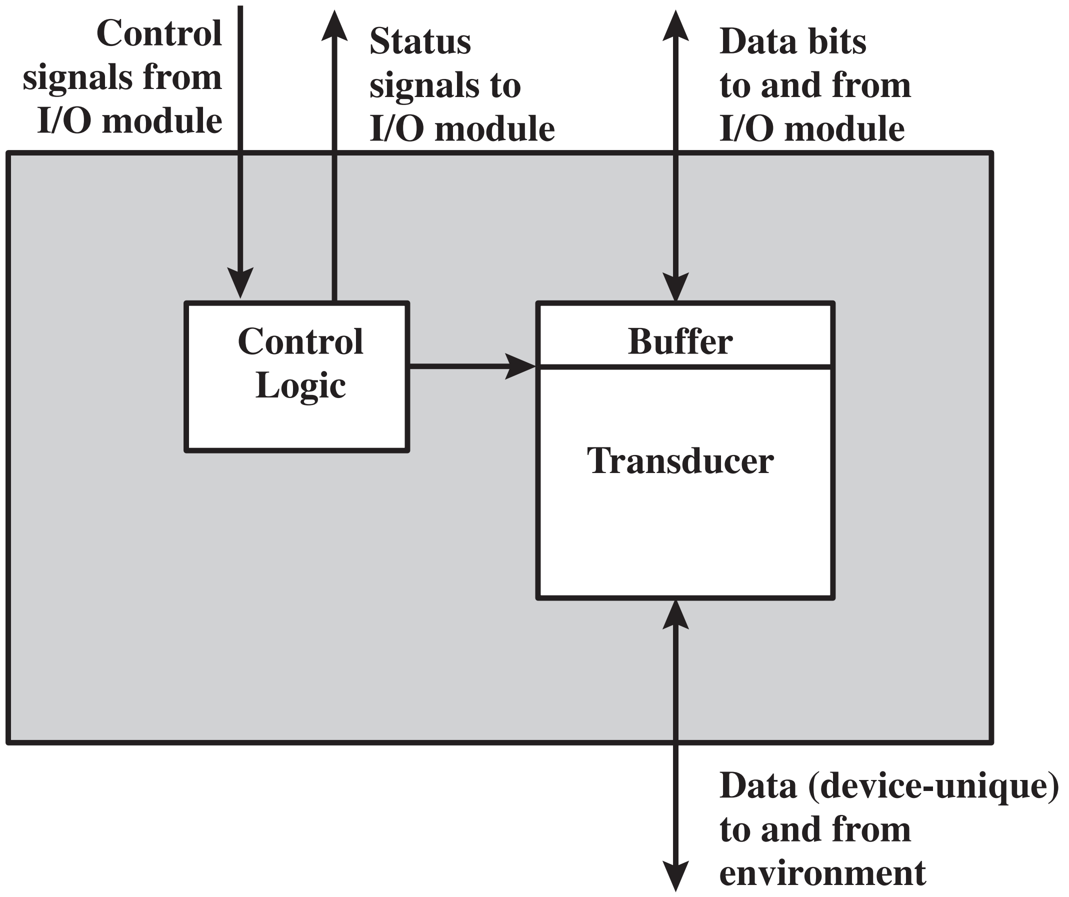
Three categories

- Human readable
  - Suitable for communicating with the computer user
  - Video display terminals (VDTs), printers

- Machine readable
  - Suitable for communicating with equipment
  - Magnetic disk and tape systems, sensors and actuators

- Communication
  - Suitable for communicating with remote devices such as a terminal, a machine readable device, or another computer

Block Diagram of an External Device


Control signals from I/O module
Status signals to I/O module
Data bits to and from I/O module
Control Logic
Buffer
Transducer
Data (device-unique) to and from environment

Keyboard/Monitor (1 of 2)

- International Reference Alphabet (IRA)
- Basic unit of exchange is the character
  - Associated with each character is a code
  - Each character in this code is represented by a unique 7-bit binary code
    - 128 different characters can be represented
- Characters are of two types:
  - Printable
    - Alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen
  - Control
    - Have to do with controlling the printing or displaying of characters
    - Example is carriage return
    - Other control characters are concerned with communications procedures

Keyboard/Monitor (2 of 2)

- Keyboard/ Monitor :
  - Most common means of computer/user interaction
  - User provides input through the keyboard
  - The monitor displays data provided by the computer

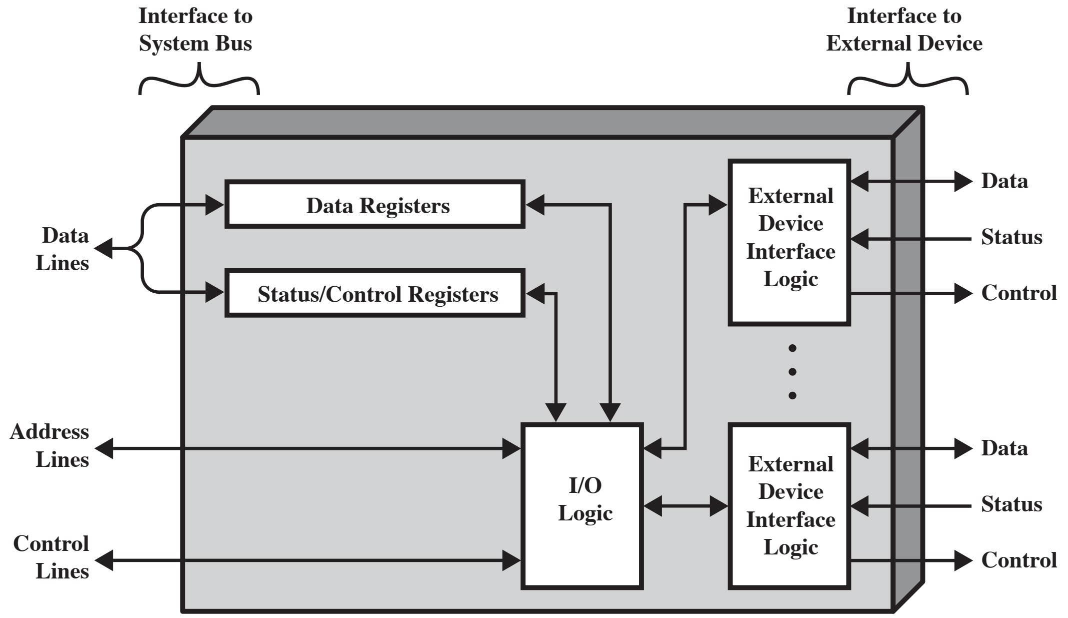
- Keyboard Codes
  - When the user depresses a key it generates an electronic signal that is interpreted by the transducer in the keyboard and translated into the bit pattern of the corresponding IRA code
  - This bit pattern is transmitted to the I/O module in the computer
  - On output, IRA code characters are transmitted to an external device from the I/O module
  - The transducer interprets the code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function

The Major Functions for an I/O Module Fall Into the Following Categories

- Control and timing
  - Coordinates the flow of traffic between internal resources and external devices
- Processor communication
  - Involves command decoding, data, status reporting, address recognition
- Device communication
  - Involves commands, status information, and data
- Data buffering
  - Performs the needed buffering operation to balance device and memory speeds
- Error detection
  - Detects and reports transmission errors

Block Diagram of an I/O Module (1 of 2)

- I/O modules vary considerably in complexity and the number of external devices that they control. Figure provides a general block diagram of an I/O module.
- The module connects to the rest of the computer through a set of signal lines (e.g., system bus lines).
- Data transferred to and from the module are buffered in one or more data registers.



Block Diagram of an I/O Module (2 of 2)

- There may also be one or more status registers that provide current status information. A status register may also function as a control register, to accept detailed control information from the processor.
- The logic within the module interacts with the processor via a set of control lines. The processor uses the control lines to issue commands to the I/O module.
- Some of the control lines may be used by the I/O module (e.g., for arbitration and status signals).
- The module must also be able to recognize and generate addresses associated with the devices it controls. Each I/O module has a unique address or, if it controls more than one external device, a unique set of addresses.
- Finally, the I/O module contains logic specific to the interface with each device that it controls.

I/O Techniques (1 of 2)

- Programmed I/O
    - Data are exchanged between the processor and the I/O module
    - Processor executes a program that gives it direct control of the I/O operation
    - When the processor issues a command it must wait until the I/O operation is complete
    - If the processor is faster than the I/O module this is wasteful of processor time
- Interrupt-driven I/O
    - Processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work
- Direct memory access (DMA)
    - The I/O module and main memory exchange data directly without processor involvement

I/O Techniques (2 of 2)

	No Interrupts	**Use of Interrupts**
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access(DMA)

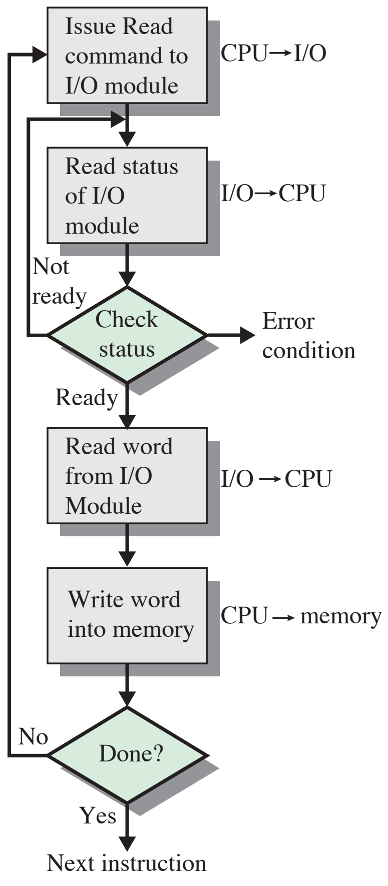
Table indicates the relationship among three I/O techniques

I/O Commands

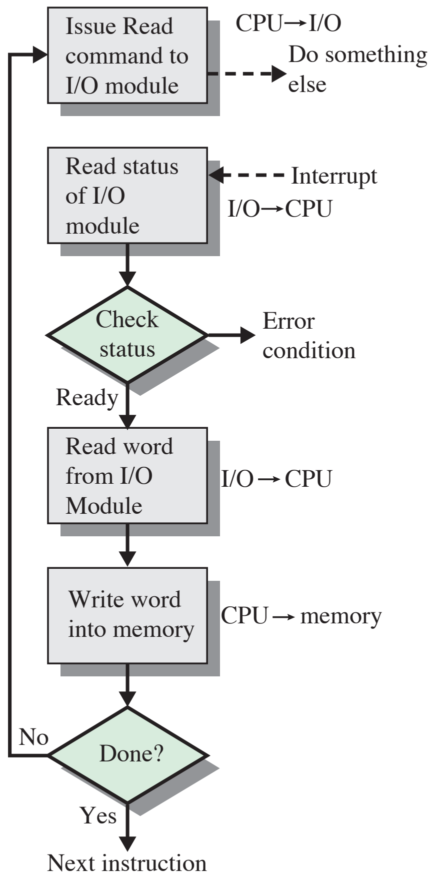
There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:

1) Control
   - used to activate a peripheral and tell it what to do
2) Test
   - used to test various status conditions associated with an I/O module and its peripherals
3) Read
   - causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer
4) Write
   - causes the I/O module to take an item of data from the data bus and subsequently transmit that data item to the peripheral

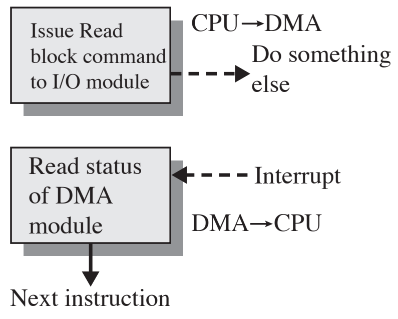
Three Techniques for Input of a Block of Data



(a) Programmed I/O



(b) Interrupt-driven I/O


Issue Read block command to I/O module
CPU→DMA
Do something else
Read status of DMA module
Interrupt
DMA→CPU
Next instruction

(c) Direct memory access

I/O Instructions

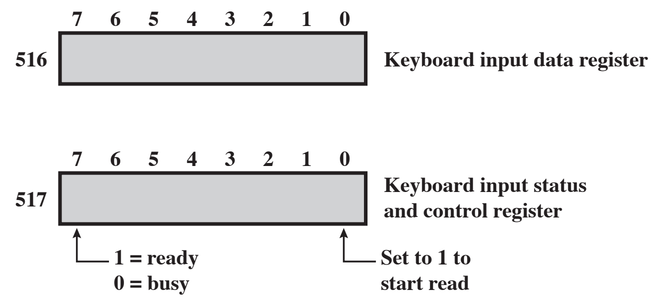
- With programmed I/O there is a close correspondence between the I/O-related instructions that the processor fetches from memory and the I/O commands that the processor issues to an I/O module to execute the instructions
  - The form of the instruction depends on the way in which external devices are addressed
- Each I/O device connected through I/O modules is given a unique identifier or address
  - When the processor issues an I/O command, the command contains the address of the desired device
  - Thus each I/O module must interpret the address lines to determine if the command is for itself
- Memory-mapped I/O
  - There is a single address space for memory locations and I/O devices
  - A single read line and a single write line are needed on the bus

I/O Mapping Summary

- 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

Memory-Mapped and Isolated I/O

- Figure (a) shows how the interface for a simple input device such as a terminal keyboard might appear to a programmer using memory-mapped I/O
- Two addresses are dedicated to keyboard input from a particular terminal. Address 516 refers to the data register and address 517 refers to the status register, which also functions as a control register for receiving processor commands
- The program shown will read 1 byte of data from the keyboard into an accumulator register in the processor. Note that the processor loops until the data byte is available
- With isolated I/O in Figure (b), the I/O ports are accessible only by special I/O commands, which activate the I/O command lines on the bus



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 (1 of 2)

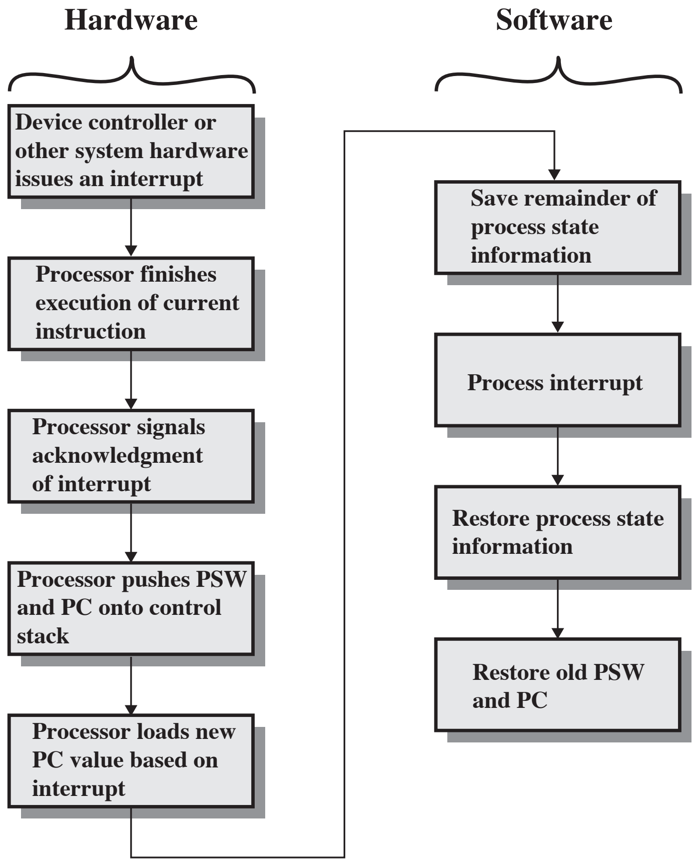
- The problem with programmed I/O is that the processor has to wait a long time for the I/O module to be ready for either reception or transmission of data
- An alternative, know as interrupt-Driven I/O, is for the processor to issue an I/O command to a module and then go on to do some other useful work
- The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor executes the data transfer and resumes its former processing
- From the point of view of the I/O module.
    - For input, the I/O module receives a READ command from the processor. The I/O module then proceeds to read data in from an associated peripheral.
    - Once the data are in the module's data register, the module signals an interrupt to the processor over a control line. The module then waits until its data are requested by the processor.

Interrupt-Driven I/O (2 of 2)

  - When the request is made, the module places its data on the data bus and is then ready for another I/O operation
- From the processor’s point of view:
  - The processor issues a READ command. It then goes off and does something else (e.g., the processor may be working on several different programs at the same time)
  - At the end of each instruction cycle, the processor checks for interrupts. When the interrupt from the I/O module occurs, the processor saves the context (e.g., program counter and processor registers) of the current program and processes the interrupt
  - In this case, the processor reads the word of data from the I/O module and stores it in memory. It then restores the context of the program it was working on (or some other program) and resumes execution

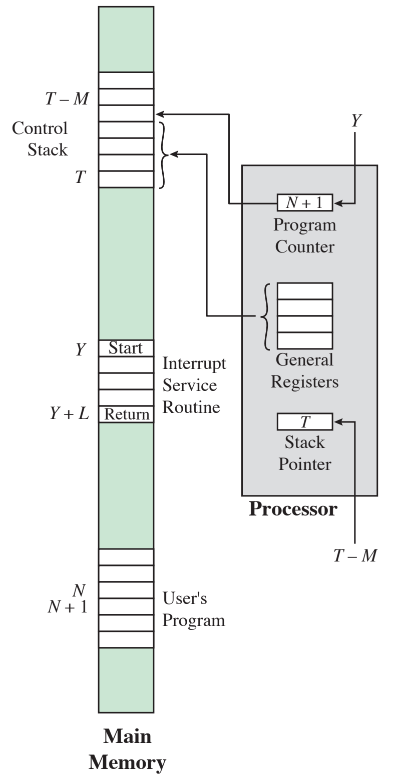
Simple Interrupt Processing

- The occurrence of an interrupt triggers a number of events, both in the processor hardware and in software. Figure shows a typical sequence.
- When an I/O device completes an I/O operation, the following sequence of hardware events occurs:
    1) The device issues an interrupt signal to the processor.
    2) The processor finishes execution of the current instruction before responding to the interrupt.
    3) The processor tests for an interrupt and sends an acknowledgment signal to the device that issued the interrupt. The acknowledgment allows the device to remove its interrupt signal.

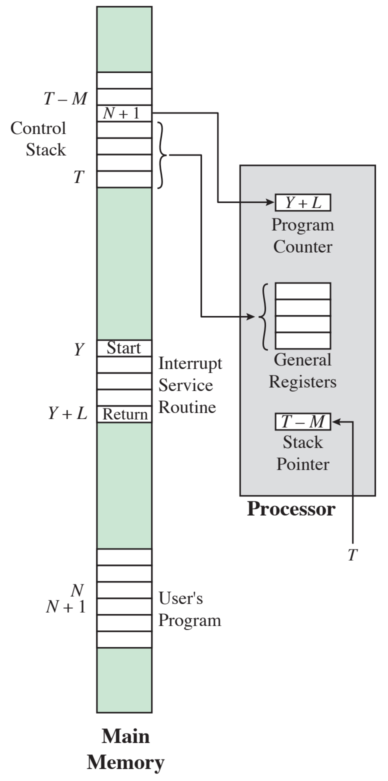

Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

Changes in Memory and Register for an Interrupt

4. The processor now needs to prepare to transfer control to the interrupt routine.
5. The processor now loads the program counter with the entry location of the interrupt-handling program that will respond to this interrupt.
6. At this point, the program counter and PSW relating to the interrupted program have been saved on the system stack.
7. The interrupt handler next processes the interrupt
8. When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers
9. The final act is to restore the PSW and program counter values from the stack



(a) Interrupt occurs after instruction at location N



(b) Return from interrupt

Design Issues

- Two design issues arise in implementing interrupt I/O:
  - Because there will be multiple I/O modules how does the processor determine which device issued the interrupt?
  - If multiple interrupts have occurred how does the processor decide which one to process?

Device Identification (1 of 2)

Four general categories of techniques are in common use:

- Multiple interrupt lines
  - Between the processor and the I/O modules
  - Most straightforward approach to the problem
  - Consequently even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it

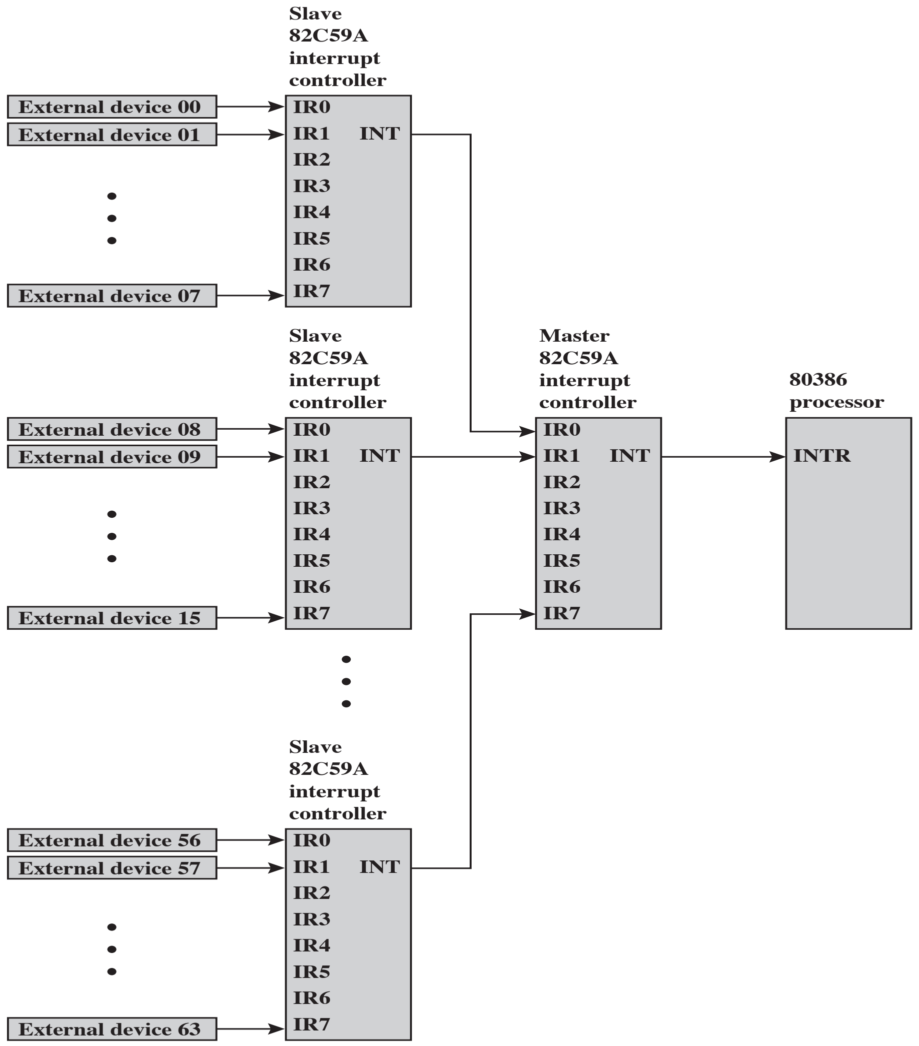
- Software poll
  - When processor detects an interrupt it branches to an interrupt-service routine (ISR) whose job is to poll each I/O module to determine which module caused the interrupt
  - Time consuming

Device Identification (2 of 2)

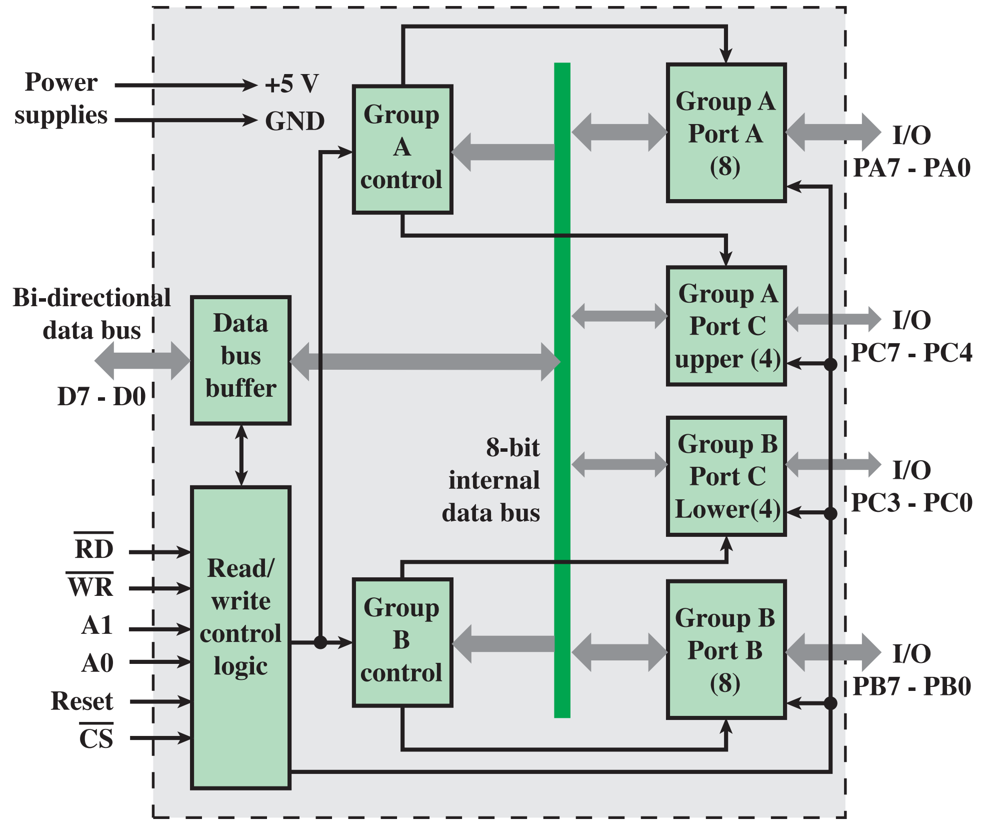
- Daisy Chain (hardware poll, vectored)
  - The interrupt acknowledge line is daisy chained through the modules
  - Vector - address of the I/O module or some other unique identifier
  - Vectored interrupt - processor uses the vector as a pointer to the appropriate device-service routine, avoiding the need to execute a general interrupt-service routine first

- Bus Arbitration (vectored)
  - An I/O module must first gain control of the bus before it can raise the interrupt request line
  - When the processor detects the interrupt it responds on the interrupt acknowledge line
  - Then the requesting module places its vector on the data lines

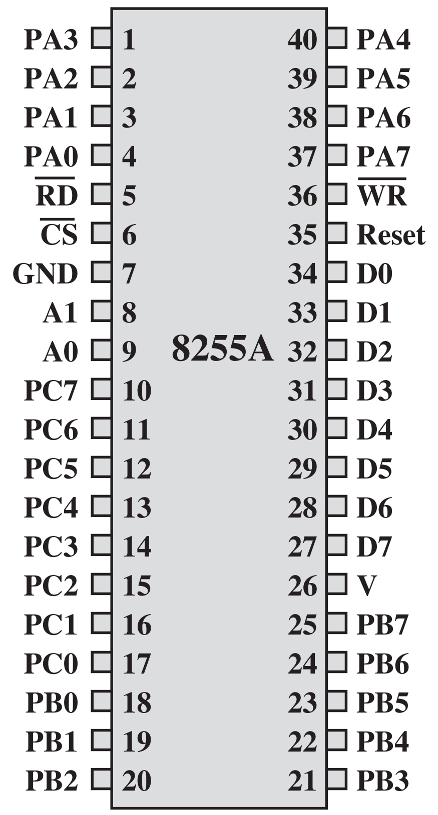
Use of the 82C59A Interrupt Controller



The Intel 8255A Programmable Peripheral Interface

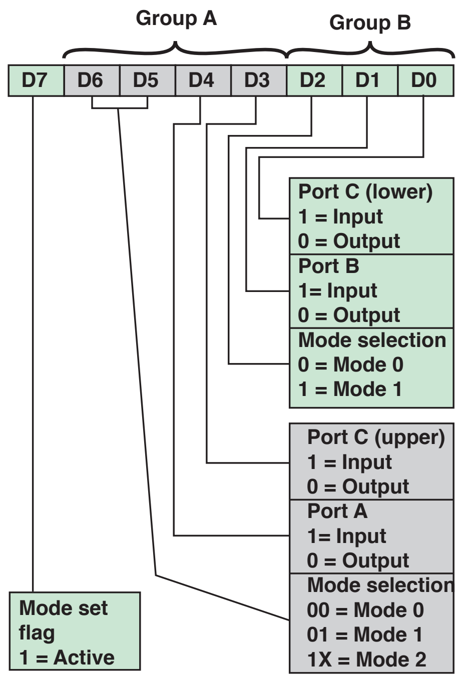


(a) Block diagram

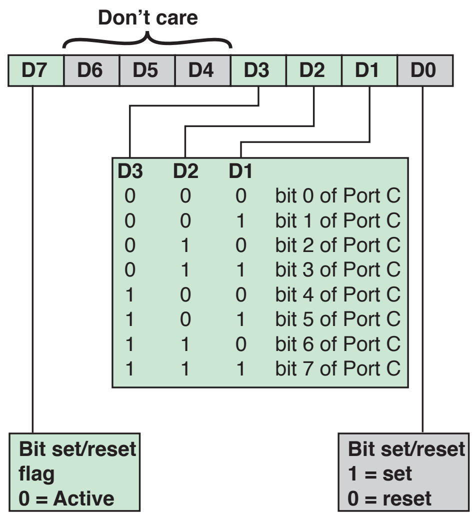


(b) Pin layout

The Intel 8255A Control Word



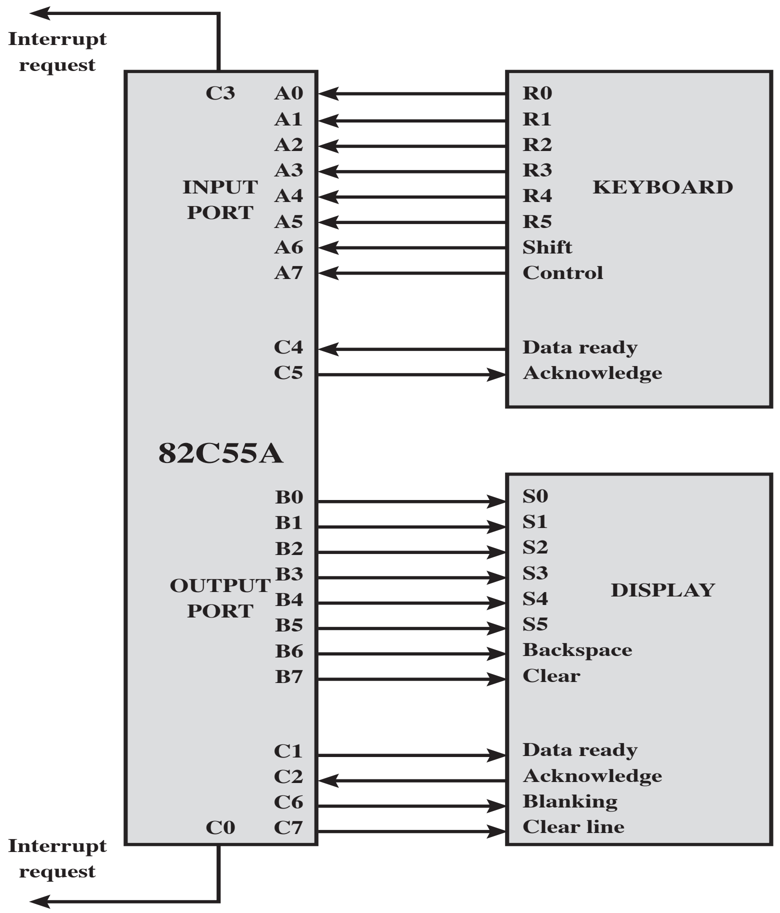
(a) Mode definition of the 8255 control register to configure the 8255



D3	D2	D1	
0	0	0	bit 0 of Port C
0	0	1	bit 1 of Port C
0	1	0	bit 2 of Port C
0	1	1	bit 3 of Port C
1	0	0	bit 4 of Port C
1	0	1	bit 5 of Port C
1	1	0	bit 6 of Port C
1	1	1	bit 7 of Port C

(b) Bit definitions of the 8255 control register to modify single bits of port C

Keyboard/Display Interface to 82C55A

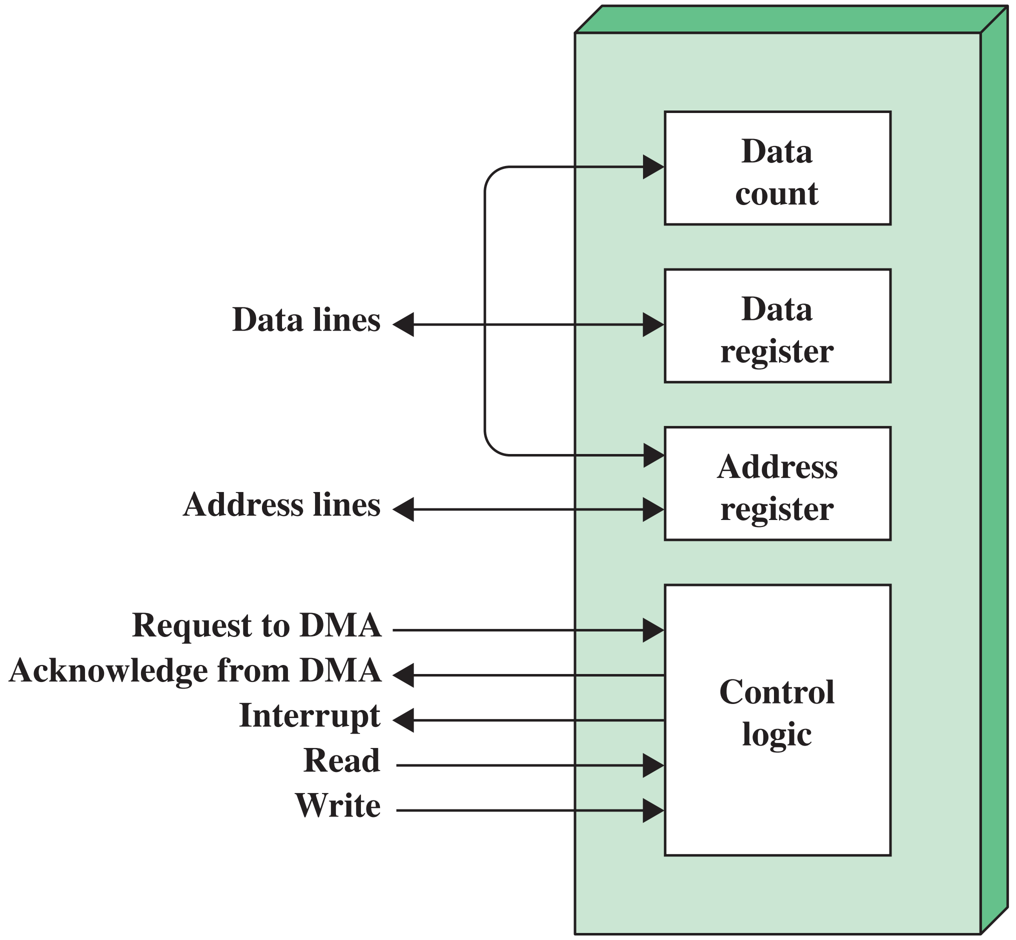


Drawbacks of Programmed and Interrupt-Driven I/O

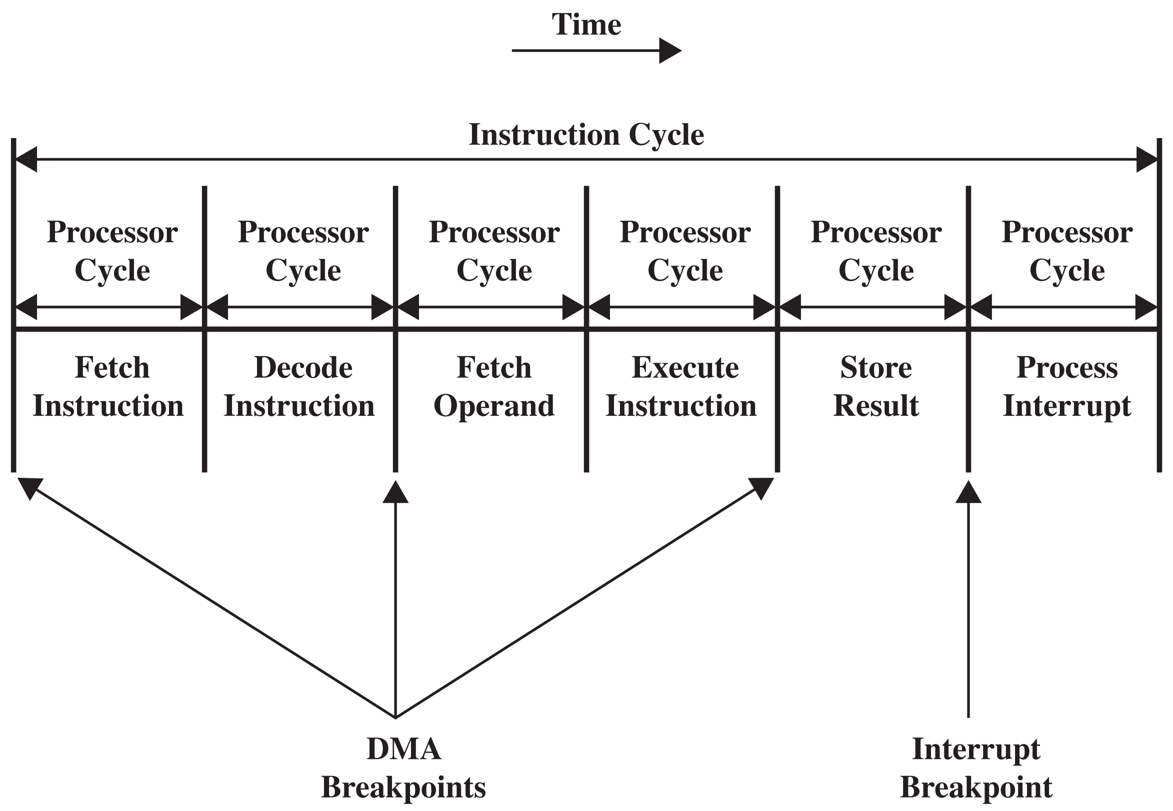
- Both forms of I/O suffer from two inherent drawbacks:
    1) The I/O transfer rate is limited by the speed with which the processor can test and service a device
    2) The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer

- When large volumes of data are to be moved a more efficient technique is direct memory access (DMA)

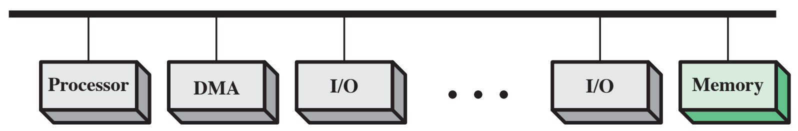
Typical DMA Block Diagram



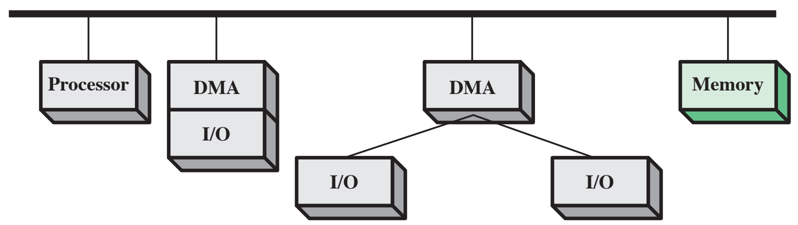
DMA and Interrupt Breakpoints During an Instruction Cycle



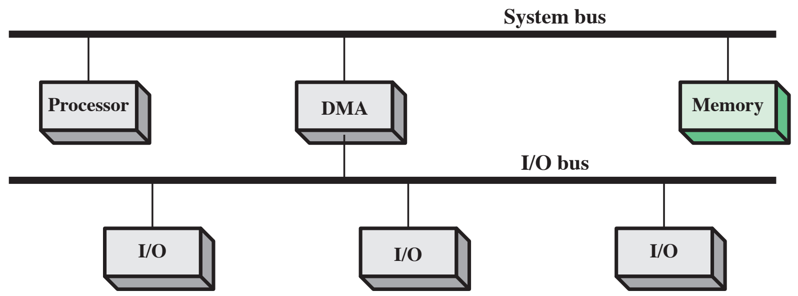
Alternative DMA Configuration



(a) Single-bus, detached DMA

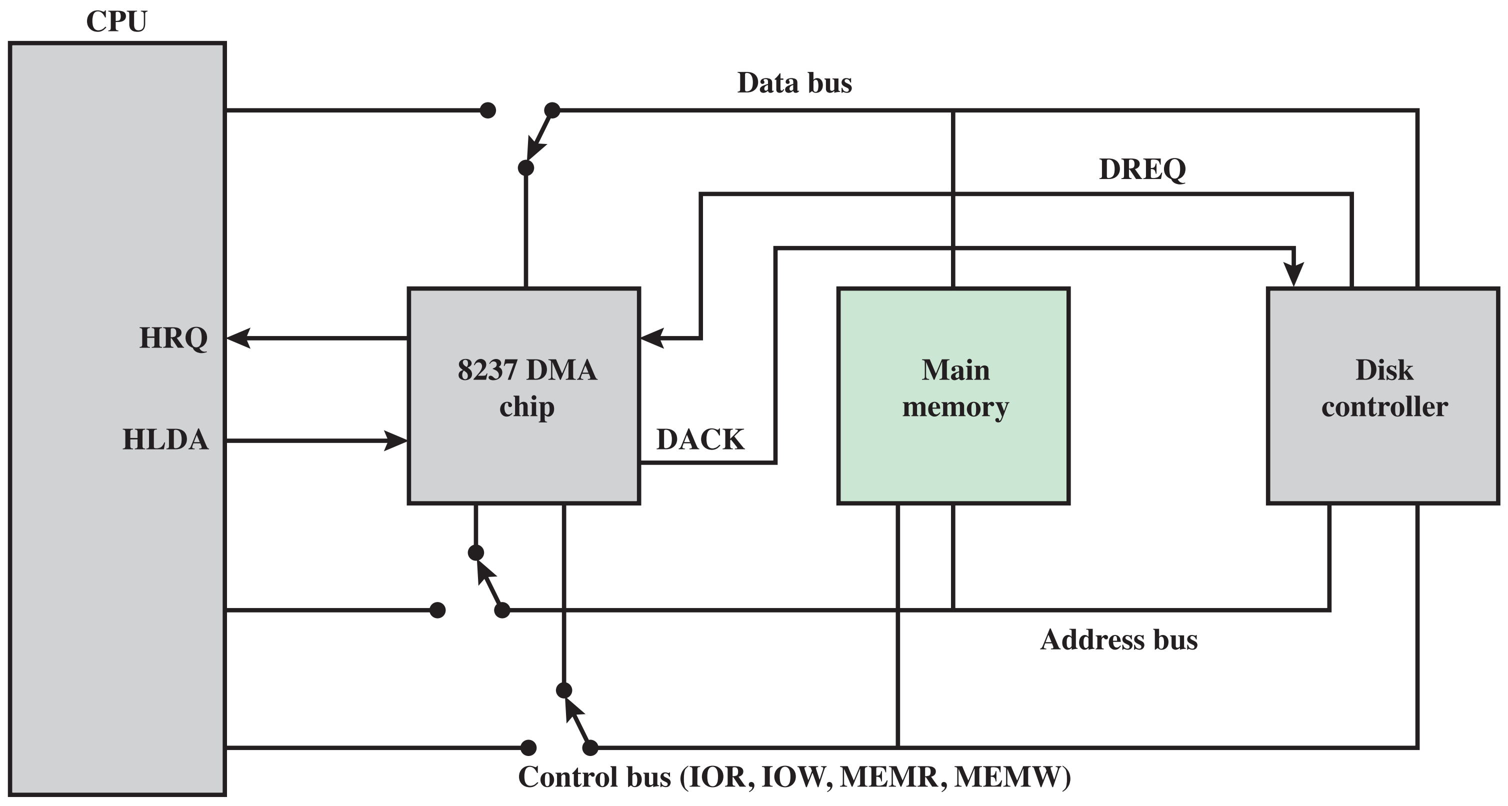


(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

8237 DMA Usage of System Bus



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

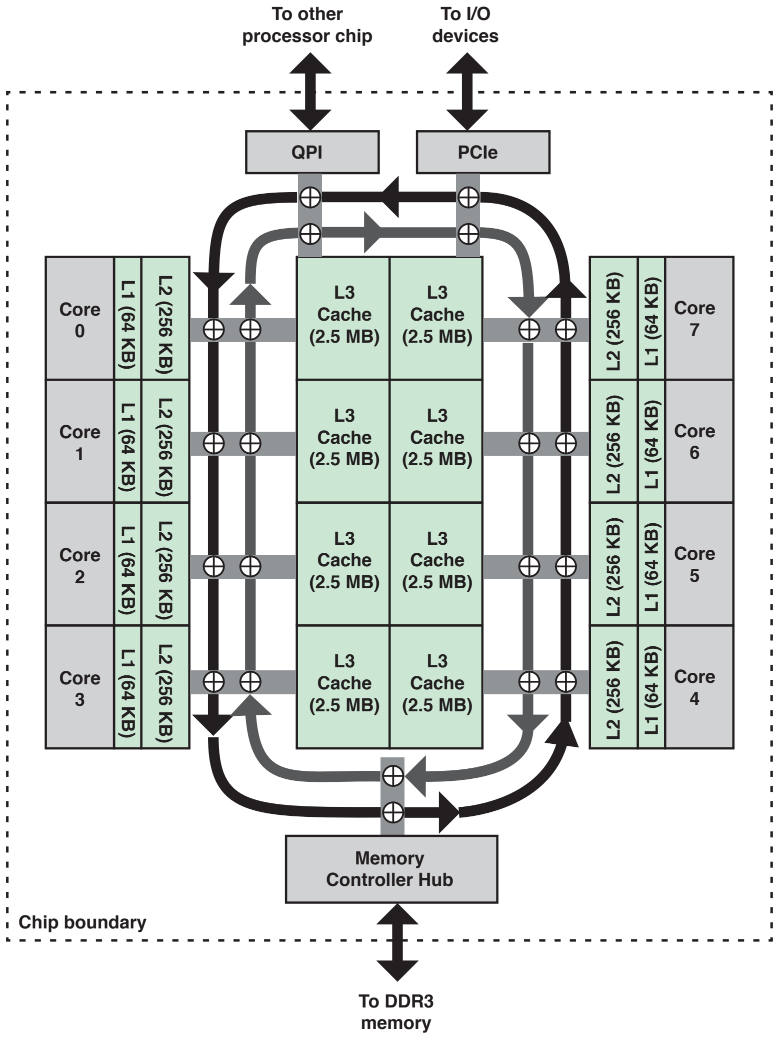
Fly-By DMA Controller

- 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
- 8237 contains four DMA channels
  - Programmed independently
  - Any one active
  - Numbered 0, 1, 2, and 3

Direct Cache Access (DCA)

- DMA is not able to scale to meet the increased demand due to dramatic increases in data rates for network I/O
- Demand is coming primarily from the widespread deployment of 10-Gbps and 100-Gbps Ethernet switches to handle massive amounts of data transfer to and from database servers and other high-performance systems
- Another source of traffic comes from Wi-Fi in the gigabit range
- Network Wi-Fi devices that handle 3.2 Gbps and 6.76 Gbps are becoming widely available and producing demand on enterprise systems

Xeon E-5-2600/4600 Chip Architecture



Cache-Related Performance Issues (1 of 3)

- Network traffic is transmitted in the form of a sequence of protocol blocks called packets or protocol data units
- The lowest, or link, level protocol is typically Ethernet, so that each arriving and departing block of data consists of an Ethernet packet containing as payload the higher-level protocol packet
- The higher-level protocols are usually the Internet Protocol (IP), operating on top of Ethernet and the Transmission Control Protocol (TCP), operating on top of IP

Cache-Related Performance Issues (2 of 3)

- The Ethernet payload consists of a block of data with a TCP header and an IP header
- For outgoing data, Ethernet packets are formed in a peripheral component, such as in I/O controller or network interface controller (NIC)
- For incoming traffic, the I/O controller strips off the Ethernet information and delivers the TCP/IP packet to the host CPU

Cache-Related Performance Issues (3 of 3)

- For both outgoing and incoming traffic the core, main memory, and cache are all involved
- In a DMA scheme, when an application wishes to transmit data, it places that data in an application-assigned buffer in main memory
  - The core transfers this to a system buffer in main memory and creates the necessary TCP and IP headers, which are also buffered in system memory
  - The packet is then picked up via DMA for transfer via the NIC
  - This activity engages not only main memory but also the cache
  - Similar transfers between system and application buffers are required for incoming traffic

Packet Traffic Steps (1 of 2)

- Incoming
  - Packet arrives
  - DMA
  - NIC interrupts host
  - Retrieve descriptors and headers
  - Cache miss occurs
  - Header is processed
  - Payload transferred

Packet Traffic Steps (2 of 2)

- Outgoing
  - Packet transfer requested
  - Packet created
  - Output operation invoked
  - DMA transfer
  - NIC signals completion
  - Driver frees buffer

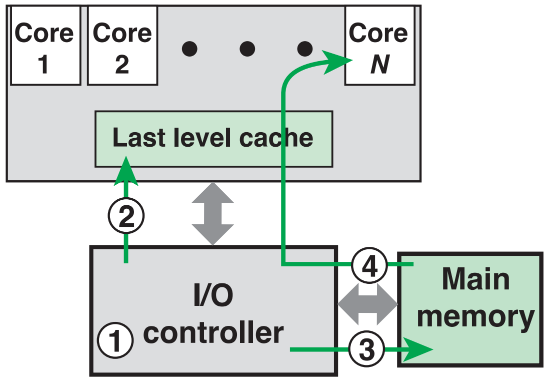
Direct Cache Access Strategies (1 of 2)

- Simplest strategy was implemented as a prototype on a number of Intel Xeon processors between 2006 and 2010
  - This form of DCA applies only to incoming network traffic
  - The DCA function in the memory controller sends a prefetch hint to the core as soon as the data is available in system memory
  - This enables the core to prefetch the data packet from the system buffer

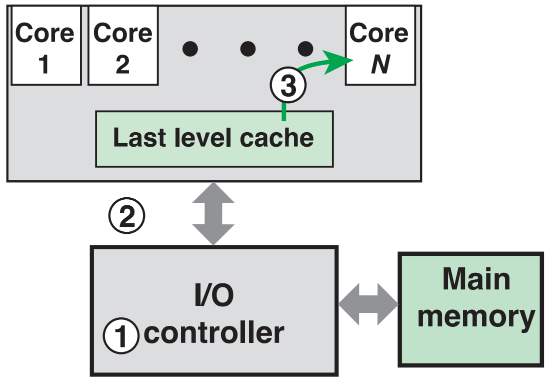
Direct Cache Access Strategies (2 of 2)

- Much more substantial gains can be realized by avoiding the system buffer in main memory altogether
  - The packet and packet descriptor information are accessed only once in the system buffer by the core
  - For incoming packets, the core reads the data from the buffer and transfers the packet payload to an application buffer
  - It has no need to access that data in the system buffer again
  - Cache injection
  - Implemented in Intel’s Xeon processor line, referred to as Direct Data I/O

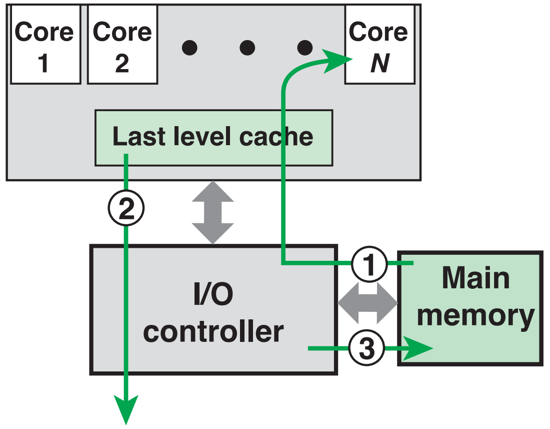
Comparison of DMA and DDIO



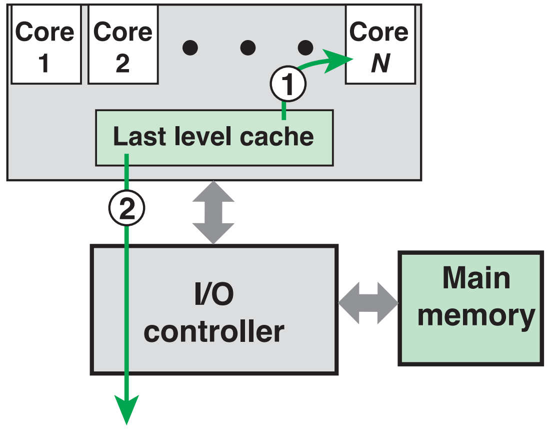
(a) Normal DMA transfer to memory



(b) DDIO transfer to cache



(c) Normal DMA transfer to I/O



(d) DDIO transfer to I/O

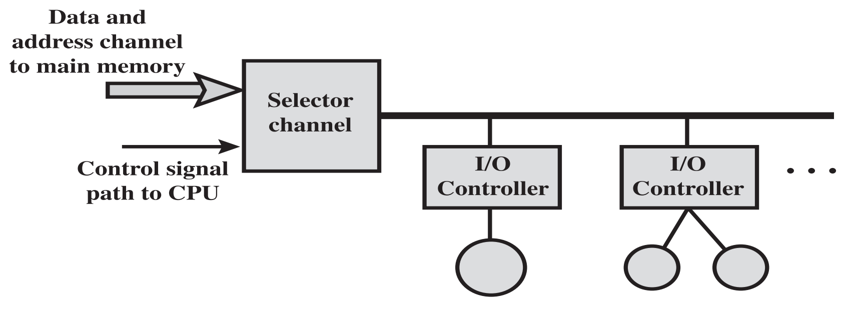
Evolution of the I/O Function (1 of 2)

1. The CPU directly controls a peripheral device.
2. A controller or I/O module is added. The CPU uses programmed I/O without interrupts.
3. Same configuration as in step 2 is used, but now interrupts are employed. The CPU need not spend time waiting for an I/O operation to be performed, thus increasing efficiency.

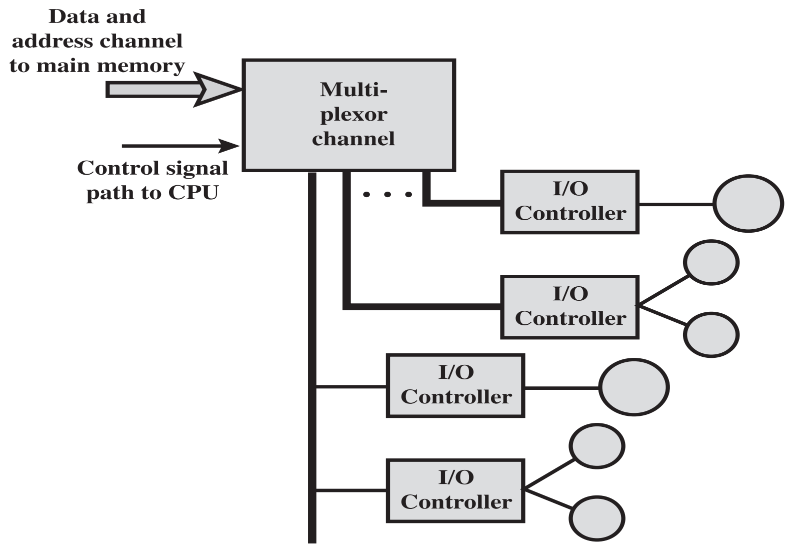
Evolution of the I/O Function (2 of 2)

4. The I/O module is given direct access to memory via DMA. It can now move a block of data to or from memory without involving the CPU, except at the beginning and end of the transfer.
5. The I/O module is enhanced to become a processor in its own right, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right. With this architecture a large set of I/O devices can be controlled with minimal CPU involvement.

I/O Channel Architecture



(a) Selector



(b) Multiplexor

Universal Serial Bus (USB) (1 of 2)

- Widely used for peripheral connections
- Is the default interface for slower speed devices
- Commonly used high-speed I/O, including printers, disk drives, and network adapters
- Has gone through multiple generations
  - USB 1.0
    - Defined a **Low Speed** data rate of 1.5 Mbps and a **Full Speed rate** of 12 Mbps
  - USB 2.0
    - Provides a data rate of 480 Mbps

Universal Serial Bus (USB) (2 of 2)

- USB 3.0
    - Higher speed bus called **SuperSpeed** in parallel with the USB 2.0 bus
    - Signaling speed of **SuperSpeed** is 5 Gbps, but due to signaling overhead the usable data rate is up to 4 Gbps
- USB 3.1
    - Includes a faster transfer mode called **SuperSpeed+**
    - This transfer mode achieves a signaling rate of 10 Gbps and a theoretical usable data rate of 9.7 Gbps
- Is controlled by a root host controller which attaches to devices to create a local network with a hierarchical tree topology

FireWire Serial Bus (1 of 2)

- Was developed as an alternative to small computer system interface (SCSI) to be used on smaller systems, such as personal computers, workstations, and servers
- Objective was to meet the increasing demands for high I/O rates while avoiding the bulky and expensive I/O channel technologies developed for mainframe and supercomputer systems
- IEEE standard 1394, for a High Performance Serial Bus
- Uses a daisy chain configuration, with up to 63 devices connected off a single port

FireWire Serial Bus (2 of 2)

- 1022 FireWire buses can be interconnected using bridges
- Provides for hot plugging which makes it possible to connect and disconnect peripherals without having to power the computer system down or reconfigure the system
- Provides for automatic configuration
- No terminations and the system automatically performs a configuration function to assign addresses

SCSI

- Small Computer System Interface
- A once common standard for connecting peripheral devices to small and medium-sized computers
- Has lost popularity to U S B and FireWire in smaller systems
- High-speed versions remain popular for mass memory support on enterprise systems
- Physical organization is a shared bus, which can support up to 16 or 32 devices, depending on the generation of the standard
  - The bus provides for parallel transmission rather than serial, with a bus width of 16 bits on earlier generations and 32 bits on later generations
  - Speeds range from 5 Mbps on the original SCSI-1 specification to 160 Mbps on S C S I-3 U3

Thunderbolt

- Most recent and fastest peripheral connection technology to become available for general-purpose use
- Developed by Intel with collaboration from Apple
- The technology combines data, video, audio, and power into a single high-speed connection for peripherals such as hard drives, RAID arrays, video-capture boxes, and network interfaces
- Provides up to 10 Gbps throughput in each direction and up to 10 Watts of power to connected peripherals

InfiniBand

- I/O specification aimed at the high-end server market
- First version was released in early 2001
- Heavily relied on by I B M zEnterprise series of mainframes
- Standard describes an architecture and specifications for data flow among processors and intelligent I/O devices
- Has become a popular interface for storage area networking and other large storage configurations
- Enables servers, remote storage, and other network devices to be attached in a central fabric of switches and links
- The switch-based architecture can connect up to 64,000 servers, storage systems, and networking devices

PCI Express and SATA

PCI Express

- High-speed bus system for connecting peripherals of a wide variety of types and speeds

SATA

- Serial Advanced Technology Attachment
- An interface for disk storage systems
- Provides data rates of up to 6 Gbps, with a maximum per device of 300 Mbps
- Widely used in desktop computers and in industrial and embedded applications

Ethernet

- Predominant wired networking technology
- Has evolved to support data rates up to 100 Gbps and distances from a few meters to tens of km
- Has become essential for supporting personal computers, workstations, servers, and massive data storage devices in organizations large and small
- Began as an experimental bus-based 3-Mbps system

Ethernet (2 of 2)

- Has moved from bus-based to switch-based
  - Data rate has periodically increased by an order of magnitude
  - There is a central switch with all of the devices connected directly to the switch
- Ethernet systems are currently available at speeds up to 100 Gbps

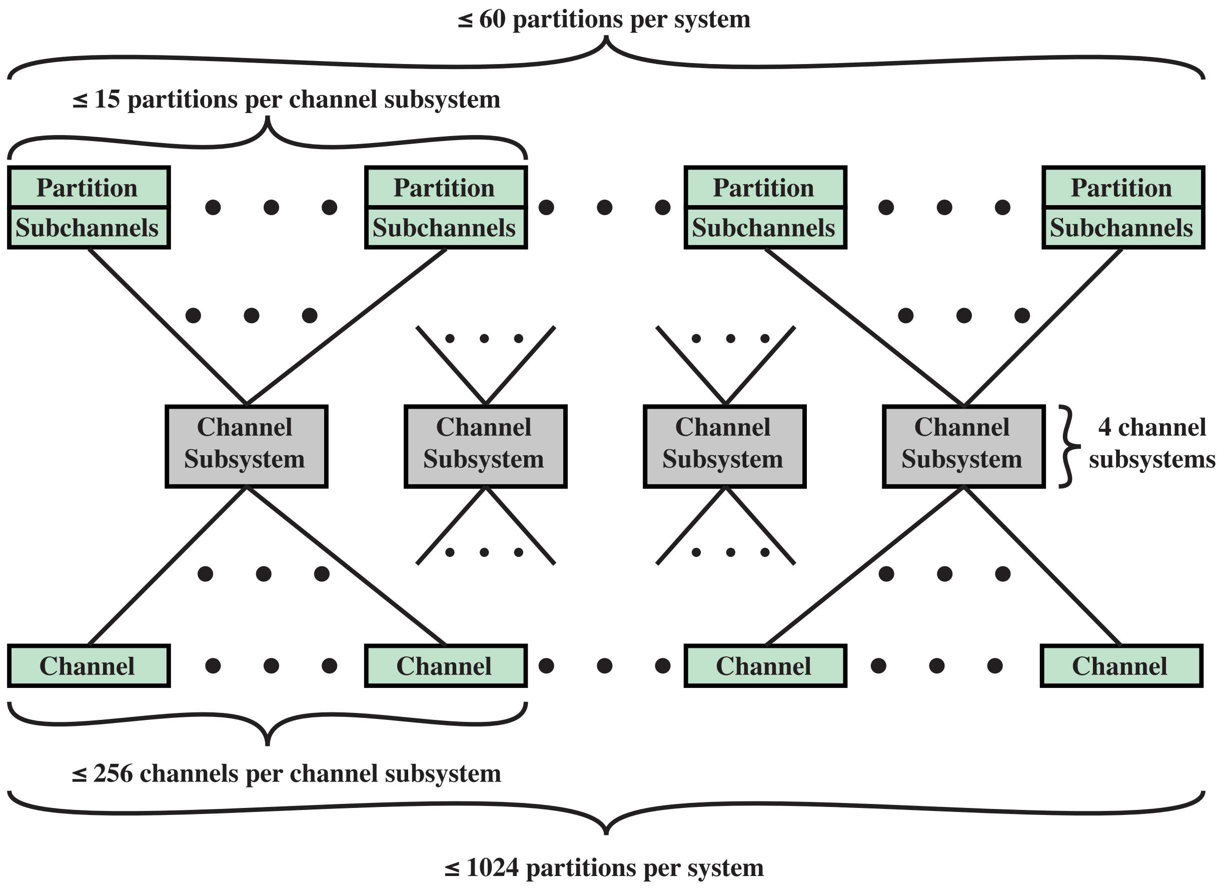
Wi-Fi (1 of 2)

- Is the predominant wireless Internet access technology
- Now connects computers, tablets, smart phones, and other electronic devices such as video cameras TVs and thermostats
- In the enterprise has become an essential means of enhancing worker productivity and network effectiveness
- Public hotspots have expanded dramatically to provide free Internet access in most public places

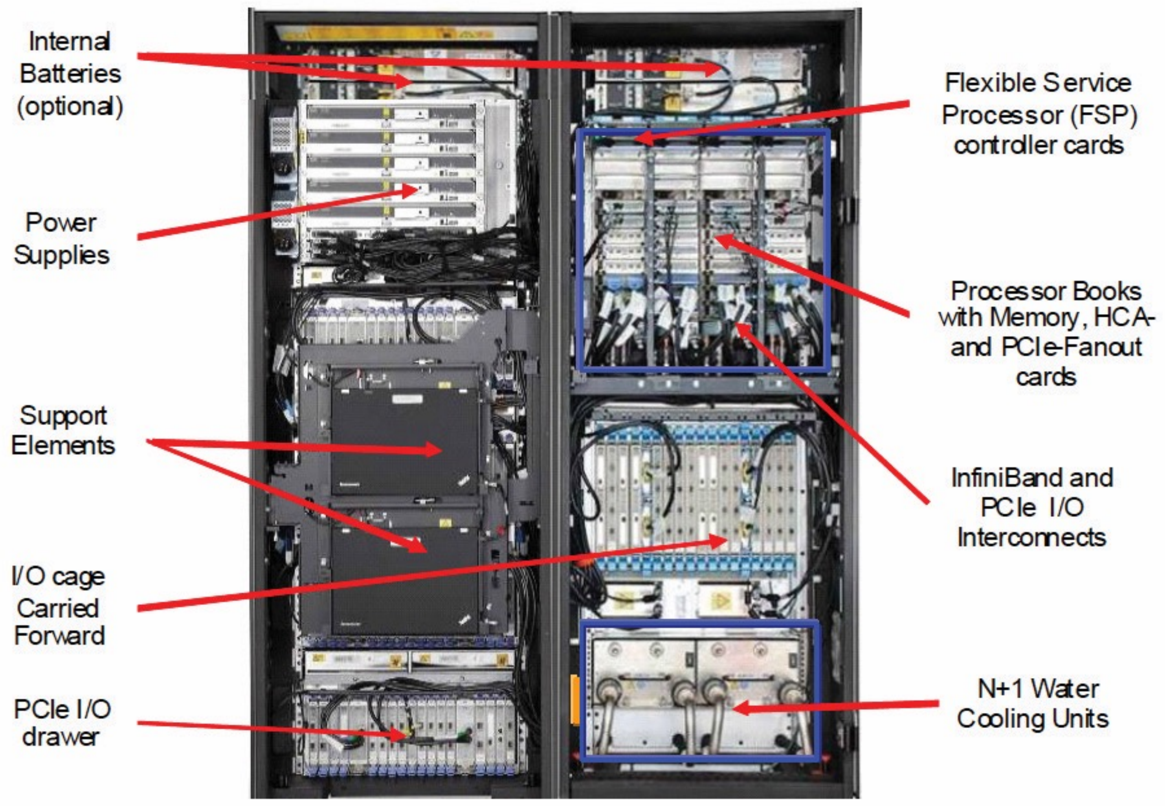
Wi-Fi (2 of 2)

- As the technology of antennas, wireless transmission techniques, and wireless protocol design has evolved, the IEEE 802.11 committee has been able to introduce standards for new versions of Wi-Fi at higher speeds
- Current version is 802.11ac (2014) with a maximum data rate of 3.2 Gbps

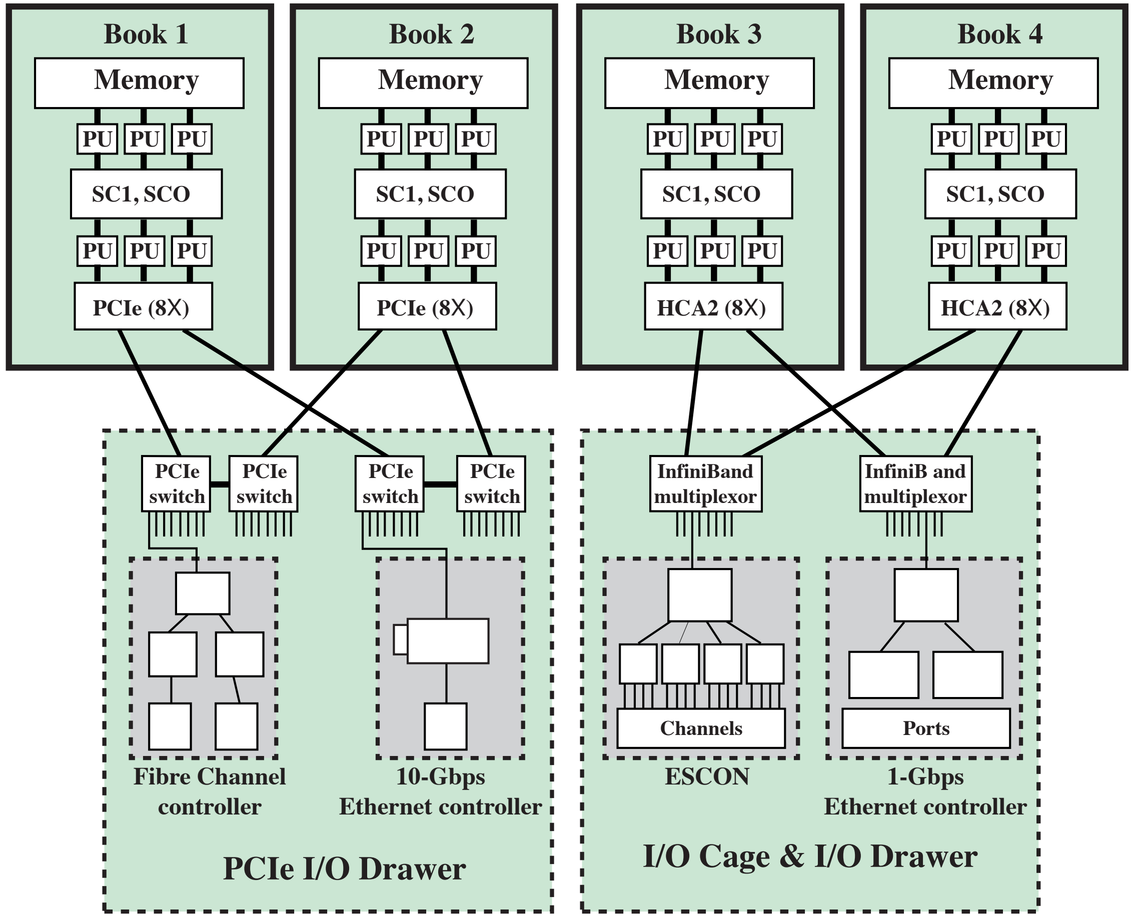
IBM EC121 I/O Channel Subsystem Structure



IBM zEC12 I/O Frames-Front View


Internal Batteries (optional)
Power Supplies
Support Elements
I/O cage Carried Forward
PCIe I/O drawer
Flexible Service Processor (FSP) controller cards
Processor Books with Memory, HCA- and PCIe-Fanout cards
InfiniBand and PCIe I/O Interconnects
N+1 Water Cooling Units

IBM EC121 I/O System Structure


Book 1
Memory
PU PU PU
SC1, SCO
PU PU PU
PCIe (8X)
Book 2
Memory
PU PU PU
SC1, SCO
PU PU PU
PCIe (8X)
Book 3
Memory
PU PU PU
SC1, SCO
PU PU PU
HCA2 (8X)
Book 4
Memory
PU PU PU
SC1, SCO
PU PU PU
HCA2 (8X)
PCIe switch
PCIe switch
PCIe switch
PCIe switch
InfiniBand multiplexor
InfiniB and multiplexor
Channels
Ports
Fibre Channel controller
10-Gbps Ethernet controller
ESCON
1-Gbps Ethernet controller
PCIe I/O Drawer
I/O Cage & I/O Drawer