The computer system’s I/O architecture is its interface to the outside world. This architecture is designed to provide a systematic means of controlling interaction with the outside world and to provide the operating system with the information it needs to manage I/O activity effectively.

There are three principal I/O techniques: **programmed I/O**, in which I/O occurs under the direct and continuous control of the program requesting the I/O operation; **interrupt-driven I/O**, in which a program issues an I/O command and then continues to execute, until it is interrupted by the I/O hardware to signal the end of the I/O operations; and **direct memory access (DMA)**, in which a specialized I/O processor takes over control of an I/O operation to move a large block of data.

Two important examples of external I/O interfaces are FireWire and Infiniband.

**Peripherals and the System Bus**
- There are a wide variety of peripherals each with varying methods of operation
  - Impractical to have the processor accommodate all
- Data transfer rates are often slower than the processor and/or memory
  - Impractical to use the high-speed system bus to communicate directly
- Data transfer rates may be faster than that of the processor and/or memory
  - This mismatch may lead to inefficiencies if improperly managed
- Peripheral often use different data formats and word lengths

**Purpose of I/O Modules**
- Interface to the processor and memory via the system bus or control switch
- Interface to one or more peripheral devices
External Devices

External device categories

- **Human readable**: communicate with the computer user – CRT
- **Machine readable**: communicate with equipment – disk drive or tape drive
- **Communication**: communicate with remote devices – may be human readable or machine readable

The External Device – I/O Module

- **Control signals**: determine the function that will be performed
- **Data**: set of bits to be sent or received
- **Status signals**: indicate the state of the device
- **Control logic**: controls the device’s operations
- **Transducer**: converts data from electrical to other forms of energy
- **Buffer**: temporarily holds data being transferred

Keyboard/Monitor

- Most common means of computer/user interaction
- Keyboard provides input that is transmitted to the computer
- Monitor displays data provided by the computer
- The character is the basic unit of exchange
- Each character is associated with a 7 or 8 bit code
Disk Drive

- Contains electronics for exchanging data, control, and status signals with an I/O module
- Contains electronics for controlling the disk read/write mechanism
- Fixed-head disk – transducer converts between magnetic patterns on the disk surface and bits in the buffer
- Moving-head disk – must move the disk arm rapidly across the surface

I/O Modules

Module Function

- Control and timing
- Processor communication
- Device communication
- Data buffering
- Error detection

I/O control steps

- Processor checks I/O module for external device status
- I/O module returns status
- If device ready, processor gives I/O module command to request data transfer
- I/O module gets a unit of data from device
- Data transferred from the I/O module to the processor

Processor communication

- **Command decoding:** I/O module accepts commands from the processor sent as signals on the control bus
- **Data:** data exchanged between the processor and I/O module over the data bus
- **Status reporting:** common status signals BUSY and READY are used because peripherals are slow
- **Address recognition:** I/O module must recognize a unique address for each peripheral that it controls
I/O module communication

- **Device communication**: commands, status information, and data
- **Data buffering**: data comes from main memory in rapid burst and must be buffered by the I/O module and then sent to the device at the device’s rate
- **Error detection**: responsible for reporting errors to the processor

Typical I/O Device Data Rates

![Typical I/O Device Data Rates graph]

I/O Module Structure

Block Diagram of an I/O Module

![Block Diagram of an I/O Module]
• Module connects to the computer through a set of signal lines – system bus
• Data transferred to and from the module are buffered with data registers
• Status provided through status registers – may also act as control registers
• Module logic interacts with processor via a set of control signal lines
• Processor uses control signal lines to issue commands to the I/O module
• Module must recognize and generate addresses for devices it controls
• Module contains logic for device interfaces to the devices it controls

• I/O module functions allow the processor to view devices is a simple-minded way
• I/O module may hide device details from the processor so the processor only functions in terms of simple read and write operations – timing, formats, etc…
• I/O module may leave much of the work of controlling a device visible to the processor – rewind a tape, etc…

I/O channel or I/O processor
• I/O module that takes on most of the detailed processing burden
• Used on mainframe computers

I/O controller of device controller
• Primitive I/O module that requires detailed control
• Used on microcomputers

Programmed I/O

Overview of Programmed I/O

• Processor executes an I/O instruction by issuing command to appropriate I/O module
• I/O module performs the requested action and then sets the appropriate bits in the I/O status register – I/O module takes not further action to alert the processor – it does not interrupt the processor
• The processor periodically checks the status of the I/O module until it determines that the operation is complete
I/O Commands

The processor issues an address, specifying I/O module and device, and an I/O command. The commands are:

- **Control**: activate a peripheral and tell it what to do
- **Test**: test various status conditions associated with an I/O module and its peripherals
- **Read**: causes the I/O module to obtain an item of data from the peripheral and place it into an internal register
- **Write**: causes the I/O module to take a unit of data from the data bus and transmit it to the peripheral

Three Techniques for Input of a Block of Data

I/O Instructions

Processor views I/O operations in a similar manner as memory operations
Each device is given a unique identifier or address
Processor issues commands containing device address – I/O module must check address lines to see if the command is for itself
I/O mapping

- Memory-mapped I/O
  - Single address space for both memory and I/O devices
    - disadvantage – uses up valuable memory address space
  - I/O module registers treated as memory addresses
  - Same machine instructions used to access both memory and I/O devices
    - advantage – allows for more efficient programming
  - Single read line and single write lines needed
  - Commonly used

- Isolated I/O
  - Separate address space for both memory and I/O devices
  - Separate memory and I/O select lines needed
  - Small number of I/O instructions
  - Commonly used

Interrupt-Driven I/O

- Overcomes the processor having to wait long periods of time for I/O modules
- The processor does not have to repeatedly check the I/O module status

I/O module view point
- I/O module receives a READ command form the processor
- I/O module reads data from desired peripheral into data register
- I/O module interrupts the processor
- I/O module waits until data is requested by the processor
- I/O module places data on the data bus when requested

Processor view point
- The processor issues a READ command
- The processor performs some other useful work
- The processor checks for interrupts at the end of the instruction cycle
- The processor saves the current context when interrupted by the I/O module
- The processor read the data from the I/O module and stores it in memory
- The processor the restores the saved context and resumes execution
Design Issues

How does the processor determine which device issued the interrupt
How are multiple interrupts dealt with

Device identification
- Multiple interrupt lines – each line may have multiple I/O modules
- Software poll – poll each I/O module
  - Separate command line – TESTI/O
  - Processor read status register of I/O module
  - Time consuming
- Daisy chain
  - Hardware poll
  - Common interrupt request line
  - Processor sends interrupt acknowledge
  - Requesting I/O module places a word of data on the data lines – “vector” that uniquely identifies the I/O module – vectored interrupt
- Bus arbitration
  - I/O module first gains control of the bus
  - I/O module sends interrupt request
  - The processor acknowledges the interrupt request
  - I/O module places its vector of the data lines

Multiple interrupts
- The techniques above not only identify the requesting I/O module but provide methods of assigning priorities
- Multiple lines – processor picks line with highest priority
- Software polling – polling order determines priority
- Daisy chain – daisy chain order of the modules determines priority
- Bus arbitration – arbitration scheme determines priority

Intel 82C59A Interrupt Controller

Intel 80386 provides
- Single Interrupt Request line – INTR
- Single Interrupt Acknowledge line – INTA
- Connects to an external interrupt arbiter, 82C59A, to handle multiple devices and priority structures
- 8 external devices can be connected to the 82C59A – can be cascaded to 64
82C59A operation – only manages interrupts
- Accepts interrupt requests
- Determines interrupt priority
- Signals the processor using INTR
- Processor acknowledges using INTA
- Places vector information of data bus
- Processor processes interrupt and communicates directly with I/O module

82C59A interrupt modes
Fully nested – priority form 0 (IR0) to 7 (IR7)
Rotating – several devices same priority - most recently device lowest priority
Special mask – processor can inhibit interrupts from selected devices

Intel 82C55A Programmable Peripheral Interface
- Single chip, general purpose I/O module
- Designed for use with the Intel 80386
- Can control a variety of simple peripheral devices
A, B, C function as 8 bit I/O ports (C can be divided into two 4 bit I/O ports). Left side of diagram show the interface to the 80386 bus.

Interface to a Keyboard/Display Terminal
Direct Memory Access

Drawback of Programmed and Interrupt-Driven I/O
- I/O transfer rate limited to speed that processor can test and service devices
- Processor tied up managing I/O transfers

DMA Function

- DMA module on system bus used to mimic the processor.
- DMA module only uses system bus when processor does not need it.
- DMA module may temporarily force processor to suspend operations – cycle stealing.

DMA Operation

- The processor issues a command to DMA module
  - Read or write
  - I/O device address using data lines
  - Starting memory address using data lines – stored in address register
  - Number of words to be transferred using data lines – stored in data register
- The processor then continues with other work
- DMA module transfers the entire block of data – one word at a time – directly to or from memory without going through the processor
- DMA module sends an interrupt to the processor when complete
DMA and Interrupt Breakpoints during Instruction Cycle

- The processor is suspended just before it needs to use the bus.
- The DMA module transfers one word and returns control to the processor.
- Since this is not an interrupt the processor does not have to save context.
- The processor executes more slowly, but this is still far more efficient than either programmed or interrupt-driven I/O.

DMA Configurations

- Single bus – detached DMA module
- Each transfer uses bus twice – I/O to DMA, DMA to memory
- Processor suspended twice

- Single bus – integrated DMA module
- Module may support more than one device
- Each transfer uses bus once – DMA to memory
- Processor suspended once
- Separate I/O bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once – DMA to memory
- Processor suspended once

**I/O Channels and Processors**

The Evolution of the I/O Function

1. Processor directly controls peripheral device
2. Addition of a controller or I/O module – programmed I/O
3. Same as 2 – interrupts added
4. I/O module direct access to memory using DMA
5. I/O module enhanced to become processor like – I/O channel
6. I/O module has local memory of its own – computer like – I/O processor

- More and more the I/O function is performed without processor involvement.
- The processor is increasingly relieved of I/O related tasks – improved performance.

**Characteristics of I/O Channels**

- Extension of the DMA concept
- Ability to execute I/O instructions – special-purpose processor on I/O channel – complete control over I/O operations
- Processor does not execute I/O instructions itself – processor initiates I/O transfer by instructing the I/O channel to execute a program in memory
- Program specifies
  - Device or devices
  - Area or areas of memory
  - Priority
  - Error condition actions
Two type of I/O channels

- **Selector channel**
  - Controls multiple high-speed devices
  - Dedicated to the transfer of data with one of the devices
  - Each device handled by a controller, or I/O module
  - I/O channel controls these I/O controllers

- **Multiplexor channel**
  - Can handle multiple devices at the same time
  - Byte multiplexor – used for low-speed devices
  - Block multiplexor – interleaves blocks of data from several devices.

The External Interface: FireWire and Infiniband

Type of Interfaces

Parallel interface – multiple bits transferred simultaneously
Serial interface – bits transferred one at a time
I/O module dialog for a write operation
1. I/O module sends control signal – requesting permission to send data
2. Peripheral acknowledges the request
3. I/O module transfer data
4. Peripheral acknowledges receipt of data

FireWire Serial Bus – IEEE 1394

- Very high speed serial bus
- Low cost
- Easy to implement
- Used with digital cameras, VCRs, and televisions

FireWire Configurations

- Daisy chain
- 63 devices on a single port – 64 if you count the interface itself
- 1022 FireWire busses can be interconnected using bridges
- Hot plugging
- Automatic configuration
- No terminations
- Can be tree structured rather than strictly daisy chained

FireWire three layer stack:

Physical layer
- Defines the transmission media that are permissible and the electrical and signaling characteristics of each
- 25 to 400 Mbps
- Converts binary data to electrical signals
- Provides arbitration services
  - Based on tree structure
  - Root acts as arbiter
  - First come first served
  - Natural priority controls simultaneous requests – nearest root
  - Fair arbitration
  - Urgent arbitration

Link layer
- Describes the transmission of data in the packets
- Asynchronous
  - Variable amount of data and several bytes of transaction data transferred as a packet
  - Uses an explicit address
  - Acknowledgement returned
- Isochronous
  - Variable amount of data in sequence of fixed sized packets at regular intervals
  - Uses simplified addressing
  - No acknowledgement

Transaction layer
- Defines a request-response protocol that hides the lower-layer detail of FireWire from applications

FireWire Protocol Stack
FireWire Subactions

InfiniBand

- Recent I/O specification aimed at high-end server market
- First version released early 2001
- Standard for data flow between processors and intelligent I/O devices
- Intended to replace PCI bus in servers
- Greater capacity, increased expandability, enhanced flexibility
- Connect servers, remote storage, network devices to central fabric of switches and links
- Greater server density
- Independent nodes added as required
- I/O distance from server up to
  - 17 meters using copper
  - 300 meters using multimode optical fiber
  - 10 kilometers using single-mode optical fiber
- Transmission rates up to 30 Gbps
InfiniBand Switch Fabric

InfiniBand Operations

- 16 logical channels (virtual lanes) per physical link
- One lane for fabric management – all other lanes for data transport
- Data sent as a stream of packets
- Virtual lane temporarily dedicated to the transfer from one end node to another
- Switch maps traffic from incoming lane to outgoing lane

InfiniBand Communication Protocol Stack