Input / Output System

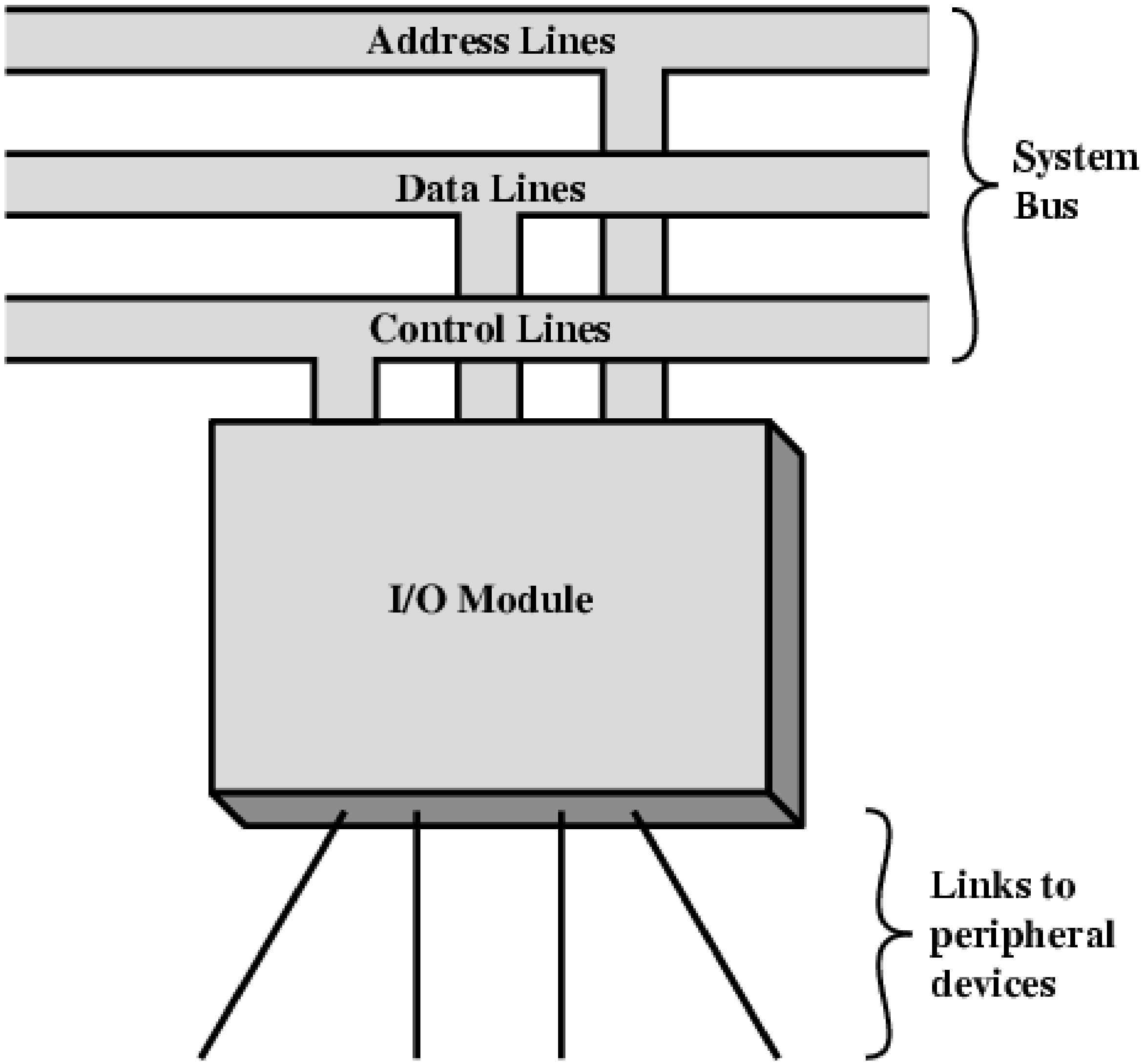
Input/Output Problems

- Wide variety of peripherals:
  - Delivering different amounts of data,
  - At different speeds,
  - In different formats.
- All are slower than CPU and RAM.
- A strong need for I/O modules.

Input/Output Module

- Interface to CPU and memory.
- Interface to one or more peripherals.
- I/O module – I/O adapter, I/O controller (equal concepts).
- The ability to work autonomously, i. e. without any attention from CPU is a very important aspect of performing peripheral operations.

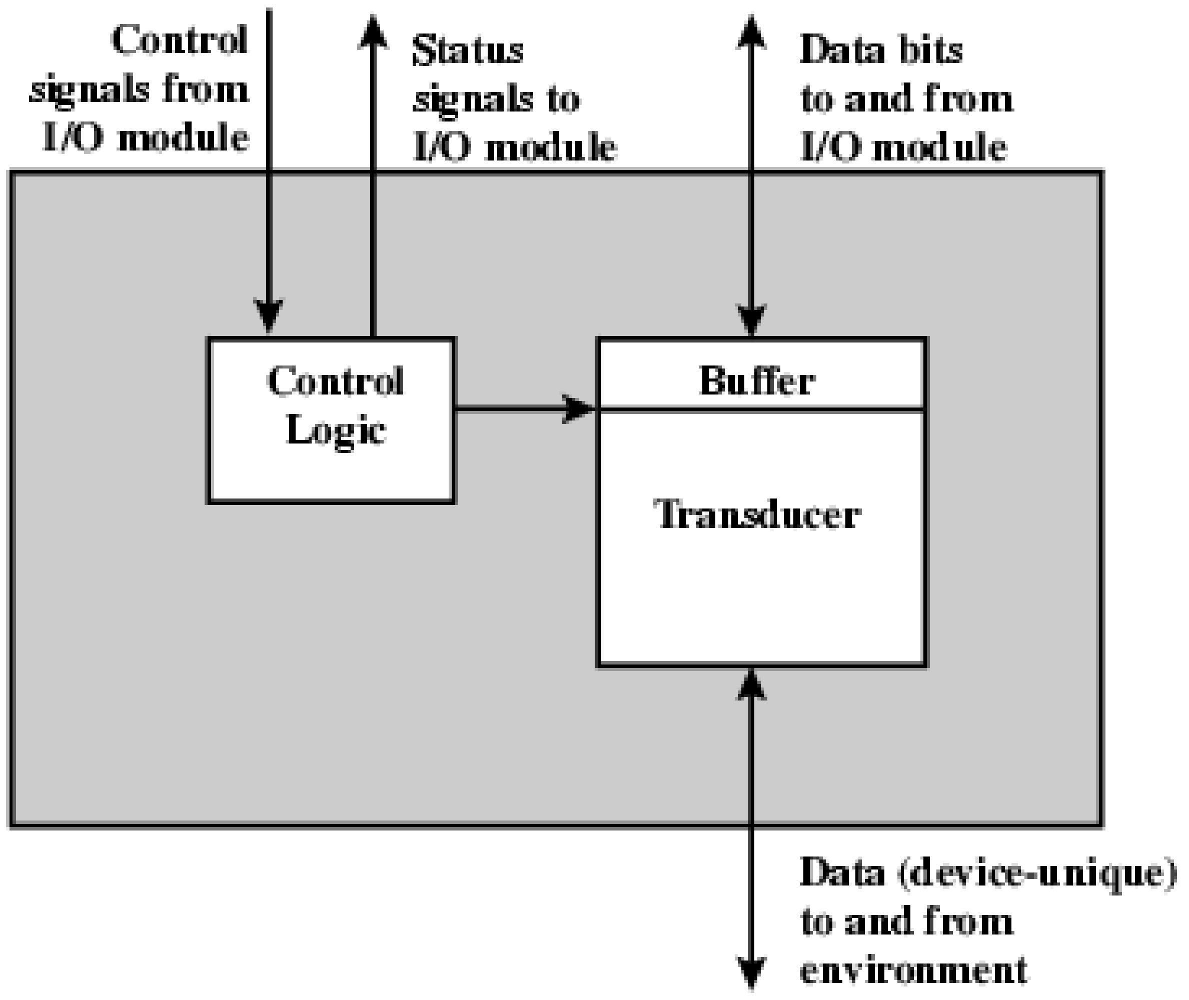
Generic Model of I/O Module



External Devices (Peripheral Devices)

- Human readable:
  - screen, printer, keyboard.
- Machine readable:
  - monitoring and control.
- Communication:
  - modem,
  - Network Interface Card (NIC).

External (Peripheral) Device Block Diagram


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

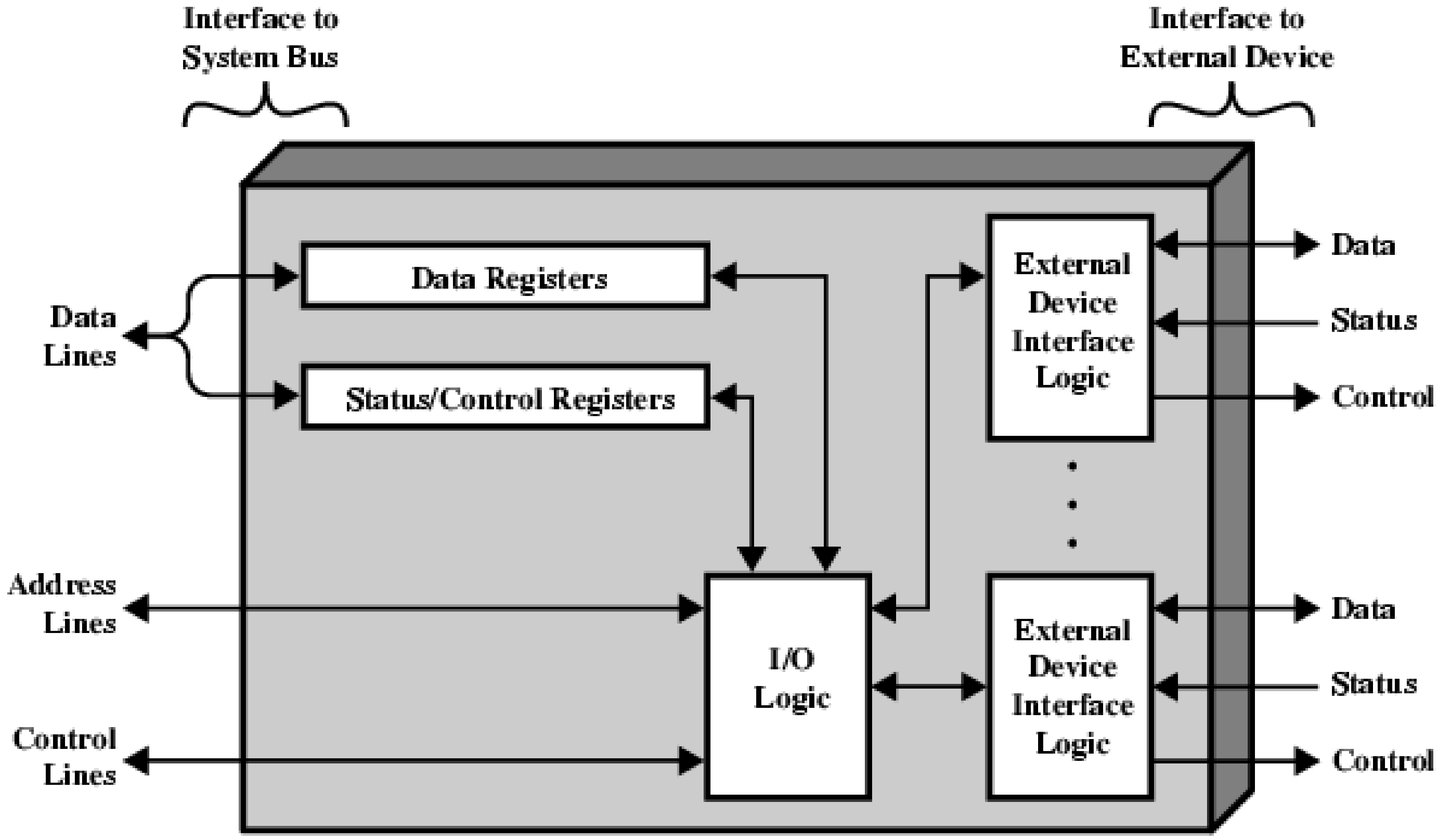
I/O Module Function

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

I/O Steps

- CPU checks I/O module and external device status (status byte, sense bytes). Sense bytes – more detailed information about errors in the peripheral device.
- I/O module returns status to CPU.
- If ready, CPU requests data transfer.
- I/O module gets data from device.
- CPU checks I/O module and external device status (status byte, sense bytes).
- I/O module transfers data to CPU.
- This is an example of programmed I/O.
- <u>Important - CPU checks repeatedly whether the peripheral operation was finished (polling)</u>.

I/O Module Diagram

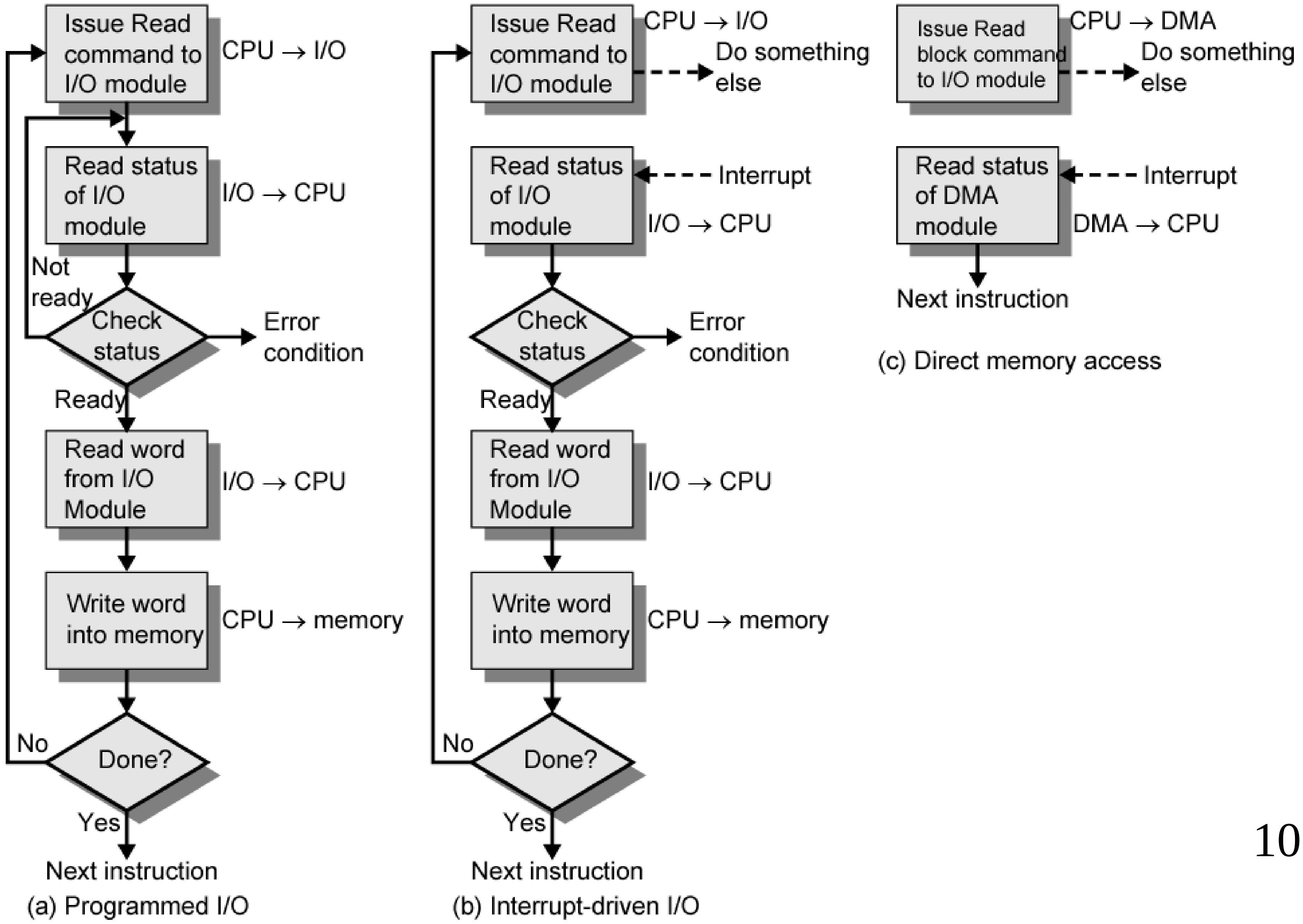


Input Output Techniques

- Programmed I/O (not used at present).
- Interrupt driven I/O.
- Direct Memory Access (DMA).

They differ in the way in which I/O module informs CPU that the peripheral operation has finished because the peripheral operation was performed autonomously (independently).

Three Techniques for Input of a Block of Data



Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Controlling read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation.
- It wastes CPU time.

Programmed I/O - detail

- 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 (polling).

I/O Commands

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

Addressing I/O Devices (registers)

- Under programmed I/O data transfer is very like memory access (CPU viewpoint).
- Each device (register) is given unique identifier.
- CPU commands contain identifier (address).

I/O Mapping

- Memory mapped I/O
  - Devices (registers) 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 (memory and registers).
  - Need I/O or memory select lines.
  - Special commands for I/O.
    - Limited set.

Interrupt Driven I/O

- Overcomes CPU waiting.
- I/O module interrupts CPU when ready.
- No repeated CPU checking of device.

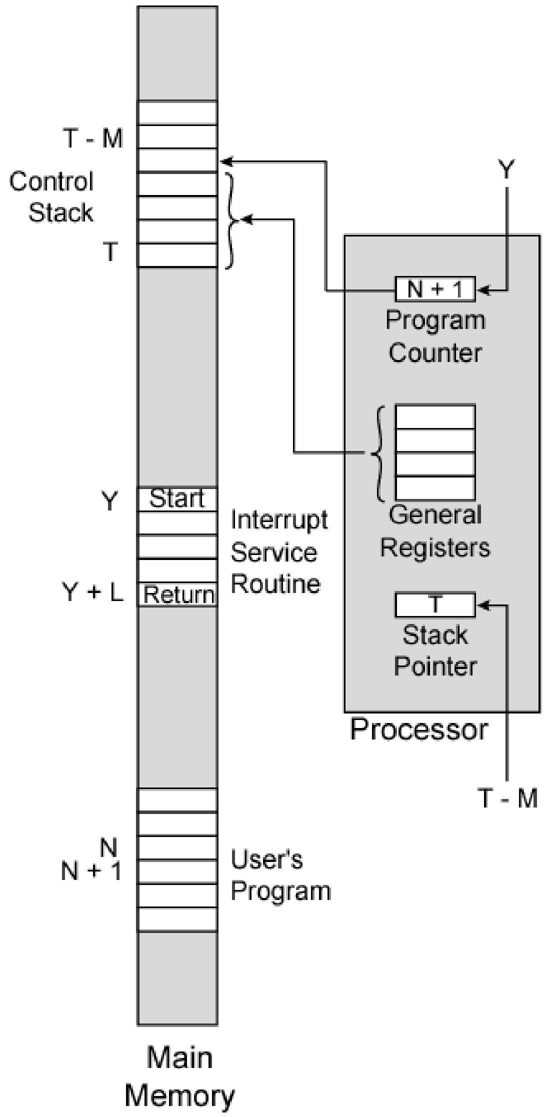
Interrupt Driven I/O
Basic Operation

- CPU issues read command.
- **I/O module gets data from peripheral whilst CPU does other work.**
- I/O module interrupts CPU.
- CPU requests data.
- I/O module transfers data.

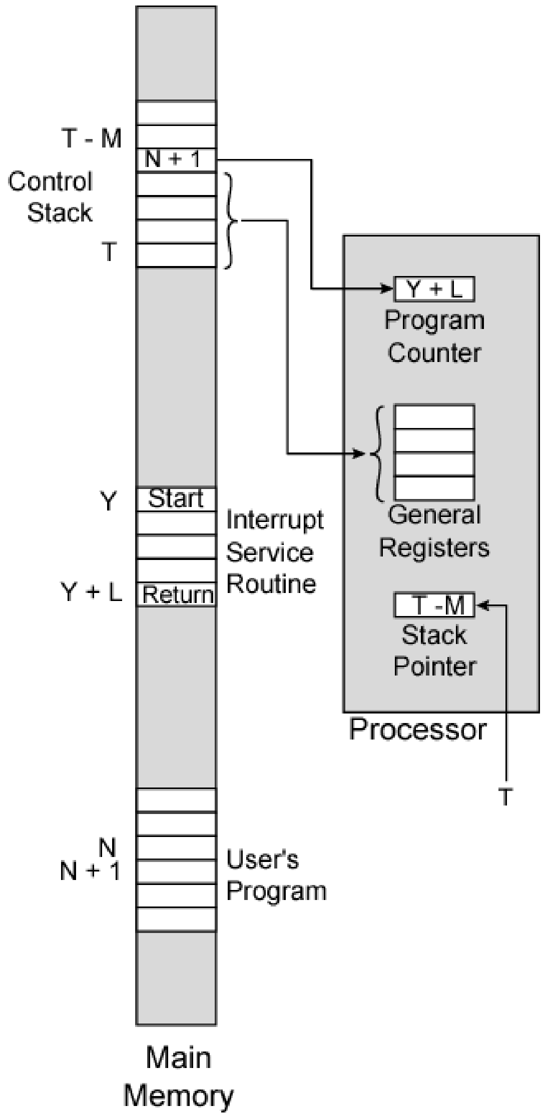
CPU Viewpoint

- CPU issues read command.
- CPU does other work.
- CPU checks for interrupt at the end of each instruction cycle.
- If interrupted:
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N



(b) Return from interrupt

Design Issues

- How do you identify the module issuing the interrupt?
- How do you deal with multiple interrupts?
  - —i.e. an interrupt handler being interrupted

Identifying Interrupting Module (1)

- Different line for each module (holds for ISA, not for PCI)
  - PC
  - The number of peripheral devices is limited (it does not hold for PCI bus).
- Software poll
  - CPU asks each module in turn
  - Slow

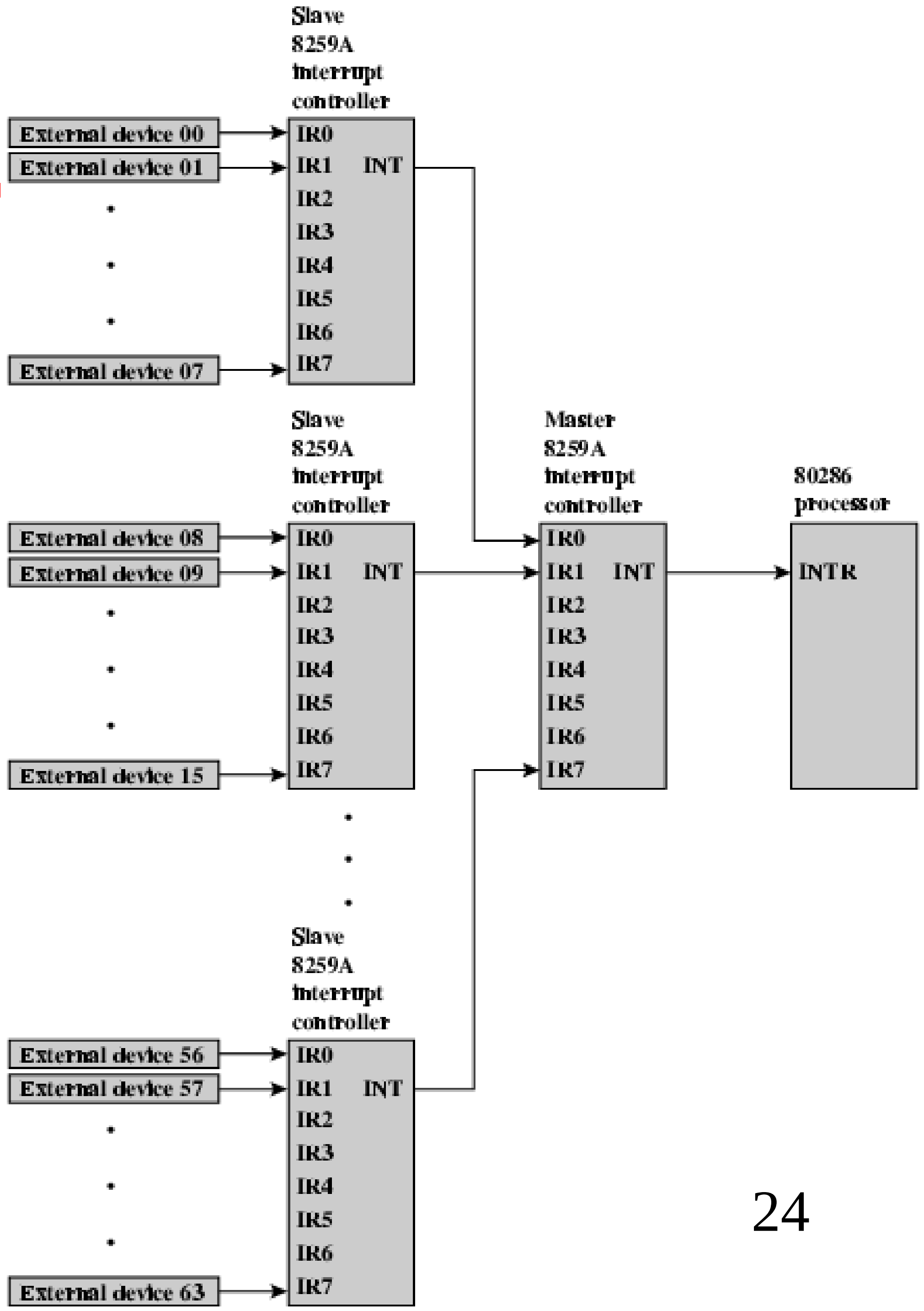
Multiple Interrupts

- Each interrupt line has a priority assigned.
- Higher priority lines can interrupt lower priority lines.

ISA Bus Interrupt System

- ISA bus chains two 8259As together
- Link is via interrupt 2
- It gives 15 lines
    - 16 lines less one for link
- IRQ 9 is used to re-route anything trying to use IRQ 2
    - Backwards compatibility
- Incorporated in chip set

82C59A Interrupt Controller



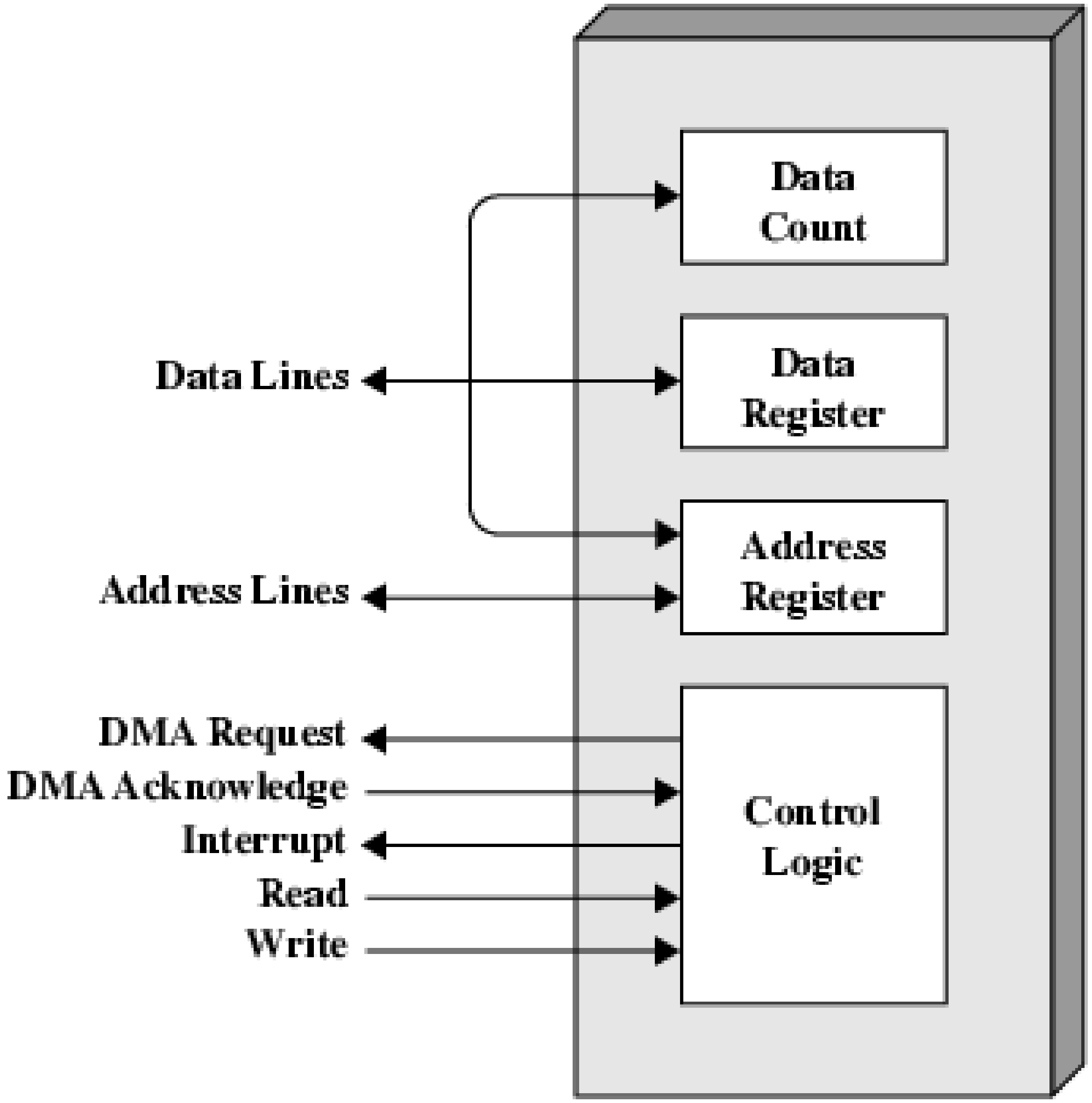
Direct Memory Access (DMA)

- Interrupt driven and programmed I/O require active CPU intervention.
  - Transfer rate is limited.
  - CPU is tied up (it cannot do anything else).
- DMA is the answer.

DMA Function

- Additional Module (hardware) on bus (DMA Controller).
- DMA controller takes over the control from CPU for I/O operation.
- At present: system bus clients are able to control system bus and generate control signals into the system bus (bus mastering).

Typical DMA Module Diagram



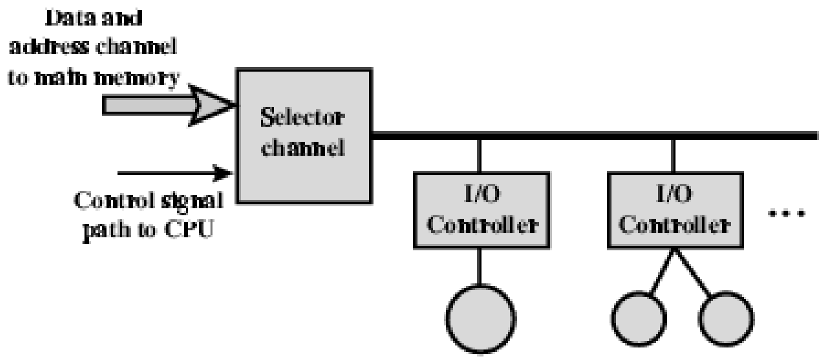
DMA Operation

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

I/O Channels

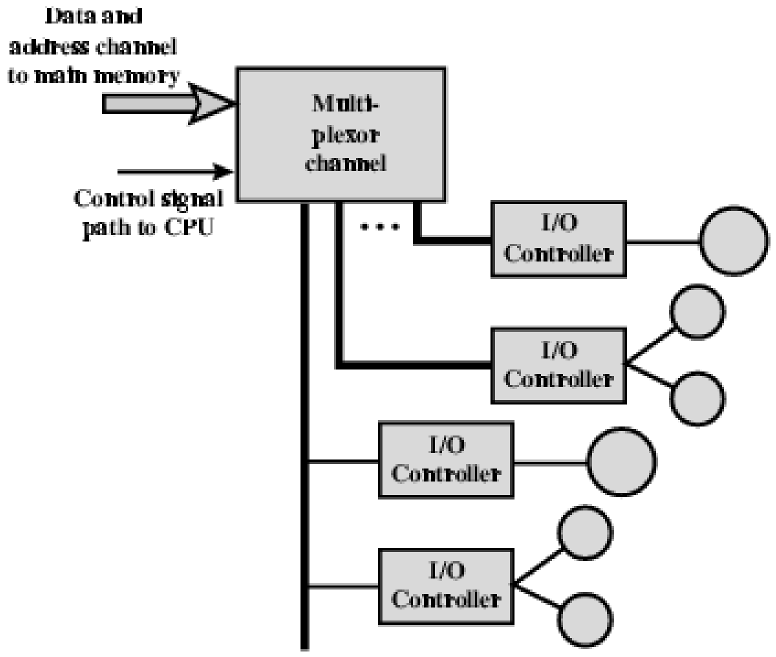
- I/O devices getting more sophisticated.
- e.g. graphics cards.
- CPU instructs I/O controller to do transfer.
- I/O controller does entire transfer.
- It improves speed
  - Takes load off CPU,
  - Dedicated processor is faster.

I/O Channel Architecture



Data and address channel to main memory
Selector channel
Control signal path to CPU
I/O Controller
I/O Controller
• • •

(a) Selector



(b) Multiplexor