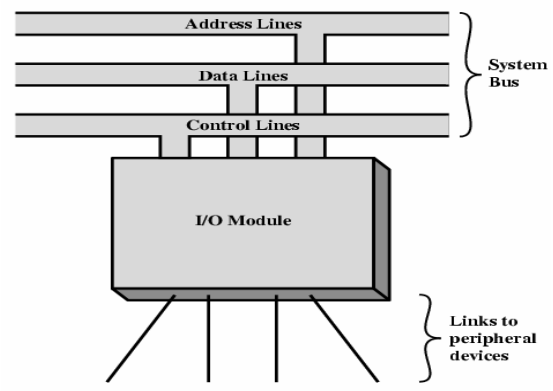
Input/Output Problems

- Wide variety of peripherals
  - — Delivering different amounts of data
  - — At different speeds
  - — In different formats
- All slower than CPU and RAM
- Need I/O modules
  - — Interface to CPU and Memory via system bus or Central Switch
  - — Interface to one or more peripherals

Generic Model of I/O Module



I/O Module Function

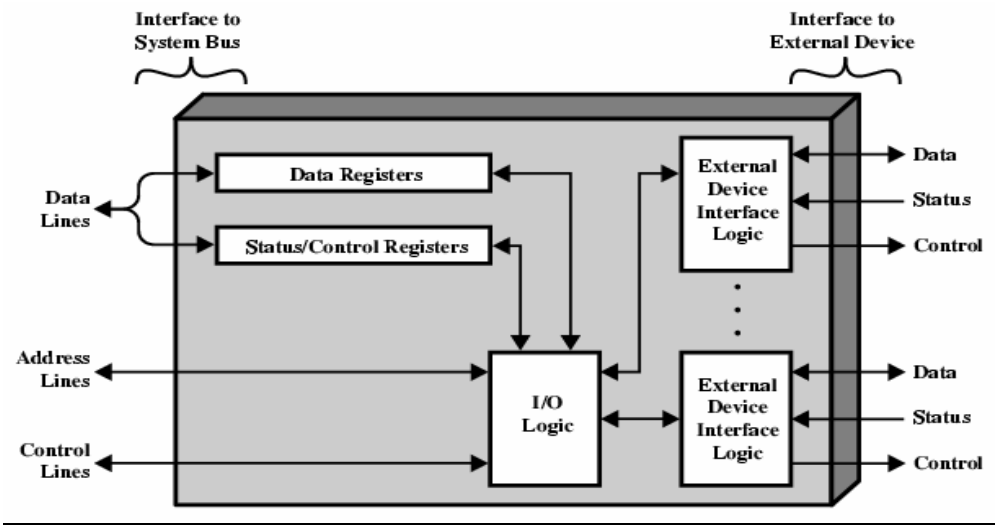
- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection

I/O 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.

Acknowledgement:
"Computer Organization and Architecture", by William Stallings, 6th edition

I/O Module Structure



The I/O module must be able to recognize and generate addresses associated with the device it controls.
Each I/O module has a unique address, or, if it controls more than one external device, a unique set of addresses.

Input Output Techniques:

a) 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 (processor is dedicated to the task of I/O and can therefore transfer data at a higher rate)

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

How does it work:-

- 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

Acknowledgement:
"Computer Organization and Architecture", by William Stallings, 6th edition

b) Interrupt-Driven I/O

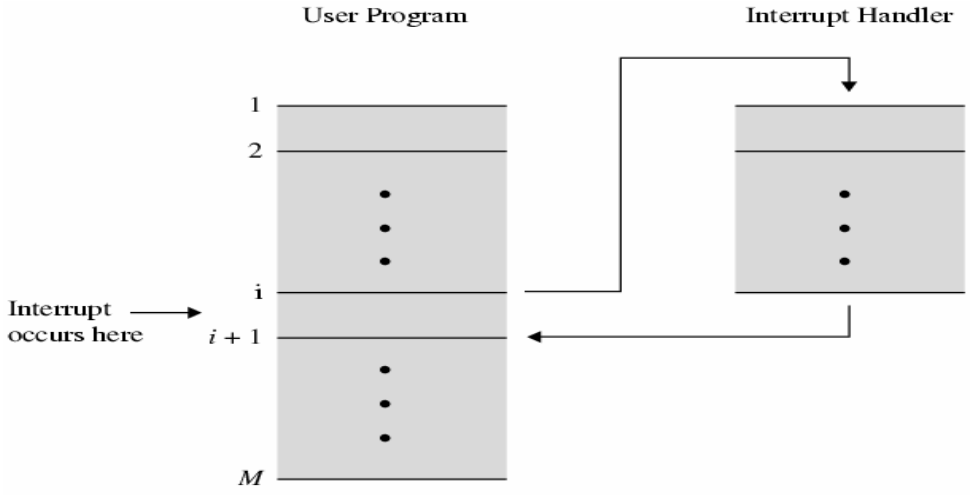
- Overcomes CPU waiting
- No repeated CPU checking of device
- I/O module interrupts when ready (frees up the processor to some extent at the expense of I/O transfer rate)

How does it work:-

- 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

Interrupt Processing:

- Device issues an interrupt signal to the processor
- Processor checks for interrupt
  - Indicated by an interrupt signal
- If no interrupt, fetch next instruction
- If interrupt pending:
  - Suspend execution of current program
  - Save context
  - Set PC to start address of interrupt handler routine
  - Process interrupt
  - Restore context and continue interrupted program



Acknowledgement:
"Computer Organization and Architecture", by William Stallings, 6th edition

Multiple Interrupts

- Disable interrupts
  - Processor will ignore further interrupts whilst processing one interrupt
  - Interrupts remain pending and are checked after first interrupt has been processed
  - Interrupts handled in sequence as they occur
- Define priorities
  - Low priority interrupts can be interrupted by higher priority interrupts
  - When higher priority interrupt has been processed, processor returns to previous interrupt

Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up managing an I/O transfer
- DMA is the answer when large volumes of data are to be moved

DMA involves an additional Module (hardware) on bus.
DMA controller takes over from CPU for I/O.

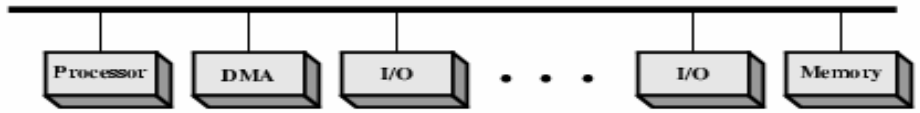
DMA Operation

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

Acknowledgement:
"Computer Organization and Architecture", by William Stallings, 6th edition

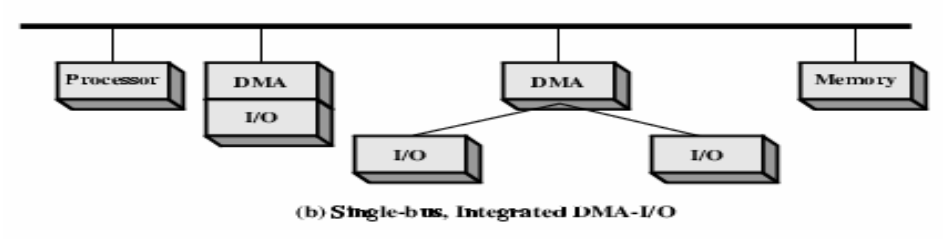
DMA Configurations

1) Single Bus, Detached DMA controller



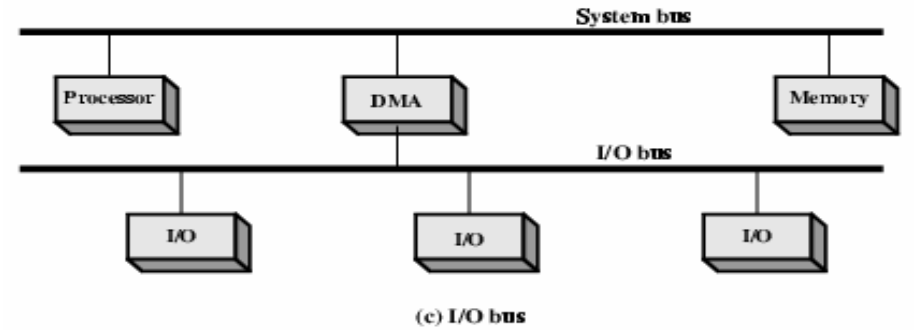
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory
- CPU is suspended twice

2) Single Bus, Integrated DMA controller



- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

3) Separate I/O Bus



- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

Acknowledgement:
"Computer Organization and Architecture", by William Stallings, 6th edition