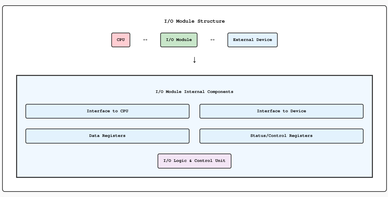
I/O Module and Interrupt System Block Diagrams

a) I/O Module Block Diagram & 2+6 Input Techniques



I/O Module Functions:

Control & Timing: Coordinates data transfer between CPU and peripherals

CPU Communication: Handles data, control, and status information exchange

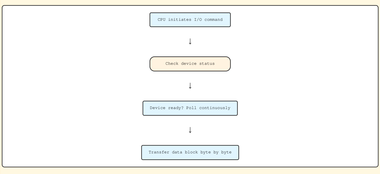
Device Communication: Manages device-specific protocols and timing

Data Buffering: Temporarily stores data during transfer operations

Error Detection: Monitors and reports transmission errors

Input Techniques for Block Data Transfer (2+6)

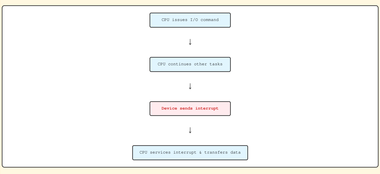
1. Programmed I/O (Polling)



Advantage: Simple implementation, full CPU control

Disadvantage: CPU wastage due to continuous polling

2. Interrupt-Driven I/O



Advantage: Better CPU utilization

Disadvantage: Overhead for each byte transfer

Additional Techniques (6 more):

3. Direct Memory Access (DMA): Direct data transfer between memory and I/O device

4. Channel I/O: Dedicated processors handle I/O operations

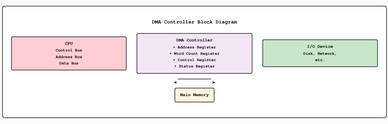
5. Memory-Mapped I/O: I/O devices accessed through memory addresses

6. Isolated I/O: Separate address space for I/O operations

7. Buffered I/O: Use of intermediate buffers for data transfer

8. Scatter-Gather I/O: Transfer data to/from multiple memory locations

b) DMA Controller Interfacing with CPU



DMA Operation Steps:

1. Initialization: CPU programs DMA controller with memory address, transfer count, and direction

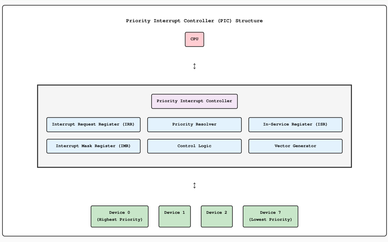
2. Bus Request: DMA controller requests bus control from CPU

3. Bus Grant: CPU grants bus control and enters wait state

4. Data Transfer: DMA directly transfers data between memory and I/O device

5. Bus Release: DMA releases bus control back to CPU after transfer completion

c) Priority Interrupt Controller Interfacing with CPU

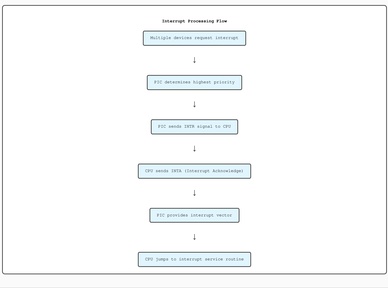


Priority Resolution Methods:

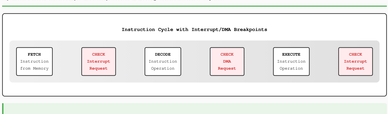
Fixed Priority: Each interrupt source has a predetermined priority level

Rotating Priority: Priority levels rotate after each interrupt service

Masked Interrupts: Software can selectively enable/disable interrupt sources



d) DMA and Interrupt Breakpoints During Instruction Cycle



Breakpoint Locations:

1. After Fetch: Check for interrupts before decoding next instruction

2. Before Execute: Check for DMA requests that need immediate bus access

3. After Execute: Check for interrupts generated by the completed instruction

Interrupt vs DMA Handling:

Interrupt Breakpoint: CPU saves context, jumps to ISR, processes interrupt, then returns

DMA Breakpoint: CPU temporarily halts, grants bus to DMA controller, resumes after transfer

Simultaneous Requests: Hardware interrupts typically have higher priority than DMA requests

Summary

These diagrams illustrate the key concepts of I/O interfacing, DMA operations, interrupt handling, and their integration within the CPU instruction cycle. Each component plays a crucial role in efficient computer system operation.