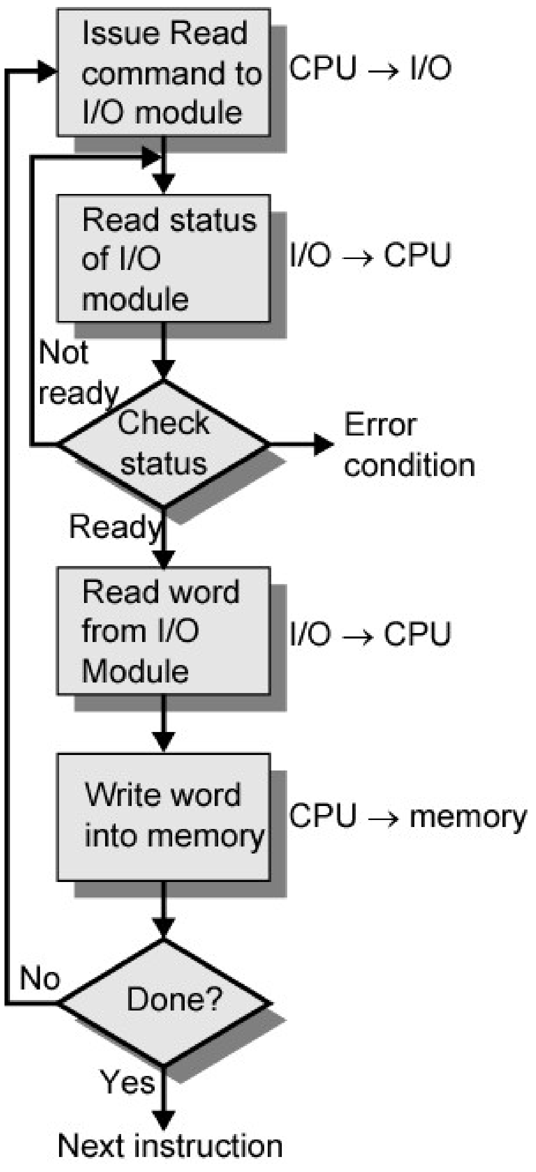
CS 223 Computer Architecture & Organization

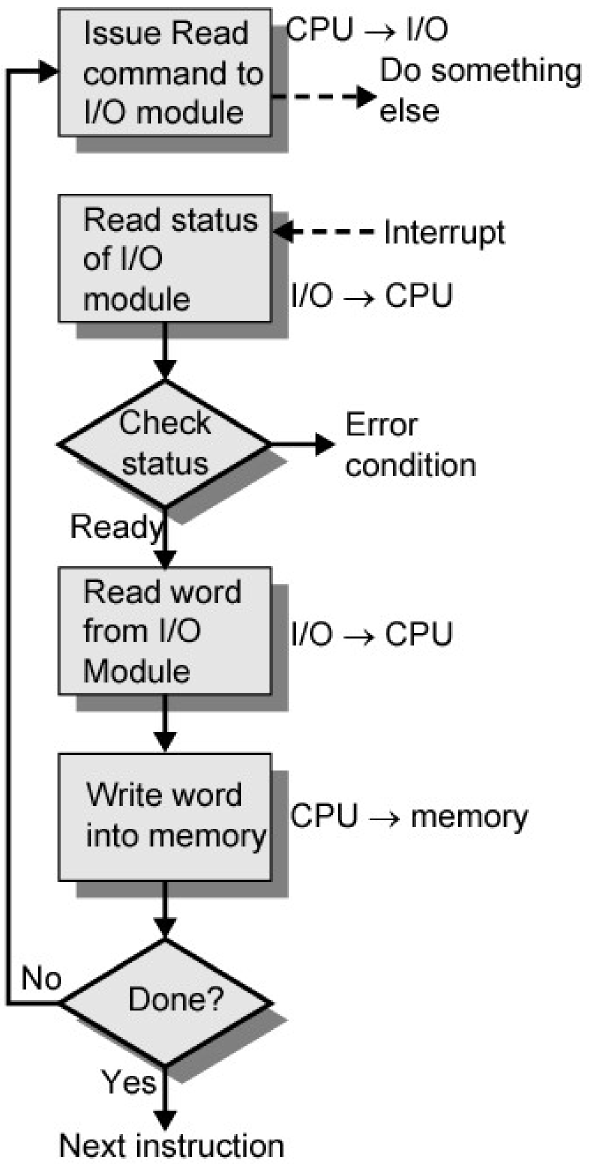
Input/Output

J. K. Deka
Professor
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati, Assam.

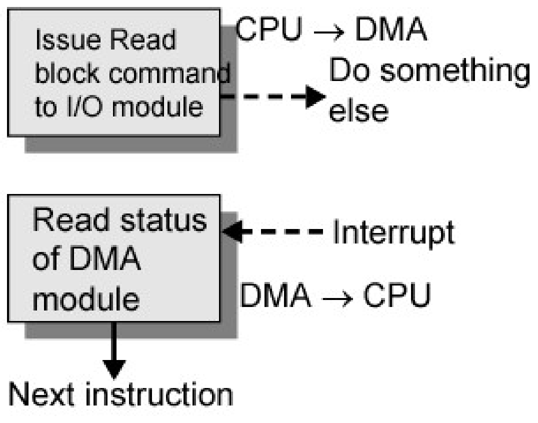
Three Techniques



(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

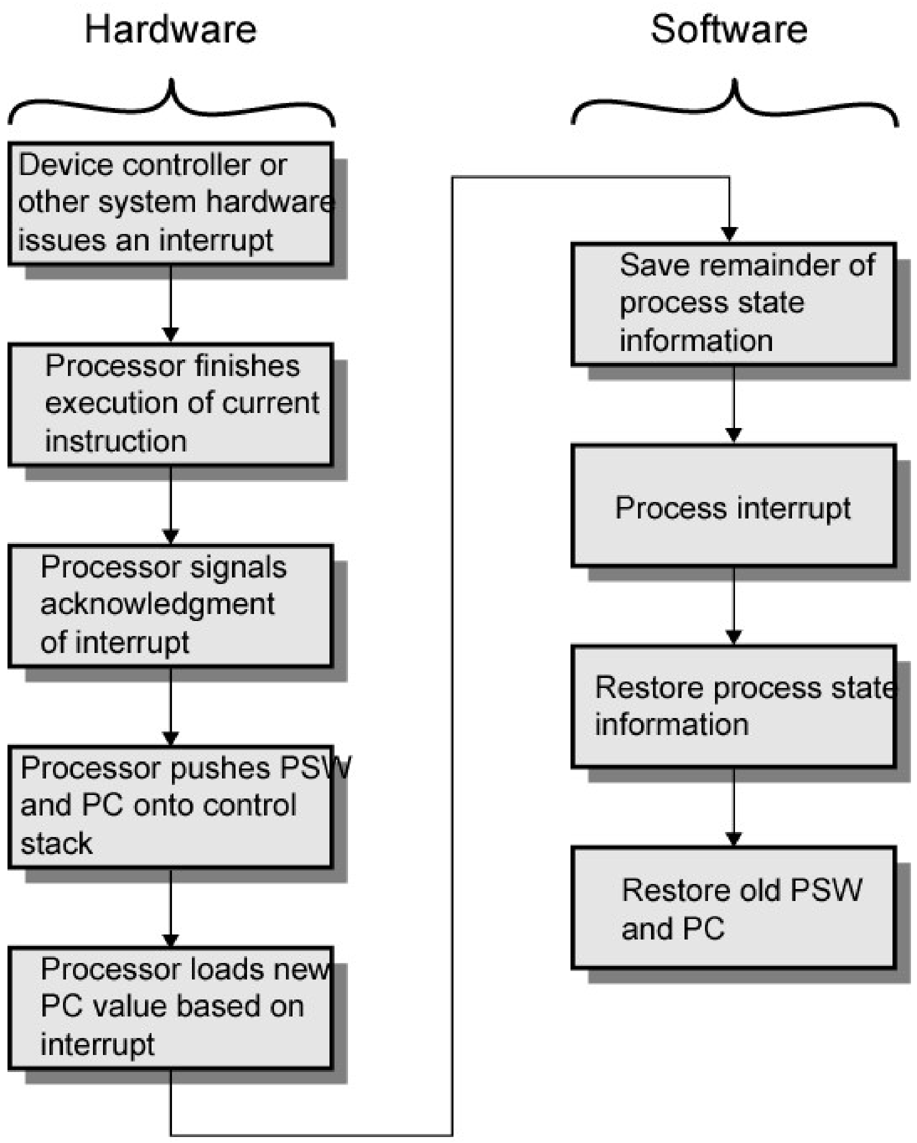
Interrupt Driven I/O

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

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

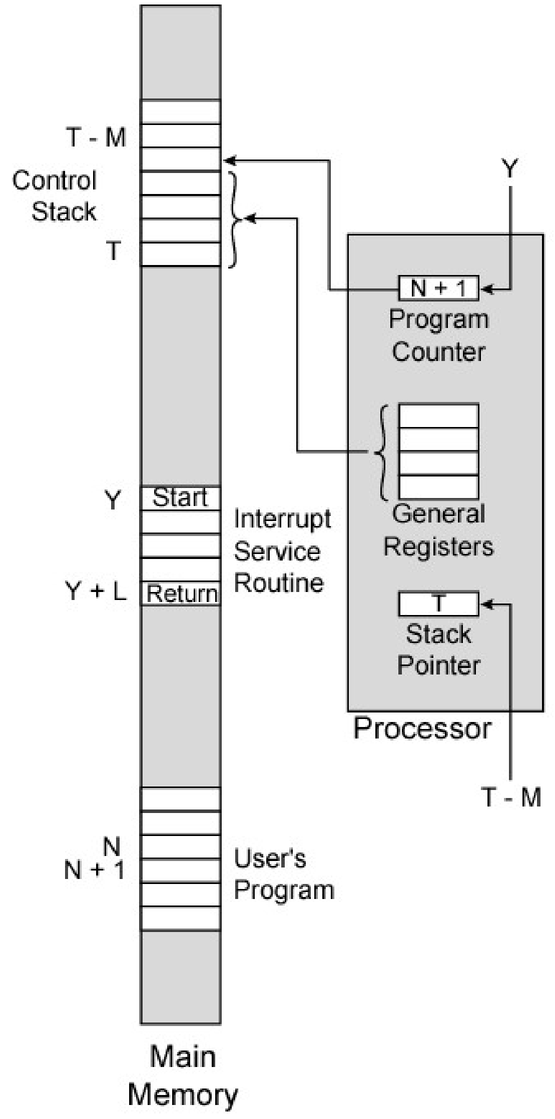
Simple Interrupt Processing


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

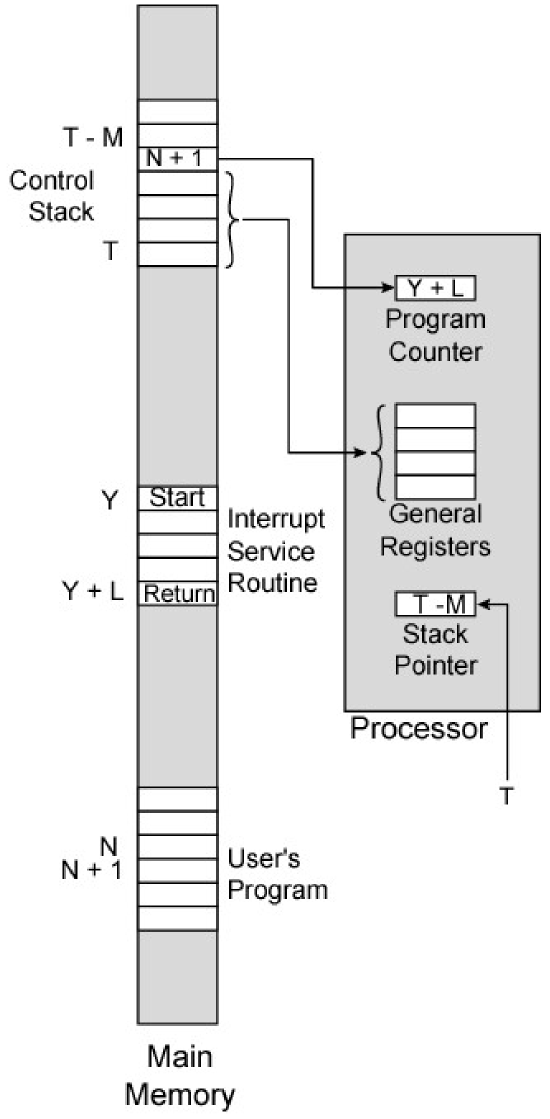
Program Status Word

- A set of bits
- Includes Condition Codes
  - Sign of last result
  - Zero
  - Carry
  - Equal
  - Overflow
- Interrupt enable/disable
- Supervisor

Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

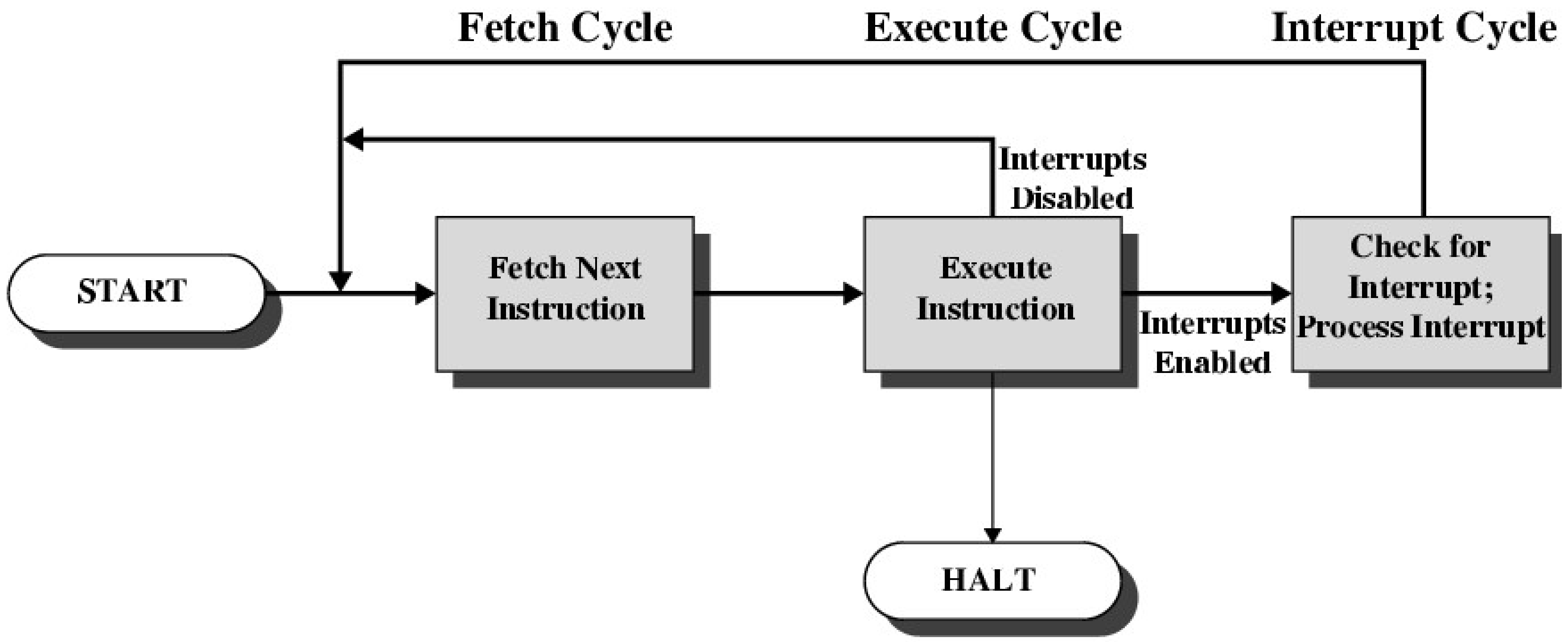


(b) Return from interrupt

CPU Viewpoint

- Issue read command
- Do other work
- Check for interrupt at end of each instruction cycle
- If interrupted:-
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

Instruction Cycle with Interrupts



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

- Different line for each module
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow

Identifying Interrupting Module

- Software poll
  - CPU branches to an interrupt service routine
  - Poll each I/O module to determine which module caused the interrupt
  - Can be done by separate command line, TESTI/O
    - The processor raises TESTI/O and places the address
    - The I/O module responds positively if it set the interrupt
  - Alternatively, by an addressable status register
    - The processor reads the status register to identify the interrupting module

Identifying Interrupting Module

- Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine
- Bus Master
  - Module must claim the bus before it can raise interrupt
  - e.g. PCI & SCSI

Identifying Interrupting Module

- Daisy Chain or Hardware poll
  - Interrupt Acknowledge sent down a chain
  - Module responsible places vector on bus
  - CPU uses vector to identify handler routine

Multiple Interrupts

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

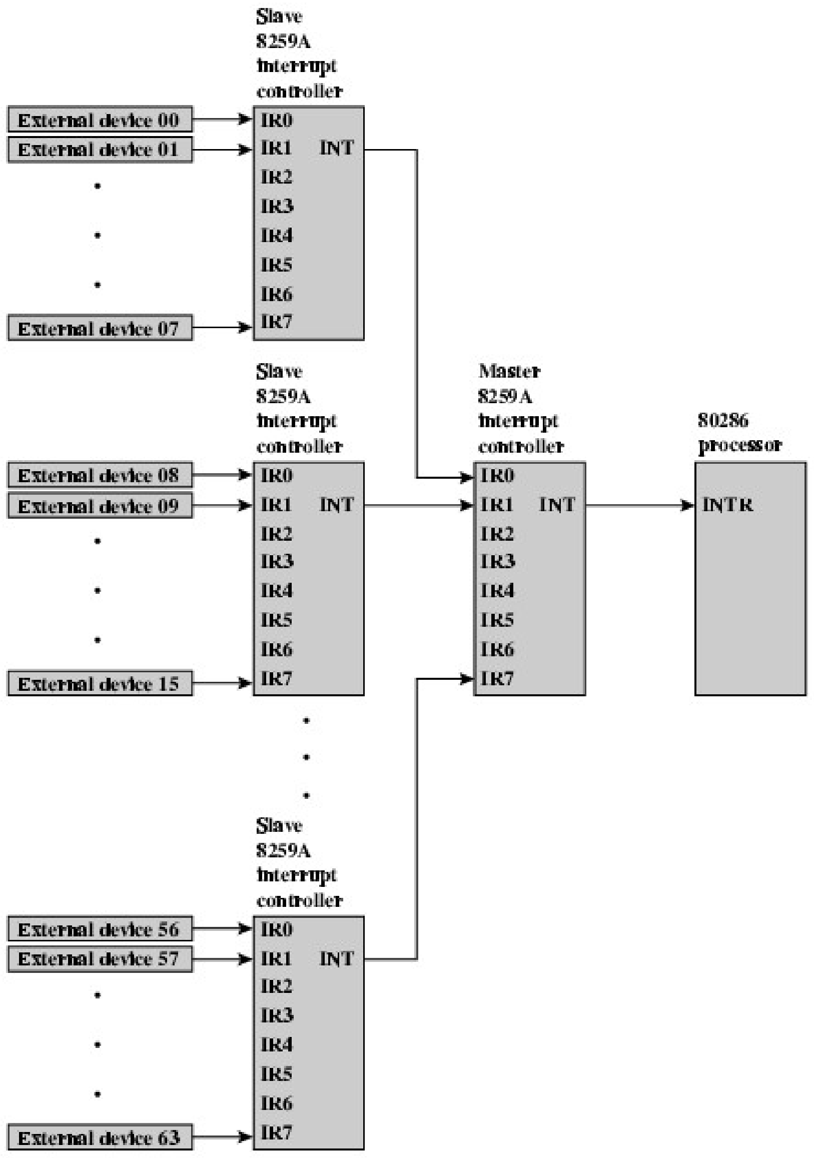
Example - PC Bus

- 80x86 has one interrupt line
- 8086 based systems use one 8259A interrupt controller
- 8259A has 8 interrupt lines

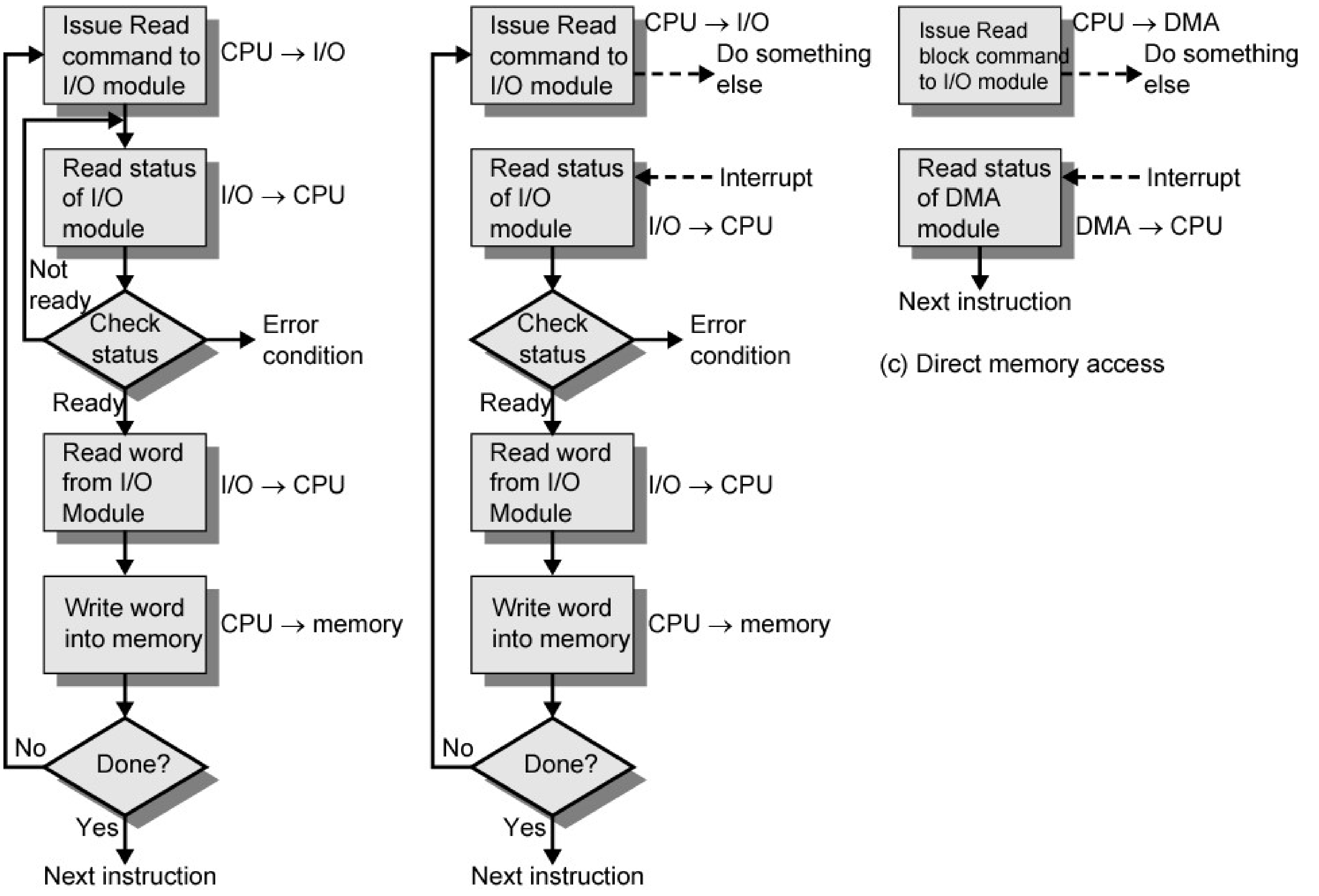
Sequence of Events

- 8259A accepts interrupts
- 8259A determines priority
- 8259A signals 8086 (raises INTR line)
- CPU Acknowledges
- 8259A puts correct vector on data bus
- CPU processes interrupt

82C59A Interrupt Controller



Three Techniques



(a) Programmed I/O

(b) Interrupt-driven I/O

(c) Direct memory access

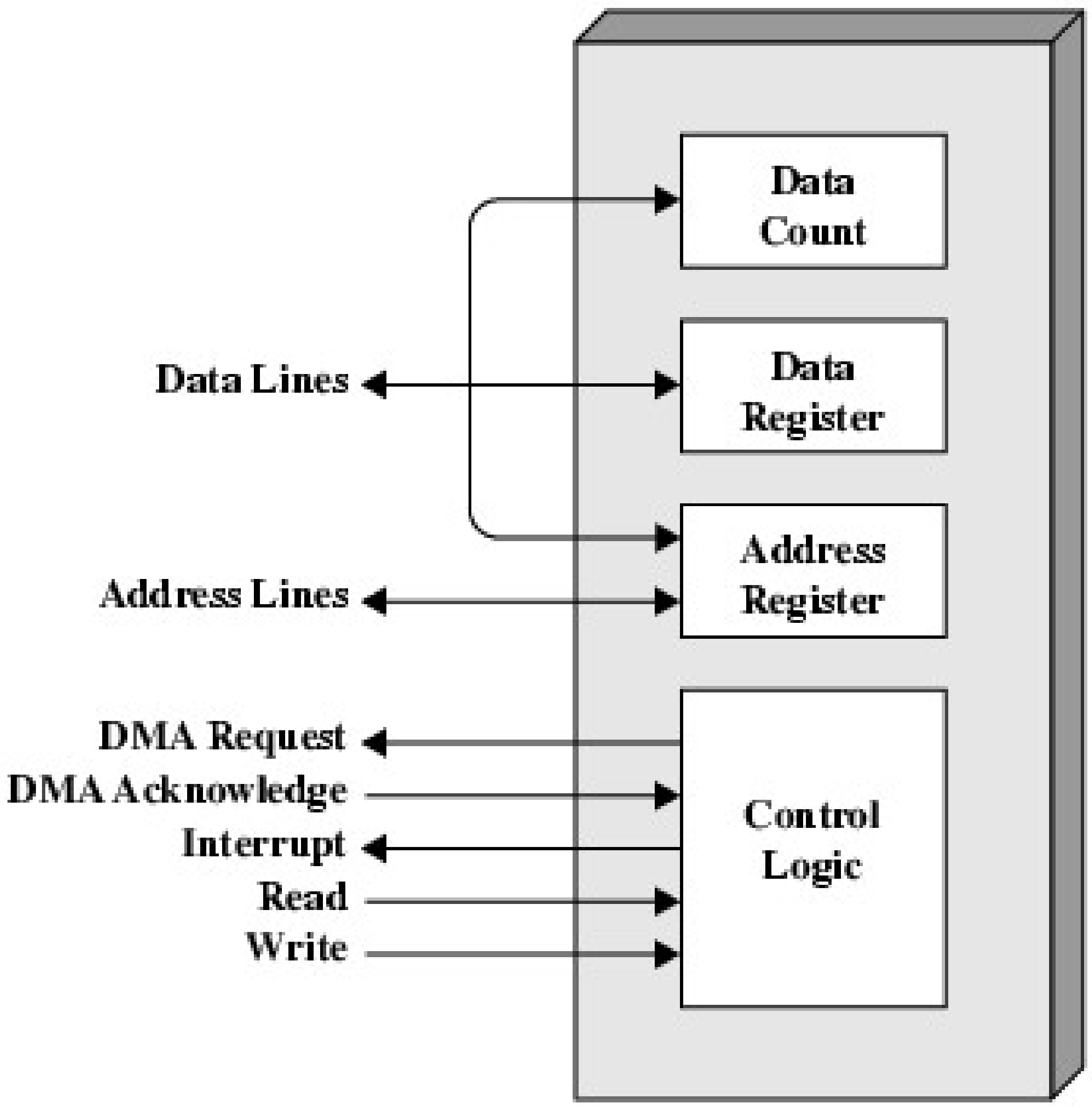
Direct Memory Access (DMA)

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA is the answer

DMA Function

- Additional Module (hardware) on bus
- DMA controller takes over from CPU for I/O

Typical DMA Module Diagram



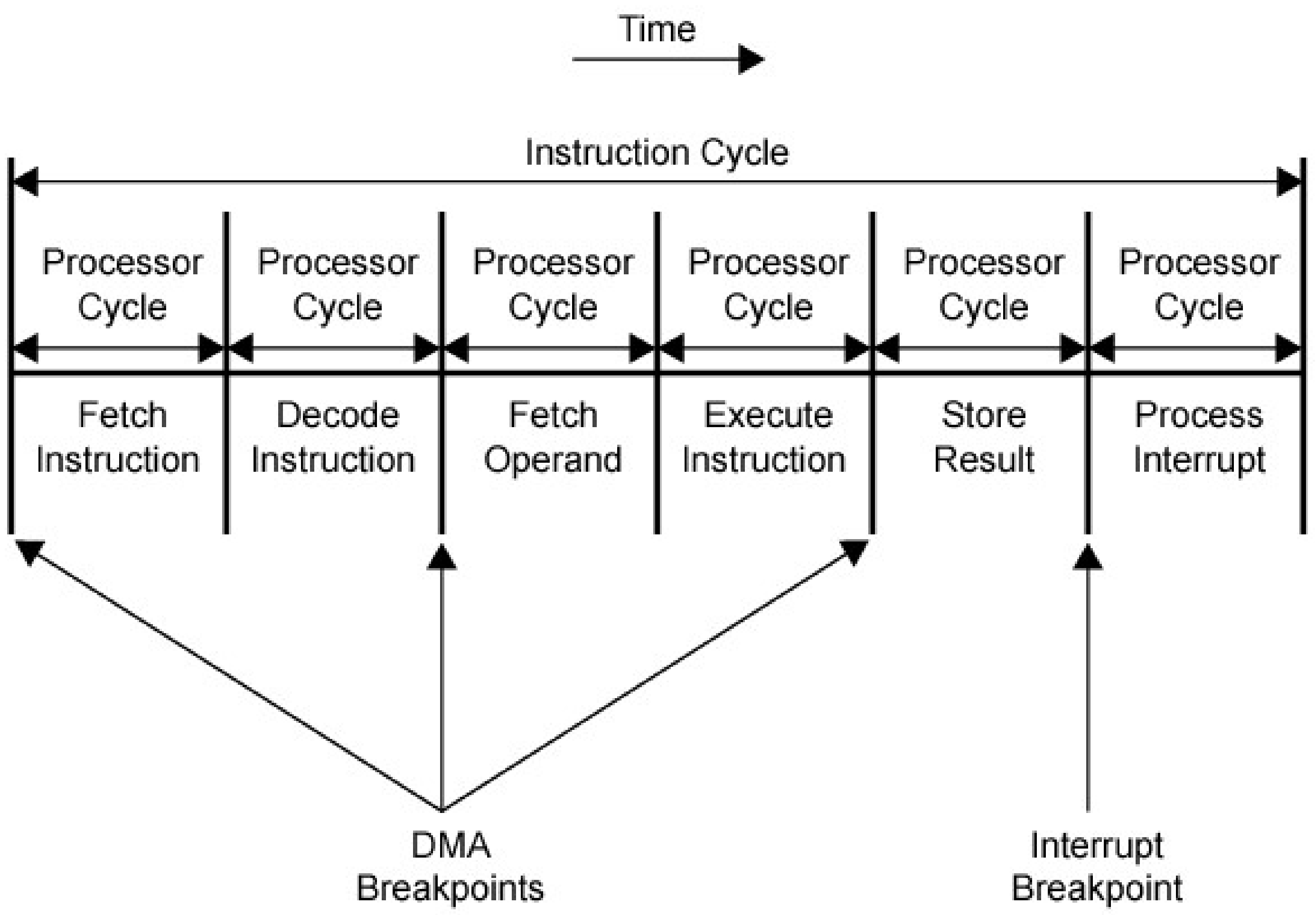
DMA Operation

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

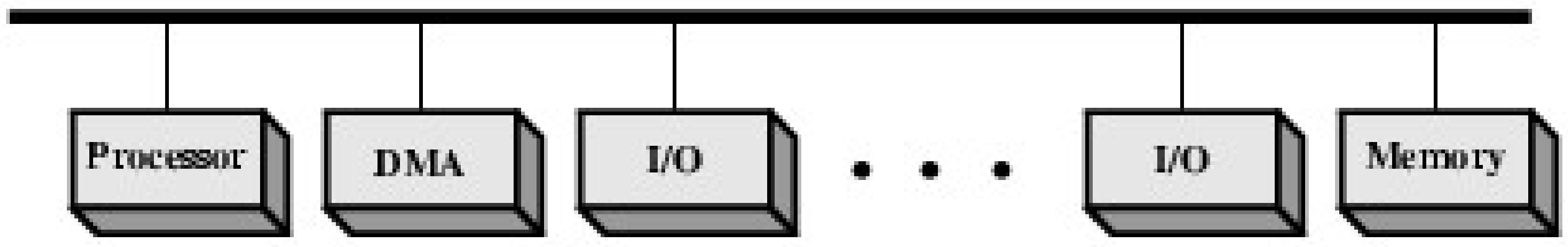
DMA Transfer

- DMA controller takes over bus
- Transfer of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

DMA and Interrupt Breakpoints During an Instruction Cycle

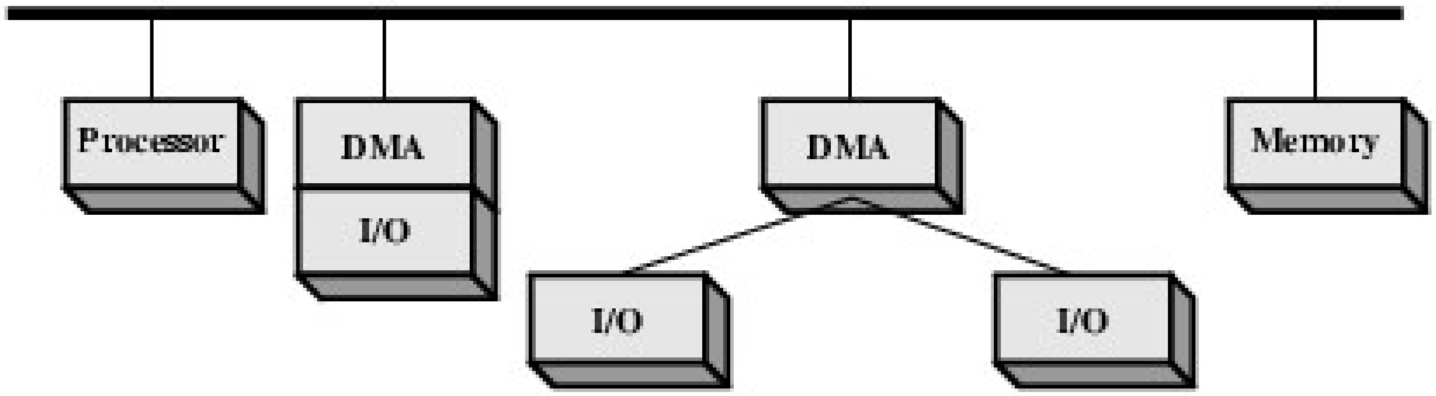


DMA Configurations (1)



- Single Bus, Detached DMA controller
- Each transfer uses bus twice
  - I/O to DMA then DMA to memory
- CPU is suspended twice

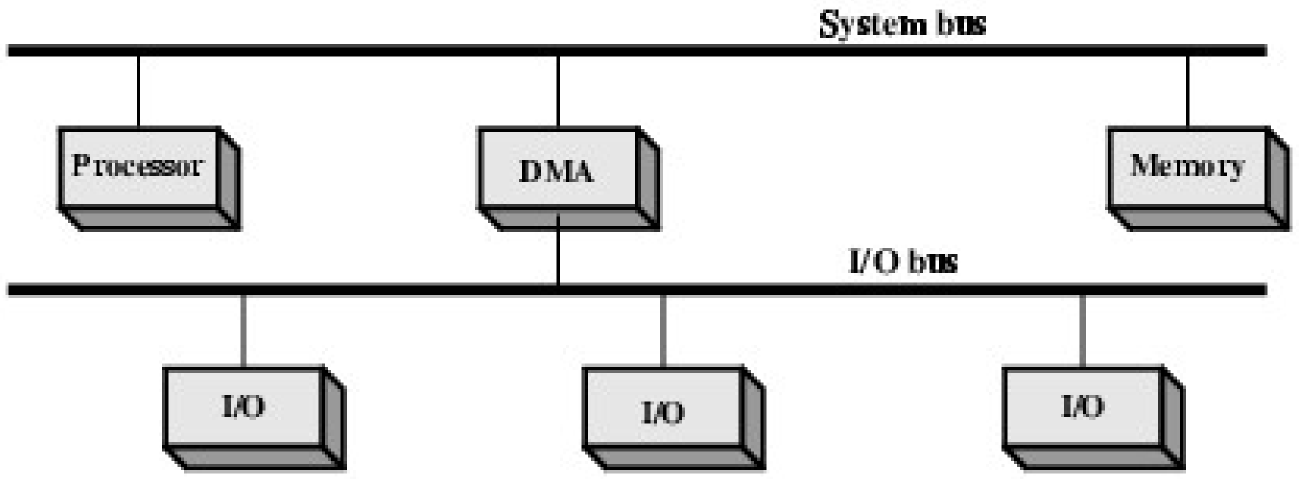
DMA Configurations (2)



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

- Single Bus, Integrated DMA controller
- Controller may support >1 device
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

DMA Configurations (3)



(c) I/O bus

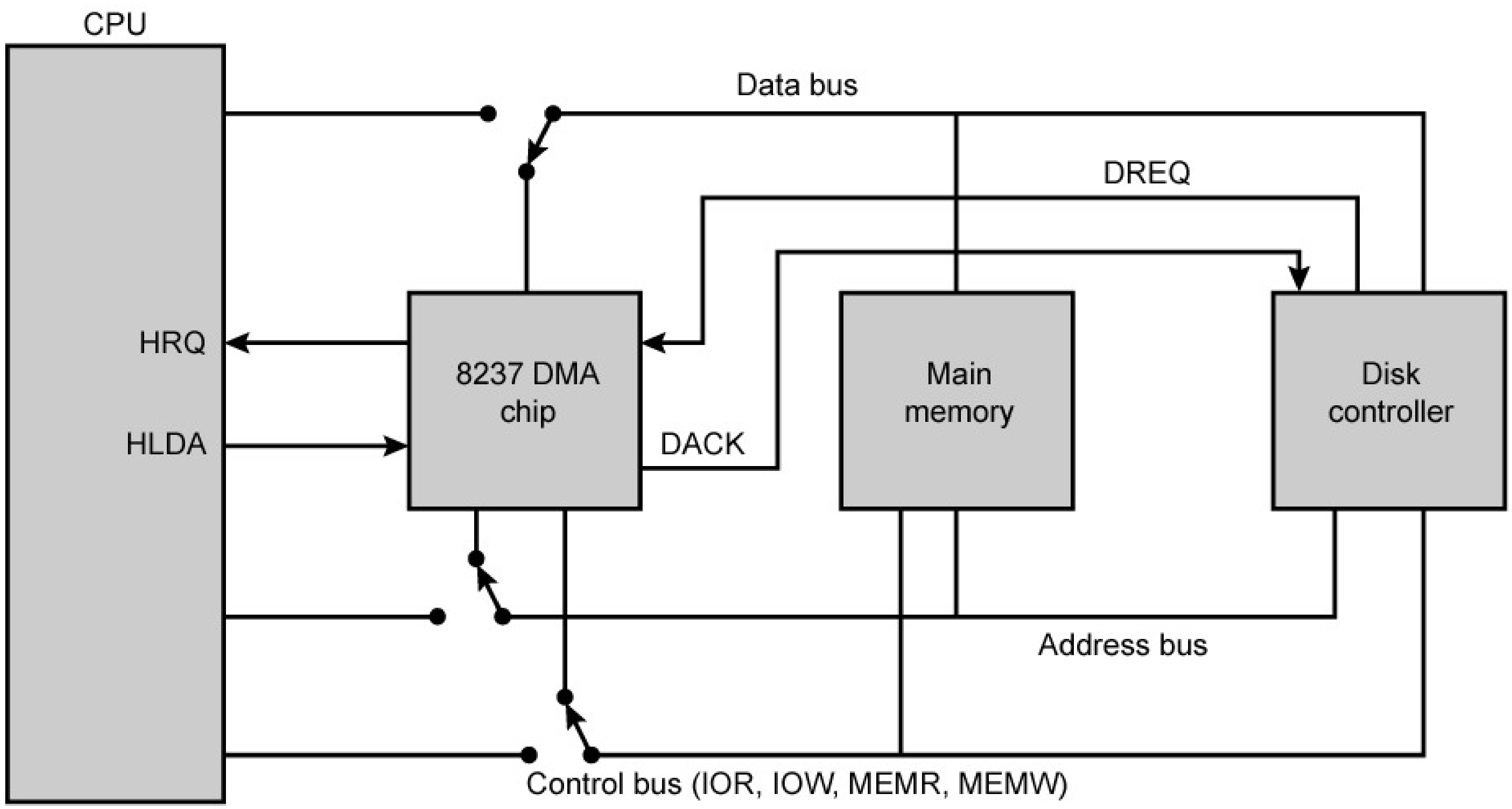
- Separate I/O Bus
- Bus supports all DMA enabled devices
- Each transfer uses bus once
  - DMA to memory
- CPU is suspended once

Intel 8237A DMA Controller

- Interfaces to 80x86 family and DRAM
- When DMA module needs buses it sends HOLD signal to processor
- CPU responds HLDA (hold acknowledge)
  - DMA module can use buses
- E.g. transfer data from memory to disk

1. Device requests service of DMA by pulling DREQ (DMA request) high
2. DMA puts high on HRQ (hold request),
3. CPU finishes present bus cycle (not necessarily present instruction) and puts high on HDLA (hold acknowledge). HOLD remains active for duration of DMA
4. DMA activates DACK (DMA acknowledge), telling device to start transfer
5. DMA starts transfer by putting address of first byte on address bus and activating MEMR; it then activates IOW to write to peripheral. DMA decrements counter and increments address pointer. Repeat until count reaches zero
6. DMA deactivates HRQ, giving bus back to CPU

8237 DMA Usage of Systems Bus



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

Reference

Computer Organization and Architecture – Designing for Performance
William Stallings, Seventh Edition

Chapter 7: Input/Output
Page No.: 200 - 227