CS209 Computer Architecture

I/O Techniques

Somanath Tripathy

IIT Patna

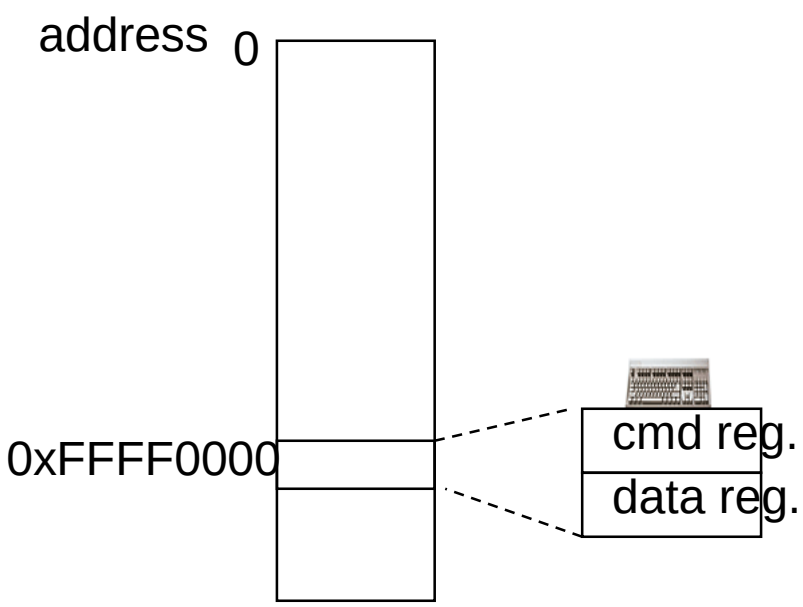
This Class

- I/O Techniques
  - Program controlled Data Transfer
    - Memory mapped and Isolated I/O
  - Interrupt Driven Data Transfer
  - Direct Memory Access (DMA)

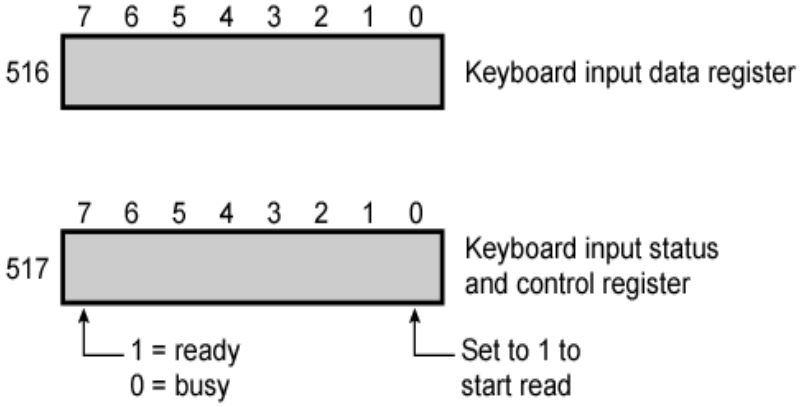
Quiz Test on Next week?

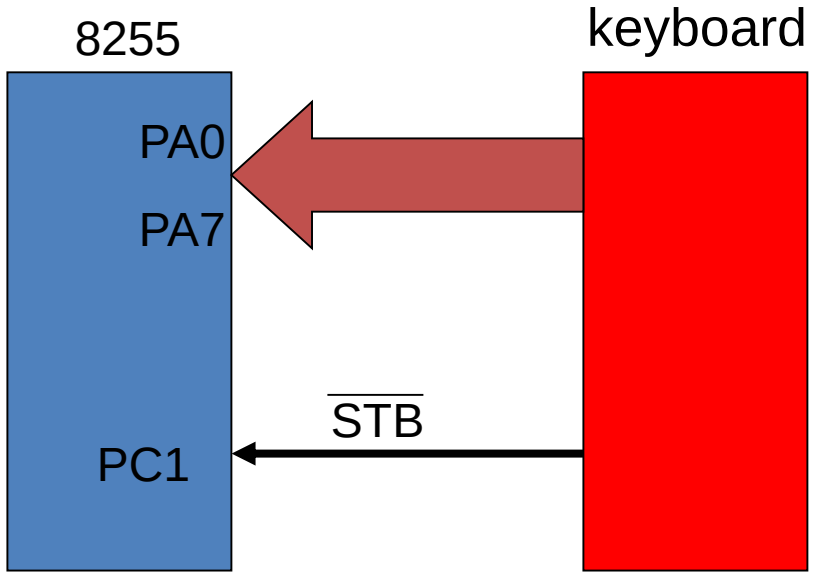
Addressing I/O Devices

- Under programmed I/O, data transfer is very like memory access (CPU viewpoint)
  - Each device given unique identifier
  - CPU commands contain identifier (address)
  - Memory mapped I/O
    - Devices 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
    - Need I/O or memory select lines
    - Special commands for I/O



Memory Mapped and Isolated I/O





ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

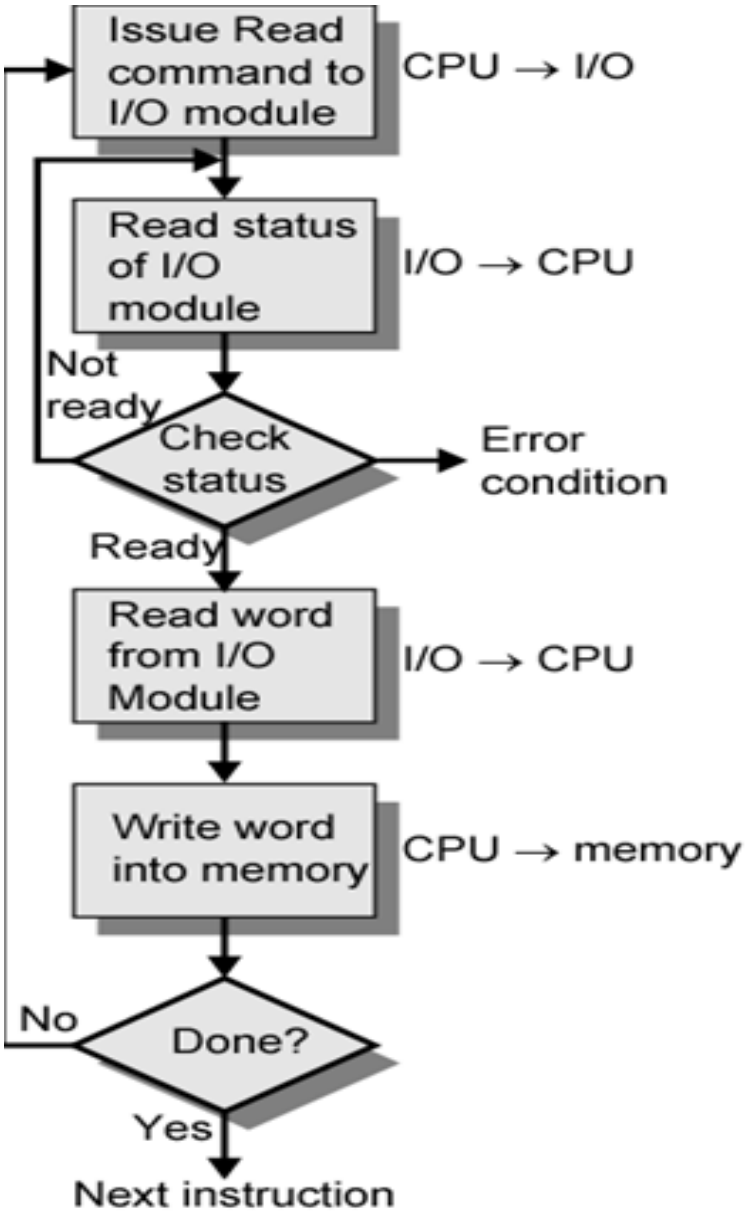
(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

Programmed I/O

- CPU has direct control over I/O
  - Sense status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Wastes CPU time
  - How to recover?



Polling Keyboard Status

- Program-controlled I/O implemented with a loop for polling keyboard status register:
-

```
READWAIT:  LoadByte   R4,      //KBD_STATUS
           And        R4, R4, #2
           Branch     R4=0    READWAIT
           LoadB      R5, PA  //KBD_DATA
```

- Keyboard circuit places character in KBD_DATA and sets KIN flag in KBD_STATUS
- Circuit clears KIN flag when KBD_STATUS read

BusyWait: CPU is not doing any usefu lwork during this period

Interrupt Driven I/O

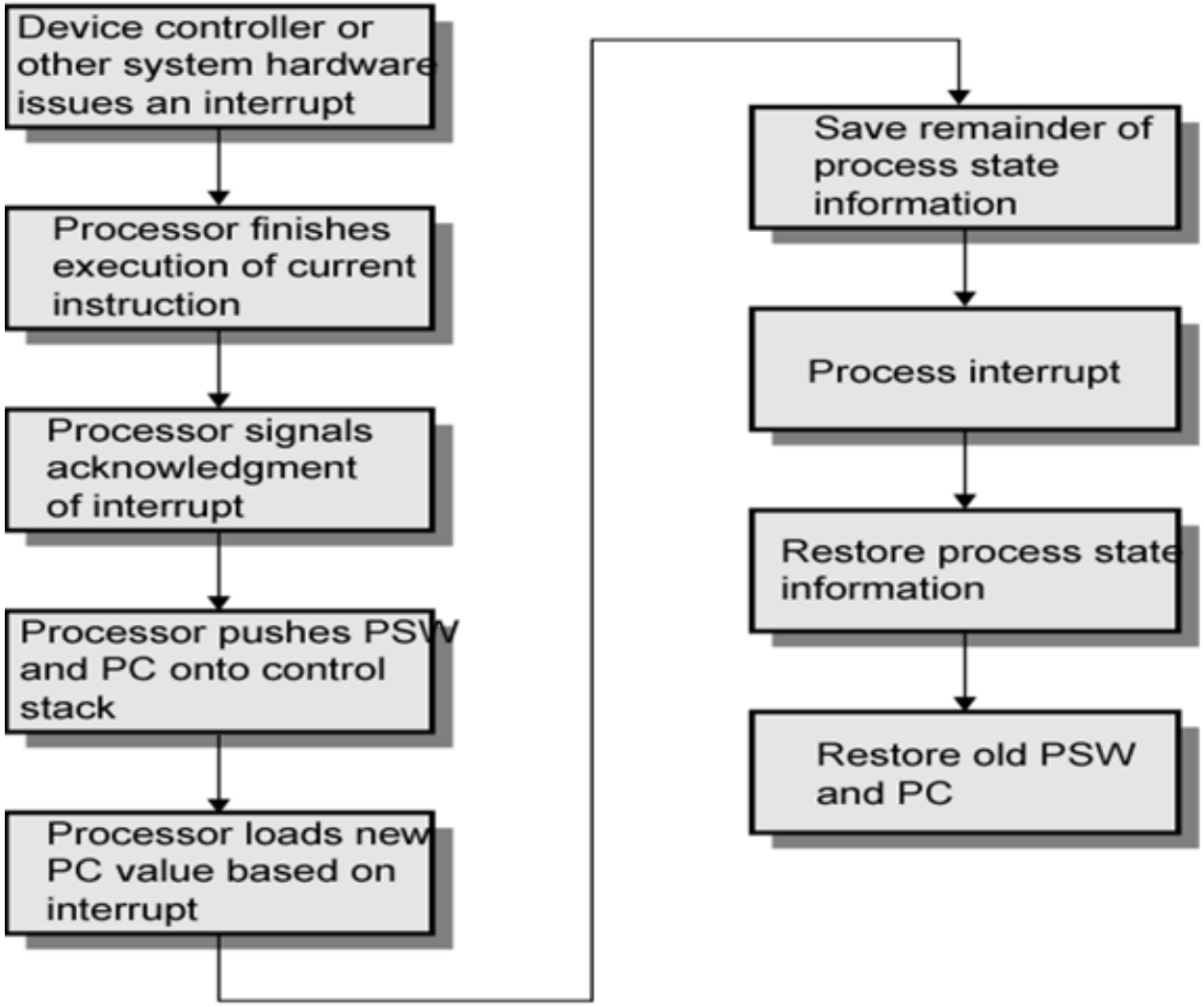
- Overcomes CPU waiting
  - No repeated CPU checking of device
- I/O module interrupts when ready

Interrupt Driven I/O
Basic Operation

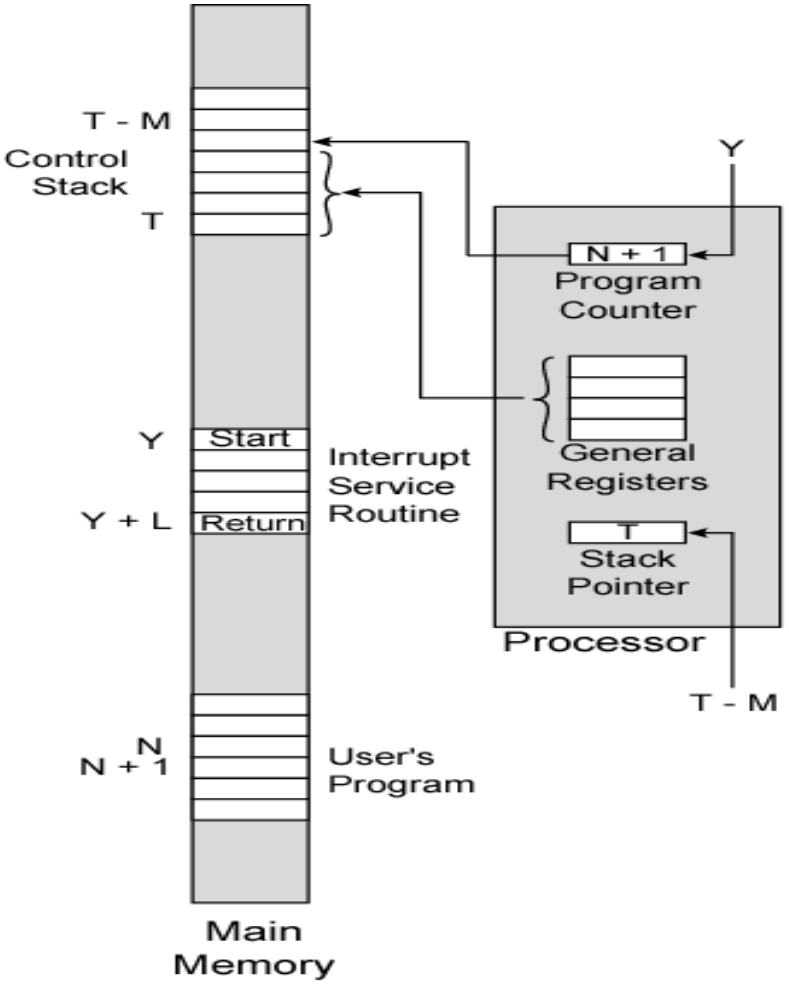
- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

CPU:
Check for interrupt at end of each instruction cycle
If interrupted:-
    Save context (registers)
    Process interrupt

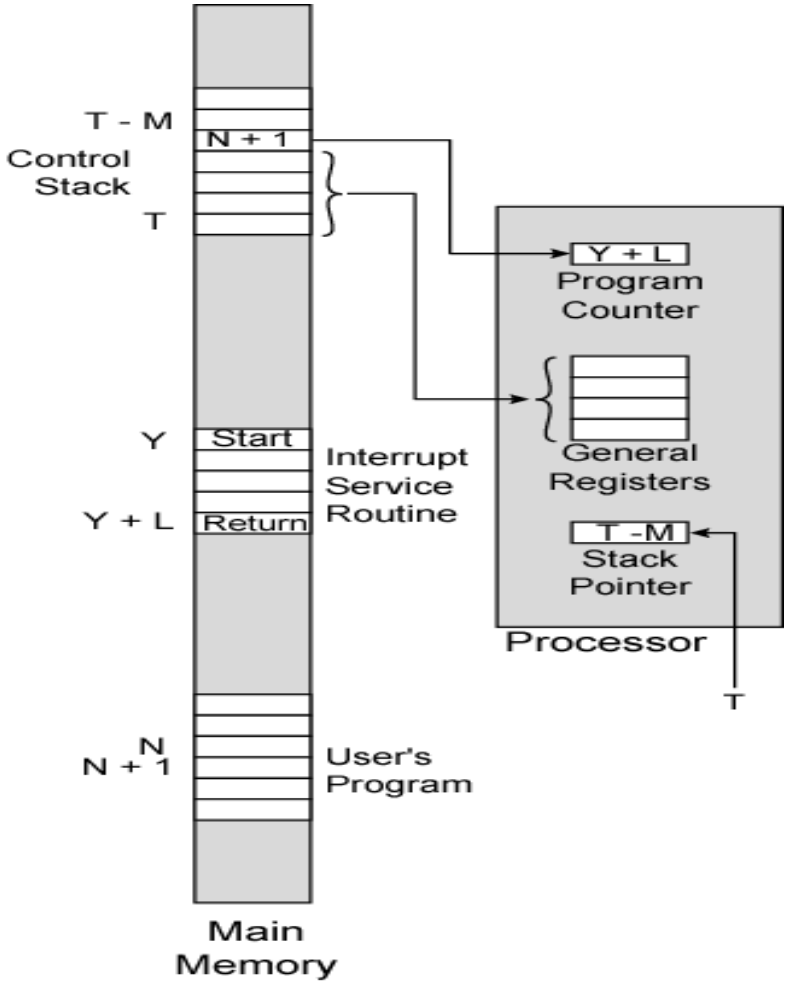
Simple Interrupt Processing



Changes in Memory and Registers for an Interrupt



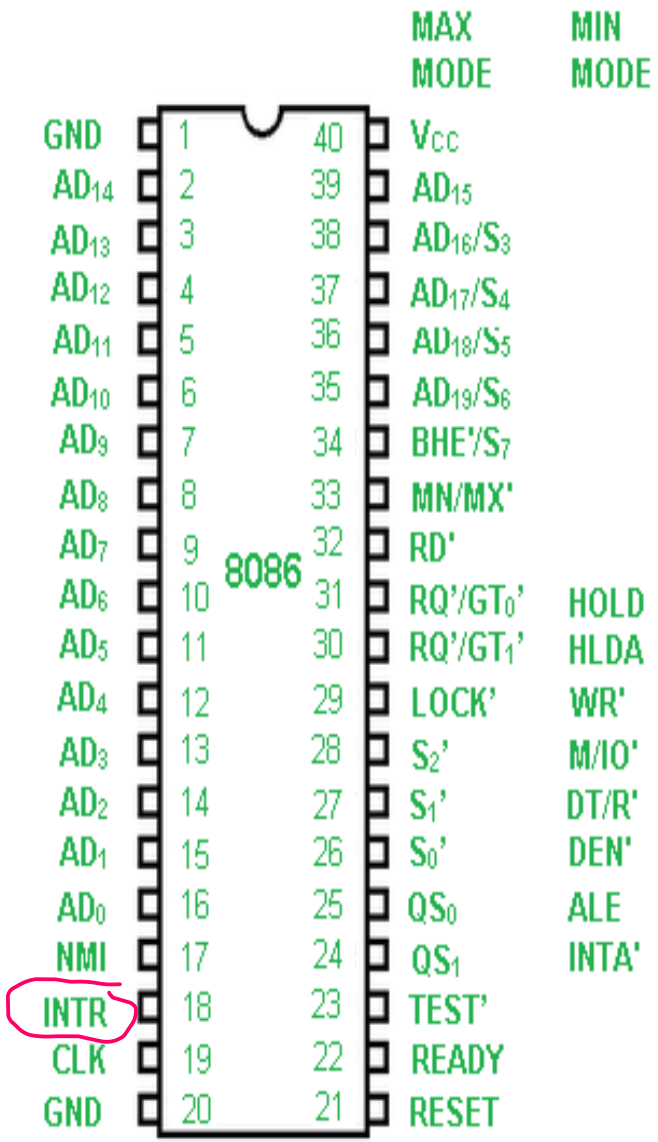
(a) Interrupt occurs after instruction at location N



(b) Return from interrupt

Design Issues

- How do you identify the module issuing the interrupt?
- Different line for each module
  - Limits number of devices
- Software poll
  - CPU asks each module in turn
  - Slow
- How do you deal with multiple interrupts?
  - i.e. an interrupt handler being interrupted



Identifying Interrupting Module

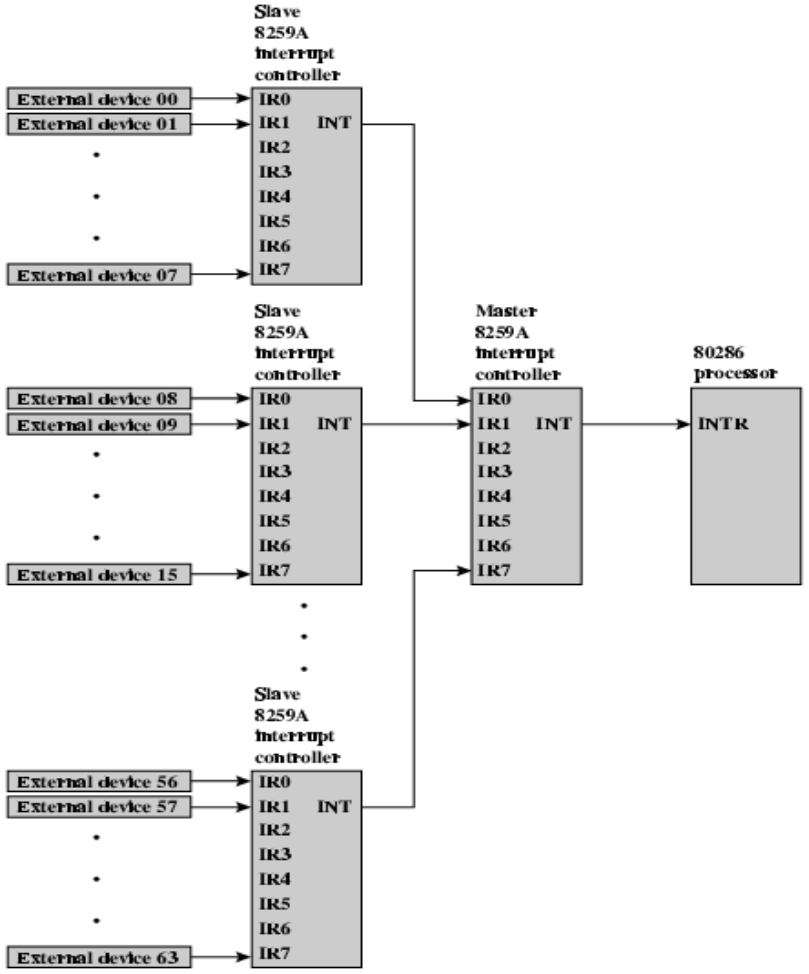
1. How do you identify the module issuing the interrupt?
   - Daisy Chain or Hardware poll
     - Interrupt Acknowledge sent down a chain
     - Module responsible places vector on bus
     - CPU uses vector to identify handler routine
   - Priority encoder

Example

- How do you deal with multiple interrupts?
- 80x86 has one interrupt line

- Multiple Interrupt (priority Encoder)
  - 80x86 based systems use 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



Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- Suitable Alternate is DMA
  - DMA controller takes over from CPU for I/O

DMA Operation

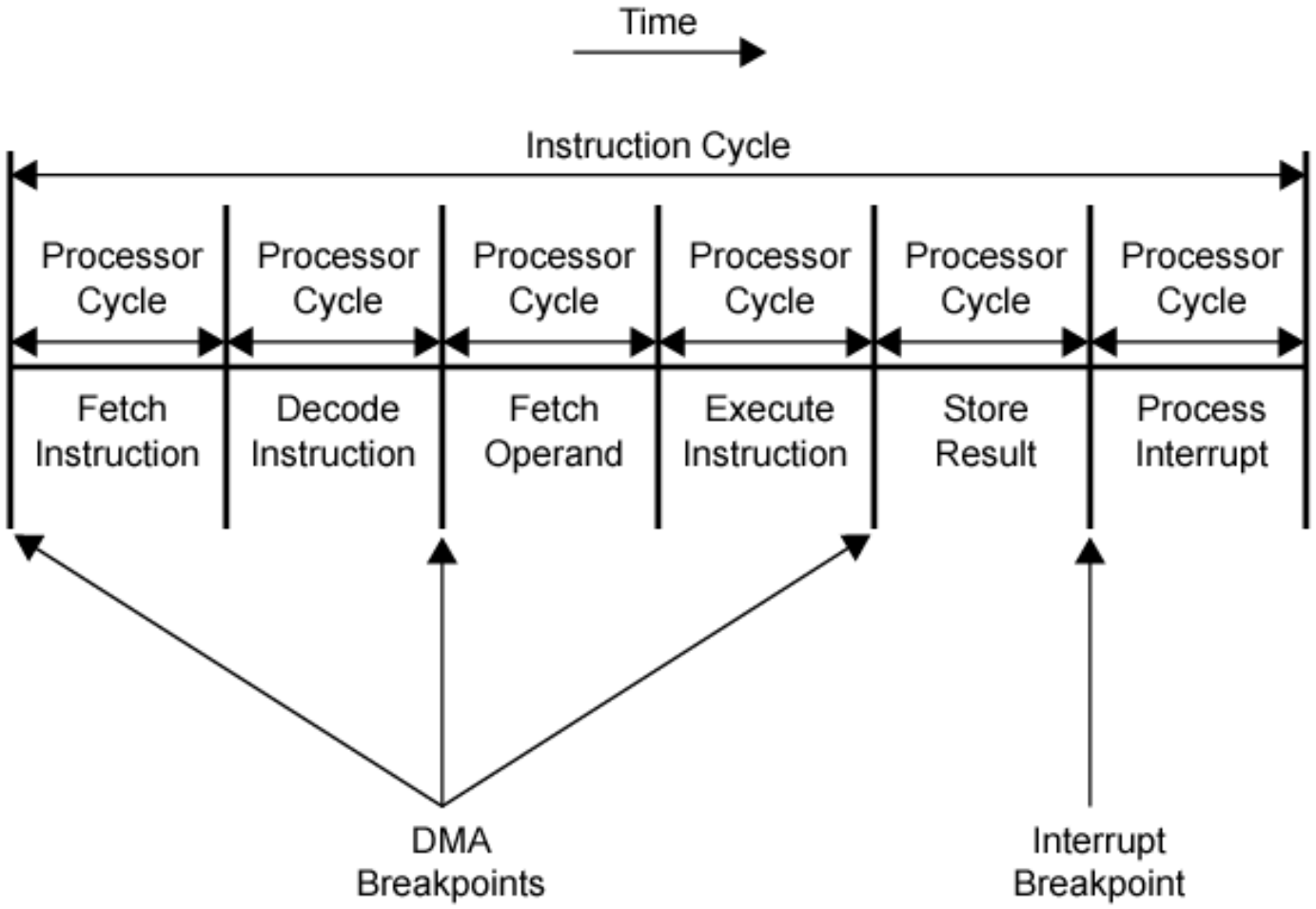
- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

DMA Transfer

Cycle Stealing:

- DMA controller takes over bus for a cycle
- Transfer of one word 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

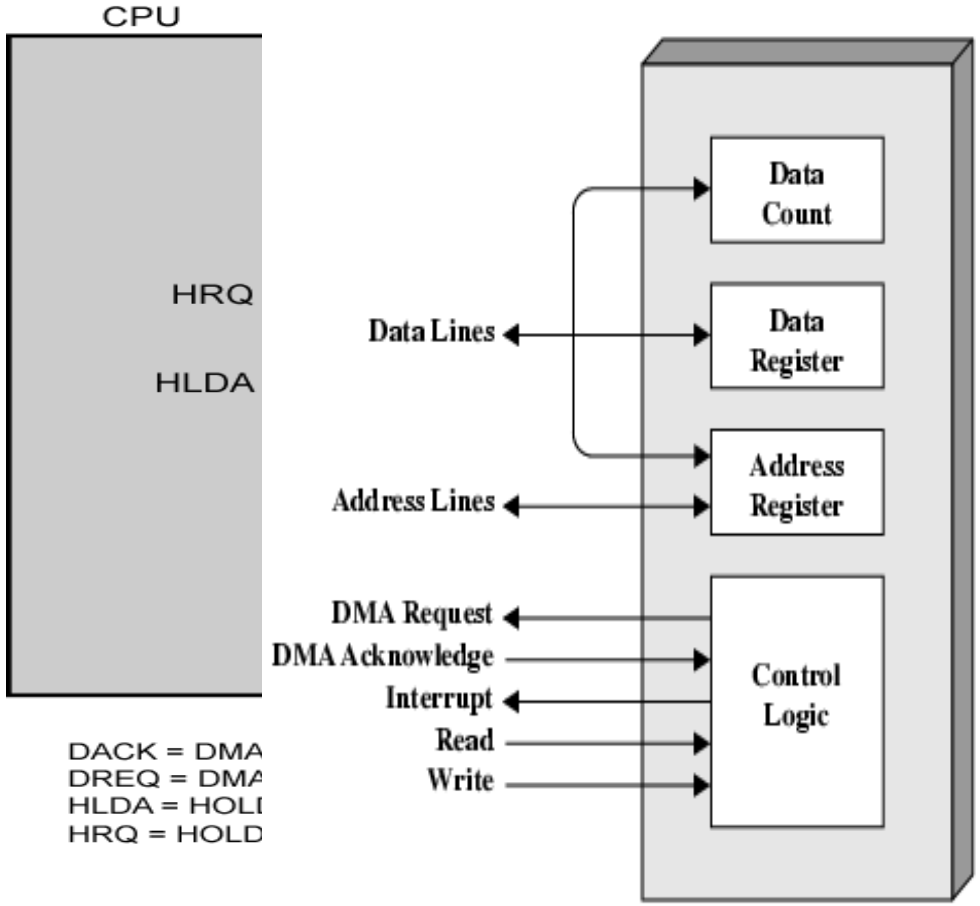
DMA and Interrupt Breakpoints During an Instruction Cycle

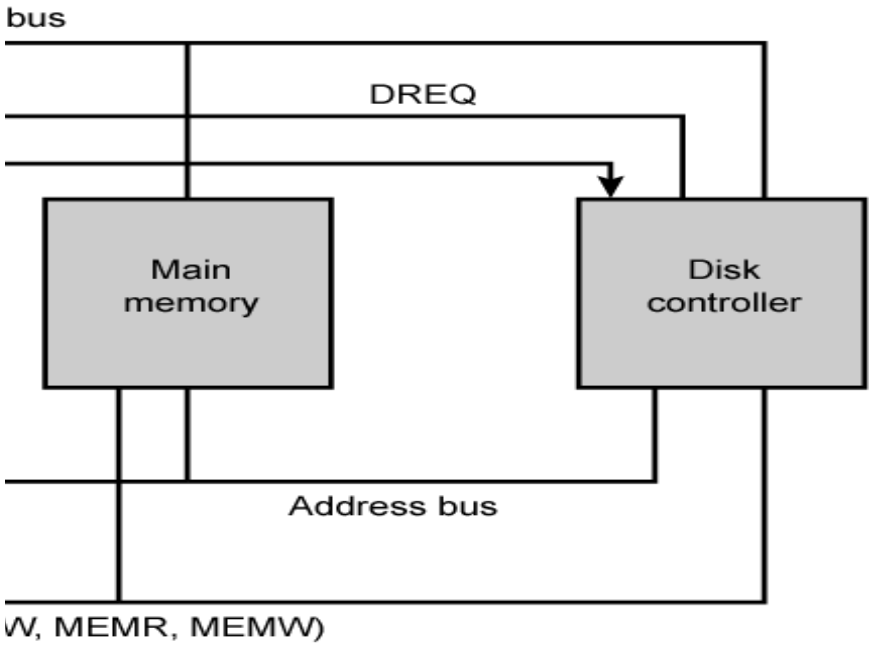


Intel 8237A DMA Controller

- 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





mand/ control registers:
ode, Single-mask and All-mask

Flow through and Fly-By

- DMA controller:
- Flow-through (Explicit) DMAC: The data, transferred between memory and I/O interface pass through the DMAC. The DMAC first reads data into an internal register and than writes it to the destination.

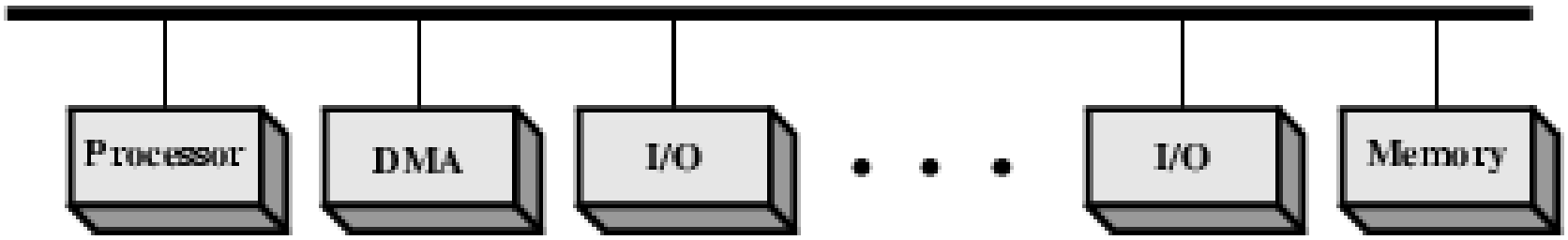
- Fly-by (Implicit) DMAC: The data don't pass through the DMAC.

- Data is not stored in DMA chip
- DMA only between I/O port and memory.
  - Not between two I/O ports
  - Not between two memory locations

- 8237 contains four DMA channels
  - Programmed independently
  - Any one active

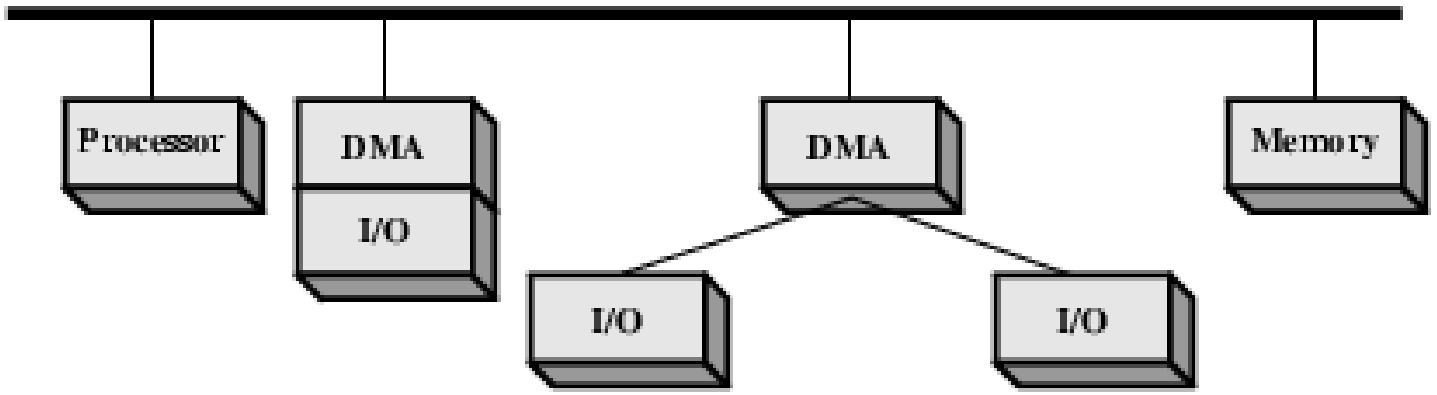
  - Memory to memory transfer may be performed through temp. Reg in DMA channel

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

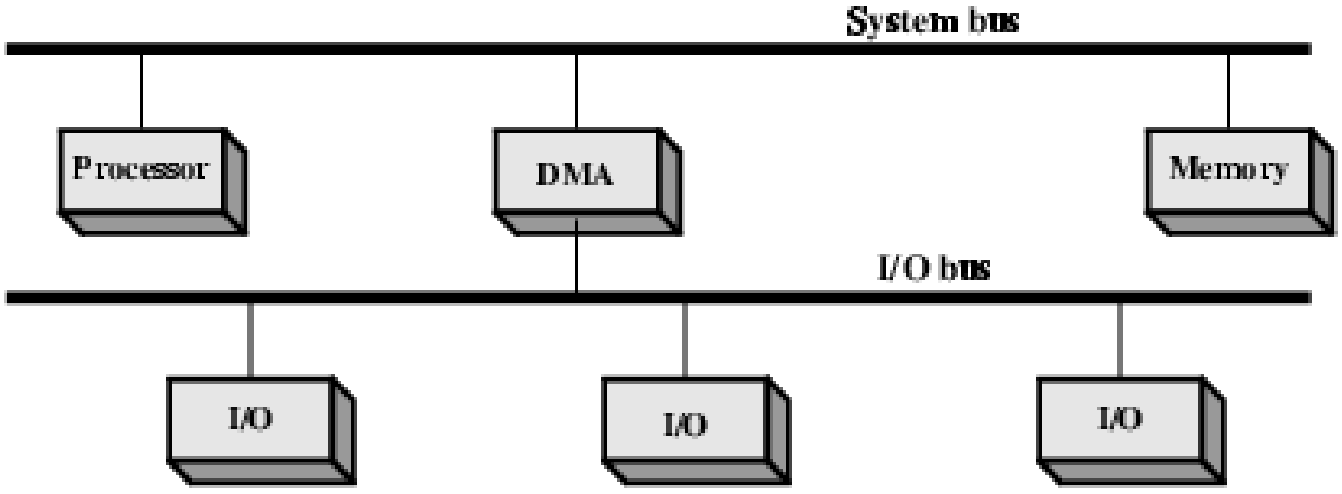
DMA Configurations (2)



(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

Evolution of I/O functions

1. CPU directly controls peripheral device
2. +I/O module: CPU uses programmed I/O
3. +I/O module: CPU will be interrupted
4. +DMA: I/O module access memory through DMA
5. +I/O Channel: I/O module is enhanced to become a processor with specialized instruction set for I/O.
6. +I/O Processor: I/O module has a local memory of its own, a computer in its own right

CPU directs I/O processor to execute I/O program in memory

- Thanks