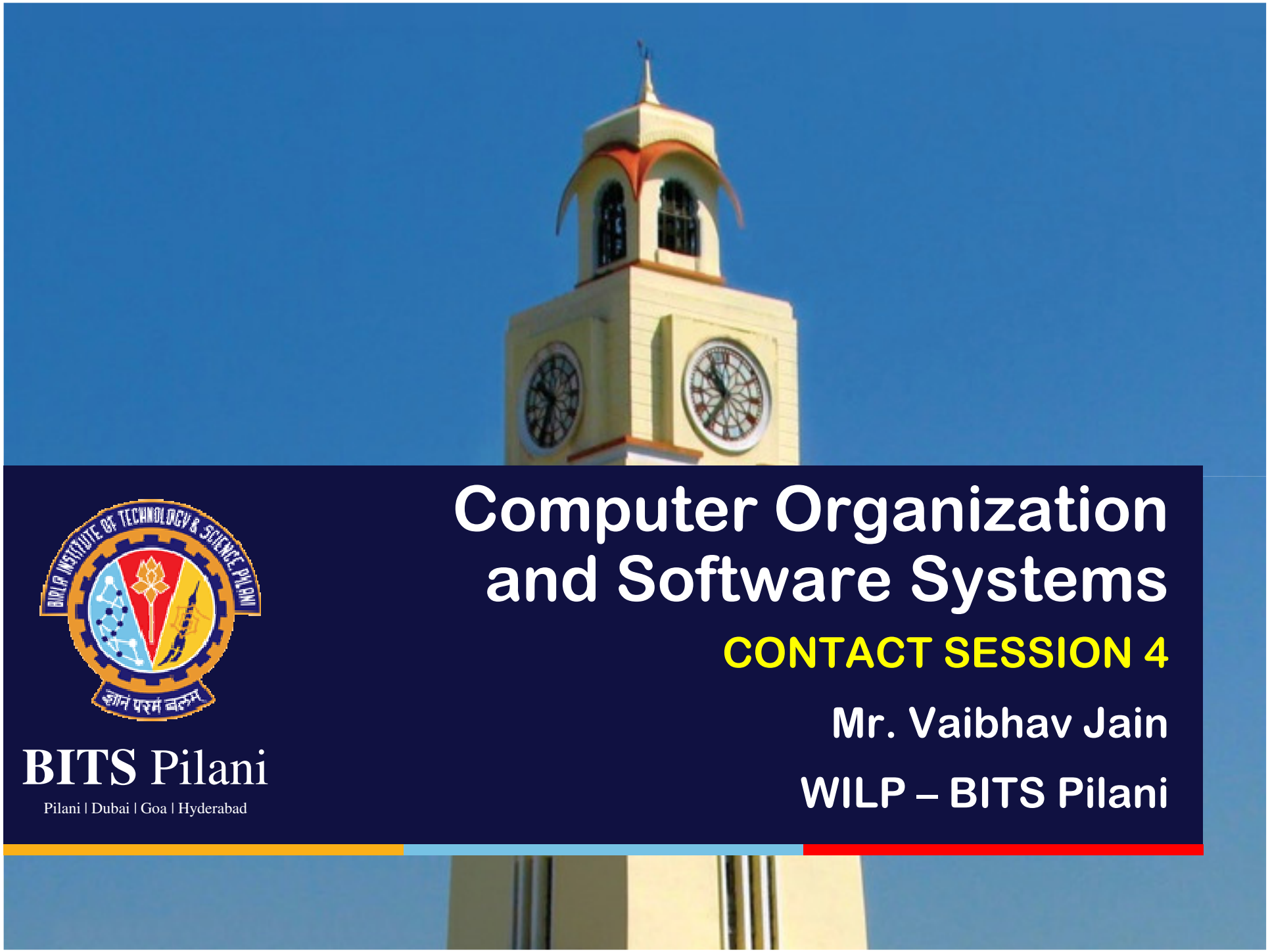


Computer Organization and Software Systems

CONTACT SESSION 4

Mr. Vaibhav Jain

WILP – BITS Pilani

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Computer Organization and Software Systems (SS ZG516)

Session _4

- Cache Performance (3 C's)
- Average Memory Access Time
- **Cache Optimization Techniques**
- **Input/Output Organization :** I/O Modules
- **Data Transfer Schemes** : Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access

Cache Performance

- More the processor references found in the cache better the performance
  - If a reference found in the cache it is HIT otherwise MISS
  - A penalty is associated with each MISS occurred (Miss Penalty)
  - More hits reduces average access time

Where to misses come from?

- **Compulsory**— Initially cache is empty or no valid data in it. So the first access to a block is always miss. Also called **cold start misses** or **first reference misses**.

- **Capacity**—If the cache size is not enough such that it can not accommodate sufficient blocks needed during execution of a program then frequent misses will occur. Capacity misses are those misses that occur regardless of associativity or block size.

- **Conflict**—If block-replacement strategy is set associative or direct mapped, conflict misses (in addition to compulsory & capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. Also called **collision misses**.

Average Memory Access Time

- Hit Ratio(H)
  - Ratio of number of references found in the higher level memory (M1) (i.e. cache memory) to the total references

- Average Access time
  - $T_s = H*T_1 + (1-H)*(T_1+T_2)$
  - T_1=Access time of M_1 (Cache)
  - T_2=Access time of M_2 (Main Memory)

Question – Average Memory Access Time (AMAT)

- Suppose a 1024-byte cache has an access time of 0.1 microseconds and the main memory stores 1 Mbytes with an access time of 1 microsecond. A referenced memory block that is not in the cache must be loaded into the cache and the reference started again.
- Answer the following questions:

  i. What is the number of bits needed to address the main memory?

  ii. If the cache hit ratio is 95%, what is the average access time for a memory reference?

i. 20 bits

ii. Avg access time = hit ratio * cache access +
(1- hit ratio) * (cache access + memory access)
= .95 * 0.1 microsec + .05 * (1 + 0.1) microsec
= .095 + .055 microsec
= 0.15 microseconds

Hit ratio Calculation Example

- Cache access time is 80 ns
- Main memory access time is 1000 ns
- Average access time is 180 ns
- What is Hit ratio?

Cache Optimization

- Reduces Average Memory Access Time
- AMAT = $H*T_1 + (1-H)*(T_1+T_2)$
- AMAT = Hit Time + Miss Rate x Miss Penalty
- Therefore, to optimize cache
  - Reducing Miss Rate
  - Reducing the miss penalty
  - Reducing Hit Time

Larger Block Size

- Larger Block size to reduce miss rate
- Advantages
  - Utilize locality
  - Reduces compulsory misses
- Disadvantages
  - Increases miss penalty
  - More time to fetch a block to the cache [bus width issue]
  - Increases conflict misses
  - More number of blocks will be mapped to the same location

Larger Caches

- Larger cache to reduce miss rate
- Advantages
  - Reduces capacity misses
  - Can accommodate larger memory footprint
- Disadvantages
  - Longer hit time
  - Higher cost, area and power

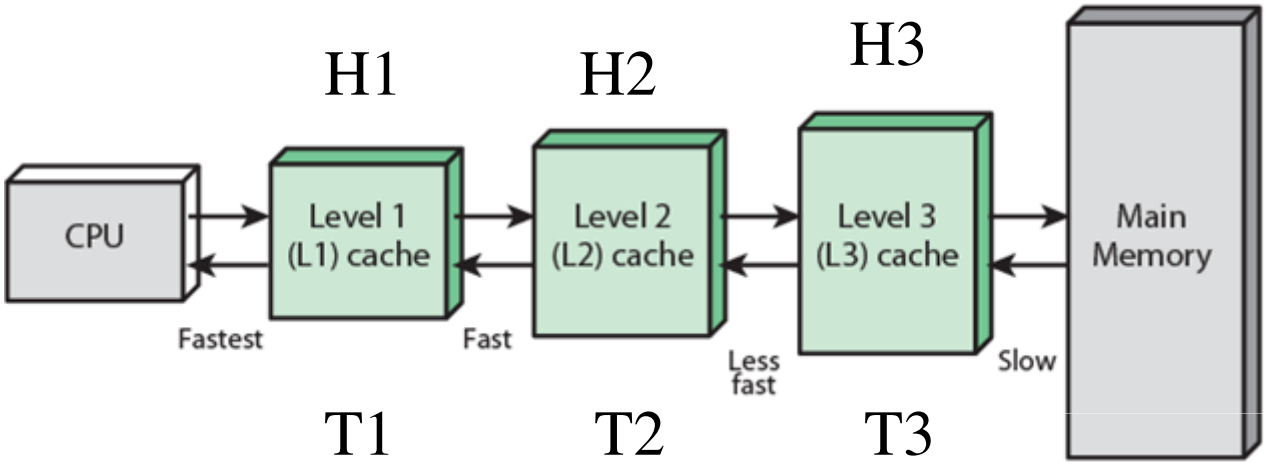
Higher Associativity

- Higher associativity to reduce miss rate
  - Fully associative caches are the best, but high hit time
  - So increases the associativity to an optimal possible level
- Advantages
  - Reduces conflict misses
  - Reduce miss rate
- Disadvantages
  - Increase in Hit time
  - Complex design than direct mapped

Multilevel Caches

- Multilevel caches to reduce miss rate
  - Performance gap between processors and memory
  - Caches should be faster to keep pace with the speed of processors, and cache should be larger to overcome the widening gap between the processor and main memory
  - Add another level of cache between the cache and memory
  - The first level (L1) can be small enough to match the clock cycle time to the fast processor. [Low Hit time]
  - The second level (L2) can be large enough to capture many accesses that would go to main memory, there by lessening the effective miss penalty. [Low Miss Rate]

Multilevel Caches



AMAT = Hit Time(L1) + Miss Time (L1) * Miss Penalty (L1)

Miss Penalty(L1) = Hit Time(L2) + Miss Time(L2) * Miss Penalty(L2)

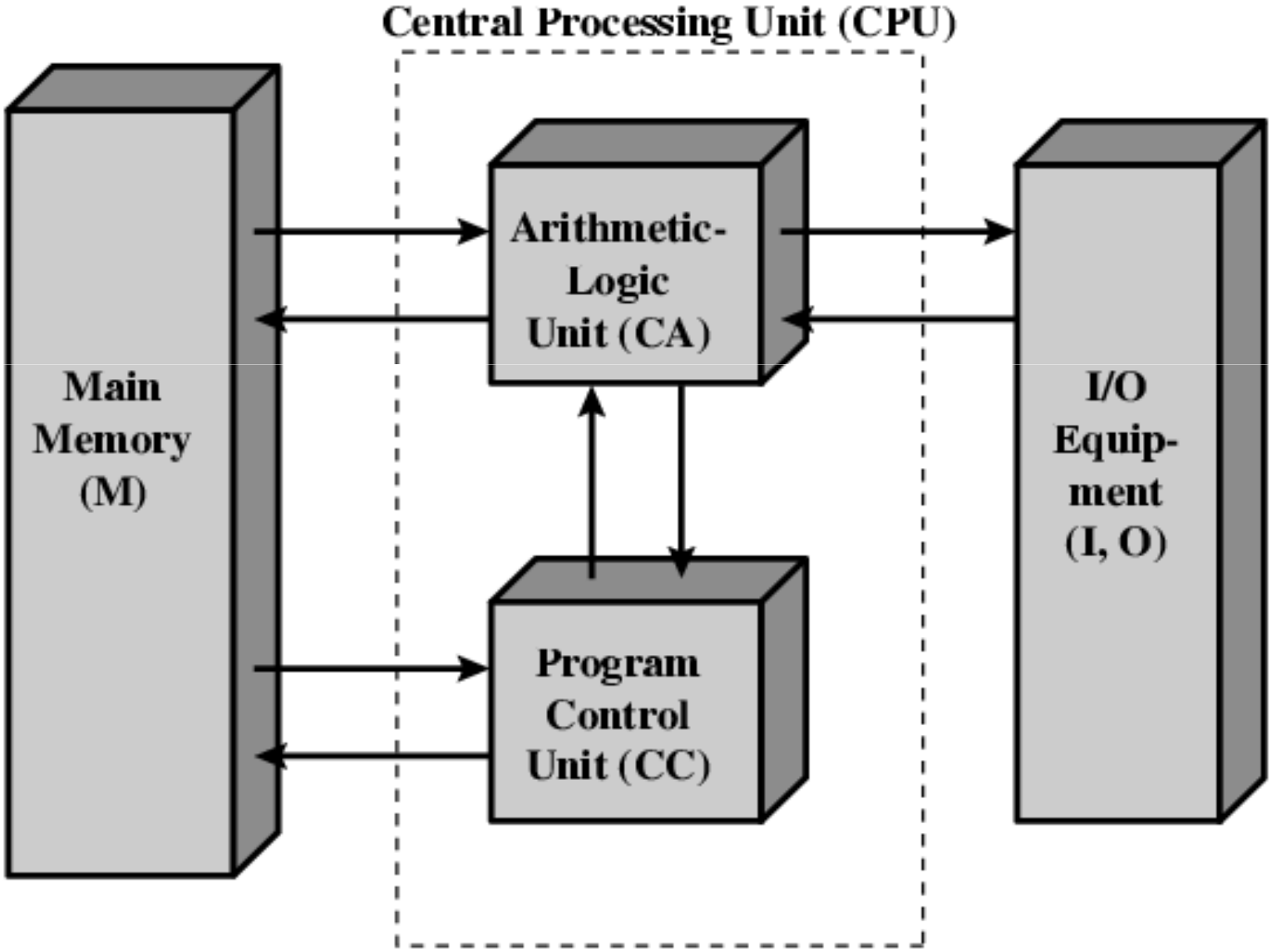
Therefore,

AMAT = Hit Time(L1) + Miss Time (L1) *
(Hit Time(L2) + Miss Time(L2) * Miss Penalty(L2))

Multilevel Caches

- Multilevel caches to reduce miss rate
- Local Miss Rate
  - Number of misses in a cache level divided by number of memory access to this level.
- Global Miss Rate
  - Number of misses in a cache level divided by number of memory access generated by the CPU.

Structure of von Neumann machine (IAS Computer)



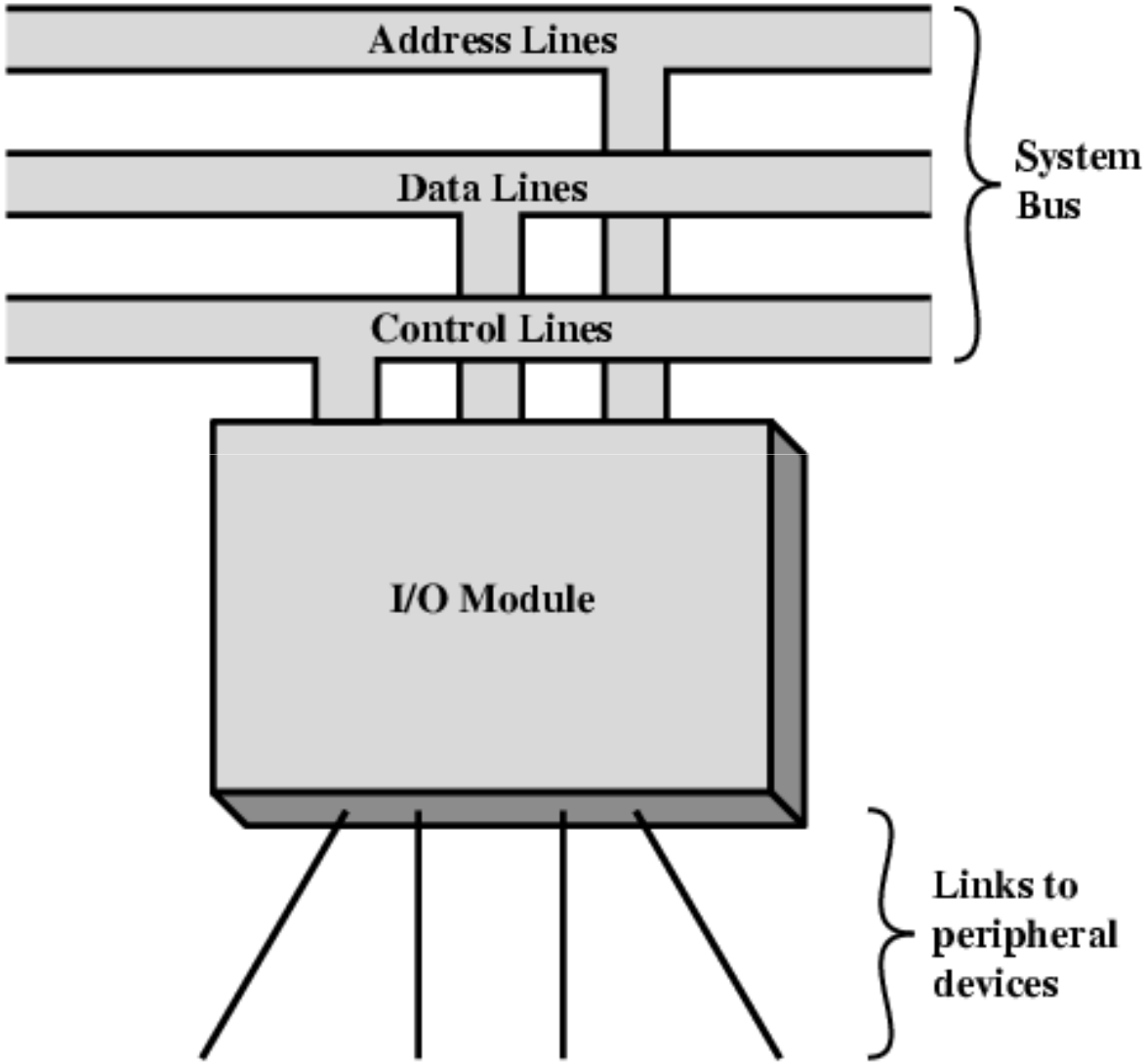
Input/Output Problems

- 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

Input/Output Module

- Interface to CPU and Memory
- Interface to one or more peripherals

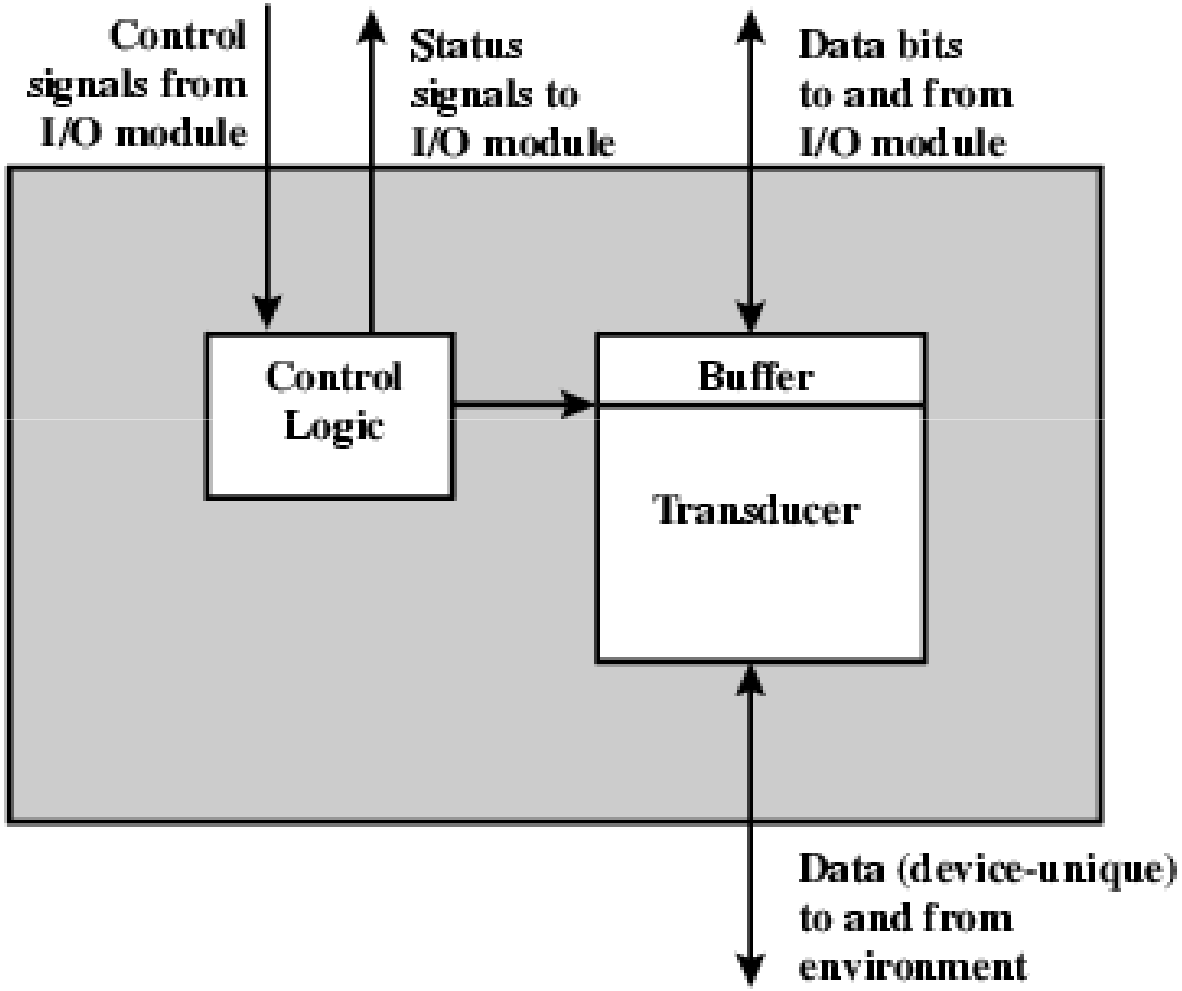
Generic Model of I/O Module



External Devices

- Human readable
  - Suitable for communicating with the computer user
  - Screen, printer, keyboard
- Machine readable
  - Suitable for communicating with equipment
  - Monitoring and control (like sensors)
- Communication
  - Suitable for communicating with remote devices
  - Modem
  - Network Interface Card (NIC)

External Device Block Diagram



External Device Block Diagram

- The interface to the I/O module is in the form of control, data, and status signals.

- Control signals determine the function that the device will perform, such as
  - send data to the I/O module (INPUT or READ)
  - accept data from the I/O module (OUTPUT or WRITE)

External Device Block Diagram

- Data are in the form of a set of bits to be sent to or received from the I/O module.

- Status signals indicate the state of the device.
  - Examples are READY/NOT-READY to show whether the device is ready for data transfer

External Device Block Diagram

- Control logic associated with the device controls the device's operation in response to direction from the I/O module.
- The transducer converts data from electrical to other forms of energy during output and from other forms to electrical during input.
- Typically, a buffer is associated with the transducer to temporarily hold data being transferred between the I/O module and the external environment.
- A buffer size of 8 to 16 bits is common

I/O Module Function

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

I/O Module Function – Control & Timing

- The internal resources, such as main memory and the system bus, must be shared among a number of activities, including data I/O.
- Thus, a control and timing required to coordinate the flow of traffic between internal resources and external devices.
- For example, the control of the transfer of data from an external device to the processor might involve the following sequence of steps:

I/O Module Function – Control & Timing

- 1. The processor interrogates the I/O module to check the status of the attached device.
- 2. The I/O module returns the device status.
- 3. If the device is operational and ready to transmit, the processor requests the transfer of data, by means of a command to the I/O module.
- 4. The I/O module obtains a unit of data (e.g., 8 or 16 bits) from the external device.
- 5. The data are transferred from the I/O module to the processor

I/O Module Function – Processor Communication

- Command decoding
  - The I/O module accepts commands from the processor typically sent as signals on the control bus.
- Data
  - Data are exchanged between the processor and the I/O module over the data bus.
- Status reporting
  - Because peripherals are so slow, it is important to know the status of the I/O module. Common status signals are BUSY and READY.

I/O Module Function – Processor Communication

- Address recognition
  - Just as each word of memory has an address, so does each I/O device. Thus an I/O module must recognize one unique address for each peripheral it controls.

I/O Module Function – Device communication

- This communication involves
  - Commands
  - Status information
  - Data Transfer between external device and I/O Module.

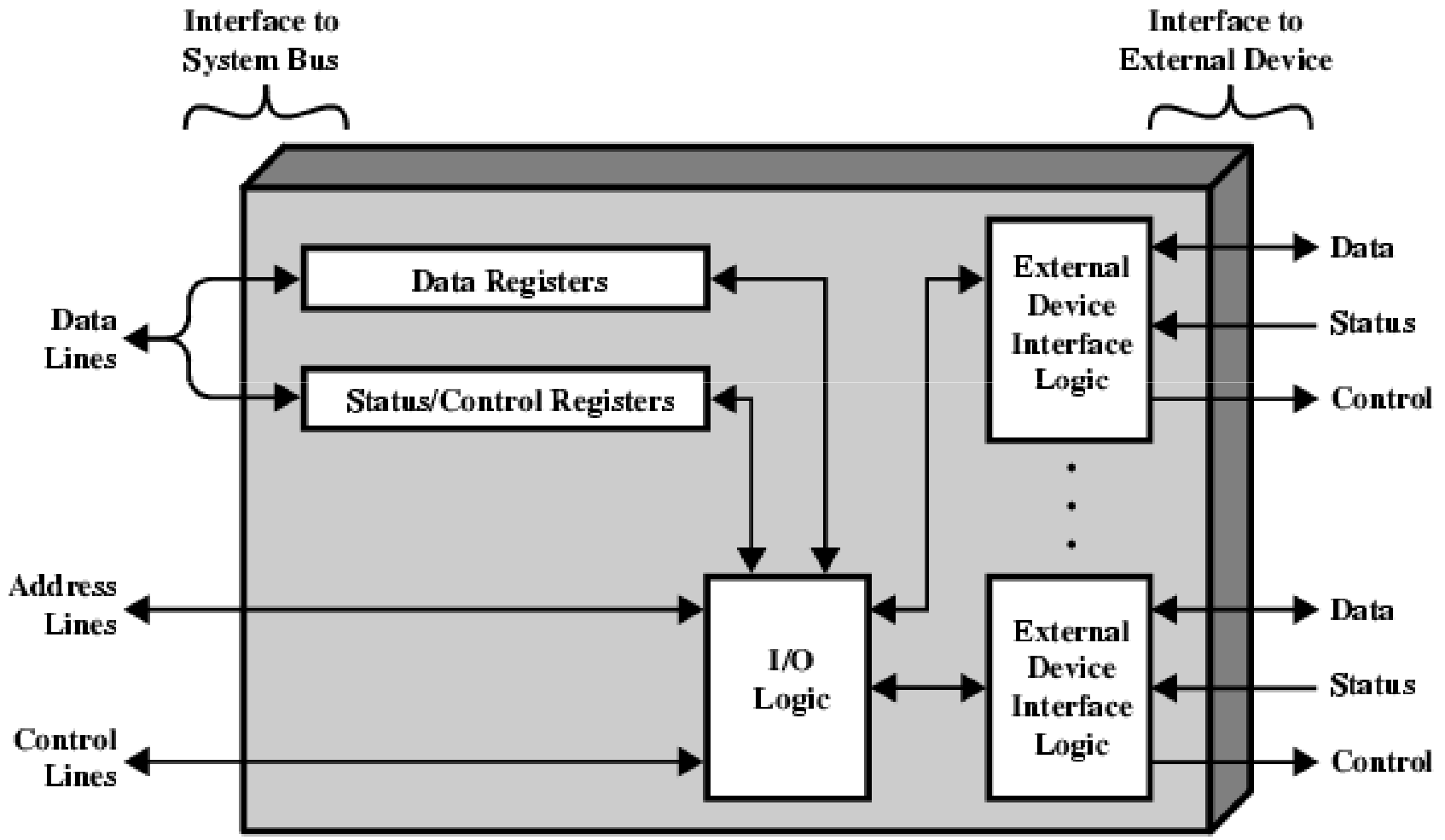
I/O Module Function – Data Buffering

- Suppose the transfer rate into and out of main memory or the processor is quite high, where the external peripheral is slow.
- Therefore, the data are buffered in the I/O module and then sent to the peripheral device at its data rate.
- In the opposite direction, data are buffered so as not to tie up the memory in a slow transfer operation.
- Similarly, if the I/O device operates at a rate higher than the memory access rate, then the I/O module performs the needed buffering operation.

I/O Module Function – Error Detection

- One class of errors includes mechanical and electrical malfunctions reported by the device (e.g., paper jam, bad disk track).
- Another class consists of unintentional changes to the bit pattern as it is transmitted from device to I/O module.
- Some form of error-detecting code is often used to detect transmission errors. Example-Parity.
- In this additional bit is transferred and set so that the total number of 1s in the byte is even (even parity) or odd (odd parity).
- When a byte is received, the I/O module checks the parity to determine whether an error has occurred.

I/O Module Structure



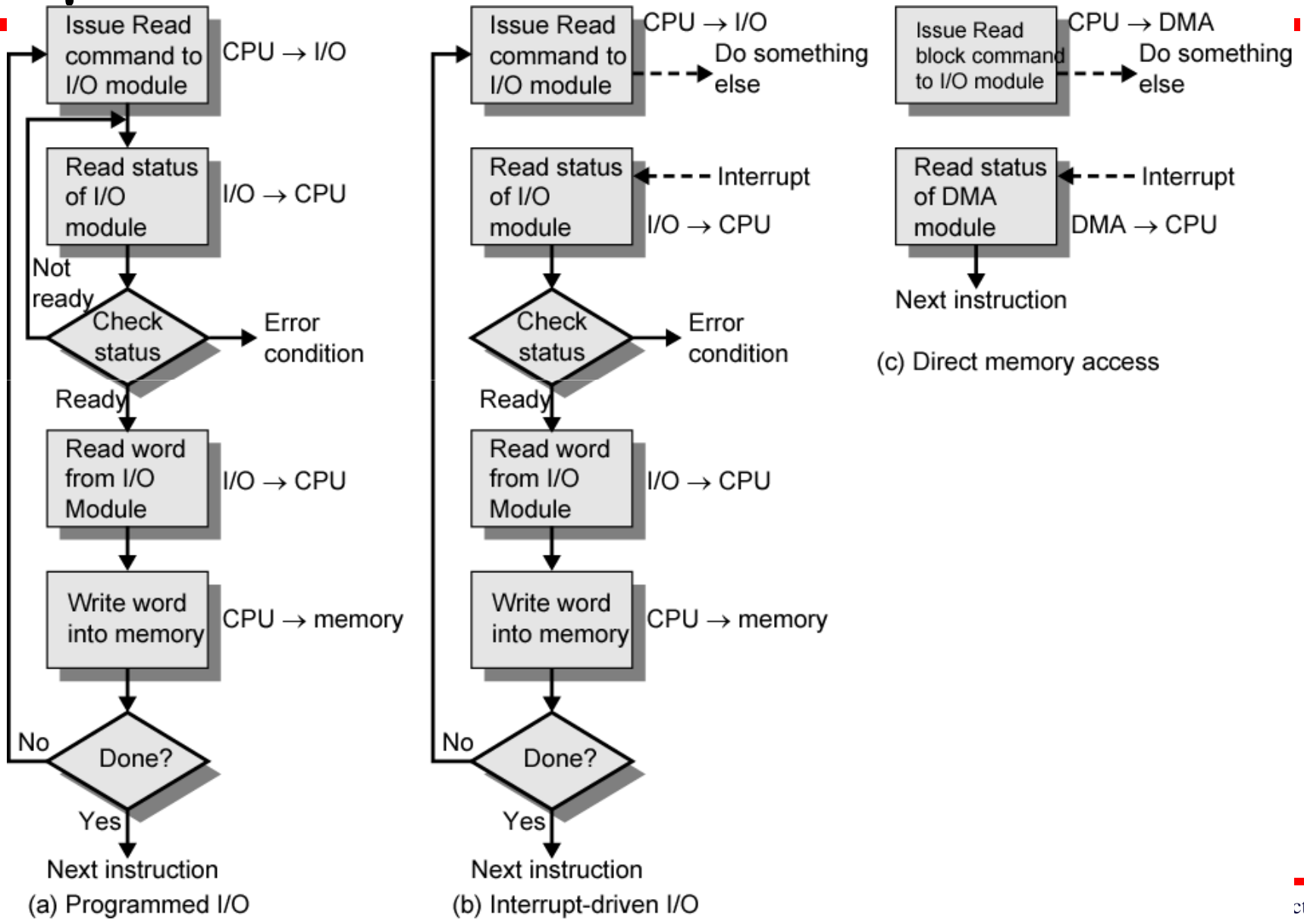
I/O Module Structure

- Data register used to buffer data transferred to and from the module.
- Status registers provide current status information and also function as a control register, to accept detailed control information from the processor.
- The processor uses the control lines to issue commands to the I/O module.
- The module must also be able to recognize and generate addresses associated with the devices it controls. Thus, the I/O module contains logic specific to the interface with each device that it controls.

Input Output Techniques

- **Programmed I/O**
  - The processor executes a program (including sensing device status, sending a read or write command, and transferring the data) that gives it direct control of the I/O operation.
  - When the processor issues a command to the I/O module, it must wait until the I/O operation is complete. If the processor is faster than the I/O module, this is wasteful of processor time.
- **Interrupt driven I/O**
  - Processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.
- **In Both Processor is responsible for extracting data from main memory for output and storing data in main memory for input.**

Three Techniques for Input of a Block of Data



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

Interrupt Driven I/O
Basic Operation

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

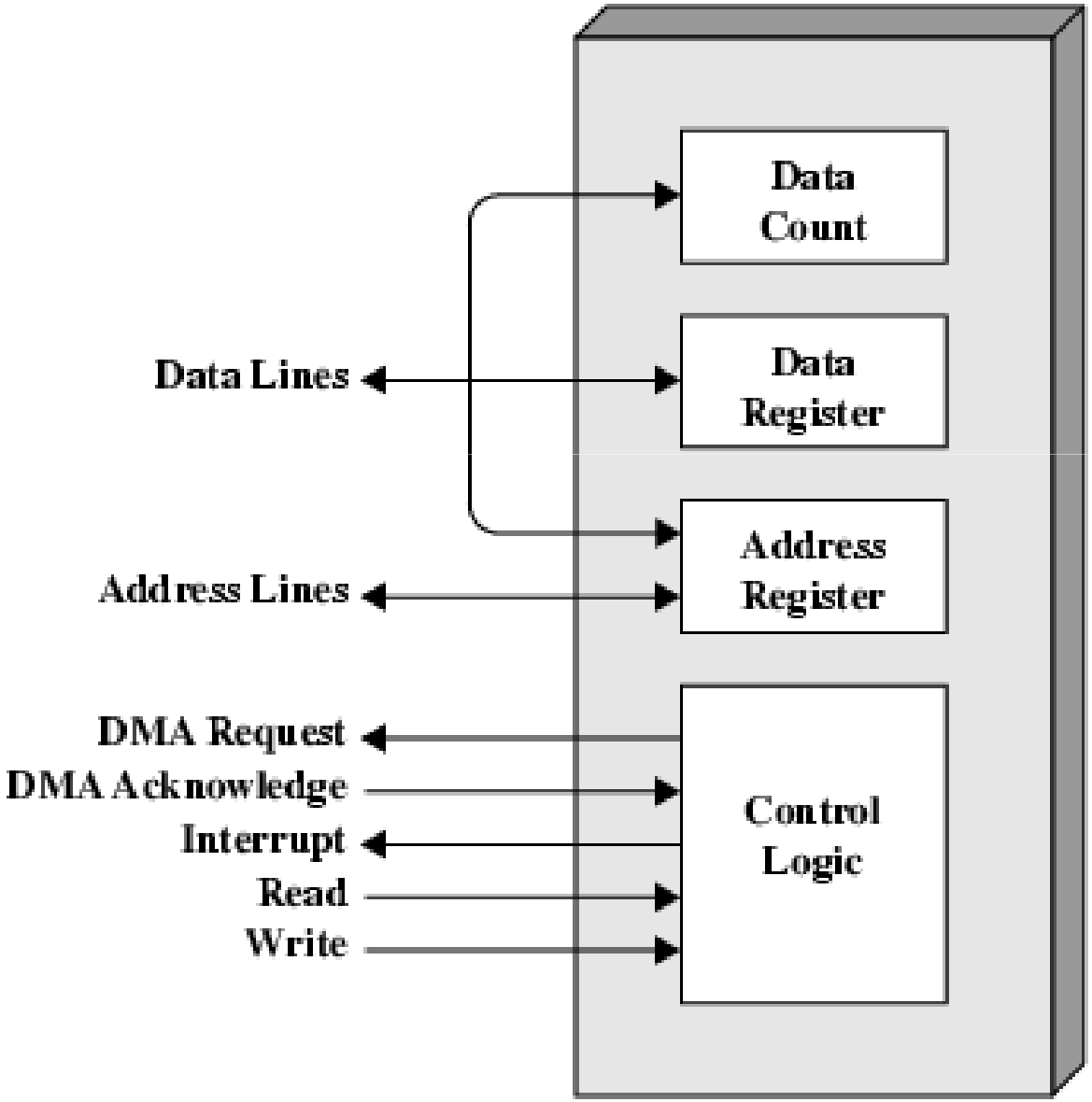
Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
  - The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
  - For large volumes of data transfer, DMA is the answer

DMA Function

- Additional Module (hardware) on bus
- DMA controller takes over control of the system bus from the processor.
- For this, the DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.
- The latter technique is more common and is referred to as *cycle stealing*, because the DMA module in effect steals a bus cycle.

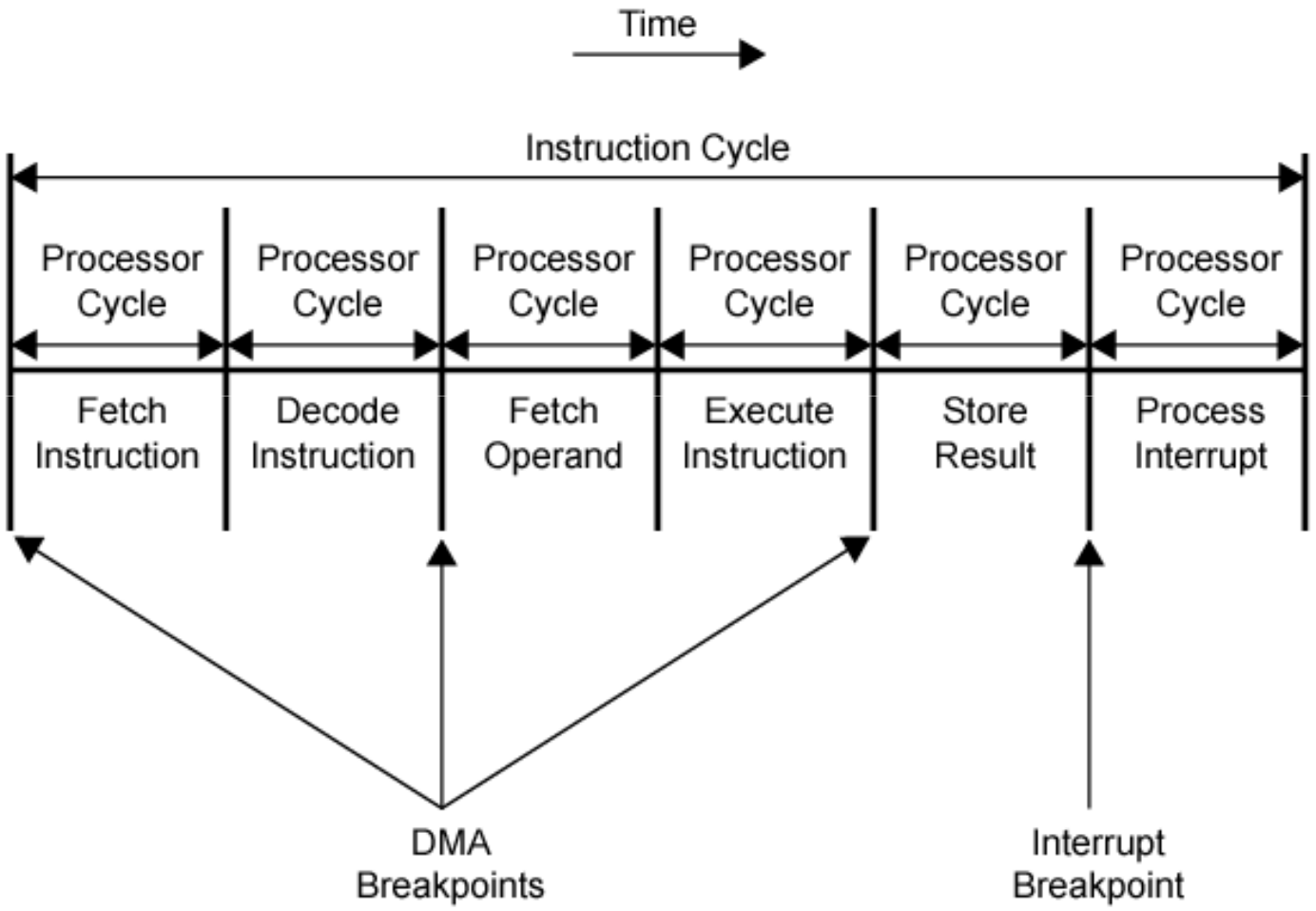
Typical DMA Module Diagram



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

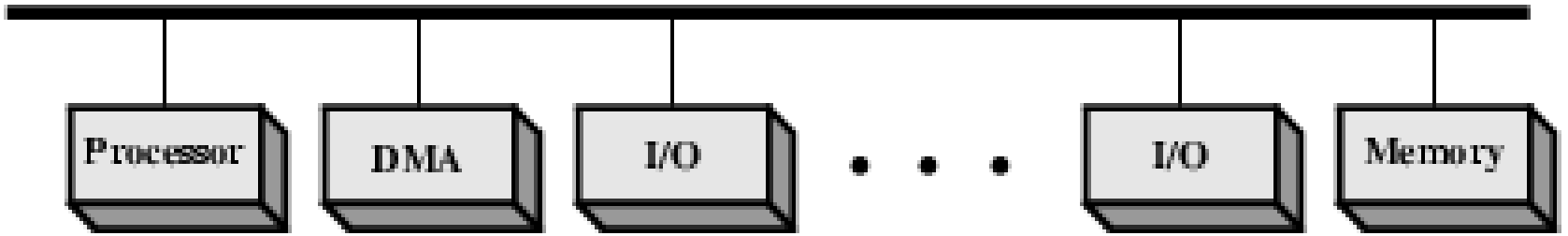
DMA and Interrupt Breakpoints an Instruction Cycle



DMA Transfer
Cycle Stealing

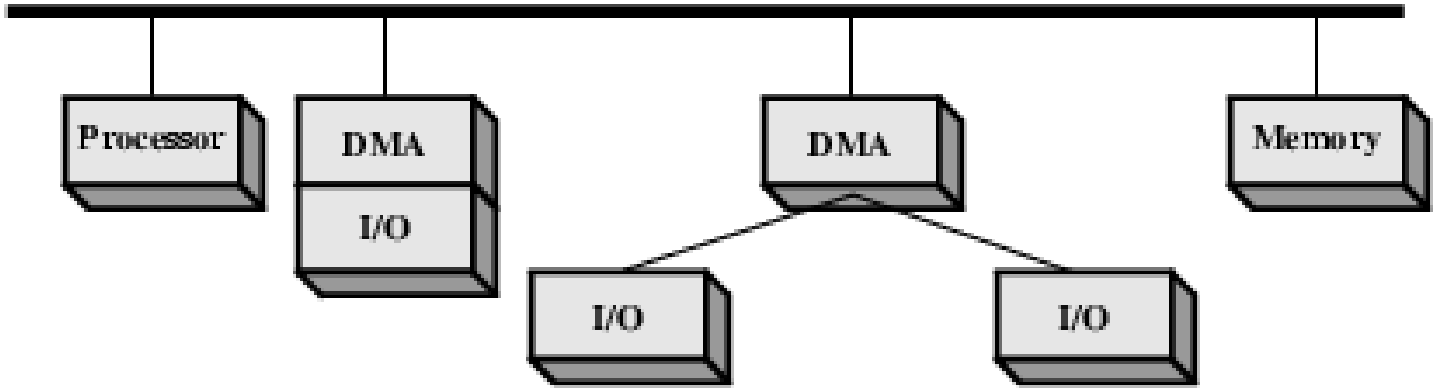
- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write

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

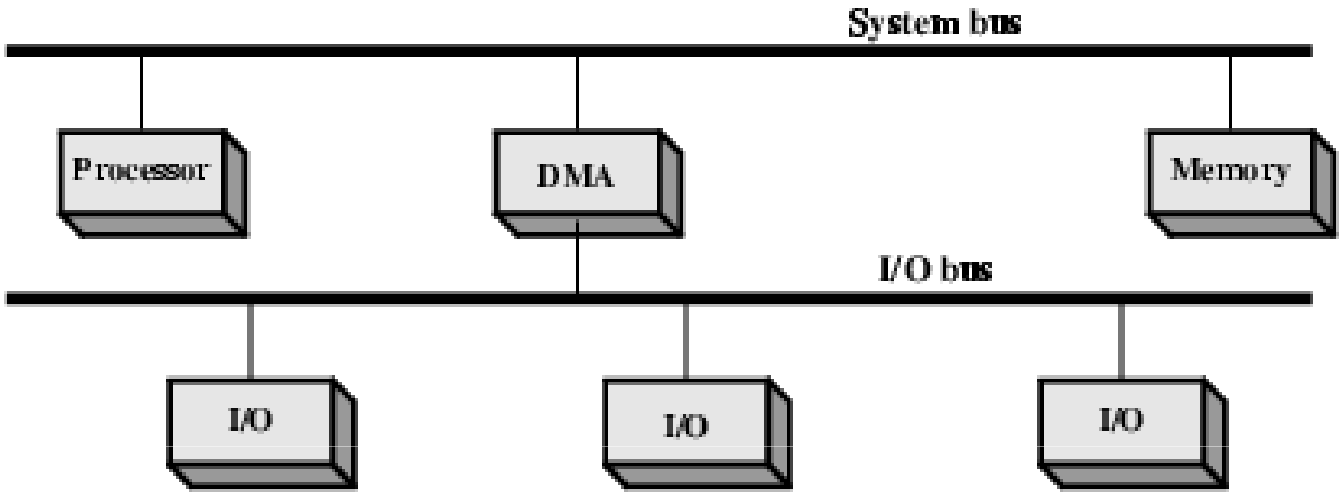
DMA Configurations (2)



(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

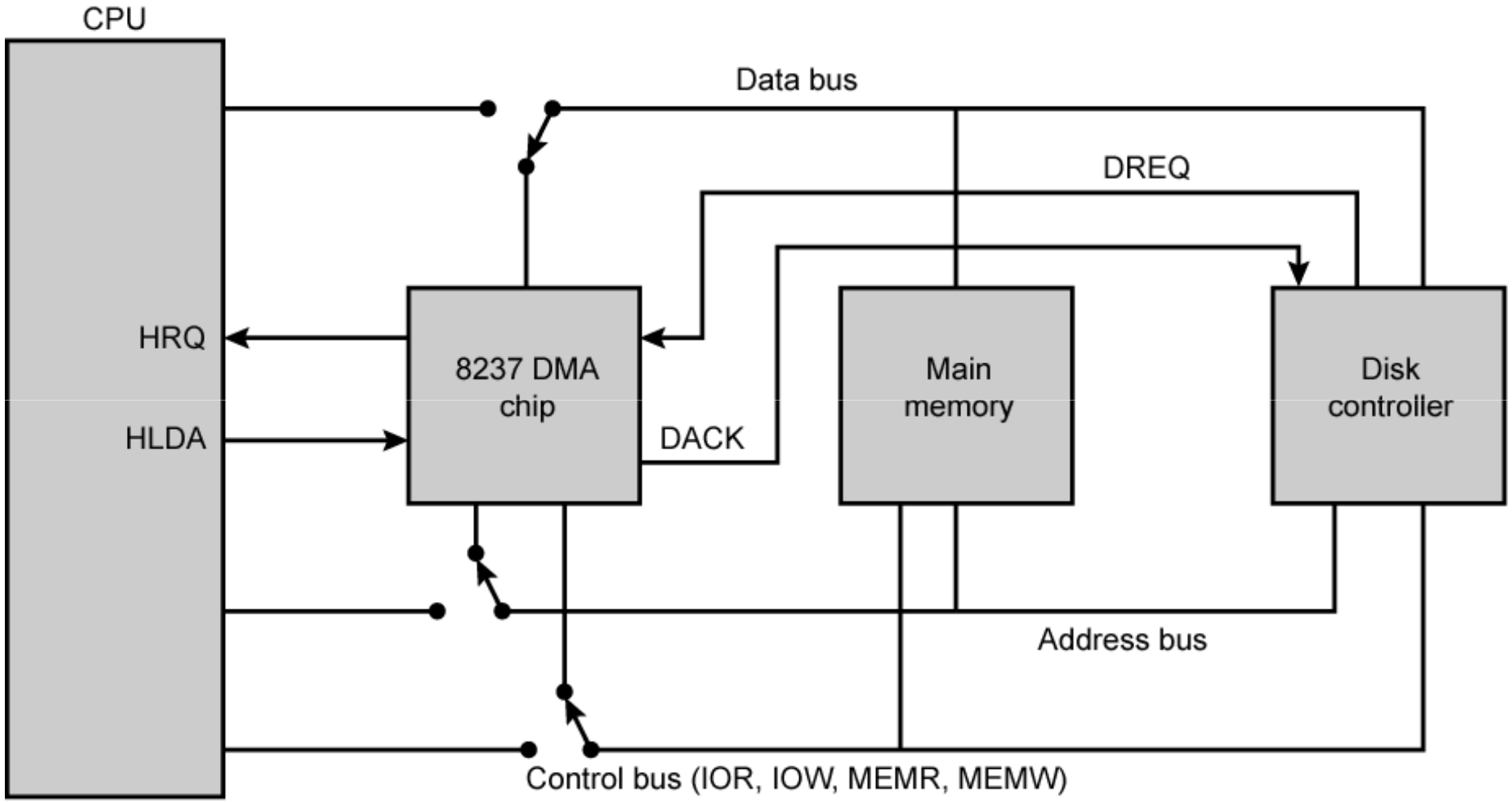
DMA Configurations (3)



(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

8237 DMA Usage of Systems Bus



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

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