ज्ञानेन प्रकाशते जगत्
INDUS UNIVERSITY

ELECTRONICS & COMMUNICATION ENGINEERING
IITE Ahmedabad

# MICROPROCESSORS

(Subject Code-CE0506/CS0506)

Unit-4

B.Tech (CE/CSE)

Semester-V

**Shikha Singh**

Academic Year 2019-2020

# 8237DMA CONTROLLER

# Introduction:

- Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU).
- It is also a fast way of transferring data within (and sometimes between) computer.
- The DMA I/O technique provides direct access to the memory while the microprocessor is temporarily disabled.
- The DMA controller temporarily borrows the address bus, data bus and control bus from the microprocessor and transfers the data directly from the external devices to a series of memory locations (and vice versa).

# Basic DMA Operation:

- Two control signals are used to request and acknowledge a direct memory access (DMA) transfer in the microprocessor-based system.
  - The HOLD signal as an input(to the processor) is used to request a DMA action.
  - The HLDA signal as an output that acknowledges the DMA action.
- When the processor recognizes the hold, it stops its execution and enters hold cycles.

# Cont.,

- HOLD input has higher priority than INTR or NMI.
  - The only microprocessor pin that has a higher priority than a HOLD is the RESET pin.
- HLDA becomes active to indicate that the processor has placed its buses at high-impedance state.

# Basic DMA Definitions:

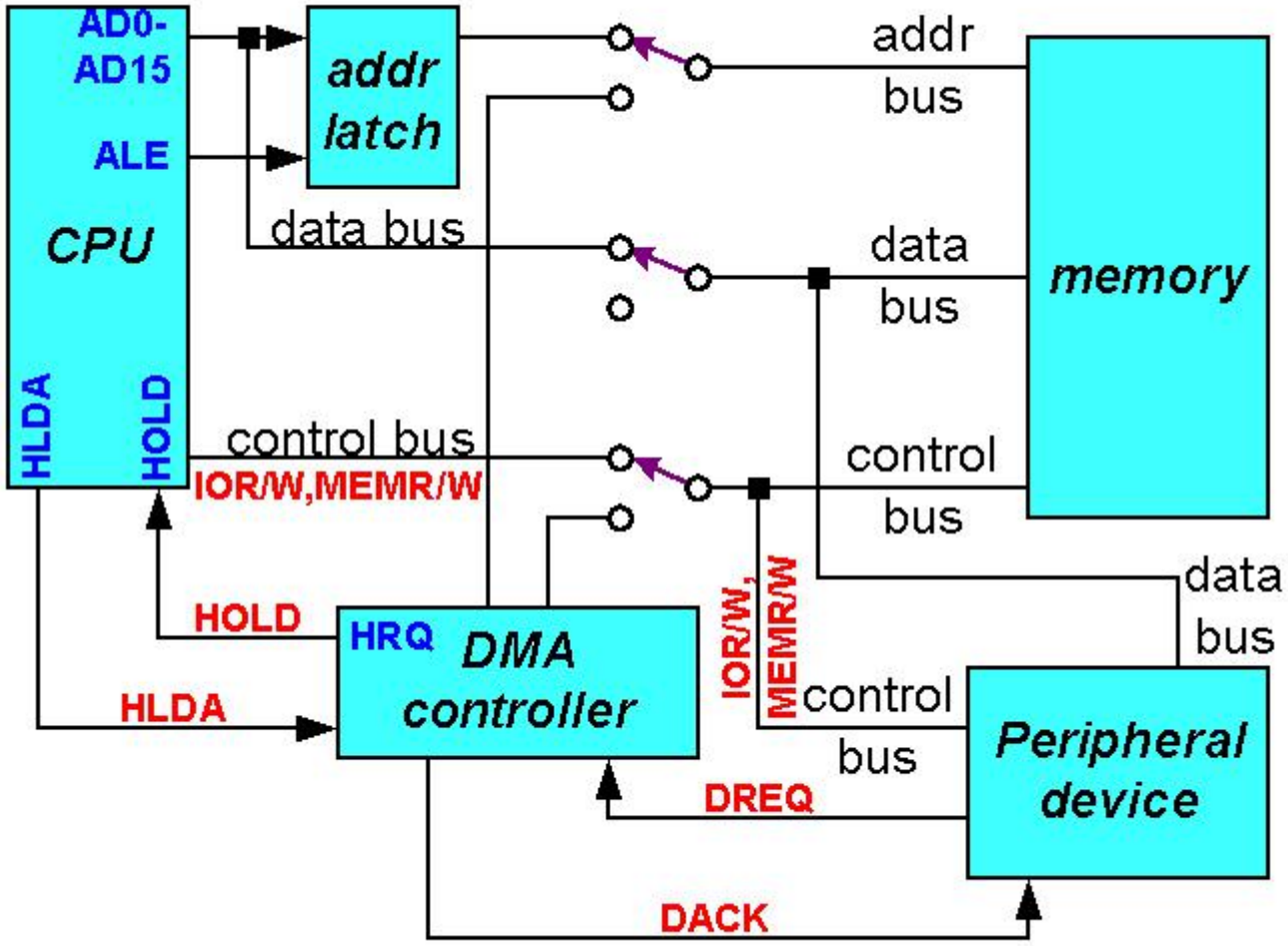
- Direct memory accesses normally occur between an I/O device and memory without the use of the microprocessor.
  - A **DMA read** transfers data from the memory to the I/O device.
  - A **DMA write** transfers data from an I/O device to memory.
- The system contains separate memory and I/O control signals.
- Hence the Memory & the I/O are controlled simultaneously

- The DMA controller provides memory with its address, and the controller signal selects the I/O device during the transfer.
- Data transfer speed is determined by speed of the memory device or a DMA controller.
- In many cases, the DMA controller ***slows*** the speed of the system when transfers occur.
- The serial PCI (Peripheral Component Interface) Express bus transfers data at rates exceeding DMA transfers.
- This in modern systems has made DMA is

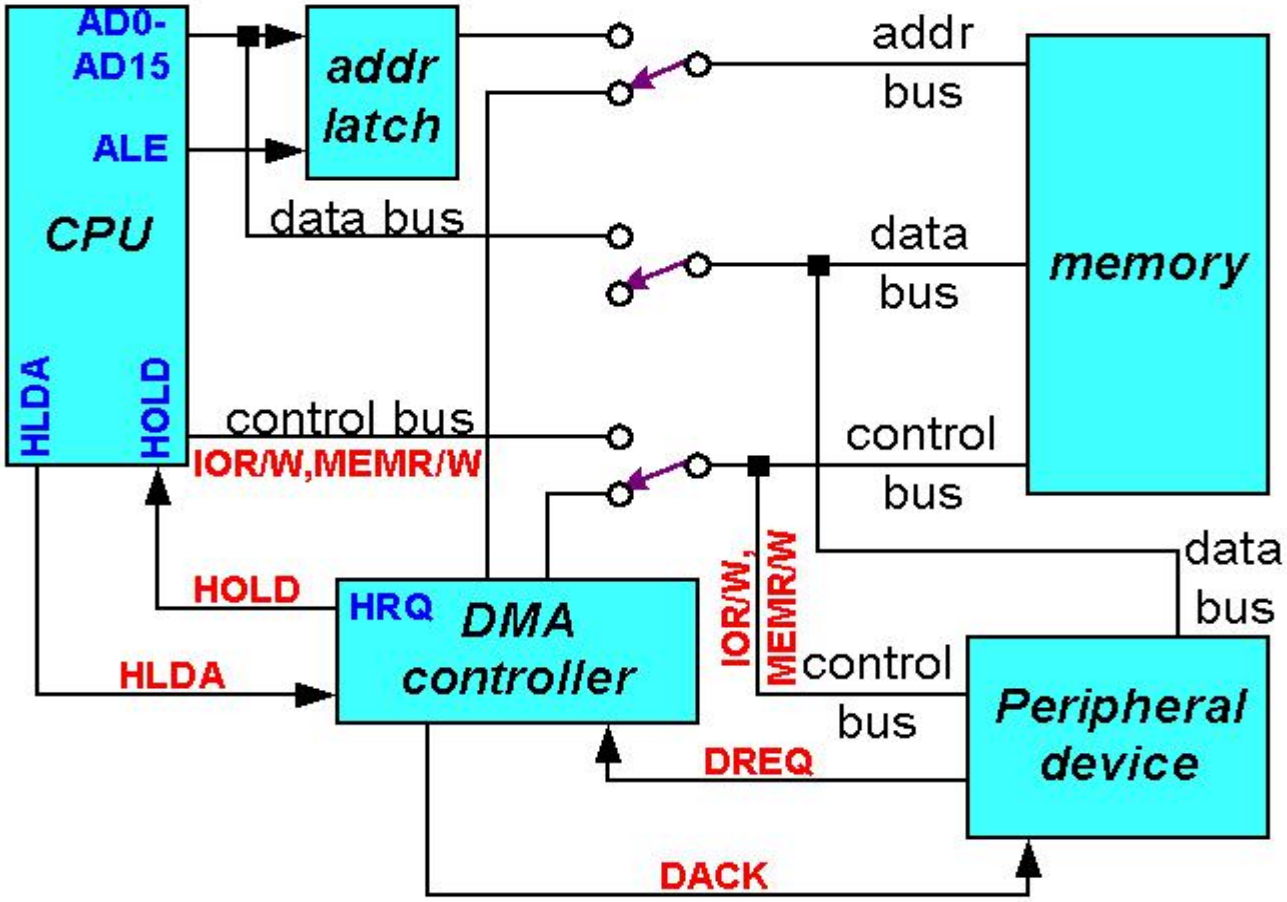
# The 8237 DMA Controller

- The 8237 supplies memory & I/O with control signals and memory address information during the DMA transfer.
- It is actually a special-purpose microprocessor whose job is high-speed data transfer between memory and I/O

# CPU having the control over the bus:

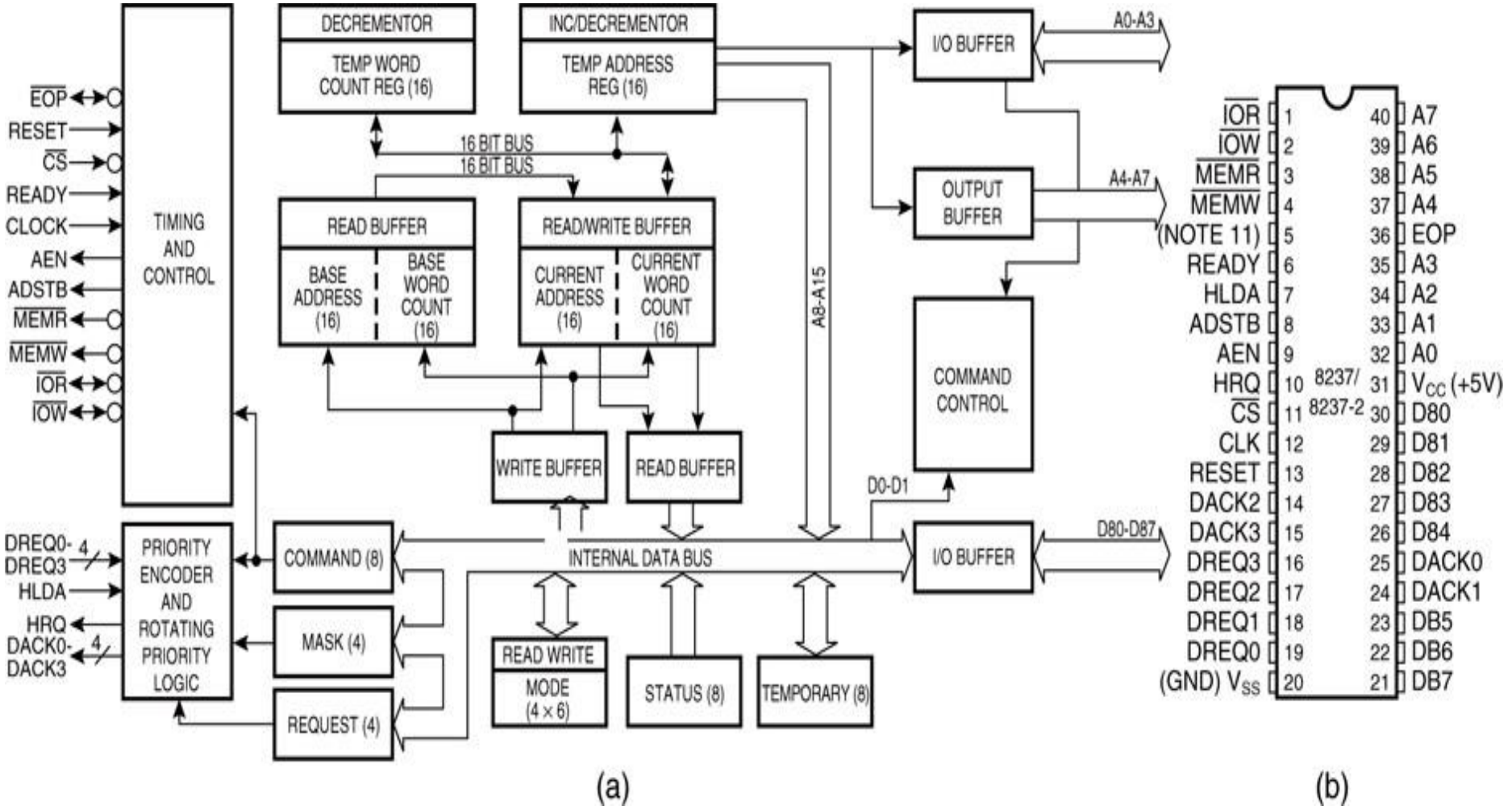


# When DMA operates:



- 8237 is not a discrete component in modern microprocessor-based systems.
  - It appears within many system controller chip sets
- 8237 is a four-channel device compatible with 8086/8088, adequate for small systems.
- Expandable to any number of DMA channel inputs
- 8237 is capable of DMA transfers at rates up to 1.6MB per second.
- Each channel is capable of addressing

Programmable DMA controller. (a) Block diagram and (b) pin-out.



# 8237 Internal Registers

## CAR

- The **current address register** holds a 16-bit memory address used for the DMA transfer.
- Each channel has its own current address register for this purpose.
- When a byte of data is transferred during a DMA operation, CAR is either incremented or decremented depending on how it is programmed.

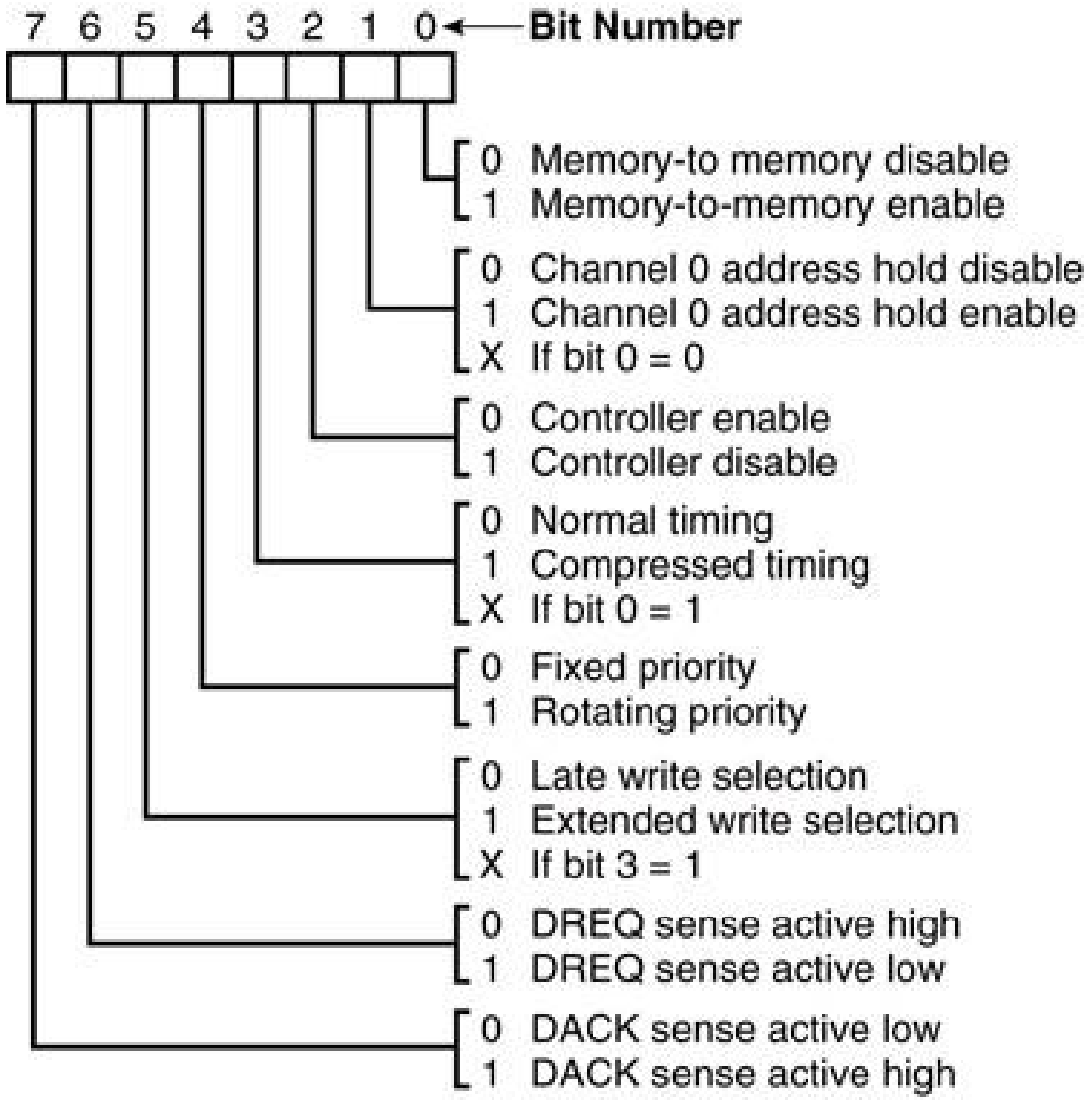
## CWCR

- The **current word count register** programs a channel for the number of bytes to transferred during a DMA action.

## CR

- The **command register** programs the operation of the 8237 DMA controller.
- The register uses bit position 0 to select the memory-to-memory DMA transfer mode.
  - Memory-to-memory DMA transfers use DMA channel 0 to hold the source address

command register.

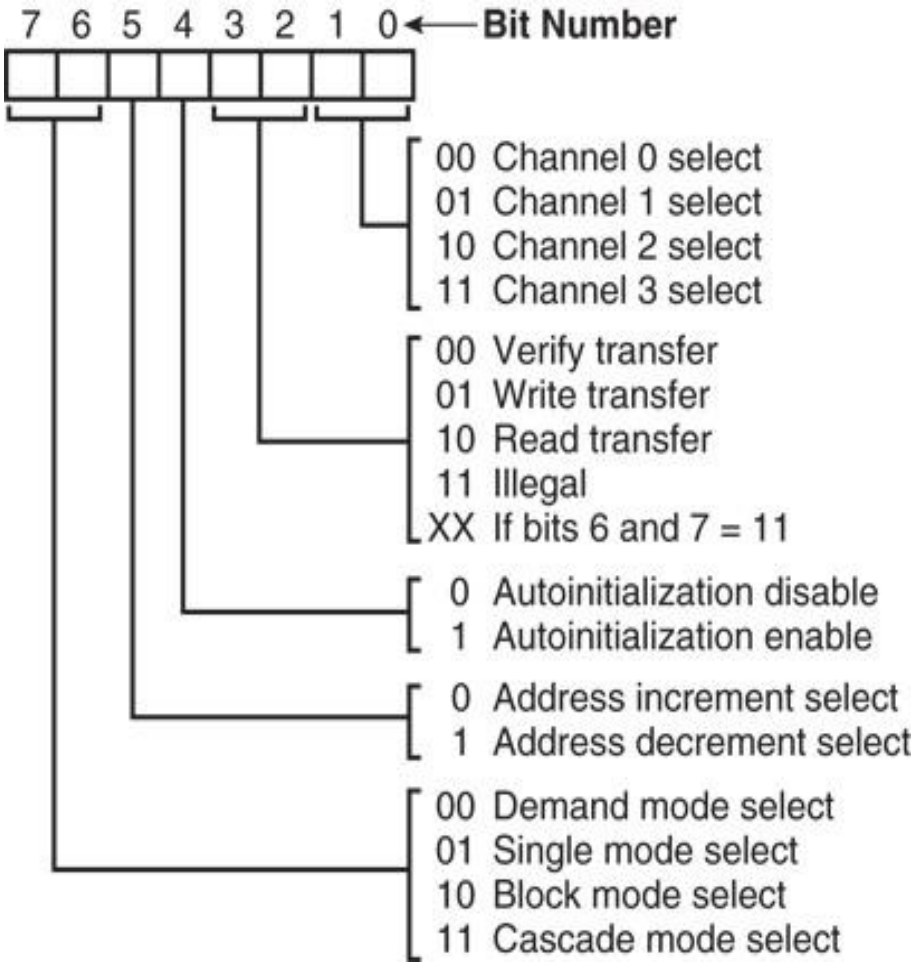


## BA and BWC

- The **base address** (BA) and **base word count** (BWC) registers are used when auto-initialization is selected for a channel.
- In auto-initialization mode, these registers are used to reload the CAR and CWCR after the DMA action is completed.

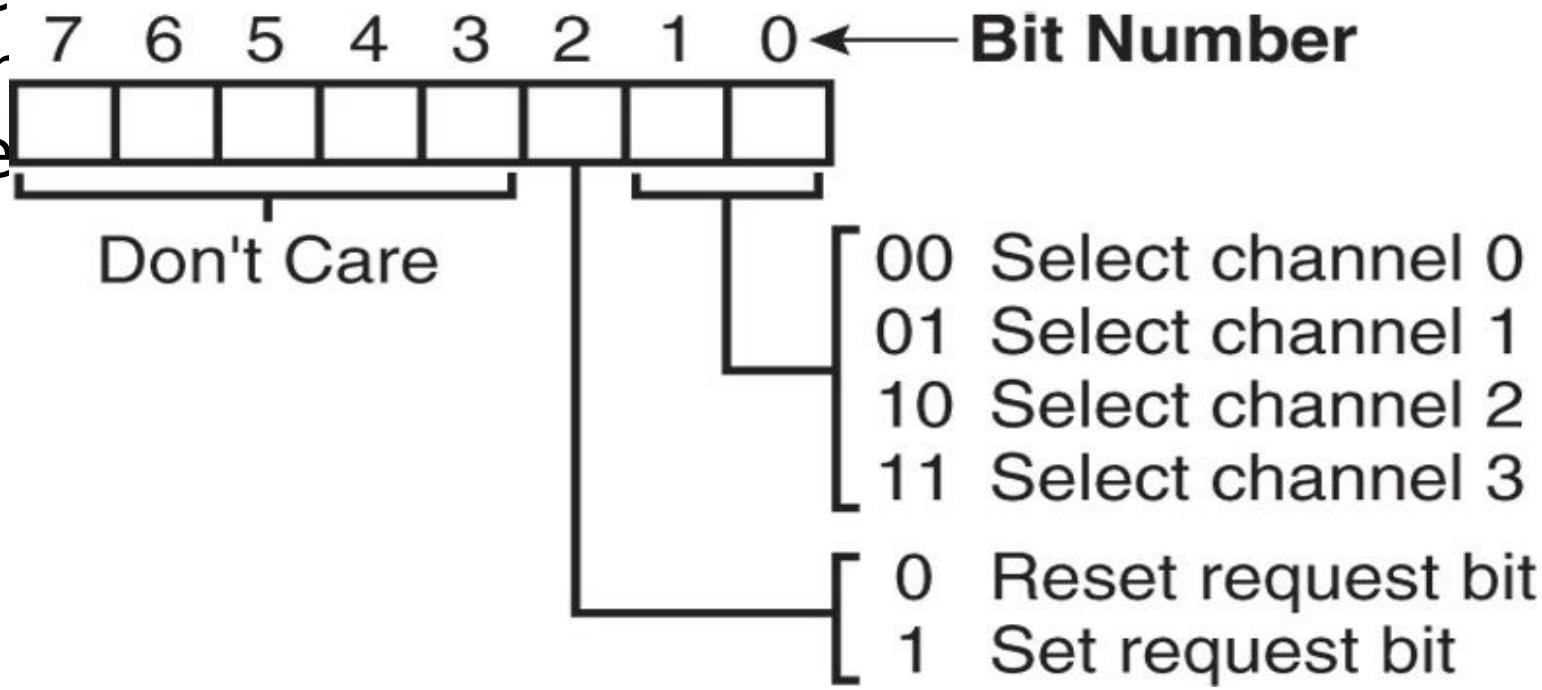
# MR

- The **mode register** programs the mode of operation for a channel.
- Each channel has its own mode register as selected by bit positions 1 and 0.
  - Remaining bits of the mode register select operation, auto-initialization, increment/decrement, and mode for the channel



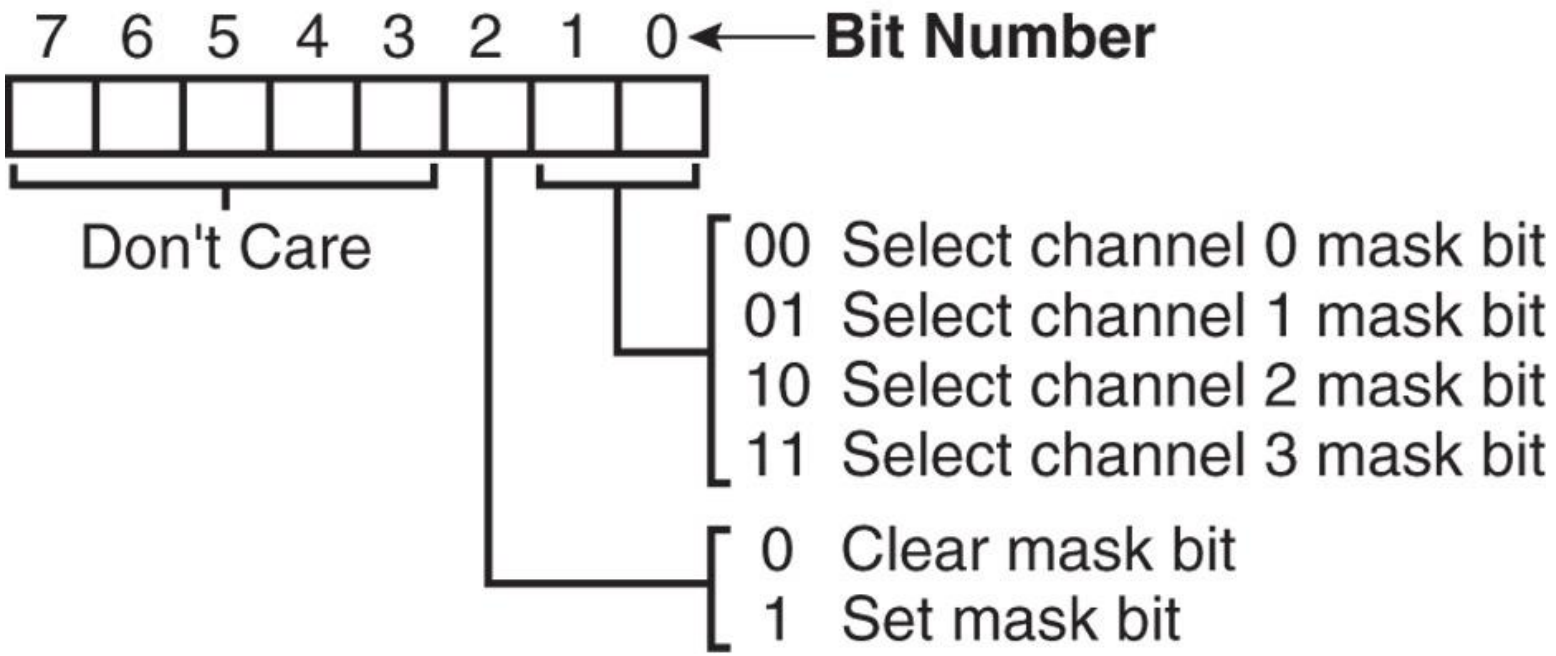
# BR

- The **bus request register** is used to request
  a DMA transfer via software.
  - very useful in memory-to-memory transfers wh
    be



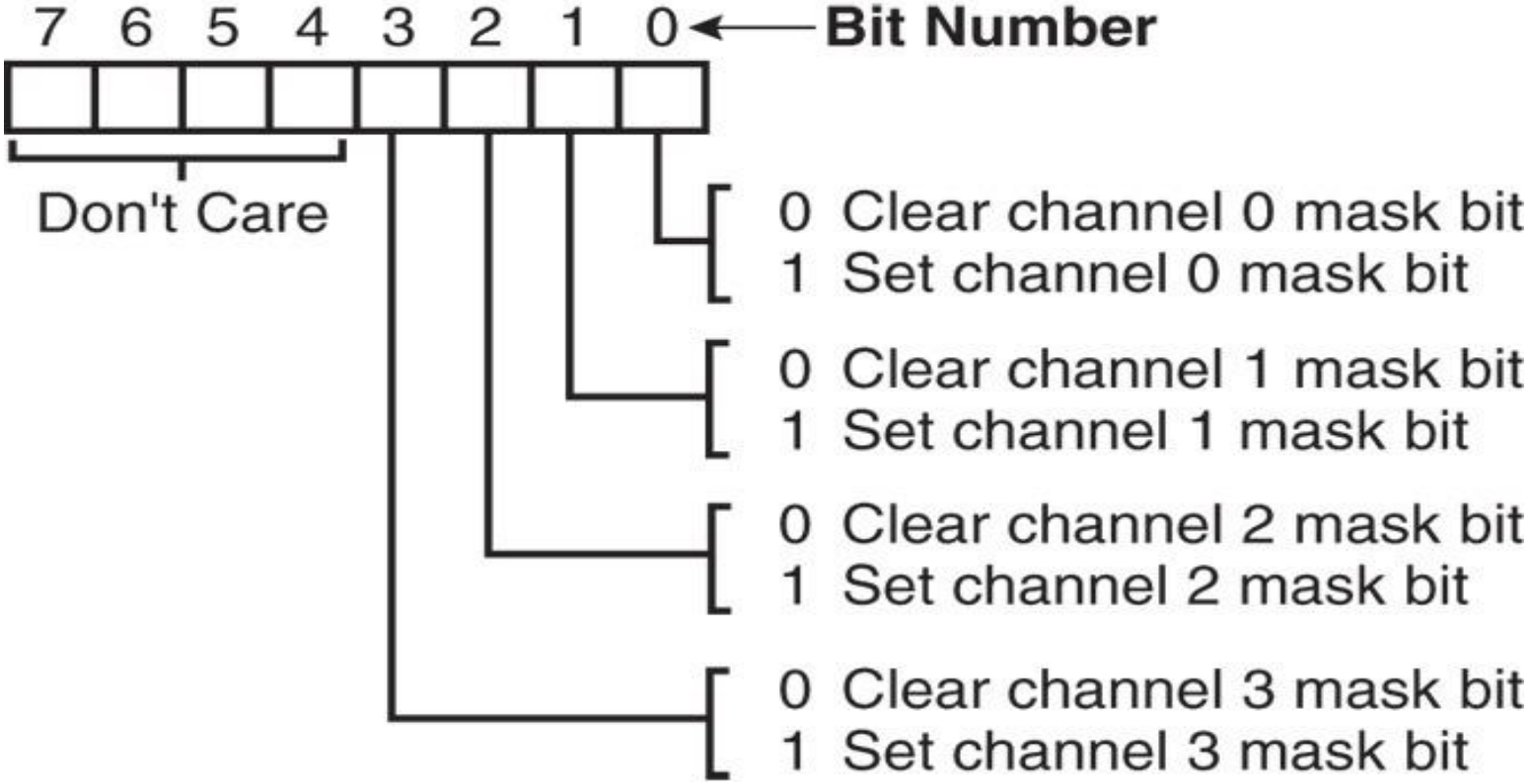
# MRSR

- The **mask register set/reset** sets or clears the channel mask.
  - if the mask is set, the channel is disabled
  - the RESET signal sets all channel masks to disable them



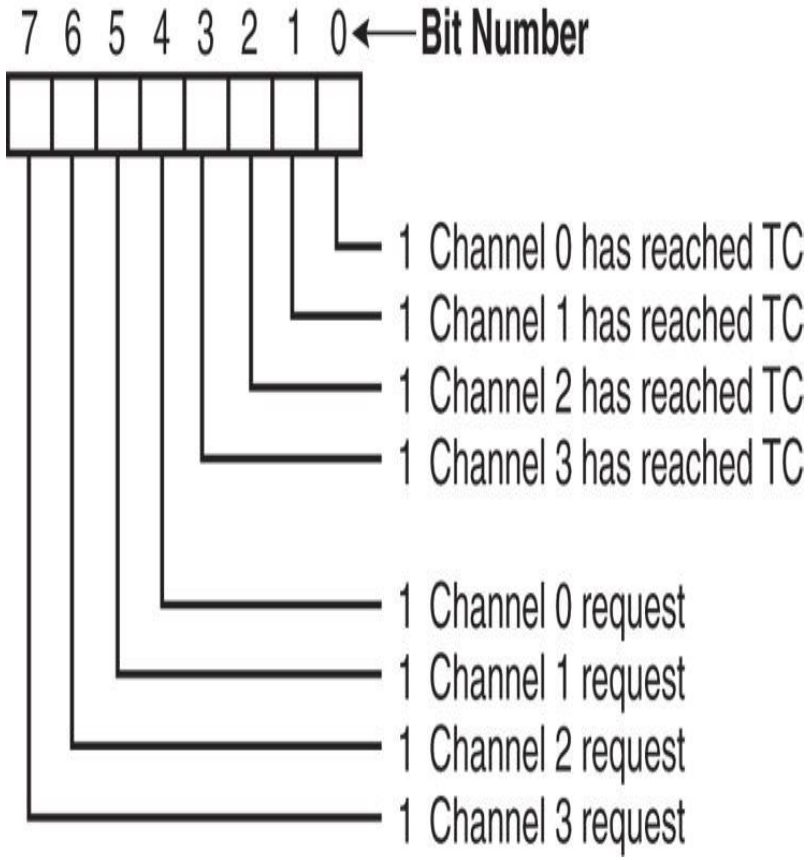
# MSR

- The **mask register** clears or sets all of the masks with one command instead of individual channels, as with the MRSR.



# SR

- The **status register** shows status of each DMA channel. The TC bits indicate if the channel has reached its terminal count (transferred all its bytes).
- When the terminal count is reached, the DMA transfer is terminated for most modes of operation.
- The request bits indicate whether the DREQ input for a



**Master clear**

- Acts exactly the same as the RESET signal to the 8237.
  - As with the RESET signal, this command disables all channels

**Clear mask register**

- Enables all four DMA channels.

**Clear the first/last flip-flop**

- Clears the first/last (F/L) flip-flop within 8237.
- The F/L flip-flop selects which byte (low or high order) is read/written in the current address and current count registers.
  - if F/L = 0, the low-order byte is selected
  - if F/L = 1, the high-order byte is selected
- Any read or write to the address or count register automatically toggles the F/L flip-

# Memory-to-Memory Transfer with the 8237 :

- Memory-to-memory transfer is much more powerful than the automatically repeated MOVSB instruction.
  - most modern chip sets do not support the memory-to-memory feature
- 8237 requires only 2.0 $\mu$s per byte, which is over twice as fast the existing data transfer.

# Unit-4 (Part – 2)

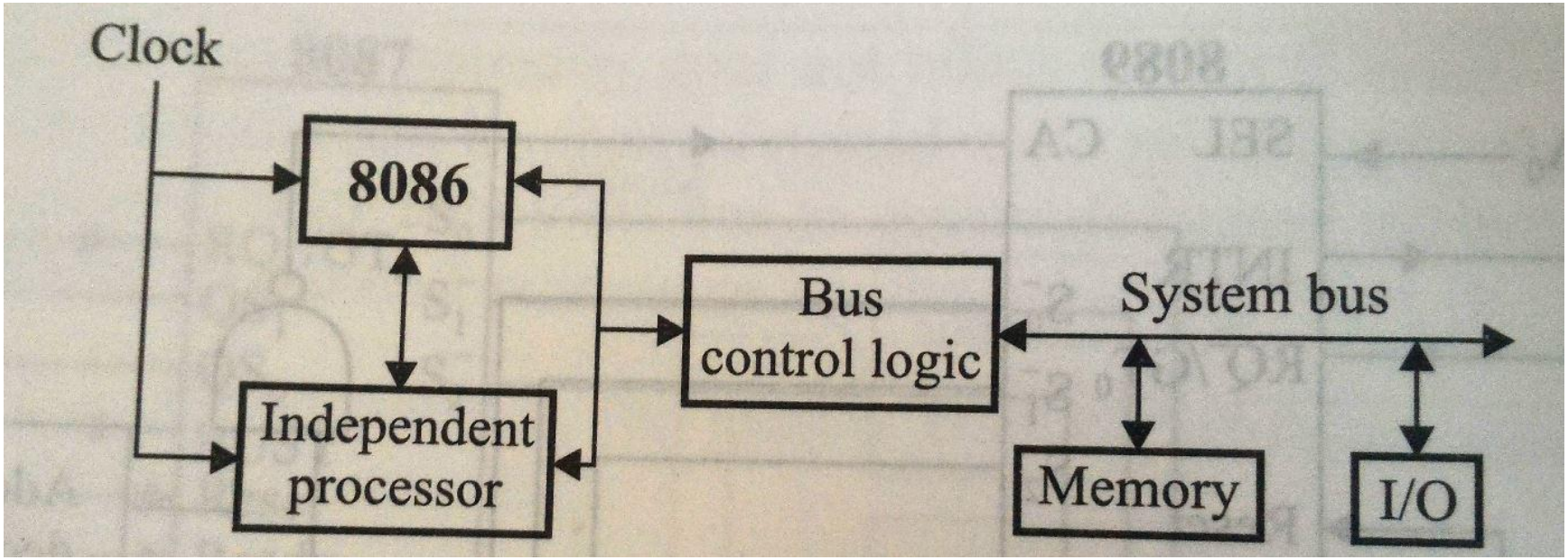
# Multiprocessing & Multiprogramming

- Multiprocessing:
  - Use of two or more processor
- Multiprogramming:
  - Parallel Processing

# 8086 Based Multiprocessing systems

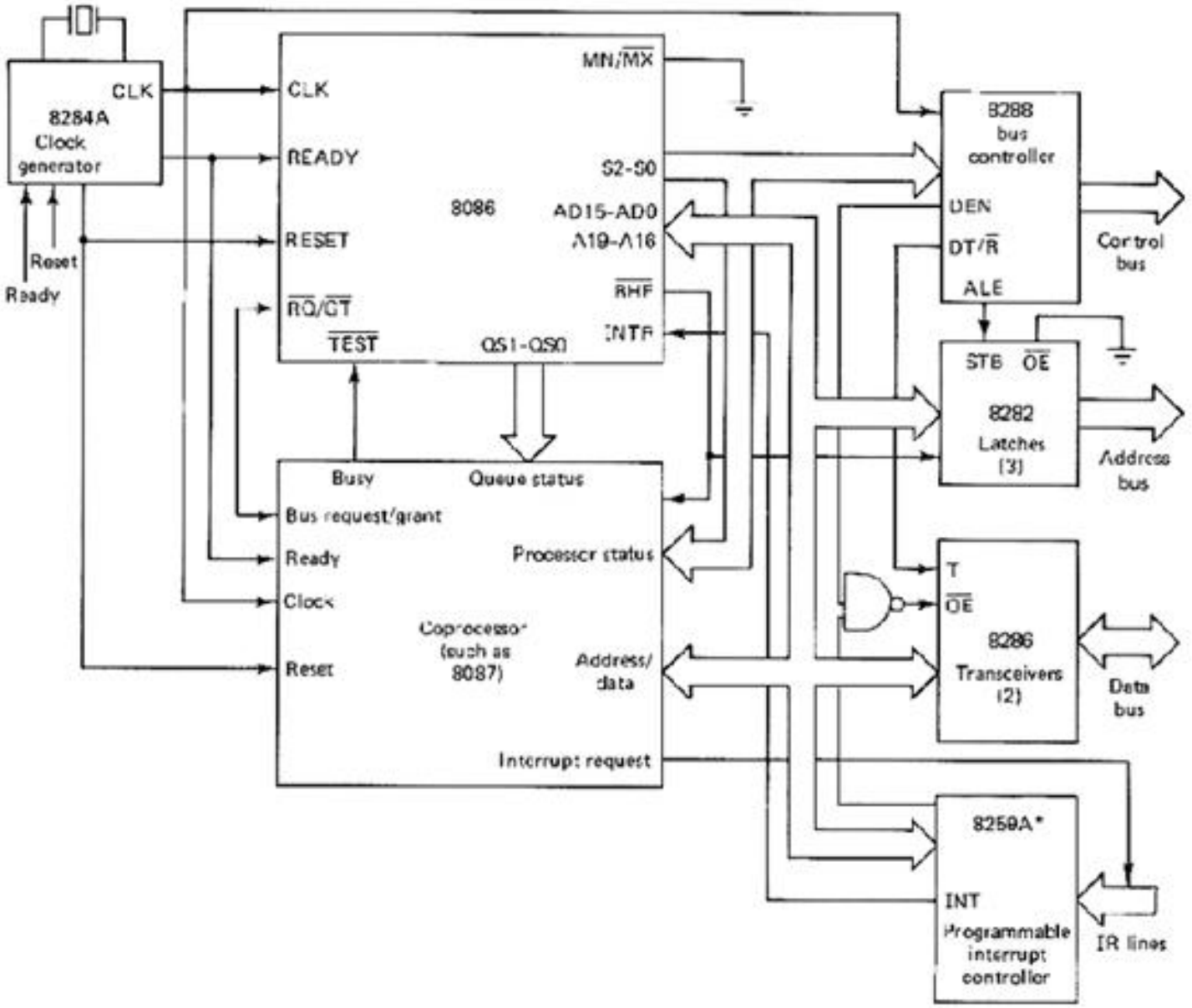
- Maximum mode of 8086 is designed to implement three basic multiprocessor configuration.
  - Coprocessor Configuration
  - Closely coupled configuration
  - Loosely coupled configuration

# Block Diagram of coprocessor configuration



# Coprocessor Configuration

- 8086 is not able to handle complex mathematical applications such as floating point operations. So 8087 is used as co-processor.
- In co-processor configuration both co-processor and host processor share the same memory, I/O, bus and bus control logic and clock generator.
- The processor and co-processor are interfaced through TEST, RQ/GT, QS1 and QS0 signals.
- TEST ------ Busy
- RQ/GT------ RQ/GT
- QS1,QS0 ------ QS1,QS0




Block Diagram of Co-Processor Configuration

# Coprocessor Configuration

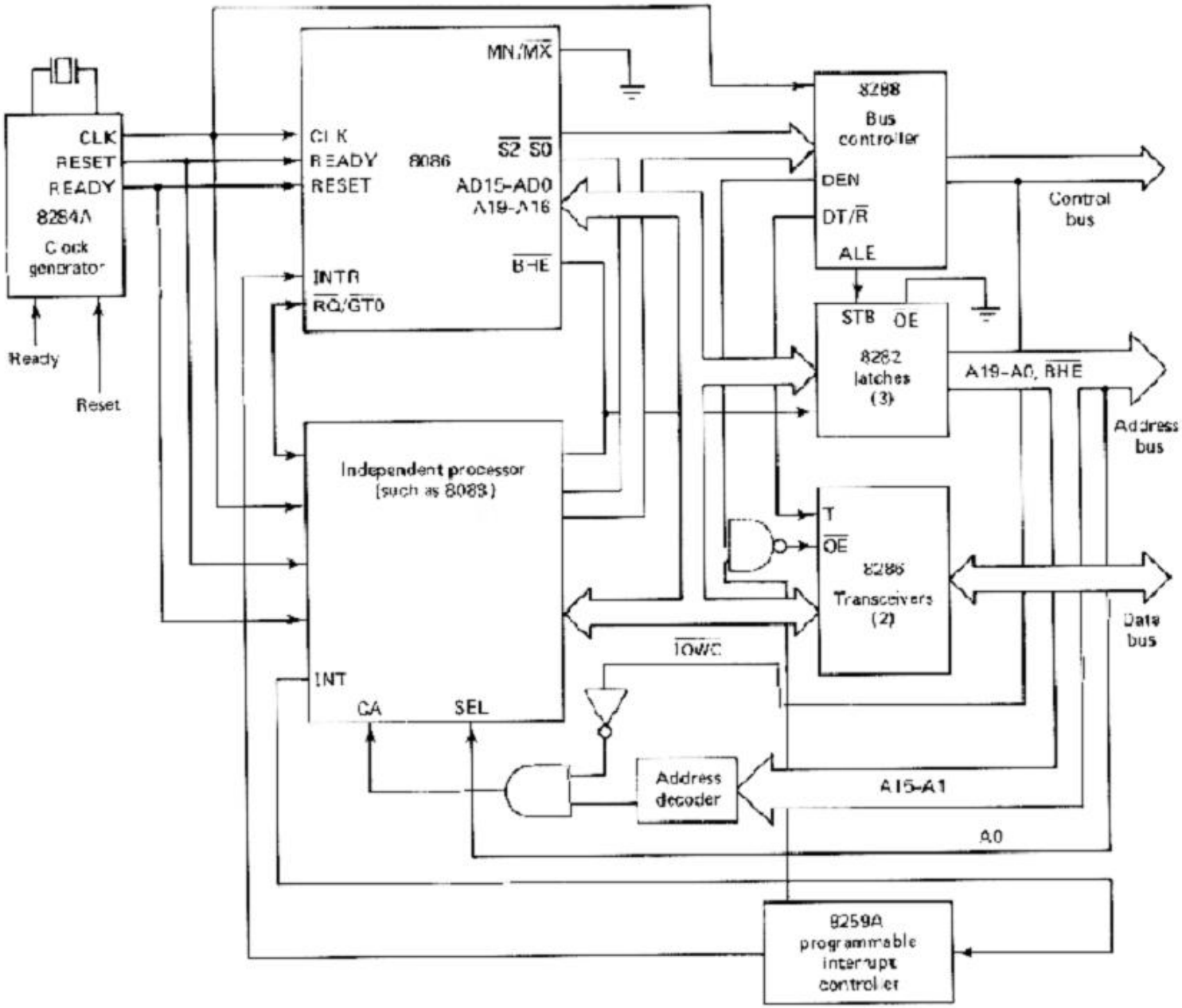
- Coprocessor drives its READY, RESET and CLK signals from the 8284A clock generator. INT is connected with IR of the 8259. S2,S1,S0 are connected to the S2,S1,S0 of the 8288 bus controller.
- Escape instruction
  - An Escape instruction contains as external operation code, indication what the co-processor has to do.
    - ESC opcode ,  operand

# Co-Processor configuration operation

- The instruction are fetched by processor and loaded in to the queues of the both host and co-processor.
- The host processor executes its own instructions and discards the coprocessor instructions and same way co-processor executes its own instruction and discards host processor instruction.

# Closely Coupled Configuration

- A closely coupled configuration is same as co-processor configuration in the sense that these configuration share the same memory , IO System and bus and bus control logic and clock generator.
- But unlike the co-processor configuration the closely coupled configuration consist of the host processor and an independent processor.
- The independent processor ,unlike a coprocessor fetches and executes its own instructions. Co-processor can not take control of bus , it does everything through processor.
- Closely coupled processor can take the control of the bus independently.
- Communication between host and independent processor is accomplished through shared memory. The host setup a message in memory and than wakes up the independent processor by sending a command to the one of the ports.
- The independent processor then access the share memory to get the assigned task in parallel with the host. After the task is

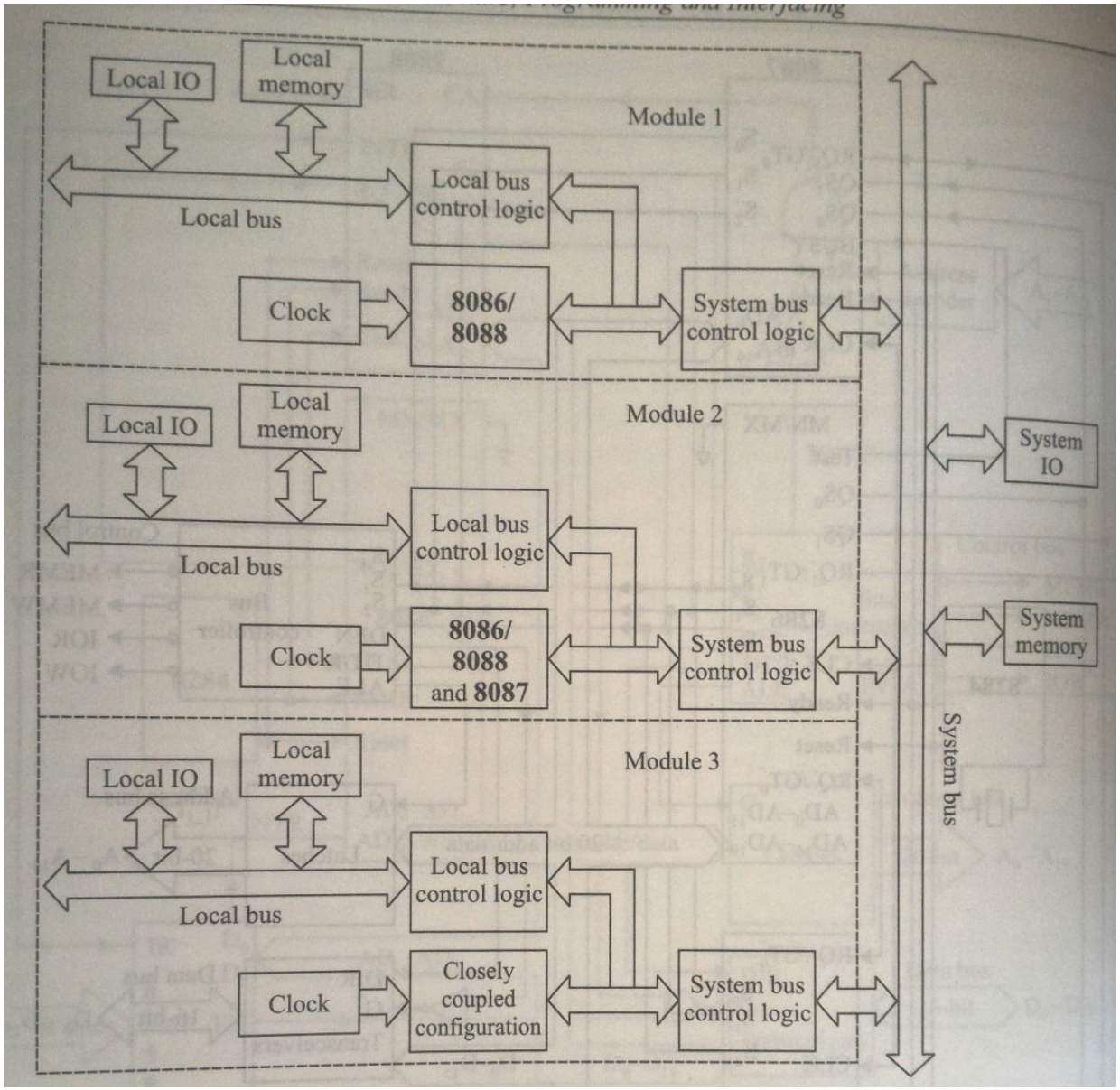



Block Diagram of Closely Coupled configuration

# Loosely Coupled Configuration

- A loosely coupled configuration consist of several independent modules. Each module may be a coprocessor configuration type or closely coupled configuration type.
- Loosely coupled configuration has shared memory , shared system bus and shared I/O. Each system has its own memory as well as its own clock.
- Loosely coupled configuration is used for medium to large multiprocessor system.

| High System Throughput | Bus arbitration |
|---|---|
| System can be expanded in modular form. | Synchronization of local and system clock for synchronous transfer |
| A failure in one module normally does not affect the breakdown of the entire system. | Requires control chips to tie into the system bus. |
| Each bus master has its own local bus to access dedicated | |



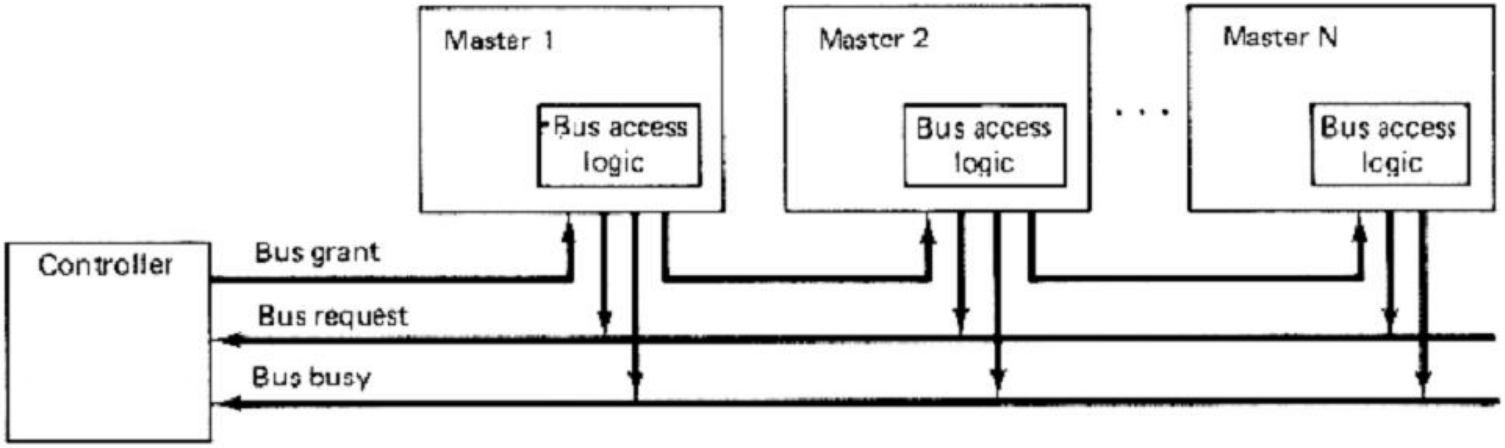

Block Diagram of Loosely coupled configuration

# BUS ALLOCATION SCHEME

- A loosely coupled configuration there are several independent modules and these modules shares the system memory and IO. So it is possible that more than one bus master module may try to access the shared system bus.
- Bus control logic makes sure that only one bus master at a time has a control of the bus. There are three methods.
  - Daisy Chaining
  - Polling
  - Independent  Requesting

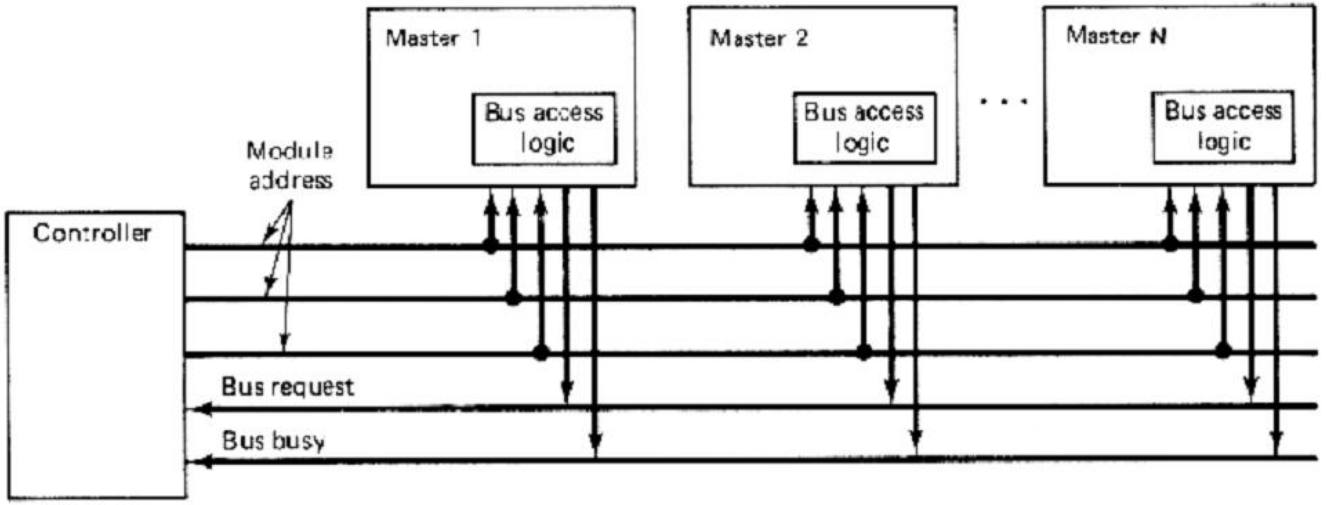
# Daisy Chaining

- Daisy chain is the simplest bus allocation scheme. It requires a bus controller to monitor bus busy(BBSY) and bus request signals (BRQ). In this scheme the same bus request line is used by all bus masters for making bus request.
- This scheme has only one bus grant (BGR) line to respond to a bus request signal. This BGR is generated by controller and propagates to each bus master sequentially starting from the nearest master until is encounters the first one that is requesting access to the bus.
- When BGR signal reaches the requesting bus master , it blocks the further propagation of BGR signal and activates th



# Polling Scheme of BUS Allocation

- This allocation scheme consist of bus controller, on bus busy line (BBSY) , one bus request(BRQ) and n bus grant (BGR) lines. In response to a bus request , the controller generates , the controller generates a sequence of the bus master address.
- When a requesting bus master recognizes its address it activates the busy line and begins to use the bus. The major advantage of polling is that the priority can be dynamically changed by altering the polling sequence stored in the controller. This scheme requires n bus request lines.



# Independent Request Scheme

- This bus allocation scheme consist of  bus controller , one bus busy line (BBSY) , N bus request line (BRQ) and N bus Grant (BGR) lines. All the bus master in the system have their own independent bus request and bus grant lines.
- The independent request scheme resolves priority in a parallel fashion. Each module’s bus request and bus grant lines pair has a priority assigned to it. The controller includes priority decoder, which select request with highest priority and returns corresponding bus grant signal.
- The independent request scheme requires more bus request and three schemes.