5. Input-Output Organization

- 5-1 Peripheral Devices
  - I/O Subsystem
    - Provides an efficient mode of communication between the central system and the outside environment
  - Peripheral (or I/O Device)
    - Input or Output devices attached to the computer
      - » Monitor (*Visual Output Device*) : CRT, LCD
      - » KBD (*Input Device*) : light pen, mouse, touch screen, joy stick, digitizer
      - » Printer (Hard Copy Device) : Dot matrix (*impact*), thermal, ink jet, laser (*non-impact*)
      - » Storage Device : Magnetic tape, magnetic disk, optical disk
  - ASCII (*American Standard Code for Information Interchange*) Alphanumeric Characters
    - I/O communications are usually involved in the transfer of ASCII information
    - ASCII Code :
      - » 7 bit : 00 - 7F (*0 - 127*)
        - 80 - FF (128 - 255) : Greek, Italic, Graphics
- 5-2 Input-Output Interface
  - Interface
    - 1) A conversion of signal values may be required

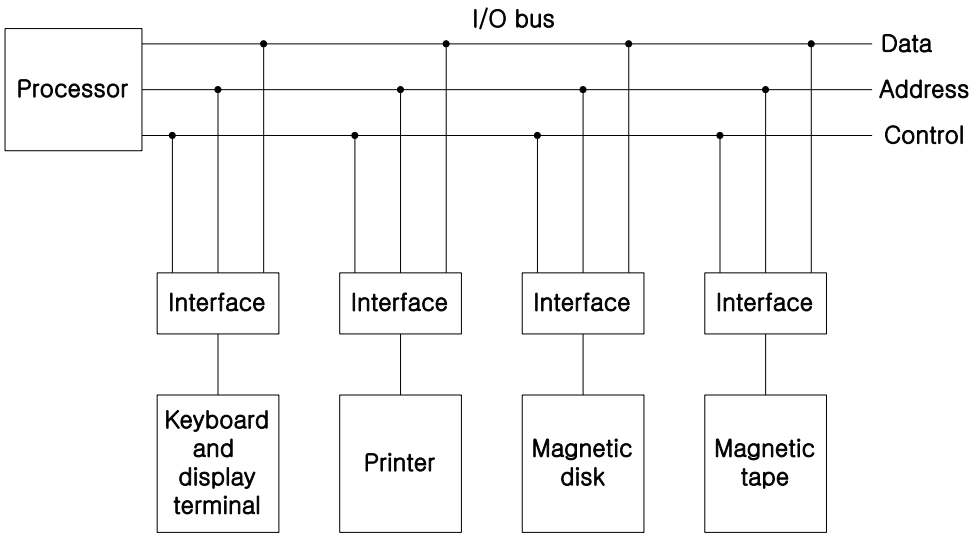
- 2) A synchronization mechanism may be needed
  - » The data transfer rate of peripherals is usually slower than the transfer rate of the CPU
- 3) Data codes and formats in peripherals differ from the word format in the CPU and Memory
- 4) The operating modes of peripherals are different from each other
  - » Each peripherals must be controlled so as not to disturb the operation of other peripherals connected to the CPU

Interface

- Special hardware components between the CPU and peripherals
- Supervise and Synchronize all input and output transfers

I/O Bus and Interface Modules :

- I/O Bus
  - » Data lines
  - » Address lines
  - » Control lines
- Interface Modules



- I/O command :
  - » Control Command
  - » Status Command
  - » Input Command
  - » Output Command

◆ I/O Bus versus Memory Bus

- Computer buses can be used to communicate with memory and I/O
  - » 1) Use two separate buses, one for memory and the other for I/O
    - I/O Processor
  - » 2) Use one common bus for both memory and I/O but have separate control lines for each : ***Isolated I/O*** or ***I/O Mapped I/O***
    - **IN**, **OUT** : I/O Instruction
    - **MOV** or **LD** : Memory read/write Instruction

    * **Control Lines**
    I/O Request, Mem Request, Read/Write

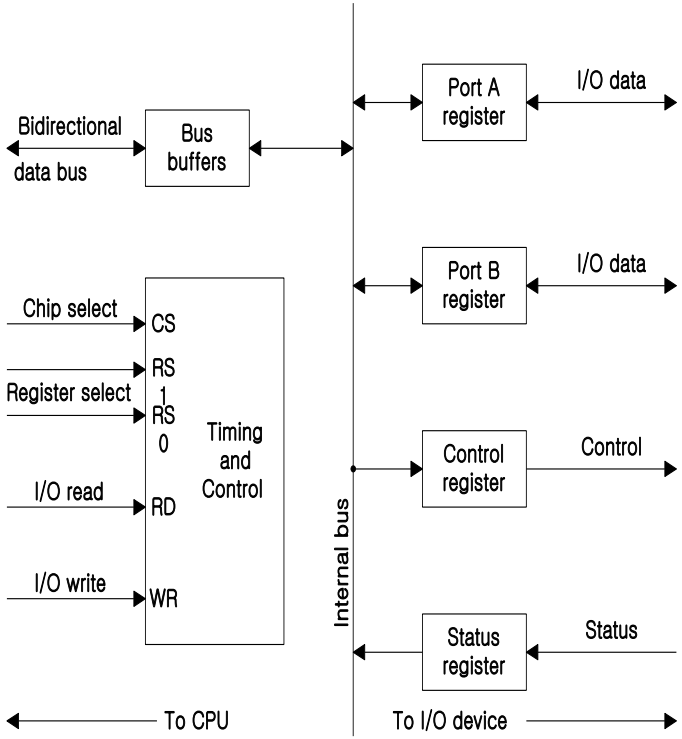
  - » 3) Use one common bus for memory and I/O with common control lines : ***Memory Mapped I/O***
    - **MOV** or **LD** : I/O and Memory read/write Instruction

    * **Control Lines**
    Read/Write

- Example of I/O Interface : ***Fig. 11-2***
  - 4 I/O port : Data port A, Data port B, Control, Status
  - Address Decode : CS, RS1, RS0

5.3 Asynchronous Data Transfer

- Synchronous Data Transfer
  - All data transfers occur simultaneously during the occurrence of a clock pulse
  - Registers in the **interface** share a common clock with **CPU** registers
- Asynchronous Data Transfer
  - Internal timing in each unit (*CPU and Interface*) is independent
  - Each unit uses its own private clock for internal registers



CS	RS 1	RS 0	Register selected
0	x	x	None : data bus in high-impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

◆ Strobe : Control signal to indicate the time at which data is being transmitted

- 1) Source-initiated strobe :
- 2) Destination-initiated strobe :

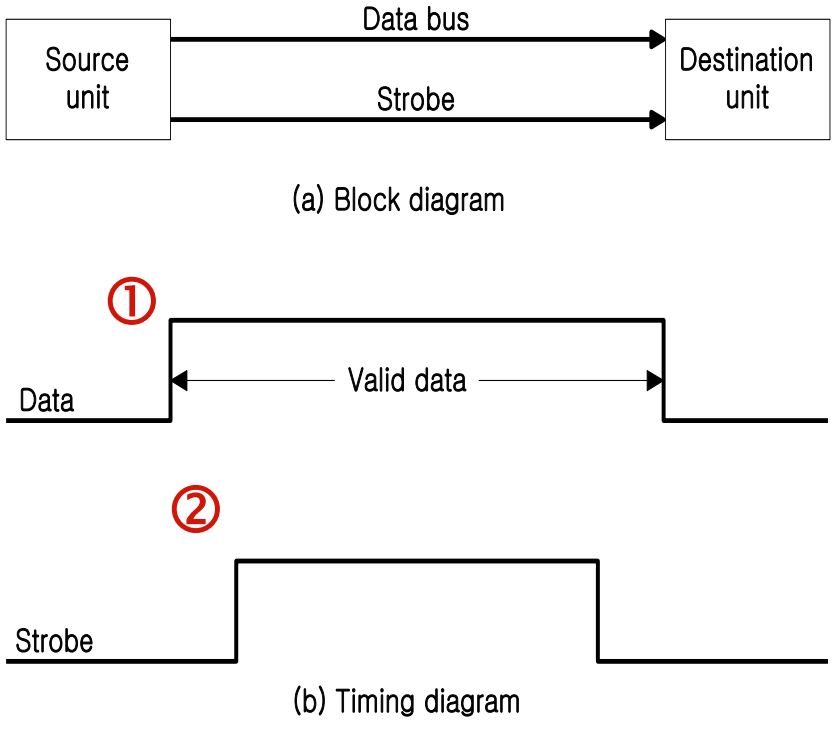


Fig. 11-3 Source-initiated strobe

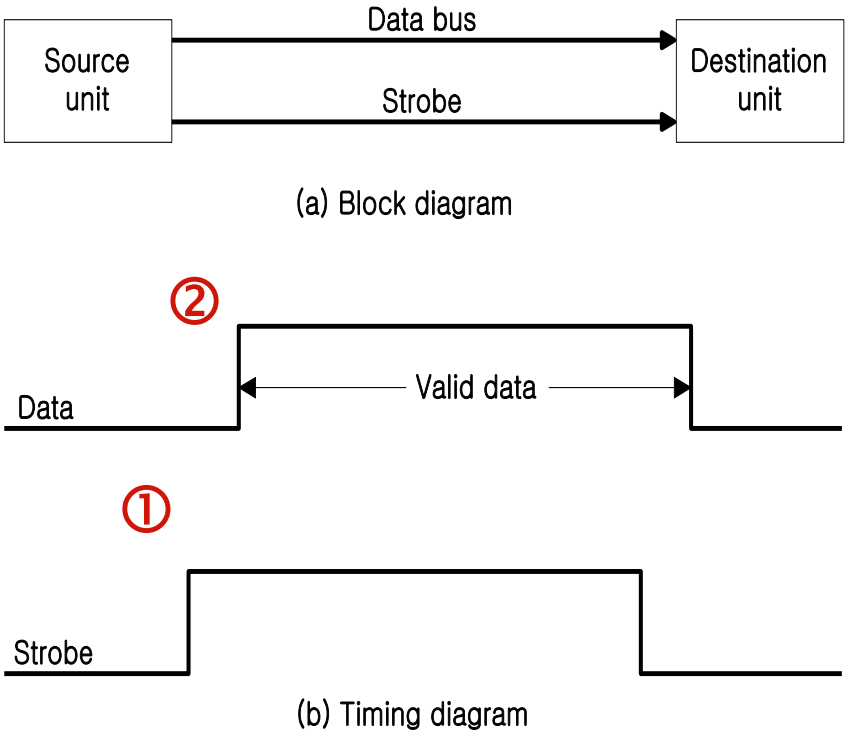


Fig. 11-4 Destination-initiated strobe

- Disadvantage of strobe method

◆ Handshake : Agreement between two independent units

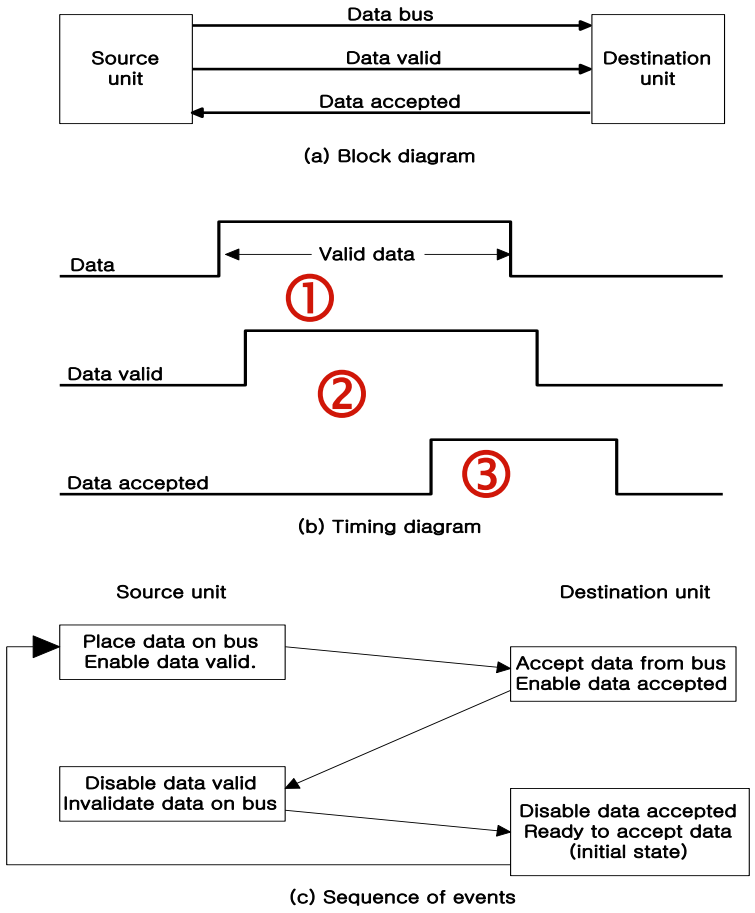


Fig. 11-5 Source-initiated handshake

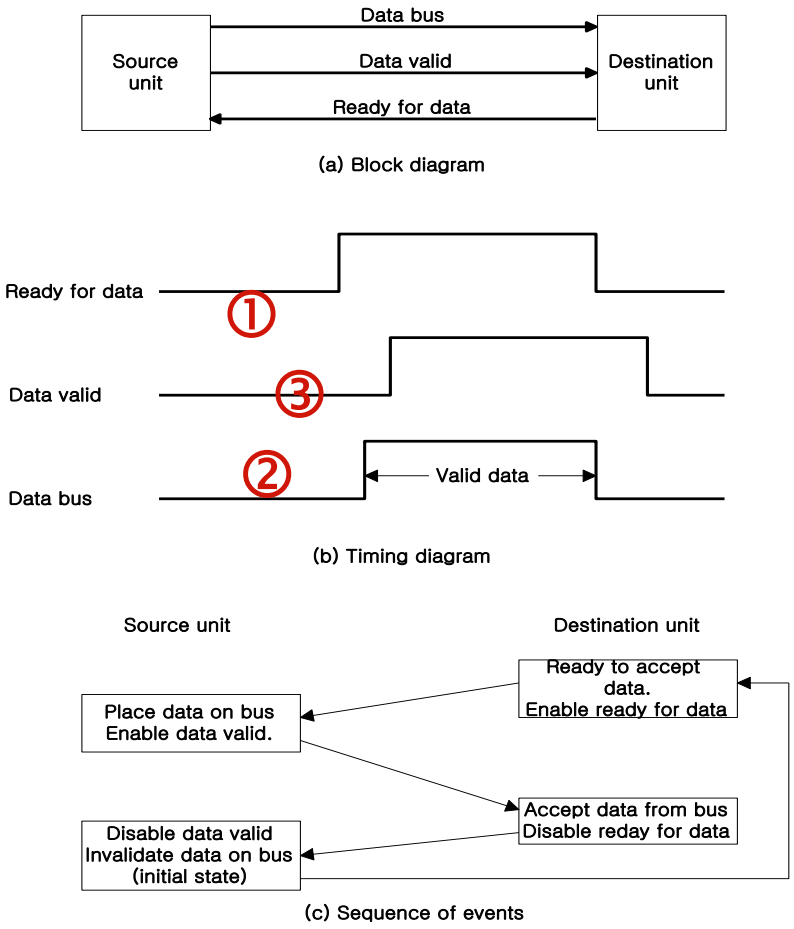
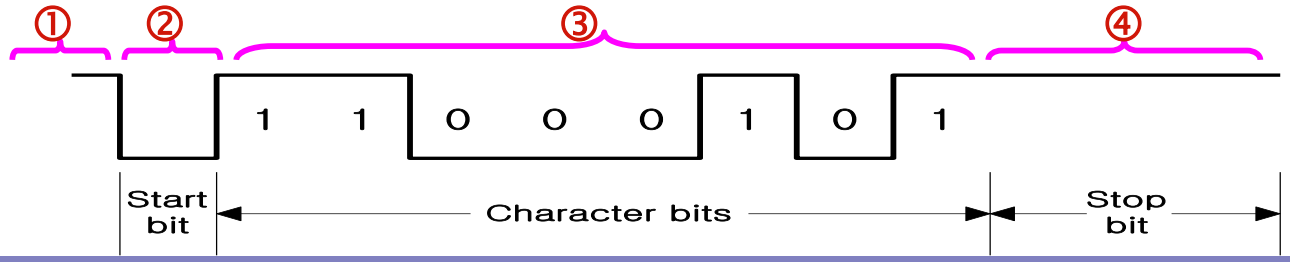


Fig. 11-6 Destination-initiated handshake

- **Timeout**

Asynchronous Serial Transfer

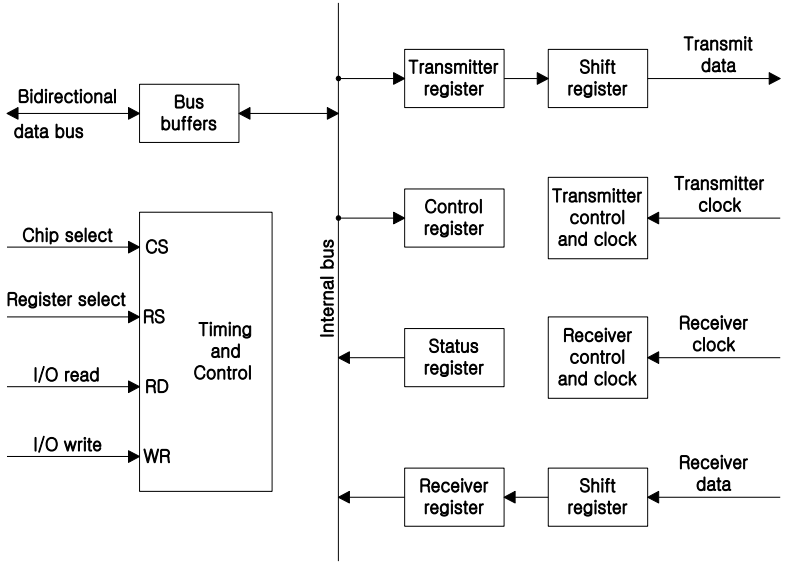
- Synchronous transmission :
  - » The two unit share a common clock frequency
  - » Bits are transmitted continuously at the rate dictated by the clock pulses
- Asynchronous transmission :
  - » Special bits are inserted at both ends of the character code
  - » Each character consists of three parts :
    - 1) start bit : always “0”, indicate the beginning of a character
    - 2) character bits : data
    - 3) stop bit : always “1”
- Asynchronous transmission rules :
  - » ① When a character is not being sent, the line is kept in the 1-state
  - » ② The initiation of a character transmission is detected from the start bit, which is always “0”
  - » ③ The character bits always follow the start bit
  - » ④ After the last bit of the character is transmitted, a stop bit is detected when the line returns to the 1-state for at least one bit time



- Baud Rate : Data transfer rate in bits per second
  - » 10 character per second with 11 bit format = 110 bit per second
- UART (Universal Asynchronous Receiver Transmitter) : 8250
- UART (Universal Synchronous/Asynchronous Receiver Transmitter) : 8251

Asynchronous Communication Interface :

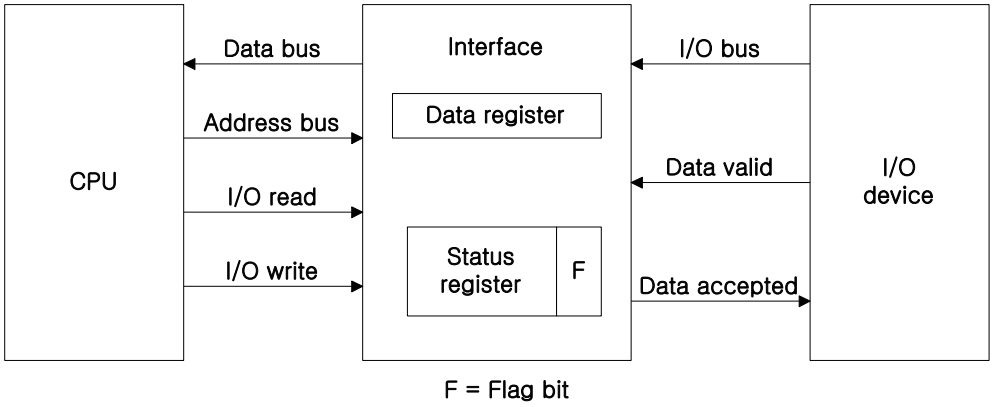
- » 80 : Data Write/Read (*Transmit/Receive*)
- » 81 : Control Write/ Status Read
  - **A0** = RS (*register select*)
- Double Buffered (*in transmit register*)
  - » New character can be loaded as soon as the previous one starts transmission
- 3 possible errors (*in status register*)
  - » 1) parity error
    - Even or Odd parity error
  - » 2) framing error
    - right number of stop bits is not detected at the end of the received character
  - » 3) overrun error
    - CPU does not read the character from the receiver register before the next one is available

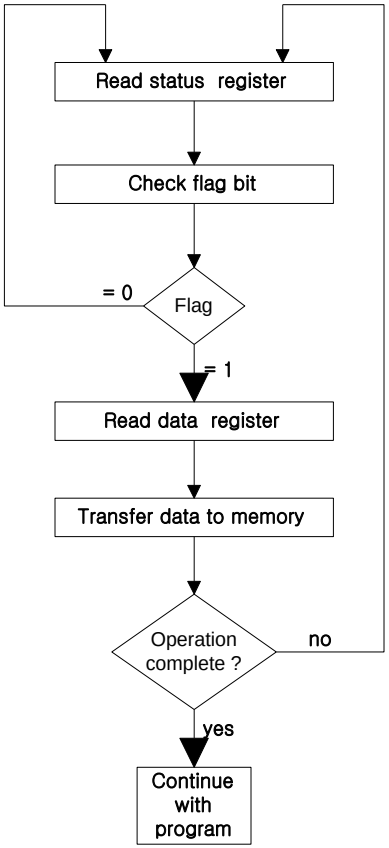


CS	RS	Operation	Register selected
0	×	×	None : data bus in high-impedance
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

5-4 Modes of Transfer

- Data transfer to and from peripherals
  - 1) Programmed I/O : *in this section*
  - 2) Interrupt-initiated I/O : *in this section and sec. 11-5*
  - 3) Direct Memory Access (**DMA**) : *sec. 11-6*
  - 4) I/O Processor (**IOP**) : *sec. 11-7*
- Example of Programmed I/O : ***Fig. 11-10, 11-11***





- Interrupt-initiated I/O
  - 1) Non-vectored : fixed branch address
  - 2) Vectored : interrupt source supplies the branch address (**interrupt vector**)

- Software Considerations
  - I/O routines
    - » software routines for controlling peripherals and for transfer of data between the processor and peripherals
  - I/O routines for standard peripherals are provided by the manufacturer (**Device driver, OS** or **BIOS**)
  - I/O routines are usually included within the operating system
  - I/O routines are usually available as operating system procedures (**OS** or **BIOS function call**)

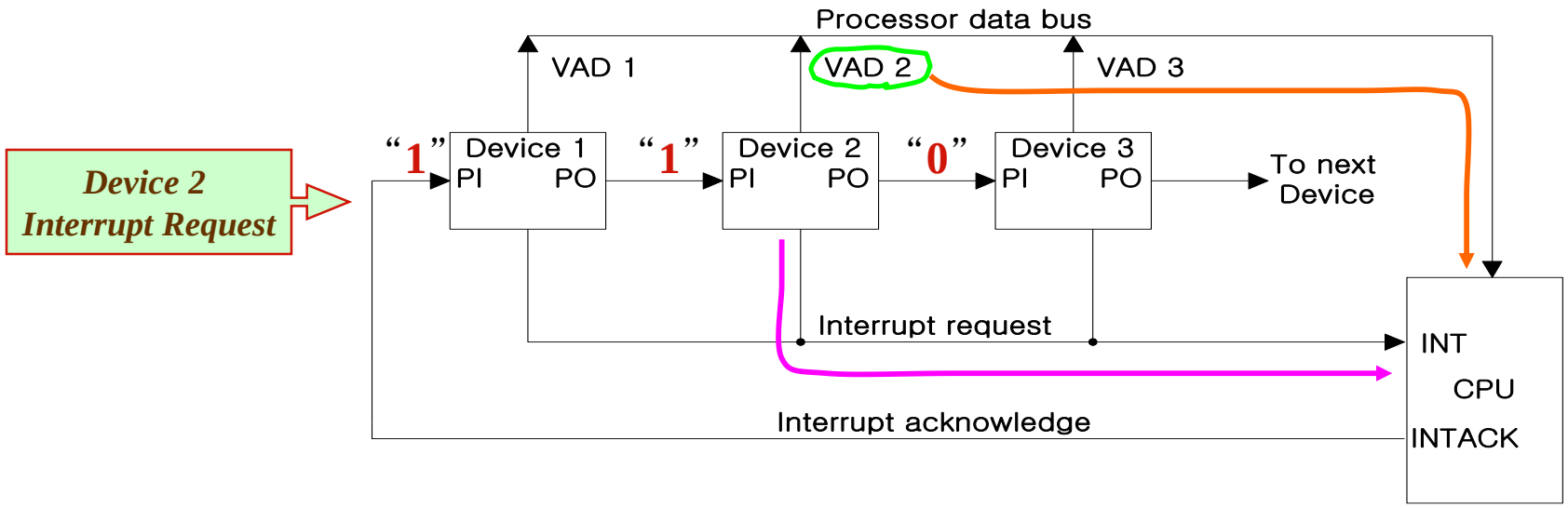
5-5 Priority Interrupt

- Priority Interrupt
  - Identify the source of the interrupt when several sources will request service simultaneously
  - Determine which condition is to be serviced first when two or more requests arrive simultaneously
    - » 1) Software : **Polling**
    - » 2) Hardware : **Daisy chain**, **Parallel priority**

Polling

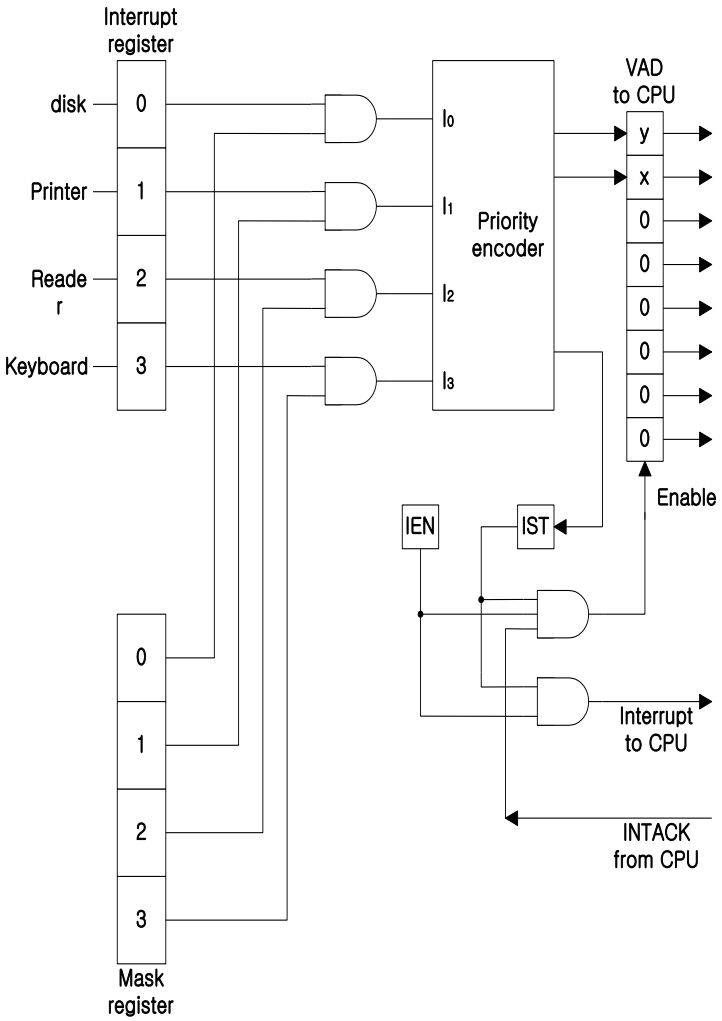
- Identify the highest-priority source by software means
  - » One common branch address is used for all interrupts
  - » Program polls the interrupt sources in sequence
  - » The highest-priority source is tested first
- Polling priority interrupt If there are many interrupt sources, the time required to poll them can exceed the time available to service the I/O device
  - » Hardware priority interrupt

Daisy-Chaining : ***Fig. 11-12***



Parallel Priority

- Priority Encoder –
- Parallel Priority : ***Fig. 11-14***
  - » Interrupt Enable F/F (**IEN**) : set or cleared by the program
  - » Interrupt Status F/F (**IST**) : set or cleared by the encoder output
- Priority Encoder Truth Table : ***Tab. 11-2***



Interrupt Cycle

- At the end of each instruction cycle, CPU checks IEN and IST
- if both IEN and IST equal to “1”
- CPU goes to an Instruction Cycle
  - » Sequence of microoperation during Instruction Cycle

Branch to ISR

$SP \leftarrow SP - 1$: Decrement stack point

$M[SP] \leftarrow PC$: Push PC into stack

$INTACK \leftarrow 1$: Enable INTACK

$PC \leftarrow VAD$: Transfer VAD to PC

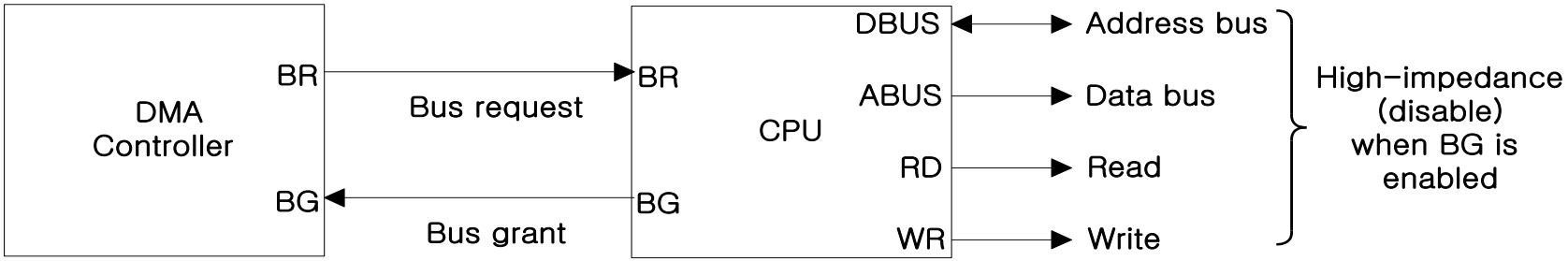
$IEN \leftarrow 0$: Disable further interrupts

Go to Fetch next instructio n

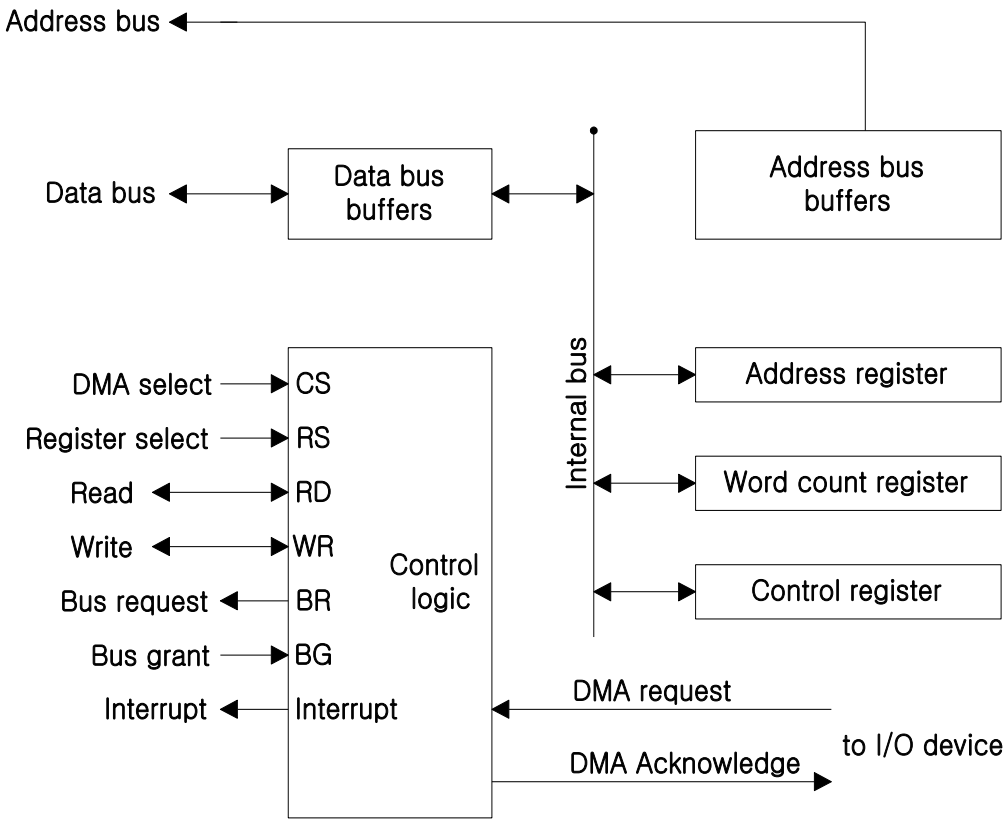
5-6 Direct Memory Access (DMA)

DMA

- DMA controller takes over the buses to manage the transfer directly between the **I/O device** and **memory** (**Bus Request/Grant**)

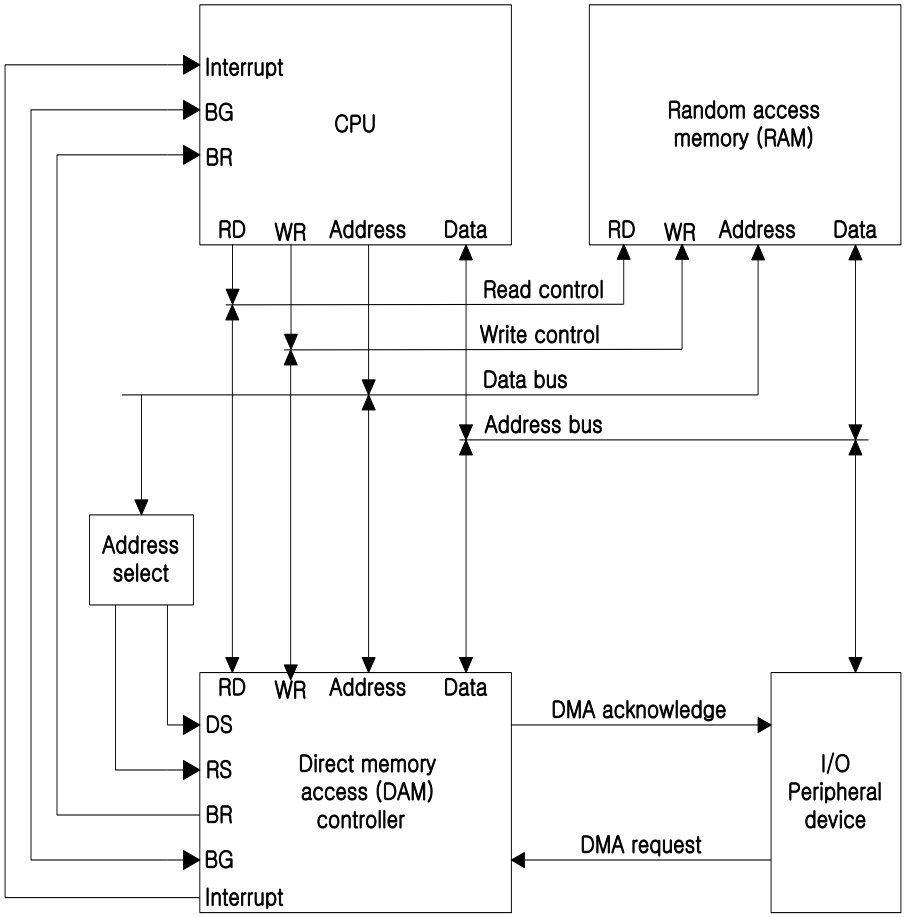


- Transfer Modes
  - 1) Burst transfer : Block
  - 2) Cycle stealing transfer : Byte
- DMA Controller (Intel 8237 DMAC) : ***Fig. 11-17***
  - DMA Initialization Process
    - » **1)** Set Address register :
      - memory address for read/write
    - » **2)** Set Word count register :
      - the number of words to transfer
    - » **3)** Set transfer mode :
      - read/write,
      - burst/cycle stealing,
      - I/O to I/O,
      - I/O to Memory,
      - Memory to Memory
      - Memory search
      - I/O search
    - » 4) DMA transfer start : *next section*
    - » 5) EOT (End of Transfer) :
      - Interrupt



DMA Transfer (I/O to Memory)

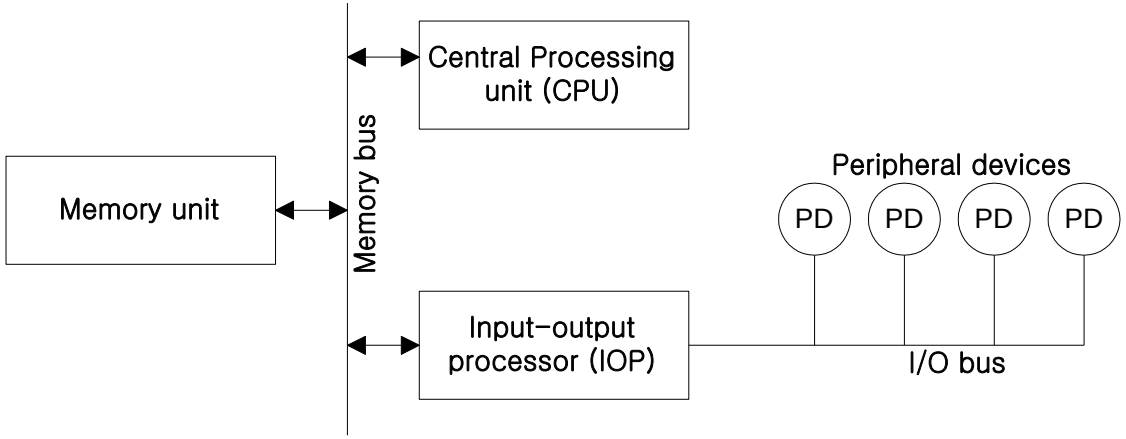
- 1) I/O Device sends a DMA request
- 2) DMAC activates the **BR** line
- 3) CPU responds with **BG** line
- 4) DMAC sends a DMA acknowledge to the I/O device
- 5) I/O device puts a word in the data bus (*for memory write*)
- 6) DMAC write a data to the address specified by **Address register**
- 7) Decrement **Word count register**
- 8) **Word count register** **EOT** interrupt CPU
- 9) **Word count register** DMAC checks the DMA request from I/O device



5-7 Input-Output Processor (IOP)

IOP :

- Communicate directly with all I/O devices
- Fetch and execute its own instruction
  - » IOP instructions are specifically designed to facilitate I/O transfer
  - » DMAC must be set up entirely by the CPU
- Designed to handle the details of I/O processing

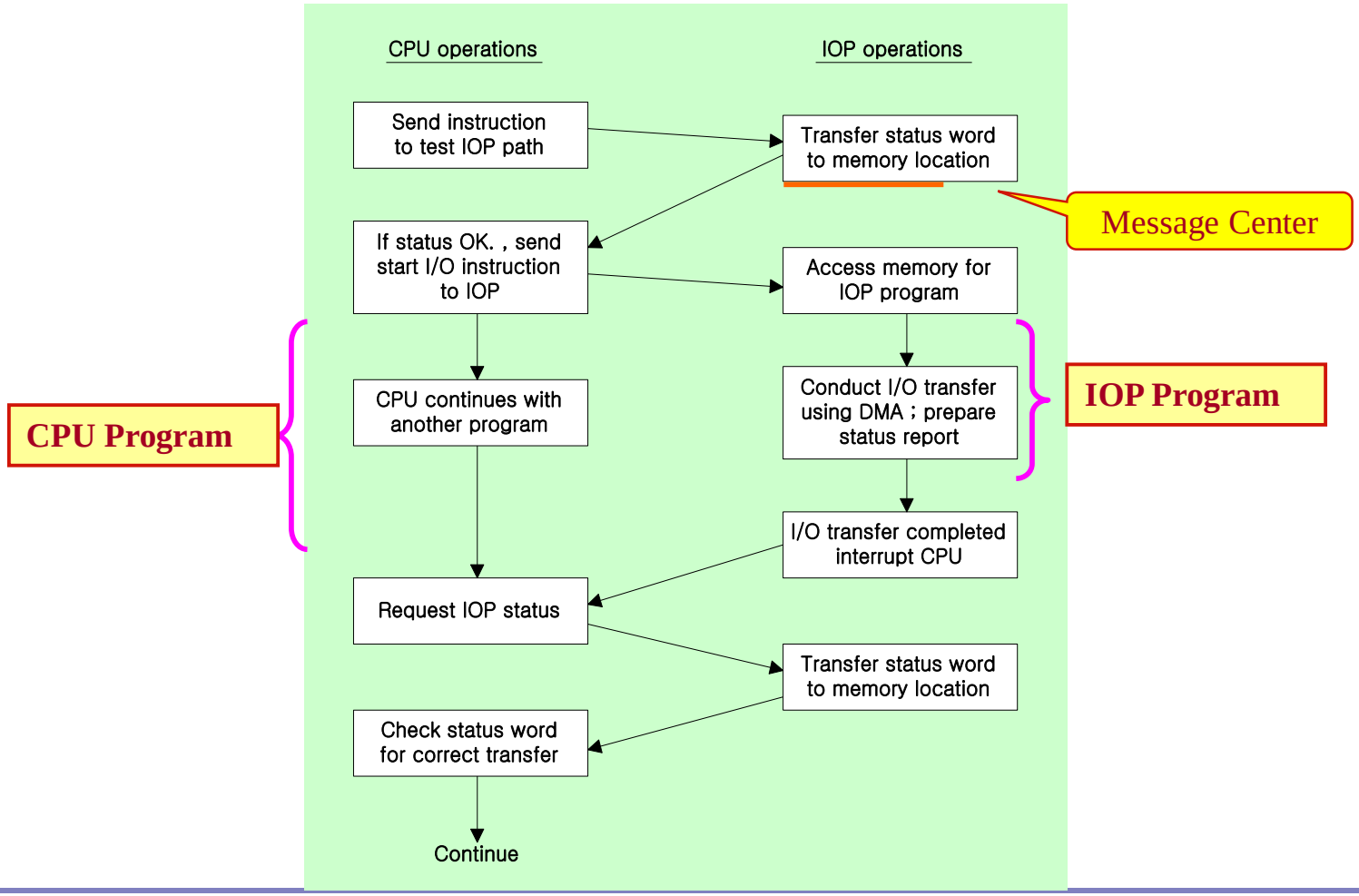


Command

- **Instruction** *that are read form memory by an* ***IOP***
  - » Distinguish from instructions that are read by the CPU
  - » Commands are prepared by experienced programmers and are stored in memory
  - » Command word = IOP program

CPU - IOP Communication :

- Memory units acts as a message center : Information
  - » each processor leaves information for the other



5-8 Serial Communication

Difference between I/O Processor and Data Communication Processor

- I/O Processor
  - » communicate with peripherals through a common I/O bus (data, address, control bus)
- Data Communication Processor
  - » communicate with each terminal through a single pair of wires

Modem

- Convert digital signals into audio tones to be transmitted over telephone lines
- Various modulation schemes are used (FM, AM, PCM)

Block transfer

- An entire block of characters is transmitted in synchronous transmission
- Transmitter sends **one more character** (**error check**) after the entire block is sent.

Error Check

- LRC (Longitudinal Redundancy Check) : XOR
- CRC (Cyclic Redundancy Check) : Polynomial

3 Transmission System

- Simplex : one direction only
- Half-duplex : both directions but only one direction at a time
- Full-duplex : both directions simultaneously

Data Link

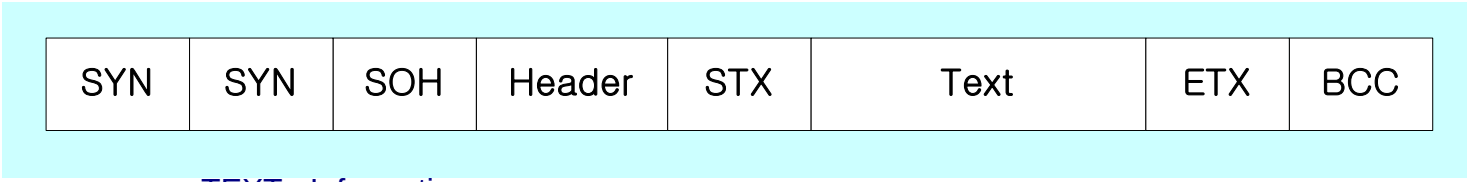
- The communication lines, modems, and other equipment used in the transmission of information between two or more stations

Data Link Protocol

- 1) Character-Oriented Protocol
- 2) Bit-Oriented Protocol

Character-Oriented Protocol

- Message format for Character-Oriented Protocol :



  - » TEXT : Information
  - » BCC : Block Check Character (LRC or CRC)
- ASCII Communication Control Character :
  - » **SYN** (0010110) : Establishes synchronism
  - » **SOH** (0000001) : Start of Header (address or control information)
  - **STX** (0000010) : Start of Text
  - **ETX** (0000011) : End of Text

Data Transparency

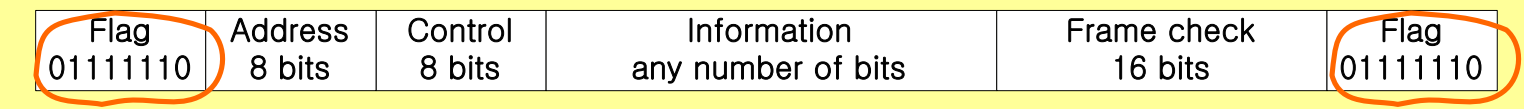
- **DLE** (Data Link Escape) Character

DLE

- Inserting a DLE character (*bit pattern = 00010000*) before each control character
  - » ***Exam)*** **DLE** ETX **DLE** SYN
- DLE character is inefficient and somewhat complicated to implement
- So, we need another protocol, i.e. Bit-Oriented Protocol

Bit-Oriented Protocol

- Transmit a serial bit stream (**Frame**) of any length without character boundaries
- Examples of bit-oriented protocol
  - » 1) **SDLC** (Synchronous Data Link Control) : IBM
  - » 2) **HDLC** (High-level Data Link Control) : ISO
  - » 3) **ADCCP** (Advanced Data Communication Control Procedure) : ANSI
- Frame format for bit-oriented protocol : ***Fig. 11-26***



  - » Flag : A frame starts and ends with 8-bit flag (**01111110**)

- Zero Insertion
  - » Prevent a flag from occurring in the middle of a data frame
  - » Zero (**0**) is inserted by transmitting station after any succession of five continuous 1's
    - Example of zero insertion : 01111110 (data) → 011111**0**10
  - » Receiver always removes a 0 that follows a succession of five 1's