Interrupt

- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data. The processor while waiting must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.
- To improve the performance, **Interrupt** (means **break the continuity**) is used. The processor issues an I/O command (read or write) to an I/O module and then go on to do some other useful work. The I/O module then interrupt the processor to exchange data with the processor when it is ready to serve. The processor now executes that data transfer and resumes the processing.
- The **interrupt** is a **signal** emitted by **hardware or software** when a process or an event needs immediate attention.
- By this mechanism, modules like **I/O or memory may interrupt the normal processing of the processor**.
- Interrupts are provided primarily as a way to **improve processing efficiency**.
- In I/O devices one of the bus control lines is dedicated for this purpose and is called the ***Interrupt Service Routine (ISR)***.

Interrupt

- **Interrupt processing:** The occurrence of an interrupt triggers a number of events, both in the processor hardware and in software. **The sequence of hardware events occurs**:
  - **Step 1**: The I/O device issues an interrupt signal to the processor.
  - **Step 2**: The processor finishes execution of the current instruction before responding to the internet.
  - **Step 3**: Then the processor checks for an interrupt and sends an acknowledgement signal to the device. The acknowledgement allows the device to remove the interrupt signal.
  - **Step 4**: The processor saves the information needed to resume the current program at the point of interrupt. The minimum information required to save is the status of the processor which is contained in a register called program status word (PSW), and the location of the next instruction to be executed contained in the program counter.
  - **Step 5**: Now the processor loads the new program counter value based on interrupt.

Next, **the sequence of software events occurs**:

- **Step 6**: At this point, The program counter and PSW relating to the interrupted program have been saved on the system stack.
- **Step 7**: The interrupt handler next processes the interrupt.
- **Step 8**: When interrupt processing is complete, the saved register values are retrieved from the stack and restored to the registers.
- **Step 9**: The final act is to restore the PSW and program counter values from the stack. Now the next instruction to be executed is from the previously interrupted.

Interrupt

- **Design issues/Controlling multiple devices**: Two issues arise in implementing interrupt I/O.
  - As there are multiple I/O devices, how does the processor determine which device issued the interrupt?
  - If multiple interrupt occurs, how does the processor determine which one to process?

  The first issue can be handled by the following techniques:
    - Multiple interrupt lines
    - Software poll
    - Daisy chain (hardware poll)
    - Bus arbitration

Interrupt

- **Multiple interrupt lines**:
  - The most straight method is to have multiple interrupt lines between the processor and the I/O modules.
  - It is impractical though.
  - Even if there are multiple lines, each line will have multiple I/O modules attached to it.
  - So, one of the other three techniques must have to use on each line.
- **Software poll:**
  - When the processor detects an interrupt, it branches to an interrupt service routine whose job is to poll each I/O module to determine which module caused the interrupt.
  - The poll could be a separate command line.
  - The processor places the address of a particular I/O on the command line, the I/O module responds positively back by sending a signal if it requested for interrupt.
  - Another way is, each I/O module may have a status register, and the processor reads the status register of each module to identify the interrupted module.
  - The dis advantage is that it is time consuming.

Interrupt

- **Daisy chain (hardware poll)**:
  - A more efficient way.
  - It is a hardware poll.
  - Each I/O module shares a common interrupt request line. The interrupt acknowledge line is daisy chained through the modules.
  - When the processor detects an interrupt, it sends out an interrupt acknowledge. The acknowledge signal propagates through a series of I/I modules until it gets to a requesting module.
  - The requesting I/I module then places a word (vector) on the data line. The word contain either the address of the I/O module or some other unique identifier.
  - Thus the processor uses the vector to identify the requesting module.
  - It is also known as **vectored interrupt**.
- **Bus arbitration**:
  - In this technique, an I/O module must first gain the control of the bus before it raises the interrupt request line.
  - Thus, only one module can request for interrupt at any time.
  - So, when the processor detects an interrupt request it responds on the acknowledge line. Then the requesting module places the vector on the data lines.

Interrupt

The second issue (if multiple interrupt occurs, how does the processor determine which one to process?) can be handled by the following way:

- Assigning of priorities when more than one device requests for interrupt.
- With multiple lines, the processor just picks the interrupt line with the highest priority.
- With software polling, the order in which modules are polled determine their priority.
- With daisy chain, the order of modules determine their priority.
- Bus arbitration can employ a priority scheme.

- **Types of interrupt**:
  - **Hardware interrupt**: caused by external devices and I/O devices. For example, pressing a key in the keyboard to do some action. This pressing of key in the keyboard will generate a signal which is given to the processor to do action, such interrupts are called hardware interrupts.
    - **Maskable interrupt**: the hardware interrupt that can be delayed when a highest priority interrupt has occurred.
    - **Non-maskable interrupt**: can not be delayed and immediately by serviced by the processor.
  - **Software interrupt**: caused by internal devices and software programs. For example, system calls.
    - **Normal interrupt**: caused by software instructions.
    - **Exception**: unplanned interrupt while executing a program. For example division by zero.

Direct Memory Access (DMA)

- Interrupt driven, though more efficient than programmed I/O, still requires the active intervention of the processor to transfer data between memory and an I/O module and any data traverse a path through the processor. So,
  - The 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.
- When large volume of data are to be moved, a more efficient way is required; direct memory access.

Direct Memory Access (DMA)

- **Direct memory access (DMA)** is a method/hardware subsystem that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.
- The process is managed by a chip known as a **DMA controller (DMAC)**.
- Certain lines on the system bus are used for the DMA channels.
- Working steps of DMA:
  - When the processor wishes to read or write a block of data, it issues a command to the DMA module using the read or write control line.
  - The address of the I/O module communicated on the data line.
  - The starting memory location to read from/write to is stored by the DMA module in the address register.
  - Then the DMA module transfers the entire block of data, one word at a time, directly to or from the memory, without going through the processor.
  - When the transfer is complete, the DMA module sends an interrupt signal to the processor.

Direct Memory Access (DMA)

- With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. The data block can be transferred to and from memory by the DMAC in three ways.
  - In **burst mode**, the system bus is released only after the data transfer is completed.
  - In **cycle stealing mode**, during the transfer of data between the DMA channel and I/O device, the system bus is relinquished for a few clock cycles so that the CPU can perform other tasks. When the data transfer is complete, the CPU receives an interrupt request from the DMA controller.
  - In **transparent mode**, the DMAC can take charge of the system bus only when it is not required by the processor.

Direct Memory Access (DMA)

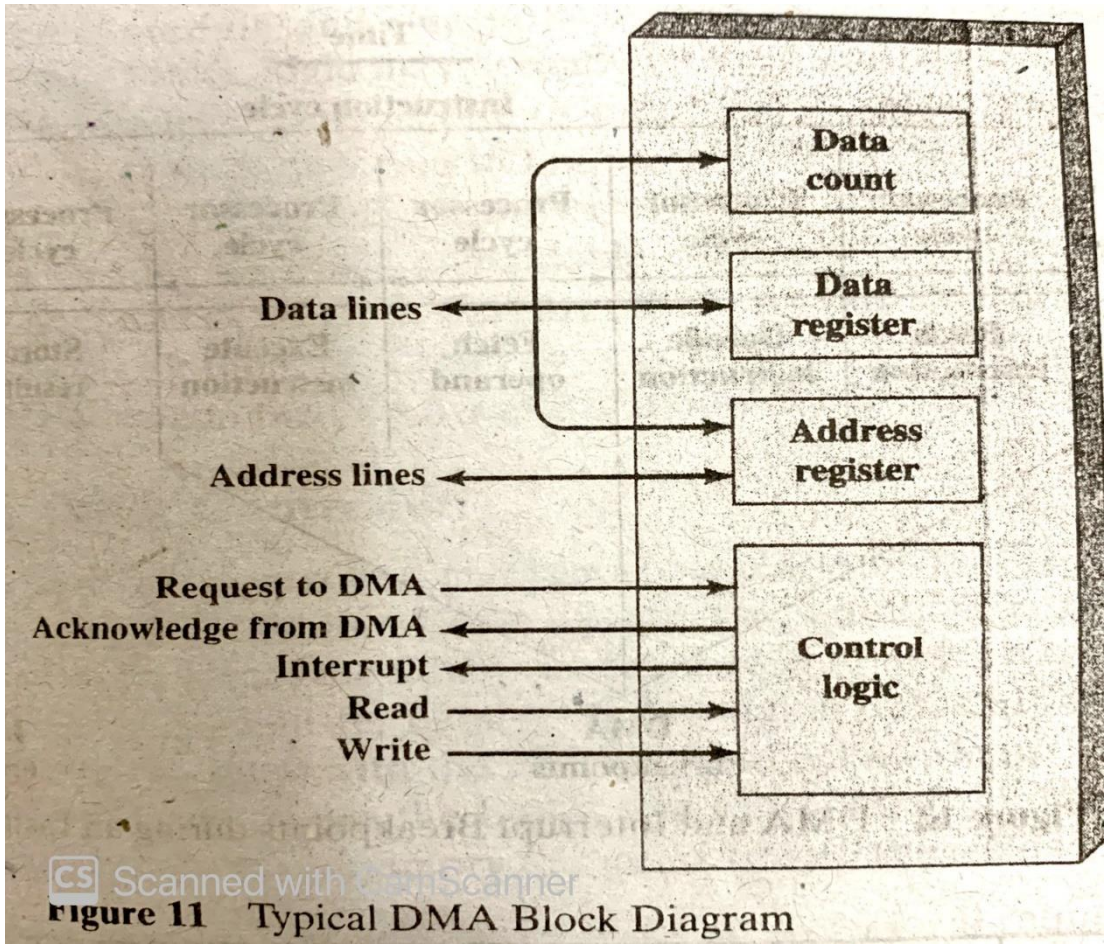

Figure 11 Typical DMA Block Diagram

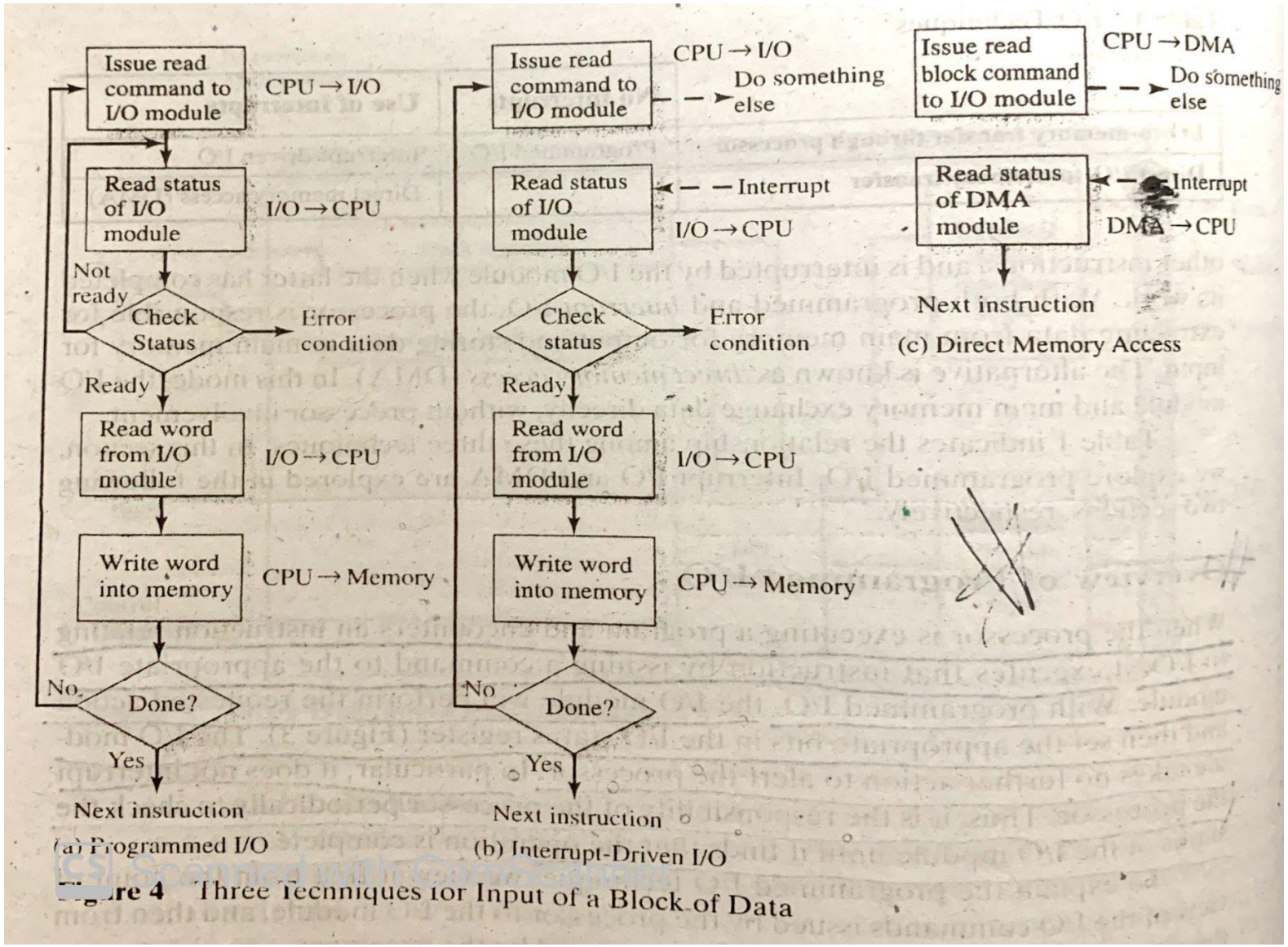

Figure 4 Three Techniques for Input of a Block of Data

Buss Arbitration

- The device that is allowed to initiate data transfers on the bus at any given time is called the **bus master**.
- When the current bus master relinquishes control of the bus, another device can acquire this status. **Bus arbitration** is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.
- There are two approaches to bus arbitration:
  - **Centralized arbitration**:
    - A single bus arbiter performs the required arbitration. The bus arbiter may be the processor or a separate unit connected to the bus.
    - The processor is normally the bus master unless it grants mastership to one of the DMA controllers.
    - A DMA controller indicates that it needs to become the bus master by activating the bus request line.

See Zaky (Chapter 4: Input/Output Organization)

Buss Arbitration

- **Distributed arbitration**:
  - All devices participate in the selection of the next bus master.
  - All the devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.
  - Each device on the bus is assigned a 4 bit identification number.
  - When one or more devices request the bus, they cause the start-arbitration signal and place the identification numbers on the 4 open-collectors lines.
  - The code on the four lines represents the request that has the highest ID number.