Interrupt-Driven I/O

- Principle of Interrupt-Driven I/O
- Multiple-Interrupt Systems
- Priority Interrupt Systems
    - Parallel Priority Interrupts
    - Daisy-Chain Priority Interrupts

Priority Interrupt Systems (1)

- In case of simultaneous requests, a priority system is needed
- Establishing the priority of simultaneous interrupts can be done in software or in hardware
- Software method:
  - Identification of the highest-priority source is made by polling
  - There is a common service routine, which polls the interrupt sources

Priority Interrupt Systems (2)

  - The order in which the sources are polled determines their priority
  - Disadvantage: if there are many sources, the time required for polling increases
- Hardware method:
  - An interrupt controller accepts interrupt requests from many sources and determines the highest priority request
  - Each source has its own interrupt vector

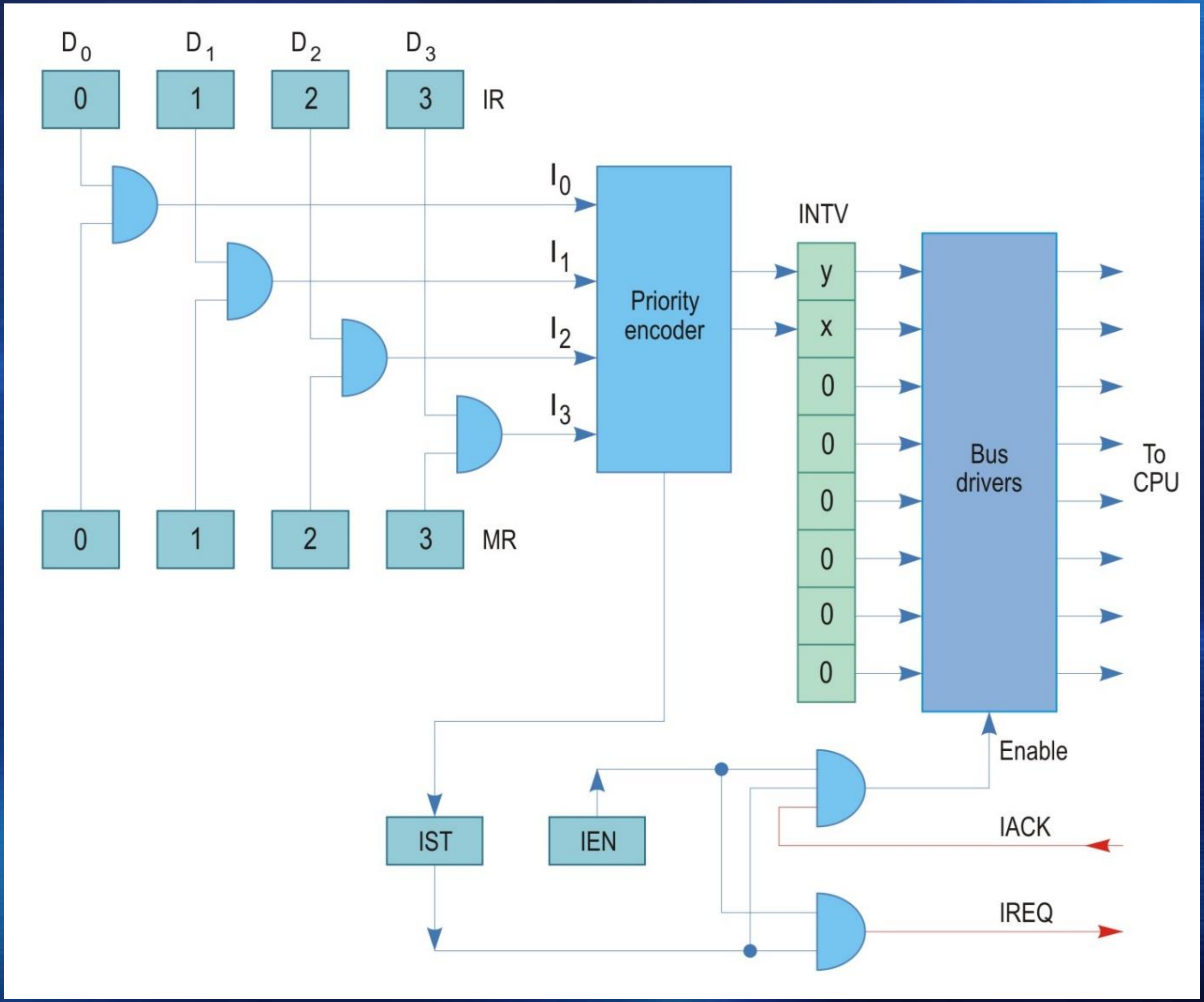
Interrupt-Driven I/O

- Principle of Interrupt-Driven I/O
- Multiple-Interrupt Systems
- Priority Interrupt Systems
  - Parallel Priority Interrupts
  - Daisy-Chain Priority Interrupts

Parallel Priority Interrupts (1)

- An interrupt request register IR is used
  - Its bits are set separately by the interrupt requests of each device
- Priority is established according to the position of bits in the register
- The interrupt mask register MR allows to control (disable) the status of each interrupt request

Parallel Priority Interrupts (2)



Parallel Priority Interrupts (3)

- The priority encoder:
  - Implements the priority function
  - Generates two bits of the interrupt vector
- The vector is transferred to the CPU via tristate buffers
- Enabling the buffers: with the *IACK* signal from the CPU, and the IST, IEN flags
  - IST – interrupt status flag
  - IEN – interrupt enable flag

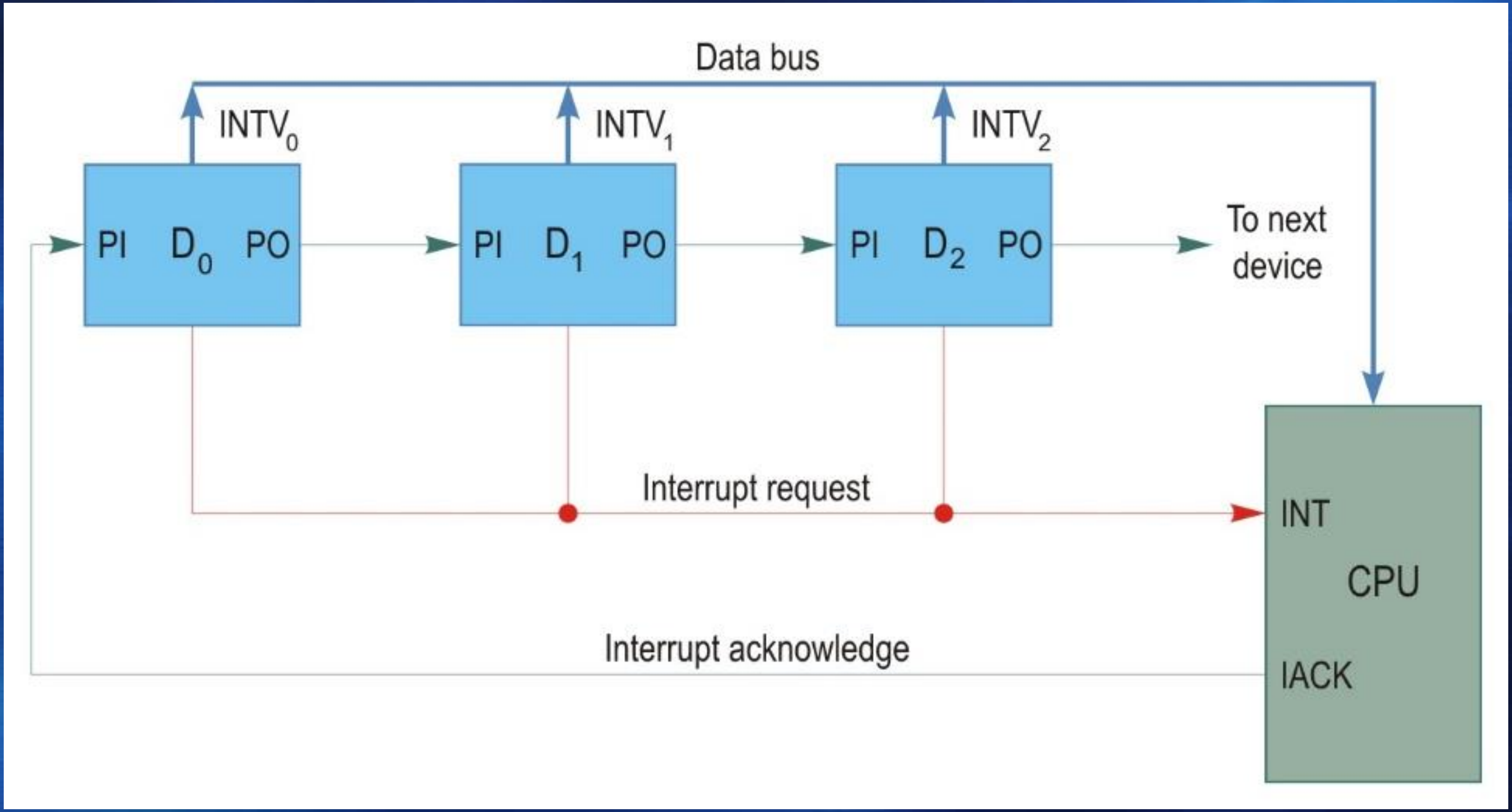
Interrupt-Driven I/O

- Principle of Interrupt-Driven I/O
- Multiple-Interrupt Systems
- Priority Interrupt Systems
    - Parallel Priority Interrupts
    - Daisy-Chain Priority Interrupts

Daisy-Chain Priority Interrupts (1)

- All devices that can generate an interrupt request are connected in a daisy-chain
  - Each device has a PI (*Priority In*) input and a PO (*Priority Out*) output
  - The device with the highest priority is placed in the first position
- The interrupt request line is shared by all devices (wired OR connection)

Daisy-Chain Priority Interrupts (2)



2. Methods for I/O Operations

- Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)
- I/O Processors

Direct Memory Access (DMA)

- Principle of I/O through DMA
- Execution of DMA Transfers
- Configurations of Systems Using DMA Transfers

Principle of I/O through DMA (1)

- Disadvantage of programmed I/O and interrupt-driven I/O: the CPU is busy with managing the I/O operations
- DMA eliminates this disadvantage → data transfers are executed directly between the internal memory and the I/O system
- An additional module is required → DMA controller
- Two methods for performing transfers through DMA →

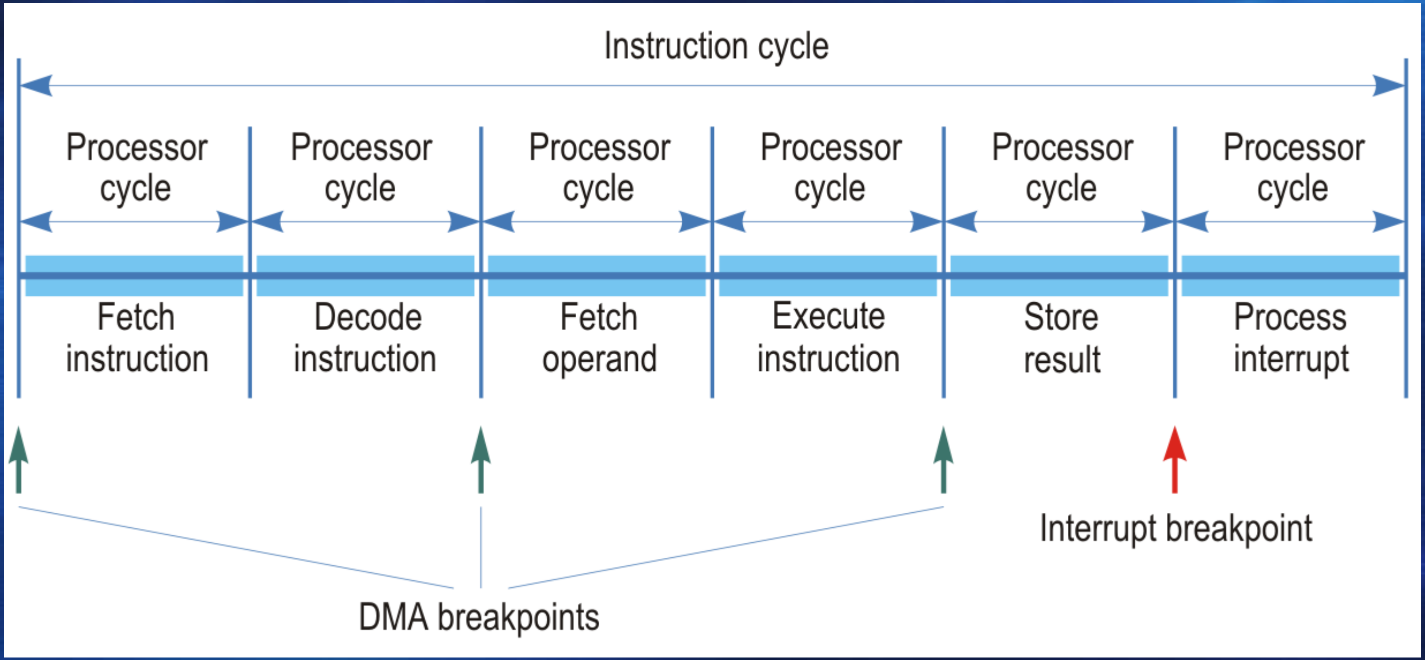
Principle of I/O through DMA (2)

1. By suspending the CPU operations and placing the bus in the high-impedance state during the transfer
   - Data break or block transfer
   - This method is required, e.g., for magnetic disk drives $\rightarrow$ data transmission cannot be stopped or slowed down
   - The CPU is inactive for relatively long time periods

Principle of I/O through DMA (3)

2. By using the time intervals when the CPU does not access the memory
   - Cycle stealing
   - Large blocks of data are transferred by a sequence of DMA bus transactions interspersed with CPU bus transactions
   - The method reduces the maximum transfer rate, but it also reduces the interference of the DMA controller in accessing memory by the CPU

Principle of I/O through DMA (4)



Breakpoints of the CPU for performing transfers through DMA and through interrupts

Direct Memory Access (DMA)

- Principle of I/O through DMA
- Execution of DMA Transfers
- Configurations of Systems Using DMA Transfers

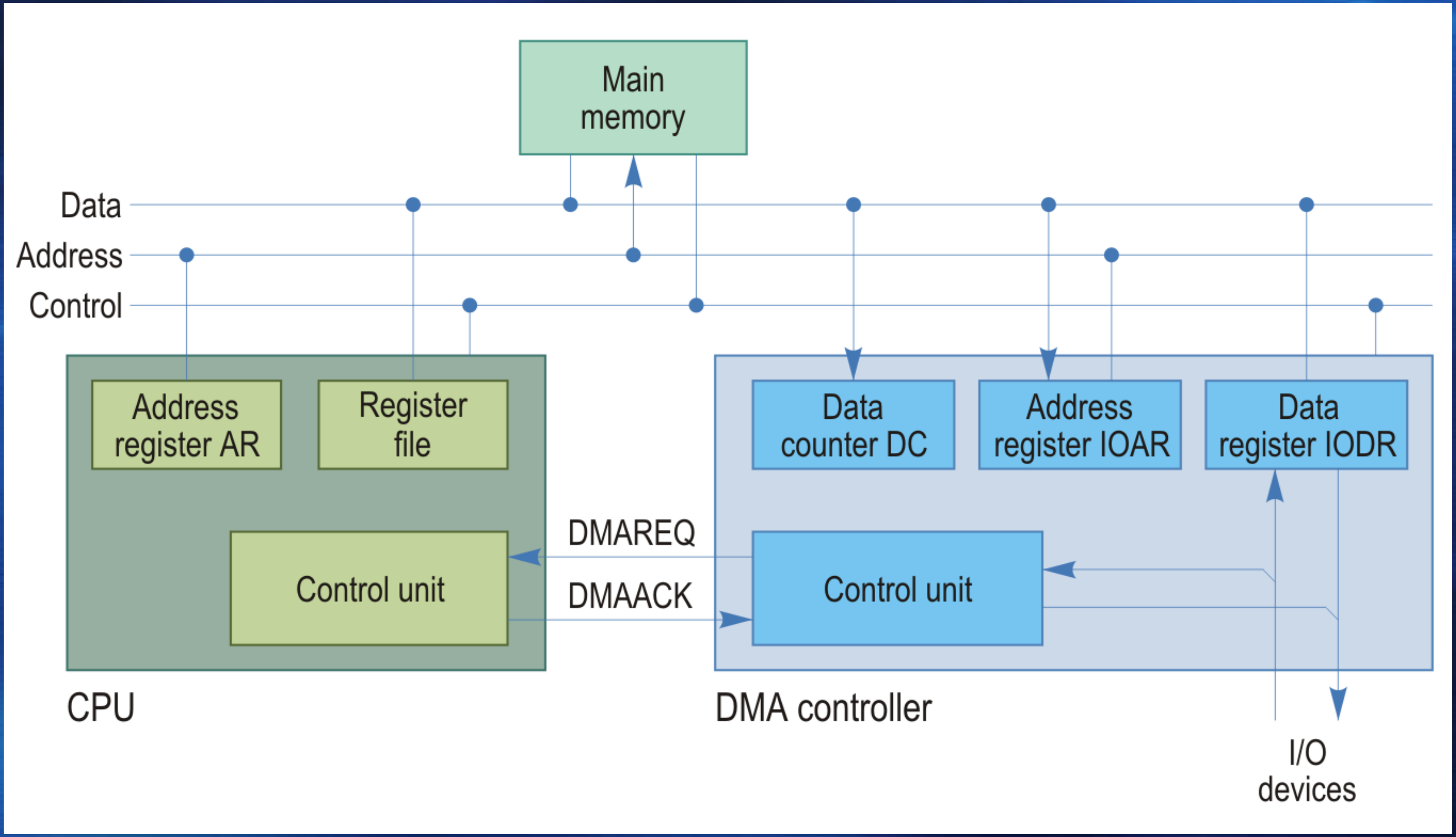
Execution of DMA Transfers (1)

- The CPU sends an initialization sequence to the DMA controller
- The initialization sequence contains:
  - Direction of transfer (read or write)
  - Address of the I/O device involved
  - Starting address of the memory area used for the transfer
  - Number of bytes or words to be transferred

Execution of DMA Transfers (2)

- The CPU releases the bus and may execute other operations
- The DMA controller generates the addresses and control signals needed for the transfer
- After a DMA cycle, other cycles may follow, or the control is transferred to the CPU
- When the transfer is complete, the DMA controller generates an interrupt request to the CPU

Execution of DMA Transfers (3)



Execution of DMA Transfers (4)

1. The CPU loads the IOAR and DC registers with initial values $\rightarrow$ I/O instructions
2. When the DMA controller is ready for the transfer, it asserts the *DMAREQ* signal
   - At the next DMA breakpoint, the CPU releases the bus and asserts *DMAACK*
3. The DMA controller transfers data directly with the main memory; the IOAR and DC registers are updated

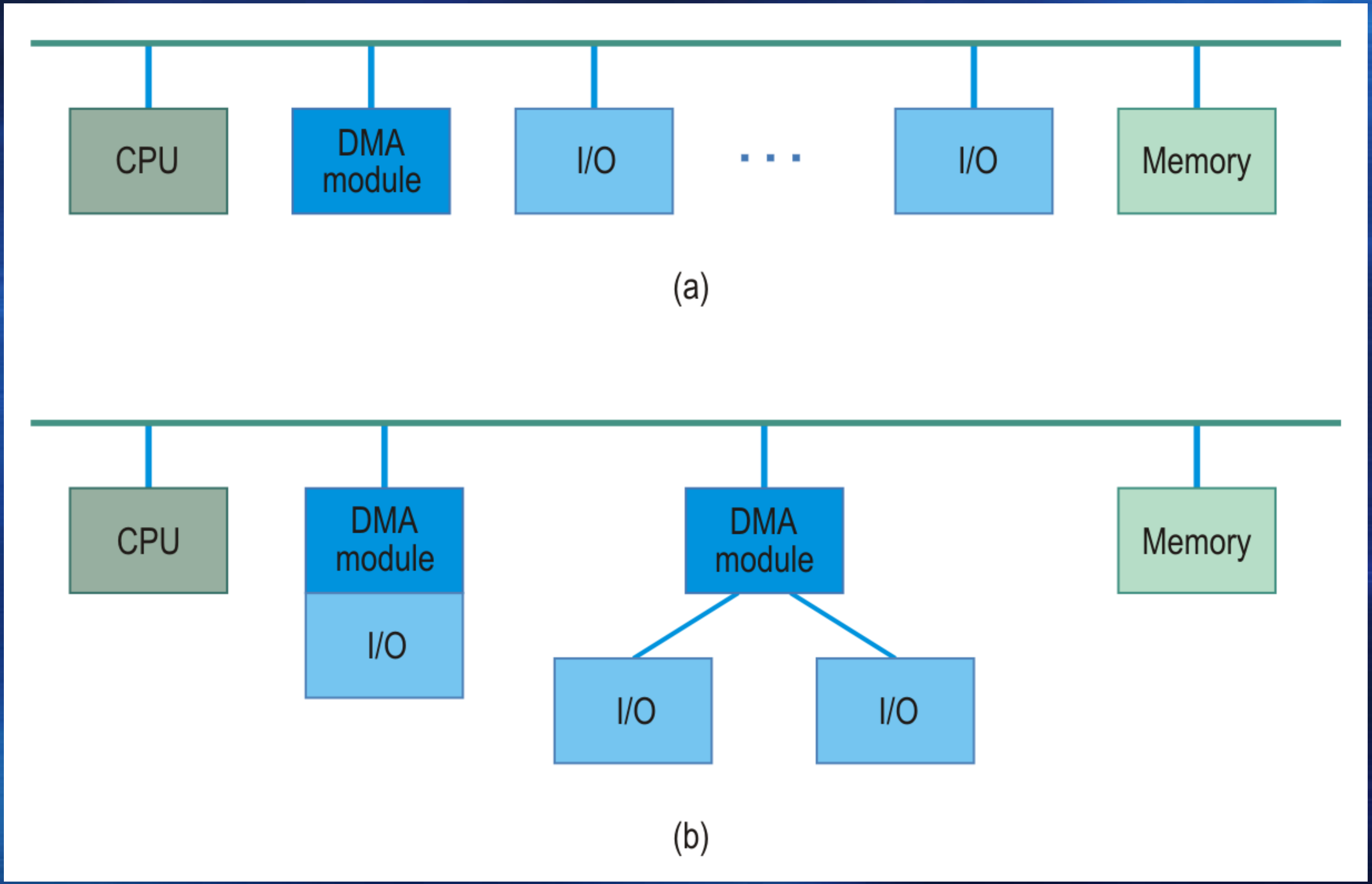
Execution of DMA Transfers (5)

4. If the DC register $\neq$ 0, but the I/O device is not ready, the DMA controller releases the bus
   - The CPU disables the *DMAACK* signal and takes control of the bus
5. If the DC register = 0, the DMA controller releases the bus and sends an interrupt request to the CPU
   - The CPU responds by halting the I/O device or by initiating a new transfer

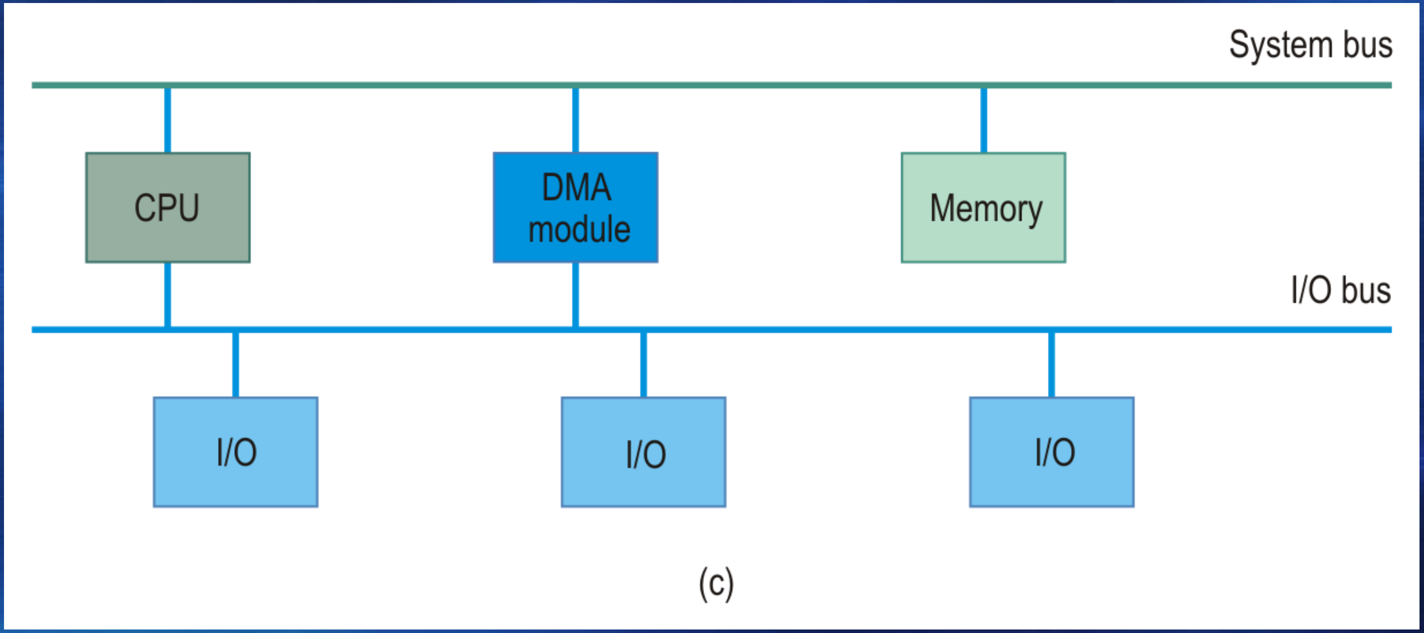
Direct Memory Access (DMA)

- Principle of I/O through DMA
- Execution of DMA Transfers
- Configurations of Systems Using DMA Transfers

Configurations of Systems Using DMA Transfers (1)



Configurations of Systems Using DMA Transfers (2)



2. Methods for I/O Operations

- Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)
- I/O Processors

I/O Processors

- Principle of I/O through I/O Processors
- Execution of an I/O Program

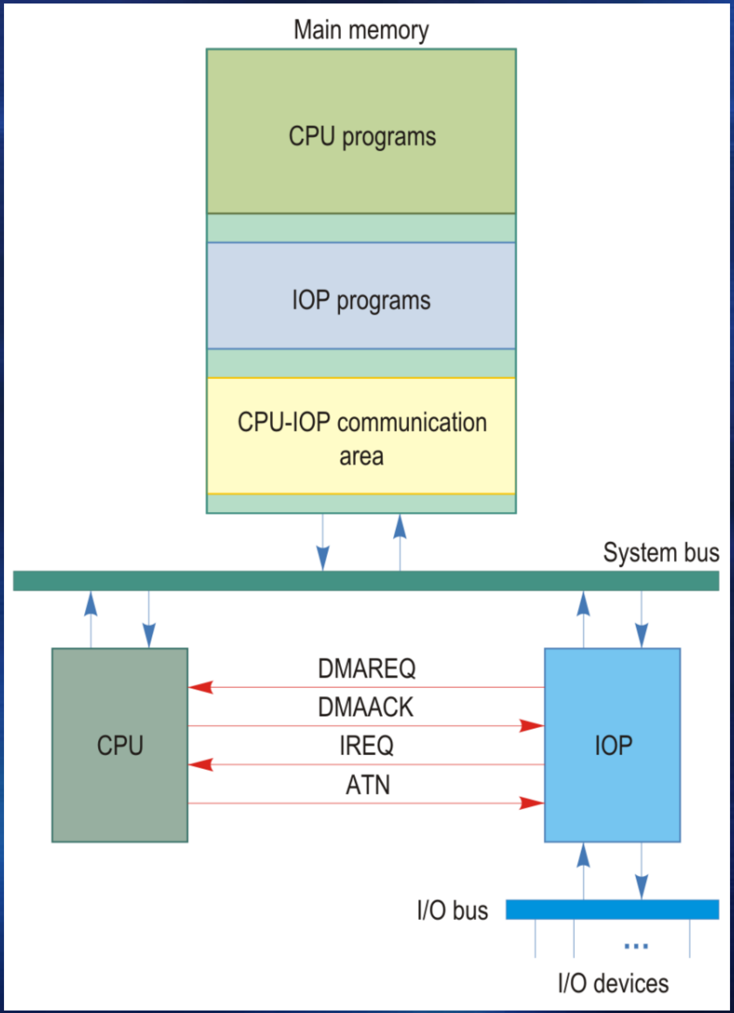
Principle of I/O through I/O Processors (1)

- While DMA releases the CPU from many I/O operations, for high-speed peripherals numerous bus cycles will be needed
  - During these cycles, the CPU enters a wait state
  - The cycle stealing will saturate the bus
  - A certain time is required to service the interrupts
- The I/O modules have been improved, becoming *I/O processors* (IOPs)

Principle of I/O through I/O Processors (2)

- Some of these I/O modules are also called *I/O channels*
- IOPs have a specialized instruction set for I/O operations
- The CPU sends a command to the IOP to execute an *I/O program* (*channel program*) located in memory
- The CPU can specify a sequence of I/O operations, and is interrupted only at the completion of the entire sequence

Principle of I/O through I/O Processors (3)



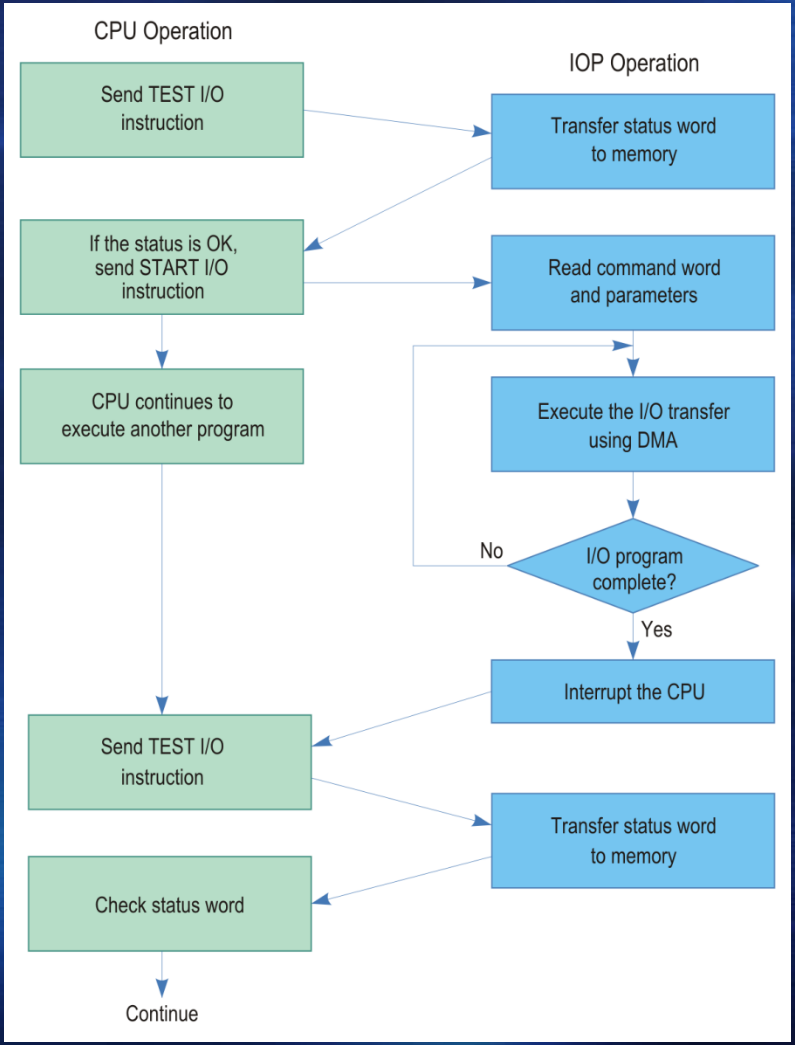
Principle of I/O through I/O Processors (4)

- The CPU and IOP can also communicate with each other directly via control lines
  - DMA Request (*DMAREQ*)
  - DMA Acknowledge (*DMAACK*)
  - The CPU may attention the IOP by asserting the *ATN* (*Attention*) signal $\rightarrow$ execution of an I/O program
  - The IOP may attention the CPU by asserting the *IREQ* signal $\rightarrow$ execution of an interrupt service routine

I/O Processors

- Principle of I/O through I/O Processors
- Execution of an I/O Program

Execution of an I/O Program


CPU Operation
IOP Operation
Send TEST I/O instruction
Transfer status word to memory
If the status is OK, send START I/O instruction
Read command word and parameters
CPU continues to execute another program
Execute the I/O transfer using DMA
No
I/O program complete?
Yes
Interrupt the CPU
Send TEST I/O instruction
Transfer status word to memory
Check status word
Continue

Summary (1)

- Establishing the priority of interrupts can be done with two methods:
  - Software method: polling in a common interrupt handler
  - Hardware method: using an interrupt controller
- The DMA technique allows performing I/O transfers without CPU intervention
  - Two methods for performing DMA transfers: data break (block transfer) or cycle stealing

Summary (2)

- An I/O processor (IOP) has specialized instructions for I/O operations
  - Can execute a sequence of I/O operations without interrupting the CPU
  - The CPU and an IOP communicate via a memory area and via control signals

Concepts, Knowledge (1)

- Software polling technique
- Hardware polling technique
- Methods for establishing the priority of simultaneous interrupts: software method; hardware method
- Parallel connection of interrupt lines
- Daisy-chain connection of interrupt lines
- Principle of I/O through DMA

Concepts, Knowledge (2)

- Data-break DMA transfer method
- Cycle-stealing DMA transfer method
- Execution of DMA transfers
- Diagram of circuitry required for DMA transfers
- Principle of I/O through IOP
- Structure of a computer with IOP
- CPU-IOP communication
- Operations for execution of an I/O program