ICT 409

04 January 2020

IO Organization

Md. Sharif Hossen, Assistant Professor, Dept. of ICT, Comilla University

I/O Interface

- The method that is used to transfer information between internal storage and external I/O devices is known as I/O interface.
- The CPU is interfaced using special communication links by the peripherals connected to any computer system. These communication links are used to resolve the differences between CPU and peripheral. There exists special hardware components between CPU and peripherals to supervise and synchronize all the input and output transfers that are called interface units.

Mode of Transfer

- Binary **information** received from an external device is usually **stored in memory** for later processing
- The CPU executes the I/O instruction and may accept the data temporarily but ultimate source and destination is the memory
- Data transfer between the **central computer (CPU) and I/O devices** may be handled in a variety of **modes**
- Some modes use the **CPU as an intermediate path**; others transfer the **data directly to and from the memory unit**

Mode of Transfer (Contd.)

- Data transfer to and from peripherals may be handled in one of three possible modes
  - Programmed I/O
  - Interrupt-initiated I/O
  - Direct Memory Access (DMA)

Programmed I/O

- Programmed I/O operations are the result of I/O instructions written in the computer program
- Each data item transfer is initiated by an instruction in the program
- Usually, the transfer is from a CPU register and memory.
- It requires constant monitoring by the CPU of the peripheral devices.

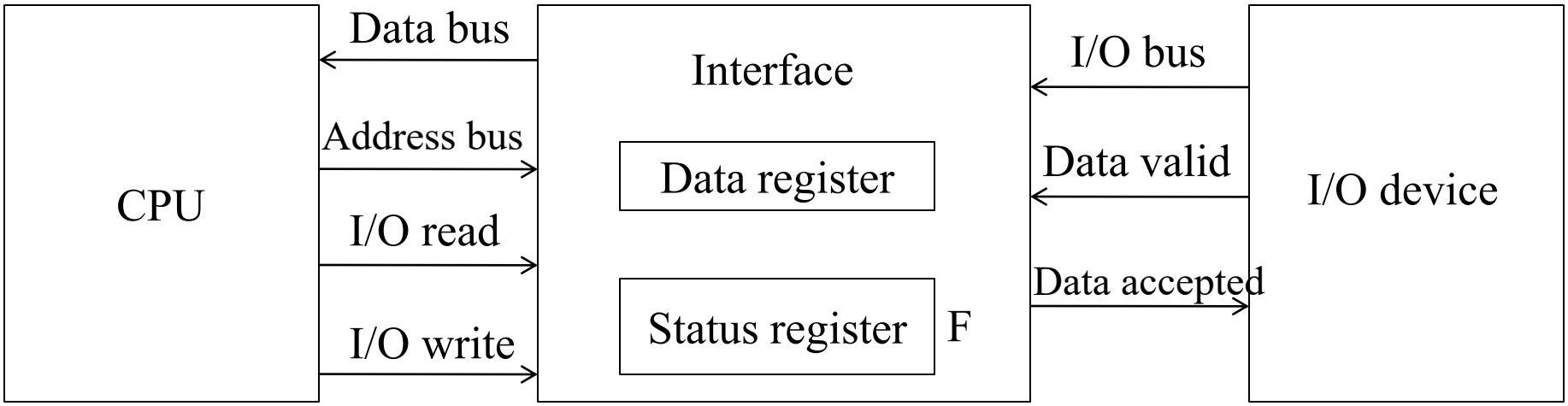
Example of Programmed I/O

loop is a sequence of instructions that is continually repeated until a certain condition is reached

- In the programmed I/O method
  - The I/O device does not have direct access to memory
- A transfer from an I/O device to memory requires the execution of several instructions by the CPU
  - including an input instruction to transfer the data **from the device to the CPU** and a store instruction to transfer the data **from the CPU to memory**
- In the programmed I/O method
  - The CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is a time consuming process since it needlessly keeps the CPU busy. This situation can be avoided by using an interrupt facility.

Example of Programmed I/O

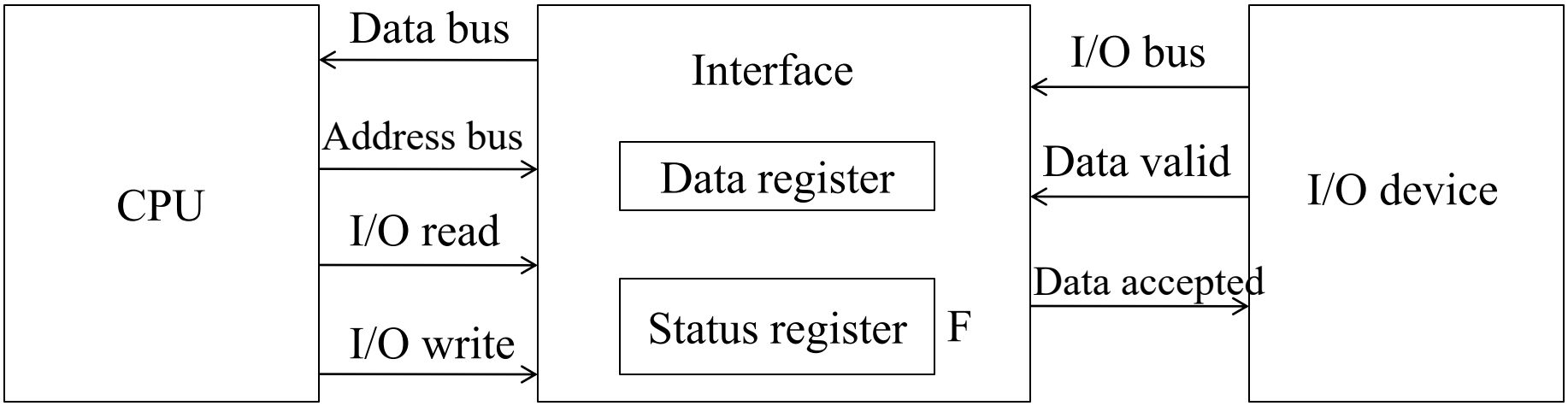
Example of data transfer from an I/O device through an interface into the CPU



- The device **transfers byte of data** one at a time as they are **available**
- When a byte of data is available, the device **places** it in the **I/O bus** and **enables** its **data valid line**
- The interface **accepts the byte into its data register** and **enables** the **accepted line**

Example of Programmed I/O (Contd.)

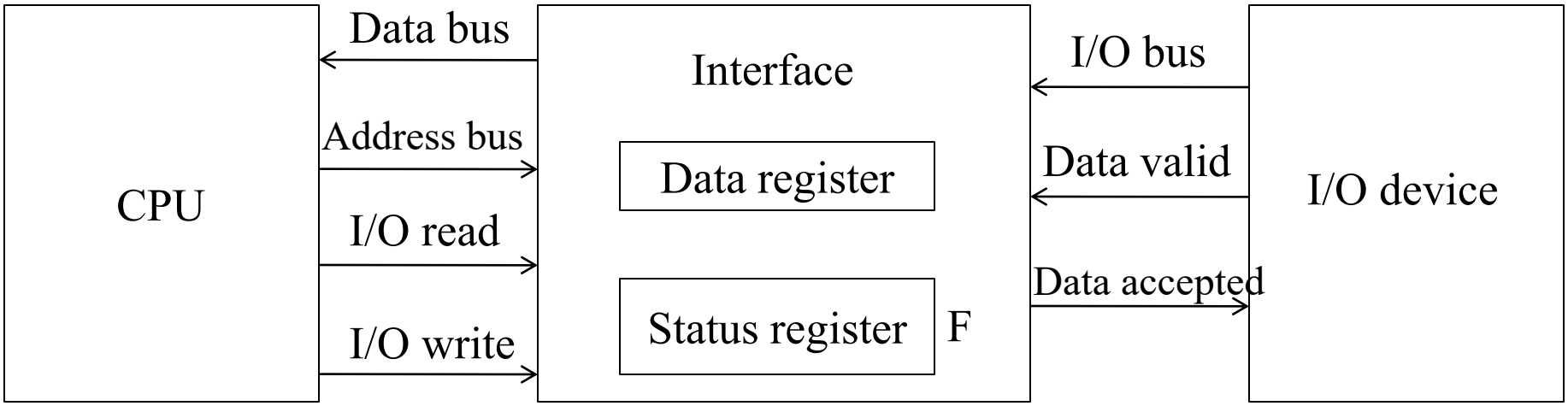
Example of data transfer from an I/O device through an interface into the CPU



- The device can now **disable** the **data valid line**
- It will **not transfer** another byte until the **data accepted line is disabled** by the interface

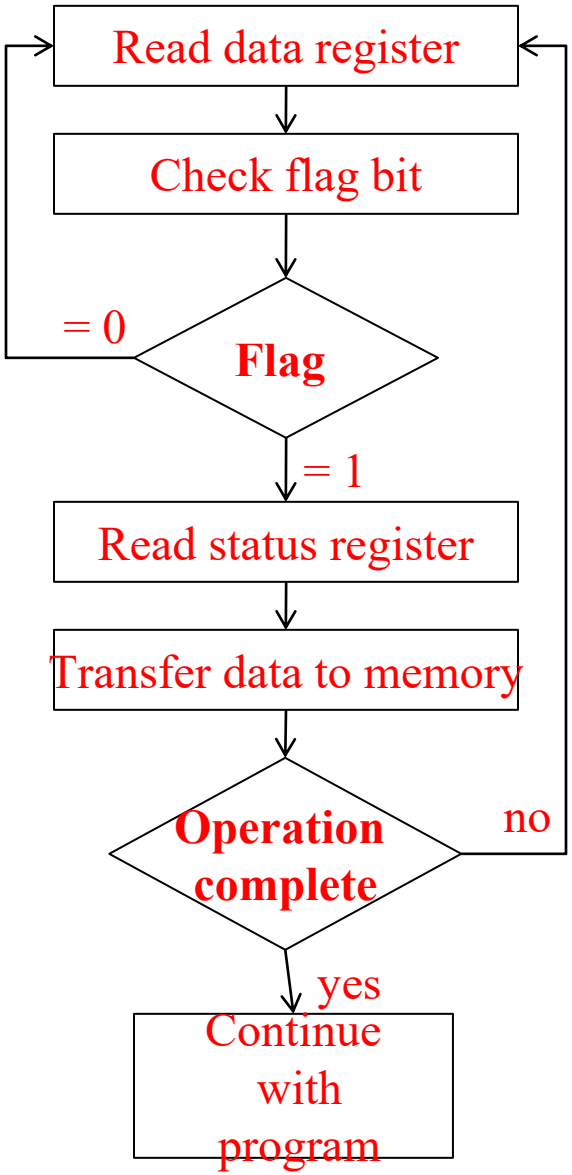
Example of Programmed I/O (Contd.)

Example of data transfer from an I/O device through an interface into the CPU



- The interface set a bit into the **status register** that we will refer to as an ***F* or flag bit**
- If the **flag is 1**, the CPU **reads the data** from the data register
- Once the **flag is cleared (0)**, the **interface disables the data accepted** line and the **device can then transfer** the data byte

Example of Programmed I/O (Contd.)



Interrupt Initiated I/O

- An interrupt is a signal sent to the processor that interrupts the current process.
- When the interface determines that the device is ready for data transfer
  - It generates an interrupts request to the computer
- Upon detecting the external interrupt signal
  - The CPU momentarily stops the task it is processing
  - Branches to a service program to process the I/O transfer
  - Then, return to the task it was originally performing

Interrupt Initiated I/O

- The word interrupt is used in a broad sense for any infrequent or exceptional event
- While the CPU is running a program, it does not check the flag
- However, when the **flag is set**, the computer is momentarily **interrupted from** proceeding with the **current program**
- The CPU responds to the interrupt signal by storing the return address from the **program counter** into a memory stack
- Then control branches to a service routine that processes the required I/O transfer

Interrupt Initiated I/O (Contd.)

- The way that the processor chooses the branch address of the service routine varies from one unit to another
- In principle, there are two methods for accomplishing this
  - Vectored interrupt
    - In vectored interrupt, processor **automatically generates the new address in** Program Counter (PC)
  - Non-vectored interrupt
    - In this interrupt, the **branch address is assigned to a fixed location** in memory

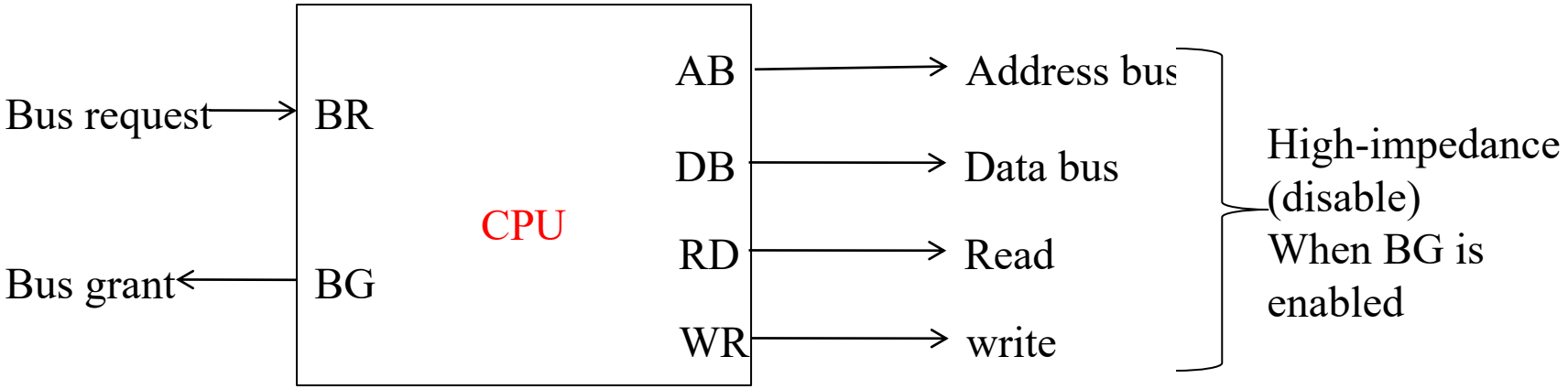
Direct Memory Access

- The transfer of data between a fast storage device and memory is often limited by the speed of the CPU
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly
- Then it would improve the speed of transfer
- The transfer technique is called direct memory access (DMA)

Direct Memory Access (Contd.)

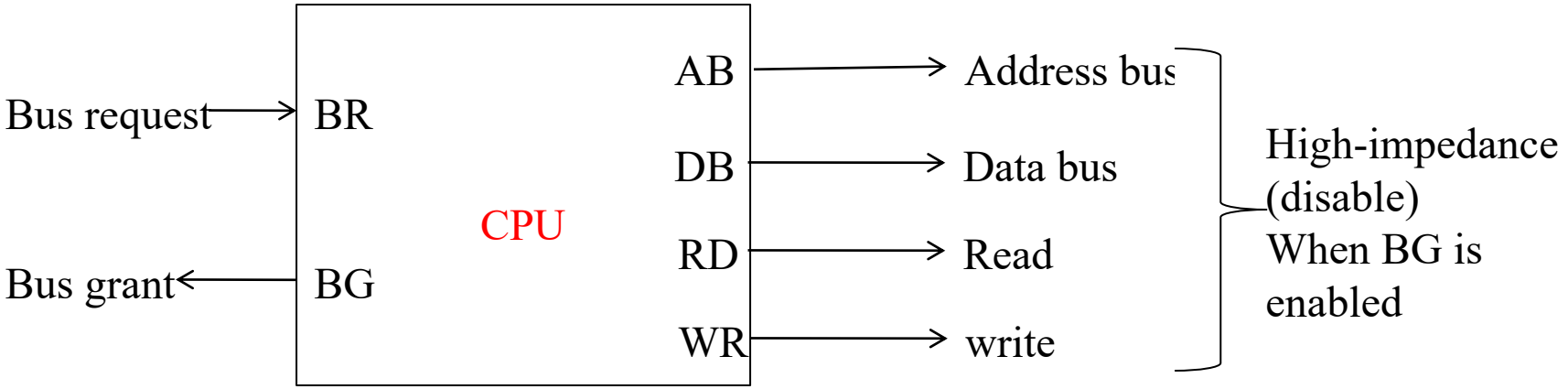
- During DMA transfer, the CPU is idle and has no control of the memory buses
- A DMA controller **takes over the buses** to manage the **transfer directly between the I/O device and memory**
- The **CPU** may be placed in an **idle state** in variety of ways
  - One common method is **special control signals**
- **Two control signals** in the CPU that facilitate the DMA controller
  - Bus request (BR)
  - Bus grant (BG)

Direct Memory Access (Contd.)



- Bus request
  - It is used by the **DMA** controller to **request** the CPU to **relinquish control** of the bus
  - When it is **active**, the CPU **terminates** the execution of the **current instruction**
  - The CPU places the address bus, the data bus, read, write lines into a **high-impedance state**
  - The high-impedance state behaves like an **open circuit**, which means that the output is disconnected

Direct Memory Access (Contd.)



- Bus grant
  - The CPU **actives the BG** to inform the external DMA that the buses are in the high-impedance state
  - The **DMA** that originated the bus request can now **take control of the buses**
  - When the DMA **terminates the transfer**, it **disables the *bus request*** line
  - The **CPU disables the bus grant**, takes **control** of the buses, and returns to its **normal operation**