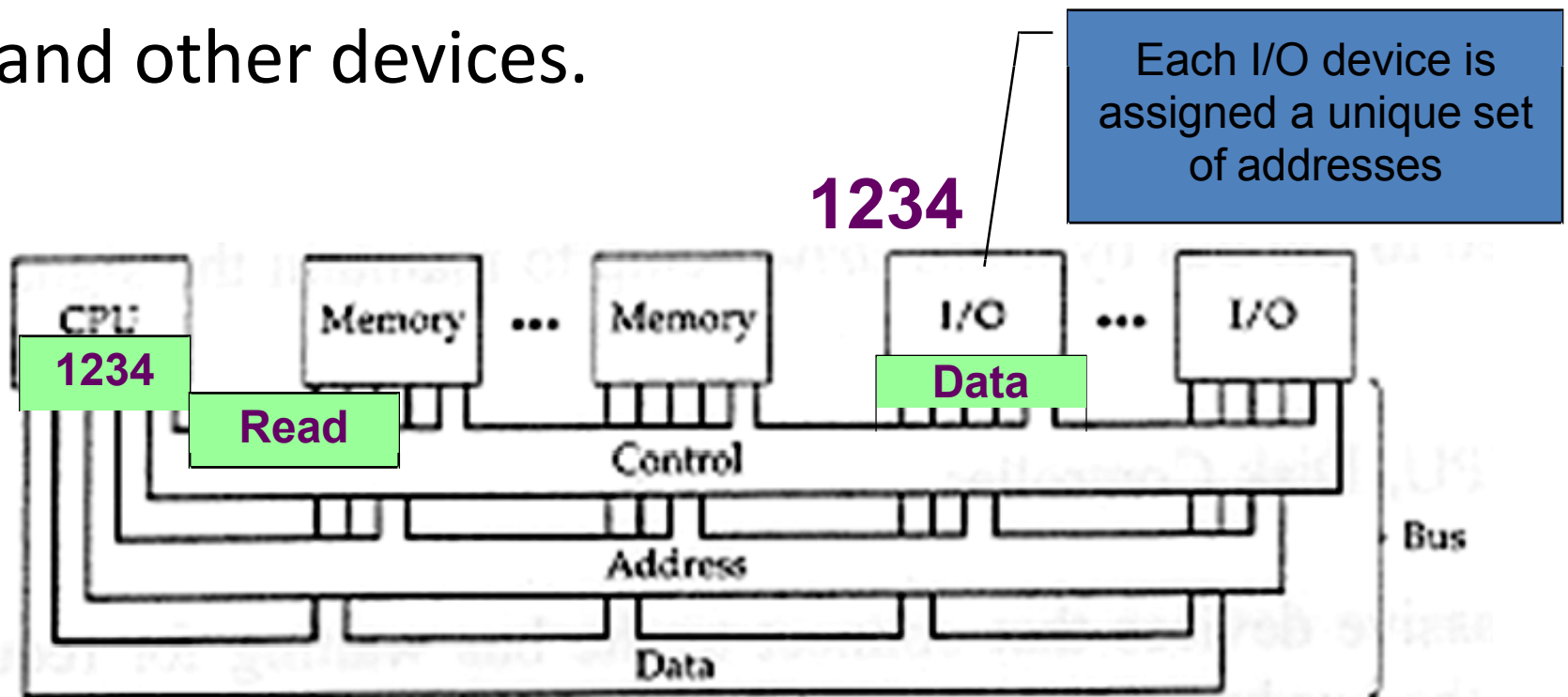


Contents

- Input / Output Operations
 - Program-controlled I/O
 - Interrupt-driven I/O
 - Direct memory access I/O
- Input / Output Interface
 - Parallel interface
 - Serial interface

Input / Output (I/O) Operations (1)

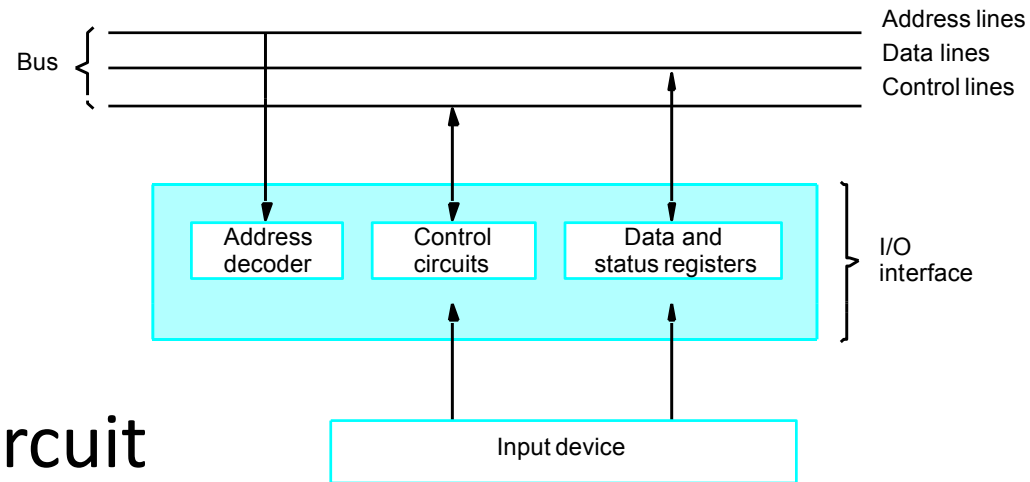
- Computers can perform I/O operations to exchange information with human operators and other devices.



I/O Operations (2)

- Usually, the CPU merely acts as an intermediate path when executing an I/O instruction but the actual data transfer is between memory unit and I/O devices.
- *Memory-mapped I/O*
 - I/O devices and the memory share the same address space
 - Any machine instruction that can access memory can be used to transfer data to or from an I/O device.

I/O Operations (3)



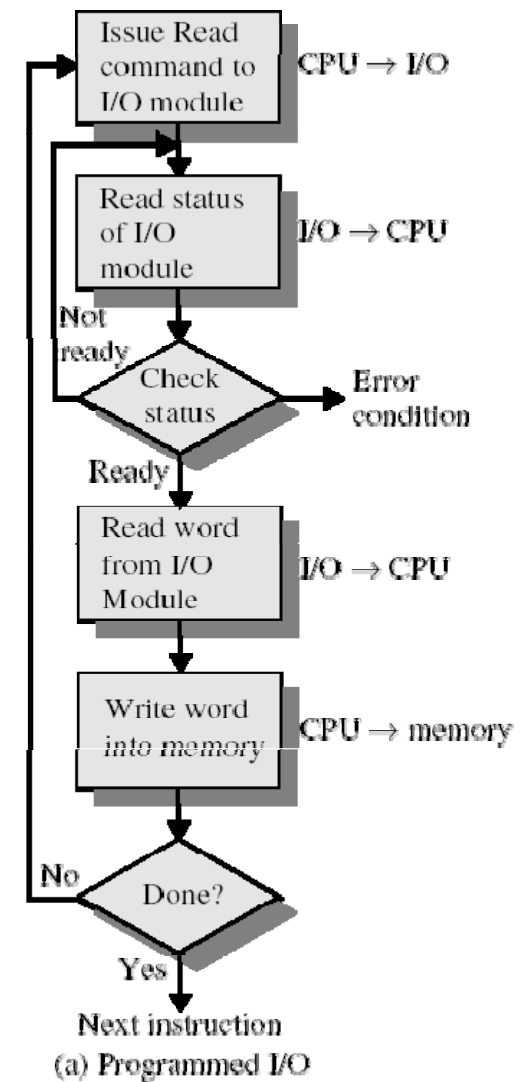
- I/O interface circuit
 - The address decoder enables the device to recognize its address.
 - The data register holds the data being transferred.
 - The status register contains information for the I/O operations.
 - The control circuits coordinate I/O transfers.

I/O Operations (4)

- The speeds of I/O devices are much slower than that of the processor, we must have some mechanisms to synchronize the transfer of data between them.
- Common used mechanisms for implementing I/O operations:
 - *Program-controlled I/O*
 - *Interrupt-driven I/O*
 - *DMA I/O*

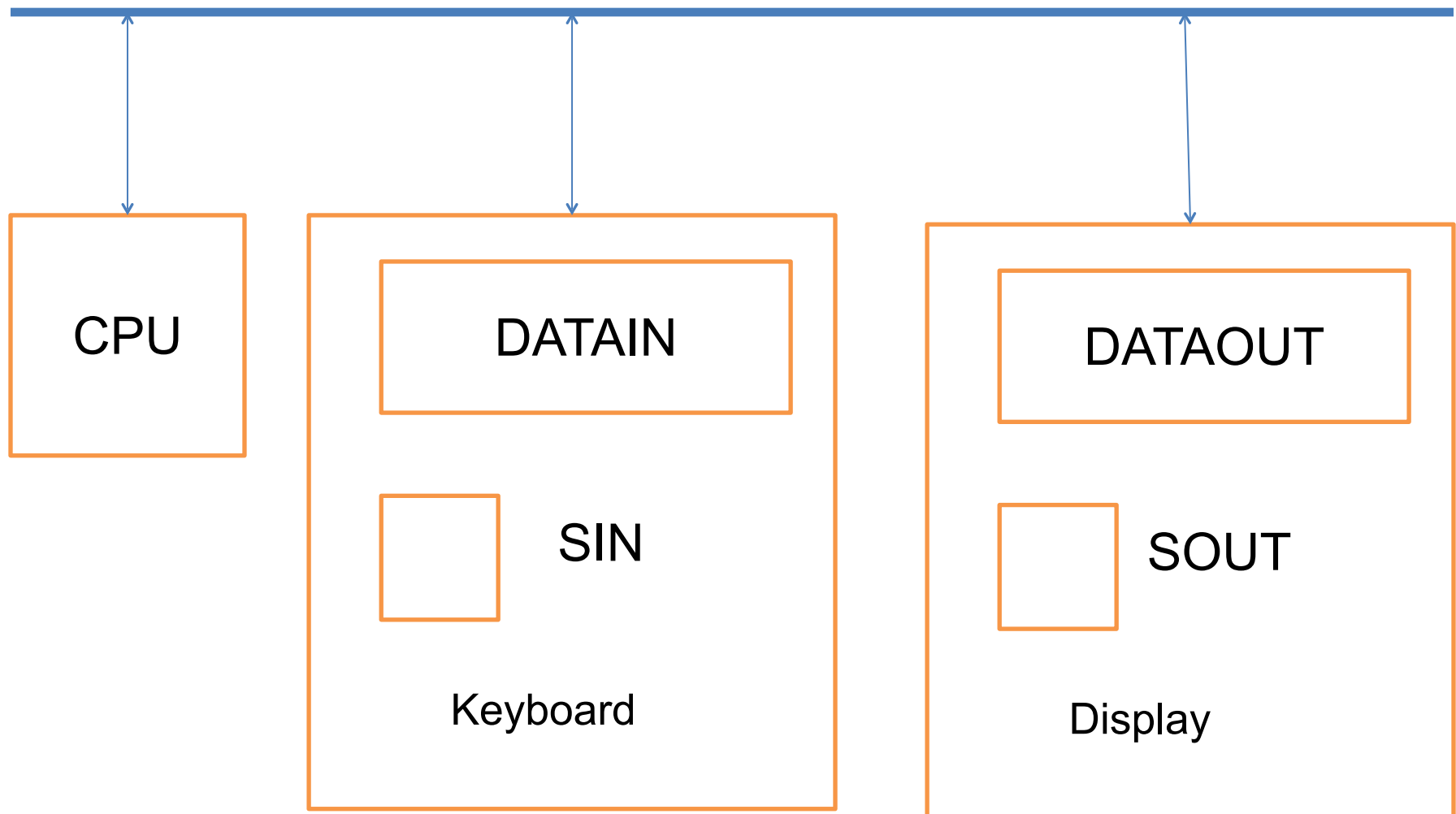
Program-controlled I/O (1)

- In *program-controlled I/O*, the CPU repeatedly checks (polls) the I/O device status register to achieve the required synchronization between the CPU and the I/O device.
- Commonly used in low-end microprocessors such as embedded systems or in real-time systems.
- Disadvantage – the CPU wastes most of its time busy waiting for I/O.



Program-controlled I/O (2)

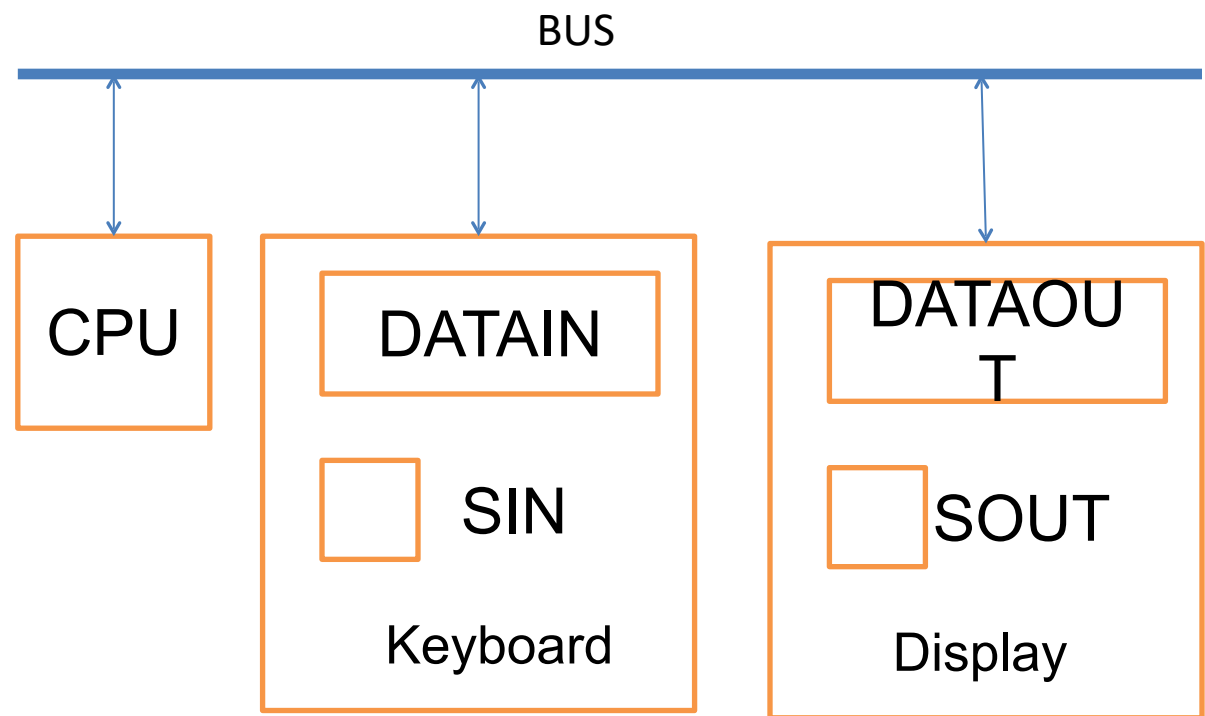
- A simple example: Read in character input from a keyboard and produce character output on a display screen.



Program-controlled I/O (3)

- Read in character input from a keyboard and produce character output on a display screen.
 - Rate of data transfer (keyboard, display, processor)
 - Difference in speed between processor and I/O device creates the need for mechanisms to synchronize the transfer of data.
 - A solution: on output, the processor sends the first character and then waits for a signal from the display that the character has been received. It then sends the second character. Input is sent from the keyboard in a similar way.

Program-controlled I/O (4)



- **DATAIN / DATAOUT**: 8-bit data registers
- **SIN**: a status control flag, **SIN=1** means a character is entered at the keyboard and is ready for processor to read.
- **SOUT**: a status control flag, **SOUT=1** means that the display is ready to receive a character.
- The data registers and the status registers (**INSTATUS** and **OUTSTATUS**) containing the status control flags in bit 3 can be addressed as if they were memory locations.

Program-controlled I/O (5)

An input operation from the keyboard.

1. Striking a key stores a character in DATAIN.
2. SIN is set to 1.
3. The processor can monitor SIN as follows.

```
READ    BTST.W    #3,INSTATUS    ; wait for a character to be entered in the keyboard buffer DATAIN
        BEQ      READ
        MOVE.B   DATAIN,D1      ; transfer the character from DATAIN into D1
```

[Motorola 68000 assembly code. Memory Mapped I/O. Bit Test Instruction. Testing Bit 3 for the variable INSTATUS. Z flag gets set if the bit 3 is zero.]

4. When SIN is set to 1, the processor transfers a character from DATAIN to D1.
5. SIN is cleared to 0.
6. If next character is entered at the keyboard, SIN is again set to 1.

Program-controlled I/O (6)

An output operation to the display

1. The processor monitors SOUT as follows.

```
WRITE  BTST.W  #3,OUTSTATUS    ; wait for the display to become ready
      BEQ      WRITE
      MOVE.B   D1,DATAOUT       ; move the character from D1 to the
                                ; output buffer DATAOUT
```

2. When SOUT is set to 1, the processor transfers a character from D1 to DATAOUT to be displayed.
3. SOUT is cleared to 0.
4. When the display device is ready to receive the next character, SOUT is again set to 1.

Interrupts (1)

- Interrupt is a mechanism for diverting the attention of a processor when a particular event occurs, such as I/O device requests.
- Interrupts cause a break in the normal execution of a program.
- Generally, three Types of Interrupt Sources:
 - External Interrupts
 - Internal Interrupts Situations
 - Software Interrupts

Interrupts (2)

- (i) Situation for External Interrupts
 - I/O devices request transfer of data
 - a timing device indicates an elapsed time of event or exceeded time allocation for an endless looping of a program
 - external hardware circuits such as power supply failure, system reset or bus error
 - asynchronous in operation between devices
 - event driven, independent of the program in execution

Interrupts (3)

- (ii) Internal Interrupts Situations(*Traps*)
 - arise from illegal or erroneous use of an instruction or data
 - examples: divide by zero, stack overflow
 - the interrupt service program (*trap handler*) determines the corrective action to be taken, e.g., print an error message
 - internal interrupts are initiated by some exceptional conditions caused by program
 - synchronous in operation with the program
 - if re-run program, event will occur in the same place

Interrupts (4)

- (iii) Software Interrupts
 - initiated by executing a *privileged instruction* as a special call instruction (behaves as an interrupt)
 - programmers often use software interrupts to initiate an interrupt procedure at a desired point in the program
 - e.g., a switch from the user mode (user program execution) to supervisor mode for input or output transfer operation by means of a *supervisor call instruction*. The user program passed data to the system program (supervisor mode) to specify the requested I/O task.

Steps taken to process Interrupt (1)

1. A device raises an interrupt request by sending a hardware signal in one of the bus control lines, called an *interrupt-request* line, to the processor.
2. The processor completes execution of the current instruction.
3. The processor *determines* the device requesting an interrupt and sends an interrupt-acknowledge signal to the device.
4. The device removes its interrupt-request signal.

Steps taken to process Interrupt (2)

5. The status of the processor (Status register) and the location of the next instruction to be executed (Program Counter) is saved on the processor stack.
 - The amount of information saved automatically by the processor should be kept to a minimum to avoid long *interrupt latency*.
6. The processor loads the PC with the address of the first instruction of the Interrupt Service Routine (ISR) associated with the device (causing the interrupt).

Steps taken to process Interrupt (3)

7. Process the ISR.

- Any additional information such as the contents of the registers must be saved at the beginning of the ISR and restored at the end of the ISR.

8. Restore the status register and program counter.

9. Resume the execution of the interrupted program.

Determining Source of Interrupt (1)

- If a number of devices capable of initiating interrupts are connected to the processor, how can the processor recognize the device requesting an interrupt?

Determining Source of Interrupt (2)

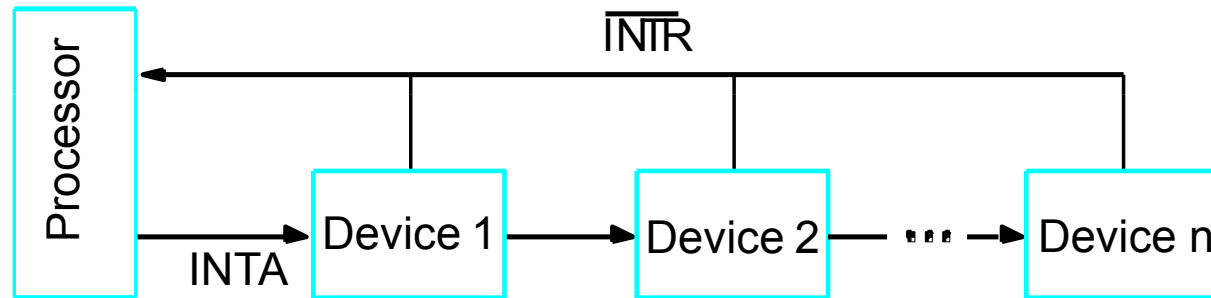
- (i) Polling
 - When a device raises an interrupt request, it sets an interrupt request bit (IRQ) in its status register to 1.
 - Whenever an interrupt is generated, all the I/O devices are polled.
 - The first device with its IRQ bit set is the device to be serviced.
 - The order in which the devices are polled determines the priorities of the devices.
 - Advantage: simple and easy to implement
 - Disadvantage: time is spent testing the IRQ bits of all the devices that may not be requesting any service.

Determining Source of Interrupt (3)

- (ii) Vectored interrupts
 - A device requesting an interrupt identifies itself by sending a vector to the processor upon the interrupt acknowledgement.
 - This vector is used as a pointer to the ISR for that device.

Determining Source of Interrupt (4)

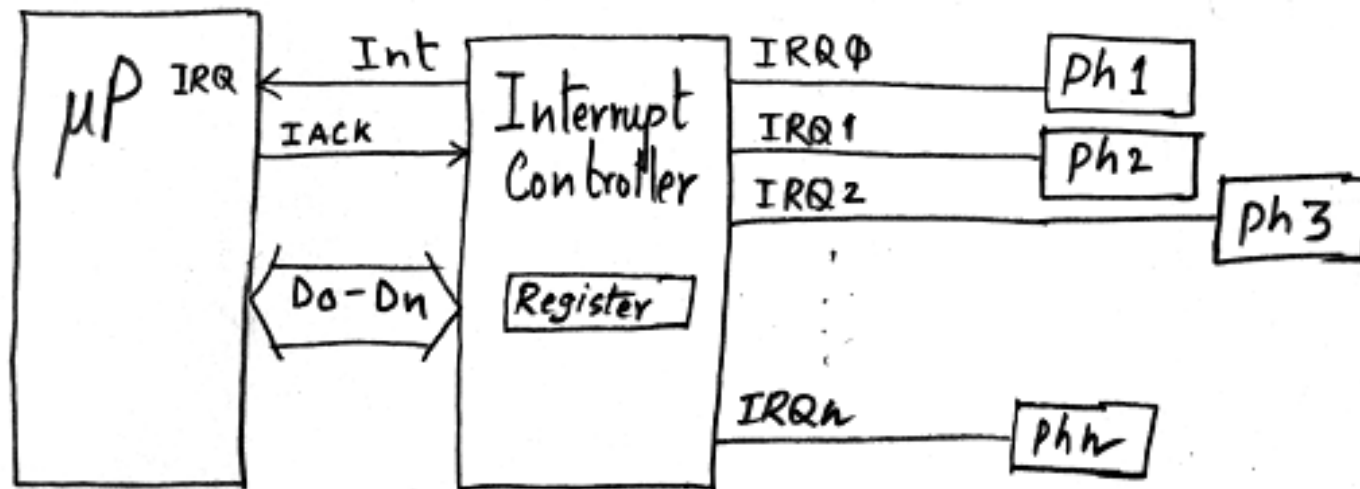
- To ensure only one device is selected to send its vector, devices can be connected to form a daisy chain.



- The interrupt-acknowledge signal propagates serially through the devices.
- The device with a pending request for interrupt blocks the signal and sends its vector to the processor.

Determining Source of Interrupt (5)

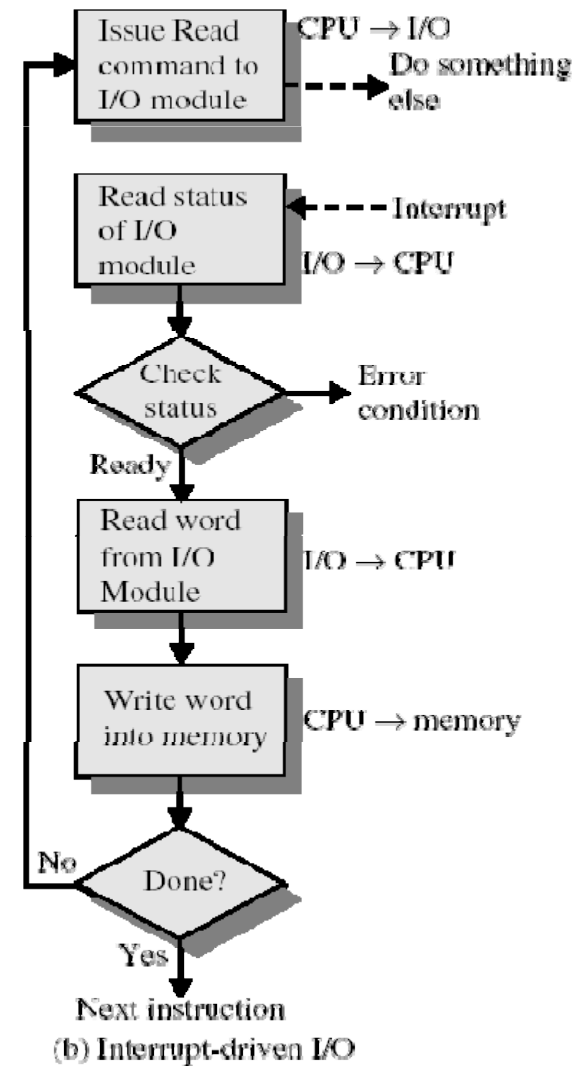
(iii) Dedicated Interrupt Handler , Like Intel 8259 device



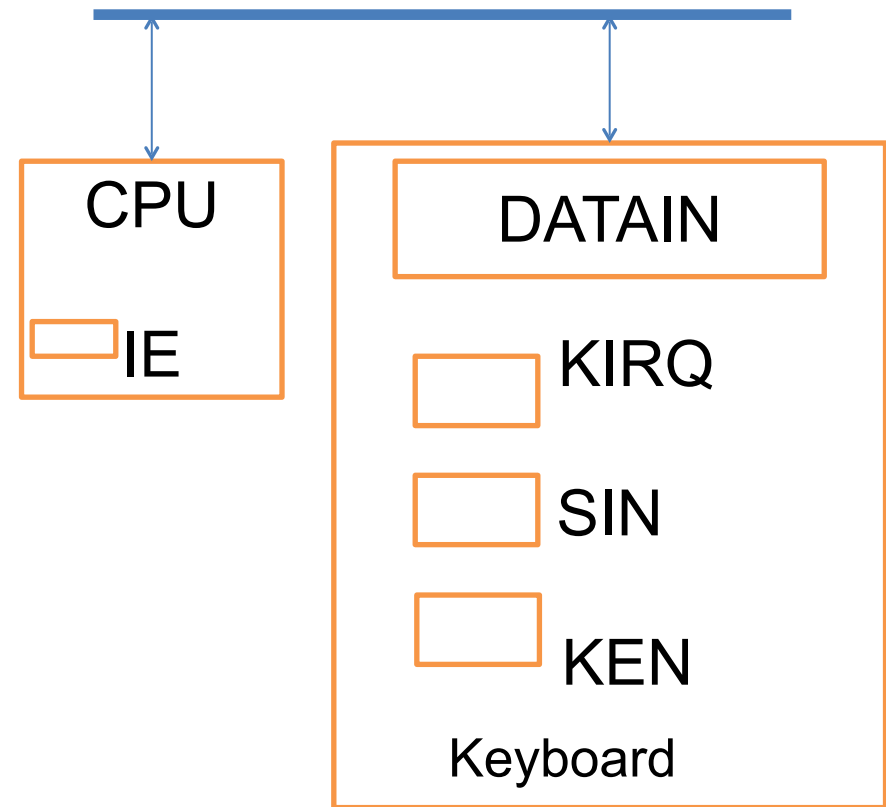
- Here the interrupt controller takes the responsibility of placing the unique Interrupt Vector to μP. For different peripheral the Interrupt Controller puts different interrupt vector. Depending upon the Interrupt Vector, the μP jumps to the corresponding ISR. The value of the Interrupt Vector can be programmed by writing on the register inside the Interrupt Controller.

Interrupt Driven I/O (1)

- Better than programmed I/O, but too many interrupts occur for every byte of input and output.
- Processing an interrupt is expensive.



Interrupt Driven I/O (2)



- Example: In a program called MAIN, it reads a line from the keyboard and store it in the memory, starting at location LINE
- Refer to the simple example used in program-controlled I/O, the following information is added.
 - KEN: keyboard interrupt-enable bit
 - If KEN is set, the interface circuit generates an interrupt request whenever the status flag SIN is set.
 - KIRQ: indicates that the keyboard is requesting an interrupt
 - IE: interrupt-enable bit at the processor.

Interrupt Driven I/O (3)

- Initialization
 - Load the starting address of the ISR.
 - Load the address LINE.
 - Enable keyboard interrupts by setting KEN to 1.
 - Enable interrupts in the processor by setting IE to 1.
- When a character is typed on the keyboard, an interrupt request will be generated.
- The program being executed will be interrupted.


Interrupt Driven I/O (4)

- The ISR will be executed as follows.
 - Read the input character from DATAIN. This will cause the interface circuit to remove its interrupt request.
 - Store the character in the memory.
 - When the end of the line is reached, clear the KEN bit to disable keyboard interrupts and inform the program MAIN.
 - Return from interrupt.

Why DMA?

- Used for high-speed block transfers between a device and memory
- During the transfer, the CPU is not involved
- Typical DMA devices:
 - Disk drives, tape drives
- Remember (1st slide)
 - Keyboard data rate → 0.01 KB/s (1 byte every 100 ms)
 - Disk drive data rate → 2,000 KB/s (1 byte every 0.5 μ s)

Transfer rate is too high to be controlled by software executing on the CPU

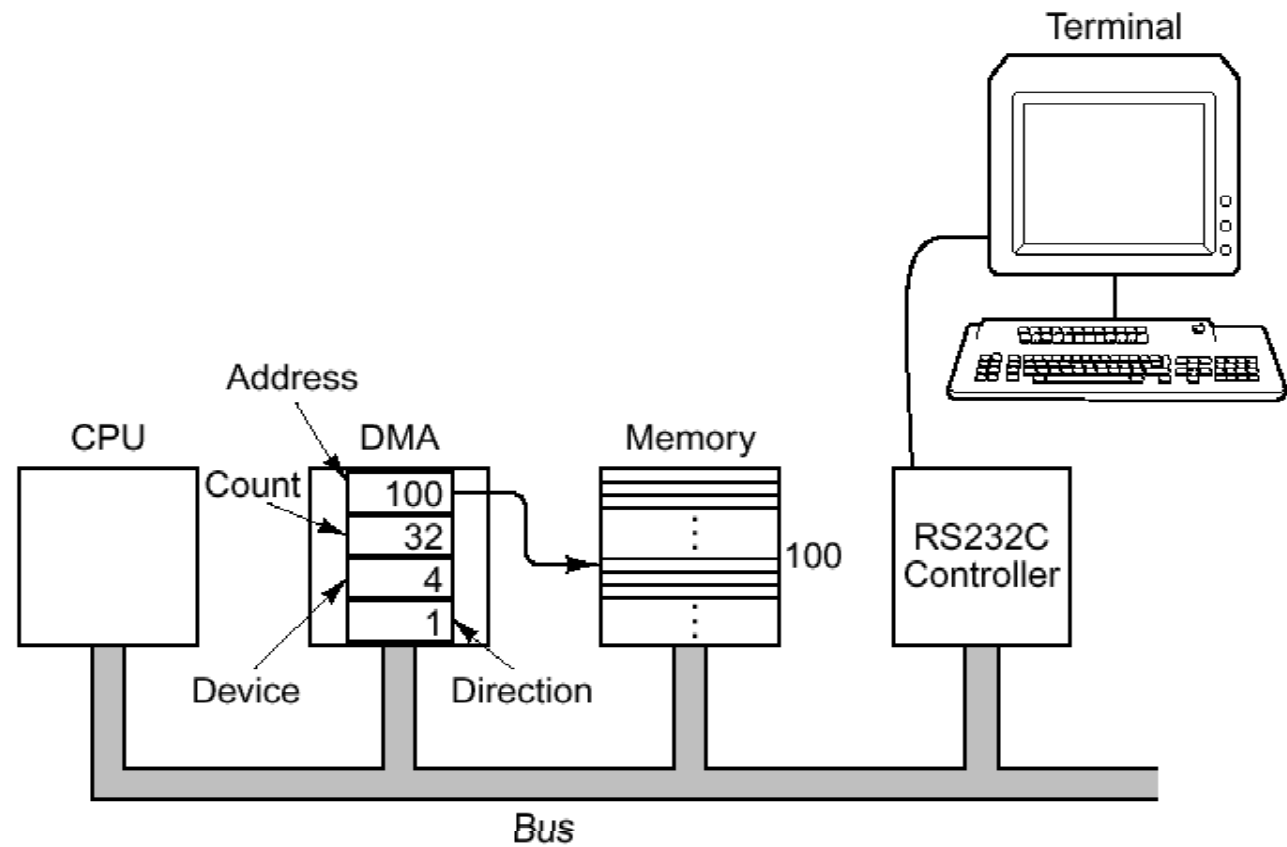


How DMA WORKS

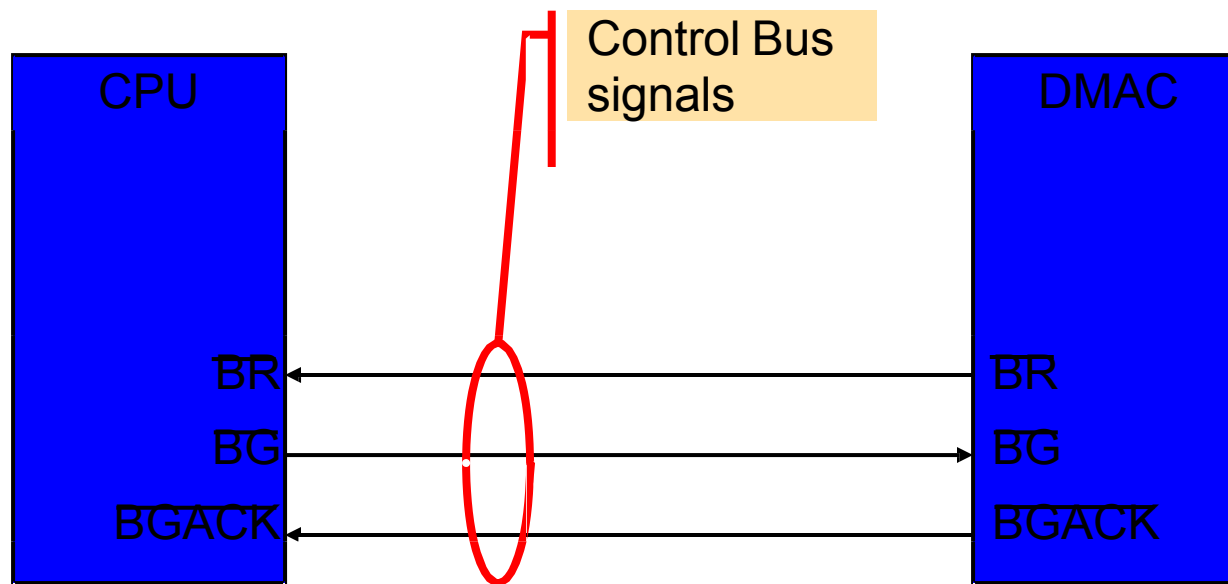
- The CPU “prepares” the DMA operation by transferring information to a DMA controller (DMAC):
 - Location of the data on the device
 - Location of the data in memory
 - Size of the block to transfer
 - Direction of the transfer
 - Mode of transfer (burst, cycle steal)
- When the device is ready to transfer data, the DMAC takes control of the system buses (next few slides)

DMA I/O

- Example:



Taking Control



\overline{BR} = Bus request (DMAC: May I take control of the system buses?)

\overline{BG} = Bus grant (CPU: Yes, here you go.)

\overline{BGACK} = BG acknowledge (DMAC: Thanks, I've got control.)

Taking Control

- DMAC issues a \overline{BR} (“bus request”) signal
- CPU halts (perhaps in the middle of an instruction!) and issues a \overline{BG} (“bus grant”) signal
- DMAC issues \overline{BGACK} (“bus grant acknowledge”) and releases \overline{BR}
- DMAC has control of the system buses
- DMAC “acts like the CPU” and generates the bus signals (e.g., address, control) for one transfer to take place
- Then...

DMA Transfers

- **Burst mode**

- This transfer is repeated until complete
- DMAC relinquishes control of the system buses by releasing BGACK

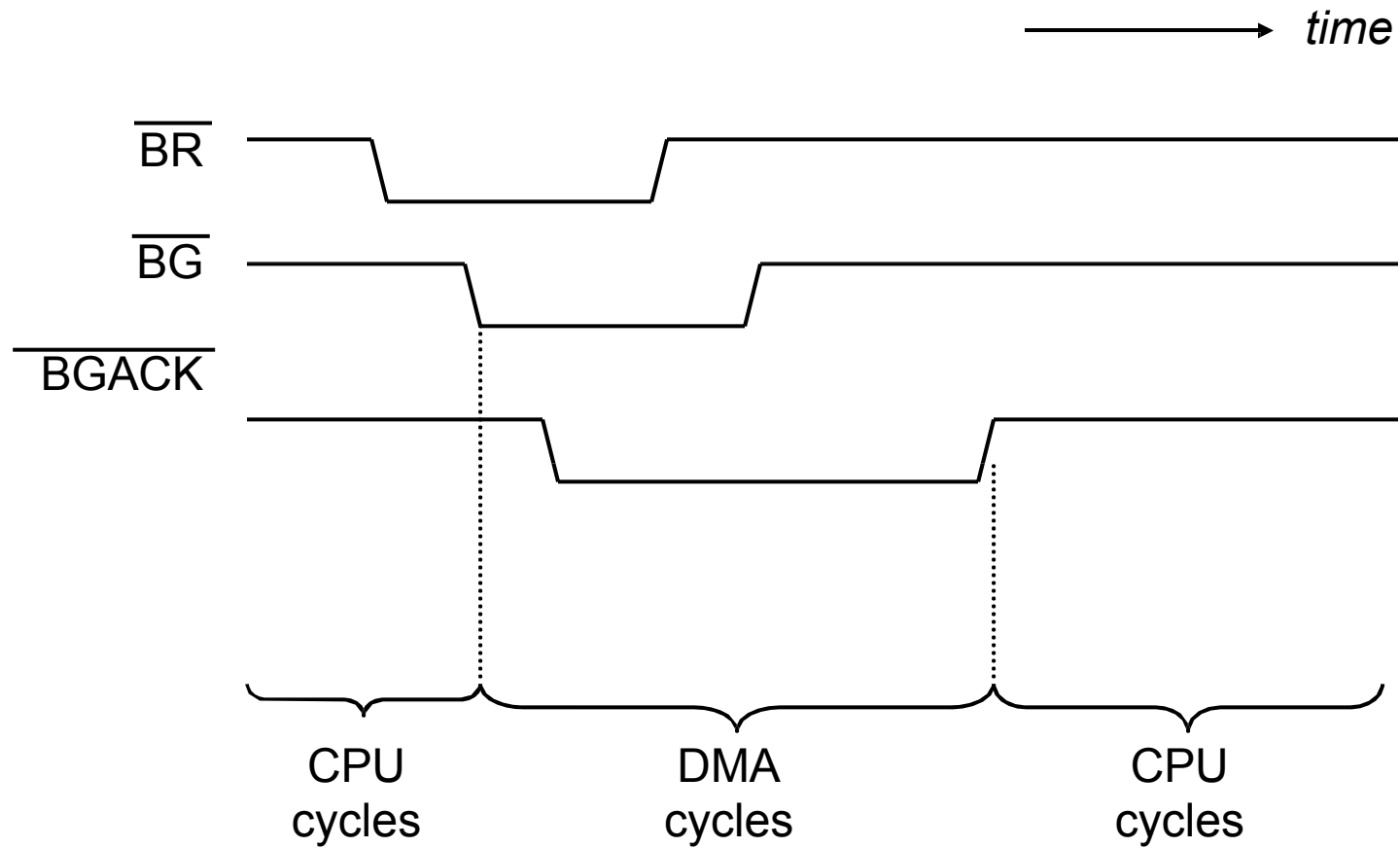
- **Cycle steal mode**

- DMAC relinquishes control of the system buses by releasing BGACK
- A ~~BR-BG-BGACK~~ sequence occurs for every transfer, until the block is completely transferred

- DMAC interrupts the CPU when the transfer is complete

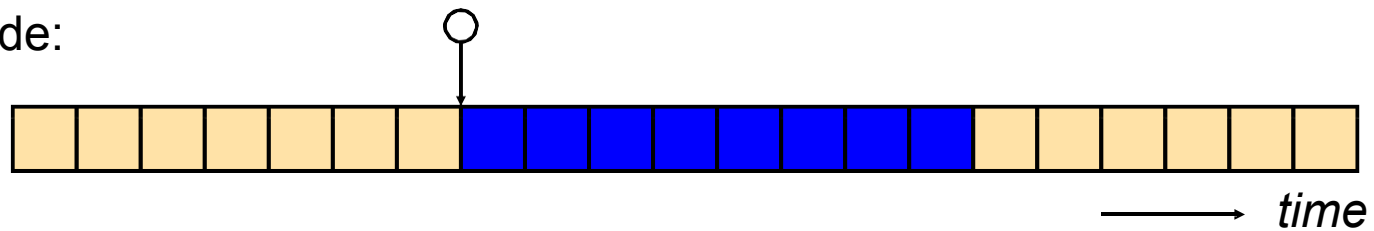
- This is an example of a “completion signal” interrupt

BR-BG-BGACK Timing

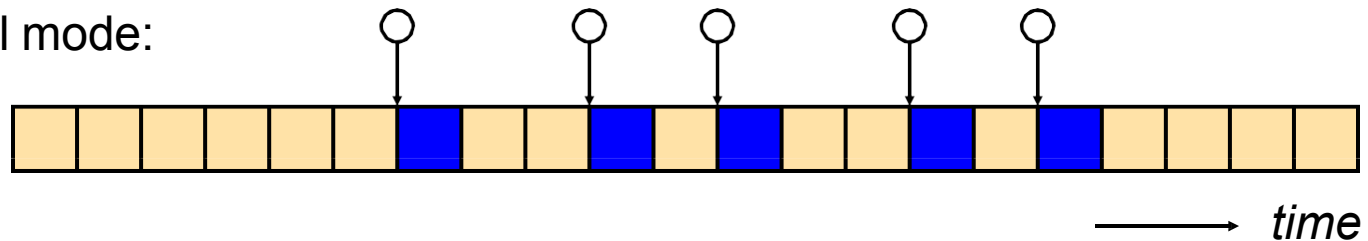


Burst Mode vs. Cycle Steal Mode

Burst mode:



Cycle steal mode:



Legend:

 CPU cycle

 DMA cycle

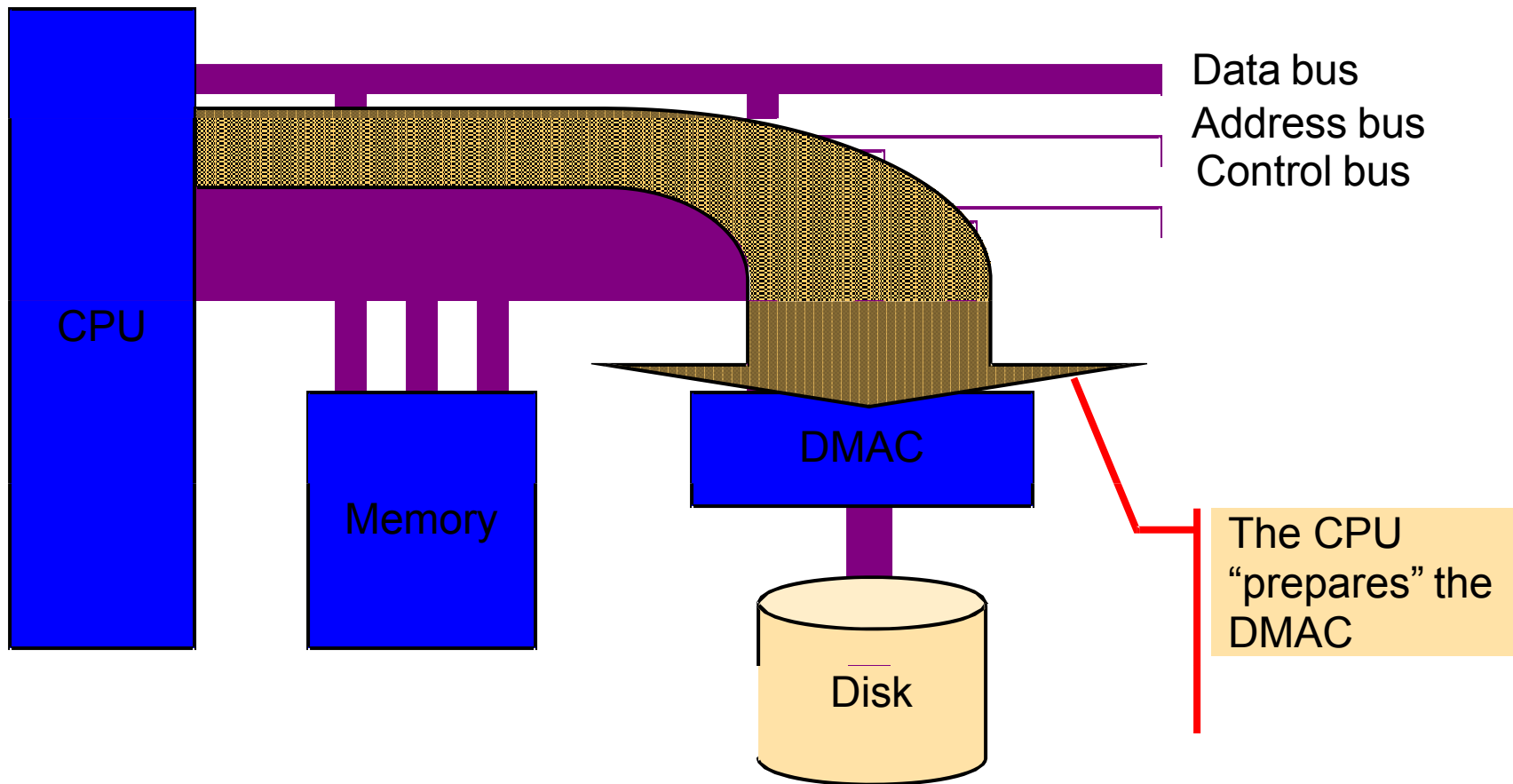
 BR/BG/BGACK sequence

Types of I/O

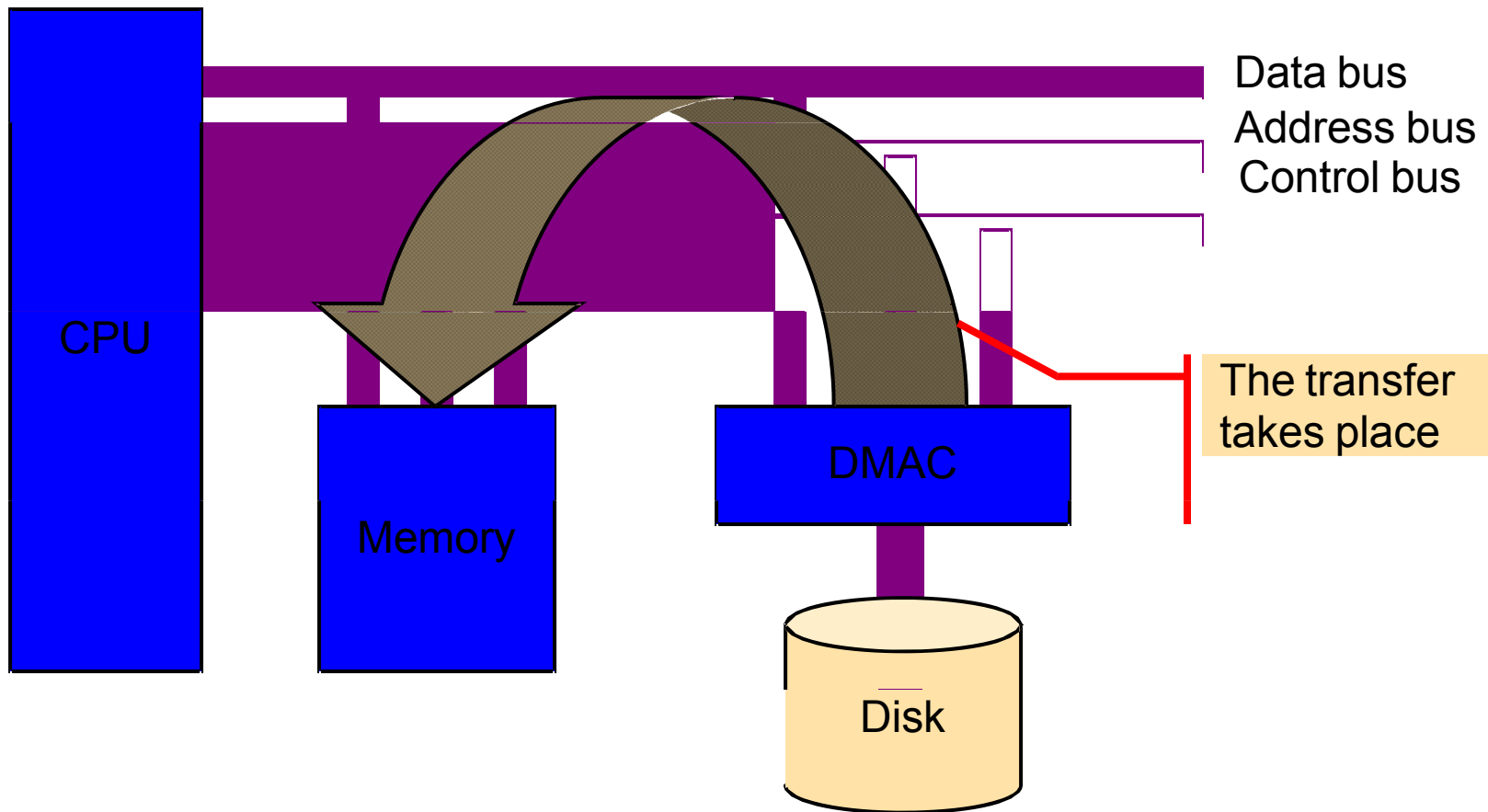
- Programmed I/O
- Interrupt-driven I/O
- Direct memory access (DMA)

DMA includes all three types of I/O.
Let's see...

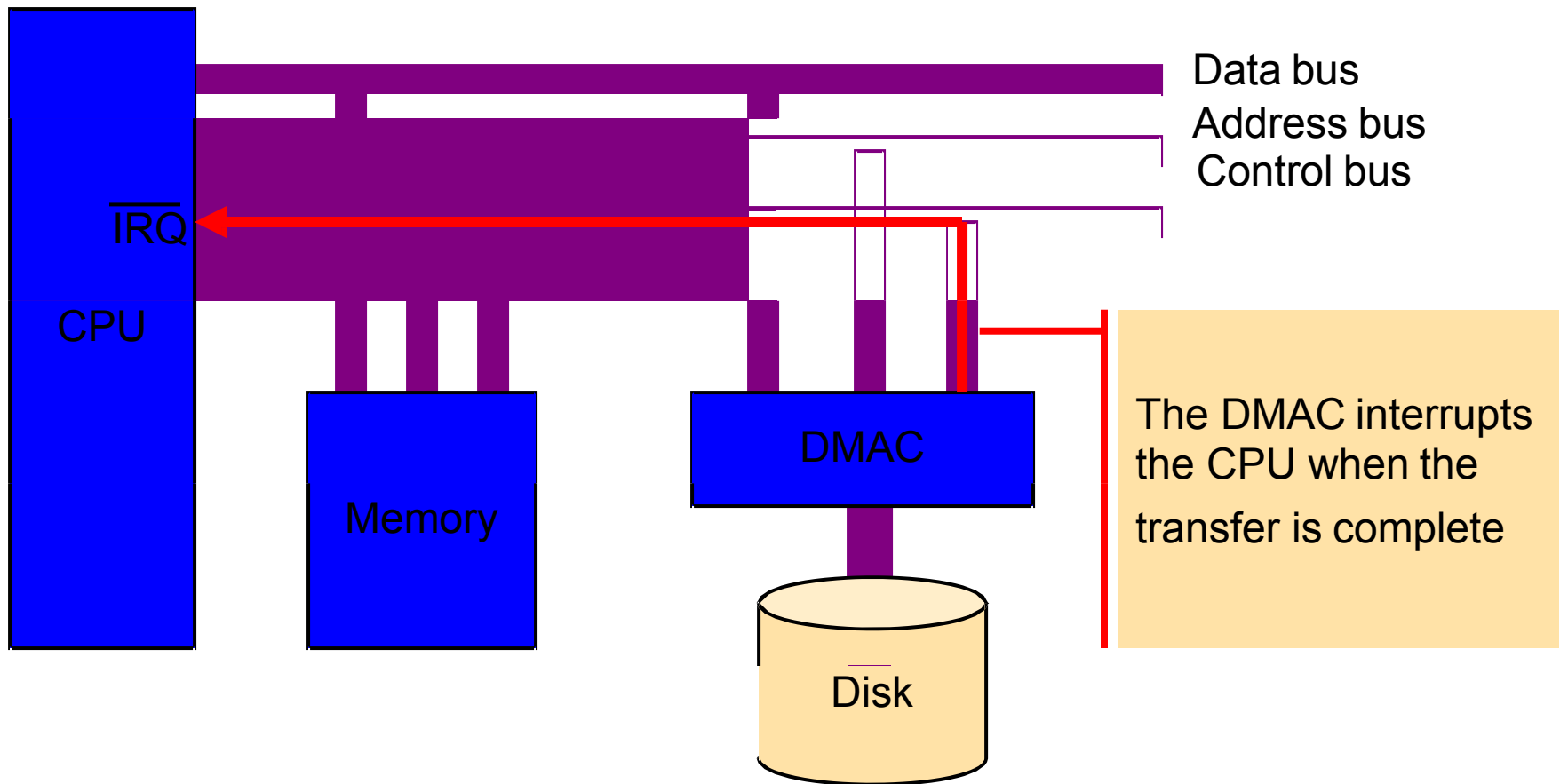
Program-Controlled I/O (in DMA)



DMA



Interrupt-driven I/O (in DMA)



Widely Used External Interface Standard

Universal Serial Bus (USB)

- Widely used for peripheral connections
- Is the default interface for slower speed devices
- Commonly used high-speed I/O
- Has gone through multiple generations
 - USB 1.0
 - Defined a *Low Speed* data rate of 1.5 Mbps and a *Full Speed* rate of 12 Mbps
 - USB 2.0
 - Provides a data rate of 480 Mbps
 - USB 3.0
 - Higher speed bus called *SuperSpeed* in parallel with the USB 2.0 bus
 - Signaling speed of *SuperSpeed* is 5 Gbps, but due to signaling overhead the usable data rate is up to 4 Gbps
 - USB 3.1
 - Includes a faster transfer mode called *SuperSpeed+*
 - This transfer mode achieves a signaling rate of 10 Gbps and a theoretical usable data rate of 9.7 Gbps
- Is controlled by a root host controller which attaches to devices to create a local network with a hierarchical tree topology

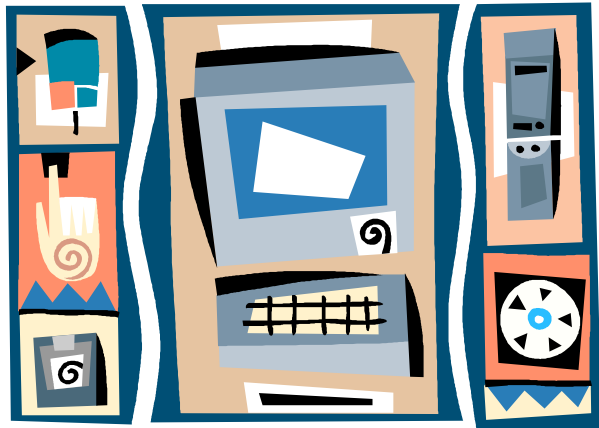
SCSI

- Small Computer System Interface
- A once common standard for connecting peripheral devices to small and medium-sized computers
- Has lost popularity to USB and FireWire in smaller systems
- High-speed versions remain popular for mass memory support on enterprise systems
- Physical organization is a shared bus, which can support up to 16 or 32 devices, depending on the generation of the standard
 - The bus provides for parallel transmission rather than serial, with a bus width of 16 bits on earlier generations and 32 bits on later generations
 - Speeds range from 5 Mbps on the original SCSI-1 specification to 160 Mbps on SCSI-3 U3



PCI Express

- High-speed bus system for connecting peripherals of a wide variety of types and speeds



SATA

- Serial Advanced Technology Attachment
- An interface for disk storage systems
- Provides data rates of up to 6 Gbps, with a maximum per device of 300 Mbps
- Widely used in desktop computers and in industrial and embedded applications

InfiniBand

- I/O specification aimed at the high-end server market
- First version was released in early 2001
- Heavily relied on by IBM zEnterprise series of mainframes
- Standard describes an architecture and specifications for data flow among processors and intelligent I/O devices
- Has become a popular interface for storage area networking and other large storage configurations
- Enables servers, remote storage, and other network devices to be attached in a central fabric of switches and links
- The switch-based architecture can connect up to 64,000 servers, storage systems, and networking devices

Ethernet



- Predominant wired networking technology
- Has evolved to support data rates up to 100 Gbps and distances from a few meters to tens of km
- Has become essential for supporting personal computers, workstations, servers, and massive data storage devices in organizations large and small
- Began as an experimental bus-based 3-Mbps system
- Has moved from bus-based to switch-based
 - Data rate has periodically increased by an order of magnitude
 - There is a central switch with all of the devices connected directly to the switch
- Ethernet systems are currently available at speeds up to 100 Gbps

Wi-Fi

- Is the predominant wireless Internet access technology
- Now connects computers, tablets, smart phones, and other electronic devices such as video cameras TVs and thermostats
- In the enterprise has become an essential means of enhancing worker productivity and network effectiveness
- Public hotspots have expanded dramatically to provide free Internet access in most public places
- As the technology of antennas, wireless transmission techniques, and wireless protocol design has evolved, the IEEE 802.11 committee has been able to introduce standards for new versions of Wi-Fi at higher speeds
- Current version is 802.11ac (2014) with a maximum data rate of 3.2 Gbps



A Small Informative Video about USB

[USB Ports, Cables, Types, & Connectors - YouTube](#)

A Small Informative Video About Various Display Port

[HDMI, DisplayPort, DVI, VGA, Thunderbolt - Video Port Comparison - YouTube](#)