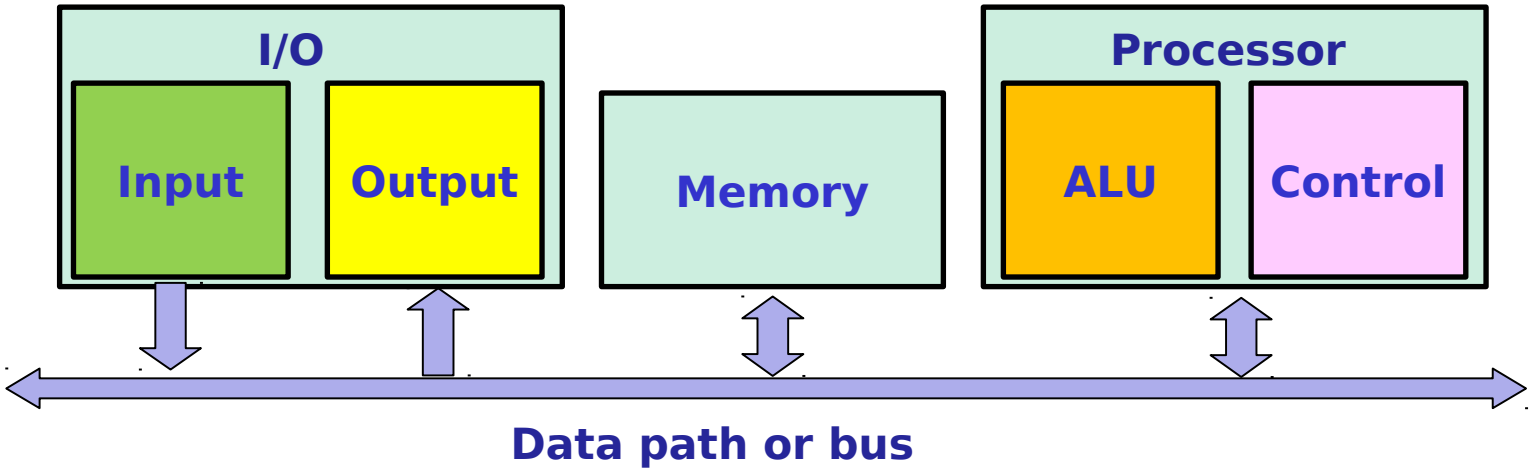
Input/Output Organization

Input/Output (I/O) Subsystem



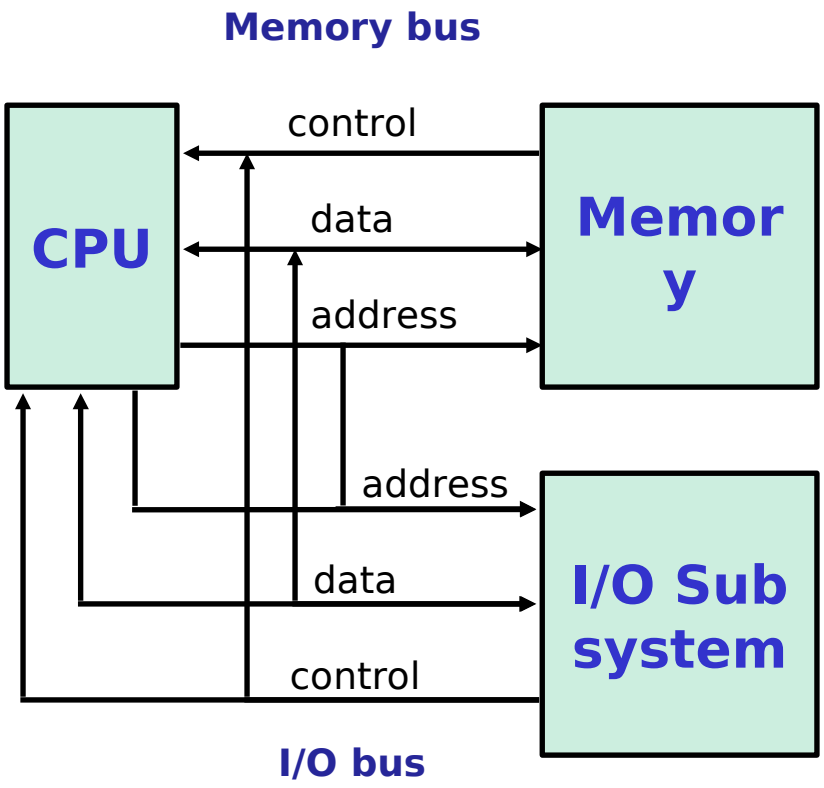
- I/O subsystem provides a mechanism for communications between the CPU and the outside world
- Variety of I/O devices
  - Keyboard and mouse – Input devices
  - Printer and Monitor – Output devices
  - Hard disk – I/O device
- Information moves through the I/O subsystem that connects processor and memory to the devices having different characteristics i.e., speed and timing

Factors Considered in Designing I/O Subsystem

- Data location:
  - It involve device selection, address of data
  - When address is issued for an I/O device, it first specifies a device and then depending on the type of the device, a location of data is specified
- Data transfer:
  - It deals with amount and rate of transfer of information to and from the device
- Synchronization:
  - It deals with synchronizing the processor and I/O devices for data transfer
  - Output only when the device is ready and input only when the data is ready

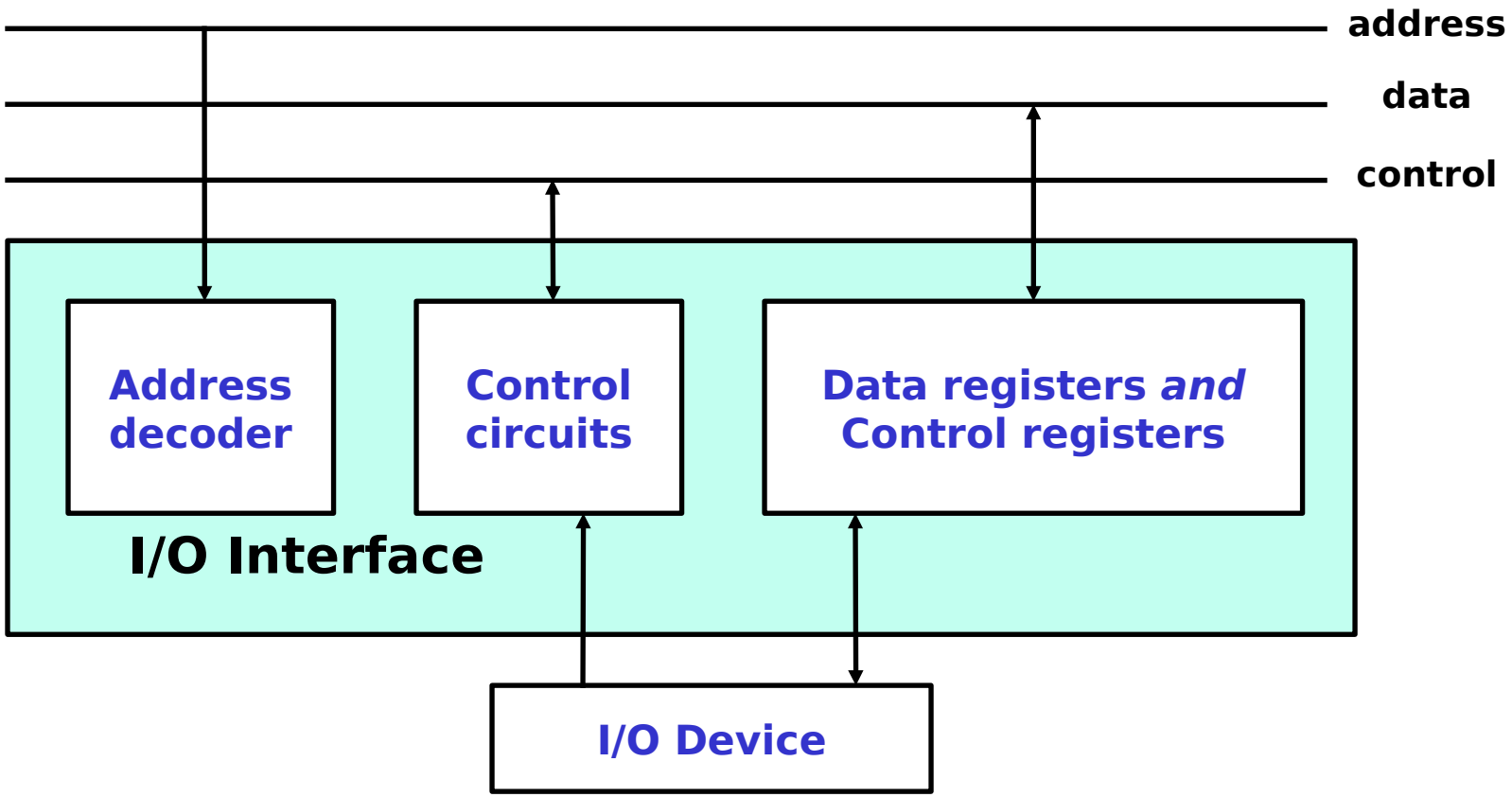
I/O Bus Structure

- It deals with the data transmission path between processor registers and I/O device registers
- Different arrangements for I/O bus structure:
  - Isolated I/O:
    - Separate set of address, control and data lines for I/O bus
  - Shared I/O:
    - The address and data lines are shared with memory bus
    - Separate control lines
  - Memory-mapped I/O:
    - Single bus is used for both I/O and memory



I/O Interface

- I/O interface an hardware to connect and I/O device to the bus
- I/O interface coordinates the I/O transfer
- There will be one I/O interface for every device



I/O Interface

- Address decoder
  - It enables the device to recognise its address when the address appears on the address lines
- Data register
  - It holds the data being transferred to or from the processor
- Status register (Control register)
  - It contains information relevant to the operation of I/O devices

  - If SIN=1, data is ready from input device (data is ready to read for the processor)
  - If SOUT=1, data is ready for output device to receive
- Both data and status registers are connected to data bus

Factors Considered in Designing I/O Subsystem

- Data location:
  - It involve device selection, address of data with in the device
  - When address is issued for an I/O device, it first specifies a device and then depending on the type of the device, a location of data is specified
- Data transfer:
  - It deals with amount and rate of transfer of information to and from the device
- Synchronization:
  - It deals with synchronizing the processor and I/O devices for data transfer
  - Output only when the device is ready and input only when the data is ready

Synchronization Methods

1. Programmed I/O (program-controlled I/O)
   - A bit in the output register may tell a device to start a specific operation and a bit in an input register may signal the completion of operation to the processor
   - In programmed I/O, processor (through program) repeatedly checks a status flag to achieve a required synchronization between the processor and an I/O device
   - Processor polls the device
   - This is not suitable when there is a large and variable latency response time from I/O device that makes the processor to wait for long time
   - In such situations, processor will keep on polling the devices till it get ready

Synchronization Methods

2. Interrupted I/O
   - Processor will not poll the devices
   - Processor is interrupted only when the device is ready

3. Direct memory access (DMA)
   - The high bandwidth bursts of data transmission are synchronized using DMA
   - This transmission happens without the intervention of processor
   - Example: Hard disk

Interrupts

- In Programmed I/O, the program enters the wait loop in which it repeatedly tests the device status
- During this period, the processor is not doing anything useful computation
- Instead, arrange for the I/O devices to alert the processor when it becomes ready
- This is done by sending a hardware signal called interrupt to the processor
- One of the control bus lines called interrupt-request line is dedicated for this
- Now, the processor uses the waiting time to perform some useful functions
- Interrupt-service routine (ISR): Routine executed in response to an interrupt-request

Interrupts

- Example: Printing a document using printer [Refer Hamacher Book]
- Interrupts resembles subroutine calls
- Interrupt-service routine (ISR) is like subroutine
- When the interrupt request arrives during execution of instruction i,
  - Processor first completes the execution of instruction i
  - Then it loads the program counter (PC) with address of the first instruction of ISR
  - After the execution of ISR, the processor has to come back to the instruction i+1
    - Therefore when interrupt occurs, the current content of PC, which points to i+1, must be put into processor stack
- Interrupt-acknowledgement signal: Special control signal in the control line through which the processor inform the device that its interrupt request has been recognised

Enabling and Disabling Interrupts

- The arrival of an interrupt request from an external device causes the processor to suspend the execution of one program and start the execution of another
- Interrupt can arrive at any time and they may alter the sequence of events
- Hence the interruption of a program execution must be carefully controlled
- Processor should have the ability to enable or disable the interrupts as desired
- Interrupts should be disabled during the execution of ISR to ensure that a request from one device will not cause more than one interrupt

Handling Multiple Devices

- Several devices capable of initiating interrupts are connected to the processor
- Several devices may request interrupts at exactly the same time
- Questions to be answered are:
  1. How can processor recognise the device requesting an interrupt?
  2. Given that different devices are likely to require different ISRs, how can processor obtain the starting address of the appropriate routine in each case?
  3. Should a device be allowed to interrupt the processor while another interrupt is being serviced?
  4. How should two or more simultaneous interrupts requests be handled?

Handling Multiple Devices

1. **How can processor recognise the device requesting an interrupt?**

- The information of whether a device is requesting an interrupt is available in its status register
- Special flag called IRQ flag is used in status register
- When a device raises an interrupt request, IRQ flag is set to 1
- Each device has its specific IRQ flag in their status register
  - KIRQ: IRQ flag for keyboard
  - DIRQ: IRQ flag for display
- Then processor polls all the I/O devices and serve the first encountered device whose IRQ bit is set
- Alternate to this is vectored interrupt

Vectored Interrupt

How can processor recognise the device requesting an interrupt?

- Device requesting an interrupt identify itself directly to the processor by sending a special code
- This special code is of length 4-8 bits that enable the processor to identify the individual devices
- The code supplied by the device represent the starting address of ISR of that device
- Processor then serve that device's request by running its ISR

•Generally the ISR or device driver of each device will be loaded to the fixed location in main memory

Nested Interrupts (Interrupt Nesting)

Should a device be allowed to interrupt the processor while another interrupt is being serviced?

- In nested interrupt, I/O devices are organised in a priority structure
- Nested interrupts can cause interrupting processor while another interrupt is being served

- Processor accept the interrupt request from the high priority device when it is serving the device with lower priority
- A device with higher priority now can interrupt the processor while the interrupt from the low priority device is being serviced

Nested Interrupts (Interrupt Nesting)

Should a device be allowed to interrupt the processor while another interrupt is being serviced?

- Implementation of nested interrupt scheme:

– The processor is assigned with a priority level that can be changed

– The processor is assigned with the priority of the program that is currently being executed

– The processor accepts interrupts only from the devices that have priority higher than its current priority

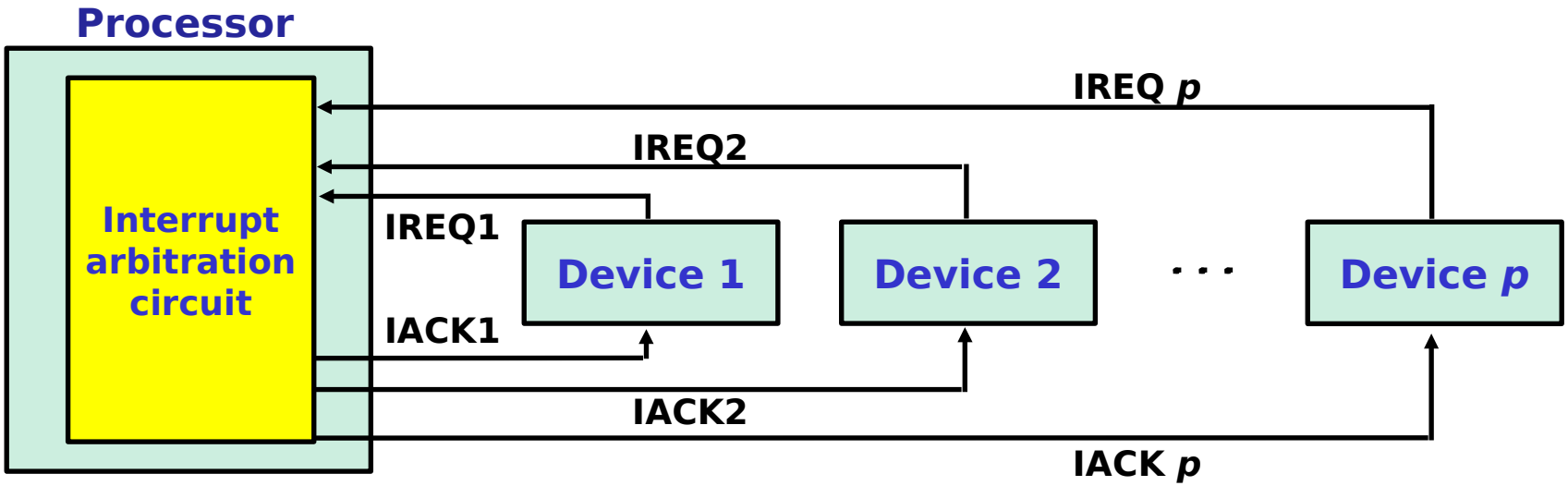
– At the time of execution of ISR for a device, processor's priority is raised to the priority of the device

– This action disables the interrupts from the devices at the same level of priority or lower

Nested Interrupts (Interrupt Nesting)

Should a device be allowed to interrupt the processor while another interrupt is being serviced?

- Implementation of nested interrupt scheme:



– IREQ: Interrupt request signal

– IACK: Interrupt acknowledge signal

Interrupt Priority for Handling Simultaneous Requests

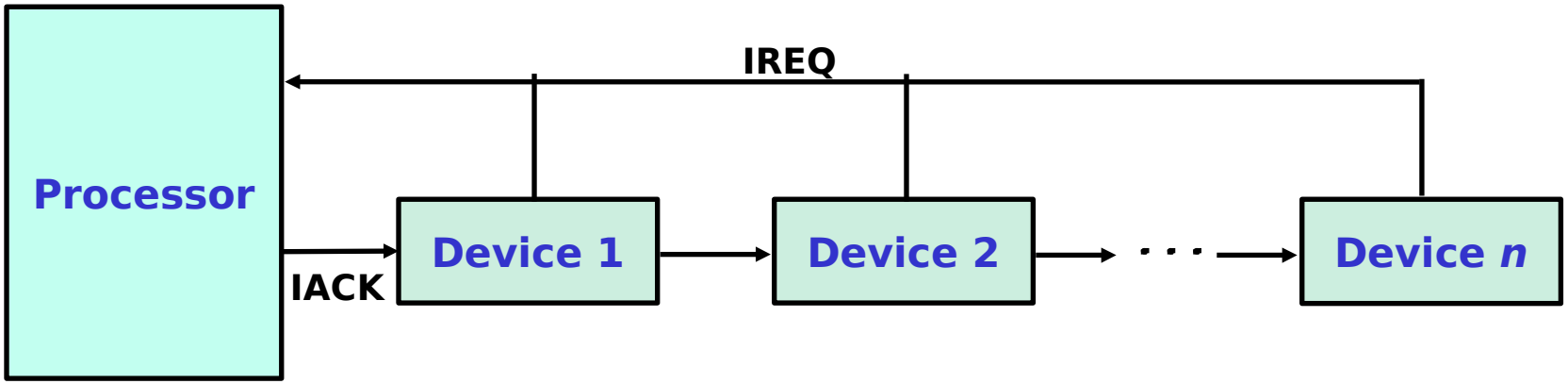
How should two or more simultaneous interrupts requests be handled?

- Processor uses priority scheme as a means of deciding which device's request to serve first
- When multiple simultaneous requests come, processor simply accepts the request having the highest priority

- In the priority scheme, the devices are connected to form a daisy chain
- In daisy chain, devices that are electronically closest to processor has highest priority

Interrupt Priority for Handling Simultaneous Requests

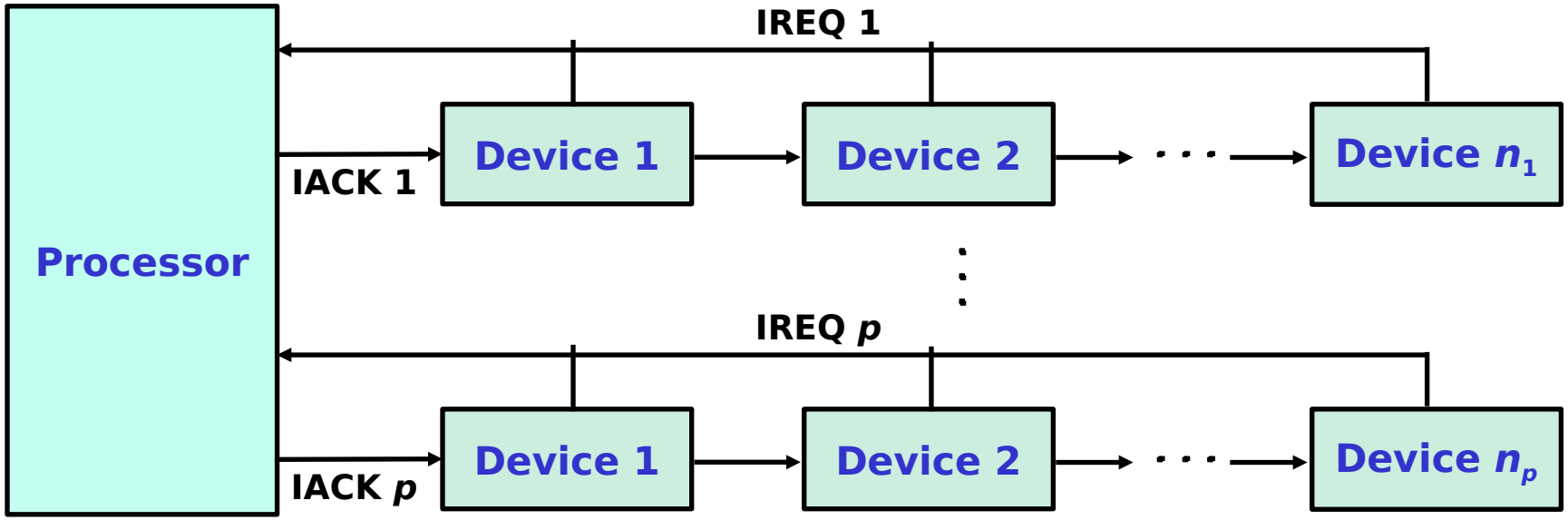
- ***Daisy chain*** for priority scheme:
  - In daisy chain, devices that are electronically closest to processor has highest priority



  - IREQ: Interrupt request signal
  - IACK: Interrupt acknowledge signal
    - It is propagated serially through the devices

Interrupt Priority for Handling Simultaneous Requests

- ***Combining nested priority with daisy chain***:
  - Devices are organised in groups
    - Each group is at different priority level
    - Within the group devices are connected in a daisy chain



  - IREQ: Interrupt request signal
  - IACK: Interrupt acknowledge signal
    - It is propagated serially through the devices

Interrupt

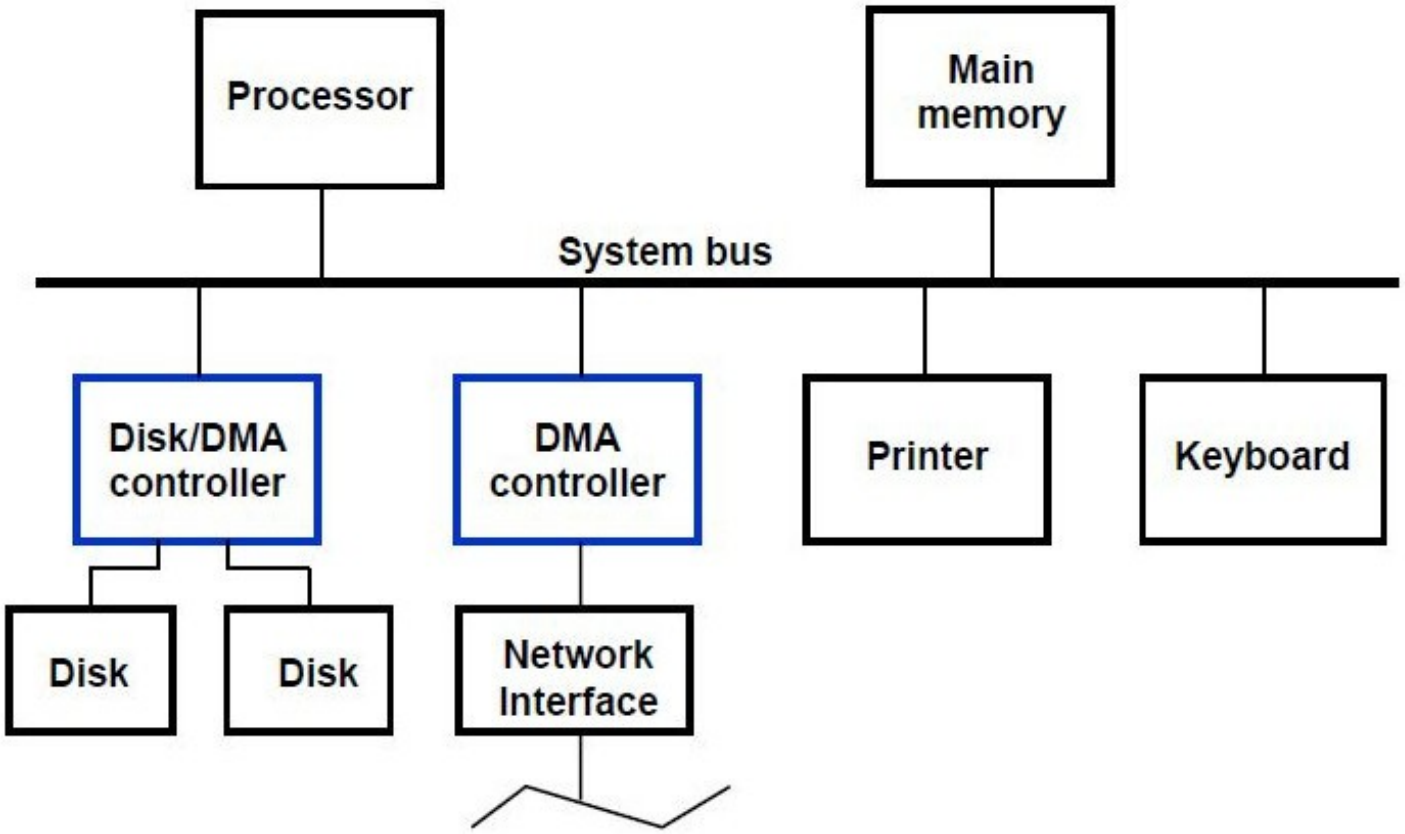
- Harware interrupt
- Software interrupt : int 080

Exception and Interrupt

- Exception:
  - Exception is often used to refer to any event that causes interruption: error checking code, illegal opcode, divide b y zero, NAN, privilege exception
  - Exception service routine
  - I/O interrupts are one example of an exception

- Interrupt:
  - It is an event that causes the execution of one program to be suspended and the execution of another program to begin

- ***Direct Memory Access (DMA)***:



Starting address, word count,
Status register: Done, R/w

Bus arbitration

- A process by which next device to become the bus master is selected and bus mastership is transferred.
  - Centralized
  - Distributed

- Buses:
  - Address
  - Data
  - Control lines: R/W, timing information

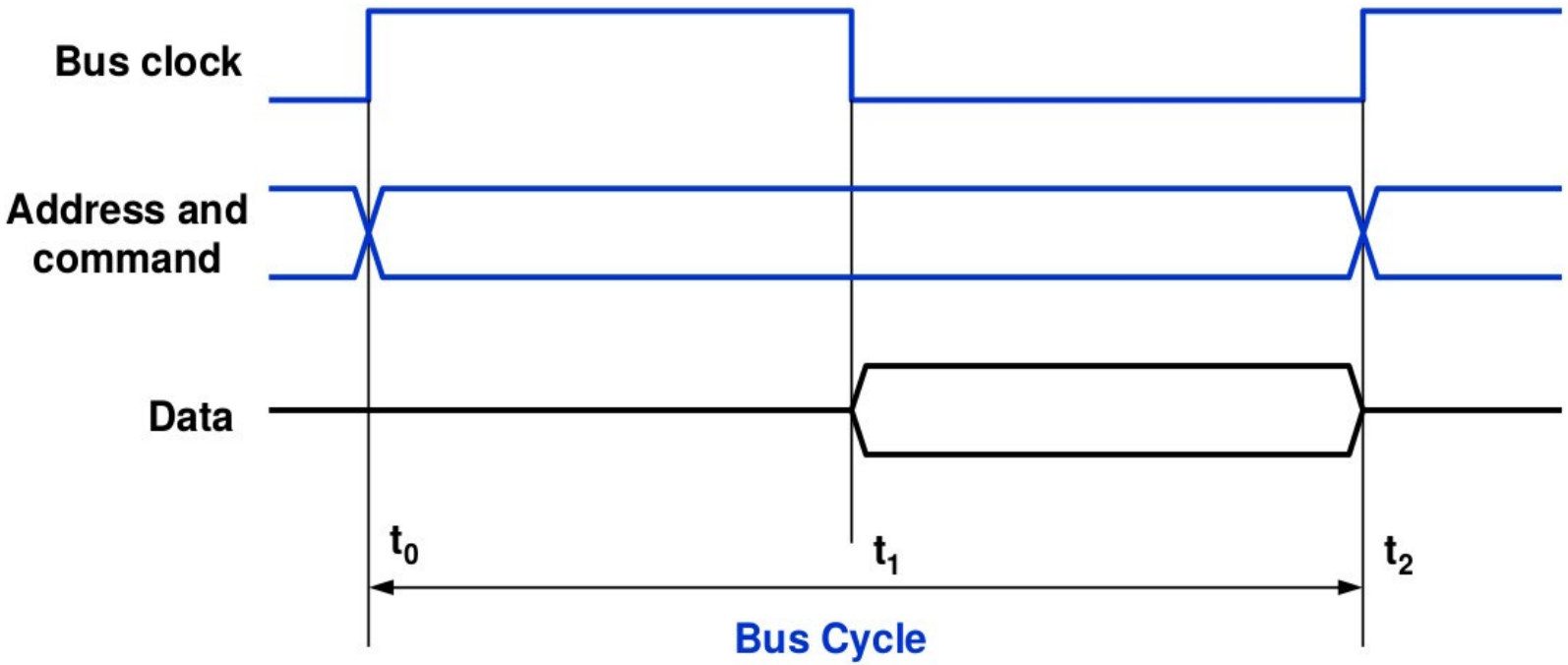
Buses

A bus protocol is the set of rules that govern the behavior of various devices connected to the bus as to when to place information on the bus, assert control signals, and so on

synchronous bus

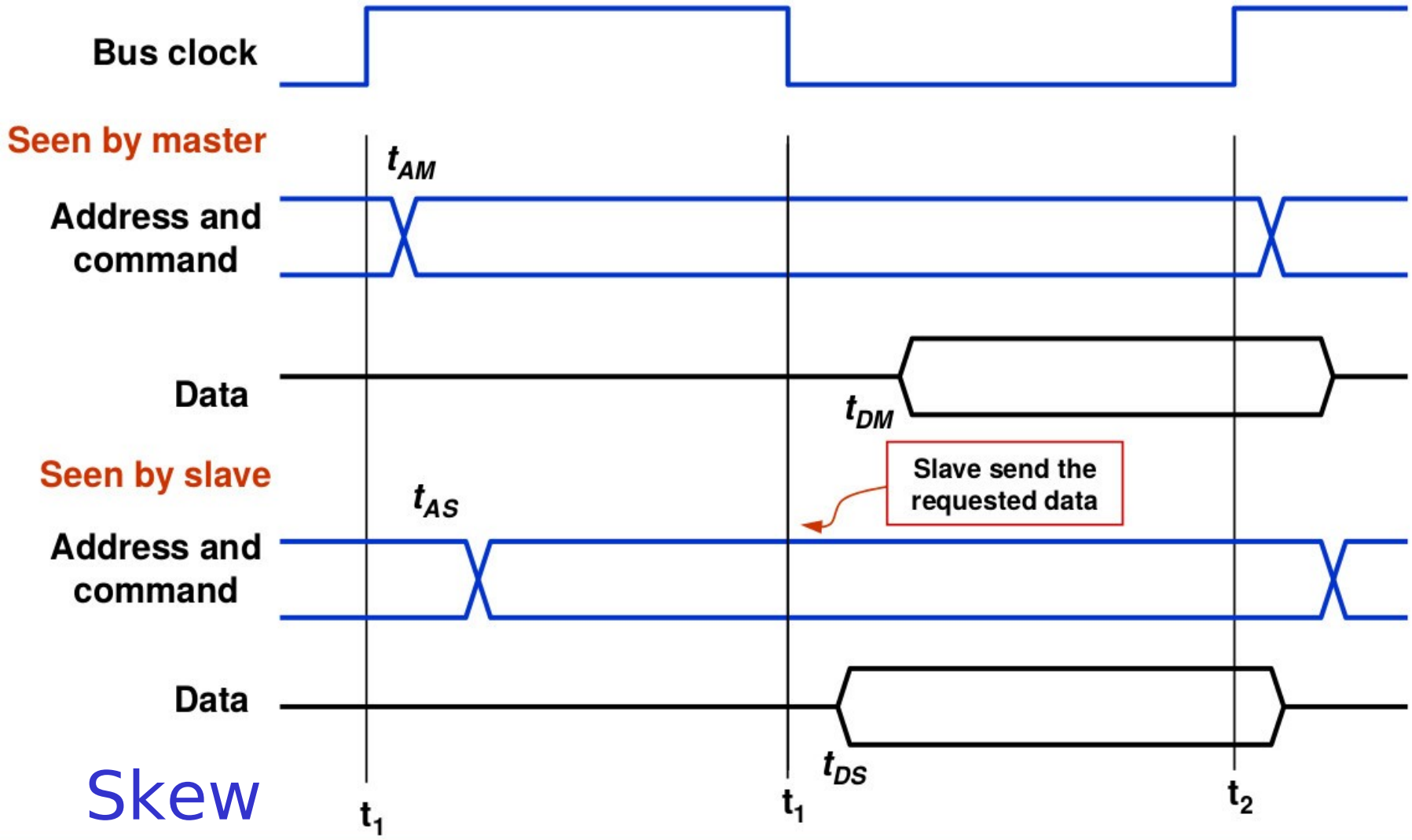
Asynchronous bus

Synchronous Bus



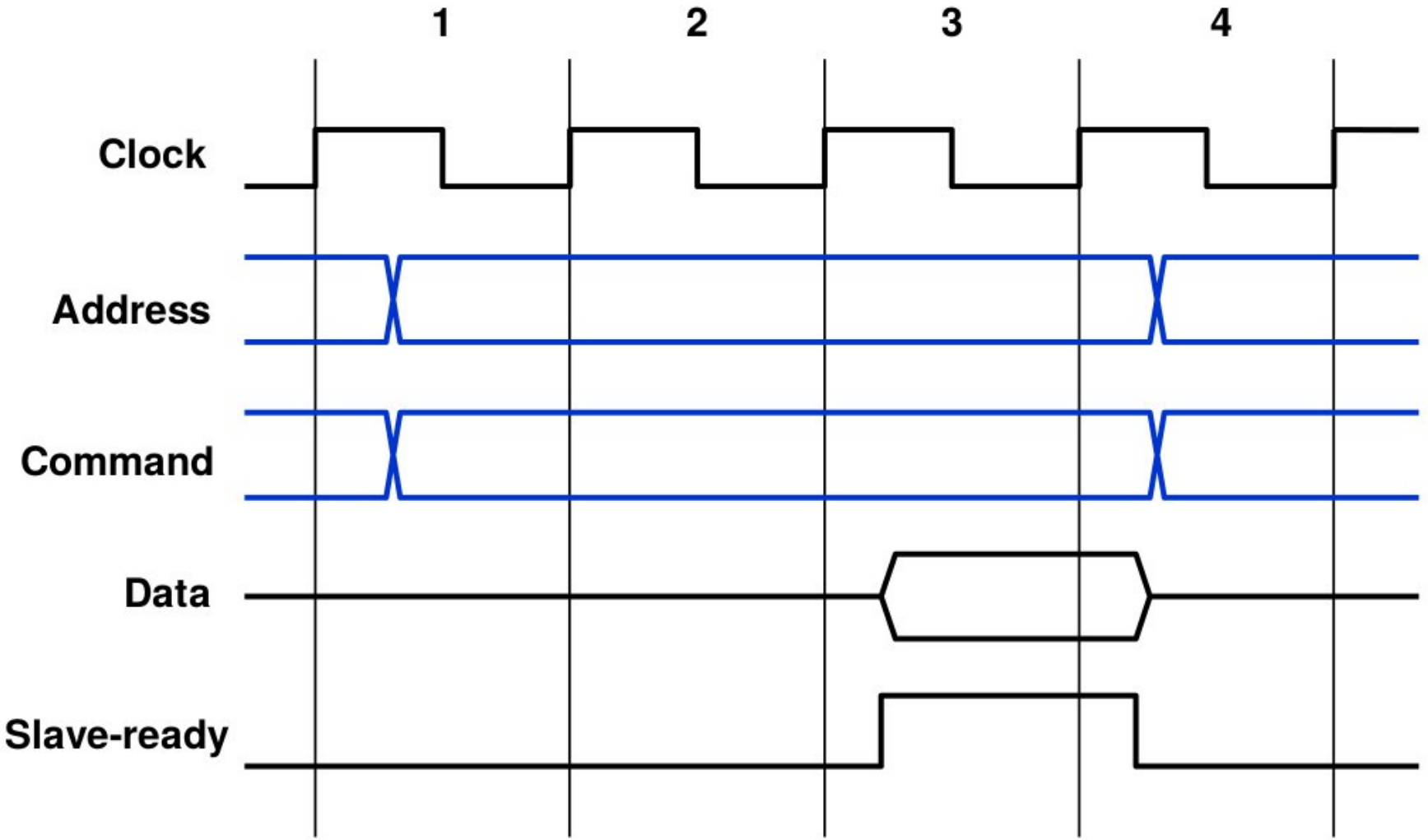
Synchronous Bus

Detailed timing diagram

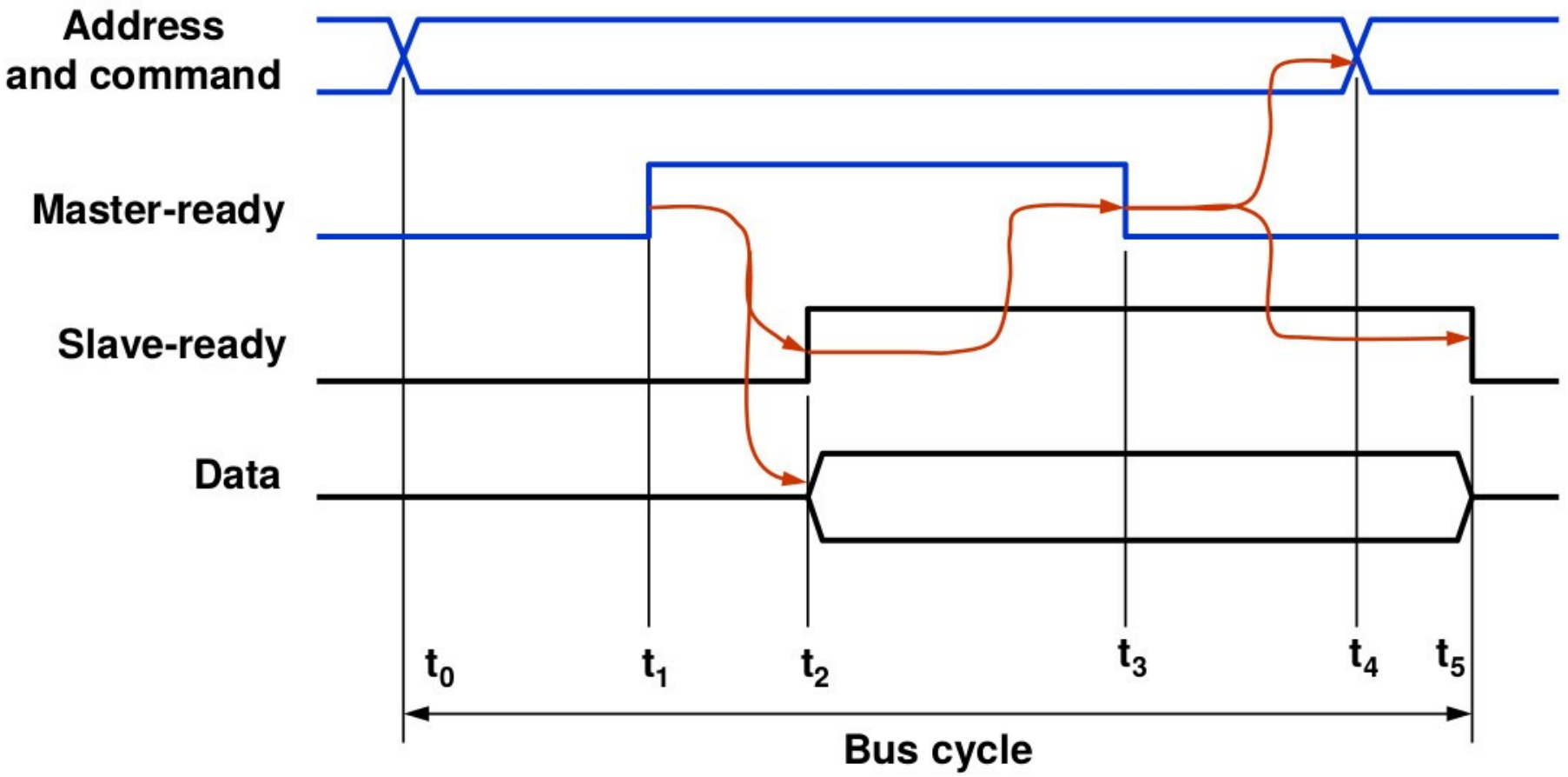


Skew

Synchronous Bus: Multiple clock cycle



ASynchronous Bus



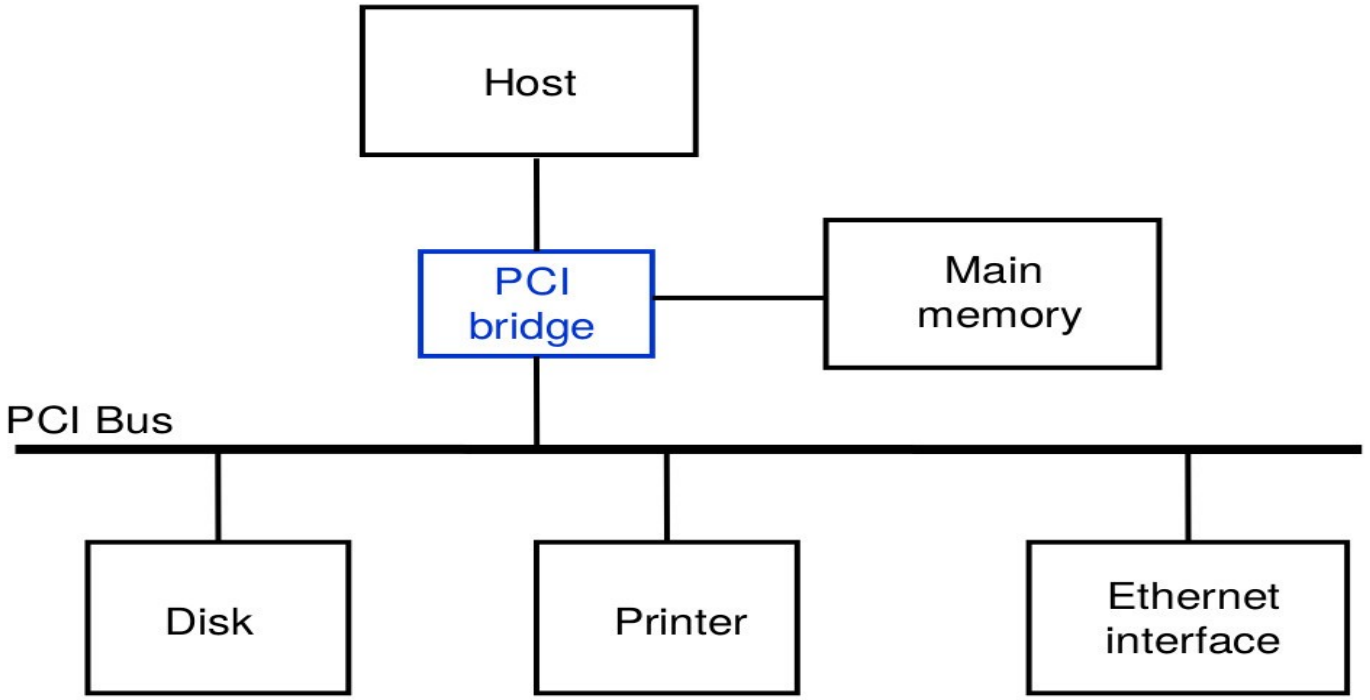
Pros/Cons

- The choice of a particular design involves trade-offs among factors such as
  - simplicity of the device interface
  - Ability to accommodate device interfaces that introduce different amounts of delay
  - Total time required for bus transfer
  - Ability to detect errors resulting from addressing a nonexistent device or from in interface malfunction

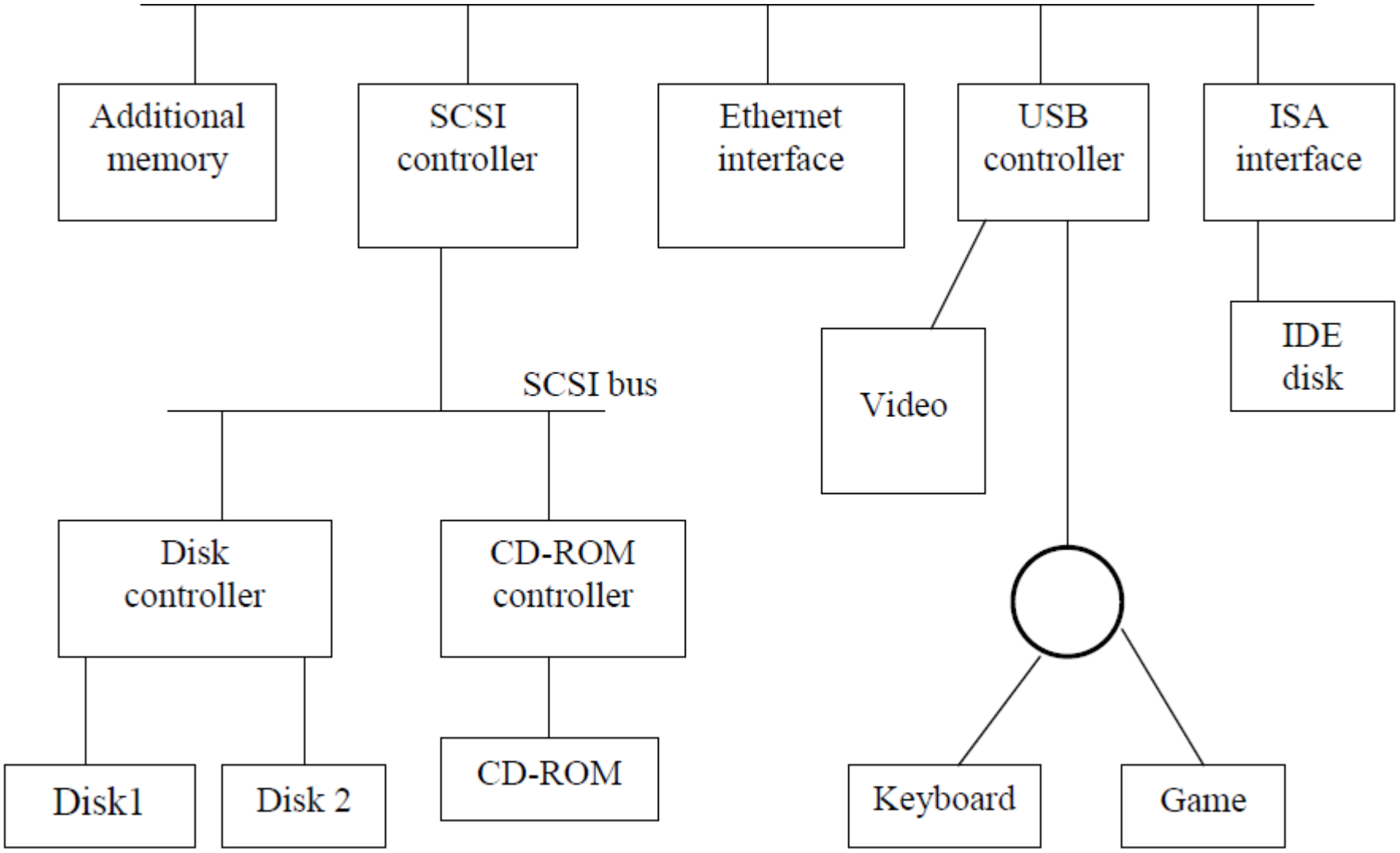
Synchronous Bus:Clock circuitry must be designed carefully to ensure proper synchronization, and delays must be kept within strict bounds

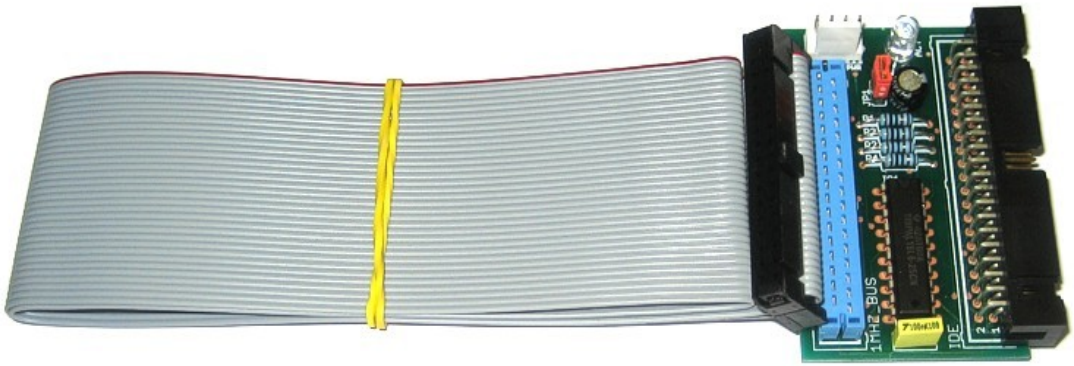
Asynchronous Bus: eliminates the need for synchronization of the sender and receiver clock, thus simplifying timing design

PCI: Peripheral component bus



IDE (Integrated Drive Electronics (IDE)),
SATA (Serial Advance Technology Attachment),
USB (Universal Serial Bus)


Additional memory
SCSI controller
Ethernet interface
USB controller
ISA interface
IDE disk
Video
SCSI bus
Disk controller
CD-ROM controller
Disk1
Disk 2
CD-ROM
Keyboard
Game



IDE

SATA

USB

- Fast
- Bi-directional
- Isochronous (An isochronous data transfer system combines the features of an asynchronous and synchronous data transfer system. An isochronous data transfer system sends blocks of data asynchronously.
- low-cost
- dynamically attachable serial interface
- consistent with the requirements of the PC platform of today and tomorrow

Data Flow Types

- Control Transfers:
  - Used to configure a device at attach time and can be used for other device-specific purposes, including control of other pipes on the device.
- Bulk Data Transfers:
  - Generated or consumed in relatively large and bursty quantities.
- Isochronous Data Transfers:

  Occupy a pre-negotiated amount of USB bandwidth with a pre-negotiated delivery latency. (Also called streaming real time transfers). Isochronous systems do not have an error detection mechanism (acknowledgment of receipt of packet)