SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME :19IT301 COMPUTER ORGANIZATION AND ARCHITECTURE

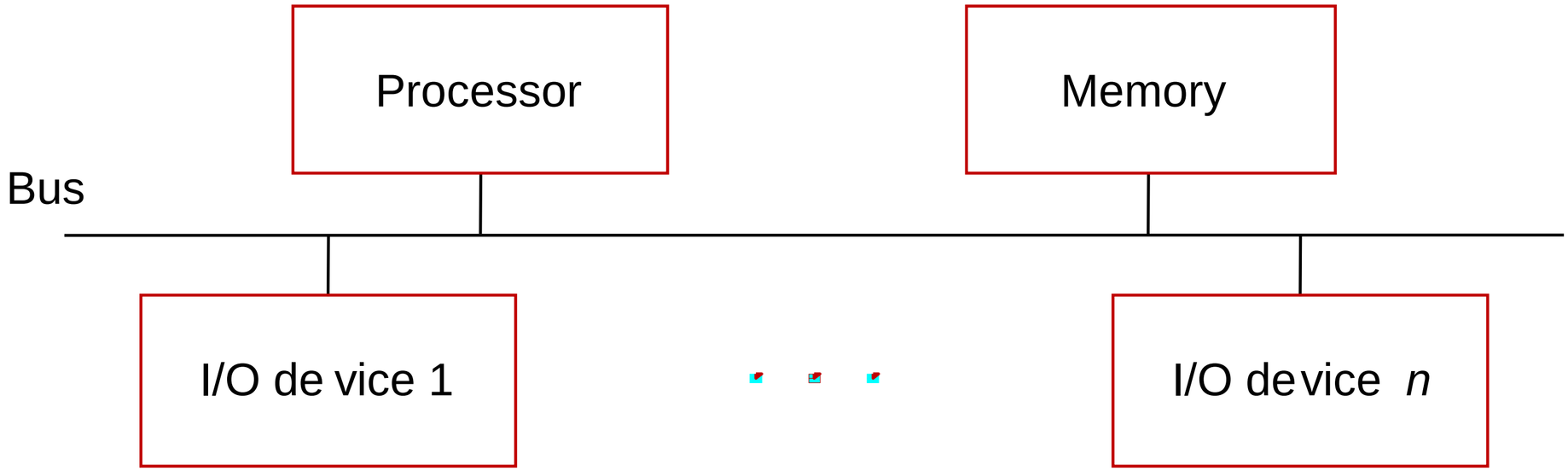
II YEAR /III SEMESTER

Unit 5: I/O ORGANIZATION AND PARALLELISM

Topic 1: Accessing I/O devices

Topic 2: Interrupts

Accessing I/O devices

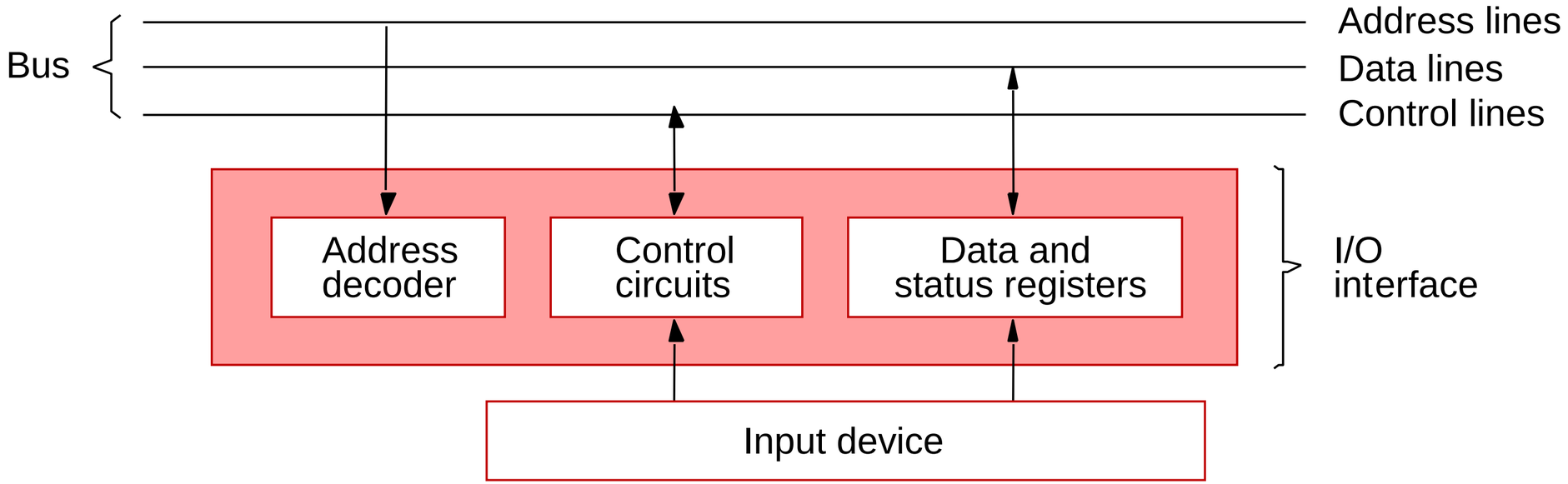


- Multiple I/O devices may be connected to the processor and the memory via a bus.
- Bus consists of three sets of lines to carry address, data and control signals.
- Each I/O device is assigned an unique address.
- To access an I/O device, the processor places the address on the address lines.
- The device recognizes the address, and responds to the control signals.

Accessing I/O devices (contd..)

- I/O devices and the memory may share the same address space:
  - Memory-mapped I/O.
  - Any machine instruction that can access memory can be used to transfer data to or from an I/O device.
  - Simpler software.
- I/O devices and the memory may have different address spaces:
  - Special instructions to transfer data to and from I/O devices.
  - I/O devices may have to deal with fewer address lines.
  - I/O address lines need not be physically separate from memory address lines.
  - In fact, address lines may be shared between I/O devices and memory, with a control signal to indicate whether it is a memory address or an I/O address.

Accessing I/O devices (contd..)



- I/O device is connected to the bus using an I/O interface circuit which has:
  - Address decoder, control circuit, and data and status registers.
- Address decoder decodes the address placed on the address lines thus enabling the device to recognize its address.
- Data register holds the data being transferred to or from the processor.
- Status register holds information necessary for the operation of the I/O device.
- Data and status registers are connected to the data lines, and have unique addresses.
- I/O interface circuit coordinates I/O transfers.

Registers in Keyboard and display interfaces

DATAIN								
DATAOUT								
STATUS					DIRQ	KIRQ	SOUT	SIN
CONTROL					DEN	KEN		

Accessing I/O devices (contd..)

- Recall that the rate of transfer to and from I/O devices is slower than the speed of the processor. This creates the need for mechanisms to synchronize data transfers between them.
- Program-controlled I/O:
  - Processor repeatedly monitors a status flag to achieve the necessary synchronization.
  - Processor polls the I/O device.
- Two other mechanisms used for synchronizing data transfers between the processor and memory:
  - Interrupts.
  - Direct Memory Access.

Interrupts

In program-controlled I/O, when the processor continuously monitors the status of the device, it does not perform any useful tasks.

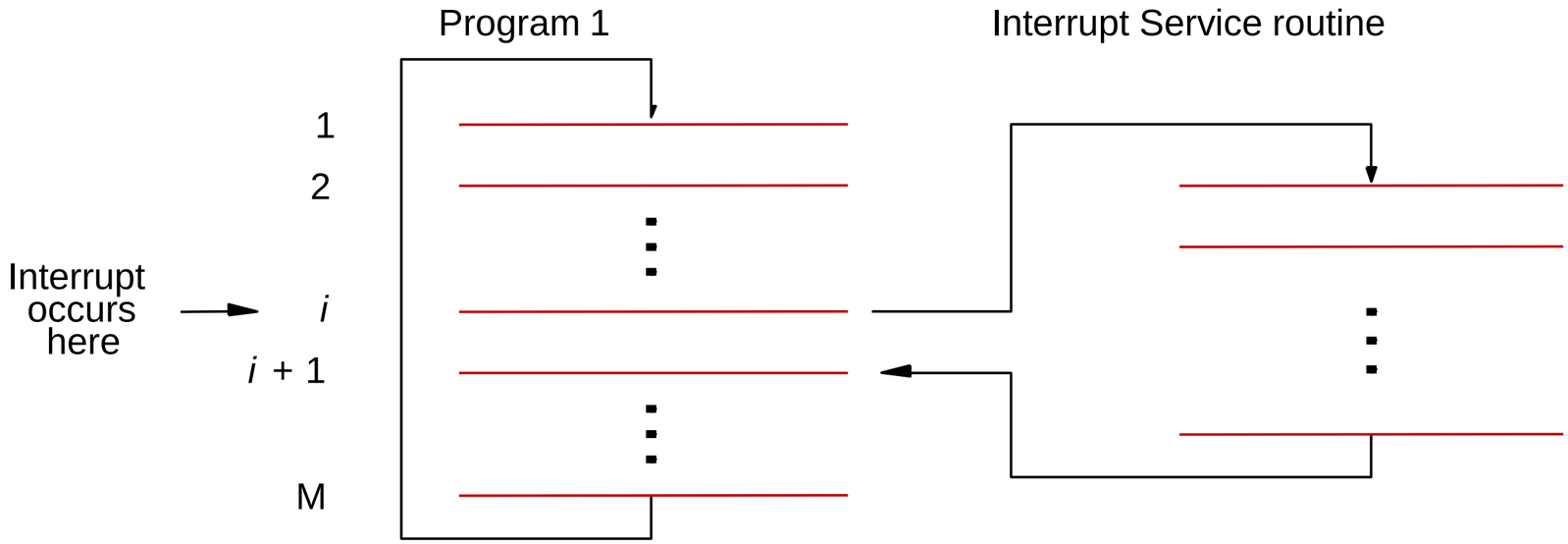
An alternate approach would be for the I/O device to alert the processor when it becomes ready.

Do so by sending a hardware signal called an interrupt to the processor.

At least one of the bus control lines, called an interrupt-request line is dedicated for this purpose.

Processor can perform other useful tasks while it is waiting for the device to be ready.

Interrupts (contd..)



- *Processor is executing the instruction located at address i when an interrupt occurs.*
- *Routine executed in response to an interrupt request is called the interrupt-service routine.*
- *When an interrupt occurs, control must be transferred to the interrupt service routine.*
- *But before transferring control, the current contents of the PC (i+1), must be saved in a known location.*
- *This will enable the return-from-interrupt instruction to resume execution at i+1.*
- *Return address, or the contents of the PC are usually stored on the processor stack.*

Handling Interrupts

Enabling and disabling interrupts
Many situations where the processor should ignore interrupt requests
- Interrupt-disable- first instruction in ISR
- Interrupt-enable -last instruction in ISR

Typical scenario (interrupt process steps)
- Device raises interrupt request
- Processor interrupts program being executed
- Processor disables interrupts and acknowledges interrupt
- Interrupt-service routine executed
- Interrupts enabled and program execution resumed

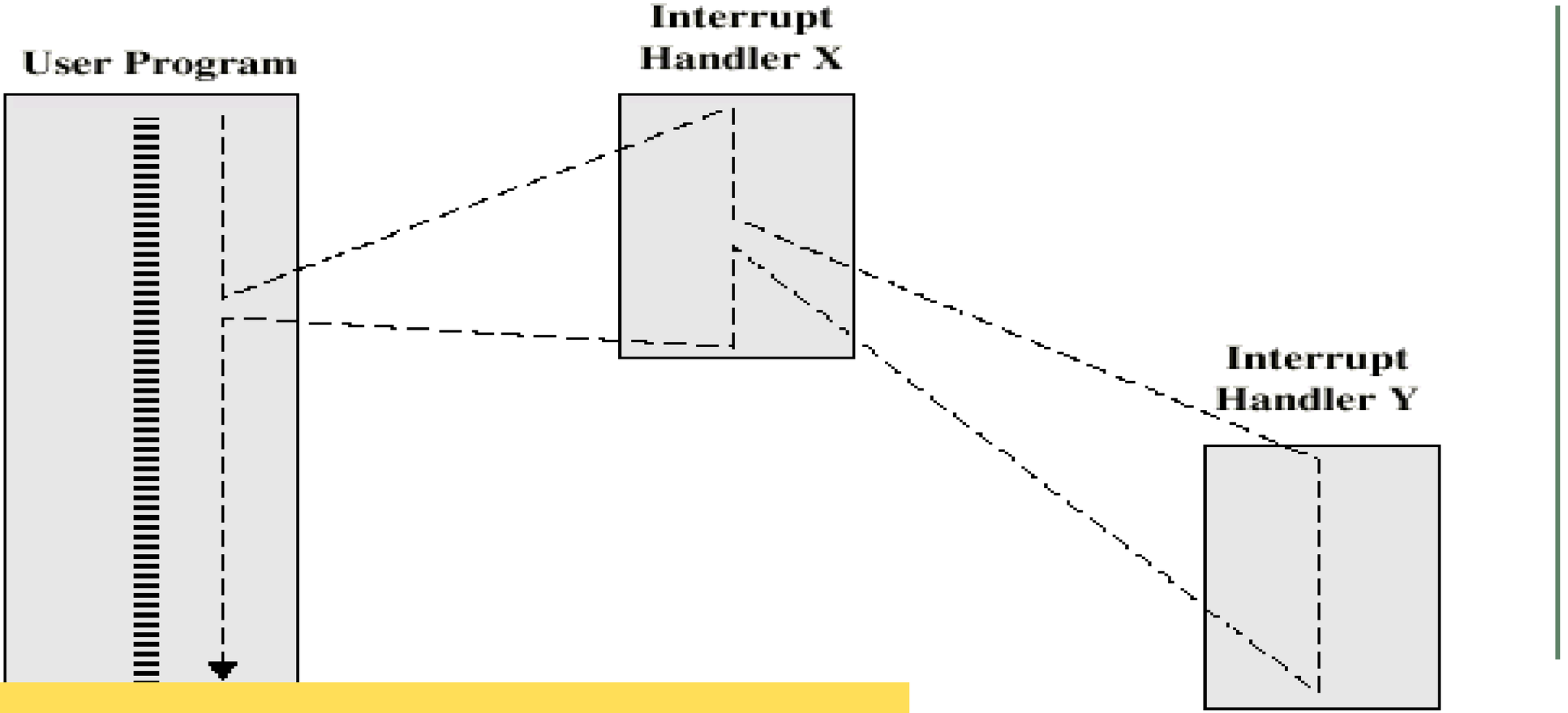
Handling Multiple Devices

How can the processor recognize the device requesting an interrupt?
How can the processor obtain the starting address of the appropriate interrupt-service routine?
Should a device be allowed to interrupt the processor while another interrupt is being serviced?
How should two or more simultaneous interrupt requests be handled?

Handling Multiple Devices

- Polling
  - Interrupt-service routine checks IRQ bits
    - DIRQ, KIRQ
  - Appropriate subroutine called to provide requested service
  - Disadvant: time consuming
- Vectored Interrupts
  - Device requesting interrupt may identify itself directly to the processor
  - Special code over bus
  - Code represents part of address for interrupt-service routine
    - Interrupt vector

Multiple Interrupts – Nested

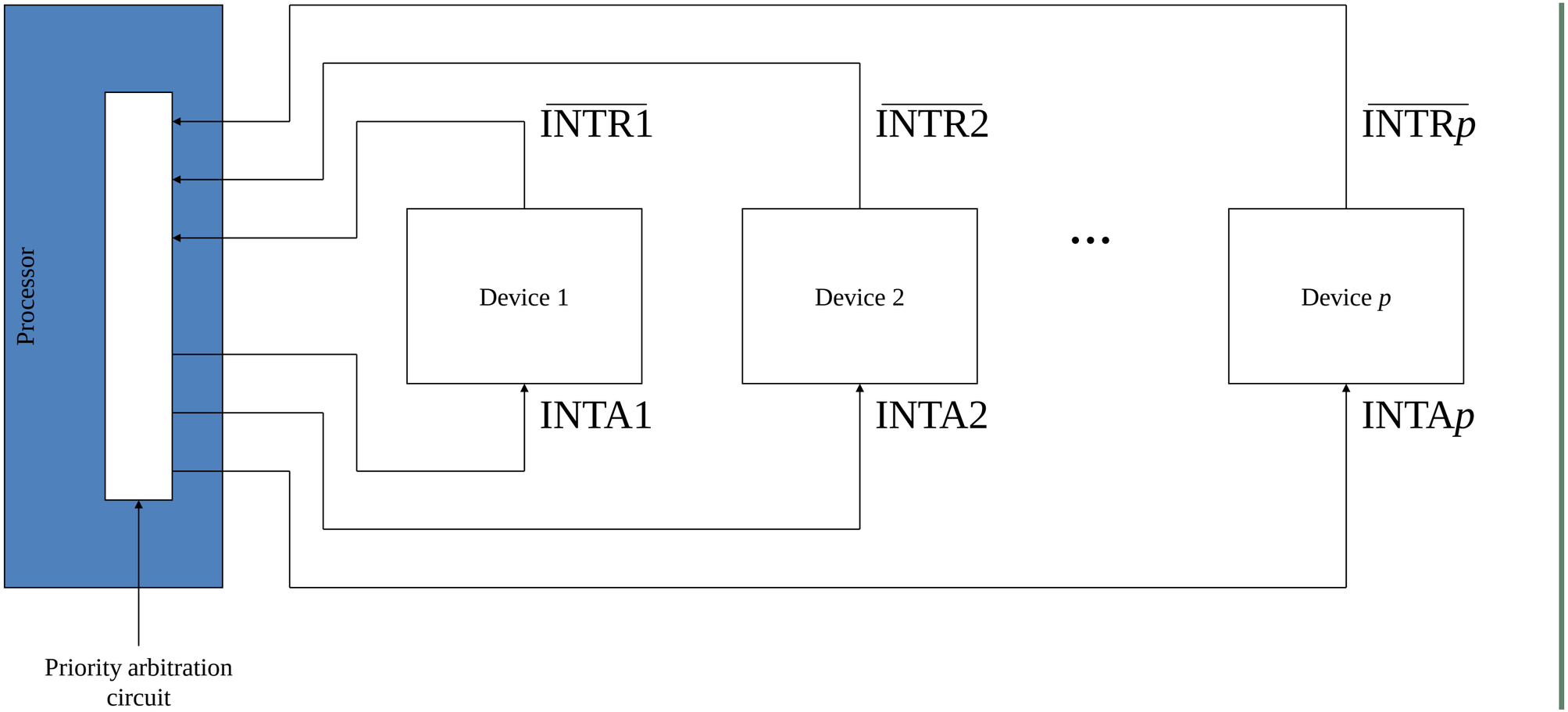


Interrupt Priority

- ✓Interrupt nesting- priority scheme
- ✓Multi-level priority organization
- ✓During execution of interrupt-service routine
  - ✓Disable interrupts from devices at the same level priority or lower
  - ✓Continue to accept interrupt requests from higher priority devices

Priority can be changed by instructions that write into the processor status register. Usually, these are privileged instructions, or instructions that can be executed only in the supervisor mode.

Interrupt Priority Schemes



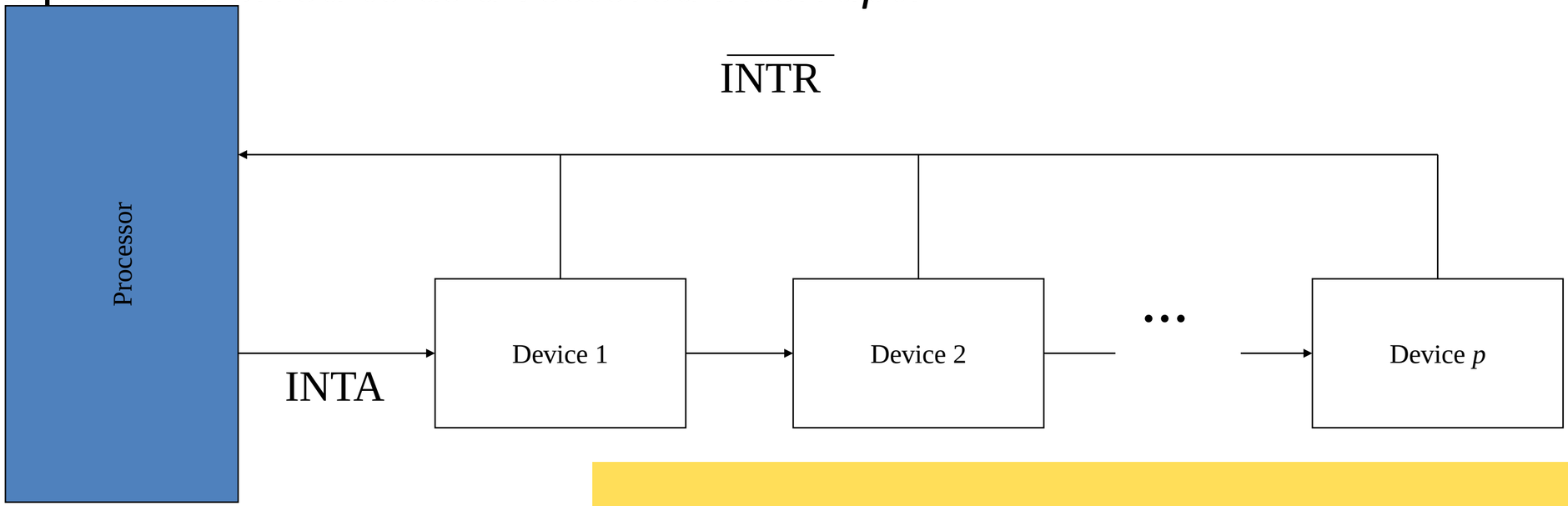
Simultaneous Requests

✓ **Daisy Chain :** In this method all I/O modules share a common interrupt request lines. However the interrupt acknowledge line is connected in a daisy chain fashion.

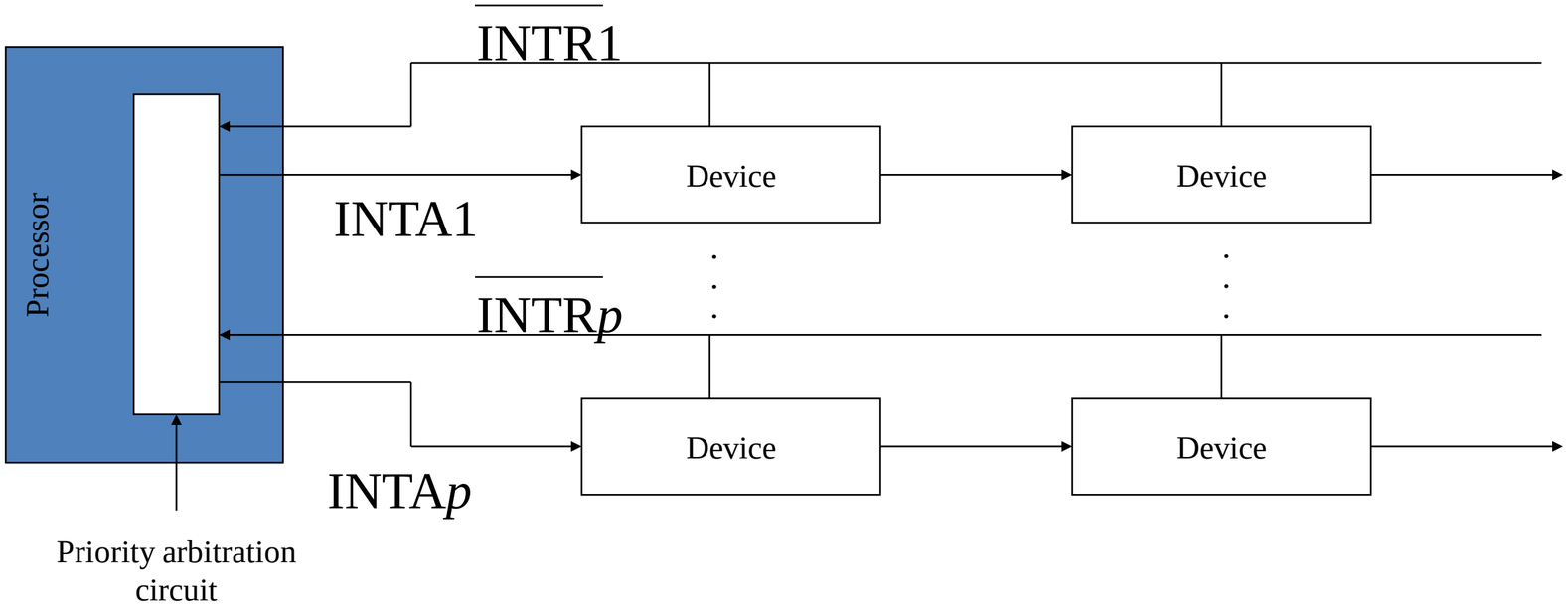
✓When the processor senses an interrupt, it sends out an interrupt acknowledgement.

✓The interrupt acknowledge signal propagates through a series of I/O module until it gets to a requesting module.

✓The requesting module typically responds by placing a word on the data lines. This technique is referred to as a *vectored interrupt*.



Interrupt Priority Groups



- Devices are organized into groups.
- Each group is assigned a different priority level.
- All the devices within a single group share an interrupt-request line, and are connected to form a daisy chain.

Controlling device requests

✓There are two independent mechanisms:

✓At the device end , INTE enable bit in a control register determines whether the device is allowed to generate an INTR.

✓At the processor end, INTE in PS or priority structure determines whether a given interrupt request will be accepted.

Interrupt-enable

KEN, DEN

Exceptions

Any event that causes an interruption.

Types of exceptions:

I. Recovery from errors

Example : 1.main memory checking

2.Illegal instructions

3.Divide by zero

If an error occurs, execute exception service routine, correct the error if possible or inform to the user

II. Debugging

To find and correct errors in the program

Debugger uses exceptions to provide two important facilities:

Trace mode—statement by statement—examine register, memory locations etc.

Breakpoints-specific points selected by the user

III . Privilege exception

Supervisor mode & user mode

In user mode attempt to access OS routines or other user programs

Operating Systems

Coordinate all activities within a computer

Make extensive use of interrupts

Perform (coordinate) I/O
Communicate with user programs
Control execution of user programs
interrupt mechanism enables OS to
Assign priorities
Switch between programs (multi-tasking)
Implement security and protection
Coordinate I/O activities

Ex1: application program(AP) access I/O through OS
The OS and AP pass control back and forth using software interrupts.

Ex2: multitasking--- time slicing----switch to program1 to program2 and so on.

Assessment

What is program controlled IO?
What is Interrupt IO?
What is vectored interrupts?
What is maskable interrupts?

Reference

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, McGraw-Hill, 6th Edition 2012.