# Interrupts

CPTS 260

# Synchronization

- If processor issues instructions too rapidly, the results can be un predictable .
- We can prevent such problems via synchronization.
- Processor operates faster than an I/O Device, programmed I/O requires the processor to synchronize with the device that is being controlled.

# Polling

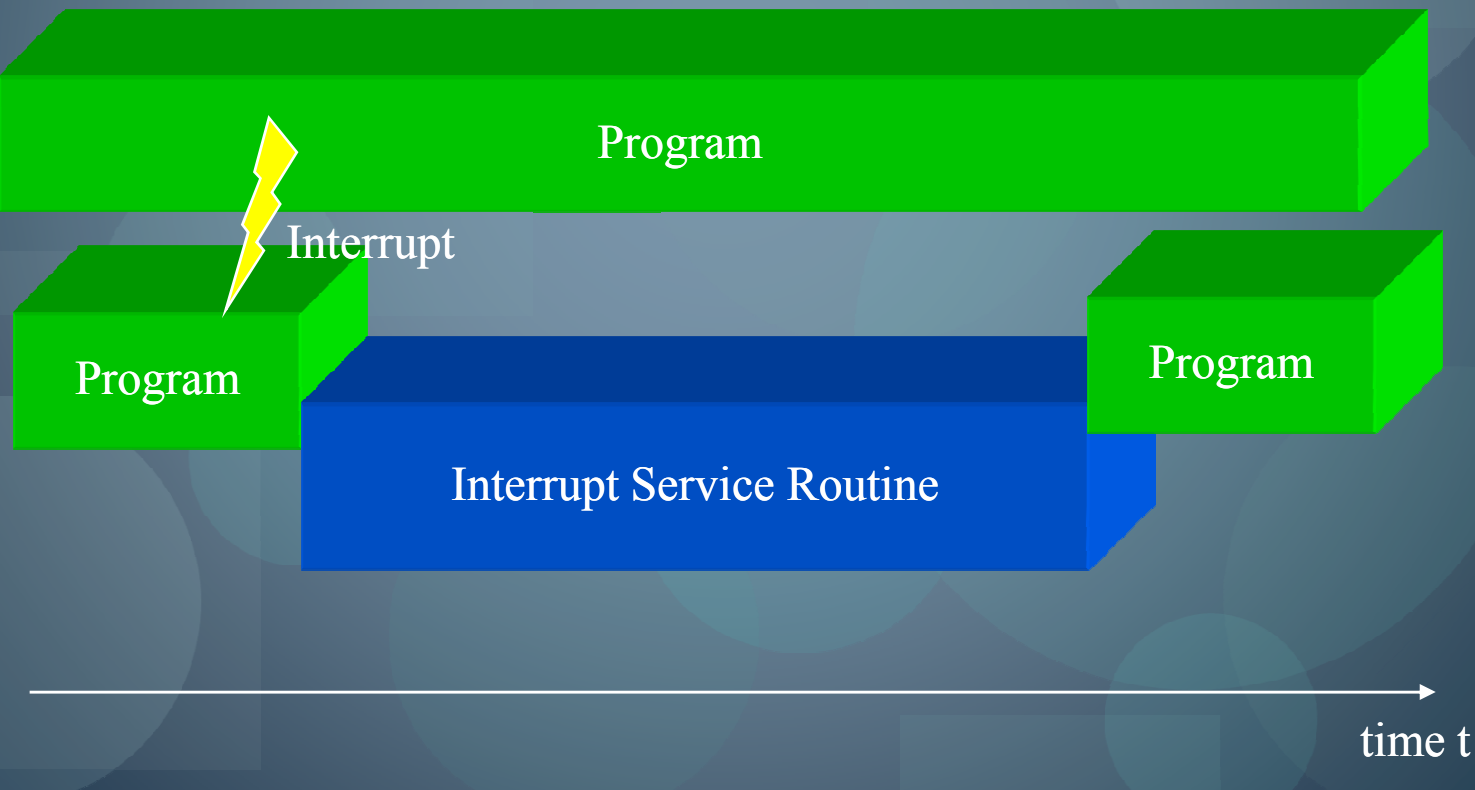
- Basic form of synchronization.
- Requires the processor to ask the device whether the operation has been completed before sending next operation.
- Advantage of Programmed I/O devices: inexpensive.
- Disadvantage of Programmed I/O devices: Computational Overhead.

# Interrupt Driven I/O

- I/O Device Hardware: Operate independent ly from the control of a processor.
- Bus Architecture: Bus must support two way communication between processor and device.
- Processor Architecture: Process to stop normal program temporarily and handle the device.
- Programming Paradigm:  Synchronous  and Asynchronous style!

# Definition

- *Event that disrupts the normal execution of a program and causes the execution of special instructions*



# INTERRUPTS

- An interrupt is any service request that causes the CPU to stop its current execution stream and to execute an instruction stream that services the interrupt

- When the CPU finishes servicing the interrupt, it returns to the original execution stream at the point where it left off.

# INTERRUPTS CAN BE REQUESTED FROM ANY OF THE FOLLOWING SOURCES:

- Hardware driven (Focus on this)
- Software interrupts (programmed interrupt requests (PIRs))

a program using interrupts is usually structured as follows

Ex:

```
    .data
theString:
    .space 64
    .text
main:
    li      $v0, 8
    la      $a0, theString
    li      $a1, 64
    syscall
    jr      $ra
```

- Exceptions, such as page faults

# Interrupts & Fetch-Execution

- Repeat Forever{

  Test: If any device has requested interrupt, handle and then continue with next iteration.

Fetch: Access the next step of the program from the location where the program is stored.

Execute: Perform the next step of the program

}

# Interrupt Request (IRQ)

- An interrupt is a signal from one part of the computer to the processor indicating that a service or special action be taken that only the CPU can perform.
- When a device needs the CPU to perform a task, transfer data from memory, issue an I/O, etc., it signals the CPU using its IRQ line.
- Each device is assigned a specific IRQ number so that the processor knows the device to which it needs to respond.

# Interrupt Request (IRQ) (Cont.)

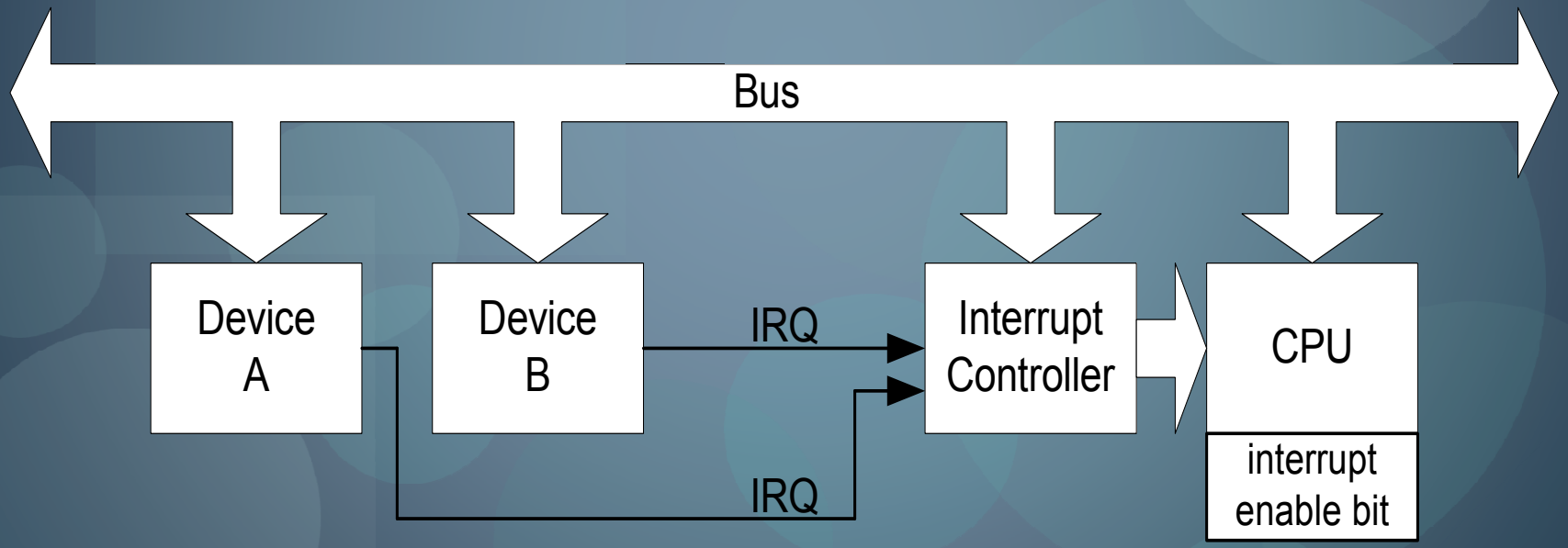
- Interrupt requests are sent to a special system component, called an *interrupt controller*.
- The interrupt controller receives and verifies requests and passes them on to the processor.

# Handling An Interrupt

- Save the current execution state.
- Determine which device interrupted.
- Call the procedure that handles the device.
- Clear the interrupt signal on the bus.
- Restore the current execution state.

# Interrupts: Overview

- Complex hardware setup
- Needed for multitasking/multiprogramming OS

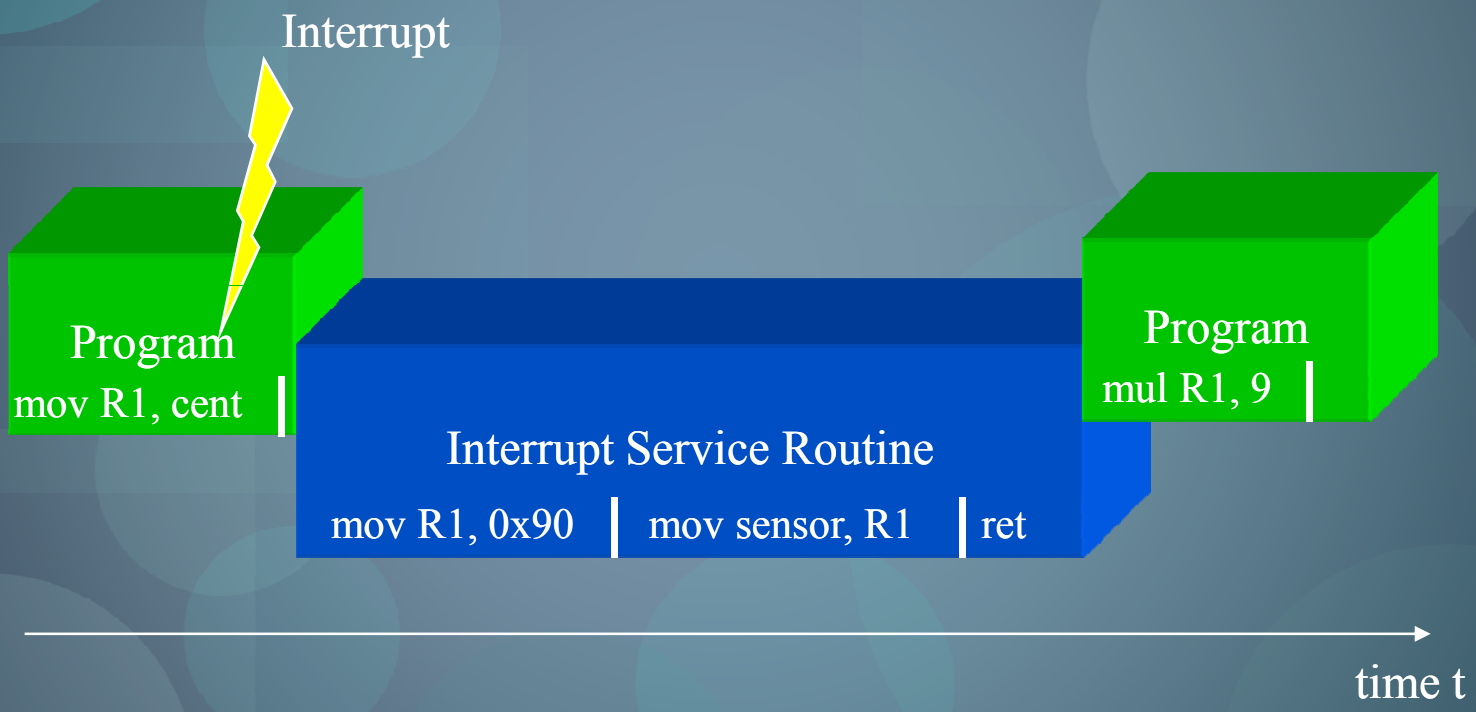


# Interrupts : An example

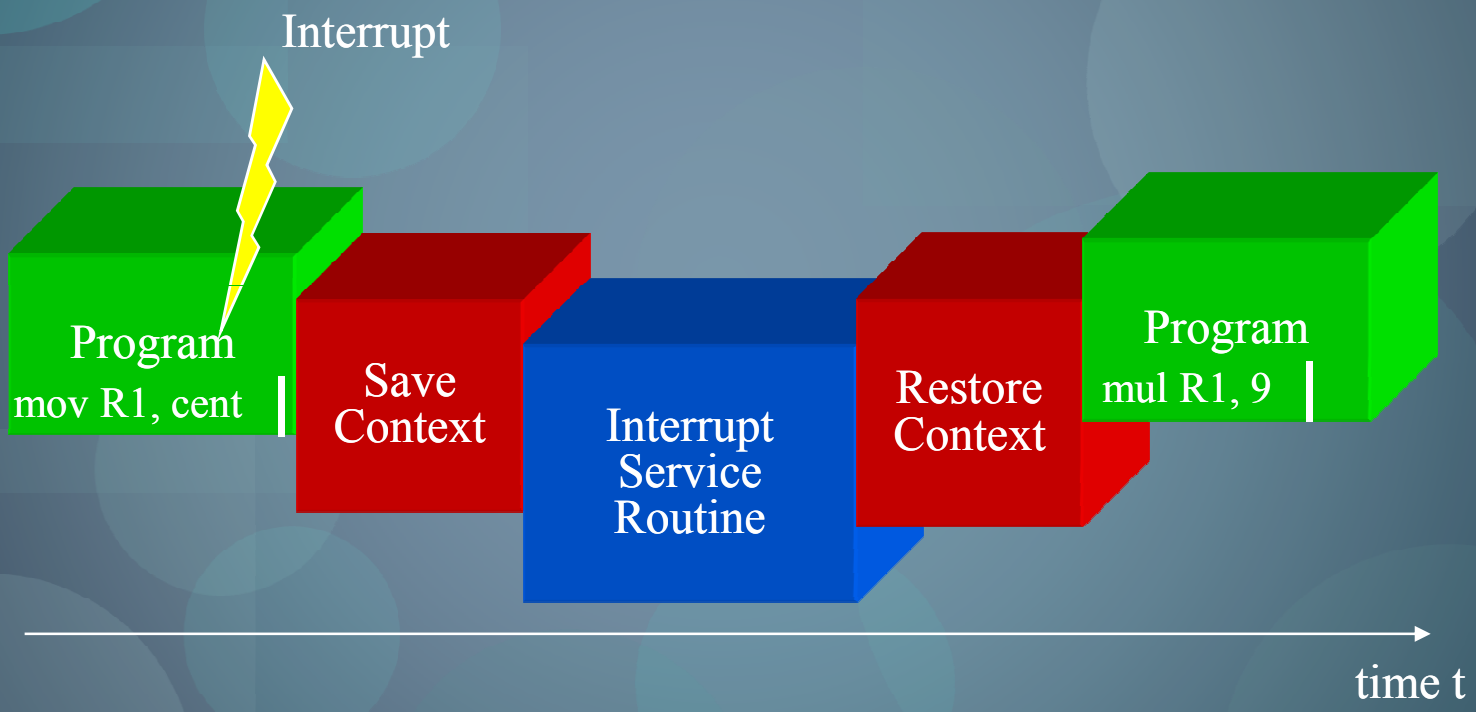
$$Far.= (cent * \frac{9}{5}) + 32$$



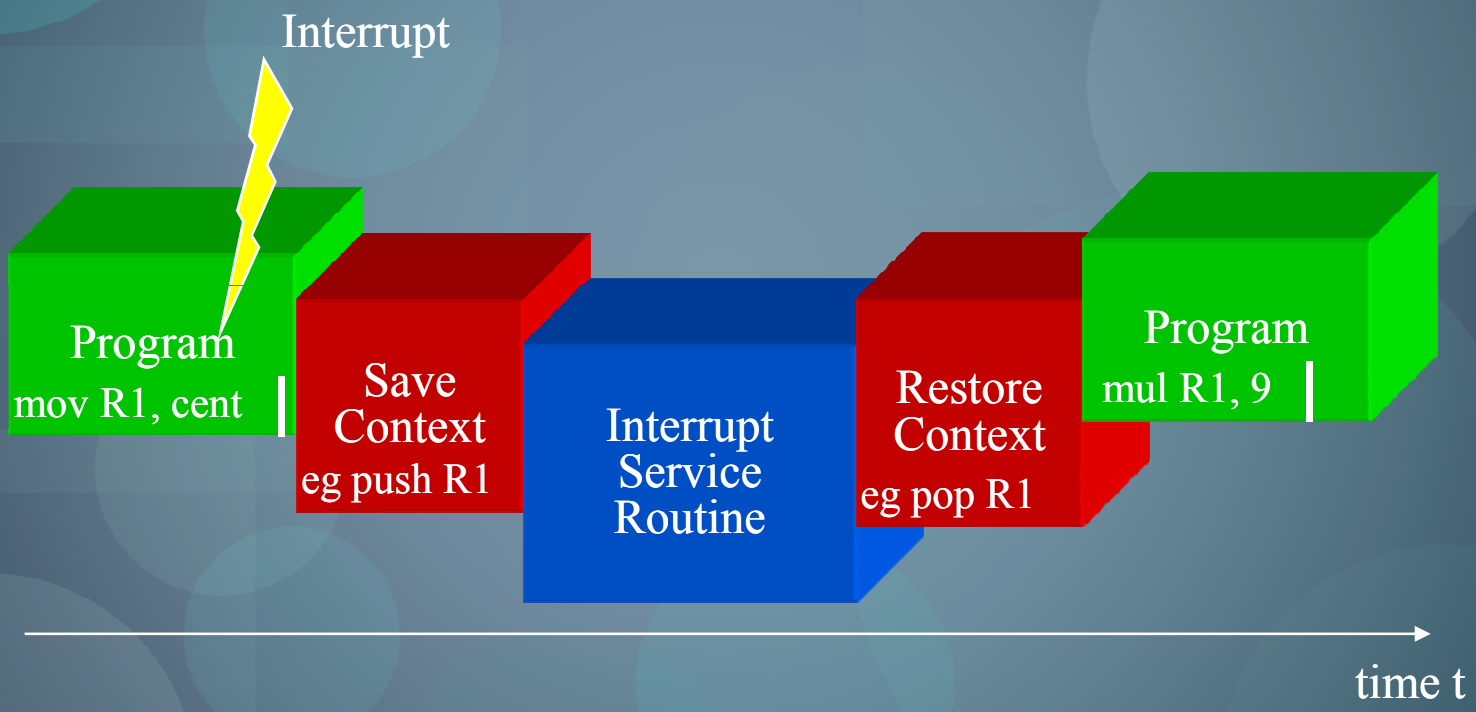
# Interrupts


Interrupt
Program
mov R1, cent
Interrupt Service Routine
mov R1, 0x90
mov sensor, R1
ret
Program
mul R1, 9
time t

# Interrupts


Interrupt
Program
mov R1, cent
Save
Context
Interrupt
Service
Routine
Restore
Context
Program
mul R1, 9
time t

# Interrupts


Interrupt
Program
mov R1, cent
Save
Context
eg push R1
Interrupt
Service
Routine
Restore
Context
eg pop R1
Program
mul R1, 9
time t

# When an Interrupt Occurs

- Finish the current instruction
- Save minimal state information on stack
- Transfer to the interrupt handler, also known as the interrupt service routine (ISR)

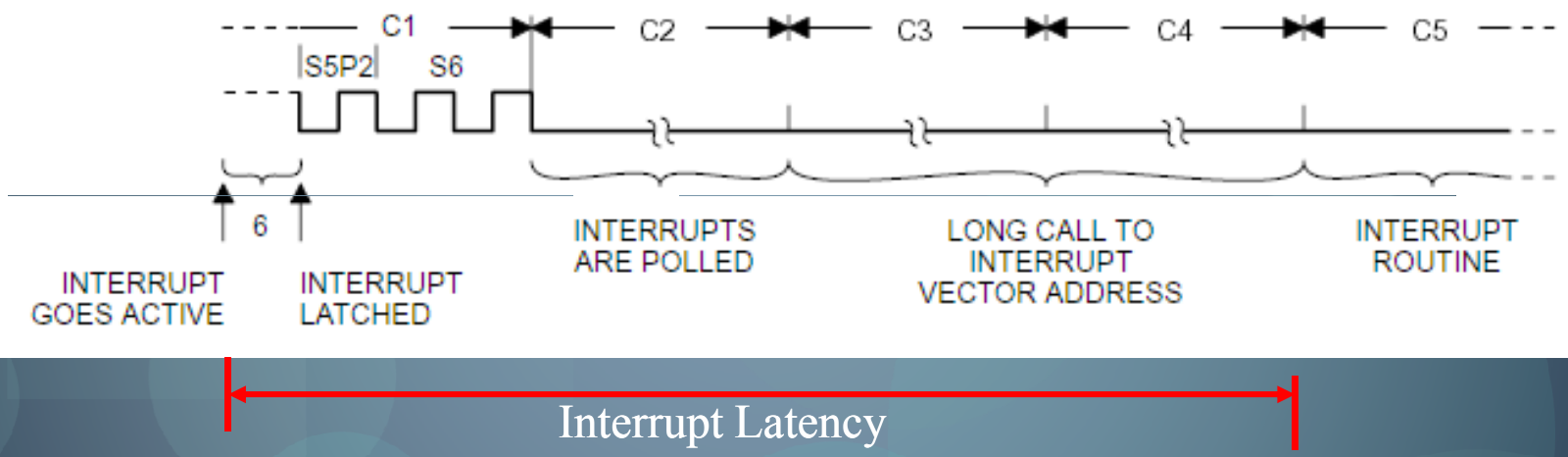
  But there is more to it than this…How do we know which device interrupted?
- And what happens if two (or more) devices request an interrupt at the same time?

# Control & Status Registers

- They are set of addresses a device uses.
- Control Register: Corresponds to a contiguous set of addresses that respond to a store operation.
- Status Register:  Corresponds to a contiguous set of addresses that respond to a fetch.

# Interrupt Response Time

**Figure 25.** Interrupt Response Timing Diagram




Interrupt Response Time= Interrupt Latency + Time in Interrupt Routine

# Advantages of Interrupts

- I/O  is important aspect for communicating.
- Interrupt driven I/O automatically overlaps computation and the programmer need not take any special action.
- Interrupt adapts to the processor speed and I/O device automatically.(Never underestimate or overestimate).

# Summary

- An interrupt is any service request that causes the CPU to stop its current execution stream and to execute an instruction stream that services the interrupt
- When the CPU finishes servicing the interrupt, it returns to the original execution stream at the point where it left off.