I/O organization and I/O communication

How I/O devices are addressed and communicate.

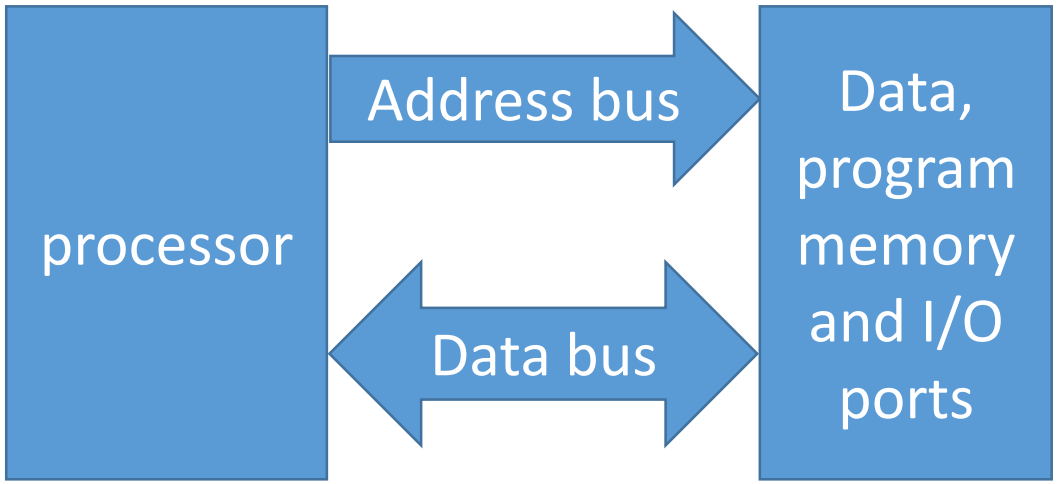
Programmed IO

Interrupt I/O

DMA

I/O mapping

- How to differentiate between I/O address and memory address?
- How 8051 is doing it?
- We have 2 options while designing the processor
  - Shared address space
  - Dedicated address space
  - Each processor supports one of the options



I/O mapping

- Shared address space
  - Same instruction for both memory and I/O access
  - Also called as memory mapped I/O
  - Will have same size address bus for both I/O and memory address

- Dedicated address space
  - Separate instructions for memory and I/O access
  - Also called as I/O mapped I/O or standard I/O
  - Usually will have 8 bit addresses; sufficient

Ports and I/O addresses

- Port is an access point where we connect I/O devices
- Every port will have an address
- I/O device address will be the address of the port that is used for accessing it
  - If we connect keyboard to port-A then keyboard address is the same as port-A's address ; for example 8020H
- Port addresses are decided during design of the hardware system (or the processor) and they remain fixed

Standard I/O vs Memory-Mapped I/O (Examples)

Standard I/O:

- IN yyyy
- OUT yyyy
- yyyy is the address of the I/O port.
- Data will flow between the port address and accumulator.

Memory-Mapped I/O:

- LDA yyyy
- STA yyyy
- yyyy is the address of the I/O port.
- Data will flow between the port address and accumulator.

- MOV Rn, yyyy
- MOV xxxx, Rn

Methods of I/O communication

1. Programmed I/O; Use a program for I/O communication
   I. Unconditional
   II. Conditional

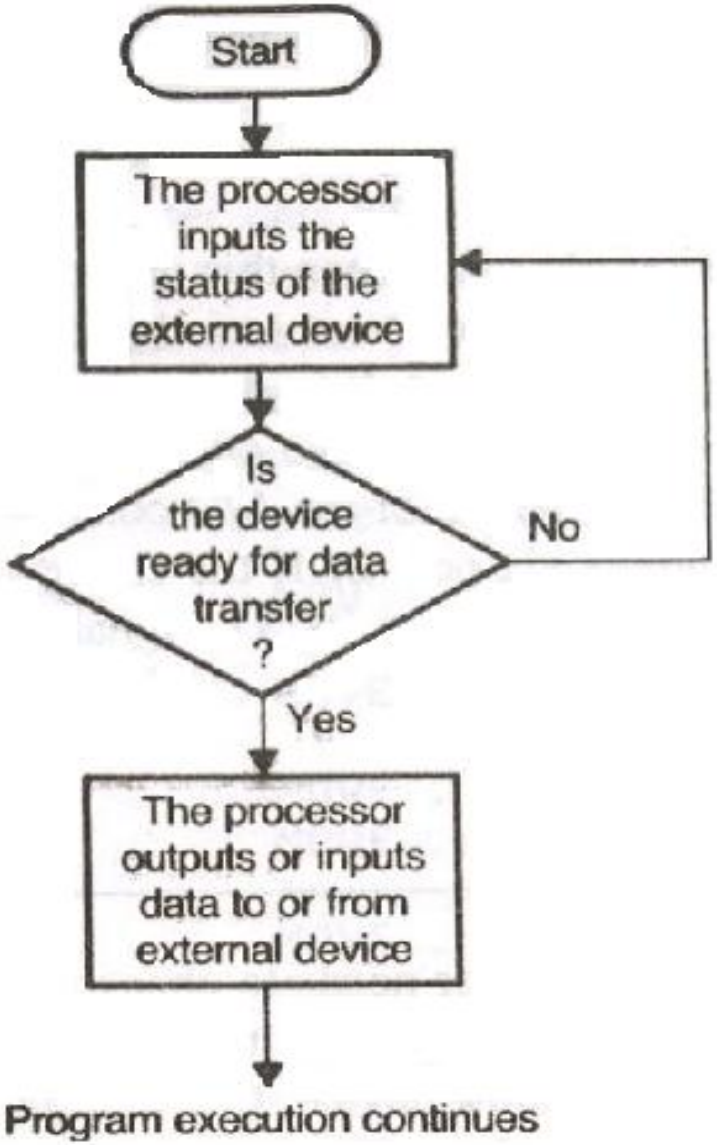
2. Interrupt I/O;
   will it not use a program for I/O communication?

3. DMA; use dedicated hardware for communication between I/O and memory

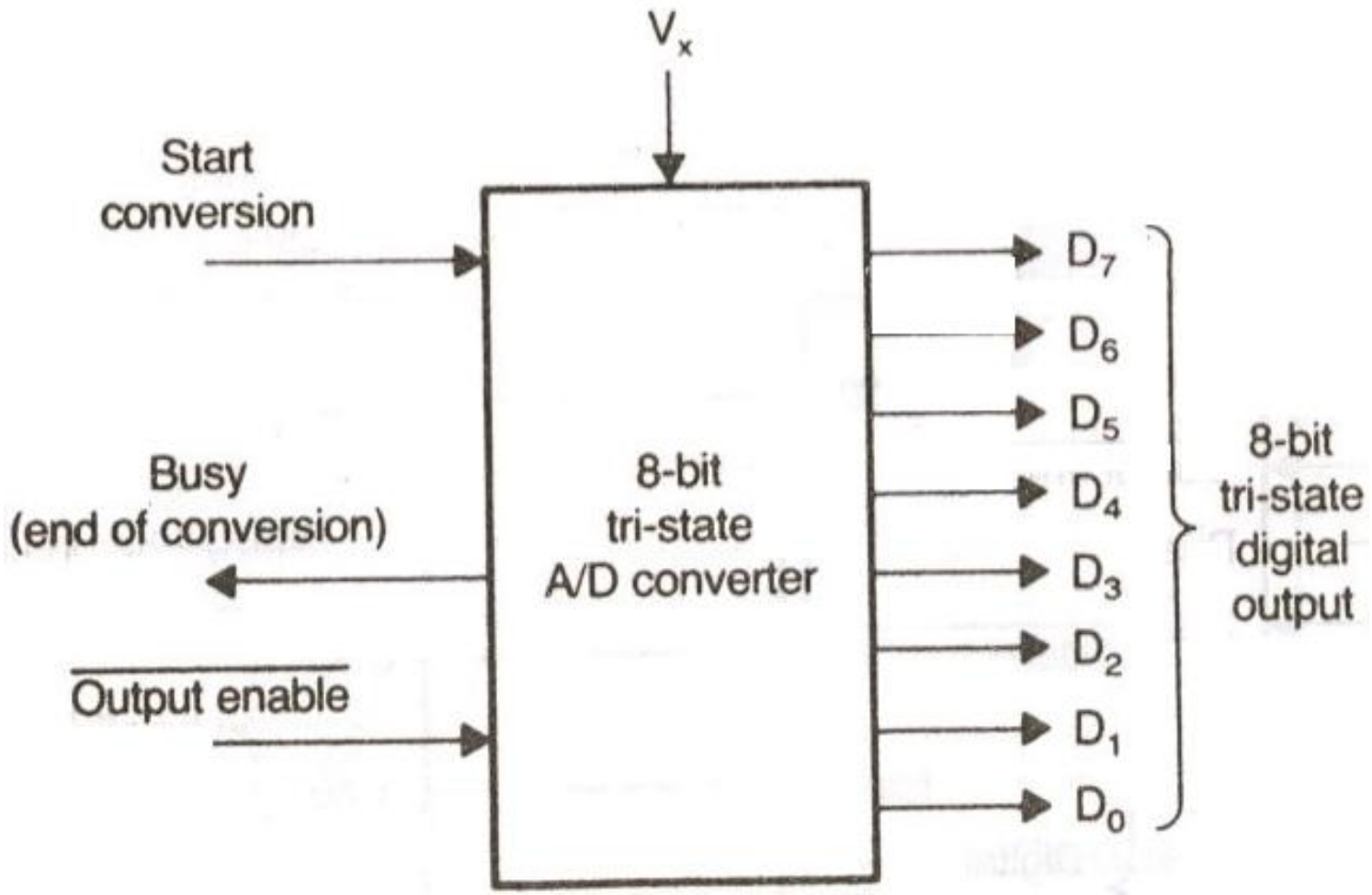
Programmed I/O

- Unconditional I/O
  - Program will not check for any conditions (status of I/O device) before initiating data transfer activity
  - LED switching on/off, DAC etc.
- Conditional I/O
  - Program will initiate data transfers only after status condition is indicating for data transfer activity
  - ADC, Keyboard etc.

Conditional Programmed I/O



Working steps of a basic ADC



Example of Conditional Programmed I/O

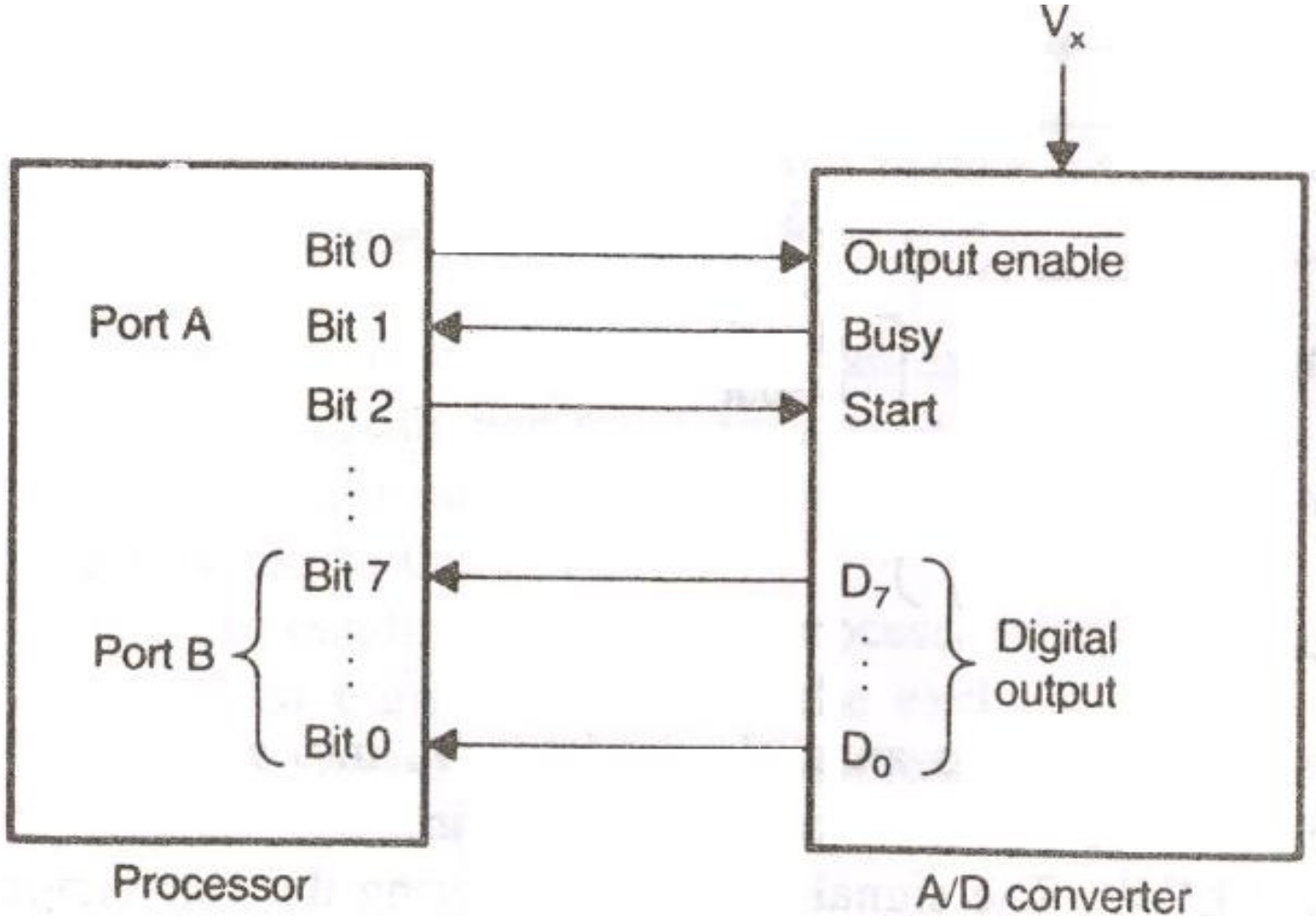
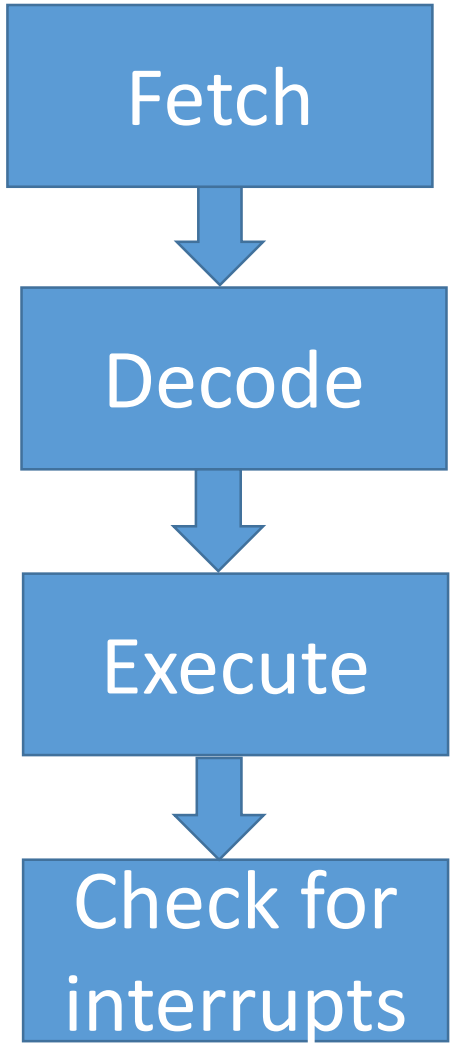


Figure: Interfacing an A/D Converter to an 8-bit Processor

INTERRUPTS

- An *interrupt* is an event (external or internal) that signals the processor to inform that the interrupting device needs its service (‘attention’).
- CPU can execute any (main) program, without wasting time by *not waiting* for the event to occur.
- quick response to events, needed for real-time control application in a multi-programming environment.

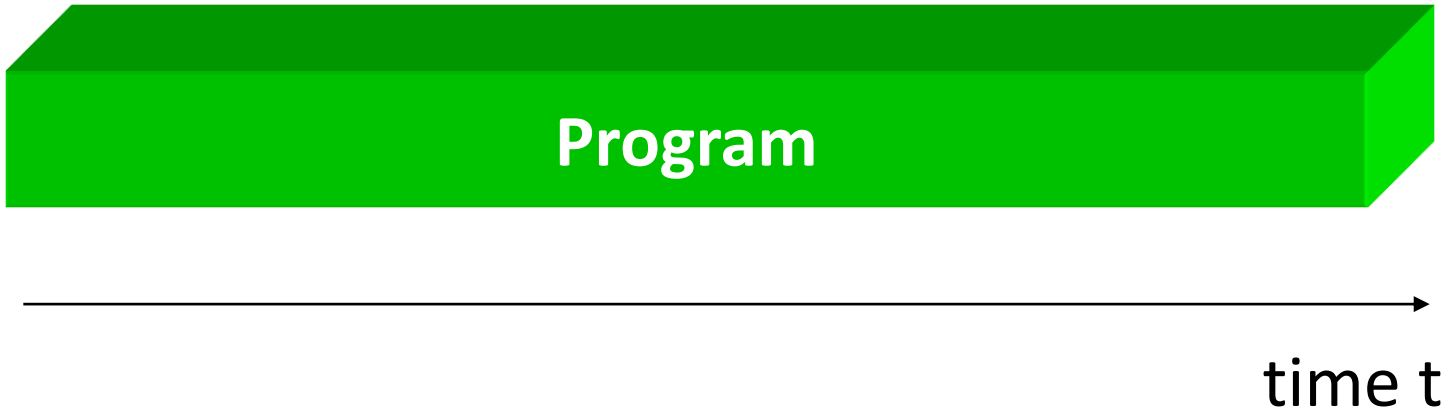
When will processor check for interrupts?



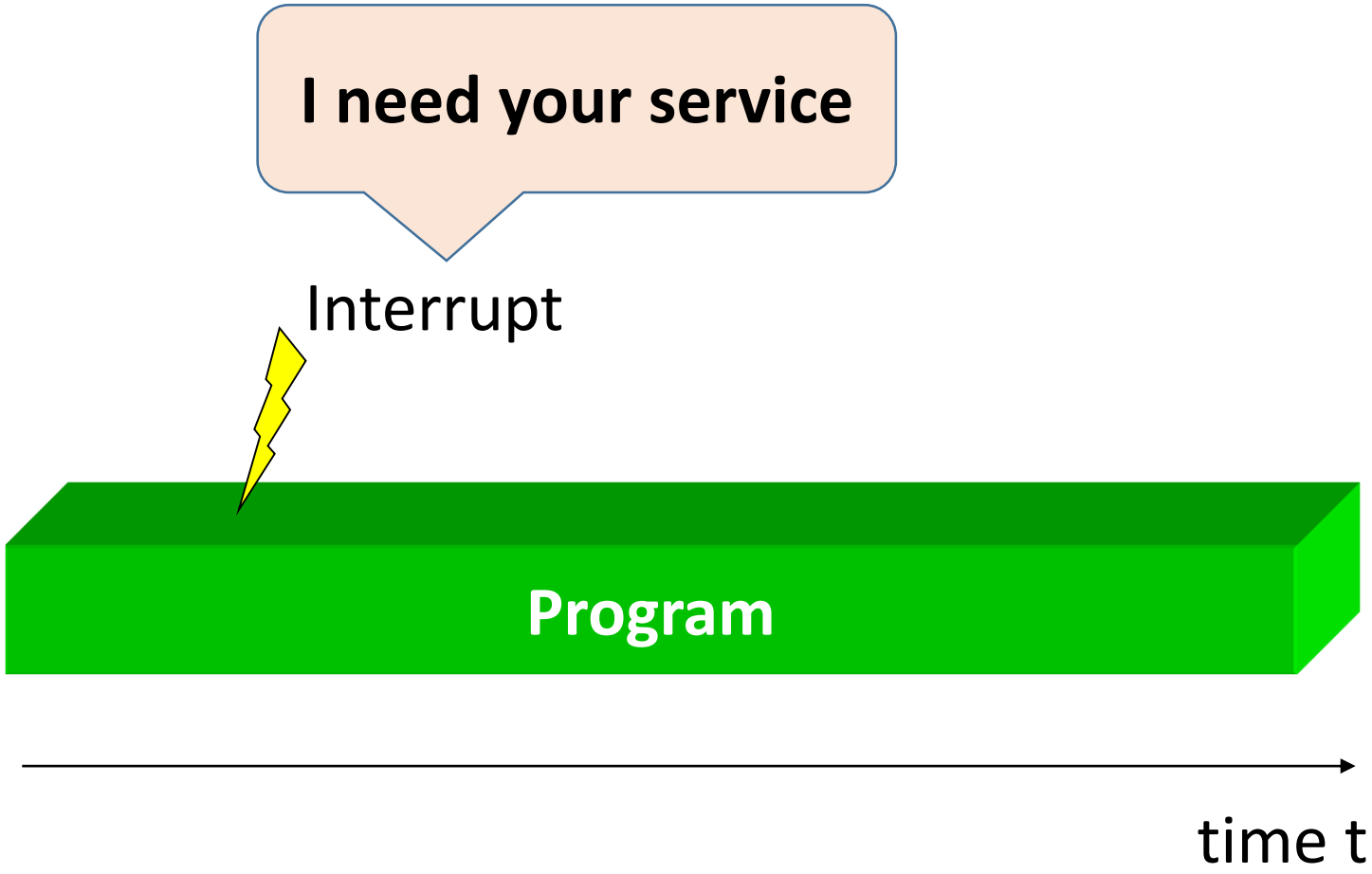
Remember the Instruction cycle?

If there is any interrupt, then start the program to service (handle) the interrupt and then come back to the program which was left on the mid way.

Interrupts

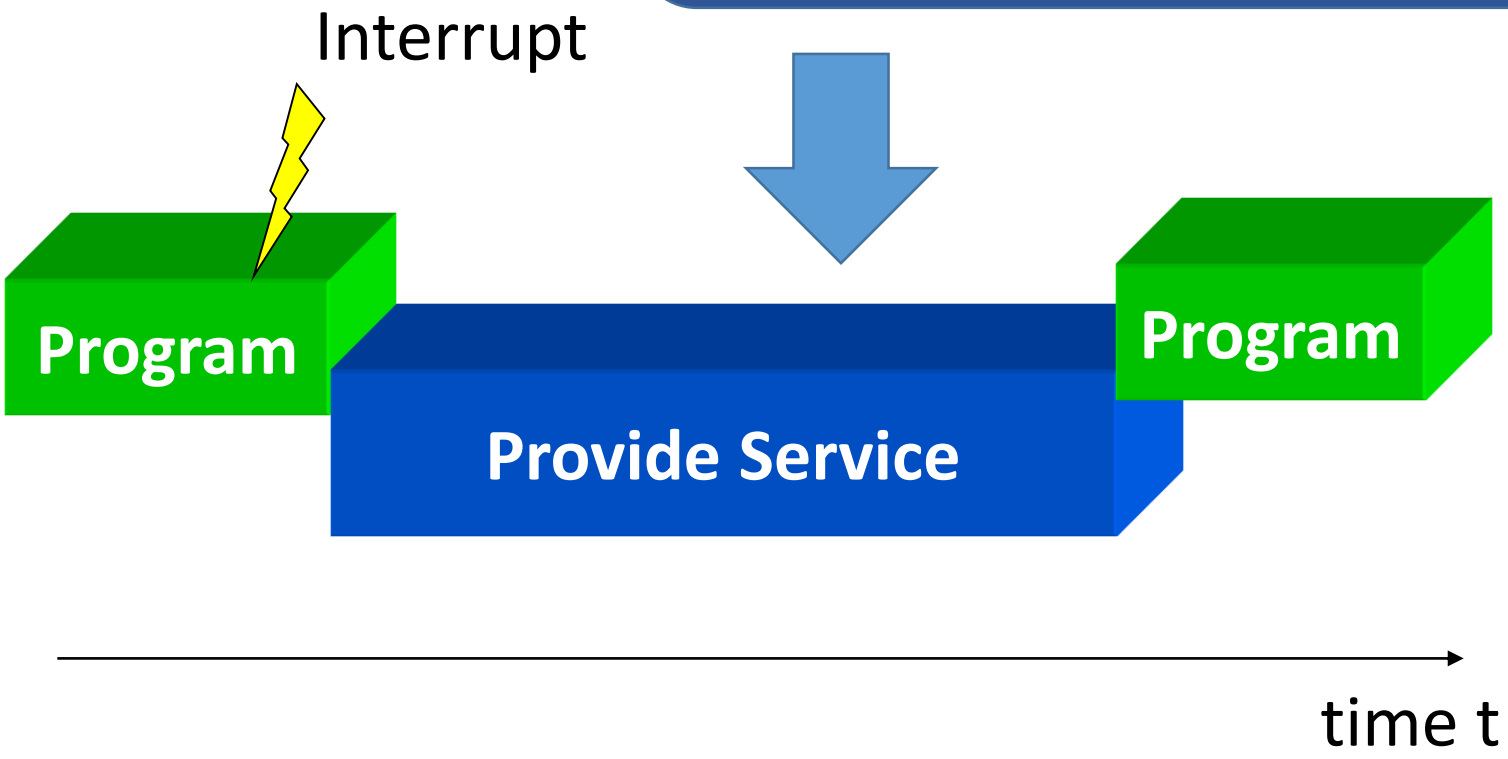


Interrupts

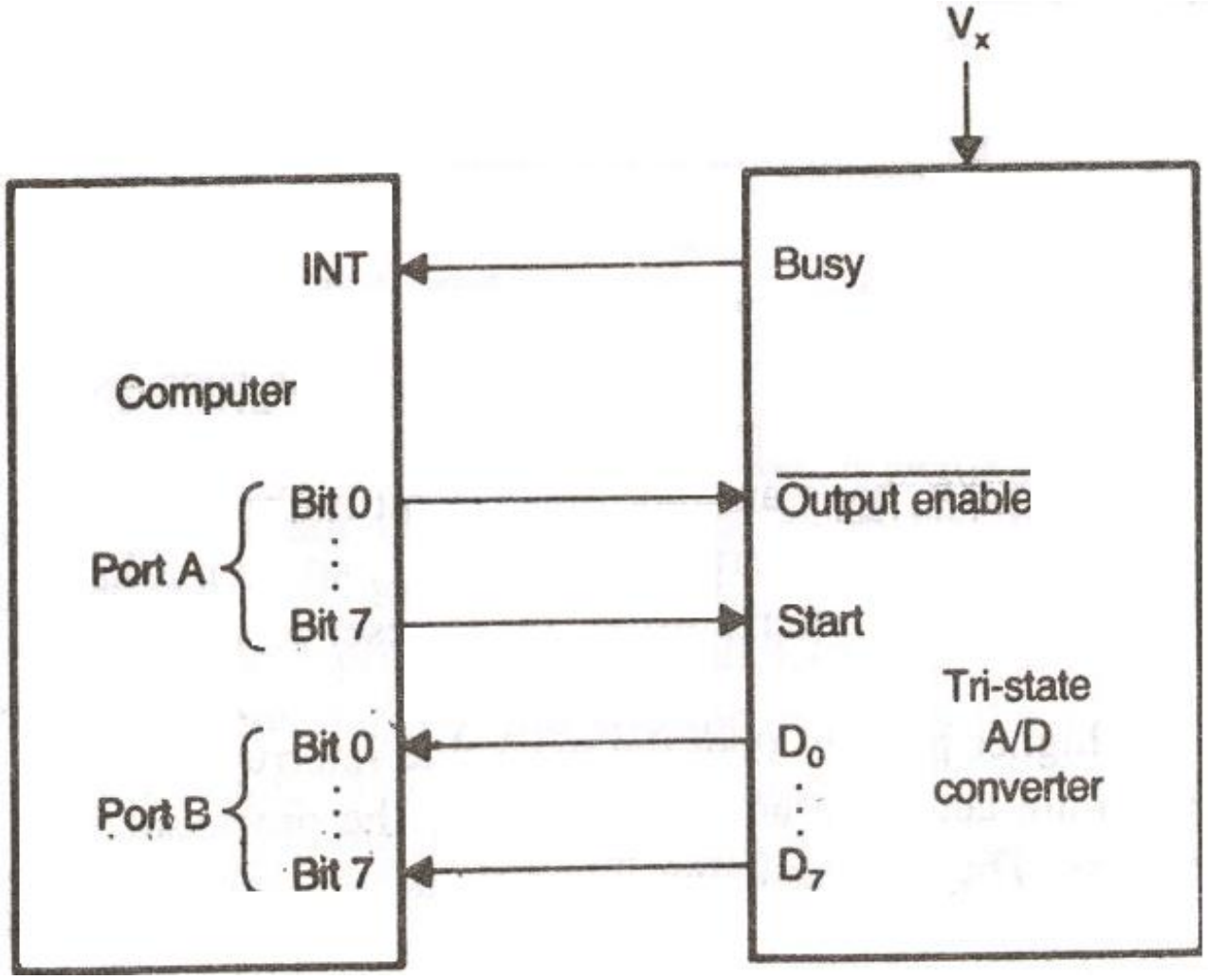


Interrupts

ISR, is a *short program* which will get executed in order to provide the service requested by the interrupting I/O device



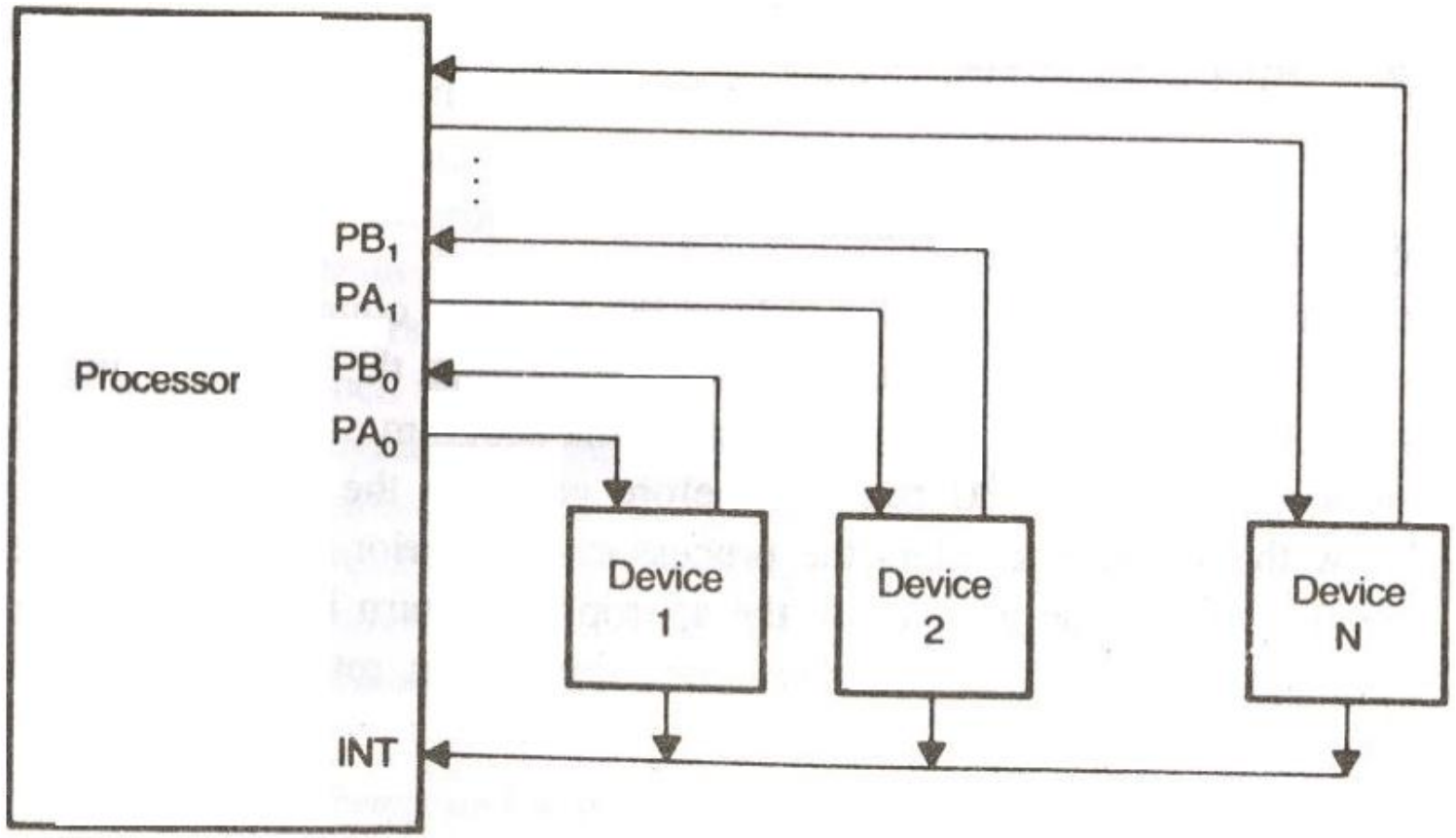
Interrupt I/O



How many interrupt signal pins should a processor have?

Can we handle all interrupts with only one INT pin?

Interrupt I/O: Polled Interrupts

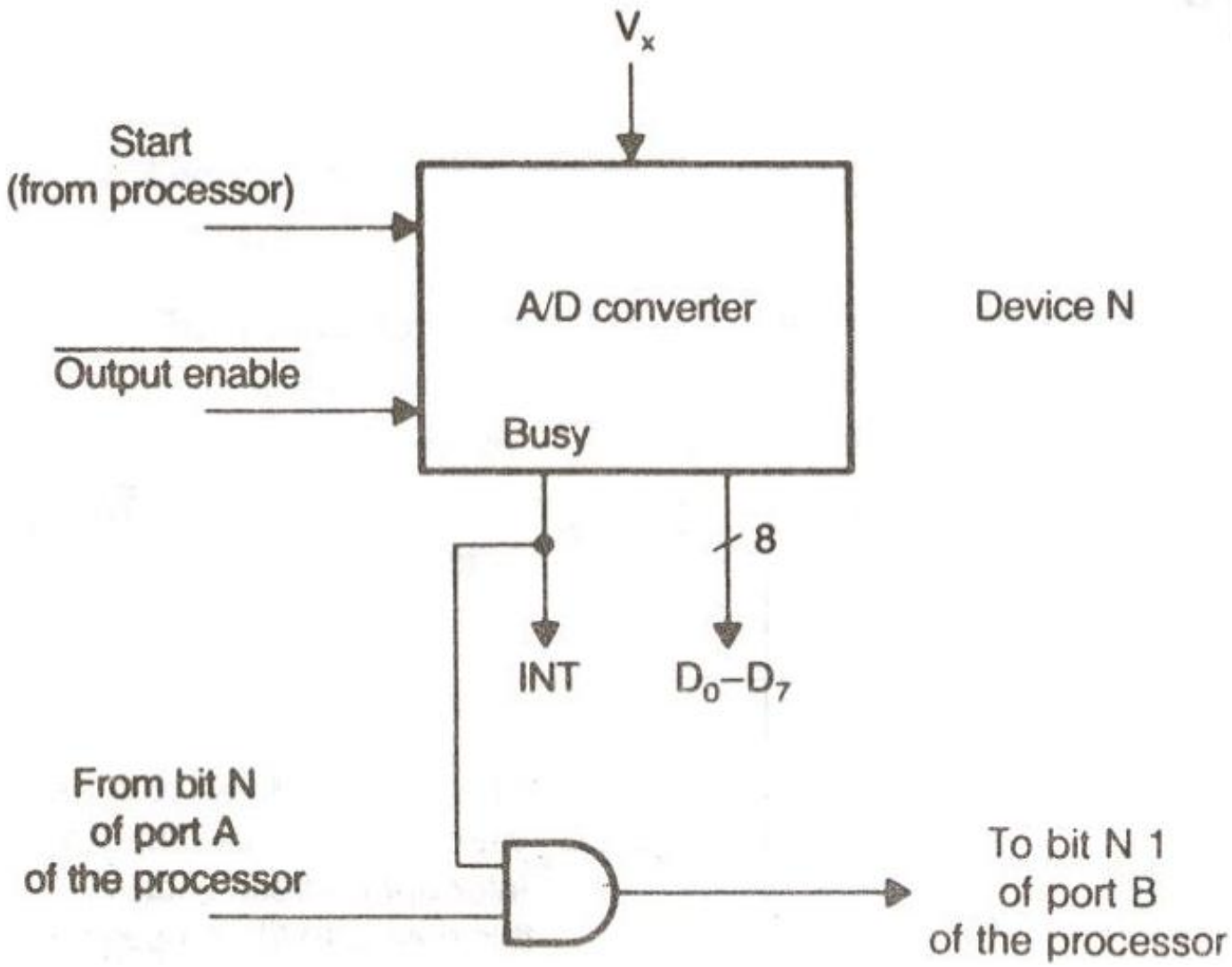


How to handle more than one device under interrupt system and processor has only one interrupt signal?

Three programs are involved in this method:

1. Main
2. Polling of interrupts
3. ISR

Interrupt I/O: Polled Interrupts

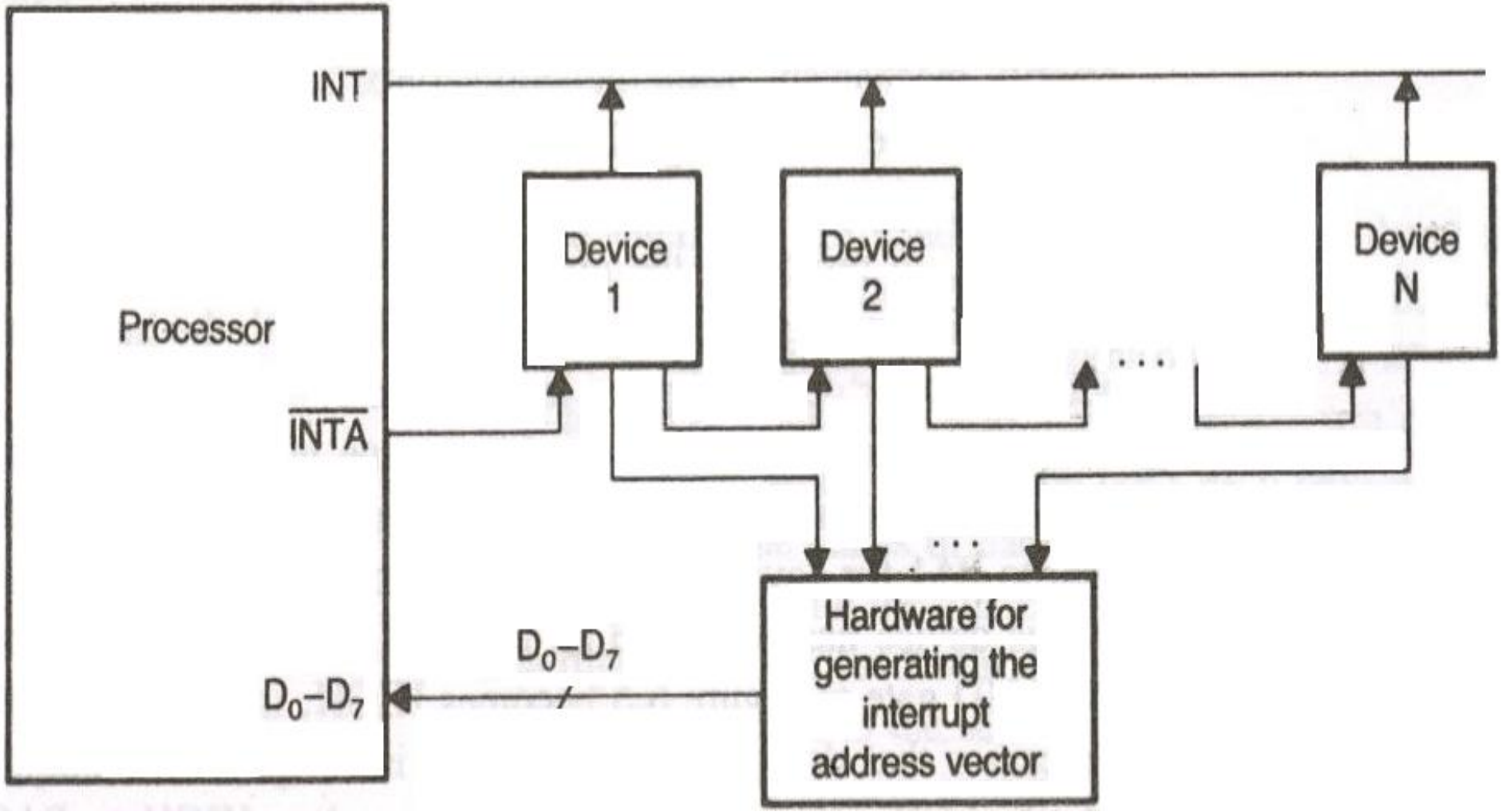


A logic '1' is sent and if logic '1' is received then that is the device which has interrupted.

This is program based polling and hence slower.

Time taken for polling itself may be too much for large number of devices.

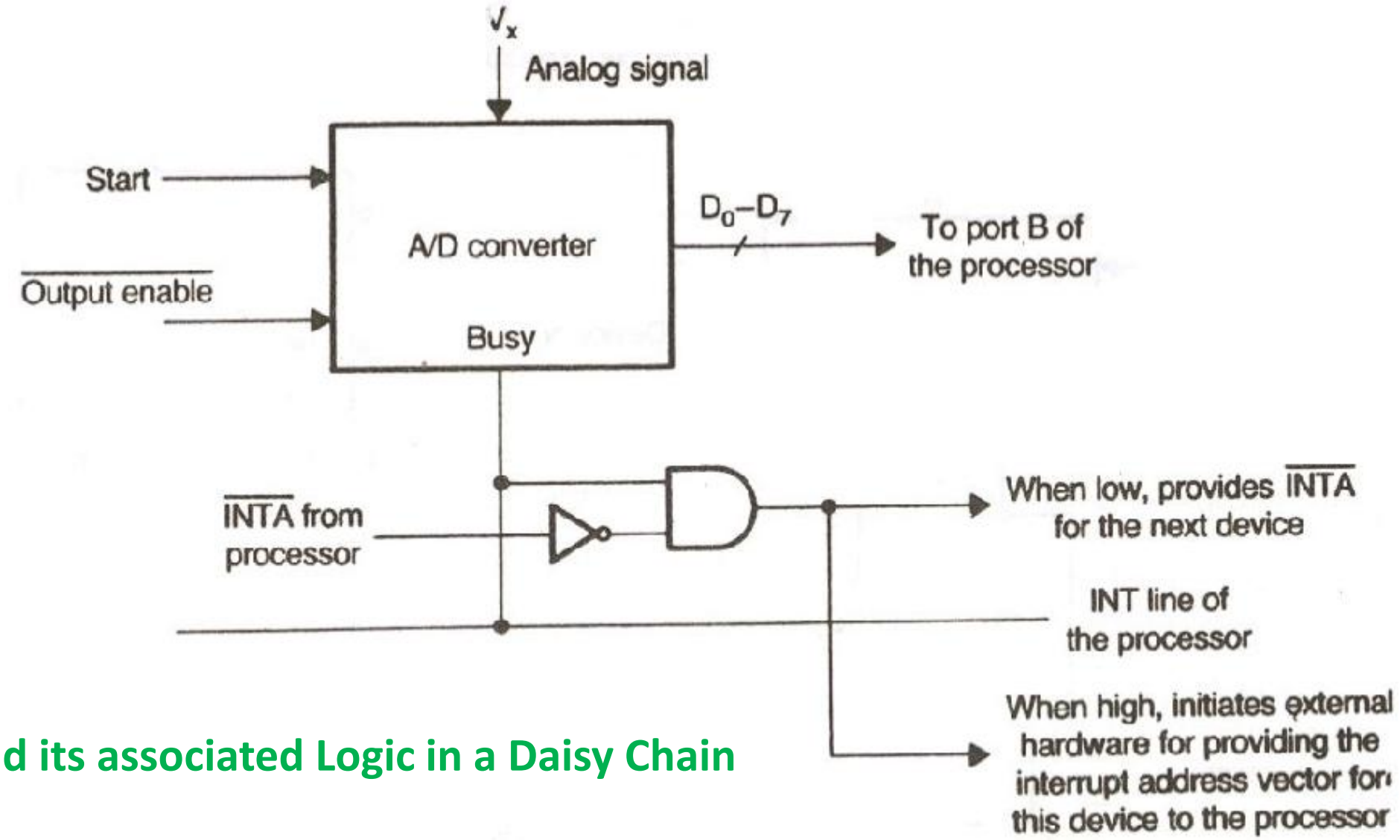
Interrupt I/O: Daisy Chain Interrupt



Hardware method of polling.

Faster than polled interrupts.

Interrupt I/O: Daisy Chain Interrupt



A device and its associated Logic in a Daisy Chain

Few comparing points

- Daisy chain has fixed priority
  - High priority is given to *electrically* closest device to $\overline{\text{INTA}}$ signal
- Fast as there is no code that needs to run

- Polled interrupt system has flexible priority
  - High priority is given to the device which is checked first by the polling program
- Slow as there is a polling program that needs to run

Few points about interrupts

- Maskable and non – maskable
- External (hardware), internal (also called as exceptions) and software interrupts (by programmer in programs)
- Software interrupts are similar to CALL and RET
- Vector address (it’s a bit sequence or an ID); specifies ISR address
- Vectored and non-vectored interrupts
- Nesting of interrupts
- Use EI and DI instructions to avoid nesting

Interrupt Overheads

Interrupt arrives

Complete current instruction execution

Save essential register information

Vector to ISR

Interrupt Latency

Save additional register information

Execute core body of ISR

Restore additional register information

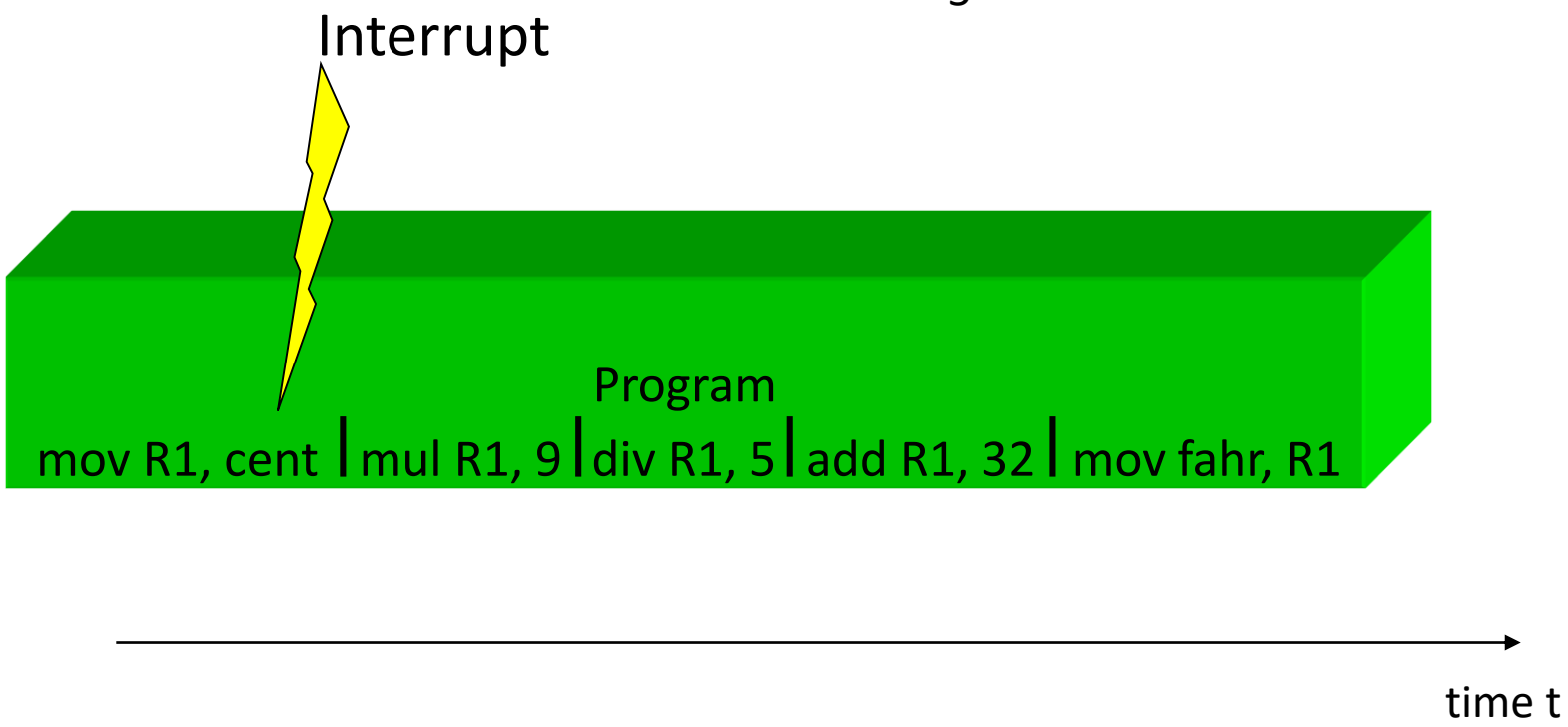
restore essential registers and return from interrupt

Resume task

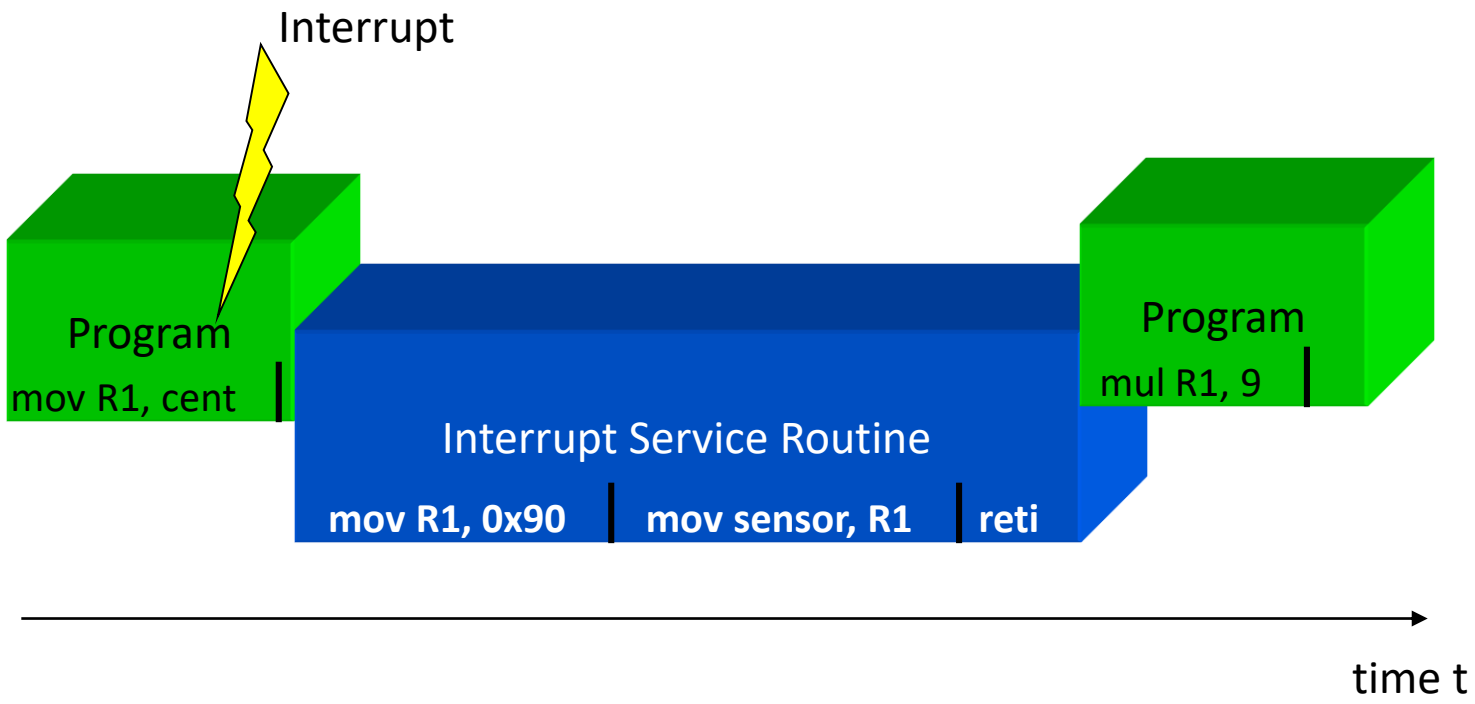
Interrupt Termination

Interrupts

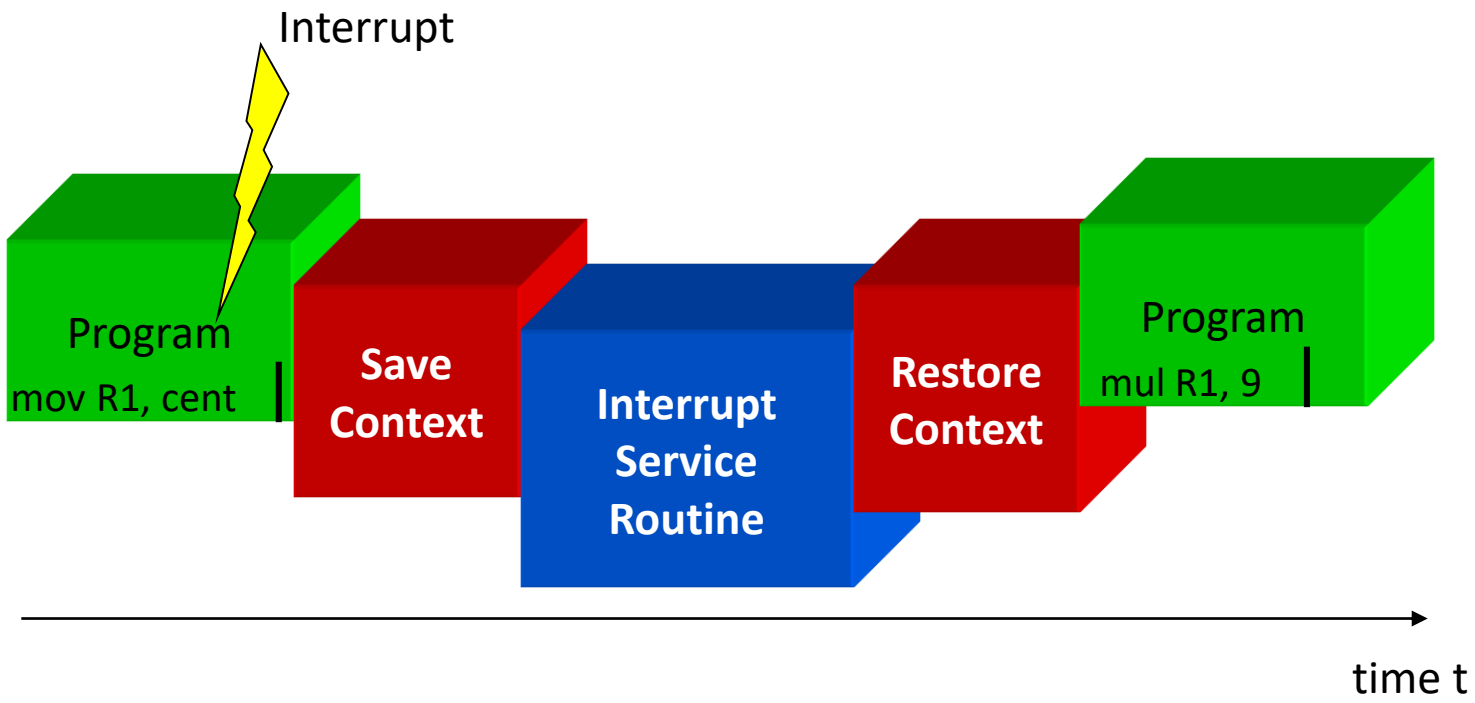
$$fahr = (cent * \frac{9}{5}) + 32$$



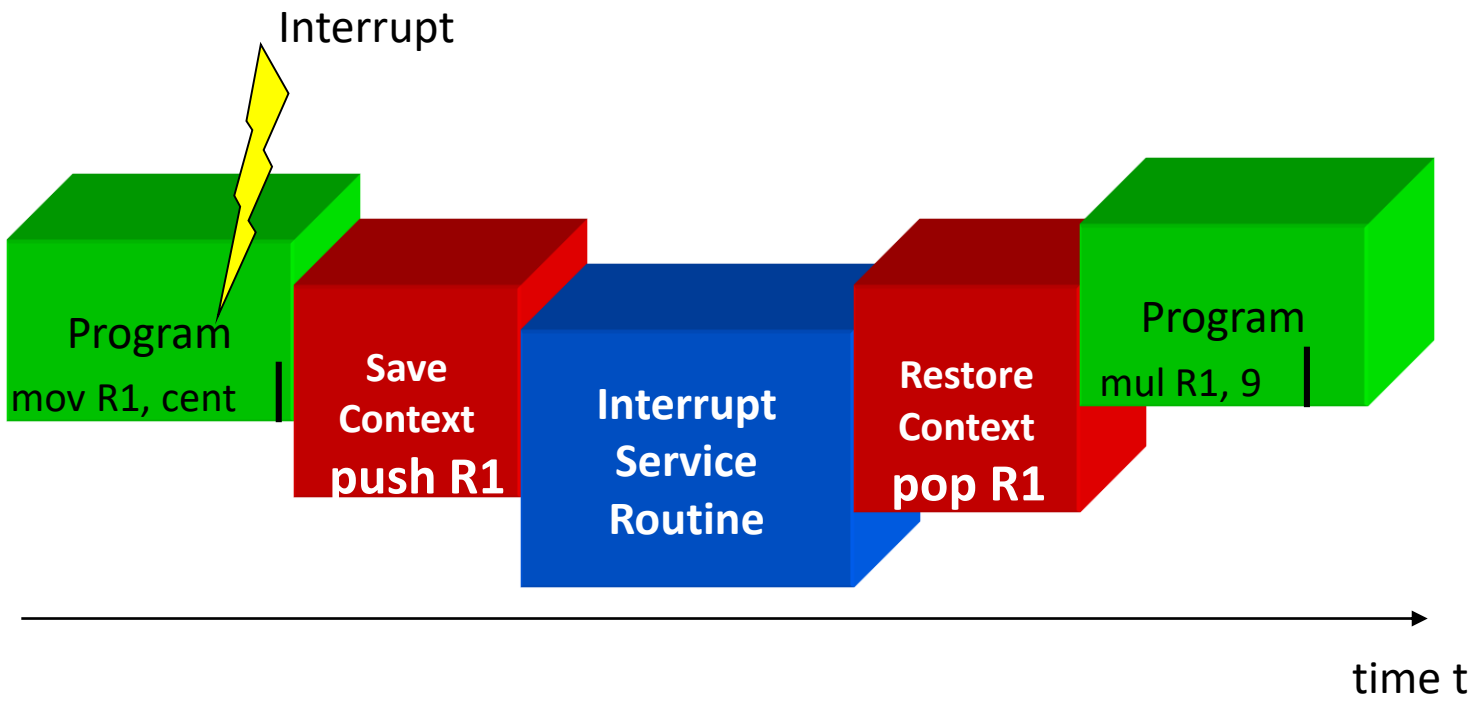
Interrupts


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

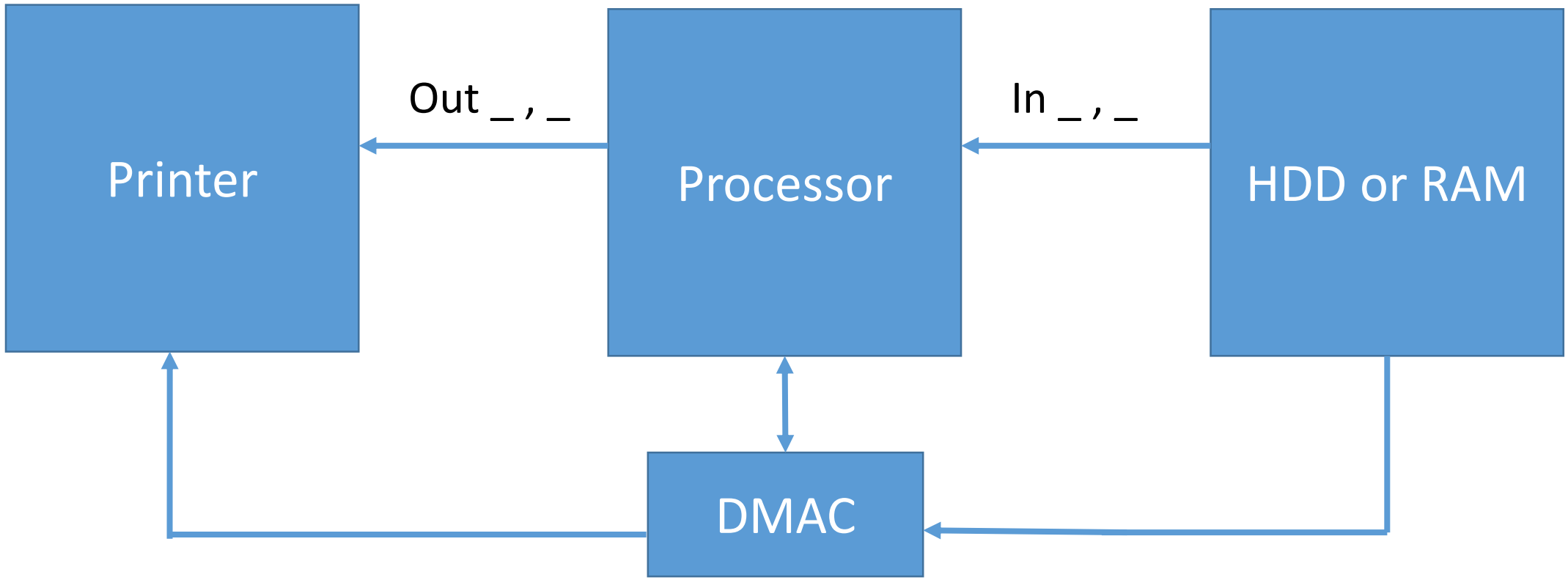
Interrupts


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
push R1
Interrupt
Service
Routine
Restore
Context
pop R1
Program
mul R1, 9
time t

Need for Direct Memory Access, DMA



DMA concept and few terms

- Bus master and slave devices
- Bus request and bus grant
- Tristate or disconnect from the bus

DMA basic concept