

# DEVICE DRIVERS AND INTERRUPTS SERVICE MECHANISM

Lesson-1: Programmed I/O (*busy and wait*) method for ports and devices, and the need for interrupt driven IOs

## Programmed IOs approach for IO ports

- Processor continuously busy in executing the program related to input or output from the port and waits for the input ready or output completion

# Programmed IOs approach for IO from Devices

- Processor continuously busy in executing the program related to device functions and waits for the device status ready most of the time or function completions

## Ports A and B with no interrupt generation and interrupt service (handling) mechanism

- Port A be in a System, and port B be its modem input which puts the characters on the output.
- Let *In\_A\_Out\_B* be a routine that receives an input-character from Port A and re-transmits the character to Port B output.

## *In\_A\_Out\_B* routine

- Has to cyclically call the following steps *a* to *e* and executes the cycles of functions *i* to *v*, thus ensuring the modem port A does not miss reading the character

## Steps a, b and c

- Step *a*: Function *i*— Check for a character at port A, if not available, then wait
- Step *b*: Function *ii*— Read Port A byte (character for message) and return to step *a* instruction, which will call function *iii*
- Step *c*: Function *iii*— Decrypt the Message and return to step *a* instruction, which will call function *iv*

## Steps d and e

- Step *d*: Function *iv*— Encode the Message and return to step *a* instruction, which will call function *v*.
- Step *e*: Function *v* — Transmit the encoded message to Port B and return to step *a* last instruction, which will start step *a* from beginning

## *In\_A\_Out\_B* routine

- Call function *i*
- Call function *ii*
- Call function *iii*
- Call function *iv*
- Call function *v*
- Loop back to step 1



## Step *a*

- Does Polling— Polling a port means to find the status of the port— whether ready with a character (byte) at input or not.
- Polling must start before time  $T_{char} \mu s$  when characters are expected at  $(1/T_{char})$  MB per s.

## Condition in which no character misses

- If the program instructions in four steps b, c, d and e (functions  $ii$  to  $v$ ) take total running time of less than  $T_{char} \mu s$  then the above programmed IO method works

## Problems with Programmed IOs approach from ports and devices

1. (a) The program must switch to execute the *In\_A\_Out\_B* cycle of steps *a* to *e* within a period less than  $T_{char}$ . (b) Programmer must ensure that steps of *In\_A\_Out\_B* and any other device program steps never exceed this time.

## Problems with Programmed IOs approach from portas

2. When the characters are not received at Port A in regular succession, the waiting period during step *a* for polling the port can be very significant.
3. Wastage of processor times for the waiting periods is the most significant disadvantage of the present approach

# Problem with Programmed IOs approach from ports and devices

4. When the other ports and devices are also present, then programming problem is to poll each port and device, and ensure that program switches to execute the *In\_A\_Out\_B* step *a* as well as switches to poll each port or device on time and then execute each service routines related to the functions of other ports and devices within specific time intervals such that each one is polled on time.

## Problems with Programmed IOs approach from ports and devices

5. The program and functions are processor and device specific in the above busy-wait approach, and all system functions must execute in synchronization and timings are completely dependent on periods of software execution

## IO based on an interrupt from Port A

- Instead of continuously checking for characters at the port A by executing function (i) we can first call step a when a modem receives an input character, sets a status bit in its status register and interrupts Port A. The interrupt should be generated by port hardware.
- In response to the interrupt, an interrupt service routine `ISR_PortA_Character` executes— an efficient solution in place the wait at step a (poll for input character)

# Application of programmed IOs

- In case of single purpose processor and dedicated IOs or device functions with continuous device status polling



# Summary

## We learnt

- Programmed IOs from ports and devices used when processor is busy with the IO or device functions only and waits for the port or device status ready for sending or receiving data and
- Useful for single purpose processor for the IOs and device-functions
- Else wastage of processor times for the waiting periods (for polling for input status)—\_most significant disadvantage
- Interrupt generation alternative to wait for the device or port status

End of Lesson 1 of Chapter 6  
on  
Programmed I/O (*busy and wait*)  
method for ports and devices, and the  
need for interrupt driven IOs