Chapter 11: Input/Output Organisation

Lesson 07:

Interrupts- Part 1

Objective

- Understand the mechanism of Interrupt driven IOs and handling of multiple interrupts
- Learn concept of Interrupt service on interrupt request by a hardware signal

Need for Interrupt based IO mode

Three modes of transfer of device data, commands and status

(i) Programmed IO

(ii) Interrupt driven IO

(iii) Direct memory access (DMA)

Disadvantage of programmed I/O mode

- A program has to wait and repeatedly tests the status; Waiting period for an asynchronous event can be too large
- Many I/O devices generate *asynchronous* events─ events that occur at times that the processor cannot predict or control, but which the processor must respond to reasonably quickly to provide acceptable performance

Example of Unpredictable wait events

- Keyboard on a workstation or PC
- The processor cannot predict when the user will press a key but must react to the key-press in well under a second or the response time will be noticeable to the user
- The programmed I/O mode therefore not appropriate due to prolonged wait states

Interrupt based IO mode

Interrupts

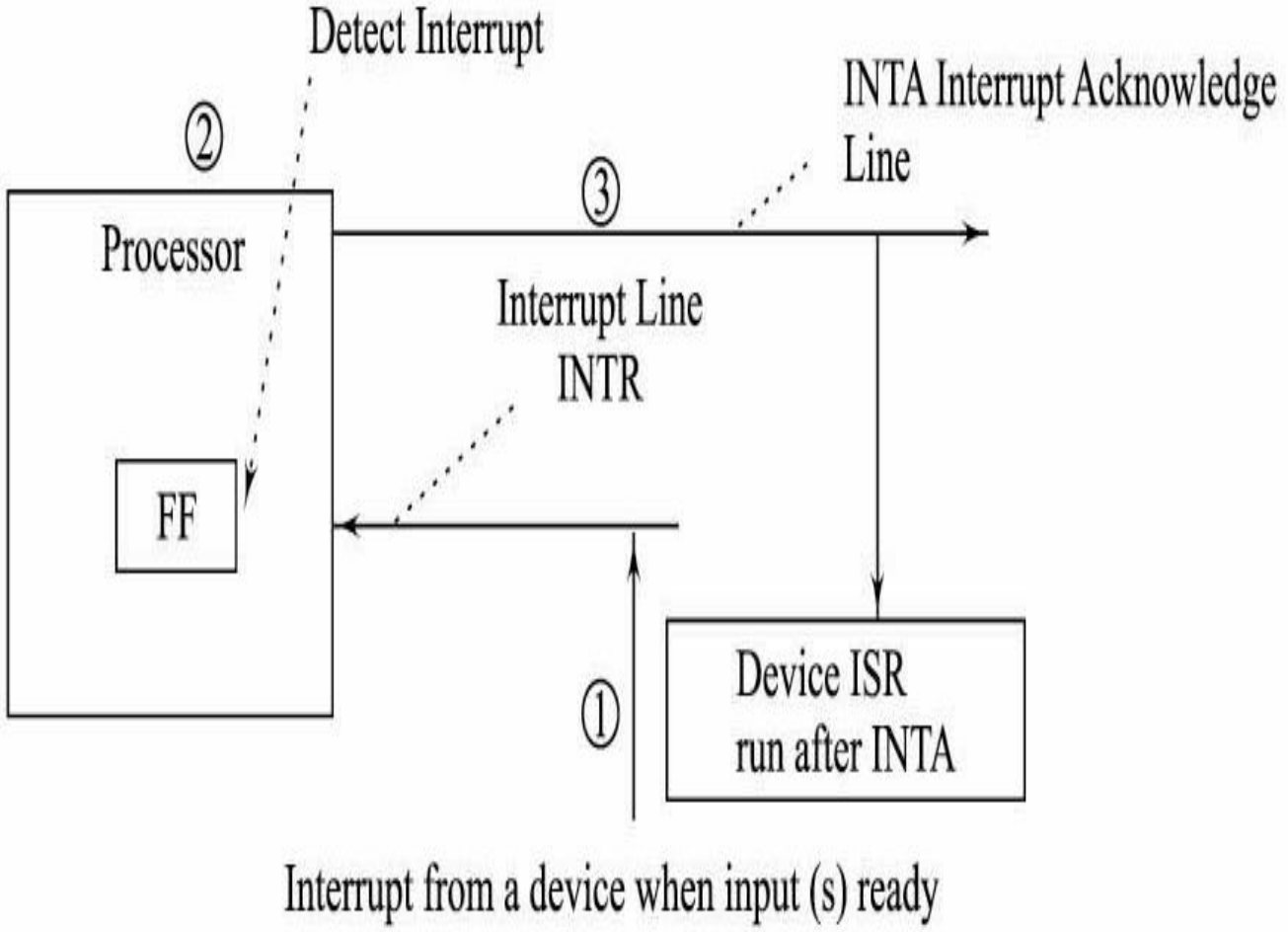
- Mechanism used by most processors to handle asynchronous type of events
- Essentially, the interrupts allow devices to request that the processor stops what it is currently doing and executes software (called interrupt service routine) to process the device's request, much like a procedure call that is initiated by the external device rather than by the program running on the processor

Interrupts

- Also used when the processor needs to perform a long-running operation on some I/O device and wants to be able to do other work while waiting for the operation to complete

Interrupt based IO mode hardware

Interrupt Hardware



Actions on Interrupt

Implementation of actions on interrupts

- Processor assigns a signal, known as an interrupt *request* signal (on a *line* INTR or IRQ), to each device that can issue an interrupt
- Typically, each device is also assigned an *interrupt acknowledge* (INTA) line that the processor uses to signal the device that it has received and begun to process the interrupt request by employing an ISR

Implementation of actions on interrupts

- Processor also provides a set of memory locations, known as the interrupt vector, that contains the addresses of the routine, called interrupt handler or interrupt service routine (ISR), that should be executed when an interrupt occurs

Context switch

- Necessary to ensure that the program running on the processor when the interrupt occurs is able to resume execution after the interrupt completes

Context switch

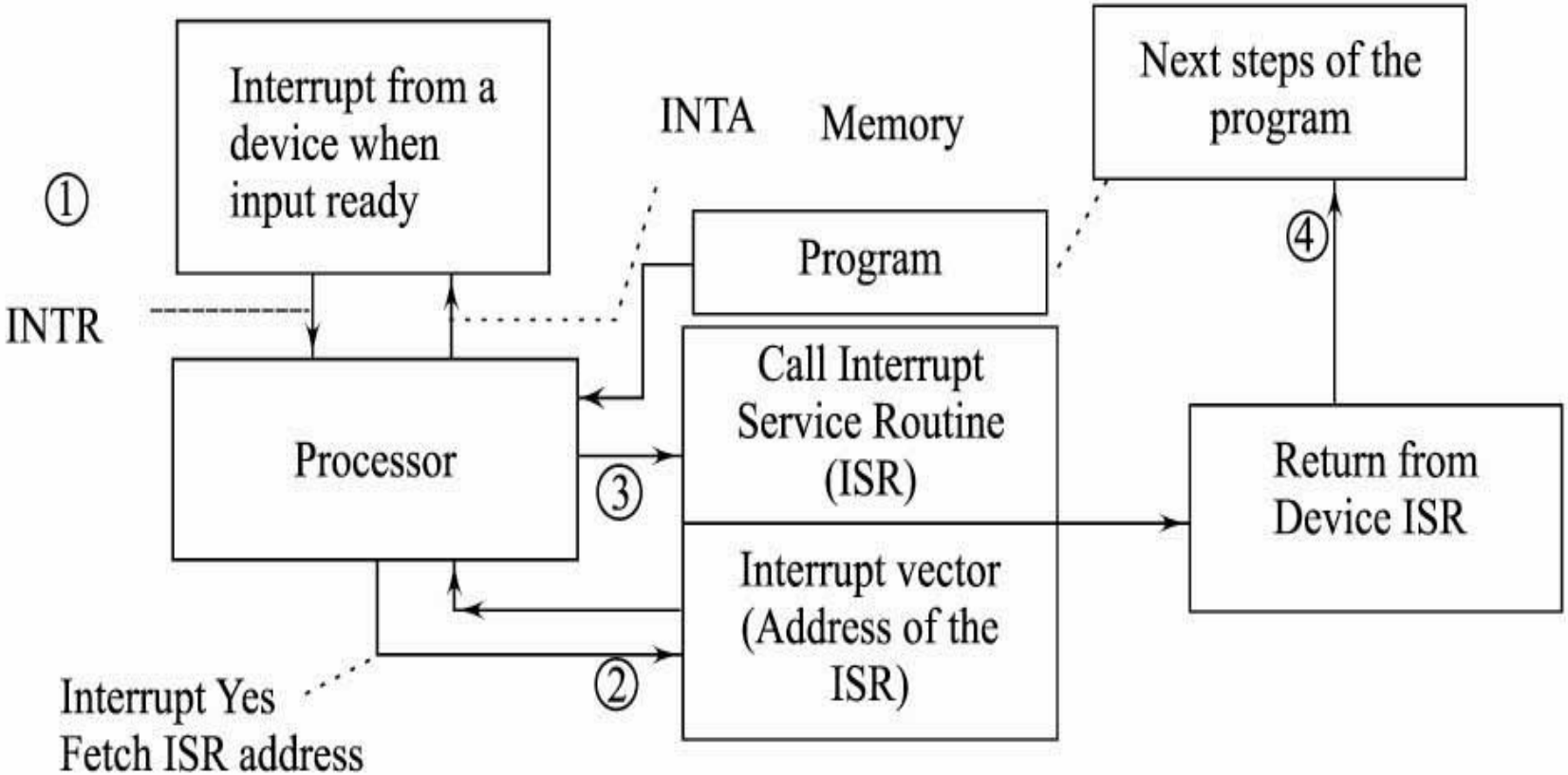
- After the interrupt handler completes, another context switch occurs and execution returns to some user program (not necessarily the one that was running when the interrupt occurred)

interrupts invisibility to user programs

- In the same way that timeslices allocated to other user programs are invisible
- The only way that a program can tell that an interrupt has occurred is to access the system's real-time clock and determine that more time than usual has elapsed between two events

Process Hardware Interrupt and Exemplary Steps on Hardware Interrupt

Process Hardware Interrupt and Exemplary Steps on Hardware Interrupt in Interrupt I/O Mode



Summary

We Learnt

- Interrupt driven IO
- Interrupt service on interrupt request by hardware signal or software interrupt instruction

End of Lesson 07 on

Interrupts- Part 1