1.3 INTERRUPTS

Interrupt is a signal that prompts the OS to stop one process and start work on another process.

- Virtually all computers provide a mechanism by which other modules (I/O, memory) may interrupt the normal sequencing of the processor.
- Interrupts are provided primarily as a way to improve processor utilization.

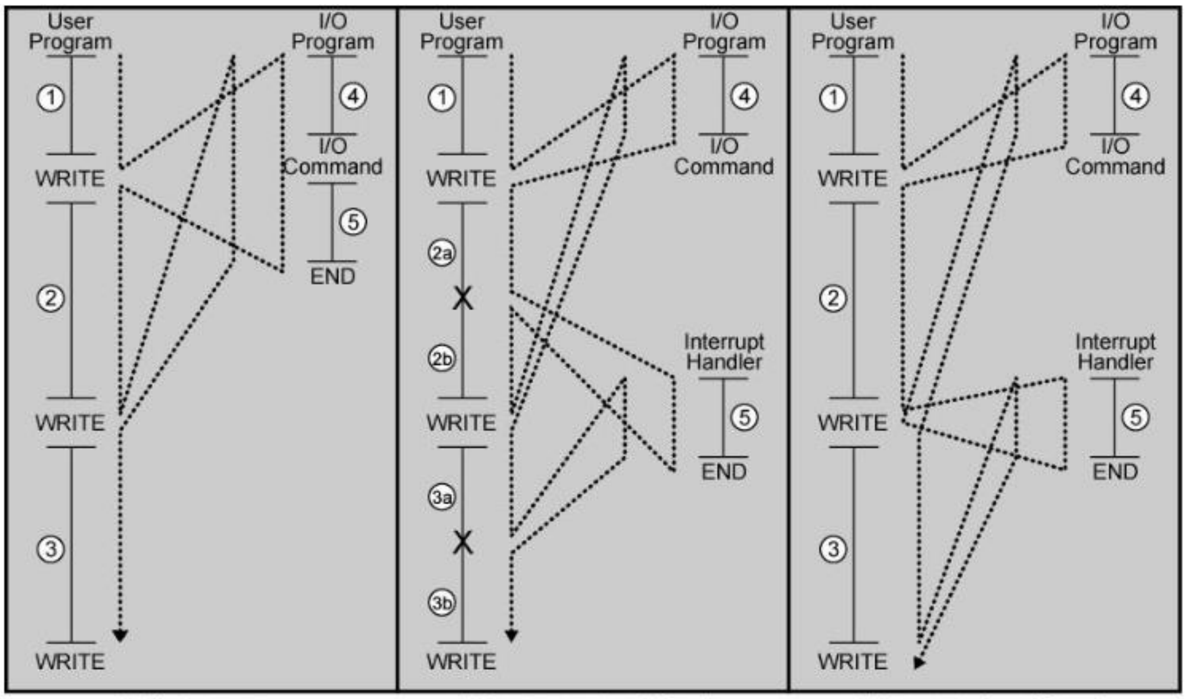
Table lists the most common classes of interrupts.

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Figure (a) illustrates this state of affairs. The user program performs a series of WRITE calls interleaved with processing. The solid vertical lines represent segments of code in a program. Code segments 1, 2, and 3 refer to sequences of instructions that do not involve I/O. The WRITE calls are to an I/O routine that is a system utility and that will perform the actual I/O operation.

The I/O program consists of three sections:

- A sequence of instructions, labeled 4 in the figure, to prepare for the actual I/O operation. This may include copying the data to be output into a special buffer and preparing the parameters for a device command.
- The actual I/O command. Without the use of interrupts, once this command is issued, the program must wait for the I/O device to perform the requested function (or periodically check the status, or poll, the I/O device).
- A sequence of instructions, labeled 5 in the figure, to complete the operation. This may include setting a flag indicating the success or failure of the operation.



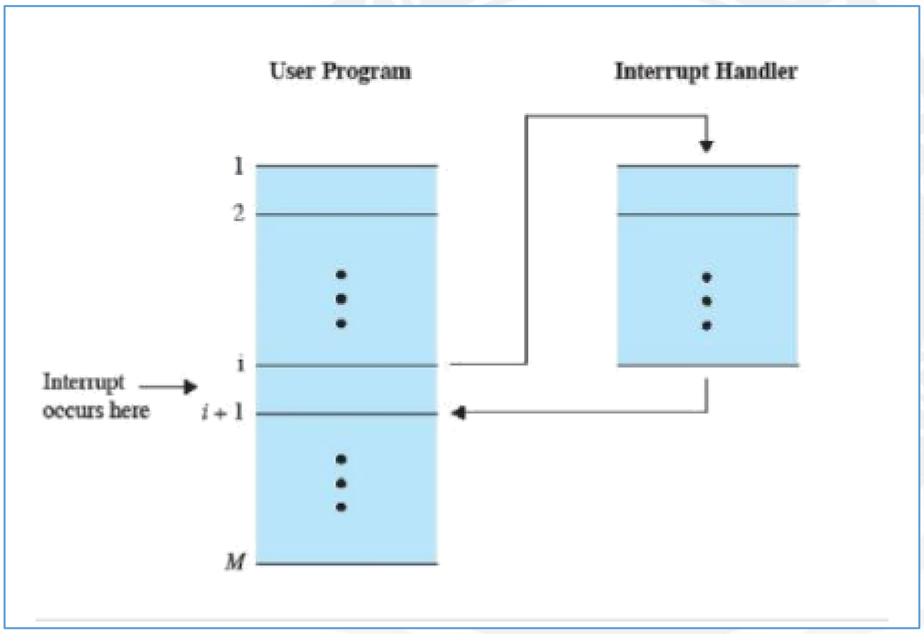
(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

- The dashed line represents the path of execution followed by the processor; Thus, after the first WRITE instruction is encountered, the user program is interrupted and execution continues with the I/O program.
- After the I/O program execution is complete, execution resumes in the user program immediately following the WRITE instruction.
- Because the I/O operation may take a relatively long time to complete, the I/O program is hung up waiting for the operation to complete; hence, the user program is stopped at the point of the WRITE call for some considerable period of time.

Interrupts and the Instruction Cycle

- With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.
- Consider the flow of control in Figure b.
- As before, the user program reaches a point at which it makes a system call in the form of a WRITE call.
- The I/O program that is invoked in this case consists only of the preparation code and the actual I/O command.

- After these few instructions have been executed, control returns to the user program. Meanwhile, the external device is busy accepting data from computer memory and printing it.
- This I/O operation is conducted concurrently with the execution of instructions in the user program.
- When the external device becomes ready to be serviced, that is, when it is ready to accept more data from the processor, the I/O module for that external device sends an *interrupt request* signal to the processor. The processor responds by suspending operation of the current program; branching off to a routine to service



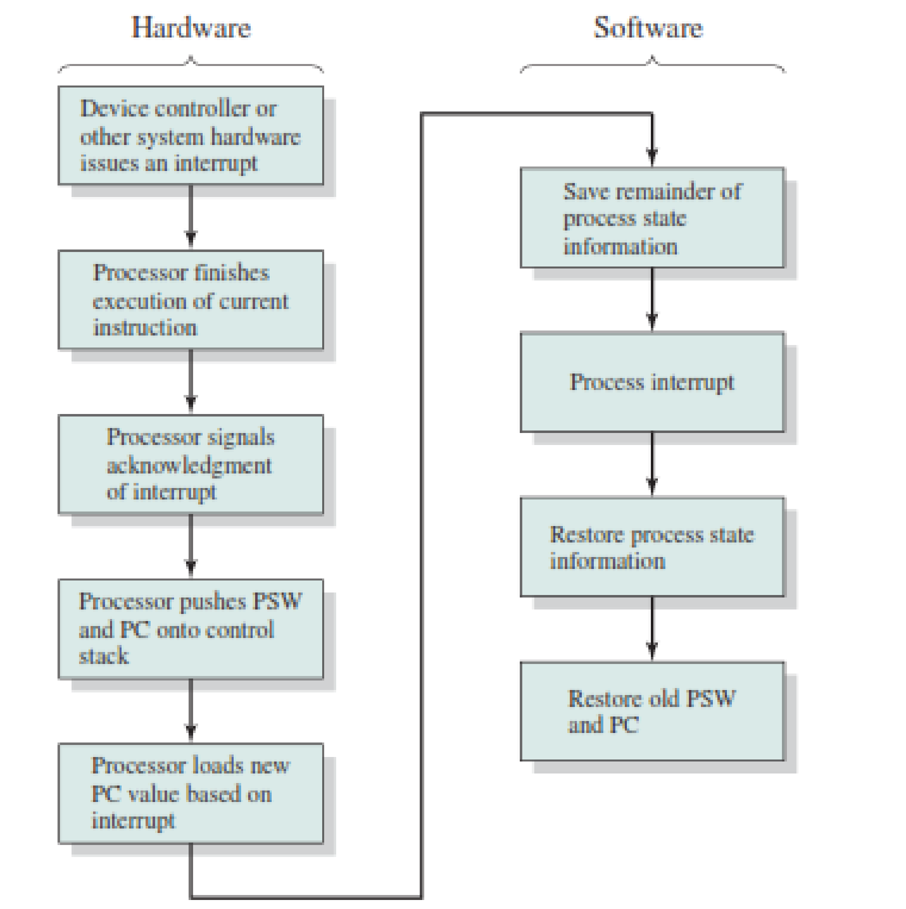

Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

Figure 1.10 Simple Interrupt Processing

Transfer of control via Interrupts

- To accommodate interrupts, an interrupt stage is added to the instruction cycle. In the interrupt stage, the processor checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal.
- If no interrupts are pending, the processor proceeds to the fetch stage and fetches the next instruction of the current program.
- If an interrupt is pending, the processor suspends execution of the current program and executes an interrupt-handler routine.
- This routine determines the nature of the interrupt and performs whatever actions are needed.

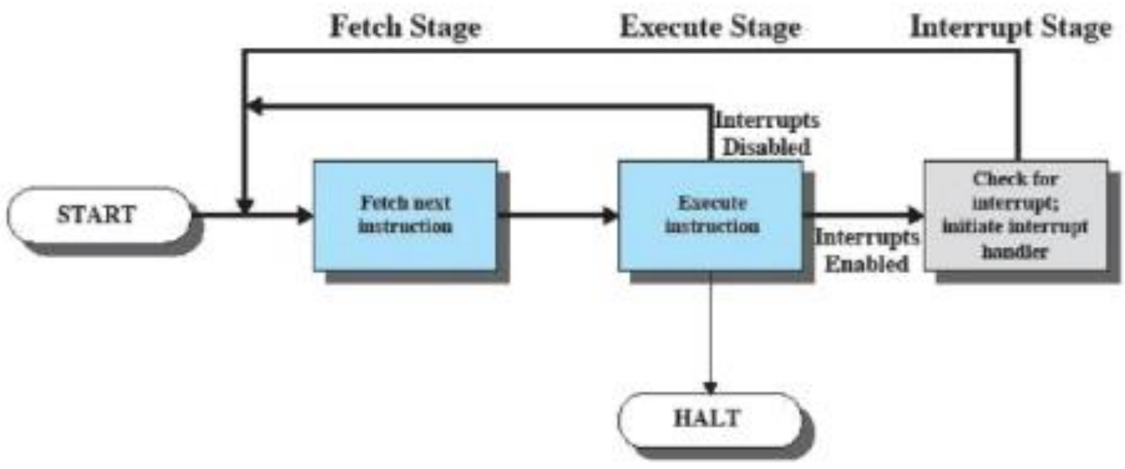


Figure 1.7 Instruction Cycle with Interrupts