Mid Assessment | Spring 2023

Md. Shafayet Hossain

CSE - 21st Batch | **Course Title:** Computer Peripherals & Interfacing

Course Code: CSE - 333 | **ID:** 2121210071

Answer to the Question no- 1

(a)

Computer Peripheral Interfacing

Computer Peripheral Interfacing is the interactions between computer processor and computer peripherals. Any input or output process that occurs through a peripheral device is known as computer peripheral interfacing.

Examples of Computer Peripheral Interfacing

- Taking input from a keyboard is a peripheral interfacing. Here keyboard is the peripheral device.
- Printing a document with a printer is a peripheral interfacing where printer is the peripheral device.
- Showing a video on the monitor is a peripheral interfacing and this case monitor is the peripheral device.

(b)

INTERRUPTS

Interrupt is an exceptional event which causes the CPU to temporarily suspend the current program being executed.

A hardware interrupt is an electronic signal from an external hardware device that indicates it needs attention from the OS.

One example of this is moving a mouse or pressing a keyboard key. In these examples of interrupts, the processor must stop to read the mouse position or keystroke at that instant.

Interrupts are primarily used for :-

- Request the CPU to initiate a new I/O operation.
- To signal the completion of an I/O operation.
- To signal the occurrence of hardware and software error.

Answer to the Question no- 2

(a)

Interrupt sources

There are many sources for interrupts varying from simply asserting an external pin to error conditions within the processor that require immediate attention.

- Hardware interrupt
- *Software interrupts*
- Maskable interrupts
- *Non-maskable interrupts*

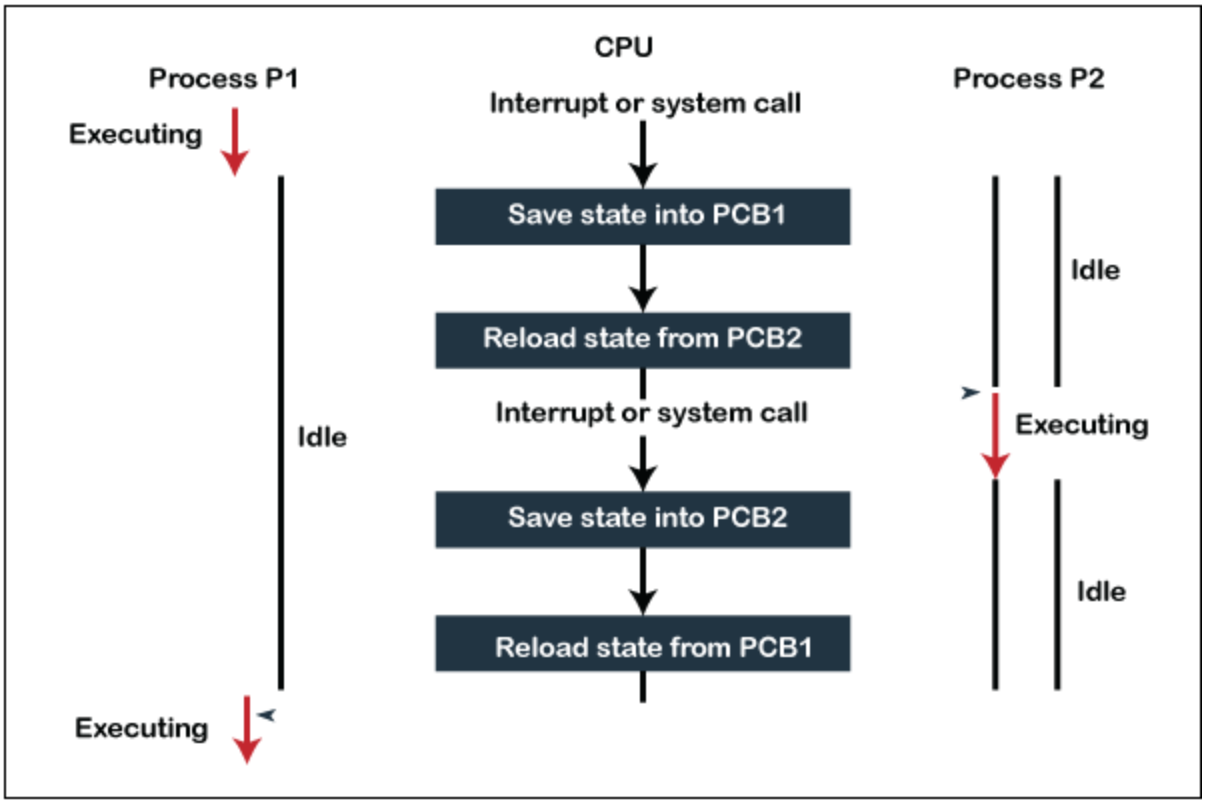
(b)

Context switching

The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system. When switching perform in the system, it stores the old running process's status in the form of registers and assigns the CPU to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same CPU to perform multiple tasks without the need for additional processors in the system.

Steps for Context Switching Mechanism

There are several steps involves in context switching of the processes. The following diagram represents the context switching of two processes, P1 to P2, when an interrupt, I/O needs, or priority-based process occurs in the ready queue of PCB.



As we can see in the diagram, initially, the P1 process is running on the CPU to execute its task, and at the same time, another process, P2, is in the ready state. If an error or interruption has occurred or the process requires input/output, the P1 process switches its state from running to the waiting state. Before changing the state of the process P1, context switching saves the context of the process P1 in the form of registers and the program counter to the **PCB1**. After that, it loads the state of the P2 process from the ready state of the **PCB2** to the running state.

The following steps are taken when switching Process P1 to Process 2:

1. First, thes context switching needs to save the state of process P1 in the form of the program counter and the registers to the PCB (Program Counter Block), which is in the running state.
2. Now update PCB1 to process P1 and moves the process to the appropriate queue, such as the ready queue, I/O queue and waiting queue.
3. After that, another process gets into the running state, or we can select a new process from the ready state, which is to be executed, or the process has a high priority to execute its task.

4. Now, we have to update the PCB (Process Control Block) for the selected process P2. It includes switching the process state from ready to running state or from another state like blocked, exit, or suspend.
5. If the CPU already executes process P2, we need to get the status of process P2 to resume its execution at the same time point where the system interrupt occurs.

Similarly, process P2 is switched off from the CPU so that the process P1 can resume execution. P1 process is reloaded from PCB1 to the running state to resume its task at the same point. Otherwise, the information is lost, and when the process is executed again, it starts execution at the initial level.

Answer to the Question no- 3

(a)

Analog Interface

An analog interface is an electrical connection that forwards analog electric signals to downstream electric and electronic devices or components for further processing. Standardised analog interfaces (e. g. 0/4 - 20 mA and 0/2 - 10 V) are also referred to as standard signals.

Since current signals are not sensitive to electromagnetic interference (activation of adjacent consumers) and loss of voltage (due to line resistance), they are preferred over voltage signals. The maximum length of a signal cable for the power source is only limited by the maximum load.

The load is the load resistance of an electrical transducer/converter with current output signal, whereby the cable resistance defined by the cable cross-section must be taken into account.

(b)

Interrupt Vector Table and how it works-

The interrupt vector table (IVT) is a data structure used by computer operating systems to manage interrupts. An interrupt is a signal sent by a hardware device or software application to the CPU, which temporarily suspends the current process and transfers control to a special routine known as the interrupt service routine (ISR) or interrupt handler.

The IVT is a table of memory addresses that maps each interrupt to its corresponding ISR. When an interrupt occurs, the CPU reads the interrupt number from a hardware register and uses it as an index into the IVT to find the address of the ISR that handles that interrupt.

Interrupt Service Routine and how it works-

The ISR is a special routine that is designed to handle the interrupt event. It is responsible for performing any necessary actions or processing related to the interrupt, such as storing data, updating registers, or calling other routines.

The ISR typically executes quickly and then returns control to the interrupted program. The interrupted program can then resume execution from where it was interrupted as if nothing had happened. If there are multiple interrupts, they are usually handled in the order they were received or according to their priority level.

The IVT and ISR work together to allow a computer system to handle multiple simultaneous events and provide a responsive and efficient computing experience. Without these mechanisms, a computer system would be unable to respond to external events in a timely and efficient manner, leading to poor performance and reduced usability.

(c)

Where CS 150 and IP 150 stored in memory

CS 150 and IP 150 are typically stored in the CPU registers during program execution.

CS (Code Segment) register holds the starting address of the code segment or the program code in memory, while IP (Instruction Pointer) register holds the address of the next instruction to be executed within the code segment.

Together, CS and IP determine the memory location of the instruction being executed by the processor. They are part of the x86 architecture, commonly used in personal computers and servers.

It's worth noting that different CPU architectures may have different register names and functions, but the concept of using registers to keep track of program execution is generally applicable across architectures.