CHAPTER 1

INTERRUPTS

Dr. Soha S. Zaghloul

LAYOUT

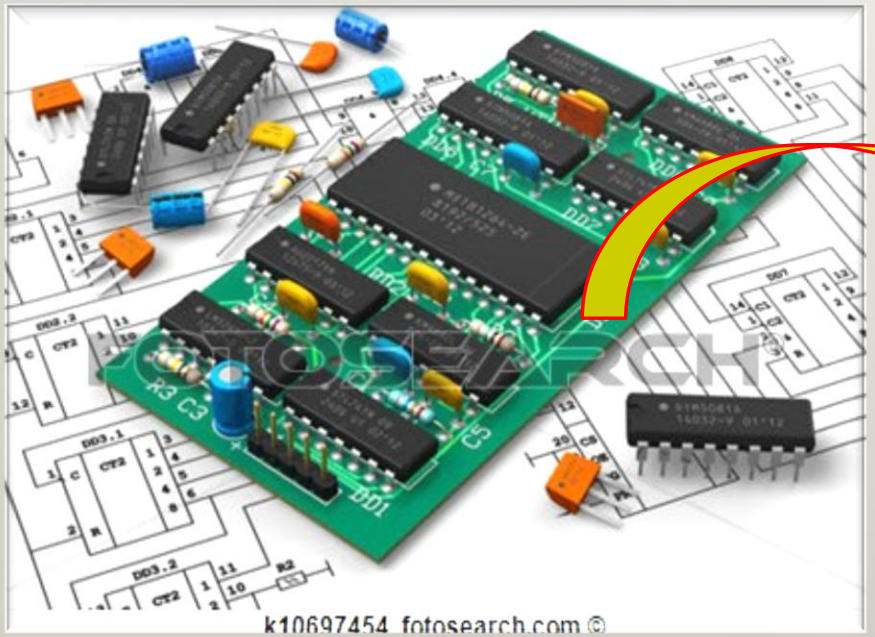
- Device Controller
- Device Driver
- Devices Controlling Techniques
  - Polling
  - Interrupt
- Interrupt Classes
  - H/W Interrupt
    - Interrupt Controller
    - Process State
    - Interrupt Vector
  - S/W Interrupt
    - Exception Classes

Device Controller

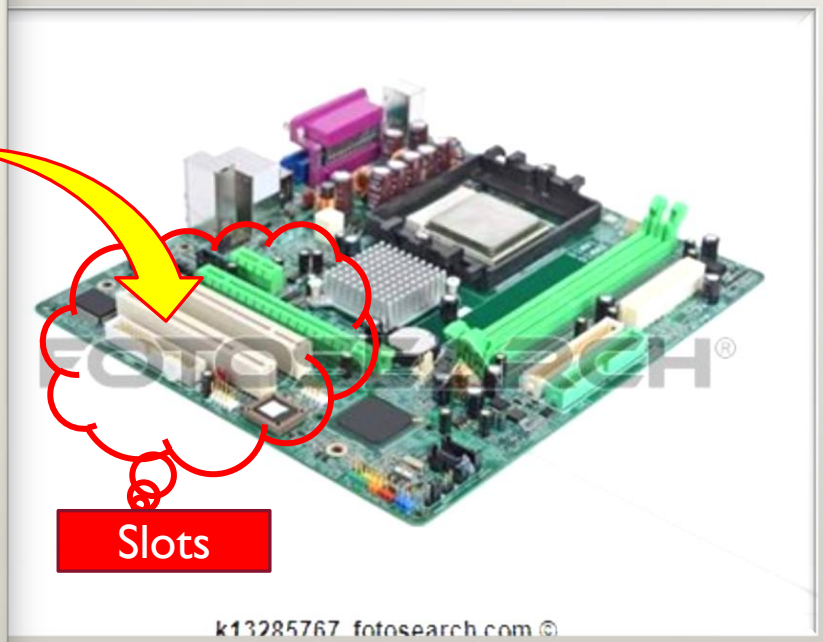
Each I/O unit consists of two components: a mechanical one and an electronic one.

The electronic one is called the device controller or device adapter.

On personal computers, it often takes the form of a printed circuit card that can be inserted into an expansion slot on the mainboard.



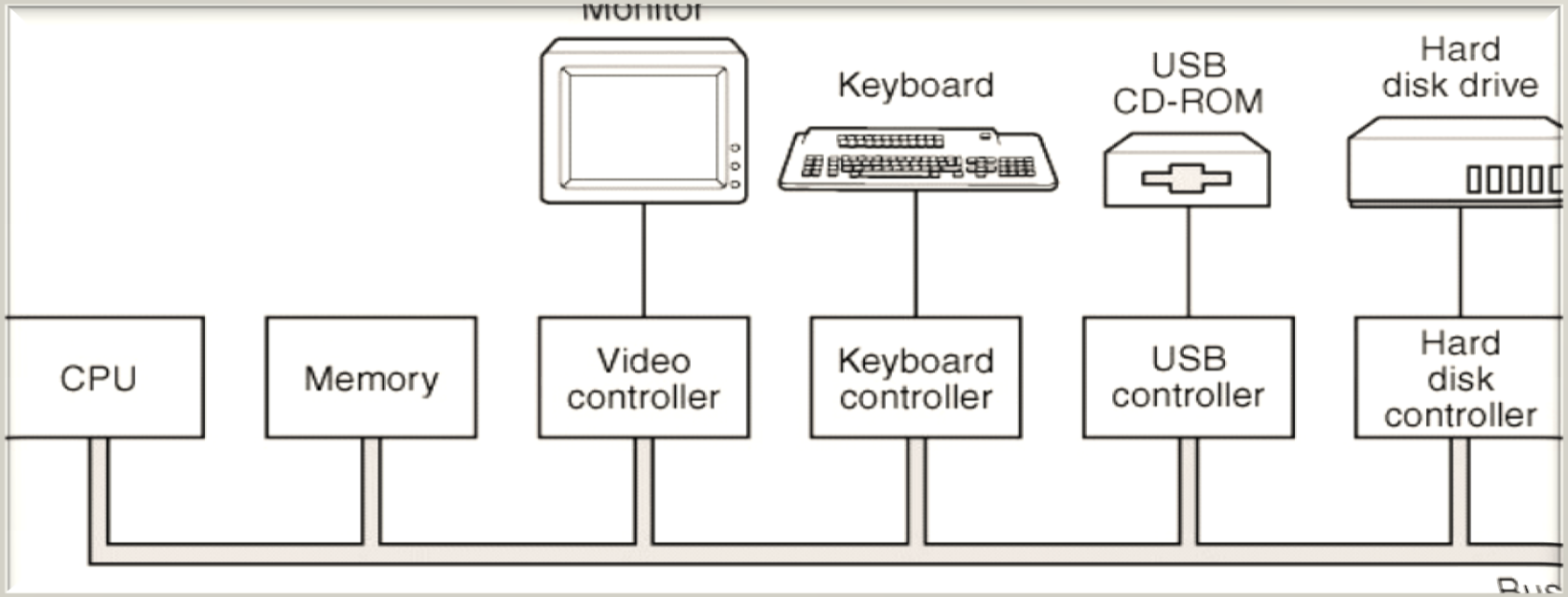
Device Controller (Adapter)



Motherboard (mainboard)

DEVICE DRIVER

The controller card usually has a connector into which a cable leading to the device itself can be plugged.



The OS deals with device controller (H/W) rather than the device itself through the corresponding device driver (S/W).

Each controller has a few registers. Based on the contents of these registers, the OS can command the device to perform a specific action such as deliver data, accept data, switch on/off, etc…

Also, the OS learns the state of the device from these registers.

In addition to the registers, a controller contains data buffers that the OS can read/write.

DEVICES CONTROLLING TECHNIQUES

It is important for the OS to recognize and manipulate the status of each device.

The status of a device includes the following information:

- Is it correctly connected or not?
- Does it work or not?
- Is it busy or available?

The OS keeps track of the statuses of all connected devices using a device-status table stored in memory.

Two techniques are applied for this purpose:

- Polling
- Interrupt

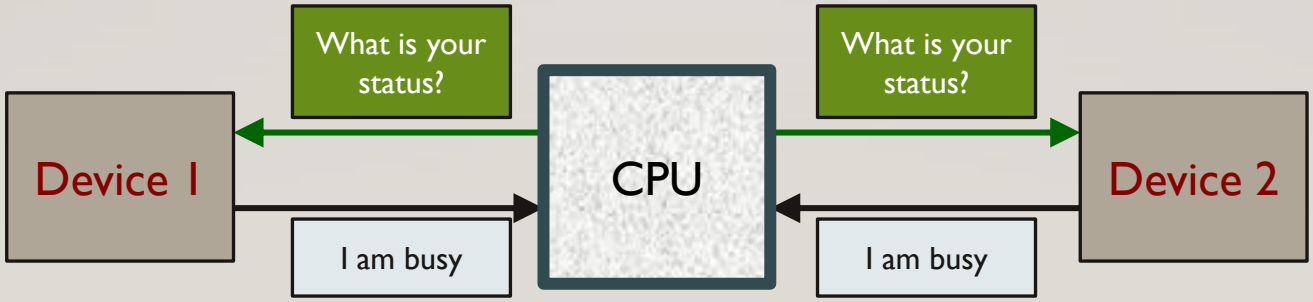
POLLING

~ Target:
In order for the CPU to recognize the status of each connected device, a communication should take place between both.

~ Concept:
The CPU repeatedly requests from each device its current status.

~ Definition:
The CPU sends electrical signals regularly to each connected device asking about its current status. The device emits another signal to respond to each signal sent by the CPU.

~ Disadvantages:
* This causes an overhead on the CPU
* The CPU stays idle ➔ waste of resources
* The mechanism is slow
* Increases the complexity of the computer system



Interrupts eliminate the need for a processor to repeatedly poll devices.

INTERRUPTS

~ Definition:
An interrupt is an electrical signal emitted by the hardware or software to the processor indicating an event that needs immediate attention.

~ Target:
Interrupts allow the OS to take control of the processor to manage system resources.

~ Concept:
An interrupt alerts the processor to a high-priority condition requiring the "interruption" of the current code that the CPU is executing.

As soon as the CPU receives an interrupt, the following steps are performed:

- (1) The processor suspends its current activity.
- (2) The processor saves its state.
- (3) The processor calls the Interrupt Handler (IH), sometimes called Interrupt Service Routine (ISR), to handle the event.

The Interrupt Handler (IH) – or Interrupt Service Routine (ISR) – is a relatively small program (procedure). It is part of the OS.

There are many types of interrupts. Each type of interrupt calls the corresponding Interrupt Handler.

INTERRUPT CLASSES

- Hardware Interrupts
- Software Interrupts

INTERRUPT CLASSES – H/W INTERRUPTS

~ Emitted by a device to indicate that it requires attention from the OS.

~ Implemented using electronic signals sent from an internal/external device.

~ For example, pressing a key on the keyboard "triggers" an interrupt that causes the processor to read the keystroke.

~ May occur in the middle of instruction execution.

~ Asynchronous: changes immediately – does not wait for the fall/rise of a clock cycle.

~ Requires special care in programming.

~ The act of initiating a hardware interrupt is known as an Interrupt Request (IRQ).

~ The number of H/W interrupts is limited by the number of IRQ lines.

H/W INTERRUPTS – INTERRUPT CONTROLLER

An interrupt line is an electrical connection between the mainboard and the CPU.

Each device has a controller that is responsible to activate the interrupt line when necessary (ie. when a status changes) to inform the CPU that an event has occurred.

The CPU is provided with an interrupt controller that sorts the received requests based on their priority.

Important interrupts are therefore serviced first. The rest of interrupts are queued and serviced after higher-priority ones.

HARDWARE INTERRUPTS – PROCESS STATE

The paused process (program) resumes after the CPU handles the current interrupt.

In order to continue a process after being paused, some information should be saved.
Examples of such information include:
~ where did the execution stop (instruction address)
~ the values of data generated during process execution: these are saved in the registers.
~ and others…

These information are known as the process state.

The process state is stored in the Process Control Block (PCB). Each process has its own (PCB).

H/W INTERRUPTS – INTERRUPT VECTOR

Each type of interrupt is assigned a unique value that the CPU uses as an index into the interrupt vector.

The interrupt vector is an array of pointers to the memory addresses of all available interrupt handlers.

The interrupt vector is loaded in a protected area of the memory, so that it cannot be erased or modified inadvertently.

H/W INTERRUPTS – MANIPULATION

Step 1:
When the status of a device changes, it activates the interrupt line to inform the CPU about the event.

Step 2:
As soon as the CPU receives the signal through the interrupt line, it completes execution of the current instruction then pauses the execution of the current process (program).

Step 3:
The CPU saves the state of the paused process.

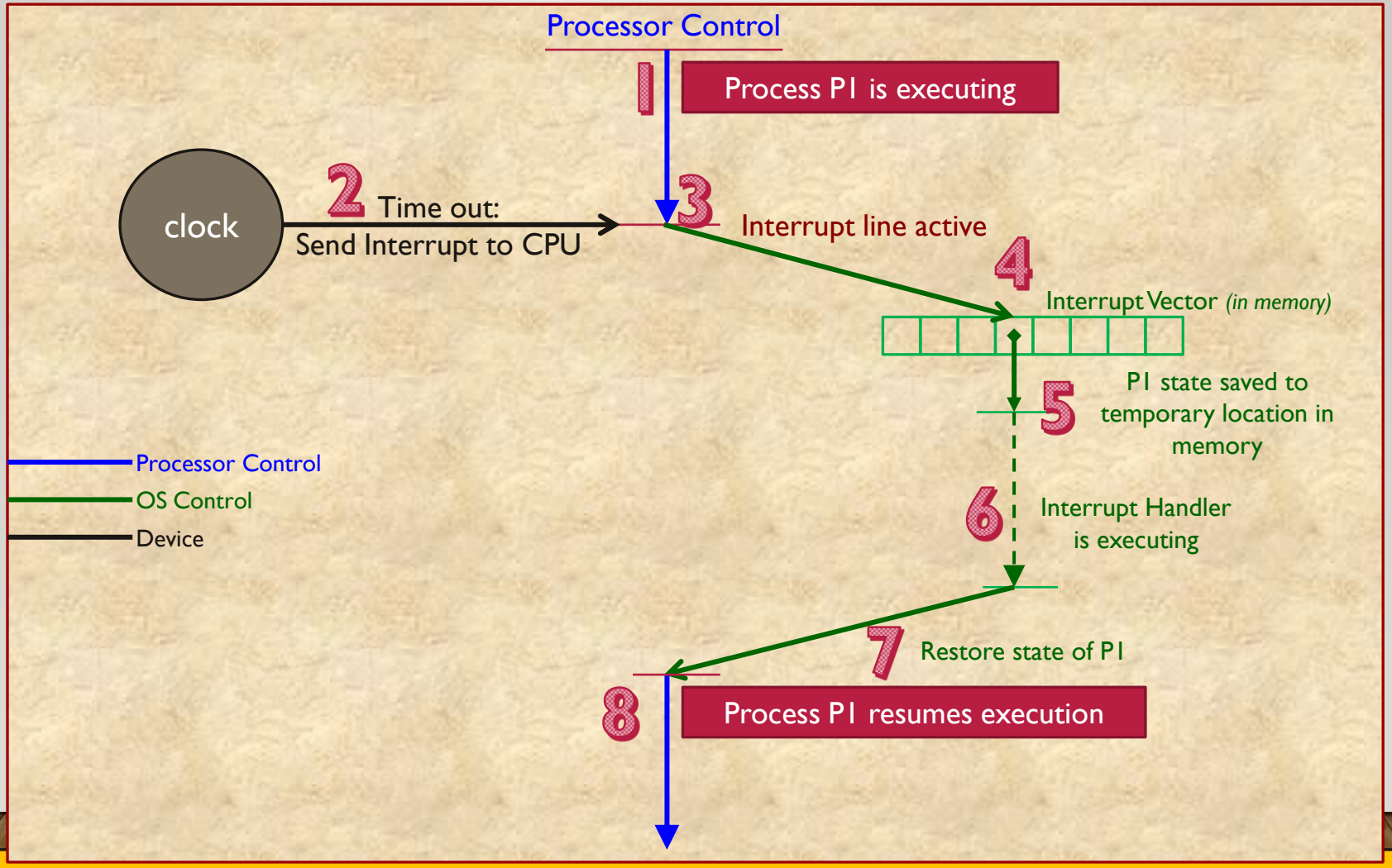
Step 4:
The CPU transfers control to the appropriate interrupt handler.

Step 5:
The interrupt handler performs appropriate actions based on the type of interrupt.

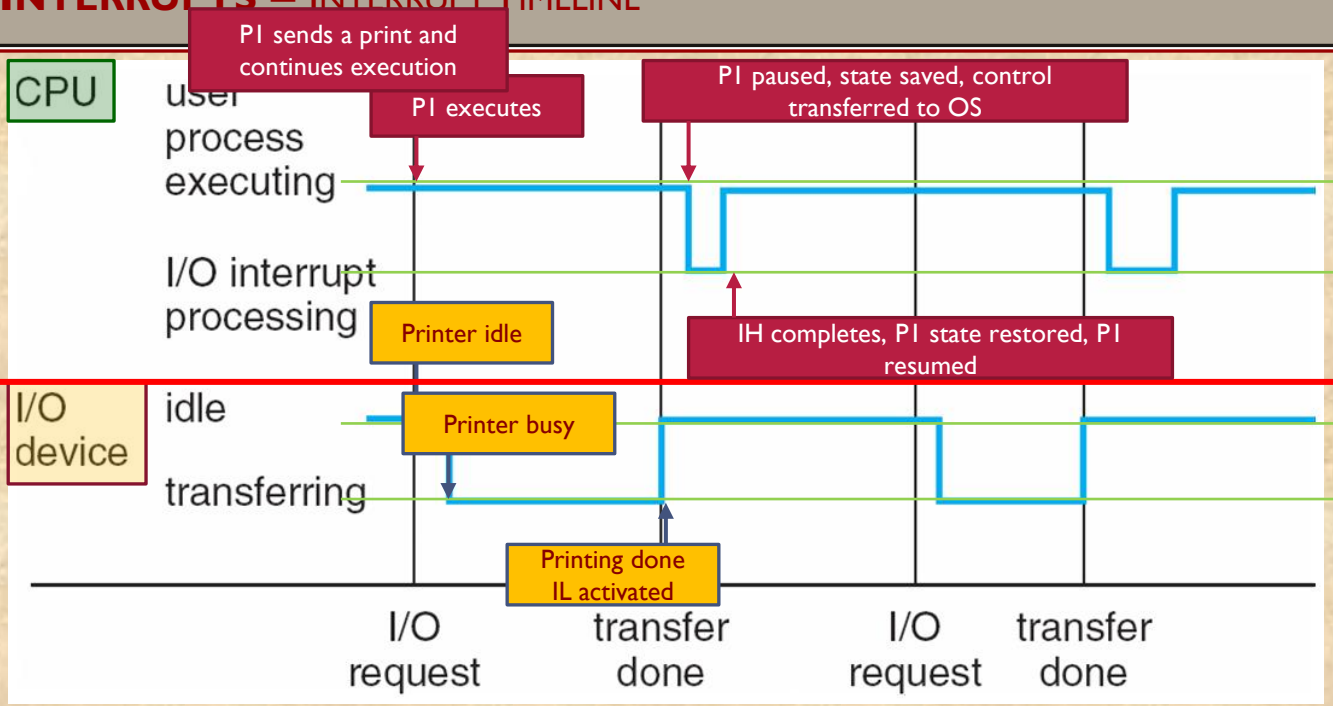
Step 6:
After the interrupt handler completes execution, the state of the previously interrupted process is restored.

H/W INTERRUPTS – EXAMPLE

A time-out is set to a specific time. When the time elapses the clock sends an interrupt.



H/W INTERRUPTS – INTERRUPT TIMELINE



IL = Interrupt Line
IH = Interrupt Handler

Note how the H/W-generated interrupts are asynchronous; ie. Changes do not wait for the rise/fall of the clock.

INTERRUPT CLASSES – S/W INTERRUPTS

~ May occur unintentionally due to an error that cannot be handled by the CPU: this is known as exception or trap.

~ For example, an exception is "thrown" when the program attempts to:
* Divide by zero.
* Access a protected memory area.
* Access an out-of-memory address.

~ In such cases, the OS decides to abort (end) the program.

~ S/W Interrupts may also be intentionally caused by a special instruction in the instruction set.

~ Examples of intentionally caused S/W interrupts include:
* Breakpoints set within a program for debugging purposes.
* Exceptions issued in Java code.

~ An OS may include up to hundreds of S/W interrupts for different purposes.

S/W INTERRUPTS – EXCEPTIONS CLASSES

The set of S/W interrupts that a computer supports depends on the system's architecture.

For example, Intel IA-32 architecture supports a set of three exceptions:

* Faults
* Traps
* Aborts

The class of exception specifies the behavior of the running process.

Faults cause the running process to stop execution (abort). After the problem is corrected, the stopped process restarts.

Traps continue executing the running process after giving a message. Examples include breakpoints and overflow (the value stored in a register exceeds its capacity).

Aborts lead to a system crash. The OS may not be even able to store the state of the running process. Examples include hardware failure; or an exception handling itself causes another exception: this is called double-fault exception.

H/W vs. S/W Interrupts

H/W-generated

- ~ Generated by H/W devices
- ~ Issued to inform the CPU that an event occurred (ie. H/W status changed)
- ~ Examples:
  * A keyboard issues an interrupt when the user presses a key
  * A printer issues an interrupt upon completion
- ~ Asynchronous: time-independent of running instruction

S/W-generated

- ~ Generated by programs
- ~ Issued when an error occurs or intentionally for debugging purposes
- ~ Examples:
  *Division by zero
  *A H/W device is malfunctioning
  *Breakpoint/Exception in the code
  *Accessing a protected memory area (reserved for the OS)
  *Accessing an out-of-memory address
- ~ Synchronous: time-dependent of the running instruction