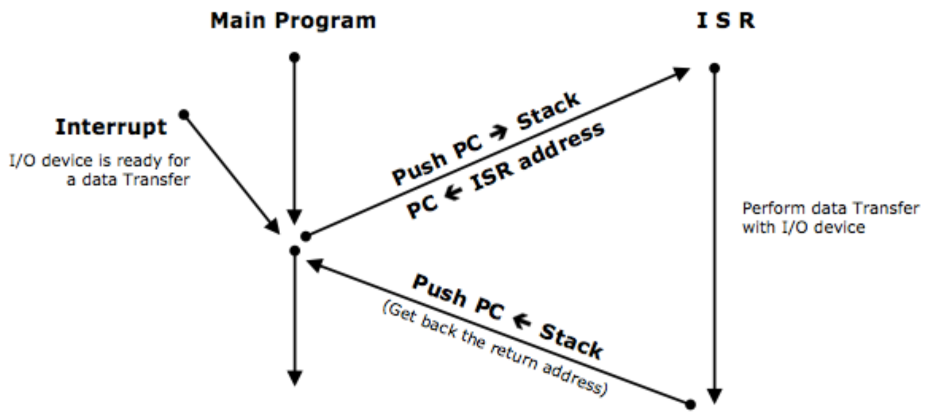
## 2) INTERRUPT DRIVEN I/O

1) In interrupt driven I/O, the transfer is not initiated by the processor.
2) Instead, **an I/O device which wants to perform a data transfer with the processor, must give an interrupt to the processor.**
3) An interrupt is a condition that makes the processor execute an ISR (Interrupt Service Routine).
4) In the ISR, processor will perform data Transfer with the I/O device.
5) This relieves the processor from periodically checking the status of every I/O device thereby saves as lot of time of the processor.
6) **The processor is free to carry on its own operations.**
7) Whenever a device wants to transfer data, it will interrupt the processor.
8) This is how many I/O devices Transfer data with the processor.
9) E.g.:: **Keyboard.** Instead of the processor checking all the time, whether a key is pressed, the keyboard interrupts the processor as an when we press a key. In the ISR of the keyboard, which is a part of the keyboard driver software, the processor will read the data from the keyboard.
10) **Hence interrupt driven I/O is much better than Polled I/O (Programmed I/O).**

### INTERRUPT HANDLING MECHANISM

1) When an interrupt occurs, processor, firstly, finishes the current instruction.
2) It then suspends the current program and executes an ISR.
3) To do so, it Pushes the value of PC (address of next instruction), into the stack.
4) Now it loads the ISR address into PC and proceeds to execute the ISR.
5) At the end of the ISR, it POPs the return address from the stack and loads it back into PC.
6) This is how the processor return to the very next instruction in the program.


Main Program
I S R
Interrupt
I/O device is ready for a data Transfer
Push PC → Stack
PC ← ISR address
Perform data Transfer with I/O device
Push PC ← Stack
(Get back the return address)

| | INTERRUPT DRIVEN I/O | POLLING (PROGRAMMED I/O) |
|---|---|---|
| 1 | **I/O device interrupts the processor whenever it wants to perform a data Transfer.** | **Processor periodically checks (polls) the status of every I/O device to know if it wants to perform a data Transfer.** |
| 2 | **Processor is free** to carry on its own operations, hence **saves system time**. | **Processor is busy** in constantly checking all I/O devices, hence **system time wasted**. |
| 3 | **Additional hardware required** to handle interrupts. E.g.:: 8259 Programmable interrupt controller. | Additional hardware **not required**. |
| 4 | Increases **cost and complexity** of the system. | System is **cheaper and less complex**. |
| 5 | Interrupt **priority** has to be managed through software or through hardware. | No such issue. |
| 6 | Interrupt **vector addresses** (ISR Addresses) need to be stored in an Interrupt Vector table - **IVT**. | No such issue. |

## Types of interrupts

### 1. VECTORED AND NON VECTORED INTERRUPTS

A key element in interrupt handling is the ISR address.

**If an interrupt has a fixed ISR address, it is called a Vectored interrupt.**
Such interrupts are **executed faster** as the ISR address is known to the processor.
But such interrupts are **rigid**. Since they have a fixed ISR address they can serve only **one device**.
They cannot accept interrupts from multiple devices.
So they cannot expand the interrupt structure.
**E.g.:: NMI interrupt of 8086** (Has a fixed vector number i.e. 2)

**If an interrupt does not have a fixed ISR address, it is called a Non-Vectored interrupt.**
Such interrupts are **executed slower**.
The ISR address is **obtained from the** interrupting **device**, usually an interrupt controller like **8259**.
But such interrupts are **flexible**.
Since they don't have a fixed ISR address they **can accept interrupts from multiple devices**.
So they can be used to **expand the interrupt structure**.
**E.g.:: INTR interrupt of 8086** (Can service any vector number from 0… 255)

## 2. MASKABLE AND NON MASKABLE INTERRUPTS

Masking an interrupt means disabling it.

A **Maskable** interrupt is an interrupt that **can be disabled**.
If disabled, whenever this interrupt occurs, the processor will ignore it and simply continue the main program. Such interrupts are generally used to handle low priority, non-critical events like keyboard presses which can be easily disabled (Keypad can be locked)
**E.g.:: INTR interrupt of 8086** (is disabled when Interrupt Flag is 0)

A **Non-Maskable** interrupt is an interrupt that **cannot be disabled**.
Whenever this interrupt occurs, the processor will have to service it.
Such interrupts are generally used to handle high priority, critical events like over-heating of the mother board, power failure etc.
**E.g.:: NMI interrupt of 8086** (can never be disabled)

## 3. SOFTWARE AND HARDWARE INTERRUPTS

This is based on how the interrupt occurs.

If an interrupt is caused by **writing an instruction**, it is called a **software interrupt**.
Software interrupts are predictable events and are given by the programmer.
**E.g.:: INT n instruction of 8086** (n can be anything between 0… 255)

If an interrupt is caused by **a signal on an external pin**, it is called a **hardware interrupt**.
Hardware interrupts are un-predictable events and are given by external devices.
**E.g.:: NMI and INTR pins of 8086**