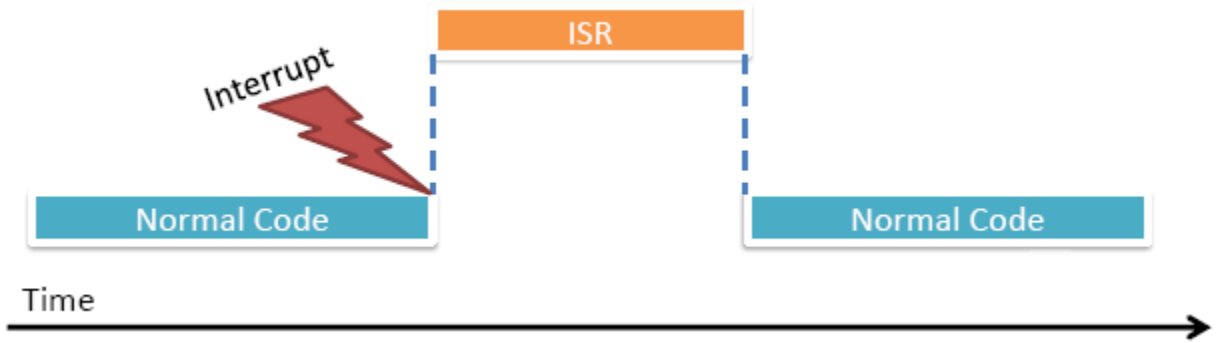
Interrupts

Lecture 4

This material is developed by IMTSchool for educational use only
All copyrights are reserved

Interrupts

Interrupt: Is a combination of software and hardware to force the processor to stop its current activity and begin to execute a particular piece of code called an interrupt service routine (ISR). Which is a response to a specific event generated by hardware change (internally or externally) or even software.



Interrupts vs Polling

We use a telephone as an example to compare polling and interrupt efficiency. Suppose you are expecting a call. In polling, you pick up your telephone every 10 seconds to check whether there is anyone on the line calling you (without ring). In the interrupt you continue to perform whatever tasks you are supposed to complete while waiting for the telephone to ring.

Interrupts types

Interrupts can be categorized according to many things According to interrupt source position:

The interrupt source could be outside the processor which is called external interrupt, and could be inside the processor which is called internal interrupt.

- **External Interrupt**

  Is an electronic alerting signal sent to the processor from an external device outside the processor.

Interrupts types

- **Internal Interrupt**:

  Is inside the processor divided to

  - trap like:

    Division by Zero
    Stack Overflow
    Incorrect Op-code

  - Software interrupt which is a specific assembly instruction called (SWI) that fires an interrupt.

Is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor.

Interrupt Handling Techniques

- ***Vectored Interrupts:***

The CPU has these addresses per-defined in memory (interrupt table) in advance. When a vectored interrupt is fired, the interrupting module/device sends its specific vector to the CPU via the data bus.

- ***Non-vectored Interrupts:***

For non-vectored interrupts, the CPU has a hardware fixed address called the interrupt vector. When an interrupt is fired, the CPU will push the PC to the stack. Then it'll jump to the interrupt vector address and then branches to the ISR handler code. Which is a hard-coded ISR in a specific portion of the memory.

Vectored vs Non Vectored

Vectored interrupts	Non Vectored interrupts

ISR (external interrupt)
{
// some codes
}
ISR (ADC interrupt)
{
// some codes
}
	ISR ( )
	{
	if (external interrupt flag==1)
	{
	// some codes
	}
	if (ADC interrupt flag==1)
	{
	// some codes
	}
	}

Interrupt vs Function call

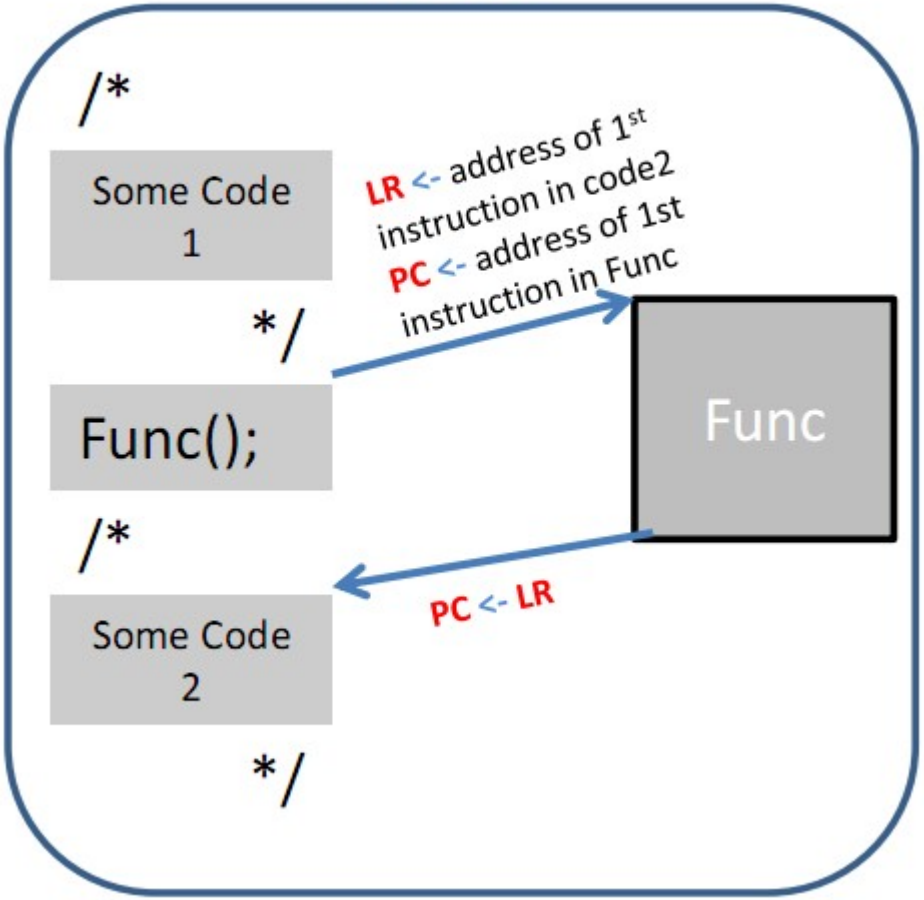
Difference between interrupt and a function call is:

The context switching in the **Function call** is known before so the compiler is responsible for the context switching but in the **Interrupt** we do not have any idea when the interrupt is coming?

So in function call the context switching is fully supported by software (compiler) but in the interrupt is fully supported by hardware (processor).

Function Call and Link Register

Example



Function Call and Link Register

void *main* (void)

```
; main assembly program
;*********************
__main  PROC

; Initialize R0
MOV   R0, #0x01

; Call the function
BL    __func

;Loop forever
B     __main

ENDP
```

void *func* (void)

```
; func assembly program
;*********************
__func  PROC

;Check if R0 equals 1 or not
;and save 0 or 1 to R1
TST       R0, #0x01
MOVEQ     R1, #0
MOVNE     R1, #1

; Return from the function
BX    LR

ENDP
```

Note

Do not use more than one context switching (nested context switching) to achieve the use of Link Register so in software design and architecture, advises you “do not call a function inside a function even if an initialization”.

IMT SCHOOL

www.imtschool.com

ww.facebook.com/imaketechnologyschool/

This material is developed by IMTSchool for educational use only
All copyrights are reserved