EE2361 - Lecture 15

10/12/16

Return graded HW 1

If you don't pick up will be in my office

Last Time -

Simple Timer routine, polled a flag bit associated with Timer 1

TMR1 == PR1 ⇒ flag bit set

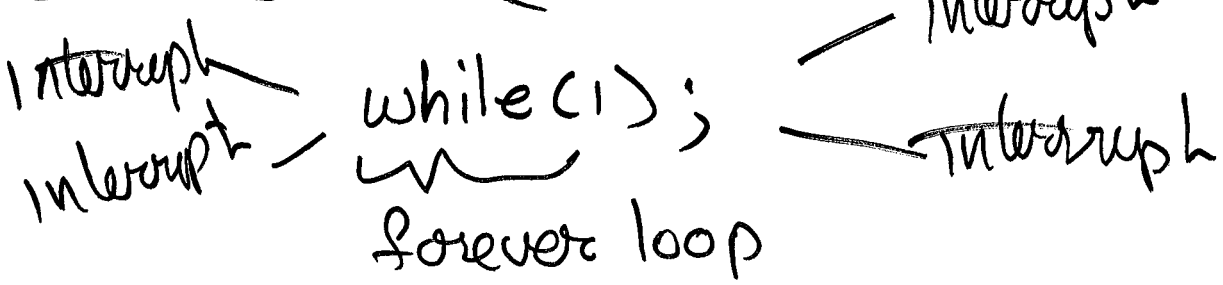
⇒ Problem with this

Spends a lot of time waiting for the timer event ⇒ wasted time

You can improve this situation by using

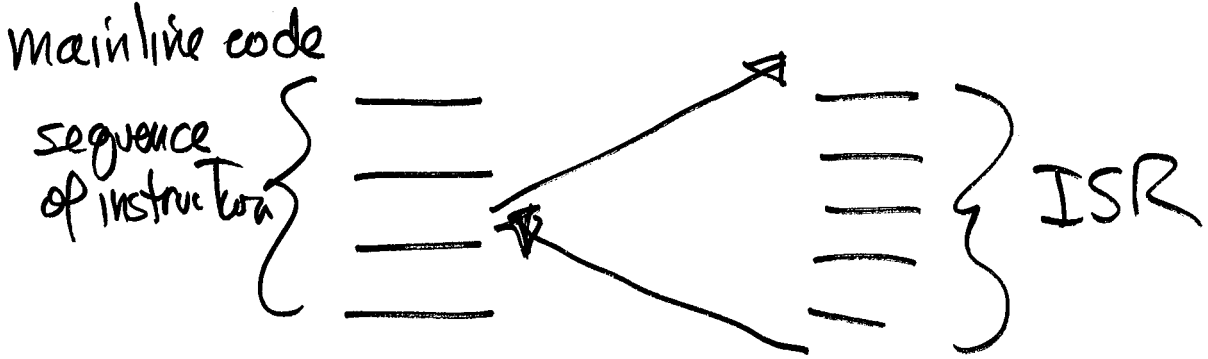
INTERRUPTS

General Idea



An interrupt

Asynchronous hardware event, causes an interrupt service routine to execute, and then return to main code



For an ISR - like a function
but not a function in that it is
not invoked in software

A hardware event initiates its execution

Since an interrupt can occur at any time

⇒ it cannot be passed parameters or return results like a function

so void isr(void) (required)

⇒ speed (latency) is important

so an isr should be short and not call functions (recommended)

⇒ we need to tell the processor where to find the ISR

How do we identify a function as an ~~IRQ~~ ISR?

Where do we locate the ISR?

→ we use an attribute declaration (other compilers use pragmas)

→ use a Interrupt vector table

When an interrupt occurs

the program code executing is suspended

=> need to save the context

load the PC with address of interrupt vector for this interrupt

=> that vector typically contains the starting address of the ISR

IVT - interrupt vector table
(also an alternate interrupt
vector table)

Natural priority in the table
Traps are highest priority
- non maskable interrupts
Interrupts
- 5 bits, flag bit, enable bit,
3-bits for priority

From the IVT we find the address of the ISR

The ISR must be declared with an attribute

(Timer1 example)

When the interrupt occurs

execute the instruction at the vector (goto)

enter the ISR

structure of ISR
- prologue * → save context
- code to service the event
- epilogue * → restore the context

For an Interrupt we need to save the context

This is the general purpose register values → put on stack

The SR values → put on stack

RCOUNT value → put on the stack

(also put return address on stack)