Interrupts in ARMlite

COS10004 Lecture 11.2

Revisiting our Flashing LED Example

- Recall our flashing LED program from last week, and in particular the delay function:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   delay function
;;   inputs R0 - time delay in seconds
delay:
      push {R3,R4,R5,R6}
      MOV R3, R0            ; move delay time param into R3
      LDR R4, .Time         ; get start ime
timer:
      LDR R5, .Time         ; update time
      SUB R6, R5, R4        ; calc elapsed time
      CMP R6, R3            ; compare elapsed to delay time
      BLT timer
      pop {R3,R4,R5,R6}
      RET
```

Revisiting our Flashing LED Example

- This is a busy-wait timer, or in other words, an example of polling:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   delay function
;;   inputs R0 - time delay in seconds
delay:
      push {R3,R4,R5,R6}
      MOV R3, R0          ; move delay time param into R3
      LDR R4, .Time       ; get start ime
timer:
      LDR R5, .Time       ; update time
      SUB R6, R5, R4      ; calc elapsed time
      CMP R6, R3          ; compare elapsed to delay time
      BLT timer
      pop {R3,R4,R5,R6}
      RET
```

Revisiting our Flashing LED Example

- The CPU constants checks for a state – in this case, that the time exceeds a delay time.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   delay function
;;   inputs R0 - time delay in seconds
delay:
      push {R3,R4,R5,R6}
      MOV R3, R0          ; move delay time param into R3
      LDR R4, .Time       ; get start ime
timer:
      LDR R5, .Time       ; update time
      SUB R6, R5, R4      ; calc elapsed time
      CMP R6, R3          ; compare elapsed to delay time
      BLT timer
      pop {R3,R4,R5,R6}
      RET
```

Revisiting our Flashing LED Example

- The CPU is 100% engaged in this task! It is doing nothing else but wasting CPU cycles

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;   delay function
;;   inputs R0 - time delay in seconds
delay:
      push {R3,R4,R5,R6}
      MOV R3, R0          ; move delay time param into R3
      LDR R4, .Time       ; get start ime
timer:
      LDR R5, .Time       ; update time
      SUB R6, R5, R4      ; calc elapsed time
      CMP R6, R3          ; compare elapsed to delay time
      BLT timer
      pop {R3,R4,R5,R6}
      RET
```

Revisiting our Flashing LED Example

- There is a better way ! Using Interrupts

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;  delay function
;;  inputs R0 - time delay in seconds
delay:
      push {R3,R4,R5,R6}
      MOV R3, R0          ; move delay time param into R3
      LDR R4, .Time       ; get start ime
timer:
      LDR R5, .Time       ; update time
      SUB R6, R5, R4      ; calc elapsed time
      CMP R6, R3          ; compare elapsed to delay time
      BLT timer
      pop {R3,R4,R5,R6}
      RET
```

Now back to our Flashing LED

- Lets use ARMlite's Clock Interrupt for this !
- To set it up the interrupt handling:
  - The handler will be called everytime the clock raises an interrupt
  - So each time it will draw either a green ("LED on") pixel or a white ("LED off") depending on what state the LED is currently in
- This is quite different to previously where we inserted a delay between each state change!
- Now we're waiting to be told when to change state:
  - Event driven !

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel1367 // draw the pixel
      POP {R0}
      RFE
```

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel367 // draw the pixel
      POP {R0}
      RFE
```

Check next state to enact (on or off)

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel1367 // draw the pixel
      POP {R0}
      RFE
```

next state is “off” ?
Then turn LED off

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel367 // draw the pixel
      POP {R0}
      RFE
```

next state is “ON” ?
Then turn LED ON

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel1367 // draw the pixel
      POP {R0}
      RFE
```

Draw the pixel (R0 will be set as either green or white)

Flashing LED with INT: Interrupt Handler

```
////////////////////////////////////
/// interrupt handler: toggleLED
//  toggles state of LED (on "green" or off "white")
//
toggleLED:
      PUSH {R0}
      CMP R3, #0          // check state
      BNE off
      MOV R3, #1          // if R3 0, then 1
      MOV R0, #.green
      B drawpixel
off:
      MOV R3, #0          // if R3 1, then 0
      MOV R0, #.white
drawpixel:
      STR R0, .Pixel367 // draw the pixel
      POP {R0}
      RFE
```

We push and pop R0 so we can restore it after the function is finished.

We don't do this for R3 though – why ?

Flashing LED with INT: Full Program

```
1|// Set up Interrupt handling
2|      MOV R0,#toggleLED
3|      STR R0,.ClockISR
4|      MOV R0,#1000
5|      STR R0,.ClockInterruptFrequency
6|      MOV R0,#1
7|      STR R0,.InterruptRegister //Enable all interrupts
8|      MOV R3, #1       // R3 keeps track of state of LED

9|mainProgram:
10|      B mainProgram     //Here, just an empty loop!
```

```
11|////////////////////////////////
12|/// interrupt handler: toggleLED
13|//  toggles state of LED (on "green" or off "white")
14|//
15|toggleLED:
16|      PUSH {R0}
17|      CMP R3, #0       // check state
18|      BNE off
19|      MOV R3, #1       // if R3 0, then 1
20|      MOV R0, #.green
21|      B drawpixel
22|off:
23|      MOV R3, #0       // if R3 1, then 0
24|      MOV R0, #.white
25|drawpixel:
26|      STR R0, .Pixel367 // draw the pixel
27|      POP {R0}
28|      RFE
```

Flashing LED with INT: Full Program

```
1|// Set up Interrupt handling
2|      MOV R0,#toggleLED
3|      STR R0,.ClockISR
4|      MOV R0,#1000
5|      STR R0,.ClockInterruptFrequency
6|      MOV R0,#1
7|      STR R0,.InterruptRegister //Enable all interrupts
8|      MOV R3, #1       // R3 keeps track of state of LED

9|mainProgram:
10|      B mainProgram     //Here, just an empty loop!
```

This is just an infinite loop because we are only flashing LED and there is nothing else to do but wait for clock interrupts, however this could potentially be doing other useful things

```
11|////////////////////////////////
12|/// interrupt handler: toggleLED
13|//  toggles state of LED (on "green" or off "white")
14|//
15|toggleLED:
16|      PUSH {R0}
17|      CMP R3, #0       // check state
18|      BNE off
19|      MOV R3, #1       // if R3 0, then 1
20|      MOV R0, #.green
21|      B drawpixel
22|off:
23|      MOV R3, #0       // if R3 1, then 0
24|      MOV R0, #.white
25|drawpixel:
26|      STR R0, .Pixel367 // draw the pixel
27|      POP {R0}
28|      RFE
```

Flashing LED with INT: Full Program

- Lets look in ARMlite