① Modes in processor

- Supervisor mode
- Interrupt Request Mode

] have all privileges. e.g: can change the mode bits in the CPSR

- User - few privileges

② "banked" registers in IRQ mode

→ different R13, R14 (LR, SP)

③ When IRQ happens → processor goes into IRQ mode

1. copies' CPSR → SPSR
2. LR_{IRQ} ← PC

④ How does processor know where to begin executing the interrupt service routine?

→ IRQ "Execution Vector" is @ address 0x18 → execute there → B interrupt_service

- This is just one kind of "Exception"
- one other example is the "power on/reset"
  - its vector is at address 0x0
  - ① power on reset → goes into supervisor mode
  - ② Interrupt request @ 0x18 → goes into IRQ mode
  - ③ System call → SVC instruction is executed @ address 0x8 → supervisor mode
- Last day's notes show how to set up exception vector table
- When you load program using this with monitor program - must select "Exceptions" in Linker section of New project.

(coded) exception vector table

address	.section	.vector	, "ax"
0x0	B	_start	
	.space	20	(3words skipped)
0x18	B	IRQ_handler	

```
//IRQ_handler handles all possible sources of interrupts - Lab 6 has two sources - key & timer
IRQ_handler: PUSH {R0-R12, LR}          //push (save all registers onto the stack)
             : code figures out which interrupt occurred - key, timer or ...
             //if interrupt was key
             BL  KEY_ISR                //go to specific interrupt service routine.
             //if interrupt was timer
             BL  TIMER_ISR
             //Reset register
             POP {R0-R12, LR}
             //go back to the main program that was executing when the interrupt request happened
               → PC ← LR, CPSR ← SPSR   (restores condition & sets the mode back)
               → return from interrupt instruction to what it was
             SUBS PC, LR, #4
```

- SUBS into PC causes the SPSR → CPSR transfer

More complexity / detail of interrupts in ARM A9

- overview of the hardware