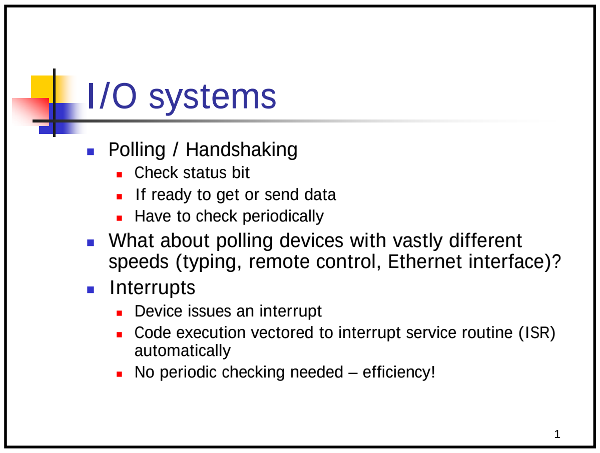
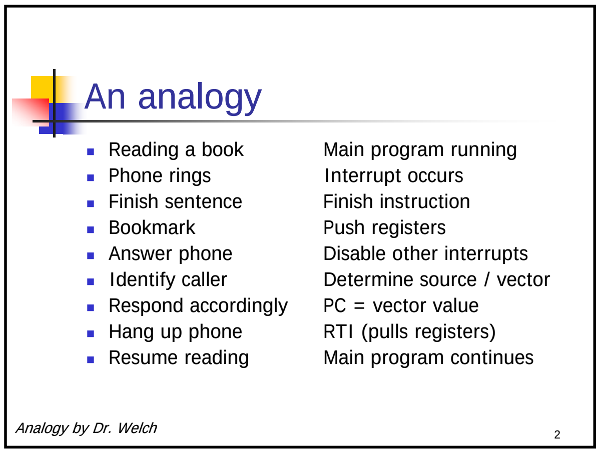


I/O systems

- Polling / Handshaking
  - Check status bit
  - If ready to get or send data
  - Have to check periodically
- What about polling devices with vastly different speeds (typing, remote control, Ethernet interface)?
- Interrupts
  - Device issues an interrupt
  - Code execution vectored to interrupt service routine (ISR) automatically
  - No periodic checking needed – efficiency!

1

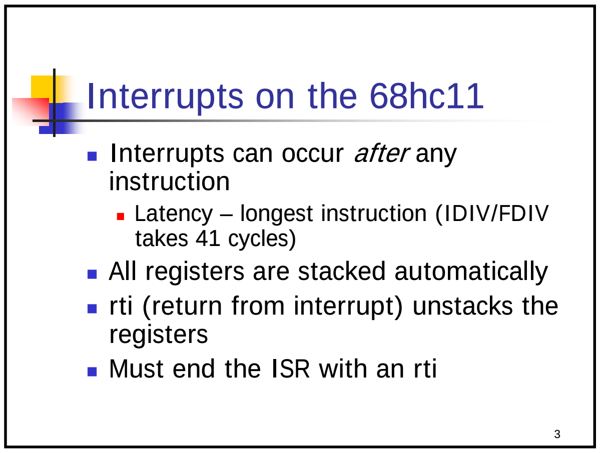


An analogy

Reading a book	Main program running
Phone rings	Interrupt occurs
Finish sentence	Finish instruction
Bookmark	Push registers
Answer phone	Disable other interrupts
Identify caller	Determine source / vector
Respond accordingly	PC = vector value
Hang up phone	RTI (pulls registers)
Resume reading	Main program continues

Analogy by Dr. Welch

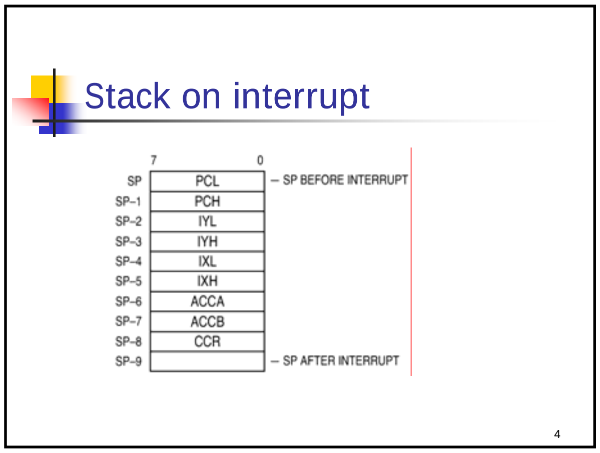
2



Interrupts on the 68hc11

- Interrupts can occur *after* any instruction
  - Latency – longest instruction (IDIV/FDIV takes 41 cycles)
- All registers are stacked automatically
- rti (return from interrupt) unstacks the registers
- Must end the ISR with an rti

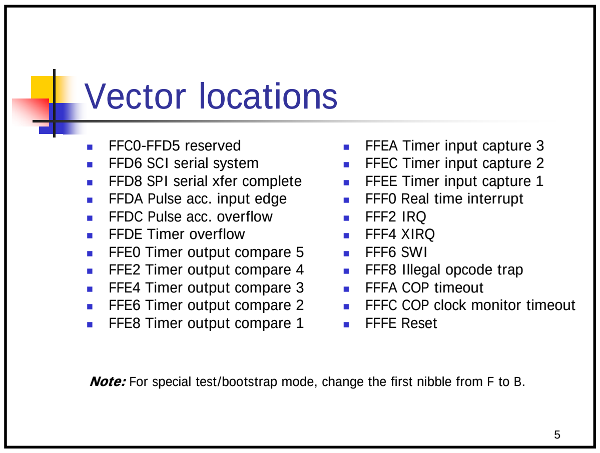
3



Stack on interrupt

	7 0	
SP	PCL	– SP BEFORE INTERRUPT
SP–1	PCH	
SP–2	IYL	
SP–3	IYH	
SP–4	IXL	
SP–5	IXH	
SP–6	ACCA	
SP–7	ACCB	
SP–8	CCR	
SP–9		– SP AFTER INTERRUPT

4

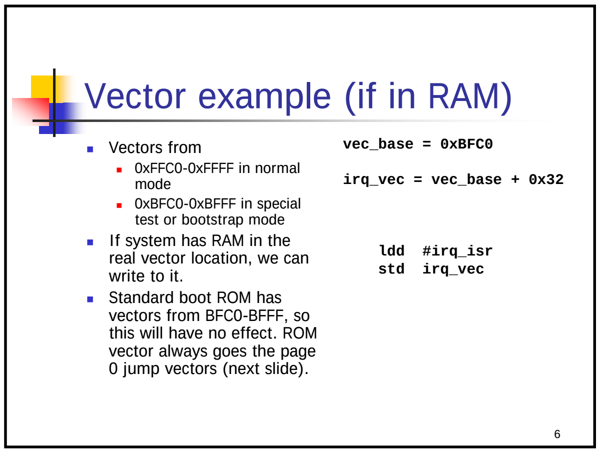


Vector locations

- FFC0-FFD5 reserved
- FFD6 SCI serial system
- FFD8 SPI serial xfer complete
- FFDA Pulse acc. input edge
- FFDC Pulse acc. overflow
- FFDE Timer overflow
- FFE0 Timer output compare 5
- FFE2 Timer output compare 4
- FFE4 Timer output compare 3
- FFE6 Timer output compare 2
- FFE8 Timer output compare 1
- FFEA Timer input capture 3
- FFEC Timer input capture 2
- FFEE Timer input capture 1
- FFF0 Real time interrupt
- FFF2 IRQ
- FFF4 XIRQ
- FFF6 SWI
- FFF8 Illegal opcode trap
- FFFA COP timeout
- FFFC COP clock monitor timeout
- FFFE Reset

Note: For special test/bootstrap mode, change the first nibble from F to B.

5



Vector example (if in RAM)

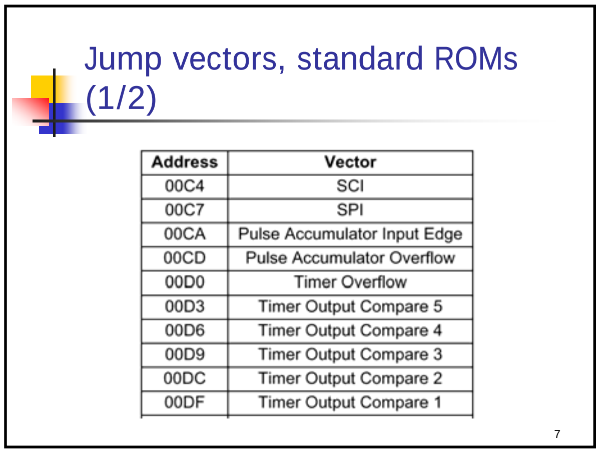
- Vectors from
  - 0xFFC0-0xFFFF in normal mode
  - 0xBFC0-0xBFFF in special test or bootstrap mode
- If system has RAM in the real vector location, we can write to it.
- Standard boot ROM has vectors from BFC0-BFFF, so this will have no effect. ROM vector always goes the page 0 jump vectors (next slide).

```
vec_base = 0xBFC0

irq_vec = vec_base + 0x32


    ldd  #irq_isr
    std  irq_vec
```

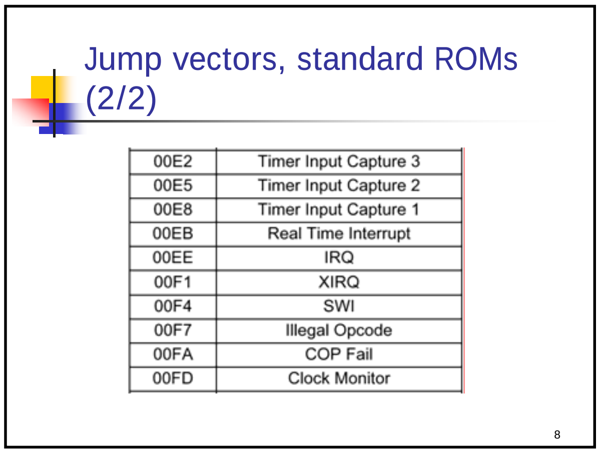
6



Jump vectors, standard ROMs (1/2)

Address	Vector
00C4	SCI
00C7	SPI
00CA	Pulse Accumulator Input Edge
00CD	Pulse Accumulator Overflow
00D0	Timer Overflow
00D3	Timer Output Compare 5
00D6	Timer Output Compare 4
00D9	Timer Output Compare 3
00DC	Timer Output Compare 2
00DF	Timer Output Compare 1

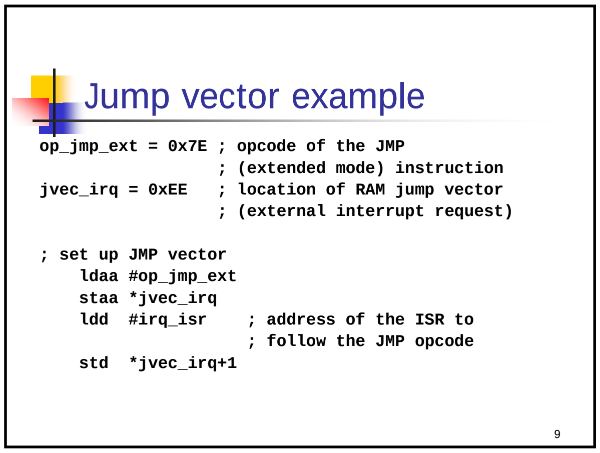
7



Jump vectors, standard ROMs (2/2)

00E2	Timer Input Capture 3
00E5	Timer Input Capture 2
00E8	Timer Input Capture 1
00EB	Real Time Interrupt
00EE	IRQ
00F1	XIRQ
00F4	SWI
00F7	Illegal Opcode
00FA	COP Fail
00FD	Clock Monitor

8



Jump vector example

```
op_jmp_ext = 0x7E ; opcode of the JMP
                  ; (extended mode) instruction
jvec_irq = 0xEE   ; location of RAM jump vector
                  ; (external interrupt request)

; set up JMP vector
    ldaa #op_jmp_ext
    staa *jvec_irq
    ldd  #irq_isr    ; address of the ISR to
                     ; follow the JMP opcode
    std  *jvec_irq+1
```

9

How do you turn on interrupts?

- Interrupts above IRQ in the previous tables are turned on/off by I bit in CCR
- CLI allows interrupts to occur
- SEI turns off interrupts
- I bit set during an interrupt so the interrupt does not interrupt itself

10

Interrupts – subsystem-specific steps

- Most HC11 subsystems (*e.g.*, timers) require additional handling
  - Initialization: set interrupt enable bit
  - Processing complete flag (tell subsystem to go on to next sample, etc.)

11

SWI – software interrupt

- Instruction that triggers an interrupt
- Uses…
  - Test portions of an ISR for hardware that isn't ready yet → install its vector in the SWI vector and write a test program
  - Convenience – like a subroutine that automatically preserves caller's registers (takes time, but not extra code)…
- Note: often used by debuggers and talkers

12

SWI example

```
vec_swi = 0xFFF6
        .section .text
        .global _start

_start: lds     #_stack

        ldd     #swi_isr
        std     vec_swi

        cli

again:  swi
        bra    again
```

```
swi_isr:
; no push/pull
; do subroutine stuff

        rti    ; (not rts)
```

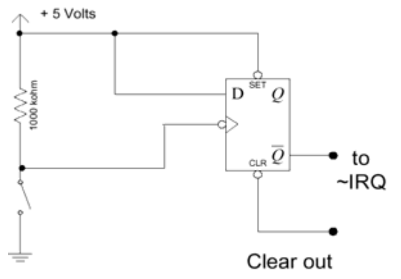
13

~IRQ / ~XIRQ

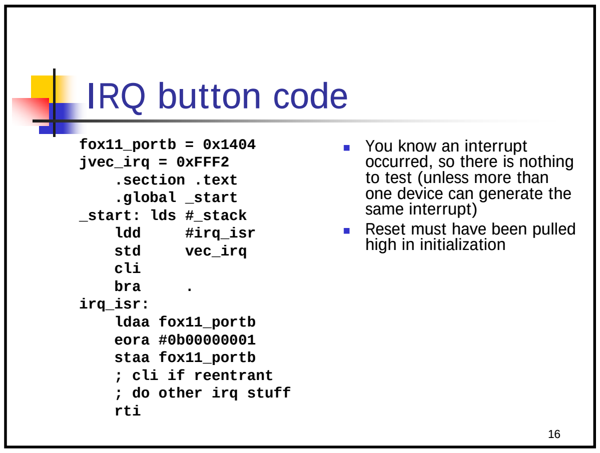
- Pins on the HC11
- ~IRQ: Recognized if CCR.I is cleared
- ~XIRQ: Recognized if CCR.X is cleared
  - CCR.X, once cleared, can only be set when chip is reset
  - Once X cleared by software, XIRQ behaves as a non-maskable interrupt

14

IRQ button hardware



15

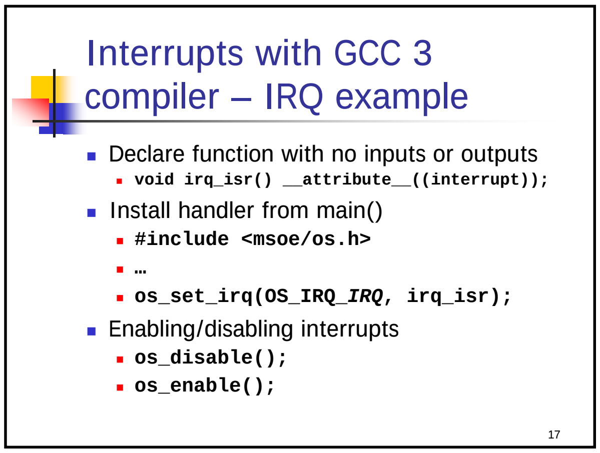


IRQ button code

```
fox11_portb = 0x1404
jvec_irq = 0xFFF2
    .section .text
    .global _start
_start: lds #_stack
    ldd     #irq_isr
    std     vec_irq
    cli
    bra     .
irq_isr:
    ldaa fox11_portb
    eora #0b00000001
    staa fox11_portb
    ; cli if reentrant
    ; do other irq stuff
    rti
```

- You know an interrupt occurred, so there is nothing to test (unless more than one device can generate the same interrupt)
- Reset must have been pulled high in initialization

16



Interrupts with GCC 3 compiler – IRQ example

- Declare function with no inputs or outputs
  - `void irq_isr() __attribute__((interrupt));`
- Install handler from main()
  - `#include <msoe/os.h>`
  - …
  - `os_set_irq(OS_IRQ_`***`IRQ`***`, irq_isr);`
- Enabling/disabling interrupts
  - `os_disable();`
  - `os_enable();`

17