22. Example Using Interrupts November 14, 2018
Chapter 10

- Review
  - Interrupt-driven I/O

- Multiple interrupts

- Interrupt state diagram and structure

- Examples
  - Executing two concurrent tasks
  - Interrupt handler code

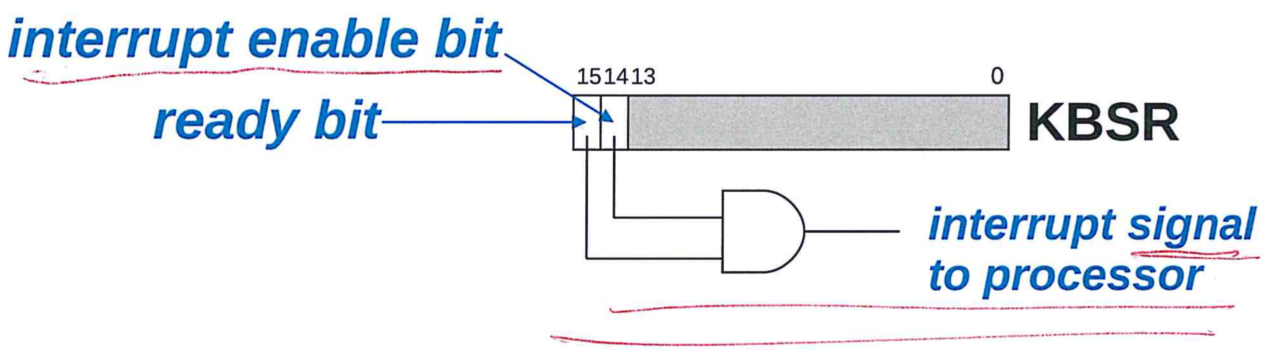
Interrupt-Driven I/O

To implement an interrupt mechanism, we need:

- A way for the I/O device to signal the CPU that an interesting event has occurred.
- A way for the CPU to test whether the interrupt signal is set and whether its priority is higher than the current program.

Generating Signal

- Software sets "interrupt enable" bit in device register.
- When ready bit is set and IE bit is set, interrupt is signaled.



Priority

Every instruction executes at a stated level of urgency.

LC-3: 8 priority levels (PL0-PL7)

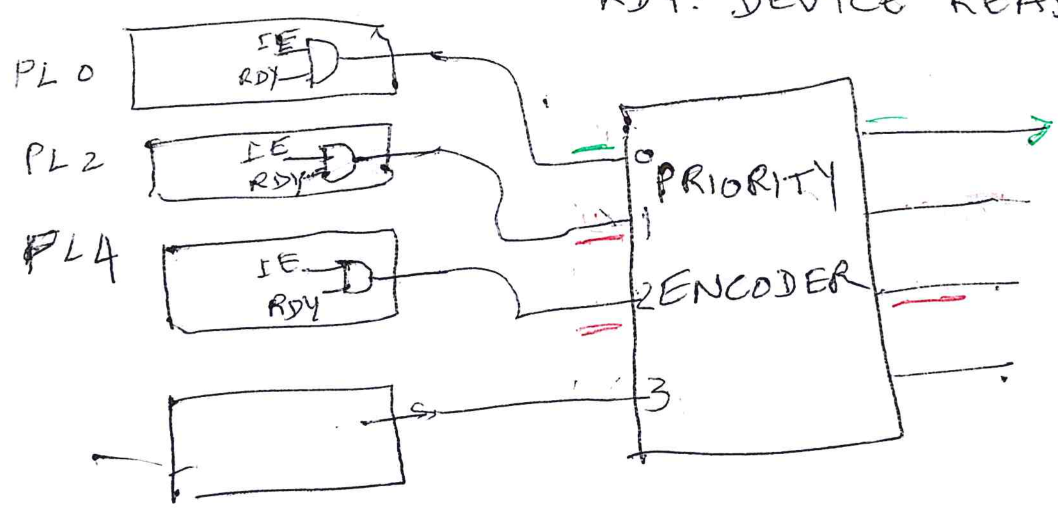
LOW HIGH

- **Example:**
  - ➢ **Payroll program runs at PL0.**
  - ➢ **Nuclear power correction program runs at PL6.**

- **It's OK for PL6 device to interrupt PL0 program, but not the other way around.**

Priority encoder selects highest-priority device, compares to current processor priority level, and generates interrupt signal if appropriate.

IE: INTR. ENABLE
RDY: DEVICE READY

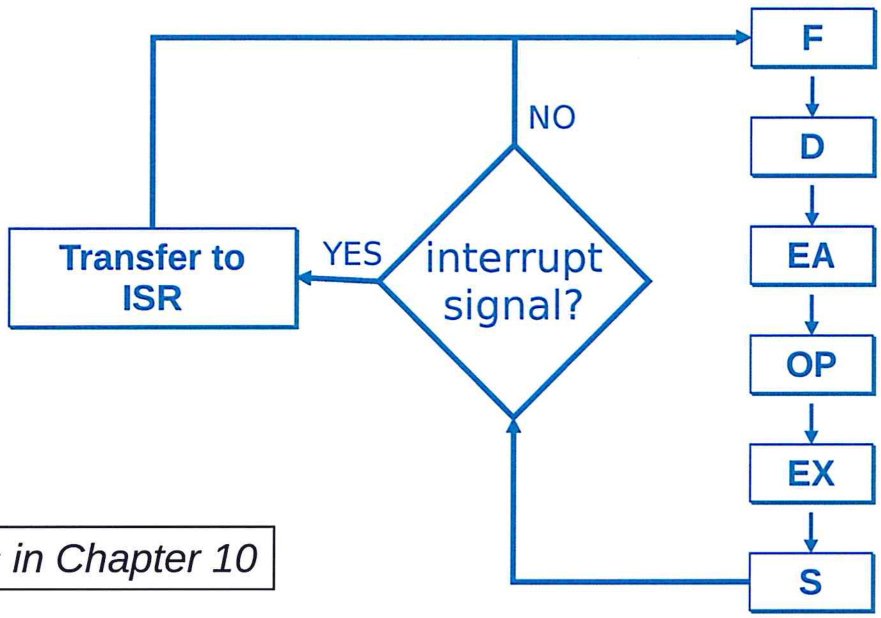


Testing for Interrupt Signal

CPU looks at signal between STORE and FETCH phases.

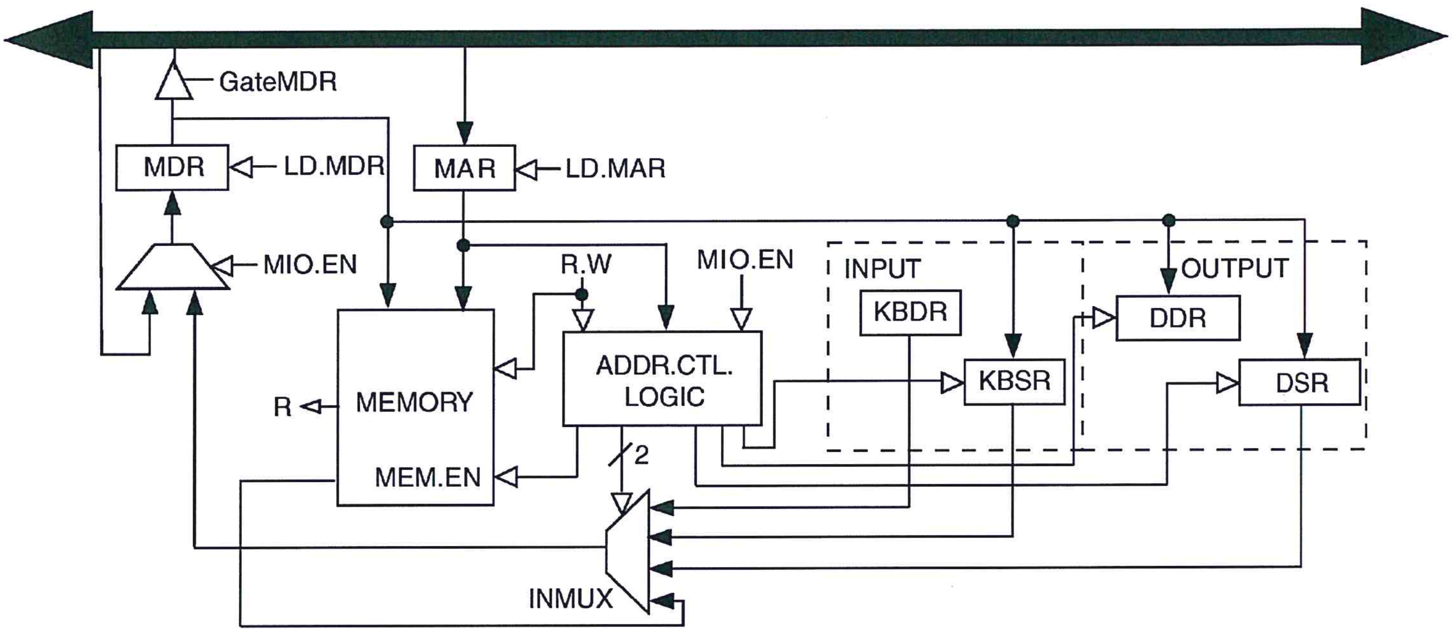
If not set, continues with next instruction.

If set, transfers control to interrupt service routine.



Details in Chapter 10

Full Implementation of LC-3 Memory-Mapped I/O



Because of interrupt enable bits, status registers (KBSR/DSR) must be written, as well as read.

Interrupt-Driven I/O (Part 2)

Interrupts were introduced in Chapter 8

1. External device signals need to be serviced
2. Processor saves state and starts service routine
3. When finished, processor restores state and resumes program

> *Interrupt is an* ***unscripted subroutine call****,*
> *triggered by an external event.*

Chapter 8 didn't explain how (2) and (3) occur, because it involves a stack

Now, we're ready…

Processor State

What state is needed to completely capture the state of a running process?

Processor Status Register

- Privilege [15], Priority Level [10:8], Condition Codes [2:0]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P						PL							N	Z	P

Program Counter

- Pointer to next instruction to be executed.

Registers

- All temporary state of the process that’s not stored in memory.

Save State on a Stack

Supervisor Stack

A special region of memory used as the stack for interrupt service routines

- Initial Supervisor Stack Pointer (SSP) stored in Saved.SSP
- Another register for storing User Stack Pointer (USP): Saved.USP

Want to use R6 as stack pointer

- So that our PUSH/POP routines still work

When switching from User mode to Supervisor mode (as result of interrupt), save R6 to Saved.USP

Invoking the Service Routine – The Details

1. If Priv = 1 (user),
   Saved.USP = R6, then R6 = Saved.SSP.
2. Push PSR and PC to Supervisor Stack.
3. Set PSR[15] = 0 (supervisor mode).
4. Set PSR[10:8] = priority of interrupt being serviced.
5. Set PSR[2:0] = 0.
6. Set MAR = x01vv, where vv = 8-bit interrupt vector provided by interrupting device (e.g., keyboard = x80).
7. Load memory location (M[x01vv]) into MDR.
8. Set PC = MDR; now first instruction of ISR will be fetched.

Note: This all happens between the STORE RESULT of the last user instruction and the FETCH of the first ISR instruction.

Returning from Interrupt

Special instruction – RTI – that restores state.

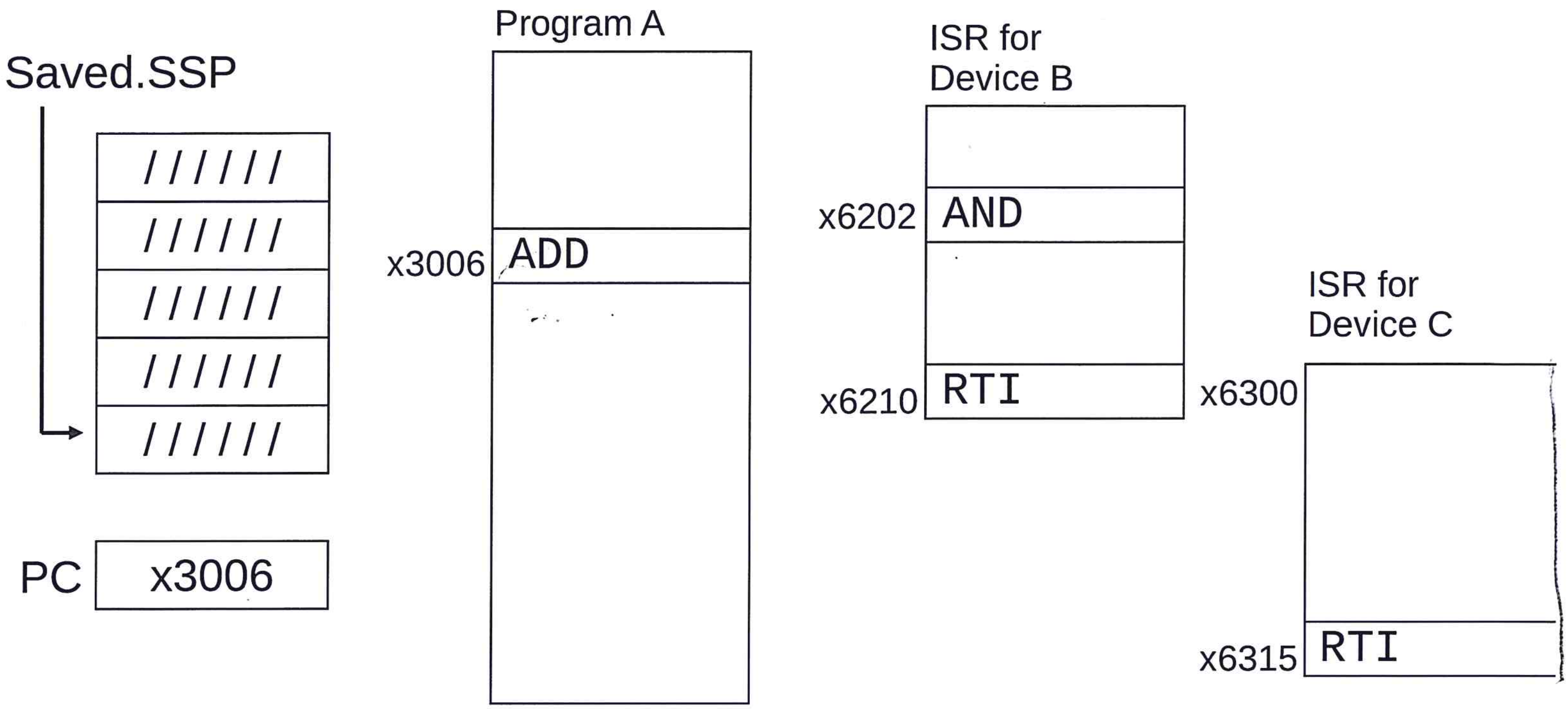
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. **Pop PC from supervisor stack.** (PC = M[R6]; R6 = R6 + 1)
2. **Pop PSR from supervisor stack.** (PSR = M[R6]; R6 = R6 + 1)
3. **If PSR[15] = 1, R6 = Saved.USP.**
   (If going back to user mode, need to restore User Stack Pointer.)

RTI is a privileged instruction.

- **Can only be executed in Supervisor Mode**
- **If executed in User Mode, causes an exception**

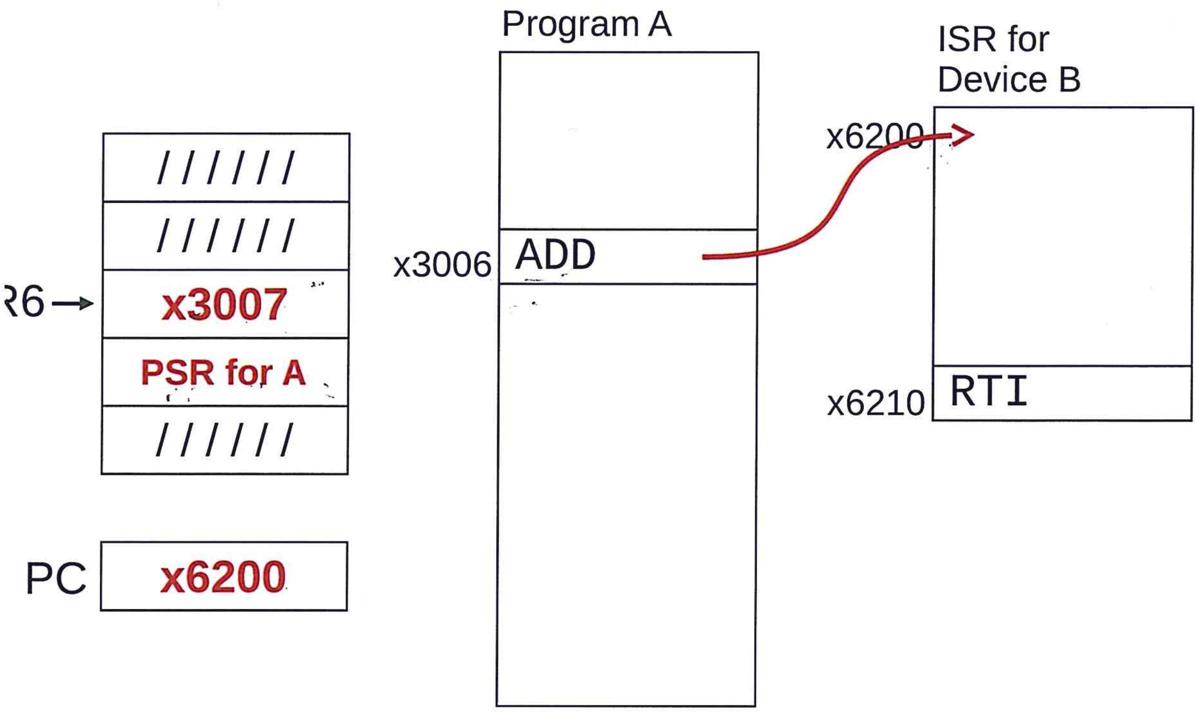
Example



Executing ADD at location x3006 when Device B interrupts

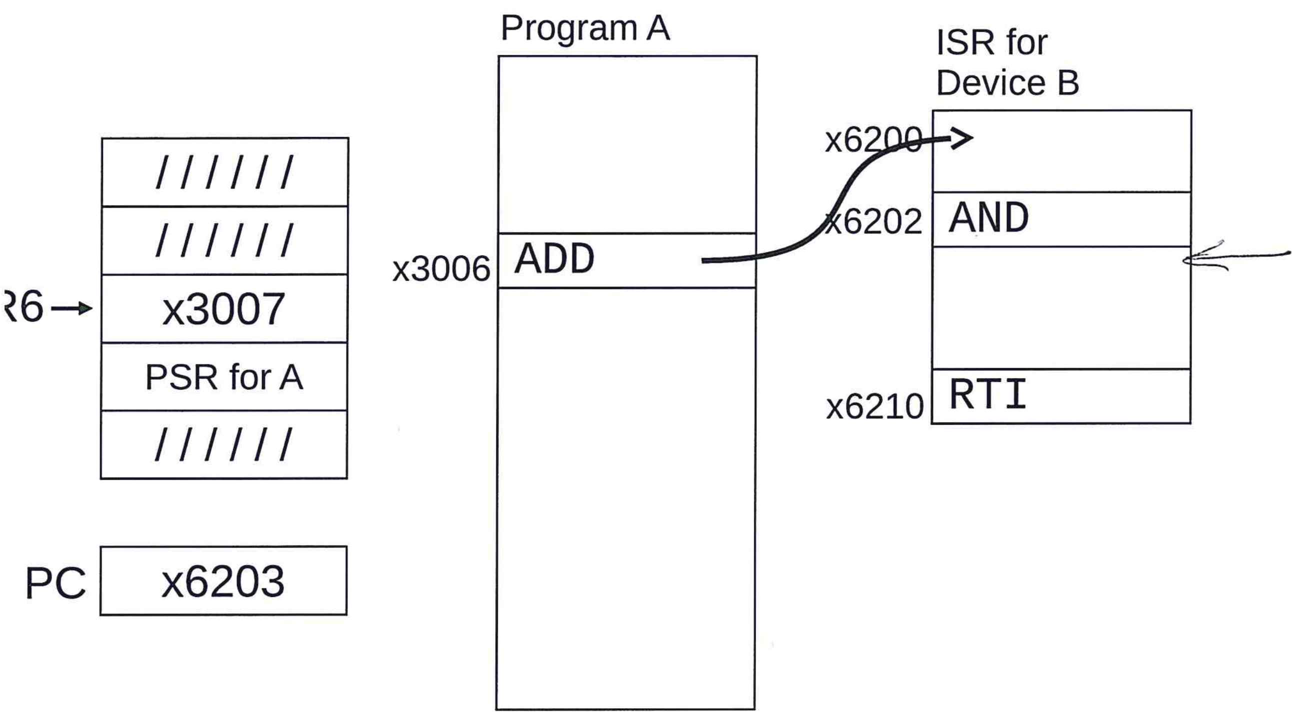
Executing AND at x6202 when Device C interrupts.

Example (2)



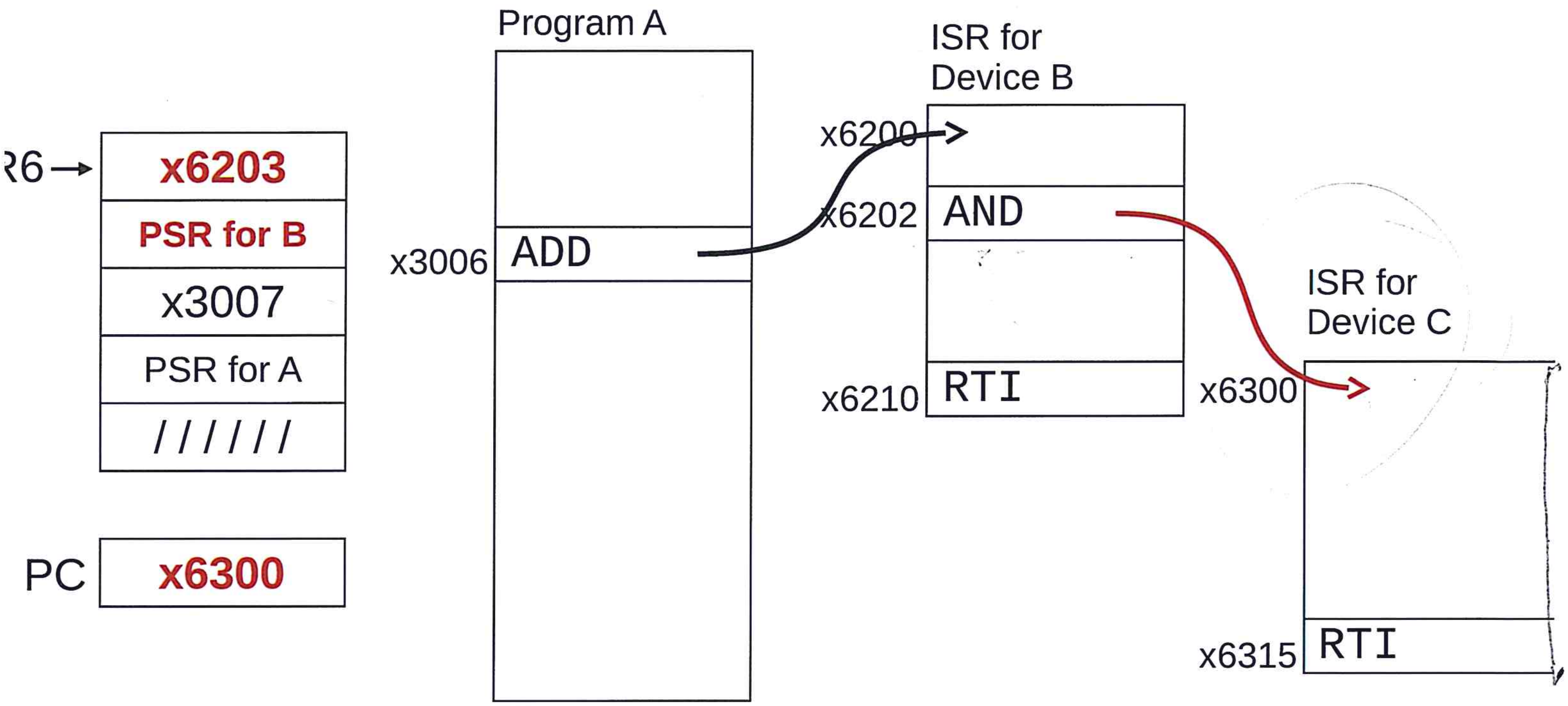
Saved.USP = R6. R6 = Saved.SSP.
Push PSR and PC onto stack, then transfer to
Device B service routine (at x6200).

Example (3)



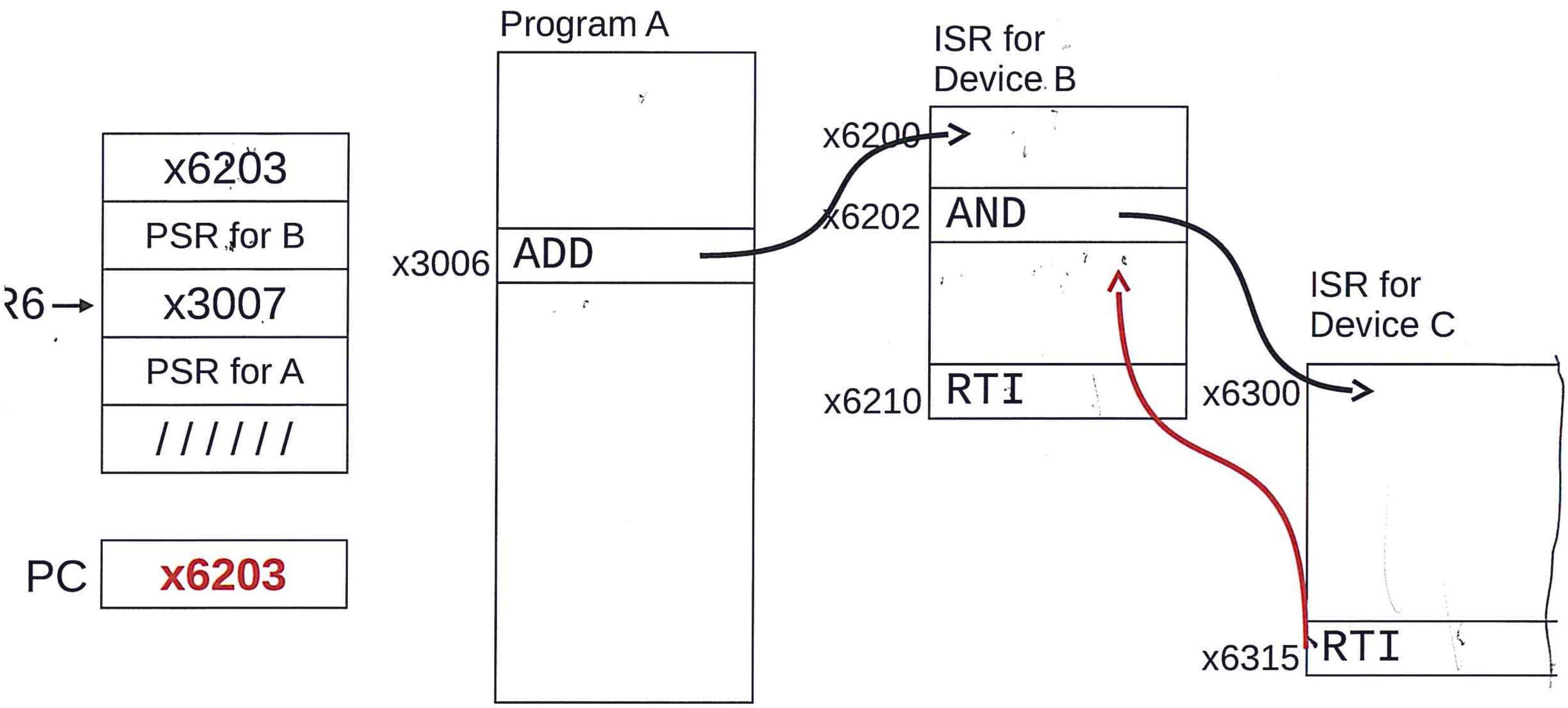
Executing AND at x6202 when Device C interrupts.

Example (4)



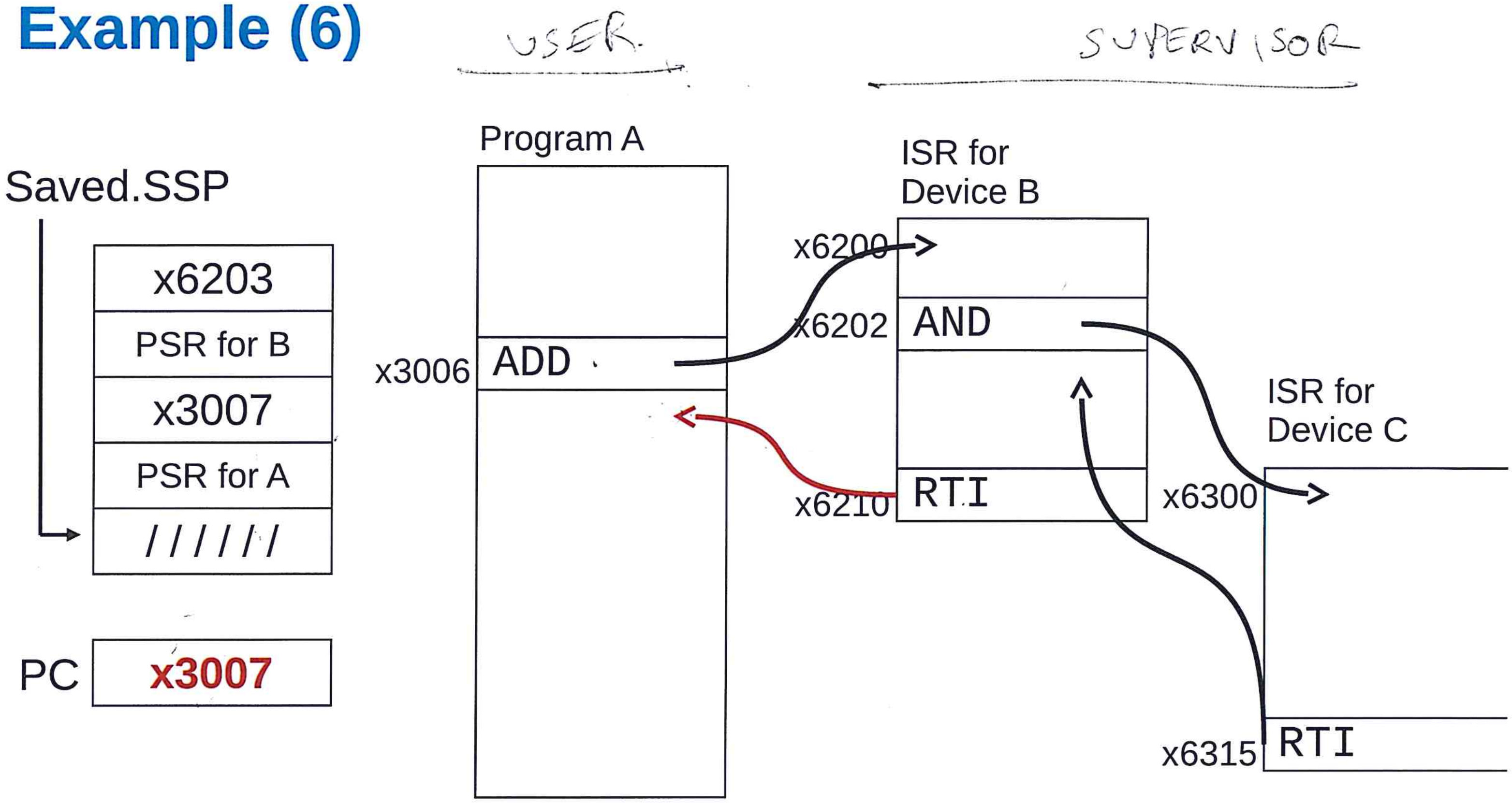
Push PSR and PC onto stack, then transfer to Device C service routine (at x6300).

Example (5)



Execute RTI at x6315; pop PC and PSR from stack.

Example (6)



Execute RTI at x6210; pop PSR and PC from stack.
Restore R6. Continue Program A as if nothing happened.

Exception: Internal Interrupt

When something unexpected happens
inside the processor, it may cause an exception

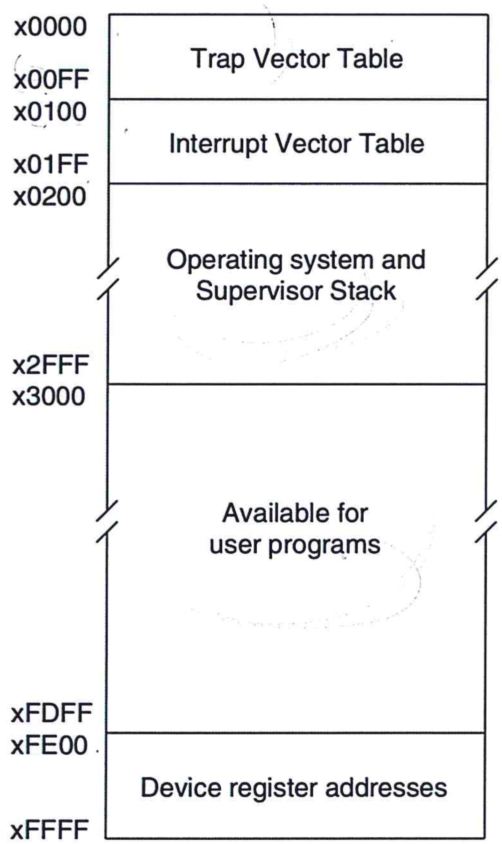
Examples:

- Privileged operation (e.g., RTI in user mode)
- Executing an illegal opcode
- Divide by zero
- Accessing an illegal address (e.g., protected system memory)

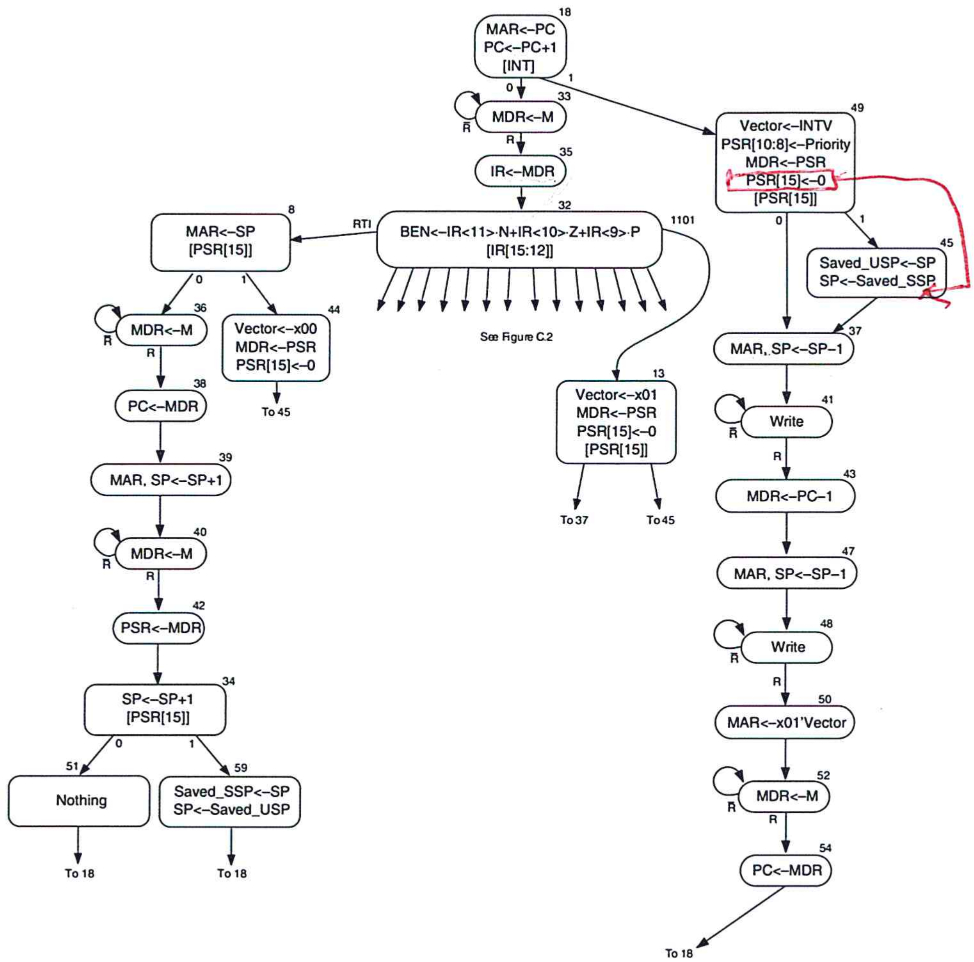
Handled just like an interrupt

- Vector is determined internally by type of exception
- Priority is the same as running program

LC-3 Memory Map (Fig. A.1)



State Diagram for Interrupt Processing



LC-3 Interrupt Structure (Fig. C.8)