Interrupts

How do we know a device is ready?

--- WAIT for device

LOOP, reading the status register

OR

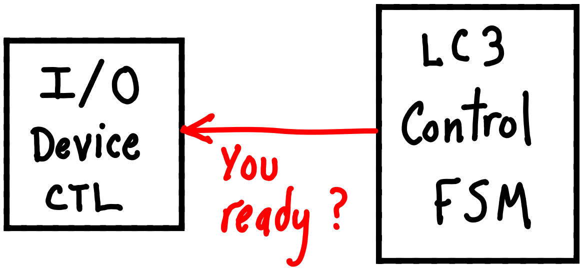
---- DON'T WAIT
Have device tell us. But,

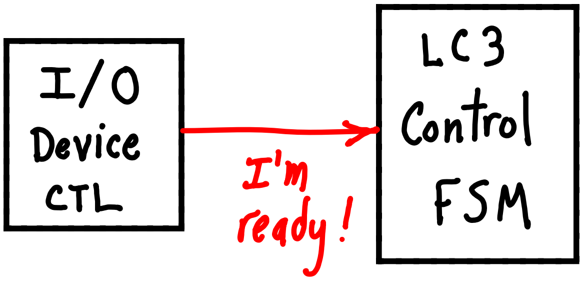
How can device talk to us?

When will it speak?

What should we do then?

What about currently executing program?

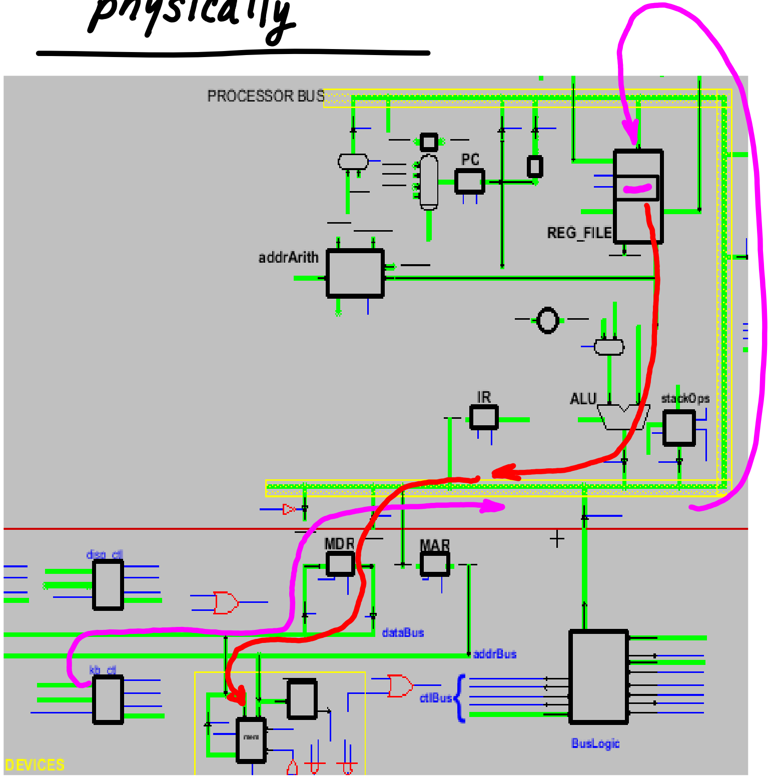




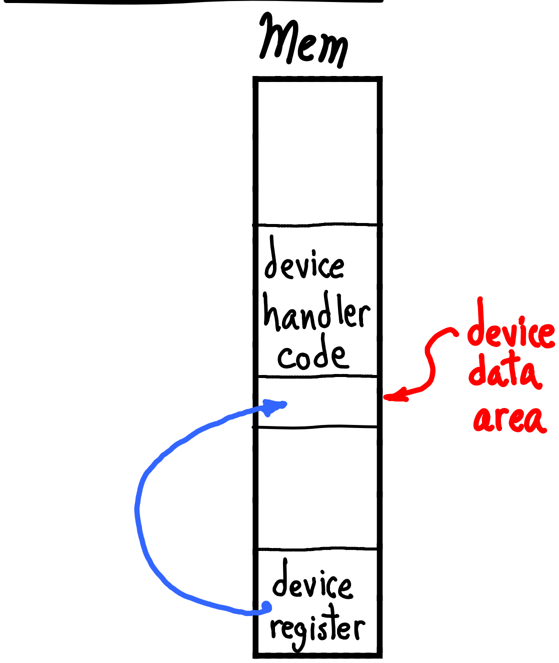
why not wait?
Got something else to do?

KB handler's job is to move data from device register to memory.

physically



Logically



Listening for the Device's Call

--- the interrupt process

Enabling interrupts:

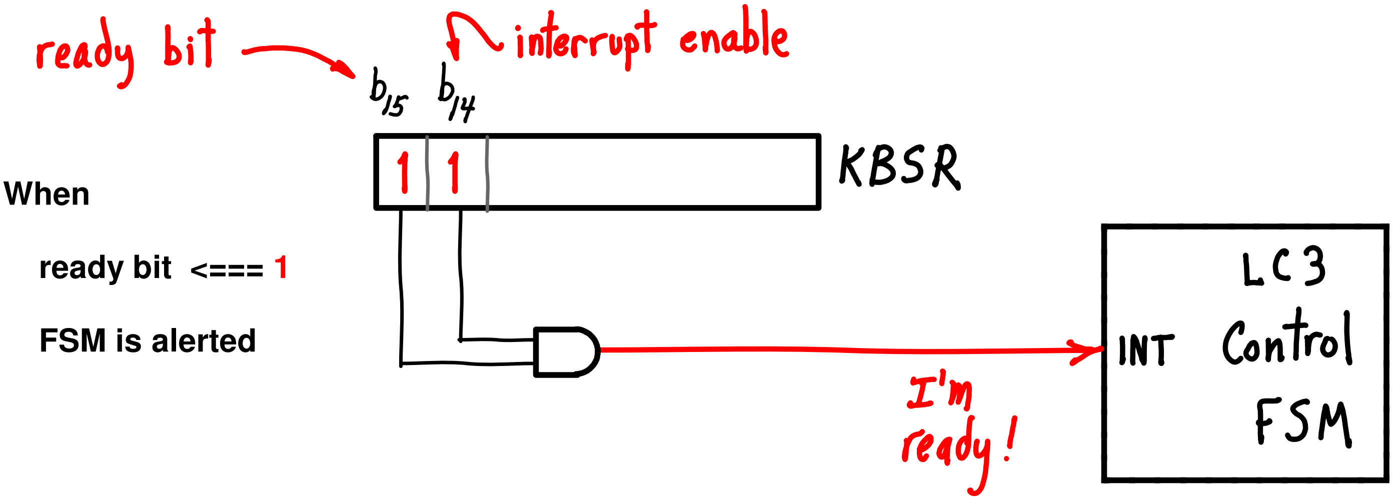
KBSR[14] <=== 1

allows controller to be interrupted.

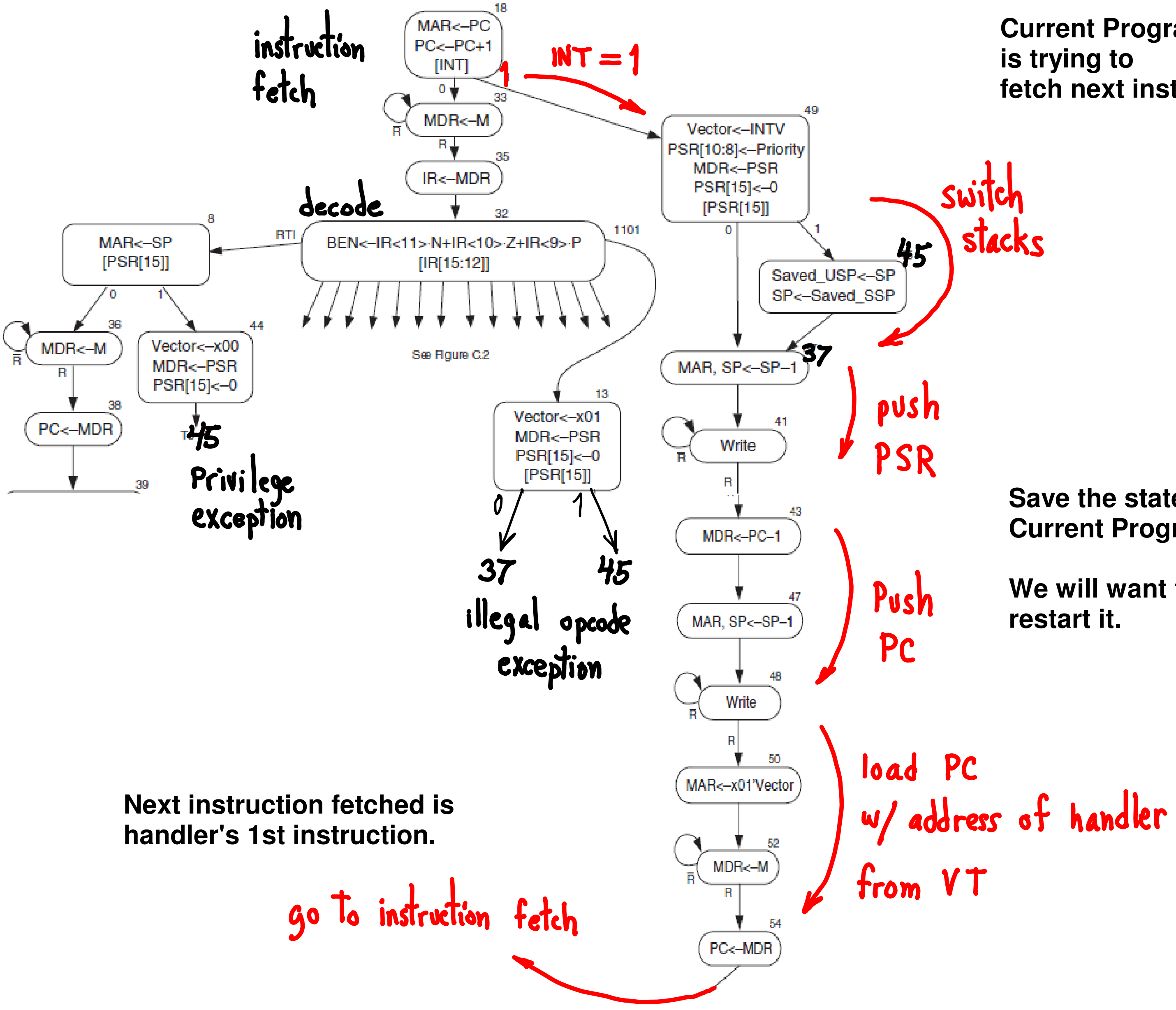
When

ready bit <=== 1

FSM is alerted



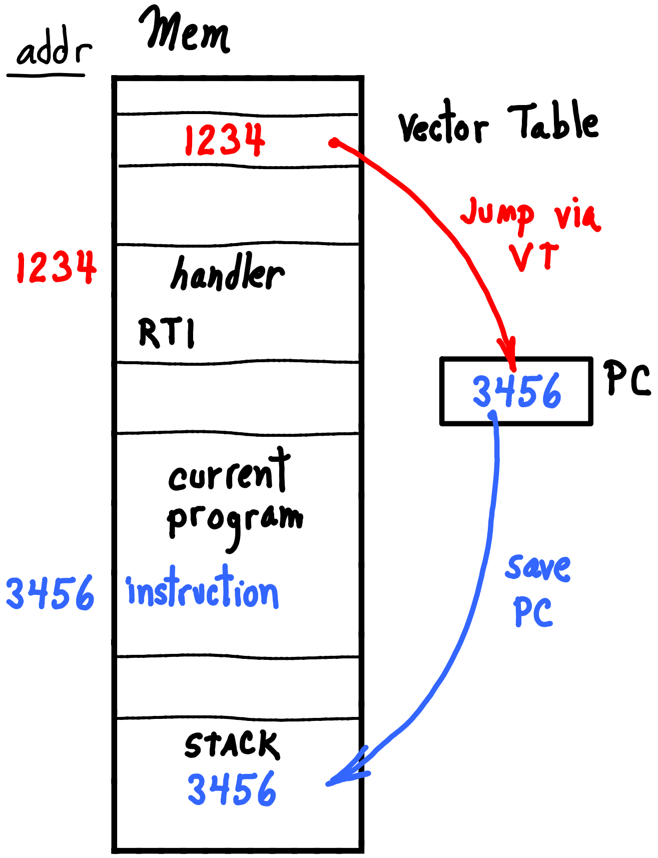
Current Program is trying to fetch next instruction



Save the state of Current Program.

We will want to restart it.

Next instruction fetched is handler's 1st instruction.



The Effect:

Device does pseudo "Trap";
gets OS's attention.

PC of Current Program saved.

Handler saves registers as needed.
Handler services device.
Handler returns by executing RTI.

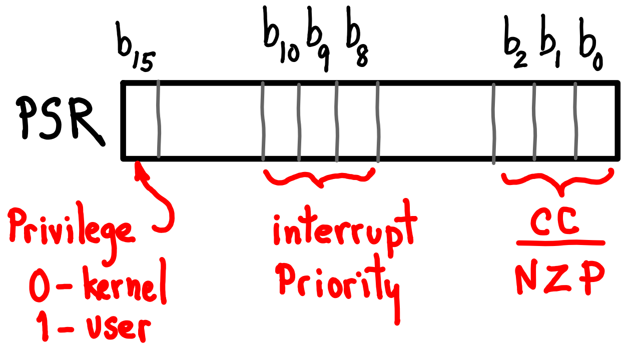
RTI unsaves PSR, PC;
restarts Current Program.

Current Program never knows anything happened
(unless checks w/ OS).

If OS is designed so that another program can be executed,
saves a lot of cycles versus polling.

Save PSR, why?

Defines Current Program's



Privilege bit:

supervisor (0) can execute
some instructions that
user (1) cannot.

supervisor (0) can R/W
some memory locations that
user (1) cannot.

Priority bits:

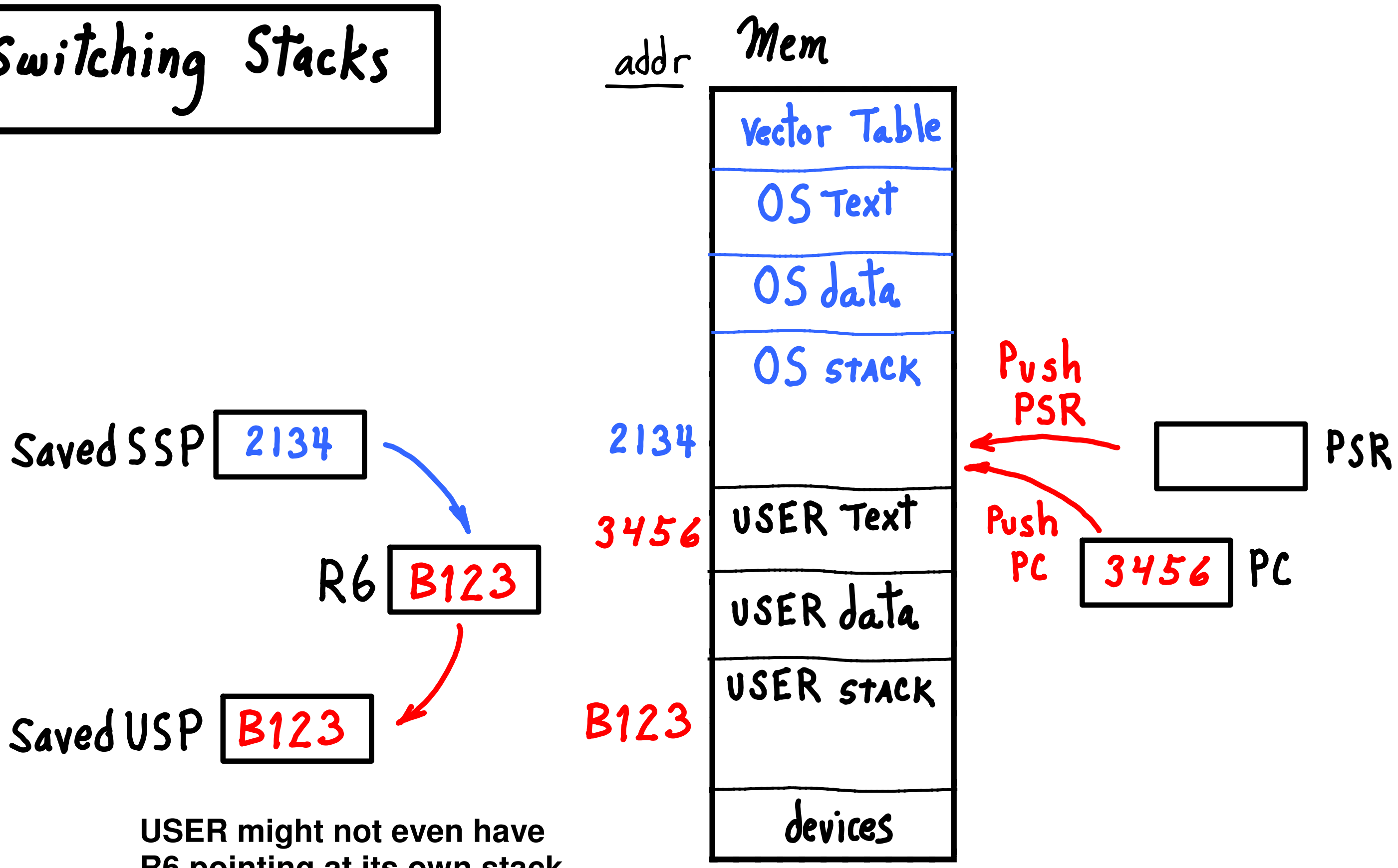
Higher priority code,
cannot be interrupted by
lower priority code.
(Handler's for prioritized devices.)

CC (**NZP**) bits:

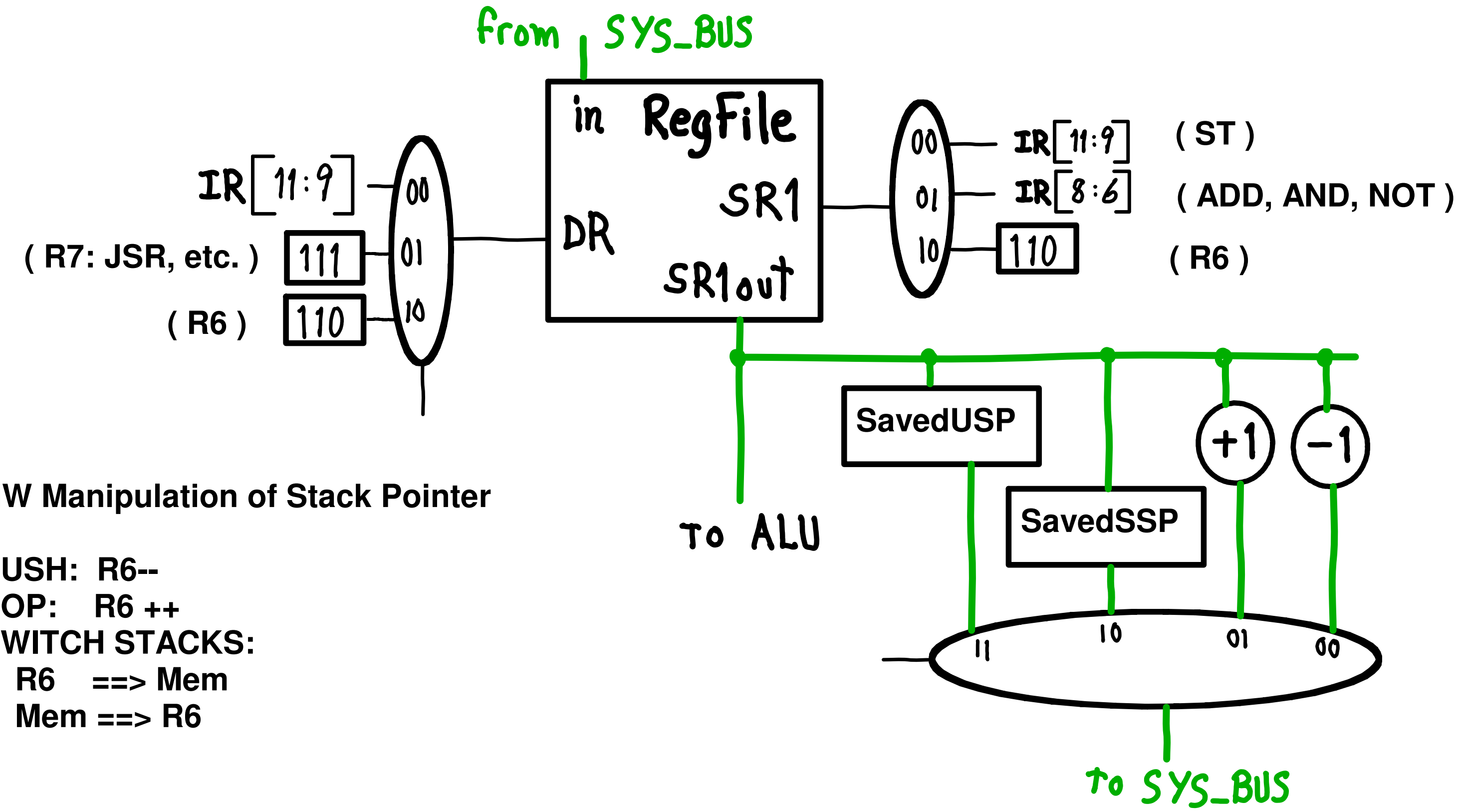
Branching depends on this,
must be saved on interrupt.

Current Program would not make
correct branches if CC not saved.

Switching Stacks



USER might not even have R6 pointing at its own stack.



HW Manipulation of Stack Pointer

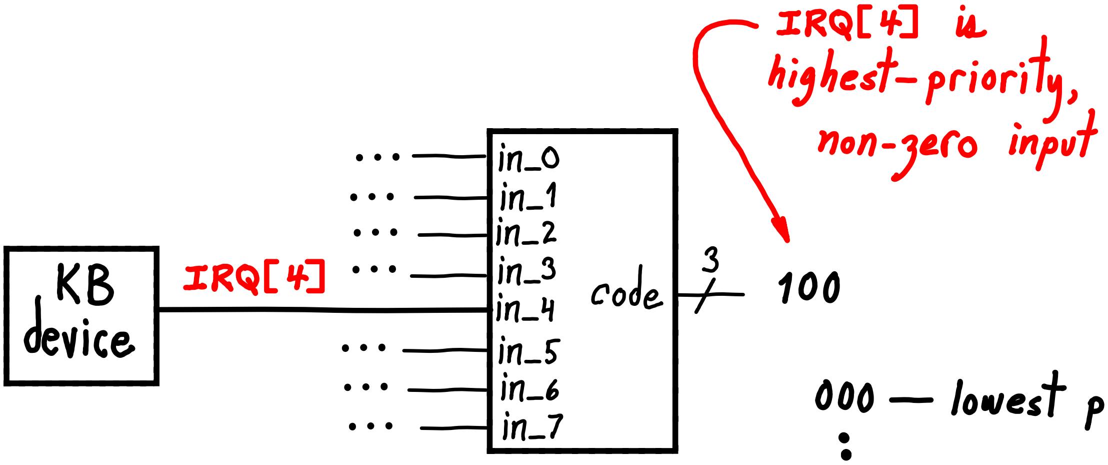
PUSH: R6--
POP: R6 ++
SWITCH STACKS:
R6 ==> Mem
Mem ==> R6

Priority encoding

How do we deal w/ multiple devices?

PRIORITY ENCODER

8 1-bit inputs ===> 3-bit code for highest-priority device.



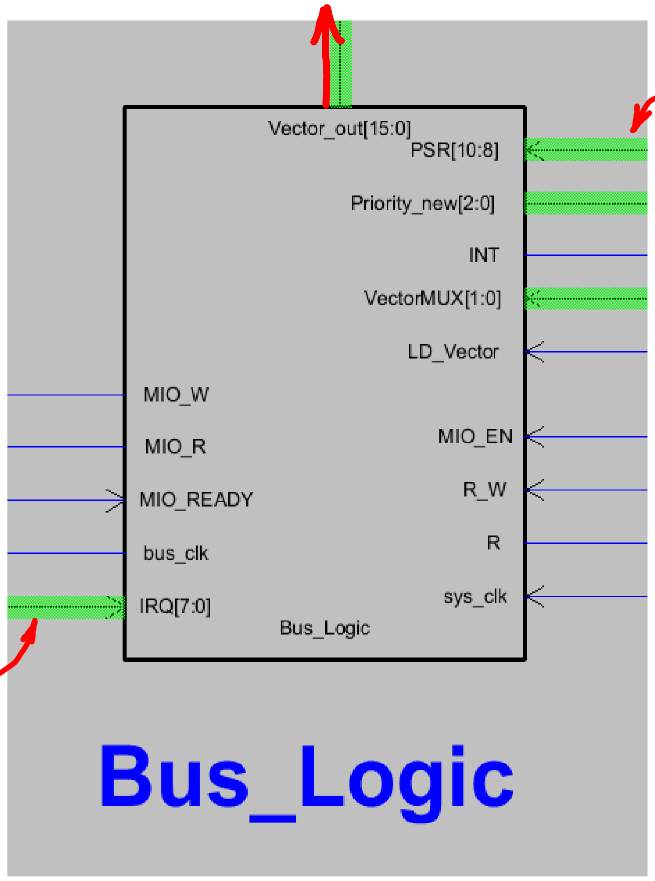
000 — lowest priority
⋮
111 — highest priority

Quirk: IRQ[0] == 1 is only non-zero input ===> code == 000
All inputs == 0 ===> code == 000

Extra output: NotZero

Interrupt Request priority encoder is part of BUS LOGIC

address of location in Vector Table, to MAR



CTL BUS

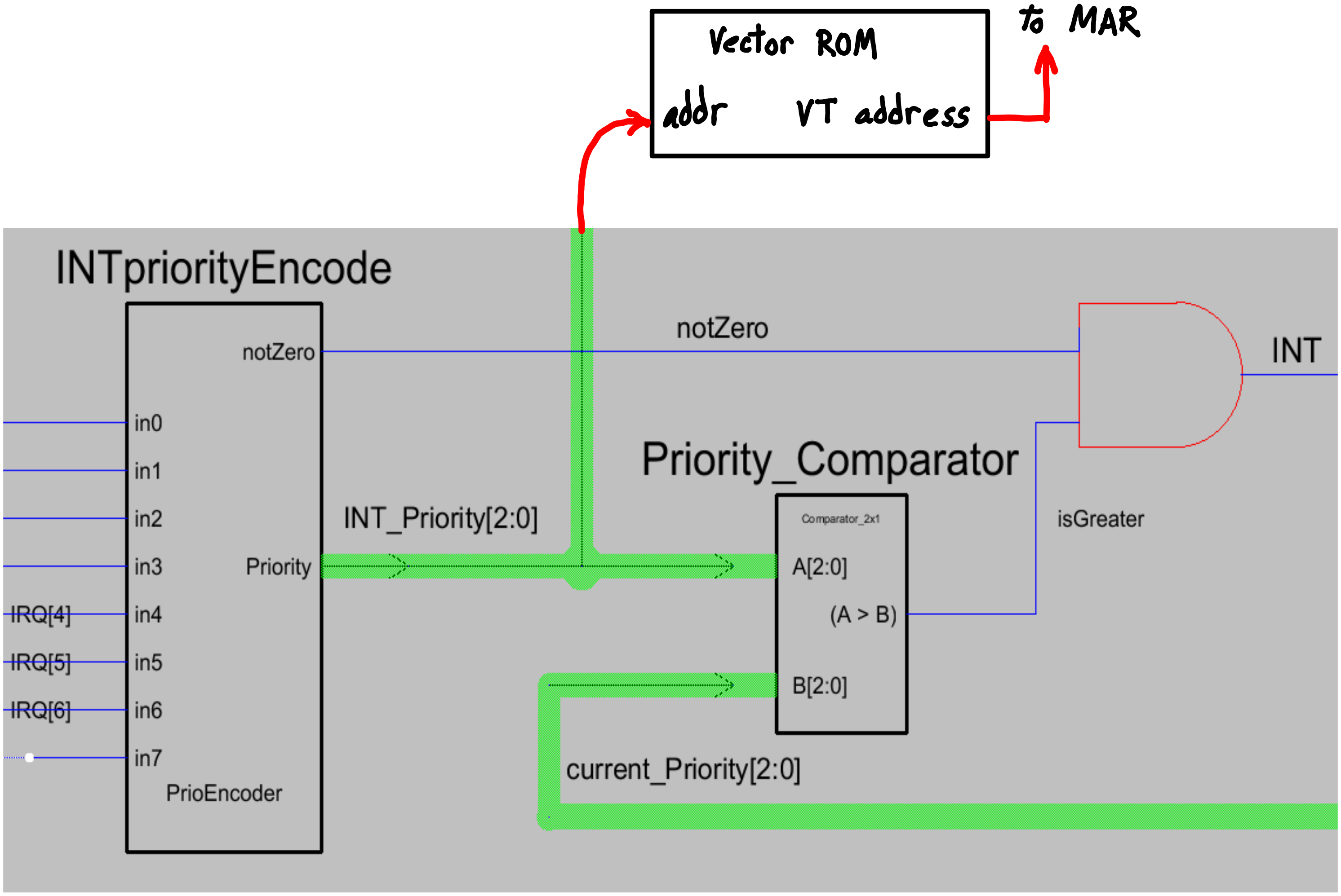
IRQ[7:0]

PSR[10:8]
current program's priority

LC3 Controller signals

Interrupted program has priority PSR[10:8].

in BUS LOGIC

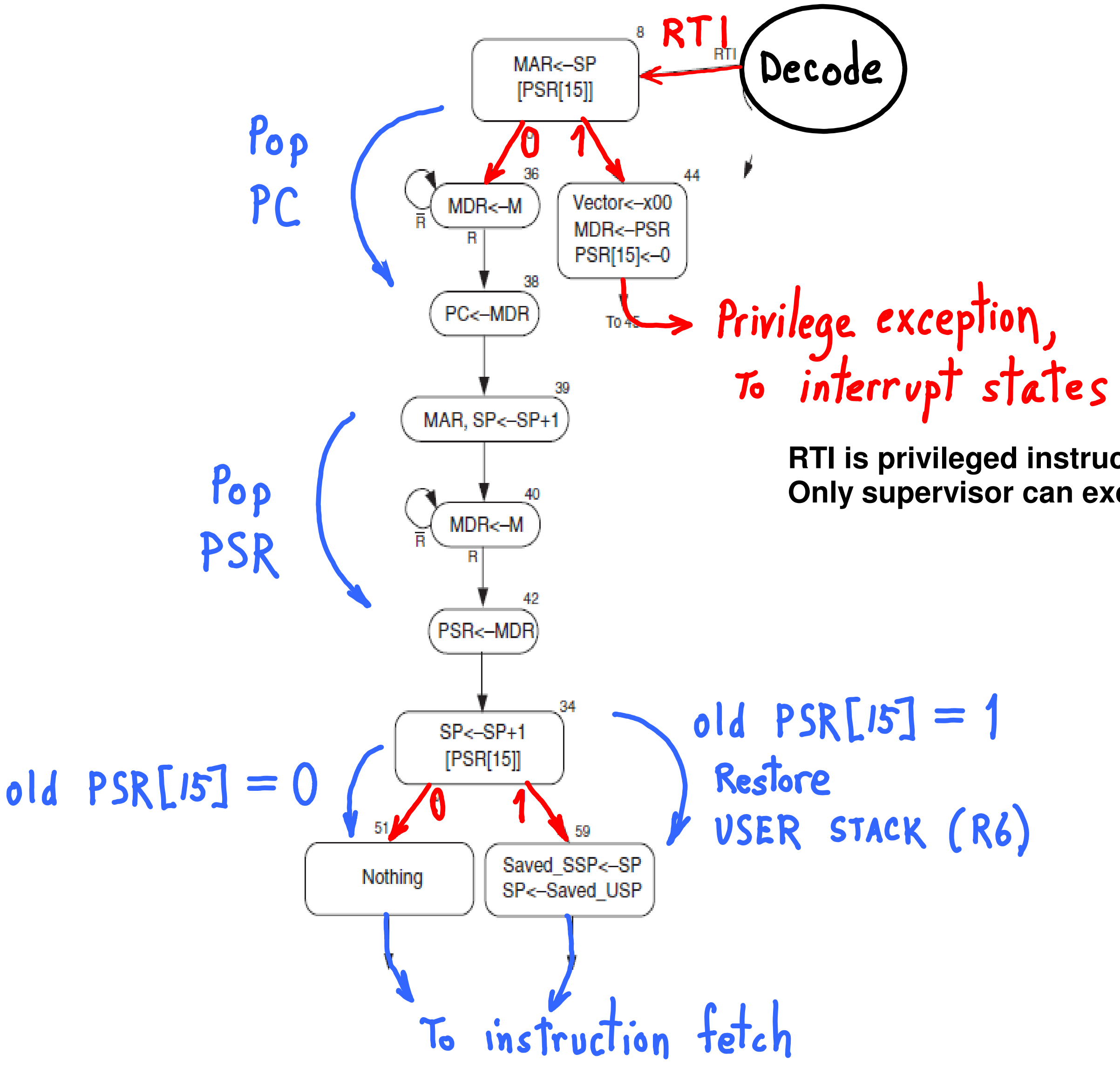


IF ((INT_Priority > current_Priority) AND notZero)

1 ===> INT

RTI instruction: restore interrupted program

10000 ··· 00 IR



RTI is privileged instruction,
Only supervisor can execute.

```
;;;===============================
;;;-- OS boot/initialization
;;;===============================

 ;;---- Set up super's stack.
 LD  R6, SUPER_STACK_ADDR

 ;;---- Init traps, exceptions, and interrupts.
 JSR kb_init

 ;;---- jump to main(), never returns.
 JSR mainOS
```

```
    ;;;------  USER code
      ...
    TRAP x33    ;;;---- Get KB data
     ...         ;;;---- Use KB data
```

```
;;;-------------------------------
;;;-- kbInt - VT x0180:
;;;--     Keyboard interrupt service
;;;-------------------------------
kb_init:
    ;;;-- Set-up interrupt vector.
    LEA R1, kb_INT
    STI  R1, KB_INT_vector
    ;;;-- Set-up KB_Data_Buffer.
    ;;;-- Set-up Trap routine vector.
    ;;;-- Enable KB interrupts.
    JMP R7

kb_INT:
    ;;;---- Disable interrupts, KBSR[14] <== 0.
    ;;;---- Read KBDR, store data.
    LDI R0, KBDR
    STI R0, KB_Buff_head
    ;;;-- Move head pointer.
    ;;;-- Enable interrupts, KBSR[14] <== 1.
    RTI

kb_Trap (x33):
    ;;;-- KB request-data service.
    ;;;-- Send data to user from buffer.
    ;;;-- (If no data, switch to other program.)
    ...
    JMP R7

kb_ConstantDataArea:
  KB_INT_vector:        .FILL x0180
  KB_TRAP_vector :      .FILL x0033
  KBSR:                  .FILL xFE00
  KBDR:                 .FILL xFE02
kb_VariableDataArea:
  KB_Data_Buffer:       .BLKW #80
  KB_Buff_head:         .BLKW #1
```