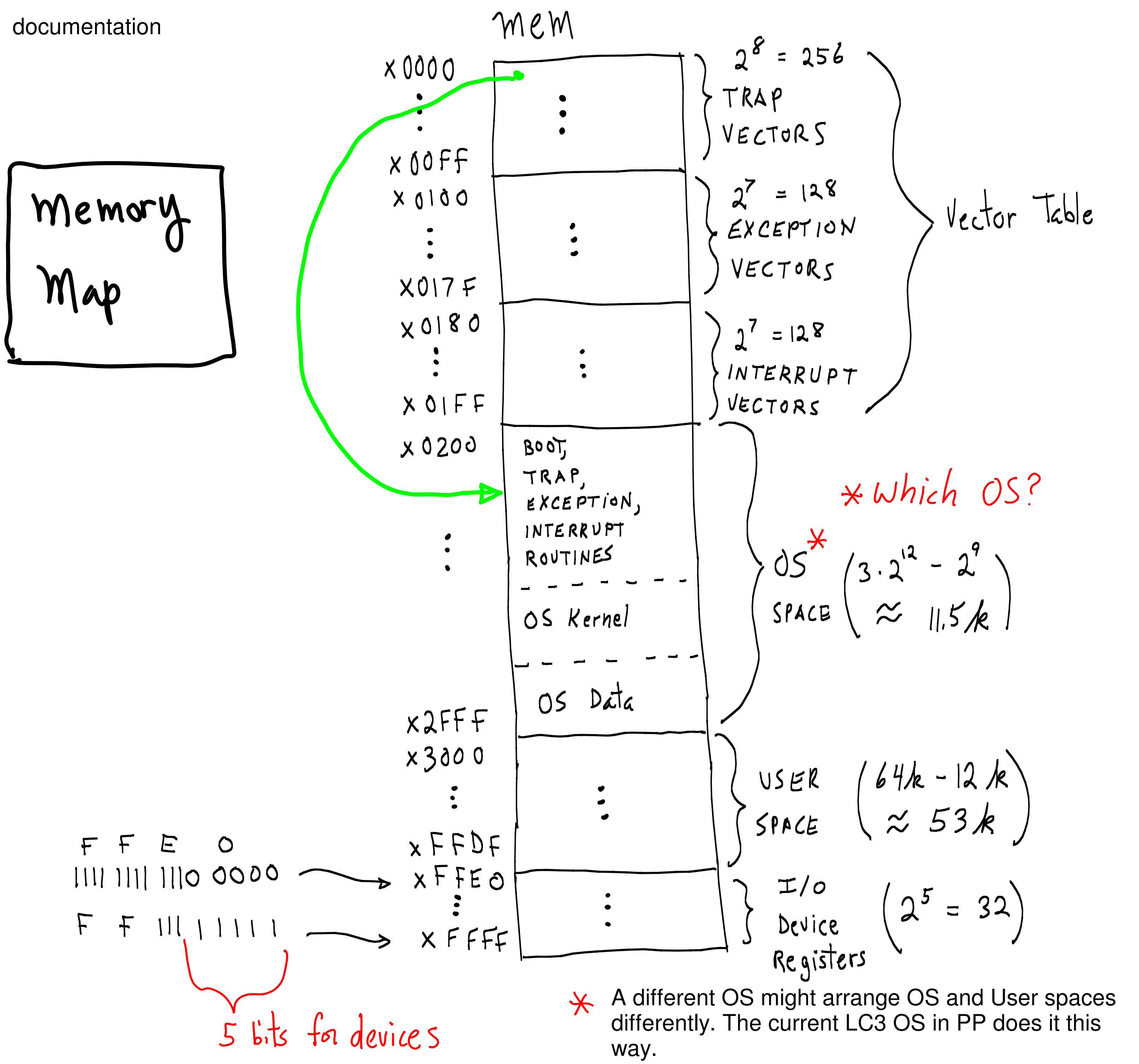
LC3-uArch-MemMapIOstacksINT

documentation

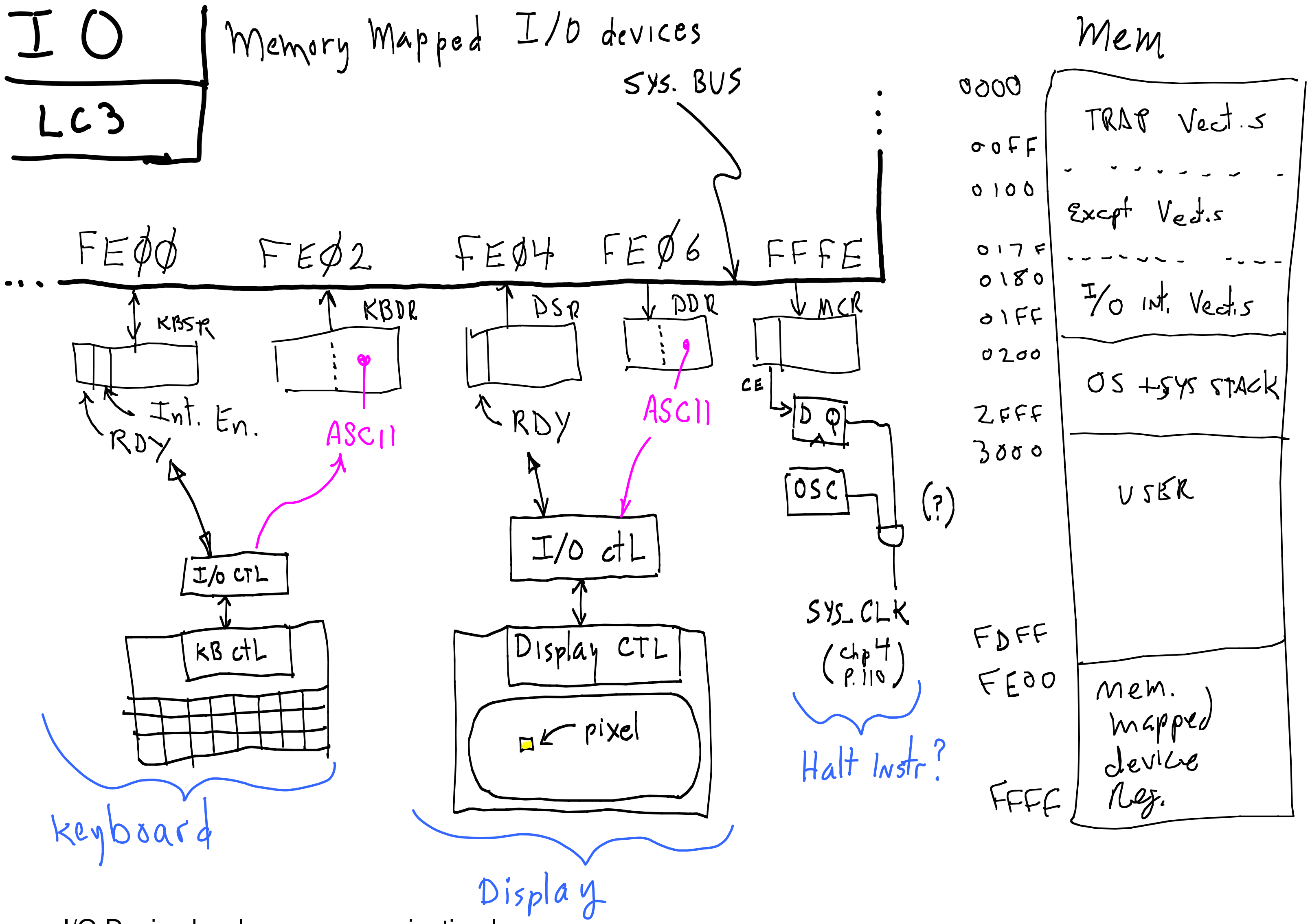

Memory Map
mem
x0000
x00FF
x0100
x017F
x0180
x01FF
x0200
x2FFF
x3000
xFFDF
xFE00
xFFFF
$2^8 = 256$ TRAP VECTORS
$2^7 = 128$ EXCEPTION VECTORS
$2^7 = 128$ INTERRUPT VECTORS
Vector Table
BOOT, TRAP, EXCEPTION, INTERRUPT ROUTINES
OS Kernel
OS Data
OS* SPACE $(3 \cdot 2^{12} - 2^9 \approx 11.5k)$
*which OS?
USER SPACE $(64k - 12k \approx 53k)$
I/O Device Registers $(2^5 = 32)$
F F E 0
1111 1111 1110 0000
F F 111 11111
5 bits for devices

* A different OS might arrange OS and User spaces differently. The current LC3 OS in PP does it this way.

OS Trap routines defined in PP's OS. (You could add to these, if you wanted to rewrite the OS.)

TRAP <n>	Assembly-psuedonymn	Description
TRAP x25	HALT	jump to OS w/ message, loops in OS forever.
TRAP x20	GETC	one char in, keyboard data ==> R0[7:0] (clears R0 first).
TRAP x21	OUT	one char out, R0[7:0] ==> display, ignores big-end byte, R0[15:8].
TRAP x22	PUTS	string out, Mem[R0++] ==> display until x0000. Ignore big-end bytes, 1 char per word.
TRAP x23	IN	displays prompt, then one char in ala GETC.
TRAP x24	PUTSP	same as PUTS, but packed (2 chars per word, little-end byte then big-end byte).

See PP, Append. A.4, Table A.2



I/O Device hardware communication layers

(0.) device's onboard controller

communicates with

(1.) device's bus interface board (aka "device controller"),

communicates with

(2.) I/O bus (PCI bus, for example).

communicates with

(3.) CPU's memory/bus unit

Device controllers are "programmed" by sending control words to their "device registers". Device status information is returned, and data is sent/received to/from the device via the device data registers. Device registers are accessed via memory LD/ST instructions using memory addresses (memory-mapped I/O).

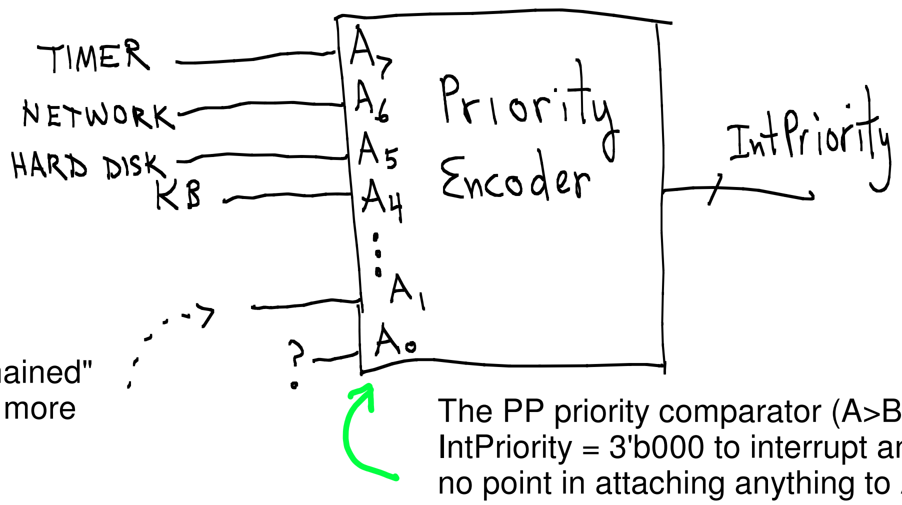
LC3 FSM
READ states:

33, 28, 24, 25, 29, 36, 40, 52*

LC3 FSM
WRITE states:
16, 41*, 48*

* Interrupt mechanism not implemented in LC3simulate.exe

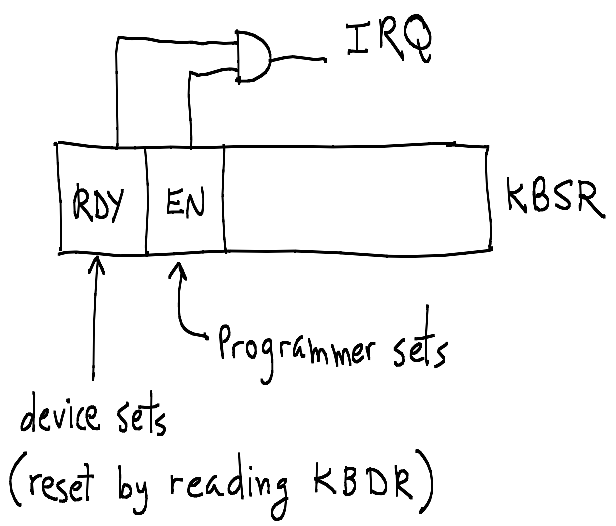
I/O Devices

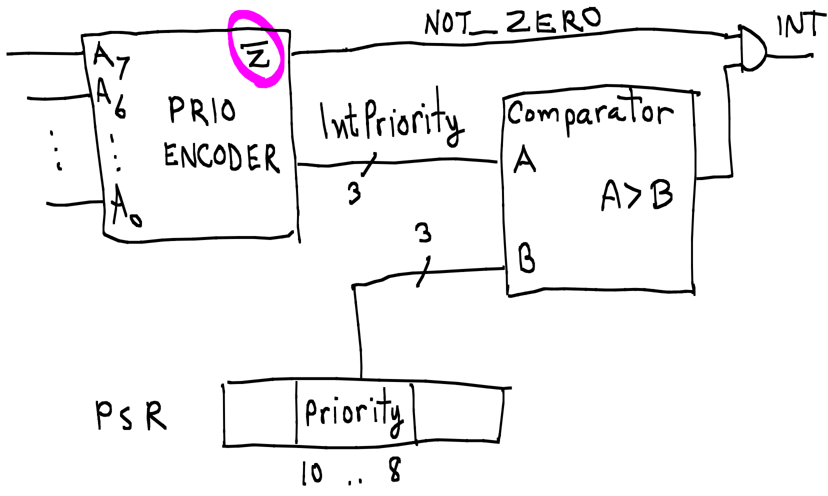


Could add another "chained" priority encoder and 7 more I/O devices.

The PP priority comparator (A>B) does not allow IntPriority = 3'b000 to interrupt any running program. So, no point in attaching anything to A_0. We could change to (A>=B). In that case, we cannot disable interrupts by setting PSR.Priority = 3'b111. We could handle that by adding an EnableINT bit to the MCR, but then we would need an instruction that can toggle that bit.

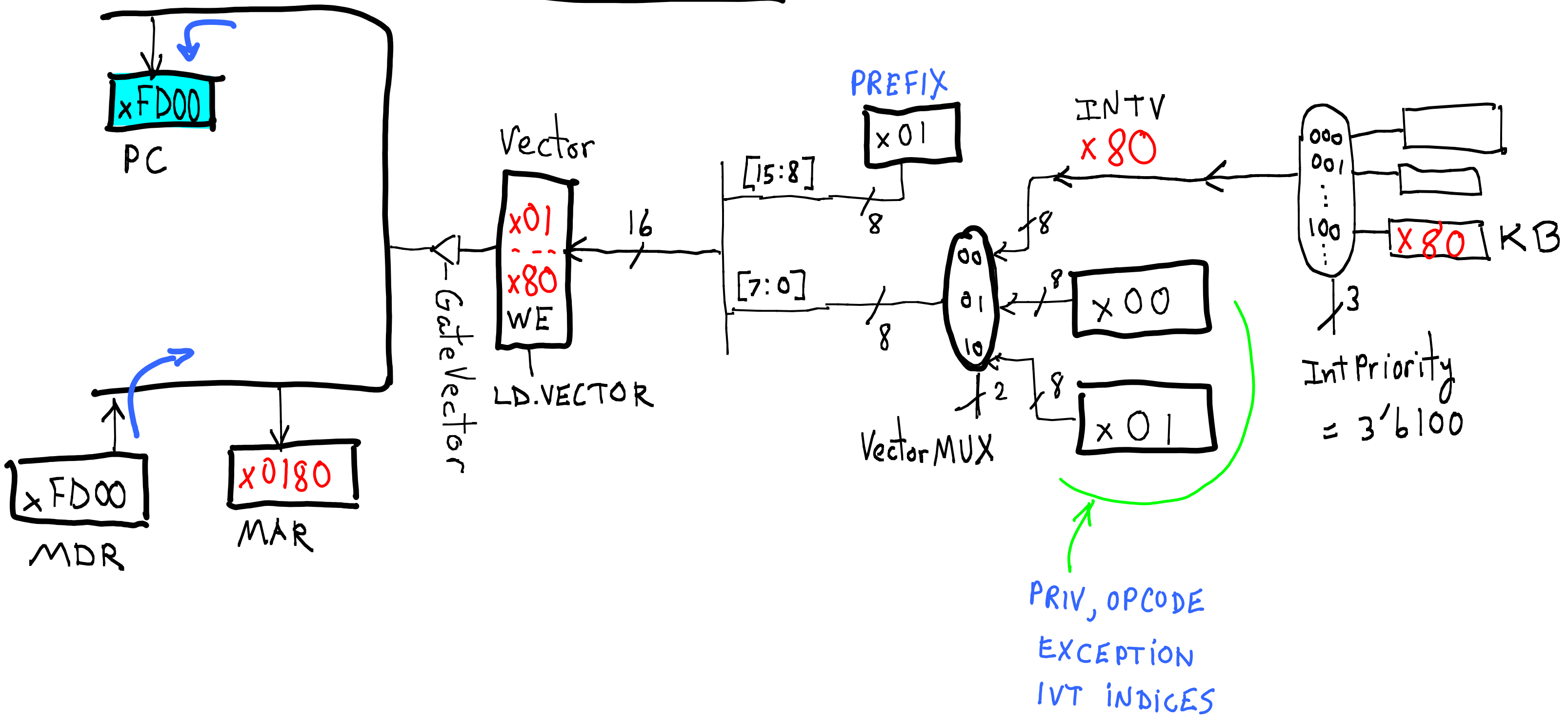
KB Ints

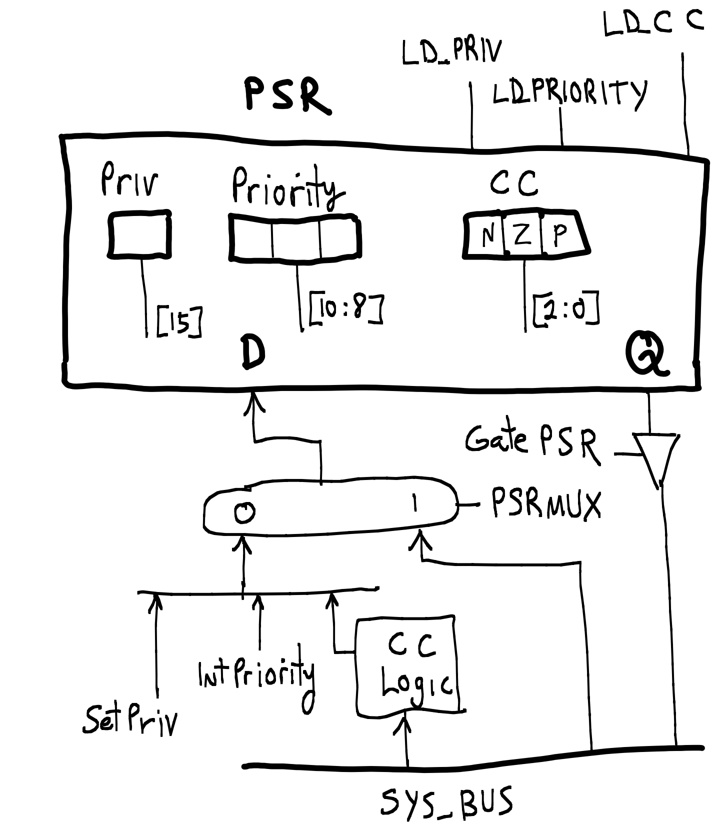




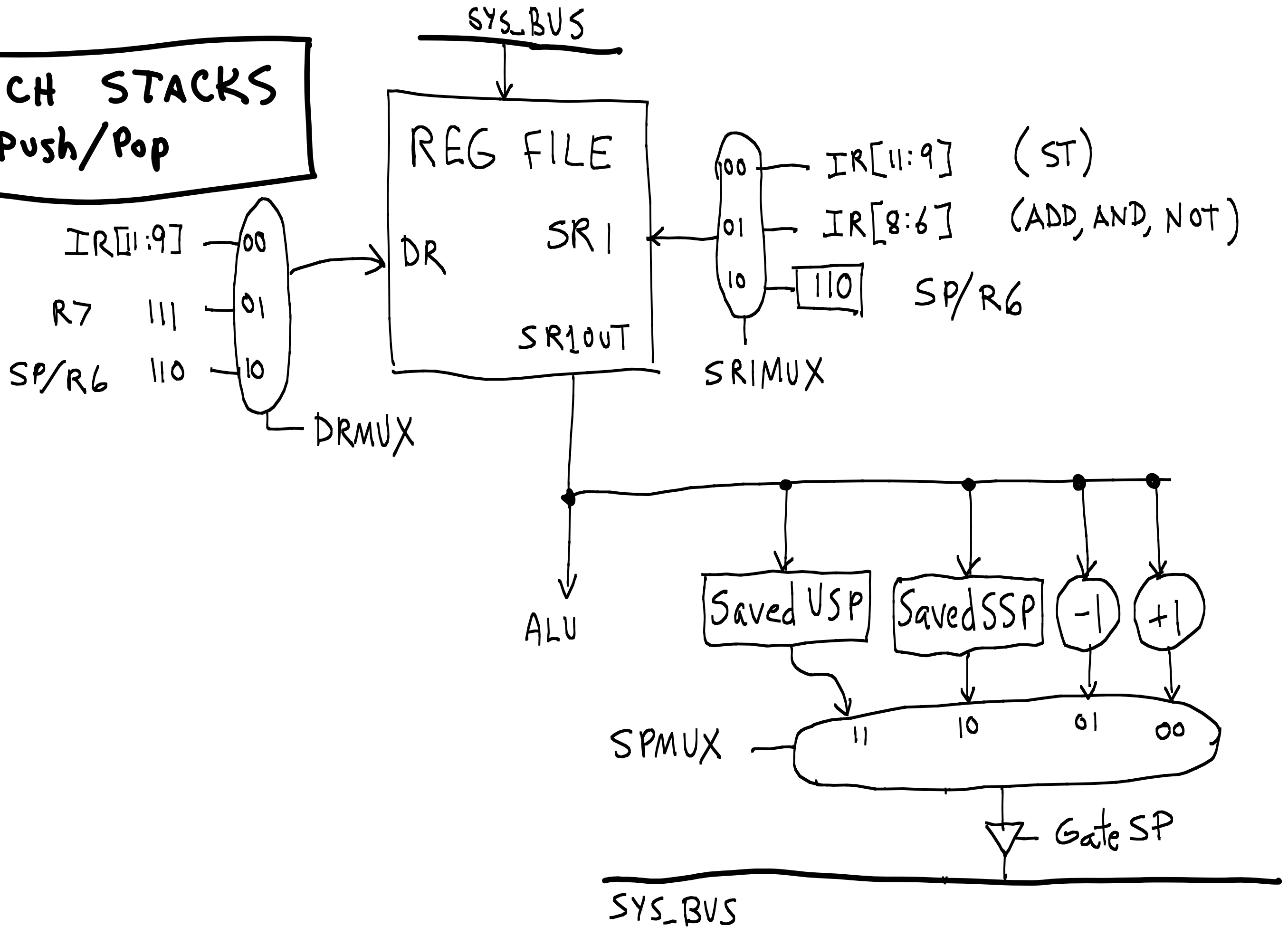
Currently executing program has priority level in PSR[10:8].
Priority encoder sends code of highest priority device requesting service.

KB interrupt





SWITCH STACKS
HW Push/Pop



SYS_BUS
REG FILE
DR
SR1
SR1OUT
IR[11:9]
00
R7
111
01
SP/R6
110
10
DRMUX
IR[11:9]
(ST)
IR[8:6]
(ADD, AND, NOT)
110
SP/R6
SR1MUX
ALU
Saved USP
Saved SSP
-1
+1
SPMUX
11
10
01
00
Gate SP
SYS_BUS