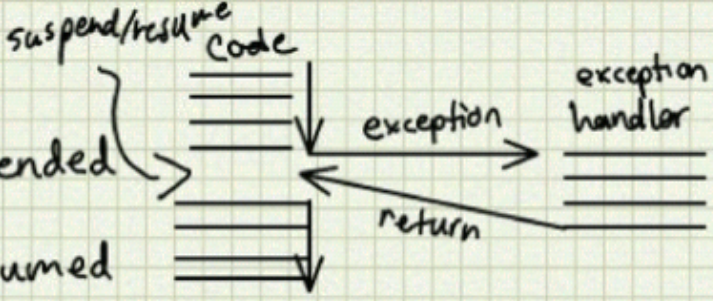
5) Exceptions



- condition requiring service
  - current execution is suspended
  - condition is serviced
  - original execution is resumed
- code is unaware that it was suspended
- Cortex-M3 exception types
  - reset (power on)
  - SVCall (supervisor call - invokes OS)
  - fault (execution error like invalid instructions, unaligned memory access)
  - interrupt (request from a peripheral device for service)
  - request (IRQ)
- exception attributes
  - number 1-255
  - priority -3 (highest) to 31 (lowest)
  - vector exception handler address
    - ↳ entry in vector table
- exception list

name	number	priority
Reset	1	-3
NMI	2	-2
HardFault	3	-1
MemManage	4	configurable
BusFault	5	↓
...	...	
SVCall	11	
...	...	
IRQ 0	16	
...	...	
IRQ 239	255	

- NMI = non-maskable interrupt
  - connected to reset button or watchdog timer (fail safe for embedded systems)
- vector table : entry points of exception handlers

			vector #
0000 0000	SP_main		0
0000 0004	Reset handler	entry point	1
0000 0008	NMI handler	entry point	2
	...		

5.1) Operating Modes

mode	used for...	can exec. privileged instr.	stack
thread	application code	no* or yes**	SP_process* or SP_main**
handler	exception handler	yes	SP_main

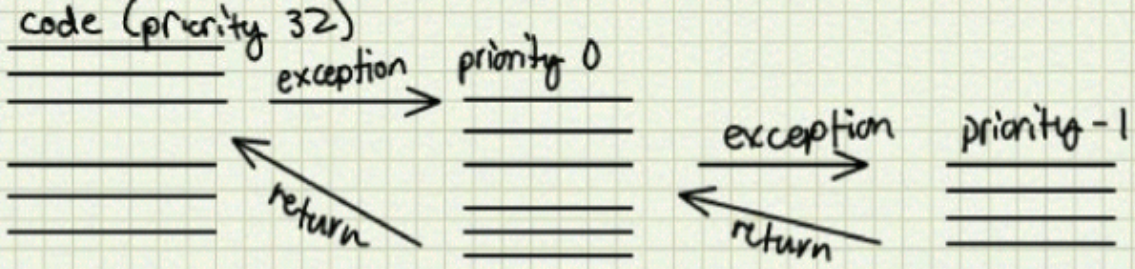
* multi-user OS - Linux, Windows, Mac etc. } determined by CONTROL register
** single-user OS

- hw switches to handler mode on exception and back to thread mode on return
- privileged instructions control system state
  e.g. CPSIE i // interrupt enable
- seperate user stacks (SP_process) and OS stack (SP_main) avoid corruption of system stack by user code

5.2) Reset
- an exception that happens on power up or warm reset
- actions taken:
  1. loads SP_main into r13 from vector 0
  2. sets operating priority to max priority +1 = 32
  3. executes code in the reset handler

5.3) Exception Handling

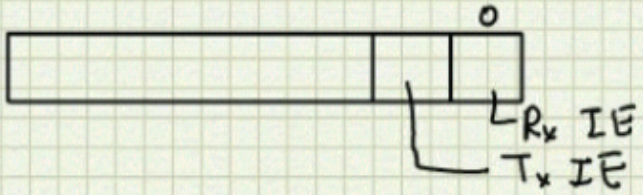
- exceptions are handled (handler is invoked) if they have higher priority (lower #) than the operating priority
- actions taken:
  1. save current context on stack
     - pushes r0-r3, r12, return address (PC), PSR (status reg), LR
  2. stores EXC_RETURN in LR indicating operating mode and stack in use
  3. loads entry point of handler from vector table into PC
- exception return is initiated by loading the EXC_RETURN code into the PC e.g. BX lr
- actions taken:
  1. uses EXC_RETURN to set operating mode and stack pointer
  2. restores context from stack
     pops r0-r3, r12, PC, PSR, LR
- exception handlers can be suspended by higher priority exceptions (nested exceptions)



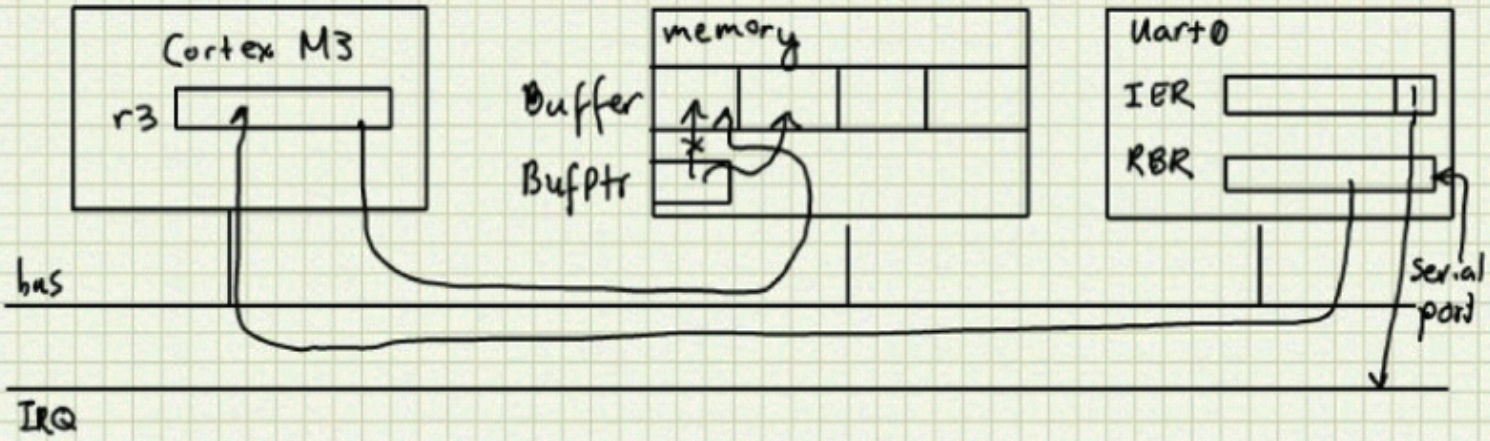
5.4) Interrupt-Driven I/O

- Set-up
  - assign exception number and priority to I/O device
  - write handler (Interrupt Service Routine) and save entry point in the vector table
  - enable interrupts on I/O device
  - after that execute application code

e.g. use interrupts to get data from UART0 on LPC1768
IER[9:0] interrupt enable register

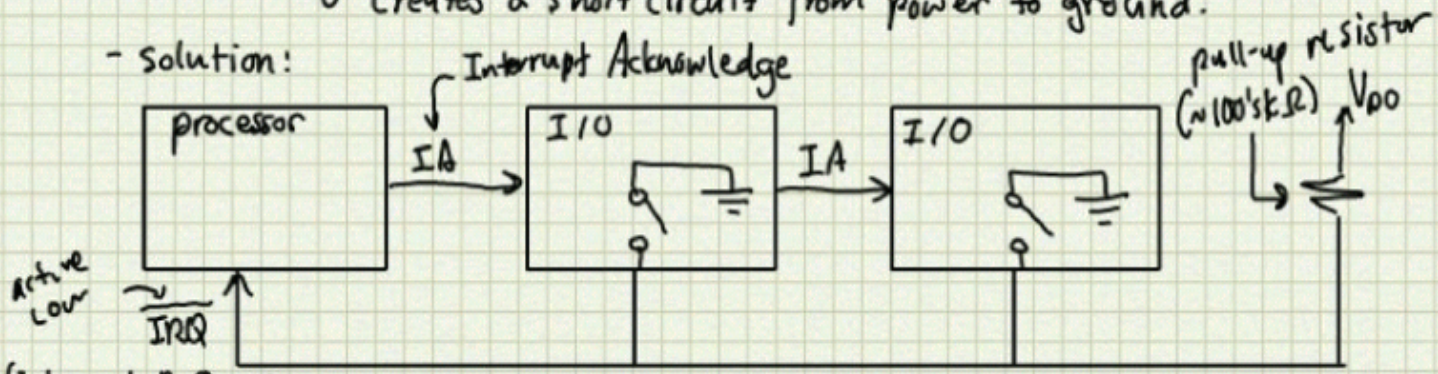


- U0IER at 0x 4000C004



5.5) Open Drain IRQ Lines

- If I/O devices outnumber IRQ lines, they must be shared
- problem: can't connect CMOS outputs because driving a '1' and a '0' creates a short circuit from power to ground.
- solution:



- $\overline{IRQ}$ floats high (1)
- to signal interrupt, I/O device closes connection to ground, pulling $\overline{IRQ}$ low (0)
- 1 = no interrupt, 0 = interrupt ∴ active low
- How does processor identify correct I/O device
  1) can check (poll) each status reg. (slow)
  2) daisy-chain Interrupt Acknowledge (IA) (fast)
     - I/O device only passes on IA, if not interrupting
     - I/O device that receives IA = 1 and needs service outputs its vector number on databus (auto-vectoring)
     - processor looks up ISR in vector table
     - two levels of priority
       - between $\overline{IRQ}$ lines
       - between I/O devices sharing an $\overline{IRQ}$ line (closer device to processor has higher priority