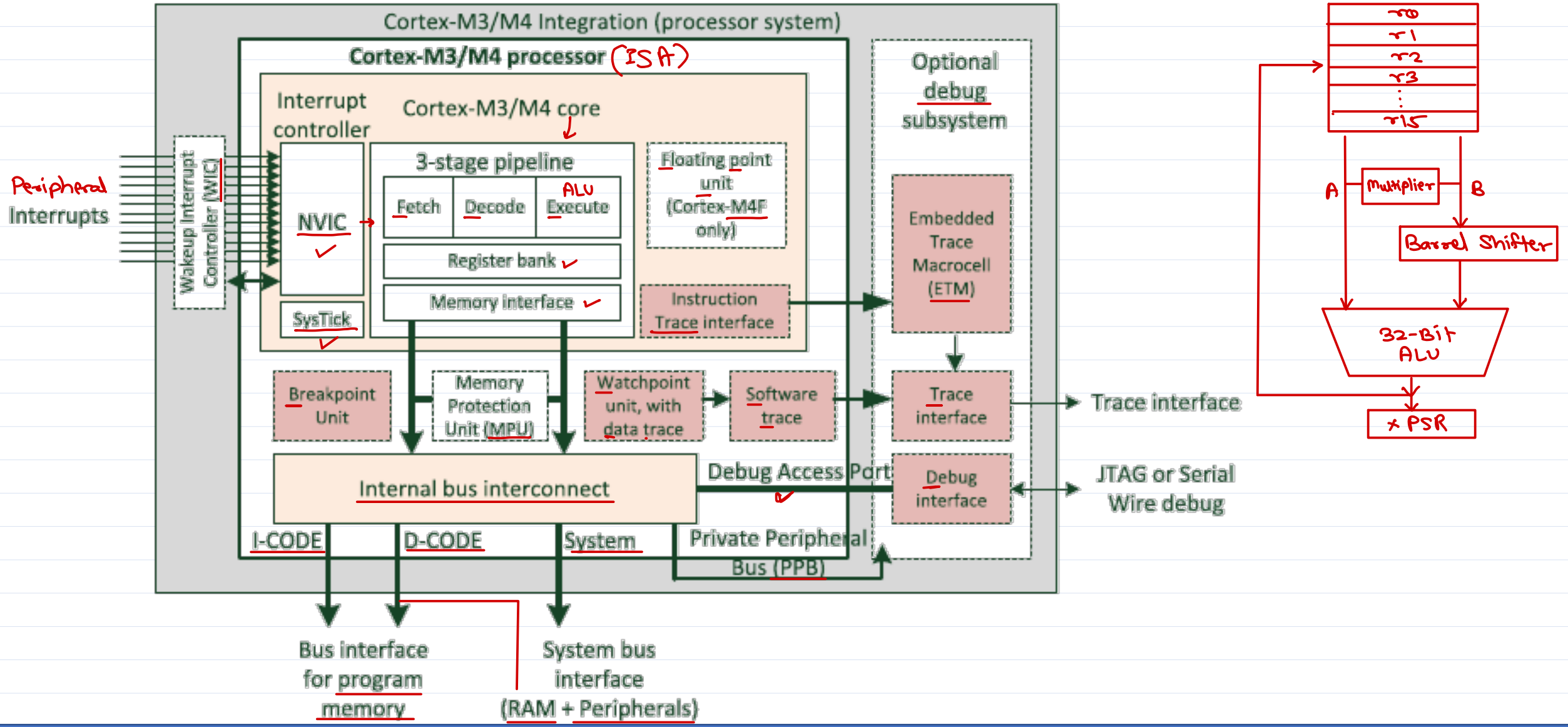
ARM Cortex-M architecture

DESD @ Sunbeam Infotech

Agenda

- CM3 states
- CM3 modes
- CM3 access levels
- CM3 programmer's model
  - Registers
  - Status register
  - Mask registers
- CM3 exceptions, faults & interrupts
- NVIC & its registers
- Exception sequence & exit
- Tail chaining & Late arrival
- EXC_RETURN
- Exception/interrupt priority

CM3 Architecture



Barrel shifter

LSL : Logical Left Shift

Multiplication by a power of 2

(num<<n)

ASR: Arithmetic Right Shift

Division by a power of 2, preserving the sign bit

(signed)num >> n

LSR : Logical Shift Right

Division by a power of 2

(unsigned)num >> n

ROR: Rotate Right

Bit rotate with wrap around from LSB to MSB

RRX: Rotate Right Extended

Single bit rotate with wrap around from CF to MSB

Barrel shifter is combinational cct that does shift operations.

- Barrel shifter instructions

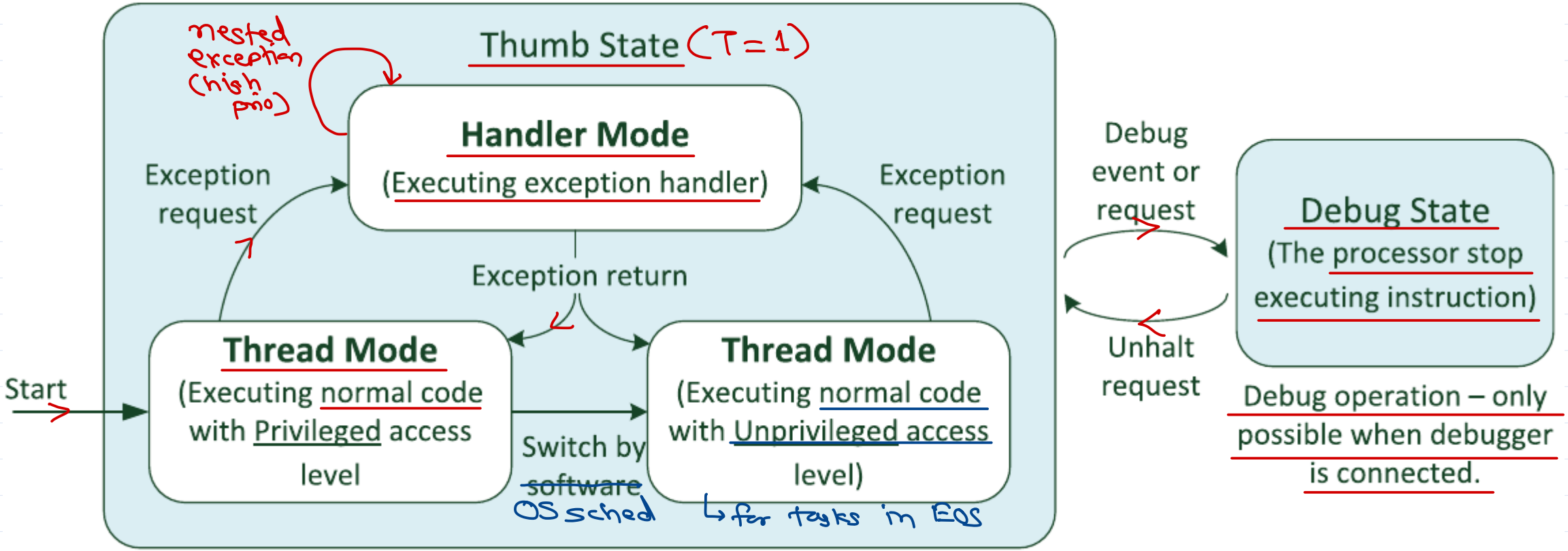
```
LSL r0, #2
r0 = r0 << 2
```

ASR
LSR
ROR
RRX

- Inline barrel shifter
  - mov rd, rs, lsl #k
    rd = rs << k
  - mov rd, rs, lsr #k
    rd = rs >> k
  - add r0, r1, r2, lsl #k
    r0 = r1 + (r2 << k)

Supported in ARM instn set.

CM3 States, Modes & Access levels



Bare metal programming

Privileged Level			Mode
①	main code + reset handler	→	Thread
②	handler code	→	Handler

Embedded OS based programming

Privileged –	① Handler code	→	Handler
Privileged –	② main code + Reset Handler + Kernel	→	Thread
Non-Privileged –	③ User tasks	→	Thread

ARM CM3 states

- ARM7 states
  - ARM state
  - Thumb state
- CM3 states
  - Thumb state
  - Debug state

- Thumb state
  - User program, embedded OS and handler execution state.
  - CM3 supports Thumb-2 instruction set.
  - No ARM state & hence no state switching overheads.
- Debug state
  - When program execution is halted due to debugger request.

ARM CM3 modes

- ARM7 modes
  - Privileged modes
    - Supervisor
    - IRQ
    - FIQ
    - Undef
    - Abort
    - System
  - Non-privileged modes
    - User
- CM3 modes
  - Thread mode
  - Handler mode

- Thread mode
  - User program or embedded OS execution mode.
  - Privileged or Non-privileged access level.
    - Privileged level – Embedded OS
    - Non-privileges level – User program /tasks
  - For bare metal programming, only privileged access level is used.
- Handler mode
  - When exception/interrupt occurs, core enters in handler mode.
  - Exception handler is executed in this mode.
  - After handler completion, it returns to thread mode.
  - Privileged access level.

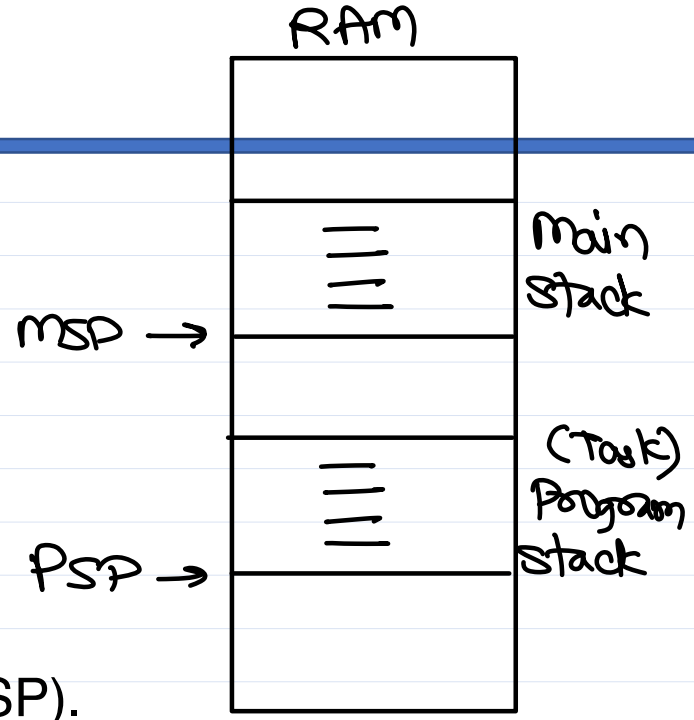
ARM CM3 access levels

- CM3 can be used with embedded OS environment or without it (bare metal).
- For bare metal development
  - Only privileged access level.
- For OS based development
  - User tasks execute in non-privileged level
  - OS & handlers run in privileged level
- Access level is controlled by
  - CONTROL register – bit[0]
    - 0 = privileged level
    - 1 = Non-privileged level
- Privileged level
  - All resources are accessible and all instructions are allowed.
- Non-Privileged level
  - Few resources are not accessible and few instructions are not allowed.
  - Few memory regions are not accessible.
  - NVIC access is not allowed.
  - Cannot access interrupt mask registers, CONTROL register.

ARM CM3 Registers

R0	
R1	
R2	
R3	
R4	
R5	
R6	
R7	
R8	
R9	
R10	
R11	
R12	
R13 (MSP)	R13 (PSP)
R14 (LR)	
R15 (PC)	

- General purpose register
- SP – Stack pointer – Only banked register
  - Different stack for different access levels.
  - Controlled via CONTROL register bit[1].
    - 0 – MSP Main SP
    - 1 – PSP Program SP
  - Single stack for bare-metal programming (MSP).
  - Different stacks for OS (MSP) & user programs (PSP).
  - Stack used for local variables and context saving in case of function calls and exceptions/interrupts.
- LR – Link register
  - Stores return address for function call
  - Stores special value EXC_RETURN in case of exceptions
- PC – Program counter
  - Keep address of next instruction to be fetched.



ARM CM3 Registers

- APSR
- EPSR
- IPSR

xPSR	N	Z	C	V	Q	ICI/IT	T=1		GE*	ICI/IT		Exception Number
	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8:0
APSR	N	Z	C	V	Q				GE*			
IPSR												Exception Number
EPSR						ICI/IT	T			ICI/IT		

- xPSR
- PRIMASK
- FAULTMASK
- BASEPRI
- CONTROL

	31:8	7:1	0
PRIMASK			
FAULTMASK			
BASEPRI			

3 bits to 8 bits

0 bit to 5 bits

	31:3	2	1	0
CONTROL			SPSEL	nPRIV

- Status register(s)
  - Application, Execution & Interrupt status.
- Mask registers
  - Disable interrupts, exceptions & faults
- Control register
  - Handle access level and stack.

ARM CM3 Registers

- xPSR
  - Provide ALU flags, exception status and current interrupt number
  - xPSR is not directly accessible to program. Its individual parts can be accessed as follows.
    - APSR: ALU flags (NZCVQ)
    - EPSR: Execution flags (IT bits + T=1)
    - IPSR: Current interrupt number

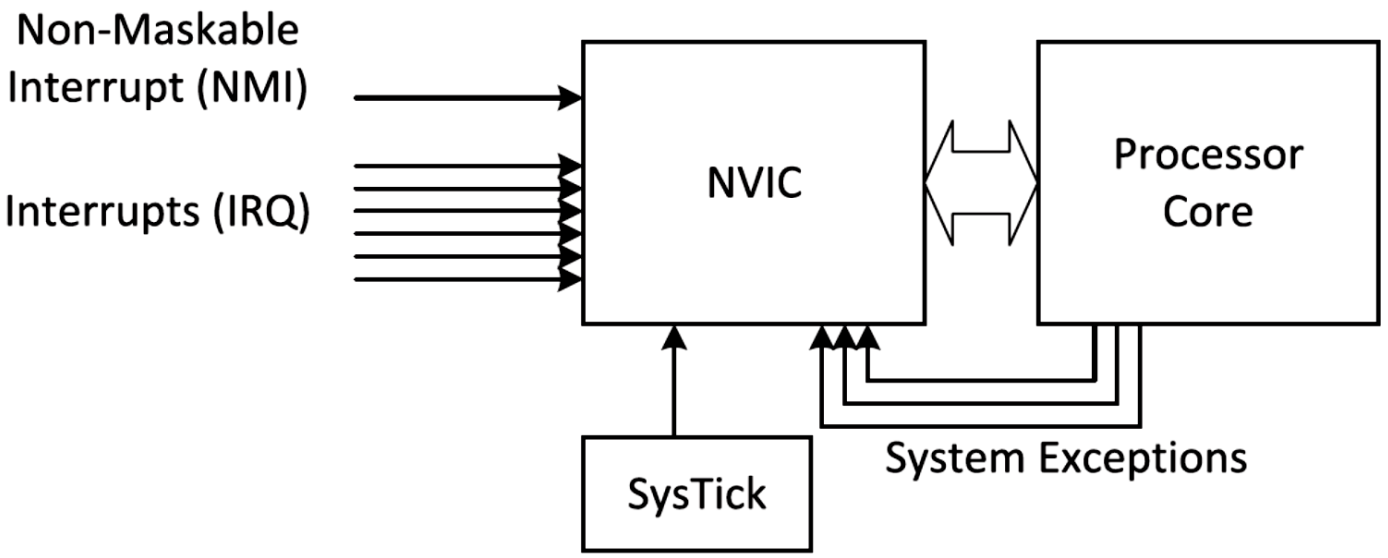
Special instructions for status register

MSR ← write status regr

MRS ← read status regr.

- PRIMASK
  - Disable all interrupts except NMI & HardFault & faults
- FAULTMASK
  - Disable all interrupts except NMI
- BASEPRI
  - Disable all interrupts of specific priority level and below
- CONTROL
  - Handles privilege level and stack pointer selection

ARM CM3 Exceptions & Interrupts



- In ARM architecture, interrupts are special kind of exception.
- CM3 exception handling model is far different than ARM7.
- CM3 have single vector table for exceptions and interrupts.
  - Entry 0: MSP initial address
  - Entries 1-15: Exceptions
  - Entries 16-255: Peripheral interrupts (depending on chip)

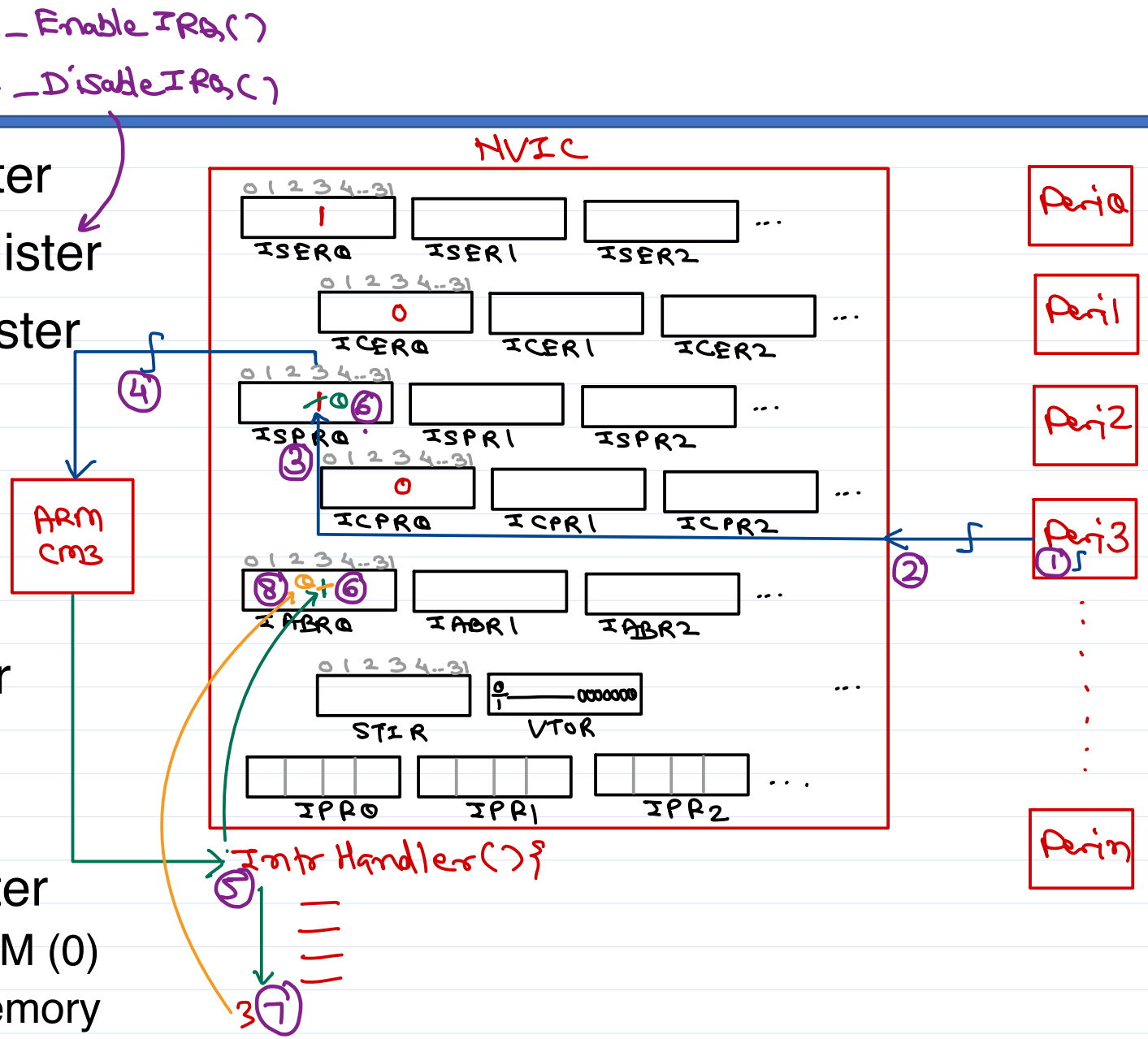
STM32 → 82
LPC1768 → 35

exception/interrupt number

	Exception	Priority	CMSIS No
0	Initial MSP		
1	Reset	-3	
2	NMI	-2	-14
3	HardFault	-1	-13
4	MemManage	Program	-12
5	BusFault	Program	-11
6	UsageFault	Program	-10
7	reserved		
8	reserved		
9	reserved		
10	reserved		
11	SVC	Program	-5
12	DebugMonitor	Program	-4
13	reserved		
14	PendSV	Program	-2
15	SysTick	Program	-1
16	Peripheral 0	Program	0
17	Peripheral 1	Program	1
...	...	...	...
255	Peripheral 239	Program	239

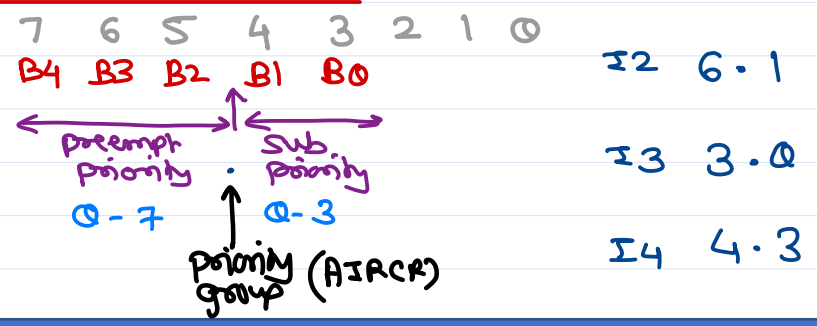
NVIC Registers

- ISER – Interrupt Set Enable Register
- ICER – Interrupt Clear Enable Register
- ISPR – Interrupt Set Pending Register
- ICPR – Interrupt Clear Pending Register
- STIR – Software Trigger Interrupt Register
- IABR – Interrupt Active Bit Register
- IPR – Interrupt priority (5 bits per interrupt)
- VTOR – Vector Table Offset Register
  - bit 29 – Table location RAM (1) or ROM (0)
  - bit 28:7 – Table offset from start of memory

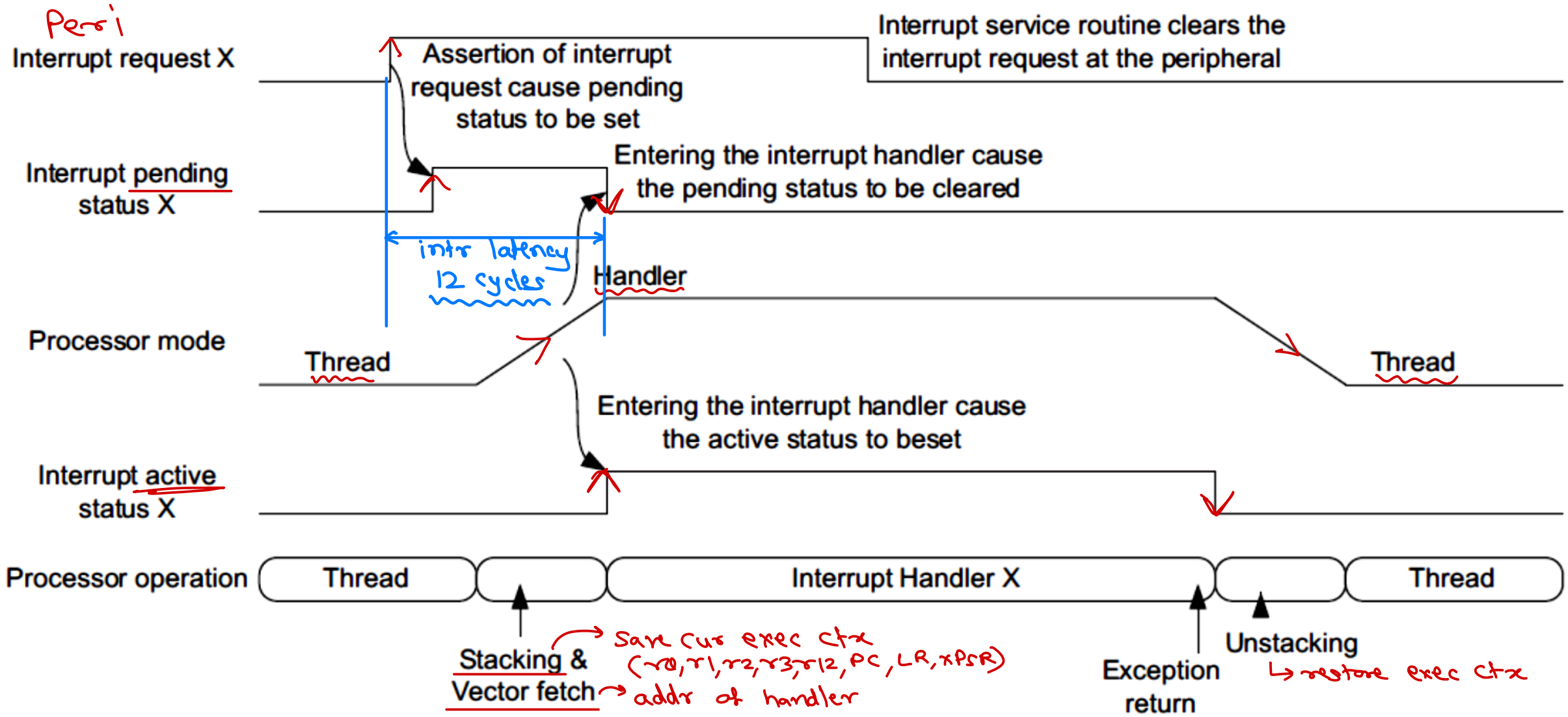


Interrupt priority

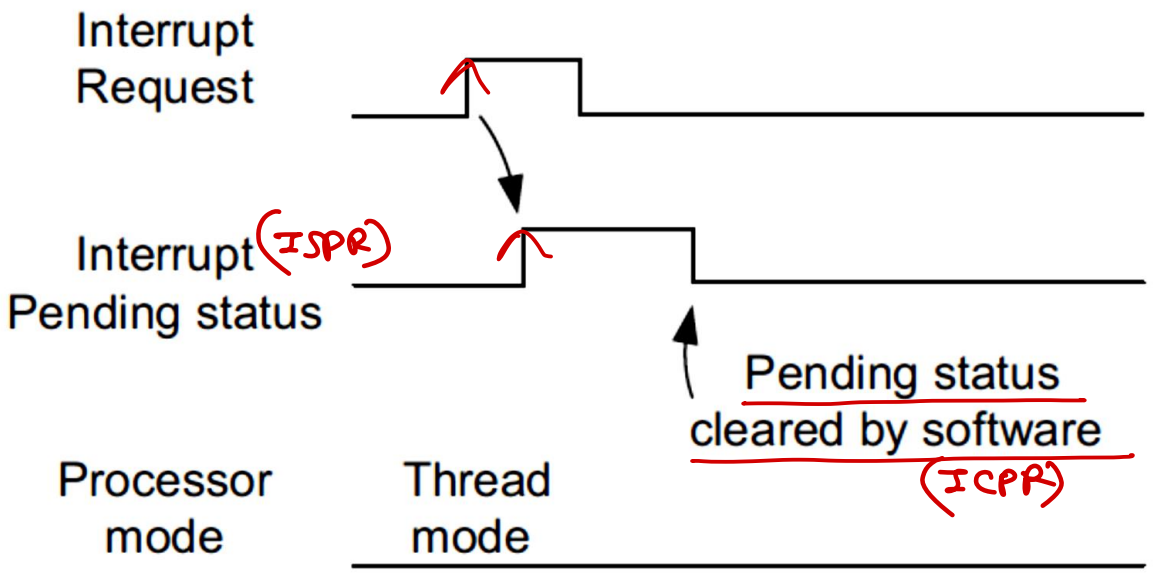
- Three fixed priority levels i.e. -3 (highest), -2 & -1.
- Up to 256 levels of programmable priority (if 8 bits are used per interrupt).
- More the priority bits increase gate count and hence power consumption.
- Most of chips use 5 bits priority per interrupt (upper 5 bits i.e. [7:3]) in IPR. e.g. LPC17xx
- It enable 32 priority levels.
- Priority levels are divided into two parts
  - ***Pre-emption priority***: Higher pre-emption priority pre-empts current handler.
  - ***Sub priority***: Interrupt handlers are executed in this order, if multiple are pending (of same pre-emption priority).
- Number of bits in pre-emption priority and sub-priority is adjustable via Priority Group setting in NVIC register AIRCR [bits 10:8].
  - PRIGROUP=2 → all 5 bits [7:3] for pre-emption priority.
  - PRIGROUP=4 → 3 bits [7:5] for pre-emption priority and 2 bits [4:3] for sub priority.
  - PRIGROUP=7 → all 5 bits [7:3] for sub priority.
- Refer section 34.4.3.6.1 from LPC1768 user manual.



ARM CM3 Interrupt input & Pending behaviour

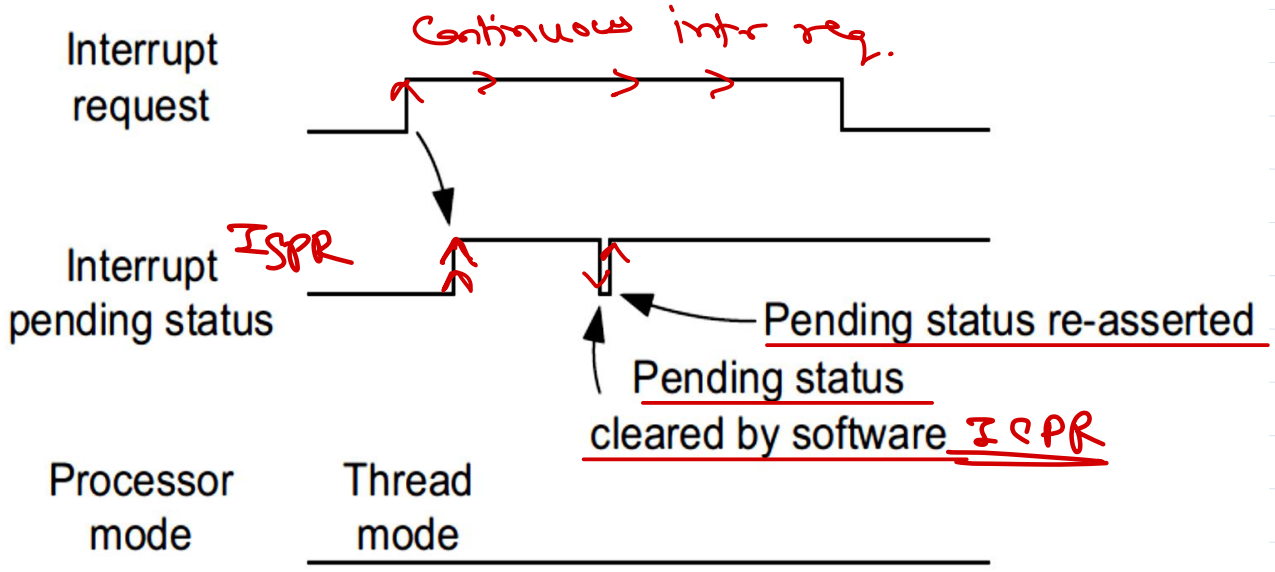


ARM CM3 Interrupt input & Pending behaviour



- Interrupt pending status cleared before processor serving interrupt
- Status cleared by software using ICPR

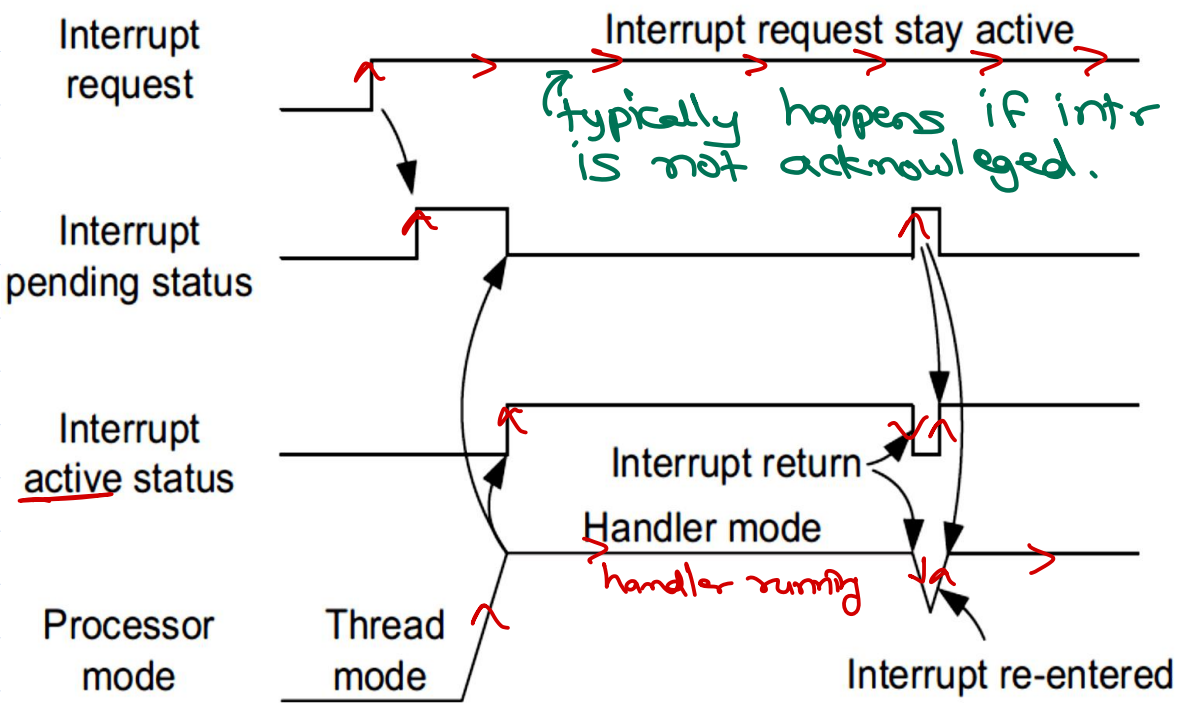
Handler is not executed.



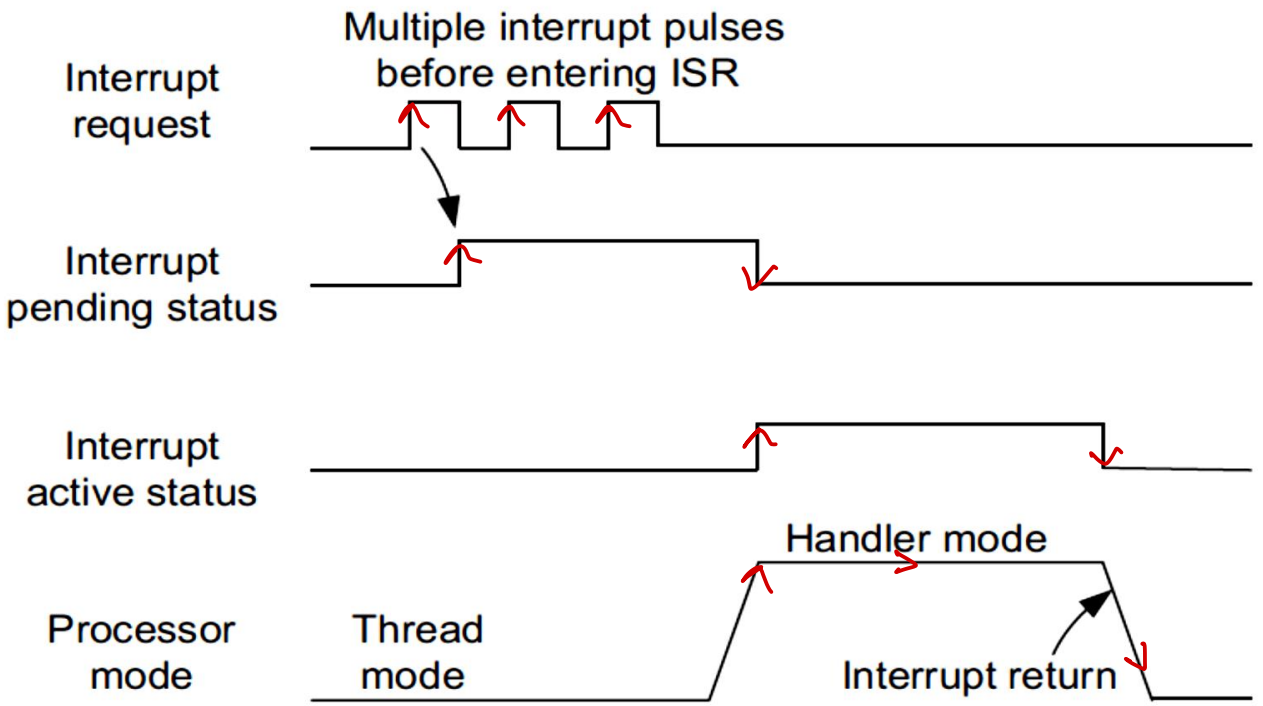
- Interrupt pending status cleared and re-asserted due to continuous interrupt request

Handler will be executed later (when intr is seen by CPU).

ARM CM3 Interrupt input & Pending behaviour



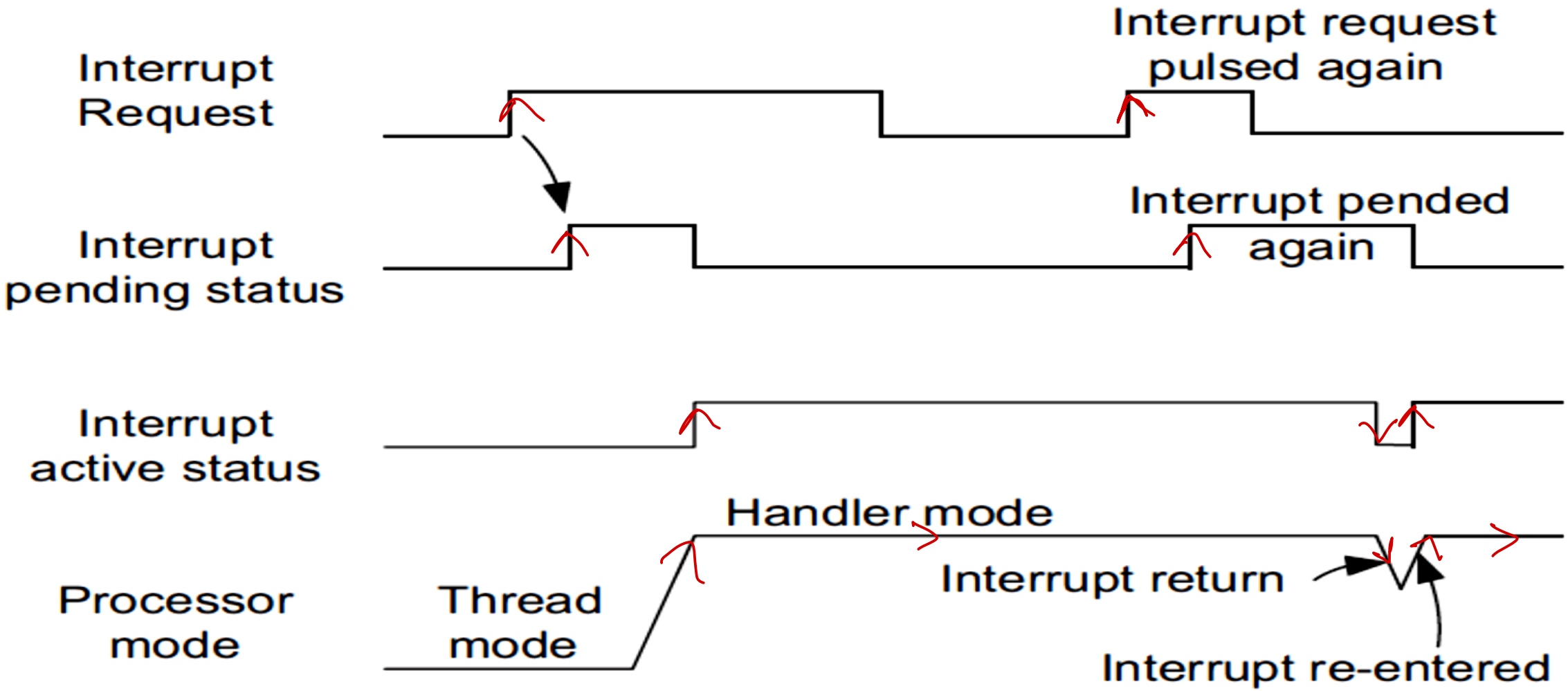
- Interrupt gets its pending status set again if the request is still asserted after exception exit



- Interrupt pending status set only once with several interrupt request pulses

In this case, handler will execute only once.

ARM CM3 Interrupt input & Pending behaviour



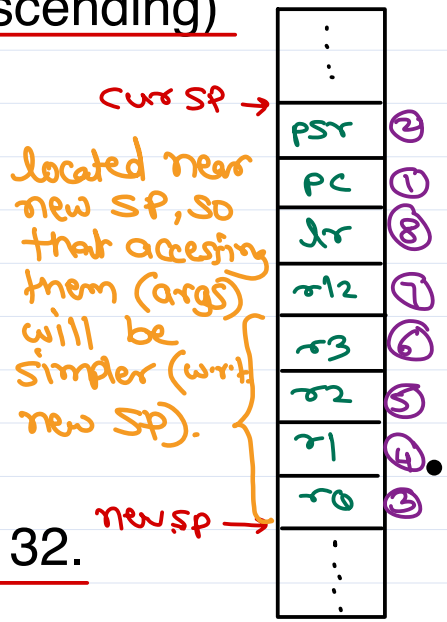
- Interrupt pending occurs again during the execution of ISR

Exception/Interrupt sequence

- When exception occurs, following things happen in ARM CM3
  1. Stacking
  2. Vector fetch
  3. Update SP, LR, PC & xPSR

12 cpu cycles (intr latency)

- Stacking
  - 8 registers (R0-R3, R12, PC, LR & PSR) pushed on stack (full descending)
    - PSR at SP-4 (push 2)
    - PC at SP-8 (push 1)
    - LR at SP-12 (push 8)
    - R12 at SP-16 (push 7)
    - R3 at SP-20 (push 6)
    - R2 at SP-24 (push 5)
    - R1 at SP-28 (push 4)
    - R0 at SP-32 (push 3)
  - New SP will be old SP – 32.

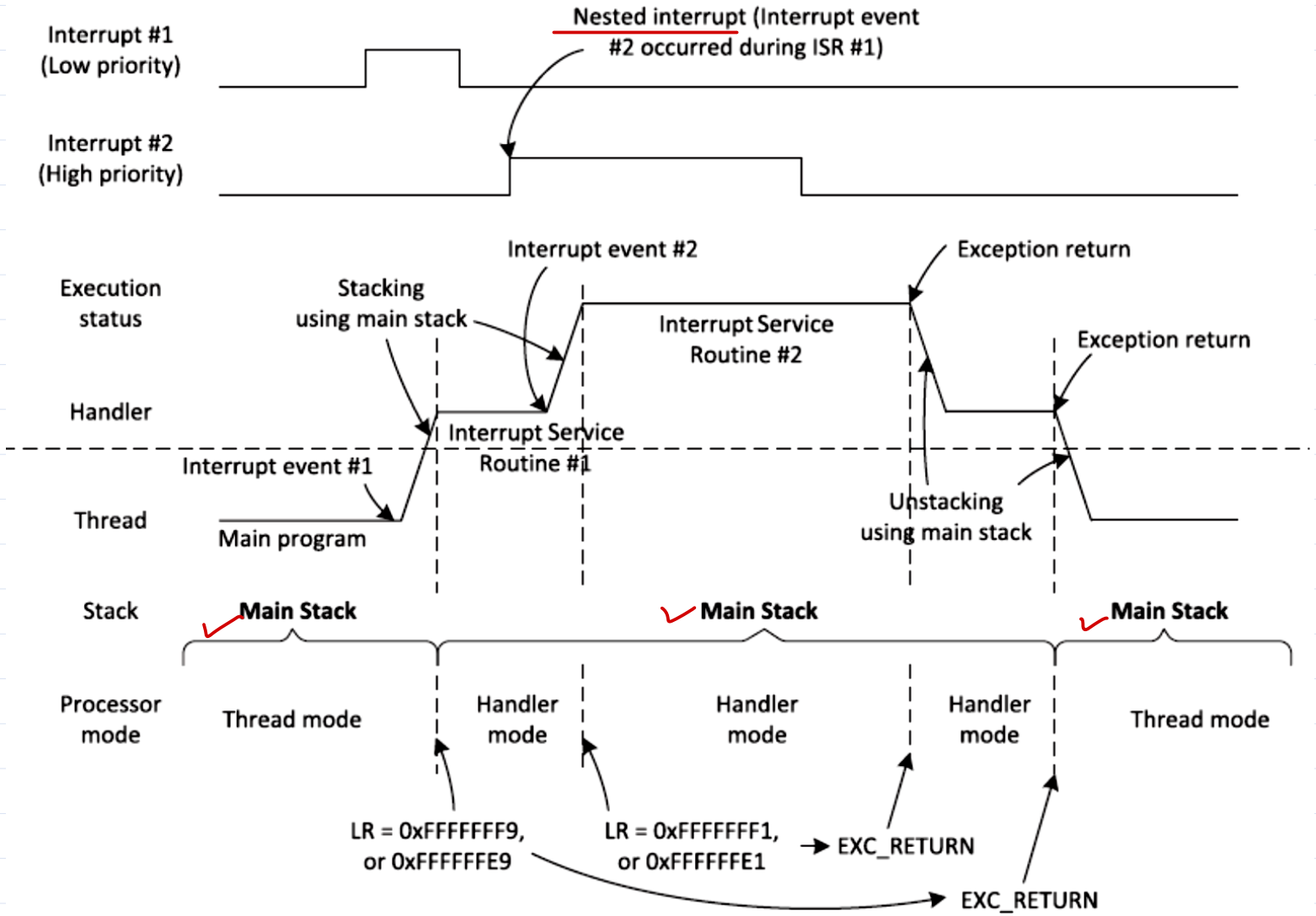


- Vector fetch
  - Fetch address handler/ISR instruction from code memory (as per vector address in exception vector table) using instruction bus.
  - Due to Harvard architecture, it is parallel to stacking (on data bus).
- Update SP, PC, LR & xPSR
  - Current **SP** is updated after stacking (8 locations = 32 bytes)
  - **xPSR** will be updated with new interrupt number
  - **PC** will load handler address (after vector fetch)
  - **LR** will be updated with special EXC_RETURN
  - NVIC registers will also be updated.
- Exception handler always runs in handler mode (irrespective of current mode).

Exception/Interrupt exits

- Upon completion of handler, system state has to be restored.
- Exception exit/return process is initiated when EXC_RETURN (in LR) value is copied into PC.
  - BX lr
  - POP {pc} – If LR is on stack.
  - LDR pc, =value – loading fixed value
- During return following steps are done
  - Unstacking: All 8 registers are restored (including SP, PC & xPSR) & LR
  - Register update: NVIC registers are updated to indicate completion of handler. (IABR)
- EXC_RETURN is special value having upper 28 bits set to 1 i.e. 0xFFFFFFFX
- Last 4 bits controls return behaviour
  - bit 0: processor state (always T=1)
  - bit 1: reserved (always 0)
  - bit 2: return stack (0 for MSP, 1 for PSP) – depend on earlier stack (usually)
  - bit 3: return mode (0 for Handler, 1 for Thread) – depend on earlier mode
- Possible values:
  - 0xFFFFFFF1: Return to handler mode
  - 0xFFFFFFF9: Return to thread mode (with MSP use after return)
  - 0xFFFFFFFD: Return to thread mode (with PSP use after return)

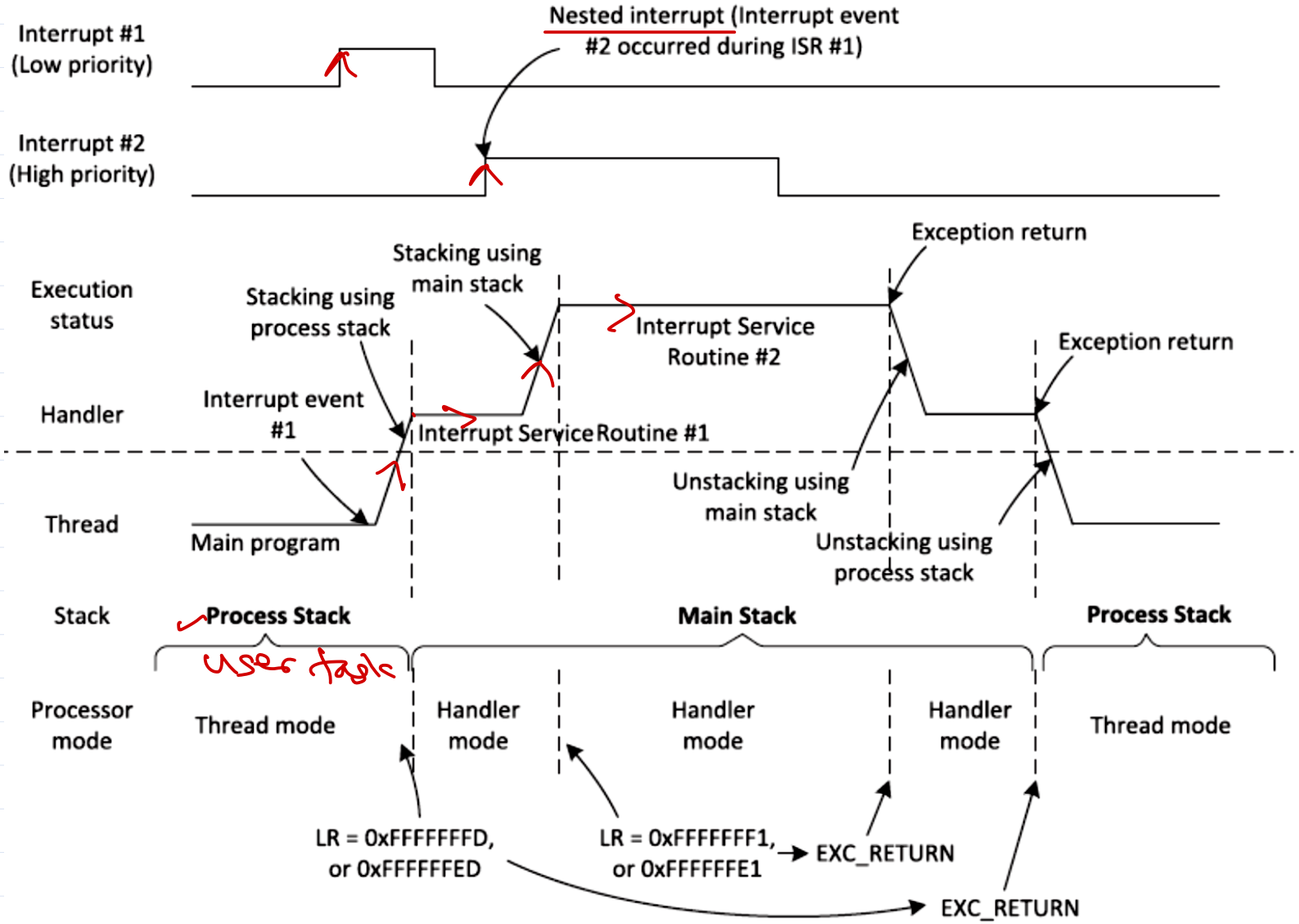
Exception/Interrupt exit – EXC_RETURN (bare metal / EOS kernel)



- LR set to EXC_RETURN at exception (Main stack used in thread mode).

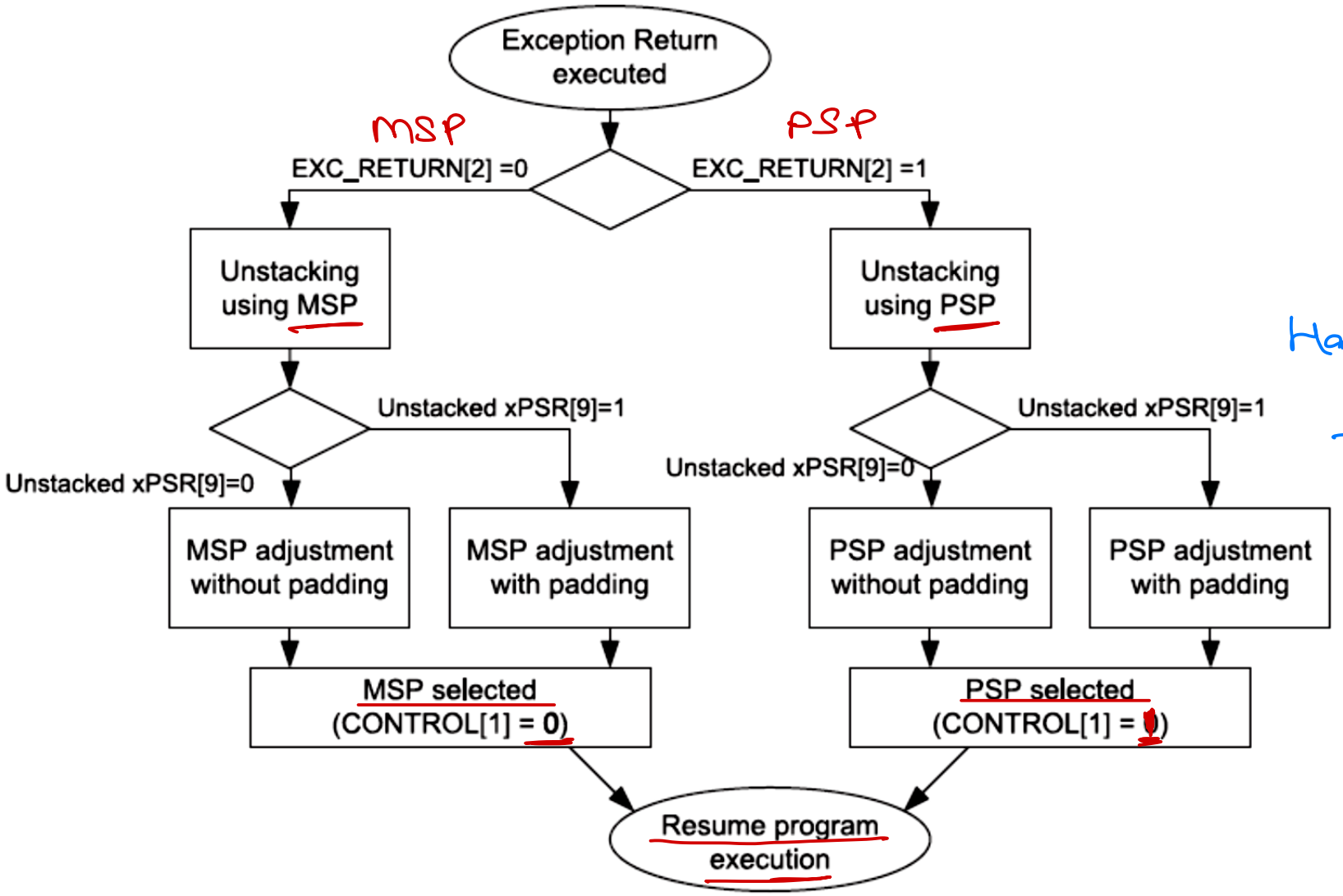
Exception/Interrupt exit – EXC_RETURN

EOS user task

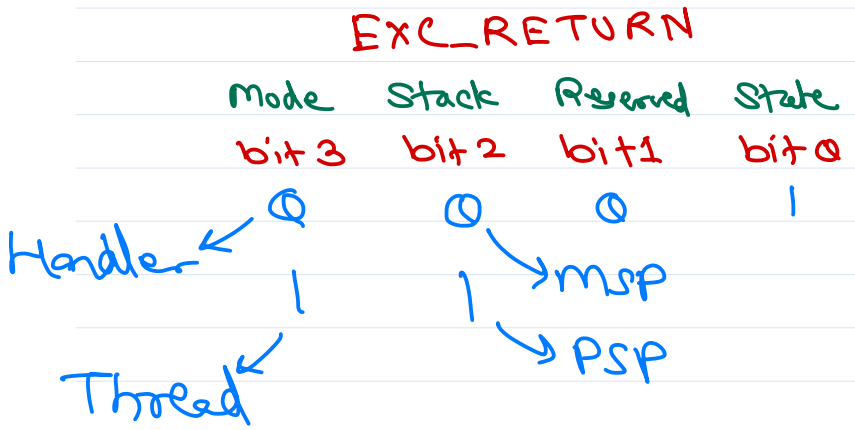


- LR set to EXC_RETURN at exception (Process stack used in thread mode)

Exception/Interrupt exit – Unstacking

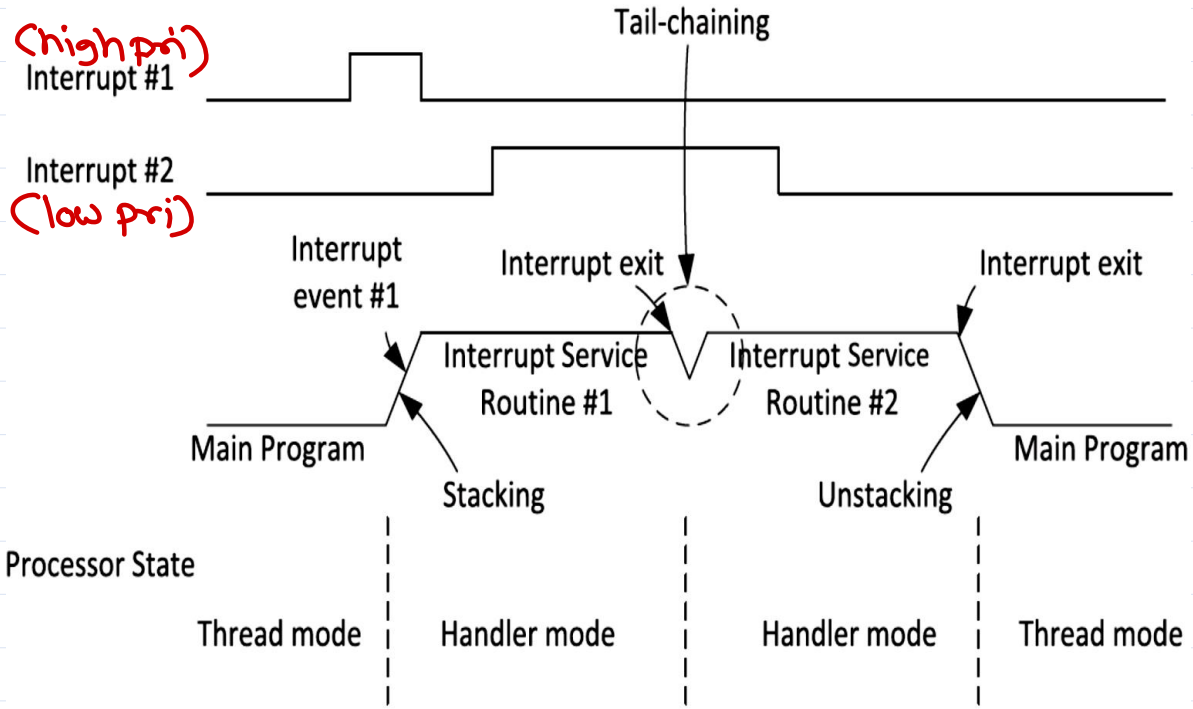


- Unstacking operation based on EXC_RETURN value

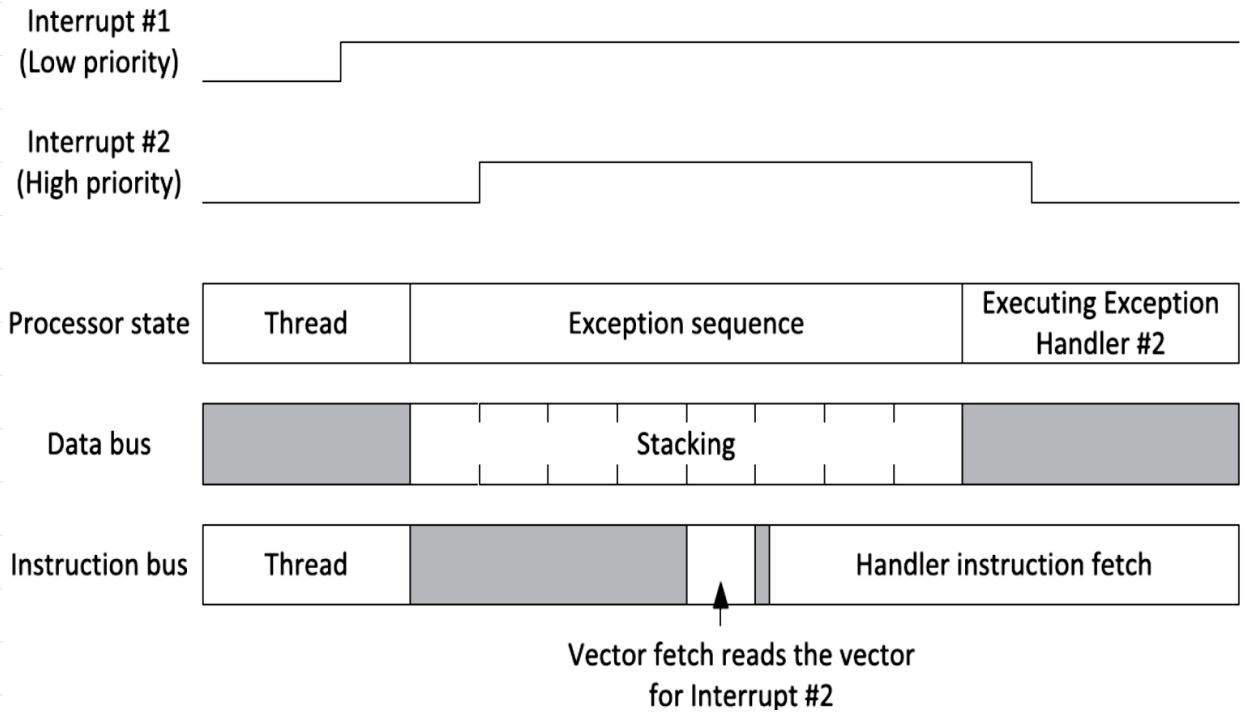


Tail chaining and Late arrival

Intr latency 6 cycles.



- If another exception is pending (lower priority) before current handler returns, unstacking is avoided and directly next handler execution continues after fetching its address.



- If another exception (higher priority) arrives before vector fetch for current exception, high priority exception handler is executed before even if it is late arrived.

ARM CM3 Faults

- MemManage Fault
  - Faults due to MPU configuration (unprivileged code accessing privileged region, accessing location not defined in MPU, writing to read only region).
  - Executing code from xN (execute never) attribute.
  - Can be enabled by
    - SCB->SHCSR |= BV(MEMFAULT); //bit 16
- Bus Fault
  - Errors received from processor bus interface during memory access.
  - Like pre-fetch abort and data abort.
  - Also when device to access is not ready.
  - Can be enabled by
    - SCB->SHCSR |= BV(BUSFAULT); //bit 17
- Usage Fault
  - Accessing a feature that is not available in hw.
  - Undefined instruction executed.
  - Accessing FPU when not available.
  - Accessing Co-processor (not available in CM3)
  - Switching to ARM state (i.e. T=0).
  - Invalid EXC_RETURN value.
  - Executing SVC when not permitted (in lower interrupt level).
  - Divide by zero or unaligned memory access (if configured).
  - Can be enabled by
    - SCB->SHCSR |= BV(~~MEM~~FAULT); //bit 18
- Hard Fault
  - When any fault occurs, but not enabled in code
  - Bus error while vector fetch.
  - BKPT when debug exception disabled.

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>