ARM EXCEPTIONS AND INTERRUPTS

By: Muhammad ElZeiny

EXCEPTION STATES

INACTIVE

The exception is not active and
not pending.

PENDING

The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

ACTIVE

An exception that is being serviced by the processor but
has not completed.
Note: An exception handler can interrupt the execution of
another exception handler. In this case, both exceptions are in the active state.

ACTIVE AND PENDING

An exception that is being serviced by the processor but
has not completed.
Note: An exception handler
can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.

EXCEPTION TYPES

RESET

• Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception.
• When reset is asserted, the operation of the processor stops, potentially at any point in an instruction.
• When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table.
• Execution restarts as privileged
execution in Thread mode.

NMI

• A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the Interrupt Control and State (INTCTRL) register.
• This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2.
• NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset

HARD FAULT

• A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism.

• Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority

EXCEPTION TYPES

MEMORY MANAGEMENT FAULT

- A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match.
- The MPU or the fixed memory protection constraints determine this fault, for both instruction and datamemory transactions.

BUS FAULT

•• A bus fault is an exception that occurs because of a memory-related fault for an instruction or data memory
transaction such as a prefetch fault or a memory access fault.

- This fault can be enabled or disabled.

USAGE FAULT

- A usage fault is an exception that occurs because of a fault related to instruction

- execution, such as:

- An undefined instruction
Invalid state on instruction execution
An error on exception return
division by zero

EXCEPTION TYPES

SVCALL

•A supervisor call (SVC) is an exception
that is triggered by the SVC instruction.

• In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.

PENDSV

• PendSV is a pendable, interrupt-driven request for system-level service.

• In an OS environment, use PendSV for context switching when no other exception is active.

• PendSV is triggered using the Interrupt Control and State (INTCTRL) register.

SYSTICK

• A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to
generate an interrupt.

• Software can also generate a SysTick exception using the Interrupt Control and State (INTCTRL) register.

• In an OS environment, the processor can use this exception as system tick.

EXCEPTION TYPES

INTERRUPTS

- An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized).

- All interrupts are asynchronous to instruction execution.

- In the system, peripherals use interrupts to communicate with the processor.

DEBUG MONITOR

- This exception is caused by the debug monitor.

- This exception does not activate if it is a lower priority than the current activation.

EXCEPTION HANDLER

INTERRUPT SERVICE ROUTINES (ISRS)

1. Interrupts (IRQx)

- are the exceptions handled by ISRs.

1. Hard fault
2. memory management fault
3. usage fault
4. bus fault

- are fault exceptions handled by the fault handlers.

FAULT HANDLERS

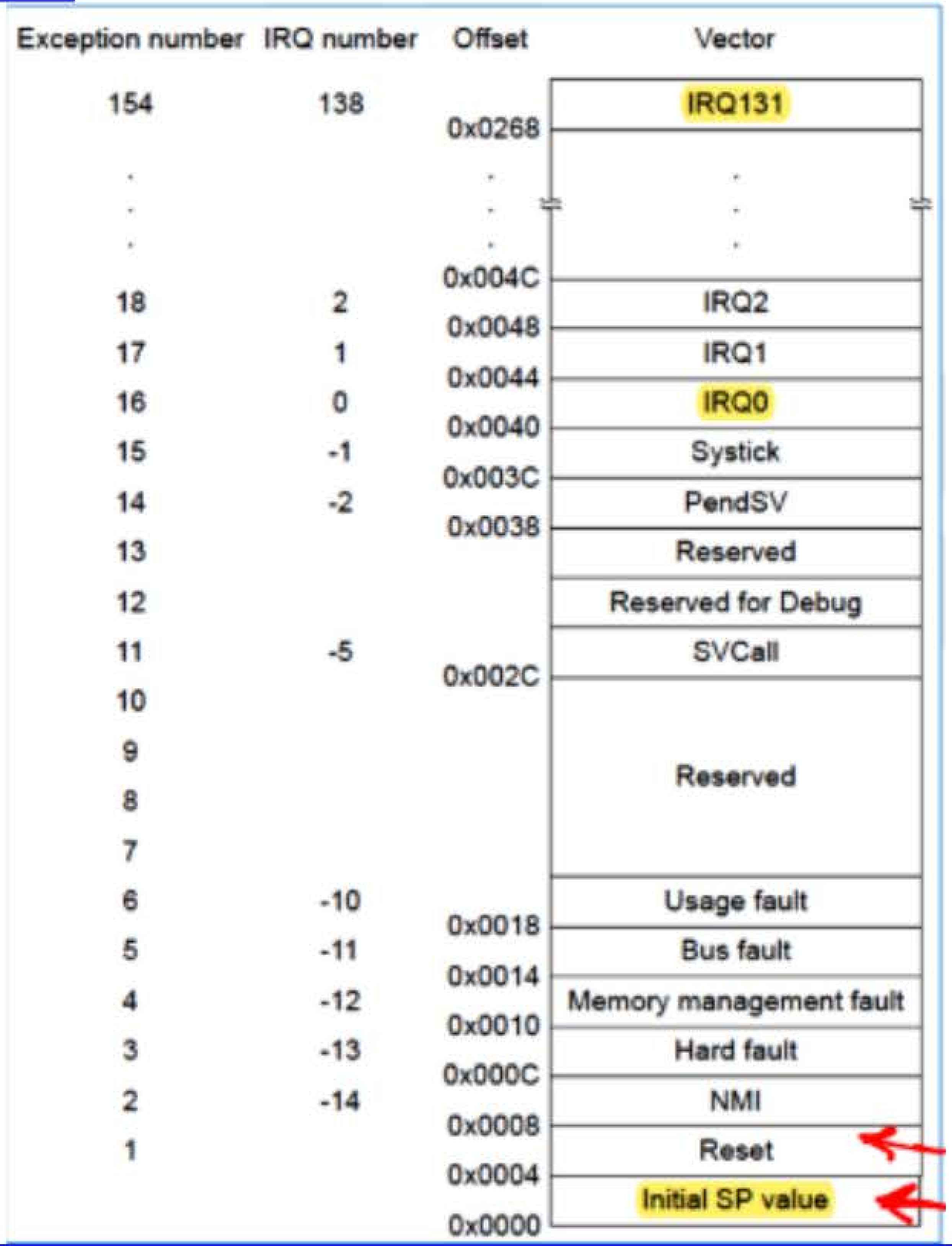
SYSTEM HANDLERS

1. NMI
2. PendSV
3. SVCall
4. SysTick
5. the fault exceptions

- are all system exceptions that are handled by system handlers.

Vector Table

- The vector table contains the reset value of the stack pointer and the start addresses of all handlers and ISR.
- On system reset, the vector table is fixed at address 0x0000.0000.
- Privileged software can relocate the vector table start address to a different memory location



Exception number	IRQ number	Offset	Vector
154	138	0x0268	IRQ131
. . .		. . .	. . .
		0x004C	
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			Reserved
9			
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

EXCEPTION RETURN AND ENTRY

Preemption

- When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled.

Return

- Return • Return occurs when the exception handler is completed, <u>and</u> there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late arriving exception.

- The processor pops the stack and restores the processor state to the state it had before the interrupt occurred.

EXCEPTION RETURN AND ENTRY

Tail Chaining

- This mechanism speeds up exception servicing.

- On completion of an exception handler, if there is a pending that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

EXCEPTION RETURN AND ENTRY

Late Arriving

- This mechanism speeds up preemption
- If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception.
- State saving is not affected by late arrival because the state saved is the same for both exceptions.
- Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor.
- On return from the exception handler of the late-arriving exception, the normal tail chaining rules apply.

FAULT HANDLING

- The following conditions generate a fault
  - A bus error on an instruction fetch or vector table load or a data access.
  - An internally detected error such as an undefined instruction or an attempt to change state with BX instruction.
  - Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
  - An MPU fault because of a privilege violation or an attempt to access an unmanaged region.

HARD FAULT ESCALATION

• In the following situations, a fault with configurable priority is treated as a hard fault :

1. A fault handler causes the same kind of fault as the one it is servicing.
It occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.

2. A fault handler causes a fault with the same or lower priority as the fault it is servicing.
This situation happens because the handler for the new fault cannot preempt the currently executing fault handler.

3. An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.

4. A fault occurs and the handler for that fault is not enabled.

FAULTS TYPE

Hard fault	Memory management fault	Bus fault	Usage fault
·Bus error on a vector read ·Fault escalated to a hard fault	·MPU or default memory mismatch on instruction access ·MPU or default memory mismatch on data access ·MPU or default memory mismatch on exception stacking ·MPU or default memory mismatch on exception unstacking ·MPU or default memory mismatch during lazy floating-point state preservation	·Bus error during exception stacking ·Bus error during exception unstacking ·Bus error during instruction prefetch	·Attempt to access a coprocessor ·Undefined instruction ·Attempt to enter an invalid instruction ·Invalid EXC_RETURN value ·Illegal unaligned load or store ·Divide by 0

QUIZ

- The Interrupt can preempt itself.

❑ TRUE
❑ FALSE

- If an exception handler causes a lower group priority interrupt. What will happen ?

❑ The new Interrupt will preempt the running one.
❑ Hard fault escalation
❑ The new Interrupt will be activated after the running one finishes ISR execution.

- If an exception handler causes a fault for which the priority is the same as. What will happen?

❑ The new Interrupt will preempt the running one.
❑ Hard fault escalation
❑ The new interrupt will be activated after the running one finishes ISR execution.

- Exception Return always occurs when the exception handler is completed.

❑ True
❑ False

- After every late-arriving mechanism occurs a tail-chaining mechanism will occur.

❑ True
❑ False

- PendSV exception used for switching to privilege mode.

❑ True
❑ False