INTERRUPT

(Blue: 4, Green: 2, Red: 1)

- exception is any condition that needs to halt normal execution of the instructions, such as core resets, failures, interrupts
- 16 cycles from exception request to execution of first instruction

Finish current instruction

Except for lengthy instructions

Push context (registers) onto current stack (MSP or PSP)

xPSR, Return Address, LR(R14), R12, R3, R2, R1, R0

Switch to handler/privileged mode, use MSP

Load PC with address of interrupt handler

Load LR with EXC_RETURN code

Load IPSR with exception number

Start executing code of interrupt handler

Bit 2

EXC_RETURN	Return Mode	Return Stack	Description
0xFFFF_FFF1	0 (Handler)	0 (MSP)	Return to exception handler
0xFFFF_FFF9	1 (Thread)	0 (MSP)	Return to thread with MSP
0xFFFF_FFFD	1 (Thread)	1 (PSP)	Return to thread with PSP

Hardware interrupts

Asynchronous: not related to what code the processor is currently executing

Exceptions, faults, software interrupts

Synchronous: are the result of specific instructions executing

Examples: undefined instructions, overflow occurs for a given instruction

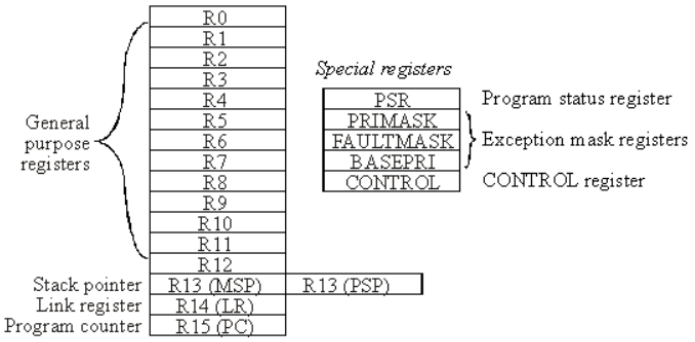
Exception states: Inactive, Pending, Active, A&P

requests (smaller number = higher priority)

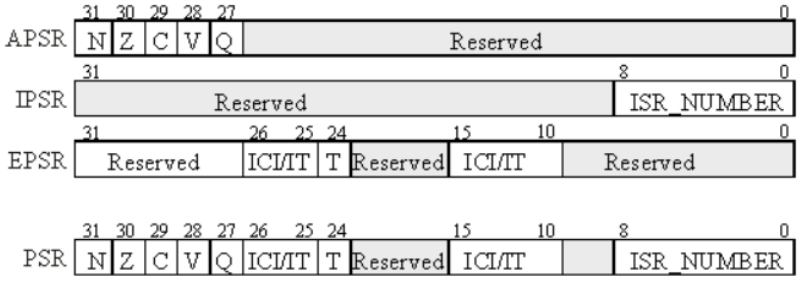
- $F_{Max\_Int} = F_{CPU}/(C_{ISR}+ C_{Overhead})$
- $U_{Int} = 100\%*F_{Int}* (C_{ISR}+C_{Overhead})/ F_{CPU}$

Compilers assume that variables in memory don't change spontaneously so we put volatile

- Race condition: Anomalous behavior due to unexpected critical dependence on the relative timing of events. Result of example code depends on the relative timing of the read and write operations.
- Critical section: A section of code which creates a possible race condition. The code section can only be executed by one process at a time. Some synchronization mechanism is required at the entry and exit of the critical section to ensure exclusive use.



16 register last 3 is msp → lr → pc , 5 special



There are three key status registers:

- **Application Program Status Register (APSR)**
- **Interrupt Program Status Register (IPSR)**
- **Execution Program Status Register (EPSR)**

These registers can be accessed together as the **Program Status Register (PSR)**.

- **N (Negative flag)**: Set if the result of an arithmetic or logical operation is negative.
- **Z (Zero flag)**: Set if the result is zero.
- **C (Carry flag)**: Indicates an **unsigned overflow** has occurred.
- **V (Overflow flag)**: Indicates a **signed overflow** has occurred.
- **Q (Saturation flag)**: Indicates **saturation** in arithmetic operations (advanced concept).

T Bit (Thumb State Indicator):

- Always set to **1** in ARM Cortex-M, indicating the processor executes **Thumb® instructions only**.

ISR_NUMBER (Interrupt Service Routine Number):

- Identifies the interrupt being handled by the processor. (IRQ)

PRIMASK Register:

- **Bit 0 (Interrupt Mask Bit)**:
    - **1**: Masks (disables) most interrupts and exceptions.
    - **0**: Allows interrupts.
- Does not affect the **Nonmaskable Interrupt (NMI) or Faults**.

FAULTMASK Register:

- **Bit 0 (Fault Mask Bit)**:
    - **1**: Masks all interrupts and faults.
    - **0**: Allows interrupts and faults.
- NMI remains unaffected.

BASEPRI Register:

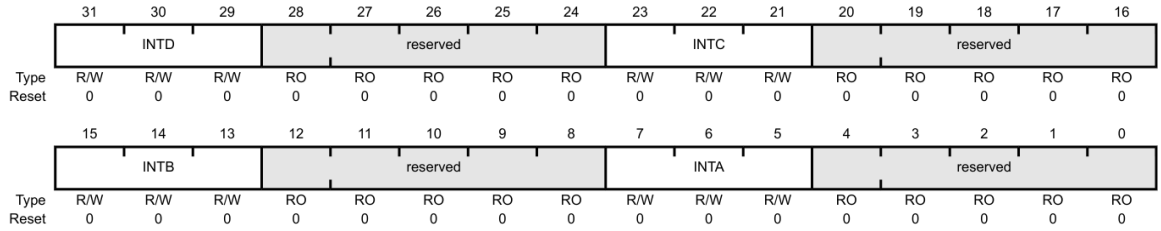
- Controls interrupt priority levels:
    - Defines the **minimum priority level** that can block interrupts.
    - Higher priority (lower numerical value) interrupts can preempt.
- Example:
    - **BASEPRI = 3**: Interrupts with priority levels 0, 1, and 2 are allowed.

- In IRQ[0]> IRQ[1] ,,, but priority number -3>-2 ,, vector number smallest the biggest

- 138 interrupts. 5 Enable register 0→4 , 32 Priority 0→31

- IRQ / 4 → priority register ,, IRQ % 4 → priority of displacement
- Enable 35 / 32 → number of enable ,, 35 % 32 → number of bit

- Vector number = IRQ+16



REGISTERS ：

GPIO_PORTF_IS_R: Configure edge sensitivity 0→edge.

GPIO_PORTF_IBE_R: Configure both edge interrupt 0→ single.

GPIO_PORTF_IEV_R: Set interrupt edge (falling /rising) 0→ falling.

GPIO_PORTF_ICR_R: Clear interrupt flag 1→ clear.

GPIO_PORTF_IM_R: Enable interrupt 1→ enable.

NVIC_EN0_R: Enable GPIO Port F interrupt in NVIC.

NVIC_PRI7_R: Set interrupt priority for GPIO Port F.