La gestione delle eccezioni in RISC-V

M. Sonza Reorda, Luca Sterpone,
M. Rebaudengo

Politecnico di Torino
Dipartimento di Automatica e Informatica

Politecnico
di Torino

Eccezioni

Si definisce eccezione (*exception*) qualsiasi modifica imprevista nel flusso di controllo, indipendentemente dalla sua origine (che può essere interna o esterna al processore).

Un'eccezione si dice *sincrona* se si verifica nello stesso punto ogni volta che viene eseguito un programma con gli stessi dati e la stessa configurazione di memoria (ad es. overflow, istruzioni illegali, page fault).

Le eccezioni *asincrone*, invece, si verificano senza relazione temporale con il programma in esecuzione (ad es. per richieste provenienti da dispositivi di I/O, errori di memoria, malfunzionamenti per cadute di tensione).

Cause di eccezione

Il processore può scatenare un'eccezione sincrona a seguito di uno dei seguenti eventi:

- **Misaligned Address: l'indirizzo di destinazione di un'istruzione di salto non è multiplo di 4**
- **Access Fault: un'operazione di fetch o un'istruzione load o store ha tentato di fare accesso a un indirizzo non permesso e l'unità Physical memory protection (PMP) ha rilevato il problema**
- **Illegal Instruction: l'istruzione di cui si è fatto il fetch non corrisponde ad una codifica nota**
- **Environment Call e Environment Break: è stata eseguita una `ecall` o una `ebreak`**
- **Page Fault.**

Interrupt e Trap

Fanno parte delle eccezioni le seguenti tipologie:

- ***Interrupt hardware*: eccezione di tipo asincrona causata da una richiesta proveniente da un dispositivo periferico esterno**
- ***Trap* o *system call*: eccezione sincrona che provoca il trasferimento del controllo intenzionale verso il sistema operativo, ad es. a seguito di un page fault**
- ***Software interrupt*: eccezione di tipo sincrona scatenata da una particolare situazione che si verifica nell'esecuzione del programma (ad es. overflow aritmetico, istruzione illegale o breakpoint)**
- ***Timer interrupt*: segnale che interrompe la CPU a intervalli regolari.**

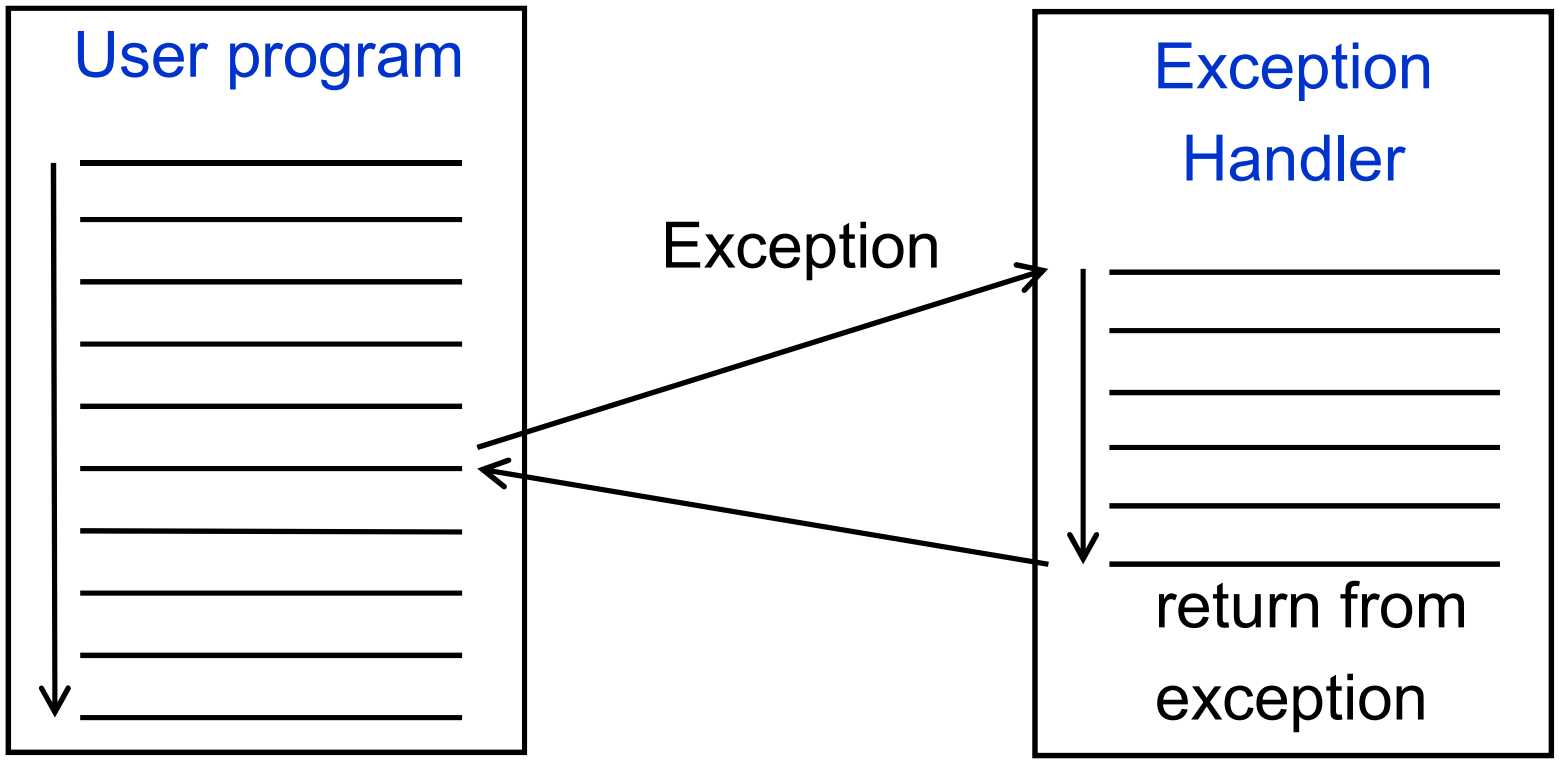
Gestione di una eccezione

Quando si verifica un'eccezione, il controllo viene trasferito a un gestore di eccezioni (*exception handler*), corrispondente ad un pezzo di codice scritto specificamente allo scopo di gestire le eccezioni.

Dopo aver eseguito l'exception handler, il controllo viene restituito al programma. Il programma continua in modo indipendente.

L'exception handler si comporta come una procedura chiamata in un generico istante, senza parametri e senza valore di ritorno.

Gestione di una eccezione



Tipi di eccezioni

In RISC-V si distinguono due tipi principali di eccezioni:

- **Interruzioni (Interrupts): eventi asincroni esterni (come timer, I/O)**
- **Exception / Trap: eventi sincroni interni causati da istruzioni (es. divisione per zero, syscall, accesso invalido alla memoria).**

Livelli di privilegio

L'idea alla base dei livelli di privilegio è quella di creare confini di sicurezza all'interno di un sistema informatico, definendo chi ha il permesso di fare cosa all'interno del sistema hardware e software.

I livelli di privilegio limitano l'accesso alla memoria o a certe istruzioni (privilegiate). I tentativi di eseguire operazioni non permesse dal livello di privilegio corrente causeranno il sollevamento di un'eccezione.

I modi di privilegio di RISC-V sono (dal più alto al più basso):

- **Modo Macchina (bare metal)**
- **Modo Supervisore (sistema operativo)**
- **Modo Utente (programma utente).**

Machine Mode (M-mode)

È il livello di privilegio più elevato e ha il controllo completo sull'hardware sottostante.

Il codice in M-mode può accedere a qualsiasi indirizzo di memoria, eseguire qualsiasi istruzione (incluse quelle privilegiate), configurare i dispositivi di I/O e manipolare direttamente i registri di controllo del processore (CSRs) che governano il funzionamento dell'intero sistema.

Un errore o un comportamento malevolo in M-mode può compromettere l'intero sistema. Per questo motivo, la quantità di codice che gira in M-mode è mantenuta al minimo necessario.

Tutti i core RISC-V devono implementare l'M-mode come base per l'avvio e la configurazione del sistema.

Tipicamente, il codice che gira in M-mode include il bootloader (il software che avvia il sistema all'accensione) e il firmware di basso livello (come BIOS o UEFI).

In sistemi embedded molto semplici, l'intero sistema operativo potrebbe essere eseguito in M-mode.

Supervisor Mode (S-mode)

È un livello intermedio progettato per ospitare un sistema operativo.

Il codice in S-mode ha una visione più astratta dell'hardware, spesso attraverso meccanismi come la memoria virtuale gestita dall'MMU (Memory Management Unit). Può gestire i processi utente, allocare risorse (memoria, CPU, I/O) e rispondere alle loro richieste (tramite chiamate di sistema).

L'S-mode introduce concetti come la protezione della memoria tra i processi utente e tra il sistema operativo stesso e i processi utente.

Le capacità dell'S-mode sono generalmente limitate dalle configurazioni impostate dall'M-mode. Ad esempio, l'M-mode può definire quali aree di memoria sono accessibili all'S-mode e quali interrupt può gestire.

User Mode (U-mode)

È il livello di privilegio più basso, destinato all'esecuzione delle applicazioni utente.

Il codice in U-mode opera in un ambiente strettamente controllato dal sistema operativo in S-mode. Ha accesso solo alla propria area di memoria virtuale e a un insieme limitato di risorse di sistema, gestite tramite le chiamate di sistema.

Le applicazioni in U-mode non possono accedere direttamente all'hardware, eseguire istruzioni privilegiate (come la modifica dei registri di controllo del sistema o l'invalidazione della cache) o interferire con il funzionamento di altre applicazioni o del sistema operativo.

Qualsiasi tentativo di un programma in U-mode di violare queste restrizioni (ad esempio, tentando di accedere a un indirizzo di memoria al di fuori della sua area allocata o eseguendo un'istruzione privilegiata) causerà un'eccezione.

Exception Registers

Ogni livello di privilegio ha i propri registri per la gestione delle eccezioni.

Questi registri sono chiamati CSR (*Control and Status Register*).

I registri CSR definiscono lo stato di funzionamento della CPU.

Quando viene scatenata un'eccezione e parte l'exception handler del modo M, l'hardware automaticamente aggiorna i seguenti CSR: `mepc`, `mtval`, `mcause` e `mstatus`.

Registro mcause

Indica la causa dell'eccezione.

Il bit più significativo indica se è un'interruzione (1) o un'eccezione (0).

I restanti 3 bit rappresentano il codice che descrive la causa dell'eccezione.

Exception	Cause
Instruction address misaligned	0
Instruction access fault	1
Illegal instruction	2
Breakpoint	3
Load address misaligned	4
Load access fault	5
Store address misaligned	6
Store access fault	7
Environment call from U-Mode	8
Environment call from S-Mode	9
Environment call from M-Mode	11

Registro mepc

Contiene il valore del Program Counter dell'istruzione che ha causato l'eccezione.

Serve per riprendere l'esecuzione al ritorno dell'eccezione.

Registro `mstatus`

È un registro a 64 bit che contiene alcuni campi per configurare il comportamento della CPU.

Tra i vari campi si possono citare:

- **MIE (Machine Interrupt Enable): abilitazione/disabilitazione degli Interrupt in M-Mode. È rappresentato su 1 bit. Quando viene attivato l'exception handler, MIE viene posto a 0 (disabilitando gli interrupt)**
- **MPP (Machine Previous Privilege): indica su 2 bit il livello di privilegio in cui si trovava la CPU prima dell'eccezione (11 $\Rightarrow$ M-mode, 01 $\Rightarrow$ S-mode, 00 $\Rightarrow$ U-mode). Al ritorno (`mret`), il livello di privilegio è settato a quanto memorizzato in MPP**
- **MPIE (Machine Previous Interrupt Enable): salva il valore di MIE prima dell'eccezione. Al ritorno (`mret`), MIE è ripristinato da MPIE. È rappresentato su 1 bit.**

Registro `mtval`

Il registro `mtval` è scritto con informazioni specifiche a seconda dell'eccezione.

Nel caso di misaligned addresses, access faults, e page faults, `mtval` contiene l'indirizzo virtuale che ha generato l'errore.

Nel caso di illegal instruction `mtval` contiene l'indirizzo dell'istruzione che ha generato l'errore

Le eccezioni scatenate da EBREAK ed ECALL forzano `mtval` a zero.

Altri registri

- **`mscratch`: utilizzato per gestire un puntatore ad uno spazio di memoria per salvare dati di contesto**
- **`mip`: contiene lo stato degli interrupt pendenti**
- **`mie`: contiene i bit di disabilitazione delle sorgenti di interrupt (interrupt esterni, timer, interrupt software).**

Istruzioni privilegiate

Accedono ai registri CSR

- **`csrr`: lettura di un registro CSR**
- **`csrw`: scrittura di un registro CSR**
- **`csrrw`: operazione di lettura e scrittura di un registro**
- **`mret`: istruzione di ritorno all'indirizzo memorizzato nel registro `mepc`**

Esempio:

```
csrr t1, mcause          # t1 = mcause
csrw mepc, t2            # mepc = t2
csrrw t0, mscratch, t1 # t0 = mscratch
                         # mscratch = t1
```

Gestione di un’eccezione

Quando si verifica un’eccezione si hanno i seguenti passaggi:

- **Rilevamento**
- **Salvataggio dello stato**
- **Trasferimento del controllo**
- **Gestione dell’eccezione**
- **Ritorno.**

Salvataggio dello stato

Lo stato corrente del processore viene automaticamente salvato dall'hardware del processore negli opportuni registri CSR, specifici per la gestione delle eccezioni:

- **PC corrente è salvato in `mepc`**
- **il livello di privilegio corrente è salvato in MPP (in `mstatus`)**
- **il vecchio valore di MIE (in `mstatus`) è salvato in MPIE (in `mstatus`)**
- **le interruzioni sono disabilitate (agendo su MIE in `mstatus`).**

Trasferimento del controllo

Il controllo dell'esecuzione viene trasferito a un exception handler, diverso a seconda del modo in cui si trova il processore al momento dell'eccezione. Per default si può usare il solo exception handler del modo M.

L'indirizzo di questo handler è specificato in un registro CSR (`mtvec` per le eccezioni gestite in M-mode, `stvec` per S-mode, ecc.).

Durante la configurazione del sistema, il registro `mtvec` (ed eventualmente gli altri due) viene inizializzato con l'indirizzo dell'exception handler.

Il livello di privilegio viene tipicamente elevato a quello in cui è configurato l'handler (ad esempio, da U-mode a S-mode o da qualsiasi livello a M-mode).

Gestione dell'eccezione

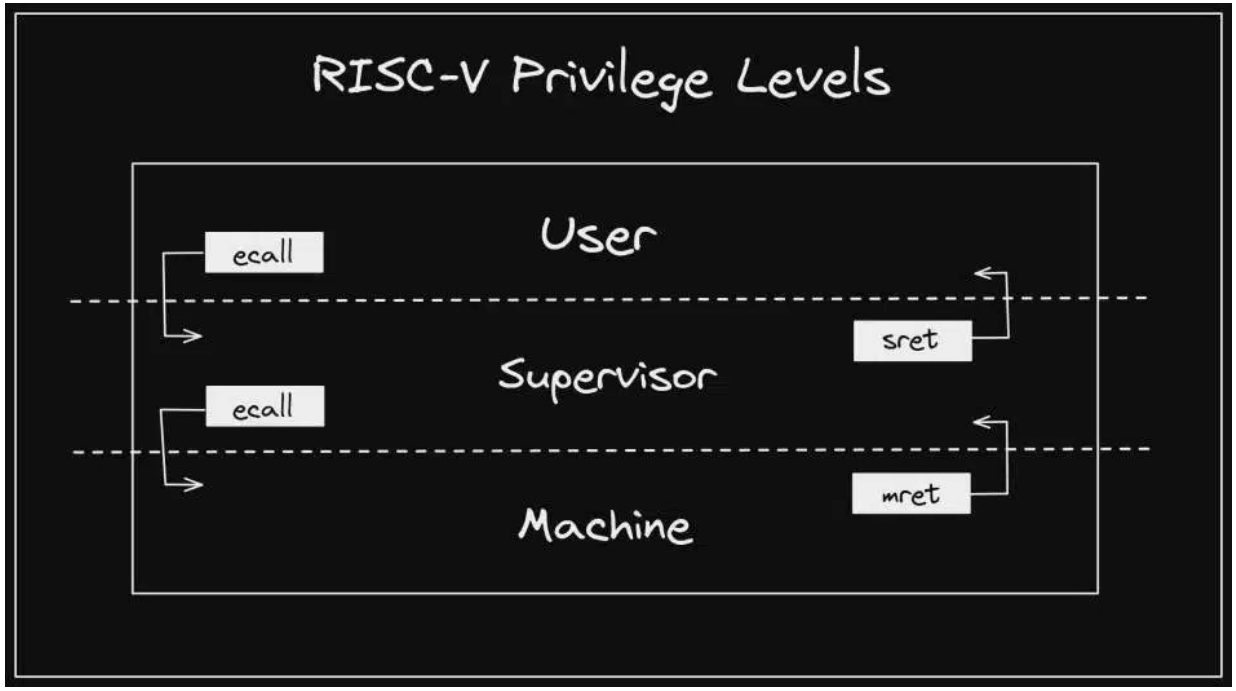
L'exception handler (che fa parte del sistema operativo o del firmware) analizza la causa dell'eccezione (tramite il registro `mcause`) e intraprende l'azione appropriata.

Questa azione può includere:

- **Terminare il programma che ha causato l'eccezione (in caso di errore grave in U-mode)**
- **Segnalare un errore all'utente**
- **Tentare di recuperare dalla condizione anomala (in alcuni casi)**
- **Inviare un segnale a un processo (in sistemi operativi)**
- **Nel caso di una chiamata di sistema (`ecall`), l'handler del sistema operativo esegue il servizio richiesto.**

Ritorno

Dopo che l'eccezione è stata gestita, l'handler esegue un'istruzione di ritorno (`mret`, `sret`, `uret`) per ripristinare lo stato precedente del processore (incluso il PC e il livello di privilegio) e riprendere l'esecuzione del programma interrotto (se appropriato).



Istruzione `mret`

Esegue le seguenti operazioni

- **Forza il PC al valore contenuto in `mepc`**
- **Forza MIE al valore presente in MPIE**
- **Ritorna il livello di privilegio a quello del momento in cui era stata scatenata l'eccezione.**

Le istruzioni `sret` e `uret` hanno un comportamento analogo.

Priorità delle eccezioni

Se un'istruzione scatena più eccezioni sincrone, queste vengono gestite secondo le priorità seguenti

Priority	Exception Code	Description
Highest	12	Instruction Page Fault
	1	Instruction Access Fault
	2	Illegal Instruction
	0	Instruction Address Misaligned
	8, 9, 11	Environment Call (U, S, M)
	3	Environment Break
	6	Store/AMO Address Misaligned
	4	Load Address Misaligned
	15	Store/AMO Page Fault
	13	Load Page Fault
	7	Store/AMO Access Fault
Lowest	5	Load Access Fault

Esempio

Codice di gestione dell'eccezione con verifica di 2 tipi di eccezione:

- **Illegal instruction (`mcause` = 2)**
  - **In questo caso l'exception handler dell'esempio passa ad eseguire l'istruzione successiva**
- **Load address misaligned (`mcause` = 4)**
  - **In questo caso l'exception handler dell'esempio interrompe il programma.**

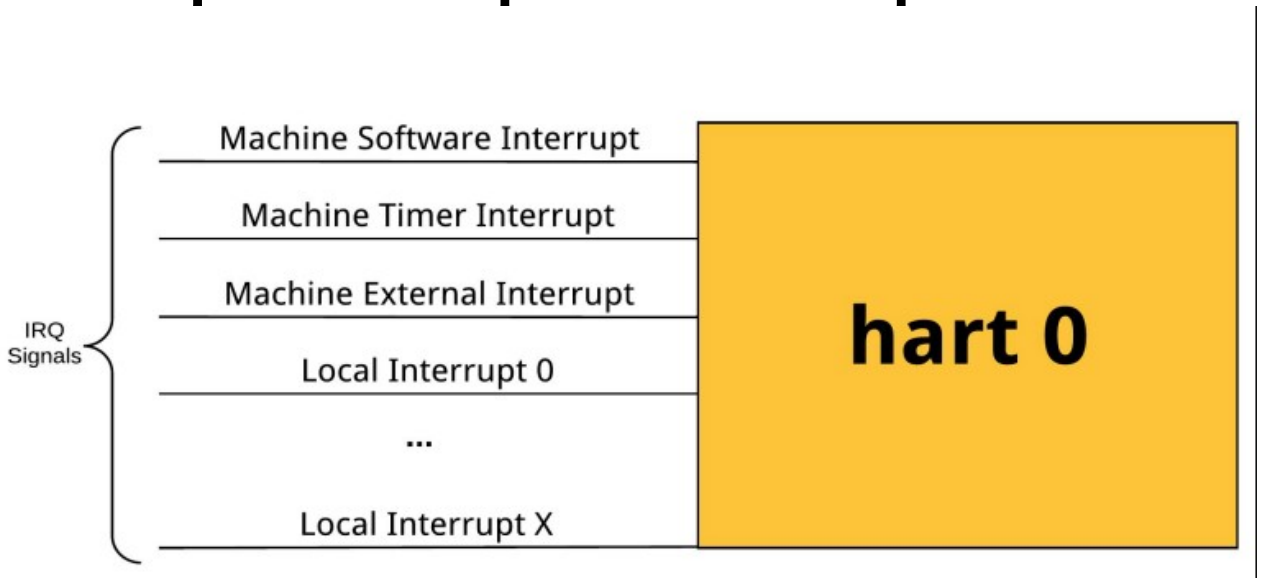
Exception Handler

```
# save registers that will be overwritten
  csrrw t0, mscratch, t0    # swap t0 and mscratch
  sw    t1, 0(t0)           # [mscratch]   = t1
  sw    t2, 4(t0)           # [mscratch+4] = t2
# check cause of exception
  csrr  t1, mcause          # t1=mcause
  addi  t2, x0, 2           # t2=2 (illegal instruction exception code)
illegalinstr:
  bne   t1, t2, checkother  # branch if not an illegal instruction
  csrr  t2, mepc            # t2=exception PC
  addi  t2, t2, 4           # increment exception PC
  csrw  mepc, t2            # mepc=t2
  j     done                # restore registers and return
checkother:
  addi  t2, x0, 4           # t2=4 (load address misaligned exception code)
  bne   t1, t2, done        # branch if not a misaligned load
  j     exit                # exit program
# restore registers and return from the exception
done:
  lw    t1, 0(t0)           # t1 = [mscratch]
  lw    t2, 4(t0)           # t2 = [mscratch+4]
  csrrw t0, mscratch, t0    # swap t0 and mscratch
  mret                      # return to program
exit:
  ...
```

Interrupt

Ogni CPU RISC-V supporta i seguenti tipi di interrupt

- **Software – architecturally defined software interrupt**
- **Timer – architecturally defined timer interrupt**
- **External – Peripheral Interrupts**
- **Local - Hart specific Peripheral Interrupts**

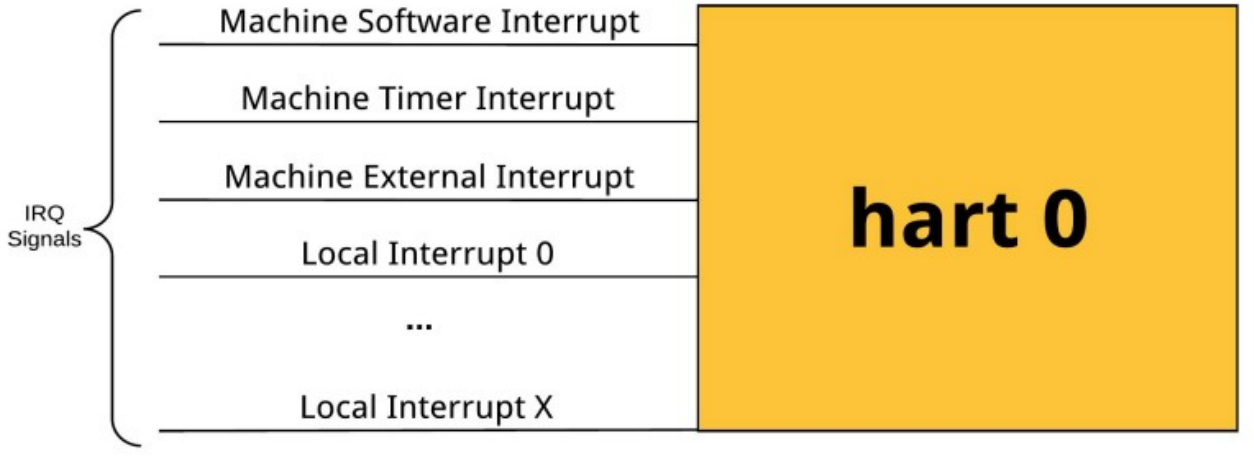


Interrupt

Ogni CPU RISC-V supporta i seguenti tipi di interrupt

- **Software – architecturally defined software interrupt**
- **Timer – architecturally defined timer interrupt**
- **External – Peripheral Interrupts**
- **Local - Hart specific Peripheral Interrupts**

Nei sistemi con più core RISC-V, ognuno prende il nome di «hart»

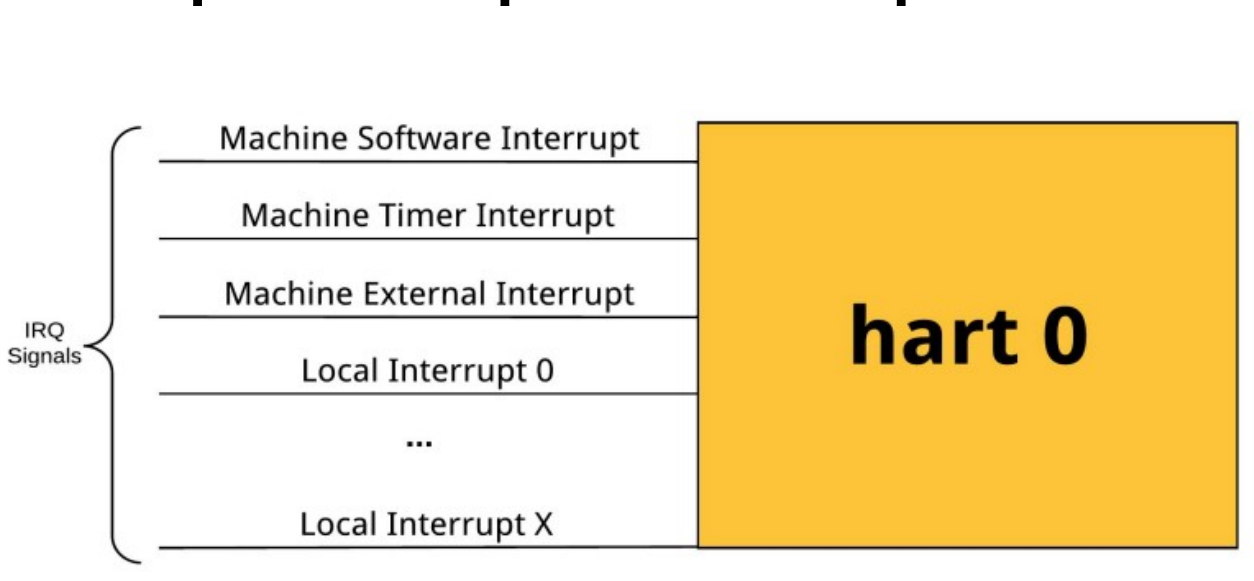


Interrupt

I software interrupt sono le eccezioni interne

Ogni CPU RISC-V supporta i seguenti tipi di interrupt

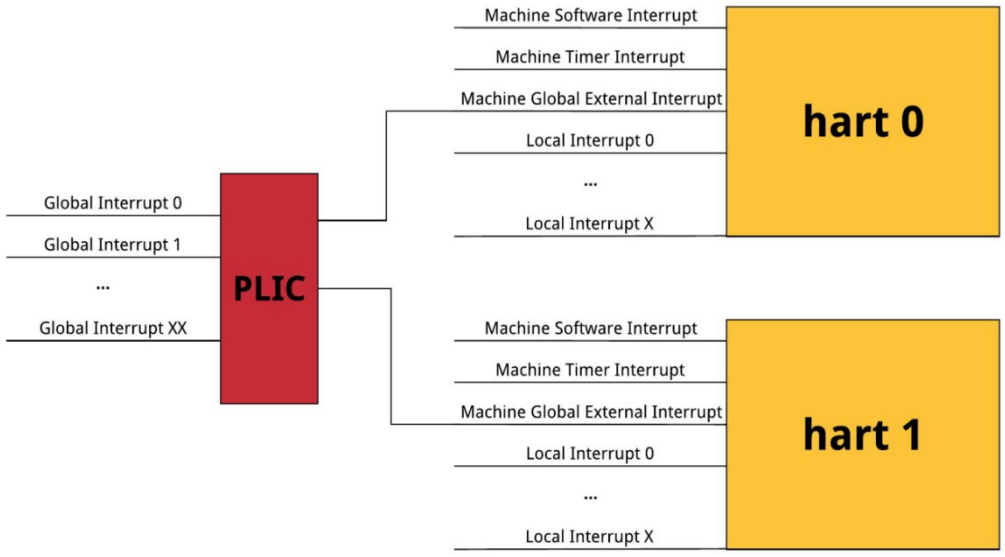
- **Software – architecturally defined software interrupt**
- **Timer – architecturally defined timer interrupt**
- **External – Peripheral Interrupts**
- **Local - Hart specific Peripheral Interrupts**



External interrupts

Sono quelli non specifici di un singolo heart in un sistema multicore.

Sono gestiti attraverso un modulo denominato PLIC (*Platform Level Interrupt Controller*).



Timer interrupts

Lo standard impone che ogni RISC-V includa

- **un CSR denominato `mtime` corrispondente ad un contatore memory mapped su 64 bit**
  - **`mtime` deve essere incrementato con una frequenza che deve essere costante e nota**
- **un CSR memory mapped denominato `mtimecmp` (anch'esso su 64 bit) che può essere scritto con un valore arbitrario**
  - **Quando il valore di `mtime` diventa maggiore o uguale di quello di `mtimecmp`, deve venire scatenato un interrupt.**

Local interrupts

Sono opzionali e specifici di ciascuna implementazione.

Sono frequentemente usati nei sistemi embedded per gestire periferiche, in particolare

- **per le periferiche critiche dal punto di vista della latenza**
- **quando il numero delle periferiche è limitato.**

Quando viene scatenato un interrupt, il processore si comporta esattamente come nel caso di un'eccezione.

Il bit più significativo di `mcause` permette di distinguere i due casi.

Gli altri bit di `mcause` permettono di comprendere la causa scatenante.

`mie` e `mip`

I CSR `mie` e `mip` permettono di

- **Mascherare una singola sorgente di interrupt (`mie`)**
- **Sapere quali richieste di interrupt sono attualmente attive (`mip`).**

`mtvec`

Il CSR `mtvec` contiene due campi

- ***Mode*: permette di scegliere l'interrupt processing mode, che può essere**
  - ***direct*, oppure**
  - ***vectored***
- ***Base*.**

Se è attivo il modo direct, allora l'interrupt handler deve risalire alla sorgente dell'interrupt leggendo il registro `mcause`.

Se è attivo il modo vectored, il processore carica nel PC l'indirizzo

mtvec.Base + (4*mcause.ExCode)

e quindi la latenza dell'interrupt si riduce.

Priorità

Nel caso di richieste di interrupt multiple (e simultanee), il processore le gestisce in base alla seguente tabella delle priorità.

Priority	Exception Code	Description
Highest	11	Machine External Interrupt
	3	Machine Software Interrupt
	7	Machine Timer Interrupt
	9	Supervisor External Interrupt
	1	Supervisor Software Interrupt
	5	Supervisor Timer Interrupt
	8	User External Interrupt
	0	User Software Interrupt
Lowest	4	User Timer Interrupt