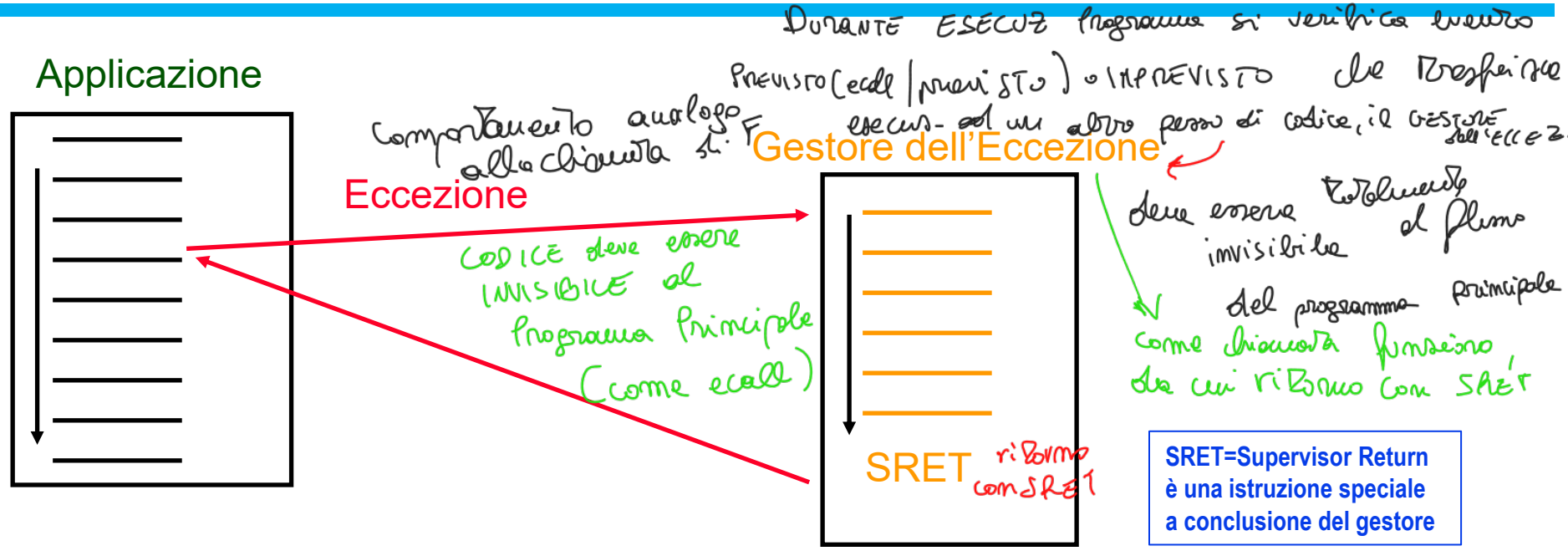
Lezione 14
Il meccanismo di interruzione del processore

Patterson: 4.9



- **Eccezione** = trasferimento del flusso di controllo non legato strettamente al codice utente
  - Il sistema gestisce l'eccezione eseguendo una funzione speciale denominata "Gestore dell'eccezione"
- Esempio
  - l'utente preme un tasto e questo interrompe l'esecuzione del programma utente (qualunque esso sia) e la CPU esegue un "gestore della pressione di un tasto" che tipicamente legge un codice corrispondente al tasto e lo trasferisce in un apposito buffer relativo ai tasti premuti

Comportamento del Calcolatore durante un'eccezione

- Il programma in esecuzione deve essere sospeso e poi riattivato
  - Se l'eccezione è dovuta all'errore di una istruzione, l'istruzione "colpevole" può essere riprovata o simulata e il programma può o continuare o essere bloccato (abort)

  → è necessario che il Calcolatore salvi il punto del codice (program counter) in cui si è verificata l'eccezione (1)

- Il controllo passa quindi ad una "funzione speciale" denominata "Gestore dell'Eccezione" (Exception Handler) o "routine agganciata"
  - Il gestore dell'eccezione deve rimediare alla situazione
  - A seconda del tipo di eccezione, il gestore esegue azioni diverse

  → è ~~necessario che il Calcolatore~~ identifichi e salvi la causa dell'eccezione (2)

- Al momento dell'eccezione, nel processore RISC-V, queste informazioni vengono automaticamente salvate nei registri speciali SEPC e SCAUSE:
  - (1) SEPC → il punto del codice in cui si verifica l'eccezione
  - (2) SCAUSE → la causa dell'eccezione

Che ne è dell'istruzione che era in esecuzione?

- L'architettura RISC-V definisce l'istruzione che causa un'eccezione come una "istruzione senza effetto" ⇒ L'ISTRUZIONE che causa interrupt deve essere ripetuta
- La Routine di Gestione dell'eccezione deve evitare assolutamente di modificare lo stato della macchina
  - → I contenuti di TUTTI i registri x1…x31 (e anche f0…f31) NON devono essere alterati il gestore deve salvare i registri che usa quando chiamo la Routine di gestione
- Durante l'esecuzione dell'eccezione la CPU deve avere pieno controllo di tutte le risorse del Calcolatore disabilito eccez. altri eventi risorse che vorrebbe non devono essere accessibili a tutti
  - → Il codice della routine di eccezione deve essere eseguito in una modalità protetta denominata "modo supervisor" (o modo sistema) – vedi slide successive
- Inoltre le eccezioni devono essere servite una alla volta quando passo alla routine di gestione eccez. Pendenti
  - →La prima cosa da fare è di disabilitare le eccezioni (ad HW); se la routine di gestione lo ritiene opportuno può ri-abilitarle le eccez. devono avere una certa priorità
- Nel processore RISC-V è cura del Gestore dell'Eccezione di salvare i registri che usa; inoltre vengono automaticamente impostati i 2 bit che indicano modalità supervisor ed eccezioni disabilitate, rispettivamente
  - ↳ il Calcolatore passa in modalità supervisor
  - ↳ disabilito le eccezioni degli altri eventi

modalità utente, accesso risorse limitato. risorse istruzioni NON disponibili a tutti

Tutti i servizi, le richieste che intanto arrivano dovranno essere gestite poi

il calcolatore deve fornire 2 Modi di gestione della macchina stessa

Modalità User/Supervisor

- Il calcolatore può gestire sé stesso utilizzando due modalità di esecuzione denominate classicamente
  - Modalità User (accesso alle risorse limitato)
  - Modalità Supervisor accesso illimitato alle risorse del calcolatore

  ⇒ il passaggio viene eseguito dal S.O. ed ha un supporto HW

- Nella modalità Supervisor viene in particolare eseguito il Sistema Operativo, che può essere visto come un programma speciale in esecuzione sul calcolatore in una modalità privilegiata in cui:

  può essere legato ad un meccanismo hardware

  - Tutte le risorse del calcolatore possono essere utilizzate direttamente
  - Al software utente vengono presentate risorse "virtuali" che sono più potenti delle risorse fisiche, es.:
    - si forniscono i "file" anziché i "settori del disco"
    - si fornisce una memoria molto più grande (memoria virtuale)

  S.O. rileva a VIRTUALIZZARE le RISORSE usando modalità sistema

  rileva a virtualizzare le risorse

  Vi rileva degno l'uso delle performance

  rispetto alla memoria reale

  Sistema operativo protegge l'HW

  - Viene garantita la protezione fra i vari programmi utente o di più utenti

- Le eccezioni consentono al sistema di rispondere ad eventi che si verificano mentre un qualsiasi programma utente è in esecuzione
  - Il Sistema Operativo entra in azione dal momento in cui inizia ad essere eseguita la routine di gestione delle eccezioni

Classificazione Generale delle Eccezioni

Eccezione

Evento Asincrono

Interrupt (1)

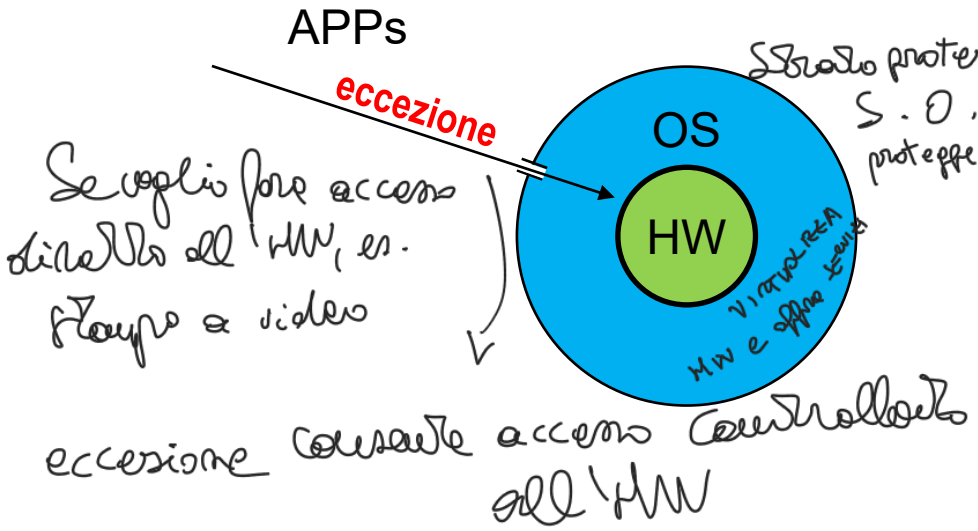
Evento Sincrono

Errore* (2)

Environment Call (3)
(detta anche "Trap" o "System Call")

Environment break (4)

- **L'eccezione permette alle Applicazioni (APPs) di avere accesso al Calcolatore (HW) in modo controllato, attraverso codice che sta nel Sistema Operativo (OS/Kernel):**



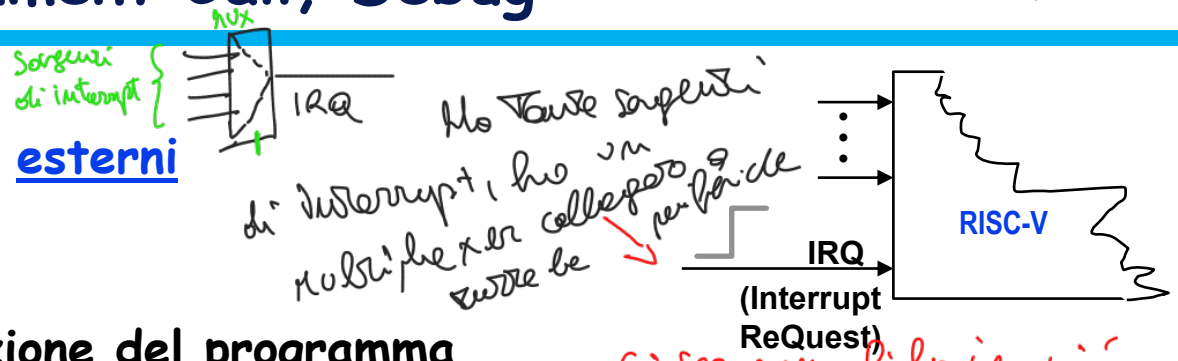
- **Durante l'eccezione il Calcolatore esegue codice del Sistema Operativo e diremo quindi che si trova in modalità Supervisor (o modalità Kernel)**
- **Le APPs vengono invece normalmente eseguite in modalità User (senza poter fare accesso alle risorse privilegiate HW del Calcolatore)**

*Nota: L' "Errore" è una condizione eccezionale, quale un overflow durante l'esecuzione di una istruzione e viene talvolta denominato esso stesso semplicemente col termine "Eccezione"

Interrupt, Errori, Environment Call, Debug

1) Interrupt (interruzione)

- **Eccezione causata da eventi esterni**
  - Pressione di un tasto
  - Movimento del mouse
  - …
- **Asincrona rispetto all'esecuzione del programma**
- **La sua gestione "accade" fra istruzioni di un altro programma**



2) Errore

- **Eccezione causata da eventi interni**
  - Condizioni eccezionali (overflow, divisione per zero)
  - Errori del sistema (parità)
  - Gestione della memoria virtuale (page fault)
- **Sincrona rispetto all'esecuzione del programma**

3) Environment Call (istruzione `ecall` del RISC-V)

- **Eccezione (sincrona) causata da richiesta di un Servizio di Sistema** dal programma
  - Stampa di un messaggio
  - Lettura di un intero
  - …

4) Environment Break (istruzione `ebreak` del RISC-V)

- **Eccezione (sincrona) causata da motivi diagnostici o debugging**

Come gestiamo via software le sorgenti di interrupt?

Routine di Gestione dell'Eccezione (Exception Handler)

verificarsi un'eccez. come chiamo la routine di gestione?

- Esistono vari metodi per implementarla, ad esempio:
  - Salto diretto all'indirizzo della routine di gestione — tutte le interrup. hanno un gestore unico e SW decide cosa fare
  - Vettore di Interruzione — insieme di informaz. hanno portare alla Routine di Gestione
- Lo stato della macchina deve essere preservato
  - Ad es. salvando i registri temporanei nello stack

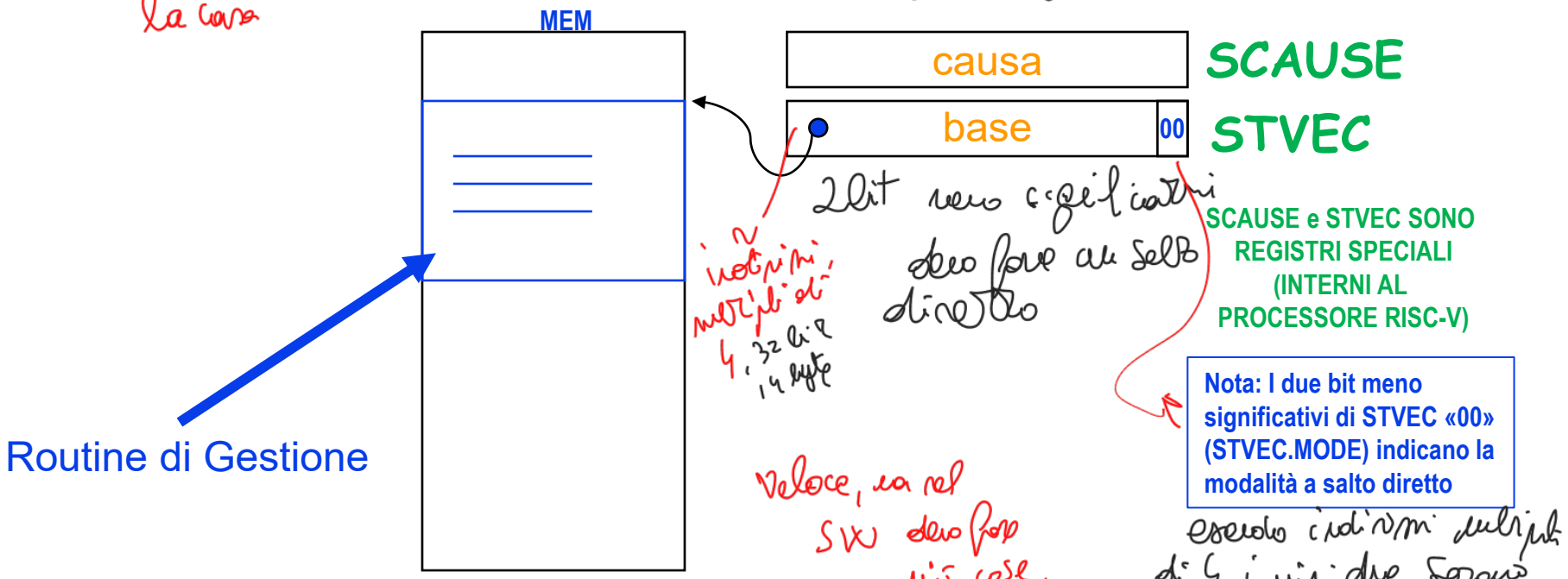
Ricorre, serve gestore unico nel PC. e a SW scelgo cosa come

ogni sorgente di interrupt ha associato in una tabella, l'indirizzo iniziale del suo gestore.

Sia 'causa' il numero di identificazione dell'eccezione — quale gestore far partire — Registro di causa

Salto diretto ad indirizzo della routine di gestione

- **Salto diretto ad un indirizzo specifico: PC ← base**
  - **Nel RISC-V è contenuto nel registro speciale STVEC**
- **Vantaggi:**
  - **Non è necessario fare un accesso in memoria per prelevare l'indirizzo della routine di gestione (più veloce)**
- **Svantaggi:**
  - **Nella routine di gestione occorre analizzare la causa dell'eccezione**

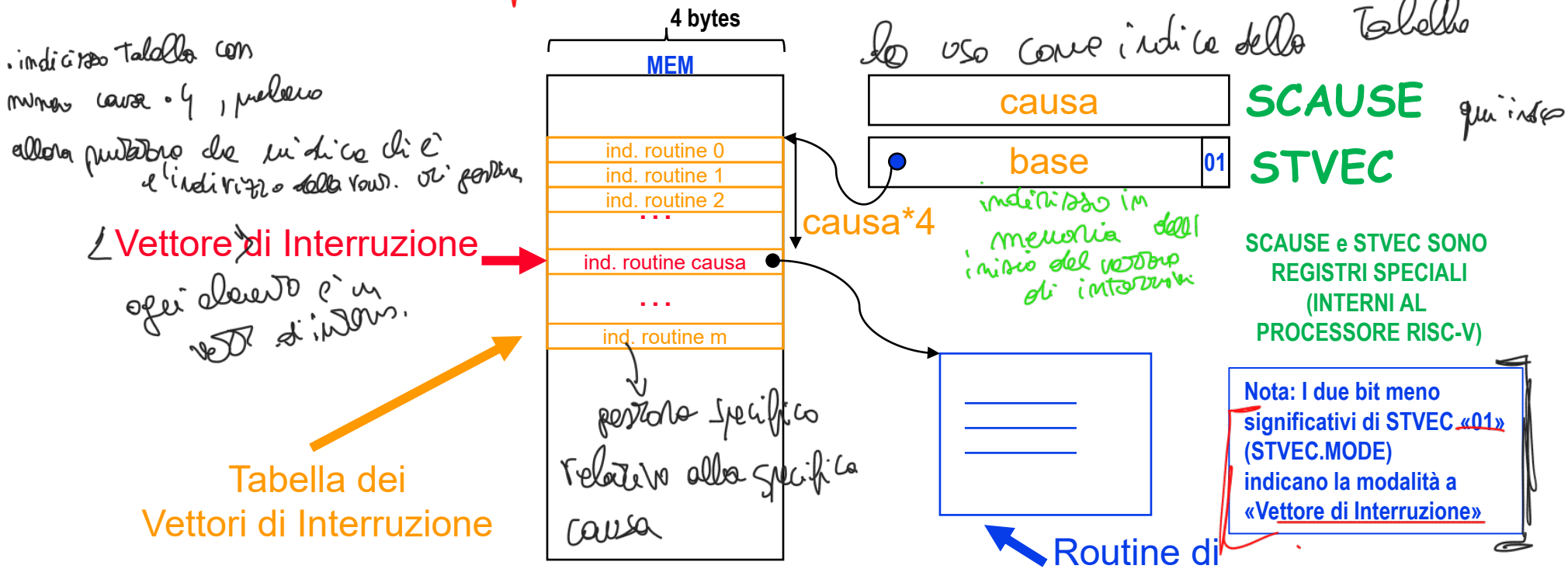


Vettore di Interruzione

- **Memorizzo in una tabella gli indirizzi delle Routine di Gestione associata ad ogni possibile causa (usato da RISC-V e storicamente da 370, 68000, Vax, 80x86, …):**

PC ← MEM[base+causa*4]

dove 'base' è l'indirizzo base di tale tabella



Salvataggio dello stato della macchina al verificarsi di un'eccezione

- Salvataggio sullo stack (Push)
  - Vax, 68k, 80x86 (intero set dei registri)
  - RISC-V, MIPS (solo i registri necessari, potenzialmente zero)

lascio al SW di salvare solo macchina

- Registri ausiliari (Shadow Registers)
  - M88k, ARM 2 copie dei registri, shadow, se ho un interrupt uso registri ombra duplicazione delle risorse
  - Lo stato è salvato in registri ausiliari sia visibili che non visibili direttamento all'utente (es. registri interni della pipeline)

con interrupt uso registro ombra

- Salvataggio in registri speciali salvataggio registri via SOFTWARE
  - RISC-V, MIPS
  - Registri Speciali: EPC, CAUSE, STATUS, TVAL (o BadVaddr), ...

STVEC, per memorizz. indirizzo del gestore specifico

STVal, interessa quando accedo ad un indirizzo di memoria protetto

Supporto alle gestione delle eccezioni nel RISC-V

- In RV64 ci sono vari registri speciali (a 64 bit)
  - SEPC
    contiene l'indirizzo dell'istruzione "colpevole"
  - SSTATUS
    contiene i bit di abilitazione globale degli interrupt
  - SCAUSE
    i bit 63 e [3:0] codificano le possibili sorgenti di eccezione, ad es.:
    - illegal instruction={0,2}
    - breakpoint={0,3}
    - timer interrupt={1,5}
    - external interrupt={1,9}
  - STVAL (Supervisor Trap Value)
    contiene l'indirizzo di memoria al quale si è verificato un riferimento di memoria "sbagliato" (anche chiamato "BadVAddr")
  - SIP (Supervisor Pending Interrupts)
    monitoraggio degli interrupt in attesa
  - SIE (Supervisor Interrupt Enable)
    Abilitazioni più fini per gli interrupt
  - STVEC (Supervisor Trap Vector)
    indirizzo base della lista dei «vettori di interrupt»
  - SSCRATCH: registro per salvataggi temporanei

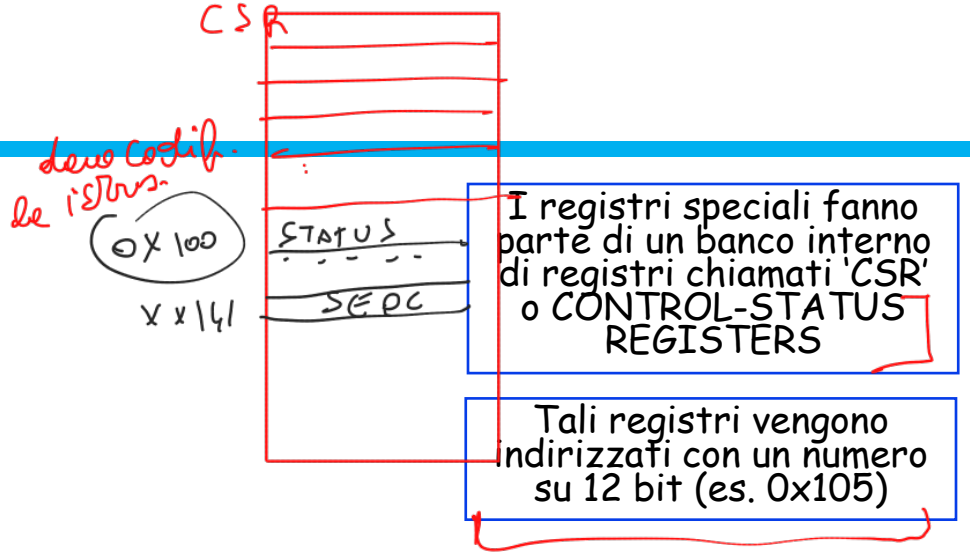
Nota: il prefisso "S" sta sempre ad indicare il modo "Supervisor"

bits 3:0
bit 63

Nota2: il bit 63 di SCAUSE specifica se si tratta di interrupt o eccezione

Dove si trovano tali registri

- STATUS registro CSR 0x100
- SEPC registro CSR 0x141
- SCAUSE registro CSR 0x142
- STVAL registro CSR 0x143
- SIP registro CSR 0x144
- SIE registro CSR 0x104
- STVEC registro CSR 0x105
- SSCRATCH registro CSR 0x140



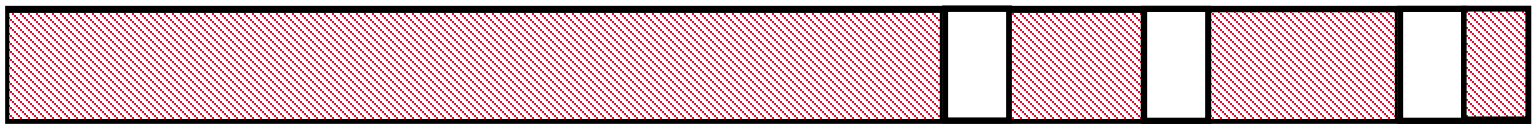
- Al verificarsi di un'eccezione, la parte di controllo della CPU modifica: SEPC, SSTATUS, SCAUSE, PC
  - Nota 1: in fase di fetch il PC (oldPC) è stato aggiornato a PC+4 (newPC). Per poter identificare l'istruzione "colpevole" è necessario far riferimento a oldPC anziché a newPC. Quindi SEPC= oldPC
  - Nota 2: in PC (newPC) viene (sovra-)scritto direttamente il nuovo valore (PC←STVEC)

Nota: nel simulatore RARS l'indirizzo della routine di gestione è modificabile

Status Register

Nota: il prefisso "S-" sta sempre ad indicare il modo "Supervisor"

SSTATUS

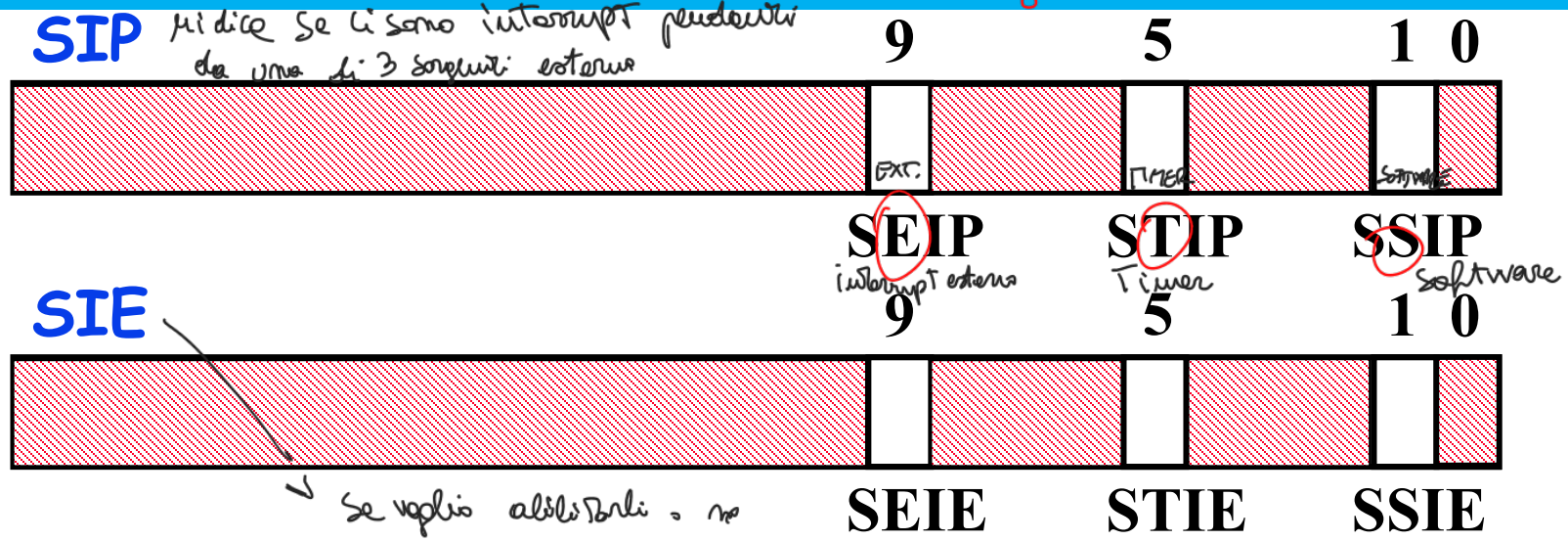


- Questi bit indicano il livello di privilegio PRECEDENTE (SPP) con il relativo valore del precedente interrupt-enable (SPIE) ed inoltre (SIE) permette l'abilitazione GLOBALE agli interrupt
- Il livello di privilegio può essere S- per Supervisor o U- per User
  - Questi sono rispettivamente codificati con i valori 1 e 0
  - Chi si trova ad un certo livello non può sapere se esistono livelli di privilegio superiori (tale informazione non è accessibile)
- SIE (S- Interrupt Enable) abilita globalmente gli interrupt a quel dato livello → per non «entrare in loop», viene subito disabilitato (0) al partire dell'eccezione
- SPIE (S- Previous Interrupt Enable) indica lo stato precedente del bit SIE al momento in cui si verifica l'eccezione
- SPP (S- Previous Privilege mode) indica il livello di privilegio precedente il momento in cui si verifica l'eccezione

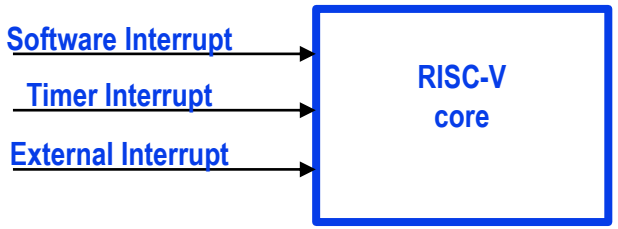
Nota: I livelli di privilegio del RISC-V sono in realtà più di questi – per un approfondimento vedere https://riscv.org/specifications/

Controllo più fine degli interrupt

Nota: il prefisso "S-" sta sempre ad indicare il modo "Supervisor"



- SxIP sono bit di indicazione di interrupt pendenti (in attesa)
- SxIE sono bit di abilitazione più fine degli interrupt
  - x si riferisce alla possibile sorgente dell'eccezione
    - x=E → interrupt Esterno
    - x=T → interrupt dal Timer
    - X=S → interrupt Software (ovvero eccezione)

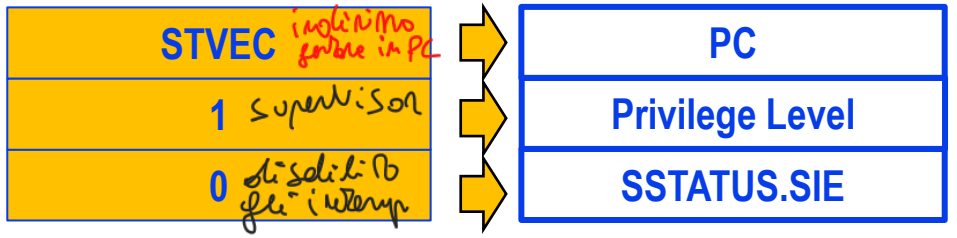


Ingresso in modalità Supervisor (e disabilitazione interrupt) e Uscita da modalità Supervisor (e ripristino)

- **Al verificarsi di un'eccezione… ho le seguenti operazioni AD HARDWARE**

1) Salvataggio dello Stato della Macchina

2) Ingresso in modalità Supervisor (per poter eseguire la routine di gestione dell'eccezione)



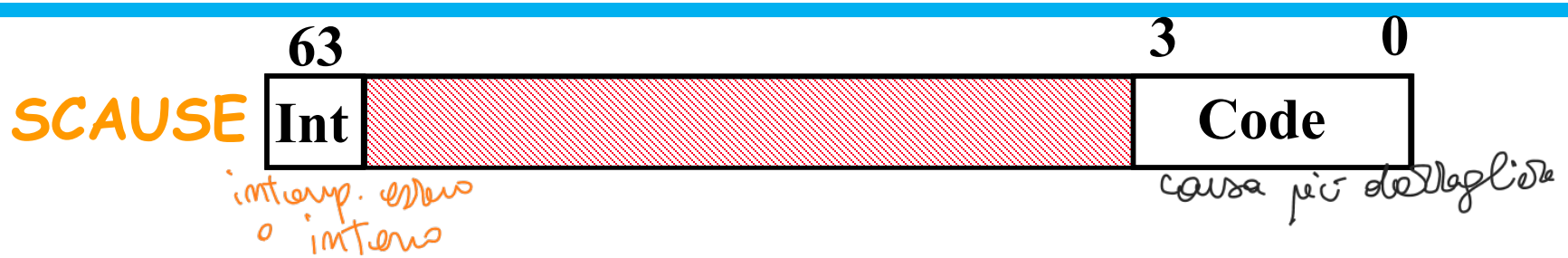
3) Eseguo la routine di gestione dell'eccezione che DEVE terminare con l'istruzione SRET

- **All'esecuzione della istruzione SRET… sempre AD HARDWARE**

4)

SCAUSE Register

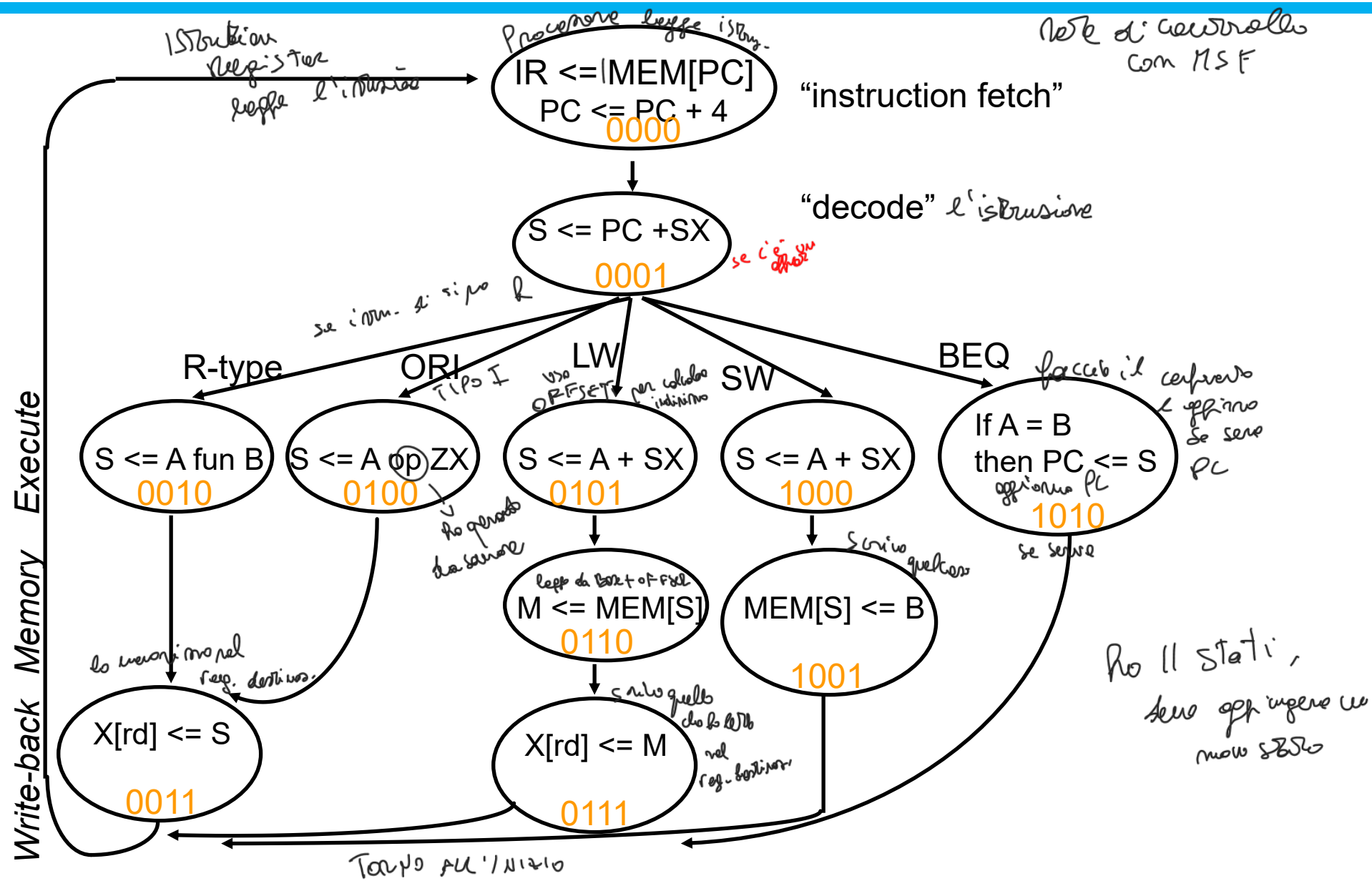
Nota: il prefisso "S-" sta sempre ad indicare il modo "Supervisor"



- Int (1 bit) se vale 1, la sorgente è un interrupt, se 0 la sorgente è un'eccezione
- Code (4 bits) codifica la ragione dell'eccezione
  - 0 → Instruction address misaligned
  - 2 → Illegal instruction
  - 3 → Breakpoint

  …altri esempi:
  - 4 → Load address misaligned
  - 5 → Load address fault
  - 6 → Store address misaligned
  - 7 → Store address fault
  - 8 → Environment call from U-mode
  - 9 → Environment call from S-mode
  - C → Instruction page fault
  - D → Load page fault
  - …

Macchina a Stati Finiti che descrive il nostro RISC-V



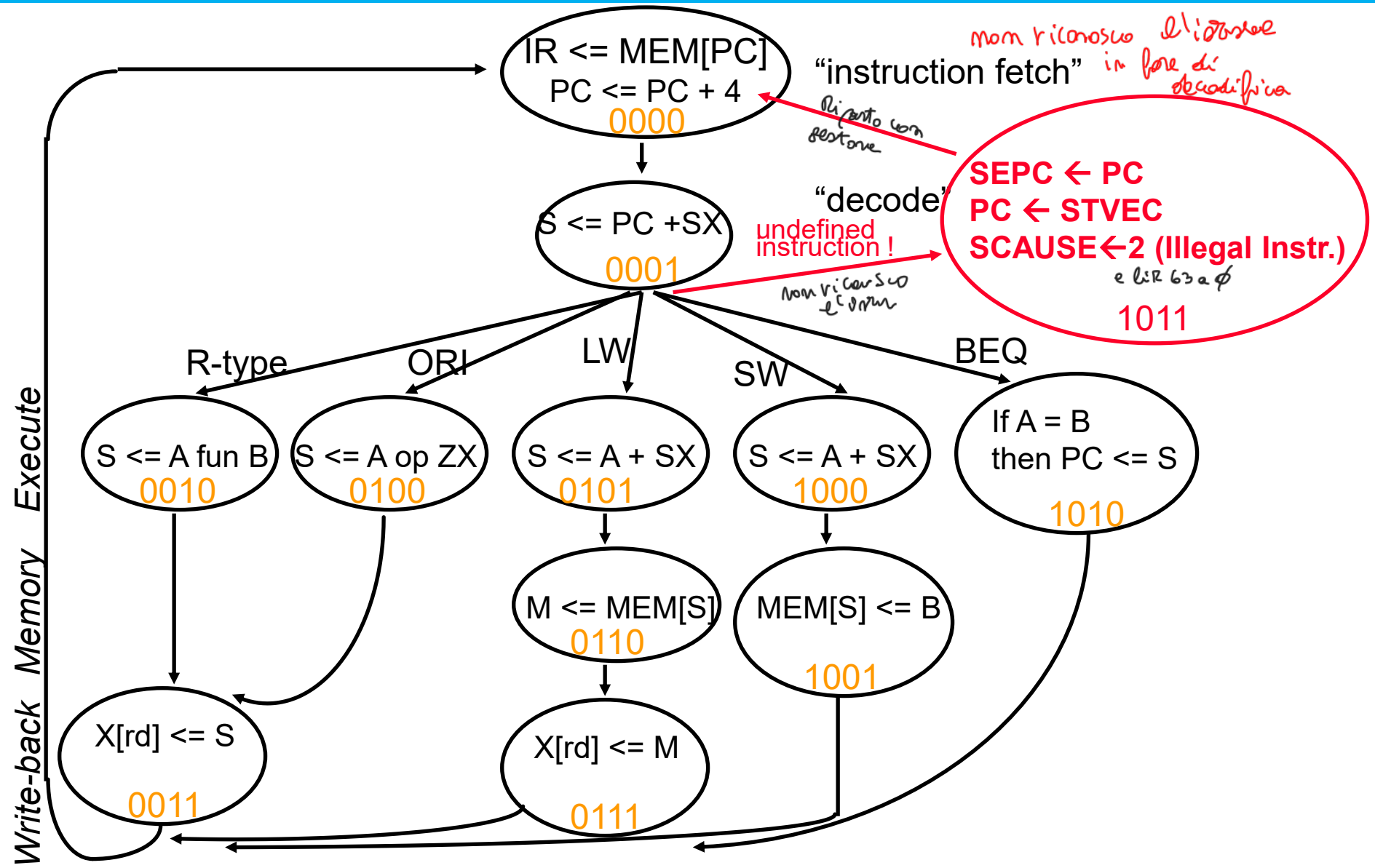
Modifica del RISC-V per poter gestire le eccezioni

- Supponiamo di partire dal processore "base" (che non gestisce eccezioni) e di voler gestire eccezioni: ad es. nel caso di «istruzione non definita»

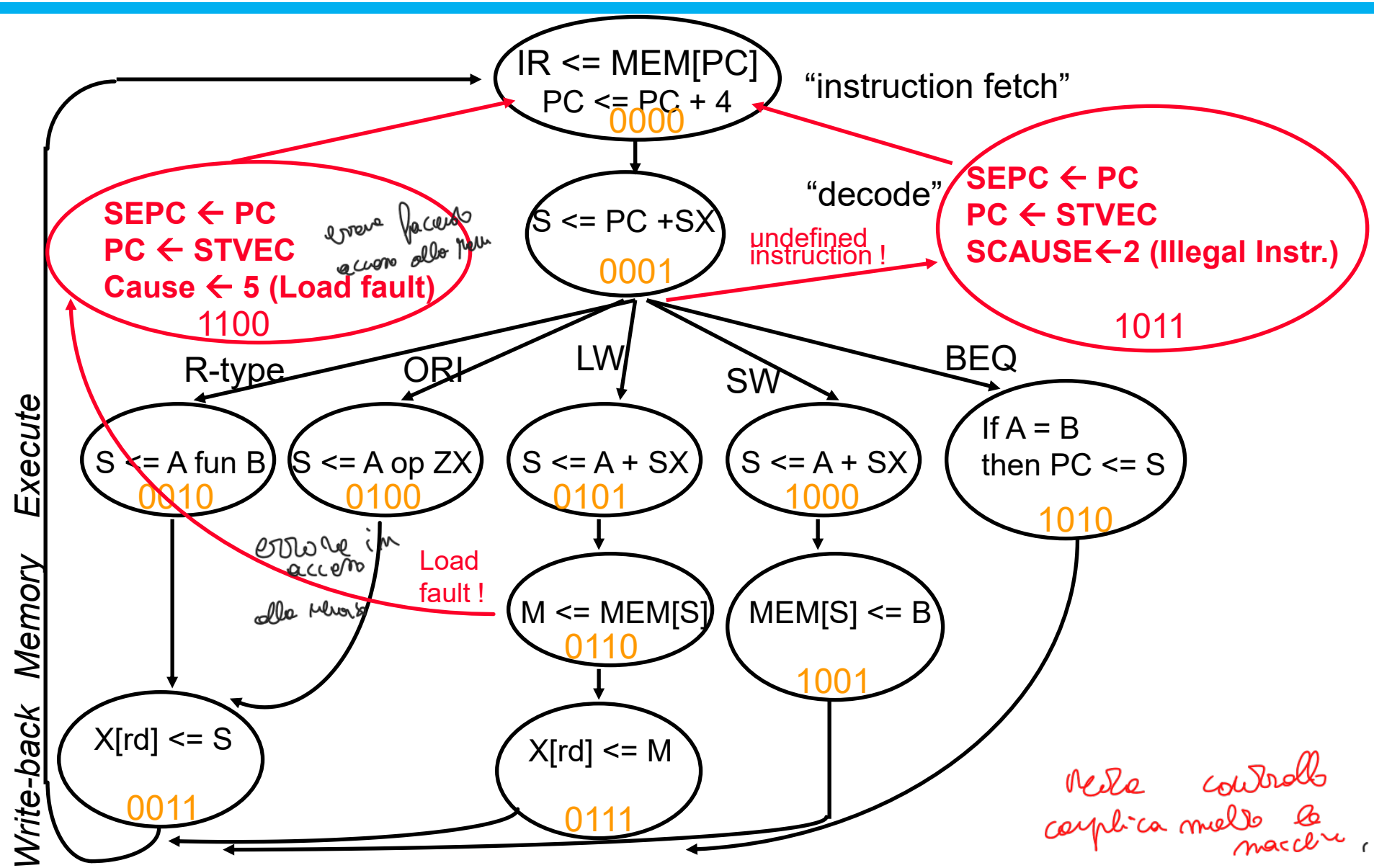
es. - non riconosco istruzione

- L'eccezione "Illegal Instruction" viene riconosciuta nel momento in cui, trovandomi nello stato 0001, non ho uno stato successivo in cui andare in quanto ho un op-code sconosciuto
  - Questo si può implementare definendo un nuovo stato (stato #11 o 1011) per tutti gli op-code diversi da lw, sw, (R-type), jmp, beq, e ori

Implementazione del meccanismo di eccezione per una istruzione non definita



Aggiunta di eccezione per «Load address fault»: stato#12

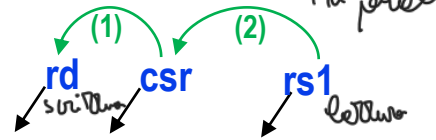


Lettura/Scrittura di SEPC, SCAUSE, STVEC, STVAL

- **Per modificare i contenuti dei registri speciali CSR:**
  - `CSRRW` CSR Read+Write
  - `CSRRS` CSR Read+Set
  - `CSRRC` CSR Read+Clear
  - `CSRRWI` CSRRW Immediate
  - `CSRRSI` CSRRW Immediate
  - `CSRRCI` CSRRW Immediate

Nota: queste istruzioni sono Codificate con il formato I: i 12 bit del campo IMM12 sono usati per indicizzare un registro CSR (es. sstatus= 0x100); la variante immediata (es. CSRRWI) sfrutta quindi il campo RS1 per codificare un immediato a 5 bit.

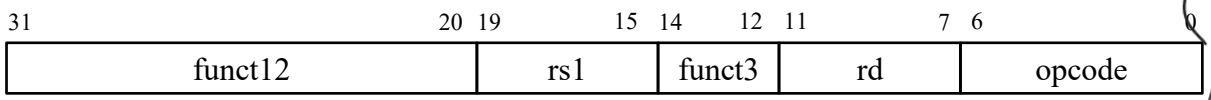
- **Esempi**




```
CSRRW   t0,stvec,t1     # carica in t0 stvec e lo aggiorna con t1
CSRRSI  t0,sie,2        # t0←sie e mette a 1 il bit-1 di sie (software int.)
CSRRSI  t0,sie,512      # NON possibile (l'imm. deve stare su 5 bit)
CSSRC   x0,sie,t1       # (t1=2^9+2^5=544) azzera i bit 9 e 5 di sie
                        (disabilita interrupt esterni e interrupt timer)
CSRRS   t0,sstatus,x0   # carica in t0 il contenuto di sstatus, sstatus non
                        viene modificato (caso particolare, essendo rs1=x0)
CSRRW   x0,sscratch,a0  # salva a0 nel reg. speciale sscratch
```

6'11 $(2)_{10} = (00\ldots010)_2$

Ricapitolazione nuove istruzioni e loro codifica



CSRRW/CSSRWI	csr	rs1	1/5	rd	0x73
CSRRS/CSRRSI	csr	rs1	2/6	rd	0x73
CSRRC/CSRRCI	csr	rs1	3/7	rd	0x73

Nota: "csr" è l'indirizzo del registro CSR su cui si opera (es. 0x142 per SCAUSE)

- Per completezza:

ECALL	0x000	0	0	0	0x73
SRET	0x102	0	0	0	0x73
EBREAK	0x001	0	0	0	0x73

- Ed inoltre, si può richiedere al processore di fermarsi per attendere un interrupt (anziché effettuare un ciclo di attesa attiva) con l'istruzione WFI (Wait For Interrupt)

WFI	0x105	0	0	0	0x73

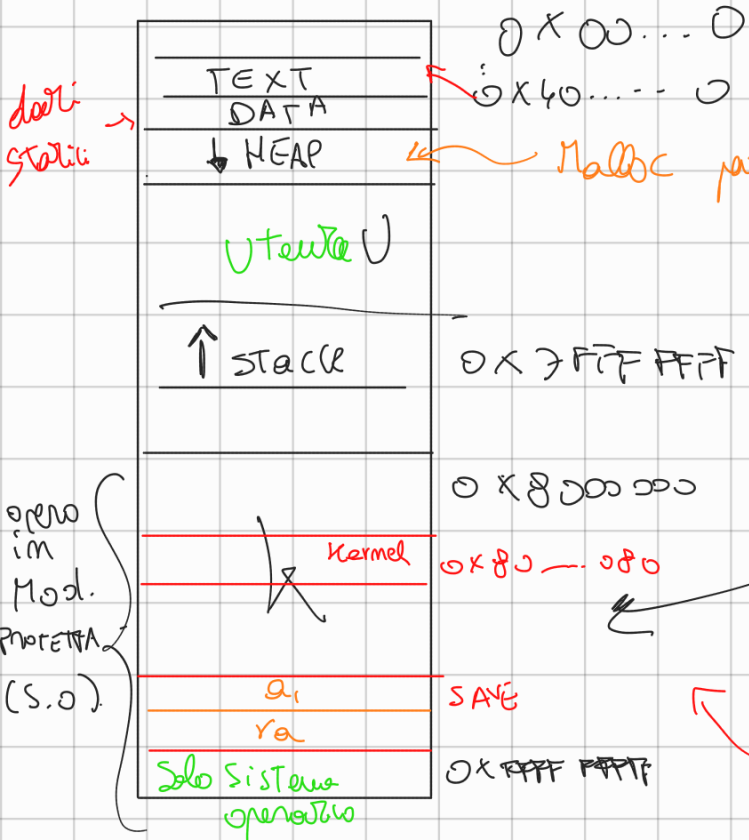
Nota: vedremo come la WFI sarà utile nella gestione dell'Input/Output del Calcolatore

Esempio di Gestore delle Eccezioni

```
.text 0x80000080
CSRRW x0,sscratch,a0   # salva a0 nel reg. speciale sscratch
LA    a0,save          # salvo anche a1,ra -- ipotesi: gestore non rientrante
SD    a1,0(a0)         # necessario evitare uso stack (può essere la causa...)
SD    ra,8(a0)         # salvo ra perché dopo faccio una jal

CSRRW a1,scause,x0     # leggo la causa dell'eccezione
MV    a0,a1            # copio a1 in a0 (se il bit 63 è 0 → indica eccezione)
SRLI  a1,a1,63         # isolo il bit 63 (shift right logical)
BNE   a1,x0,fatto      # ignoro le interruzioni (ma non le eccezioni)

CSRRW a1,SEPC,x0       # carico il PC a cui si è verificata l'eccezione
JAL   gestione         # routine di gestione eccezione, a0=causa,a1=SEPC

fatto:
LA    a0,save          # ripristino i registri a0,a1,ra
LD    a1,0(a0)
LD    ra,8(a0)
CSRRW a0,sscratch,x0   # carico in a0 il valore salvato inizialmente in sscratch

SRET                   # PC←SEPC, ripristino modalità privilegiata,
                       # ripristino stato int.

.data 0x81000000
save: .space 16        # riservo una zona di 16 byte per salvare i registri
```


Nota: questo non è il gestore più efficiente possibile ma solo un esempio di come potrebbe essere strutturato il Gestore delle Eccezioni

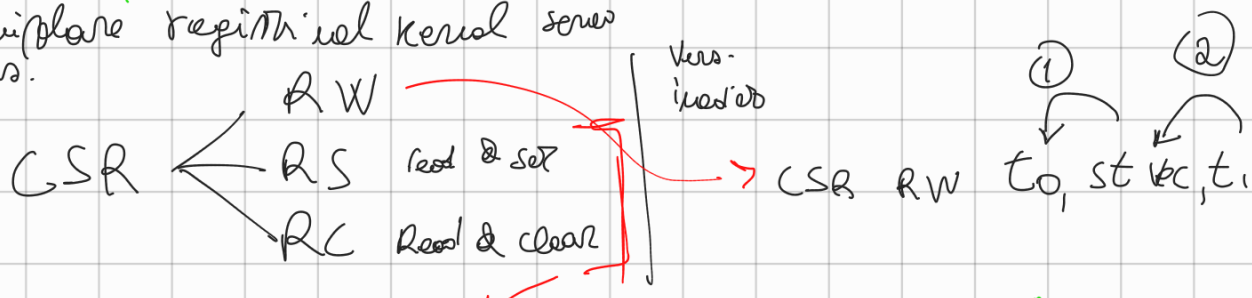
locazione nello spazio di indirizzo, diviso in 2 parti: spazio kernel e spazio utente



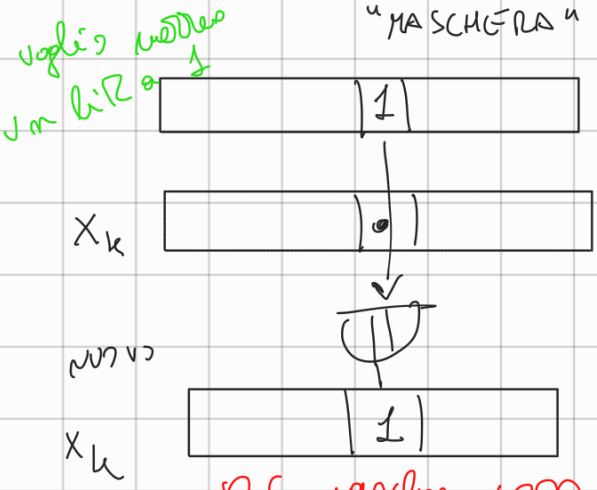
Passo poi il mio gestore di interruzioni

Solo un processo alla volta può accedere

per manipolare registri nel kernel servono 3 operazioni.



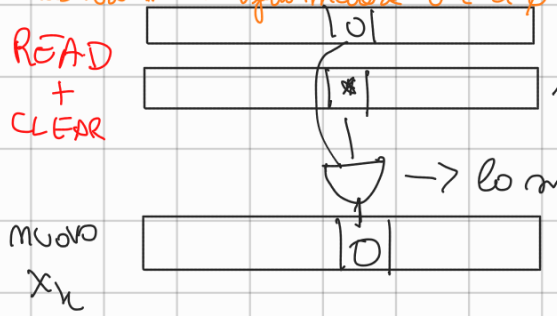
maschera viene messa in AND o OR



RS maschera con un OR

Riporto un altro registro X generico

maschera in cui voglio mettere bit a 0

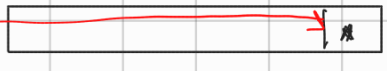


RC mette in AND

vale anche per invediato, per leggere e scrivere RS e RC

Formato Scause

Shiftot verso destra per isolare il 63



a1

Gestore, devo salvare contesto se uso registri x devo salvarli, perché qualunque istruz. faccio, di fatto, devo salvare (in SSCRATCH)