CALCOLATORI
I/O

Giovanni Iacca
giovanni.iacca@unitn.it

Lezione basata su materiale preparato con i Prof. Luigi Palopoli e Marco Roveri

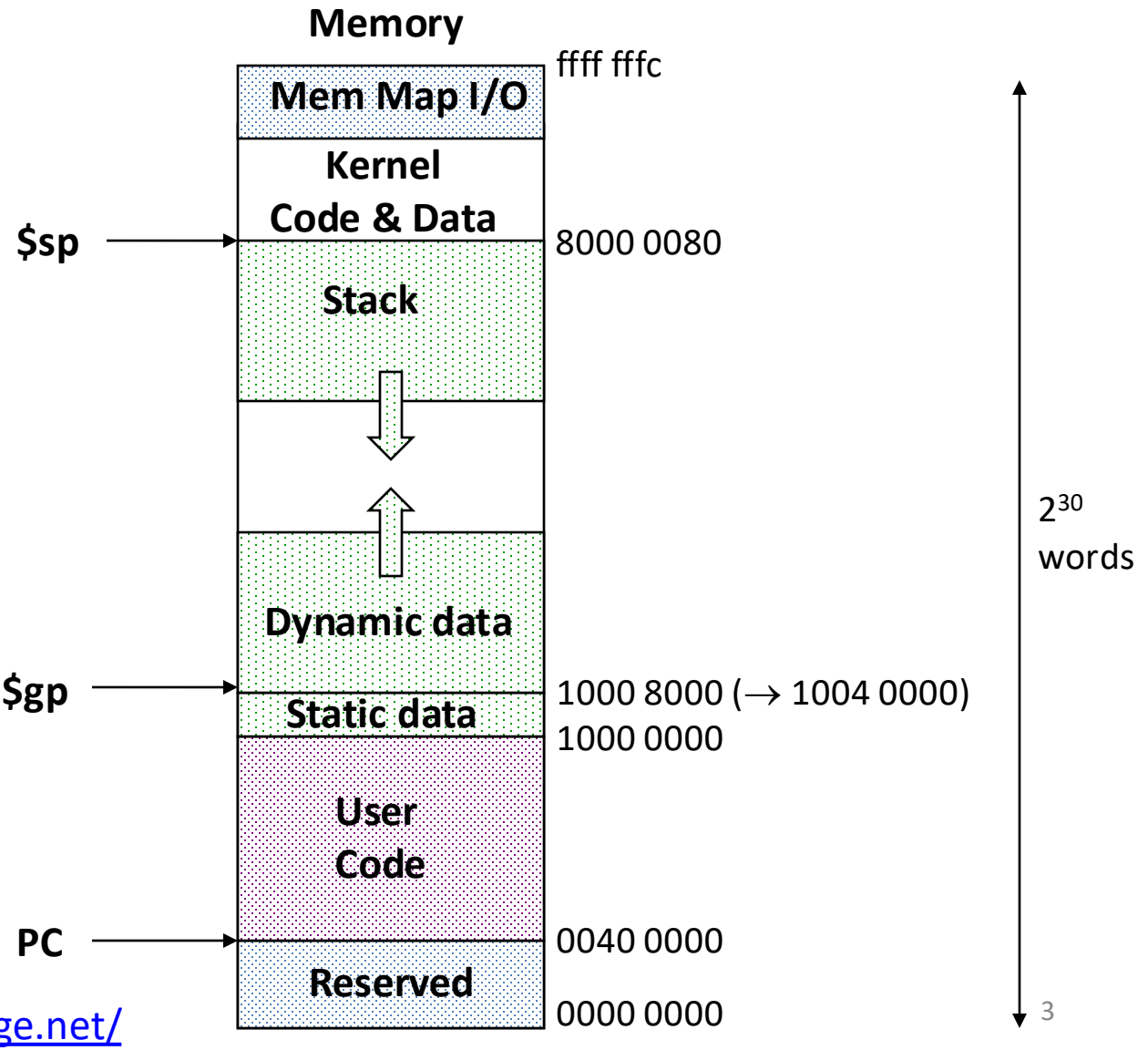
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Considerazioni sul polling

- L'attesa attiva fa perdere tempo al processore che dedica cicli macchina a letture inutili
- Il polling può essere usato quando le operazioni di I/O avvengono con velocità di trasferimento predeterminata (es. applicazioni di controllo) e comunque il processore ha poco altro da fare
- Sicuramente se i dati vengono trasferiti con elevati *bitrate* il ciclo di attesa attiva dura poco
- In altri casi lo spreco è inaccettabile e per questo motivo è stato inventato l'I/O a interruzione di programma (interrupt driven I/O)

Organizzazione memoria SPIM (MIPS)

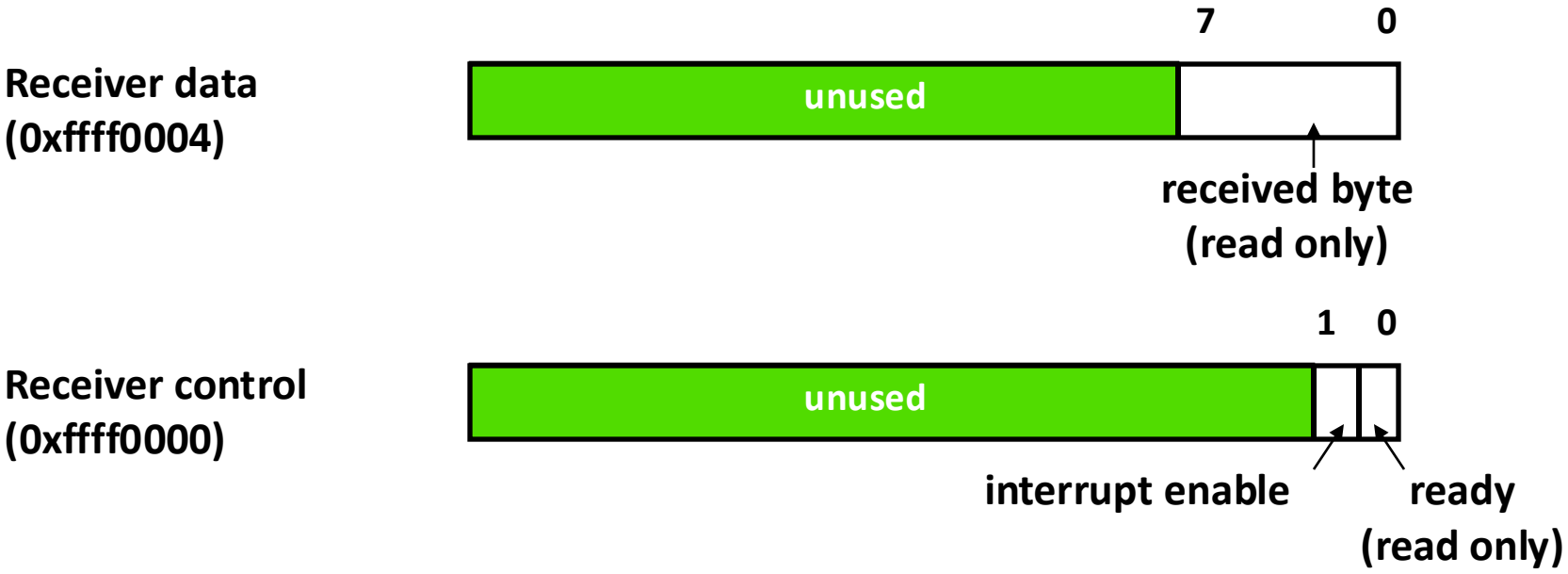
- Partiamo da una tipica organizzazione di memoria (ad es. quella dell'emulatore SPIM)



http://spimsimulator.sourceforge.net/

Controllo del terminale in SPIM (INPUT)

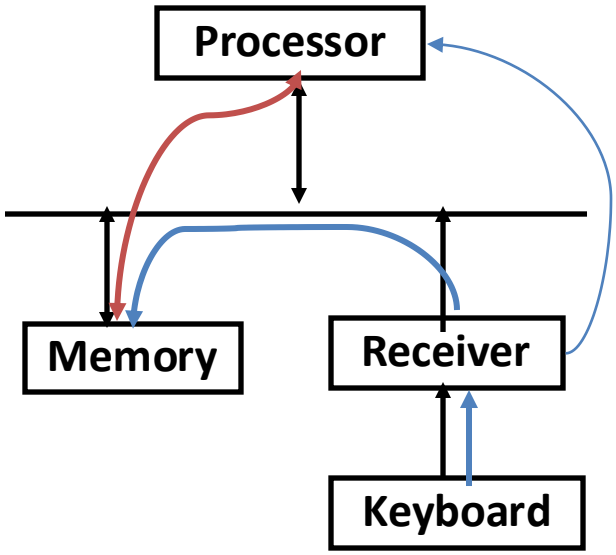
- Osserviamo da vicino le locazioni per il controllo del terminale (INPUT)

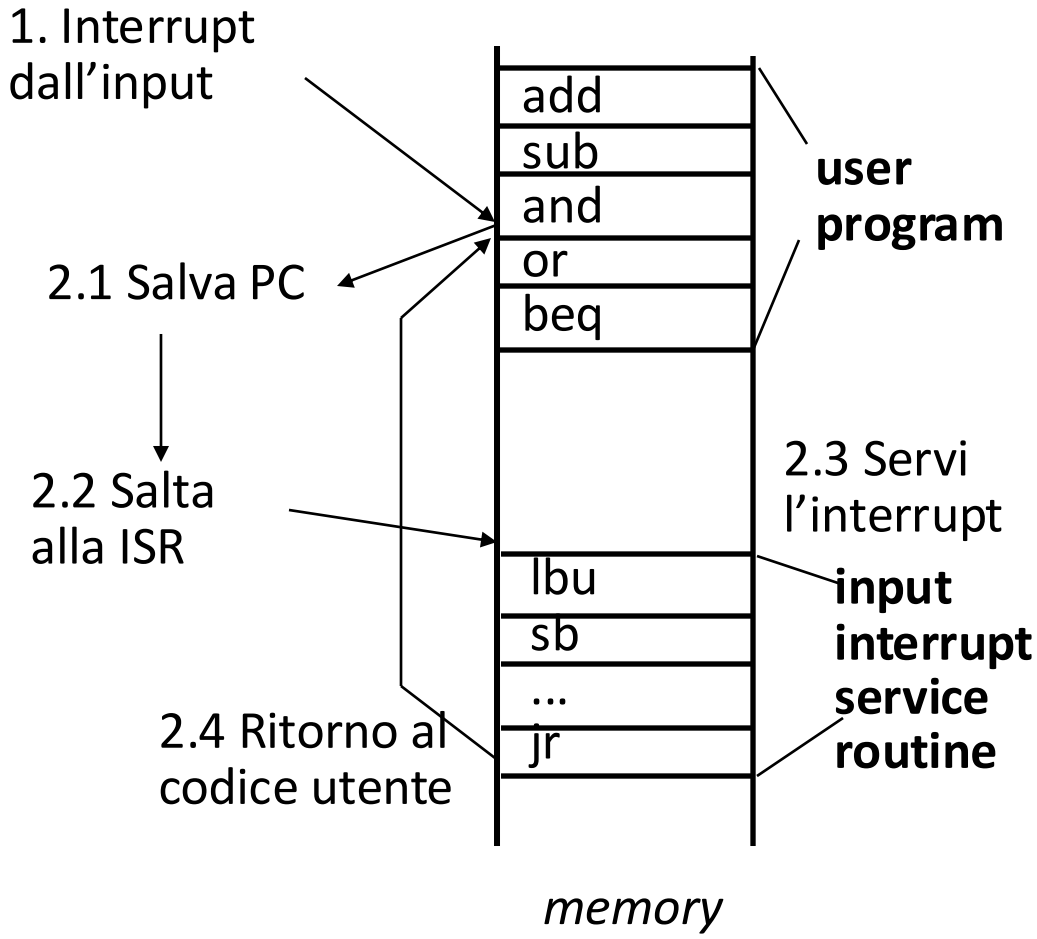


Interruzioni di programma

- Un'interruzione I/O è un segnale usato per segnalare al processore che la periferica è pronta ad eseguire il trasferimento richiesto
  - Le interruzioni possono avere diverso grado di urgenza (è possibile definire priorità).
  - Occorre un modo per segnalare al processore quale periferica richiede l'interruzione.
- Le interruzioni I/O sono sempre asincrone rispetto all'esecuzione delle istruzioni
  - Non esistono particolari istruzioni assembly per eseguire le interruzioni. Un'interruzione può arrivare mentre una qualsiasi istruzione viene eseguita, e dà comunque modo di terminare l'esecuzione dell'istruzione.
    - ✓ Il programmatore può spesso differire l'esecuzione dell'interruzione a un momento più conveniente (es. sezioni non interrompibili nel codice del kernel).
- Vantaggio
  - Non occorre interrompere l'esecuzione del programma se non quando il dato può essere effettivamente riferito in memoria.
- Svantaggio – occorre un hardware speciale per:
  - Permettere ai dispositivi di I/O di generare un'interruzione.
  - Rilevare l'interruzione, salvare lo stato del processore per eseguire una particolare routine di servizio (Interrupt Service Routine, ISR) e poi riprendere dal punto dove si era interrotto.

Input a interruzione di programma

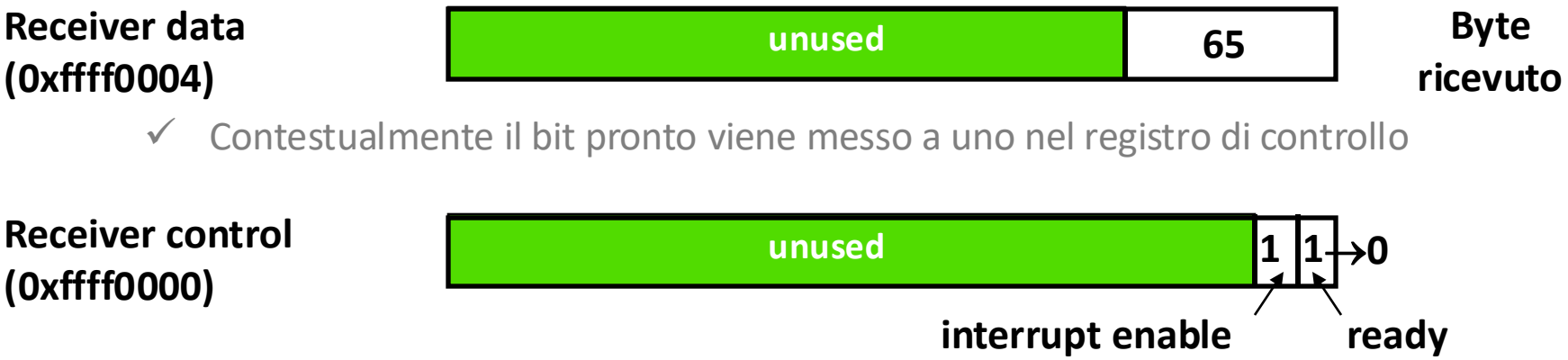




Nel mezzo c'è una commutazione in modalità «Supervisore» (solo il SO può gestire l'interruzione)

Esempio controllo terminale SPIM

1. La periferica indica con un'interruzione che ha un nuovo carattere dalla tastiera nell'opportuno registro di ricezione



   - ✓ Contestualmente il bit pronto viene messo a uno nel registro di controllo

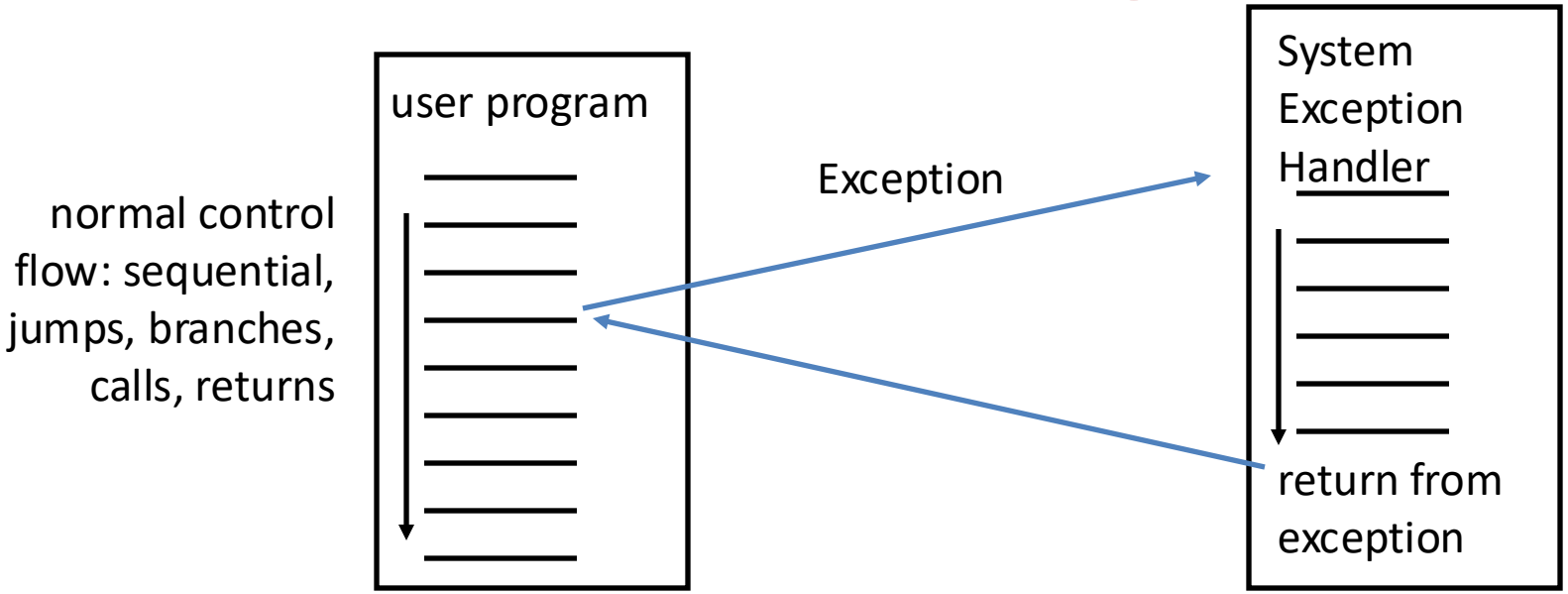
2. Il processo utente viene interrotto trasferendo il controllo a una ISR che copia il dato in memoria utente
   - ✓ All'atto della lettura il bit ready viene ri-azzerato
   - ✓ Notare che prima di effettuare il trasferimento il bit interrupt enable era stato posto a 1 per abilitare le interruzioni

Che ci si guadagna?

- Ritorniamo all'esempio di prima dell'hard disk e supponiamo che un interrupt costi 500 cicli di clock (plausibile che costi di più del polling)
- Se le interruzioni vengono generate alla frequenza di polling
  - Disk Interrupts/sec = 8 MB/s /16B
    = 500K interrupts/sec
  - Disk Interrupt Clocks/sec = 500K * 500
    = 250,000,000 clocks/sec
  - % Processor: $250*10^6/500*10^6$= 50%
  - Sembrerebbe che non ci sia guadagno…anzi
- Tuttavia se l'hard disk è attivo solo per il 5% del tempo, gli interrupt generati saranno il 5% e la spesa di processore sarà:
  5% * 50%=2.5%

L'overhead si paga solo quando vengono effettivamente generate richieste

Eccezioni in generale



- Eccezione = trasferimento (non programmato) del controllo del programma
  - Il sistema effettua delle azioni per gestire le eccezioni
    - ✓ Ad esempio deve sapere dove registrare il punto di interruzione e dove salvare lo stato, e poi (a eccezione finita) come riprendere dal punto immediatamente successivo al punto di interruzione

Tre tipi di eccezioni

- Interrupts
  - Causate da eventi esterni (I/O)
  - Asincrone
  - Possono essere gestite nello spazio tra due istruzioni
  - Semplicemente sospendono il programma e riprendono dal punto in cui era stato interrotto

- Traps (Eccezioni)
  - Causate da eventi interni al programma
    - ✓ Condizioni eccezionali (ad es. arithmetic overflow, undefined instr.)
    - ✓ Errori (ad es. hardware malfunction, memory parity error, segmentation fault)
    - ✓ Fault (ad es. non-resident page – page fault)
  - Sincrone all'esecuzione del programma
  - Gestite da un trap handler
  - E' possibile riprovare ad eseguire l'istruzione che ha causato l'eccezione o abortire il programma

- Environment call/break
  - La environment call (istruzione ecall) è causata da una esplicita richiesta di un servizio di sistema (stampa di un carattere, un intero, …) attraverso esplicita ecall.
  - La environment break (istruzione ebreak) è causata da una esplicita chiamata a ebreak per motivi diagnostici o di debug (ad es. la break nel GDB)

Gestione delle eccezioni
Exception Handler

- Esistono vari metodi per gestire le eccezioni
  - Salto diretto all'indirizzo della routine di gestione
  - Vettore di interruzione

- In entrambi gli approcci lo stato della macchina deve essere preservato
  - Ad es. salvando il contenuto dei registri nello stack o in appositi registri

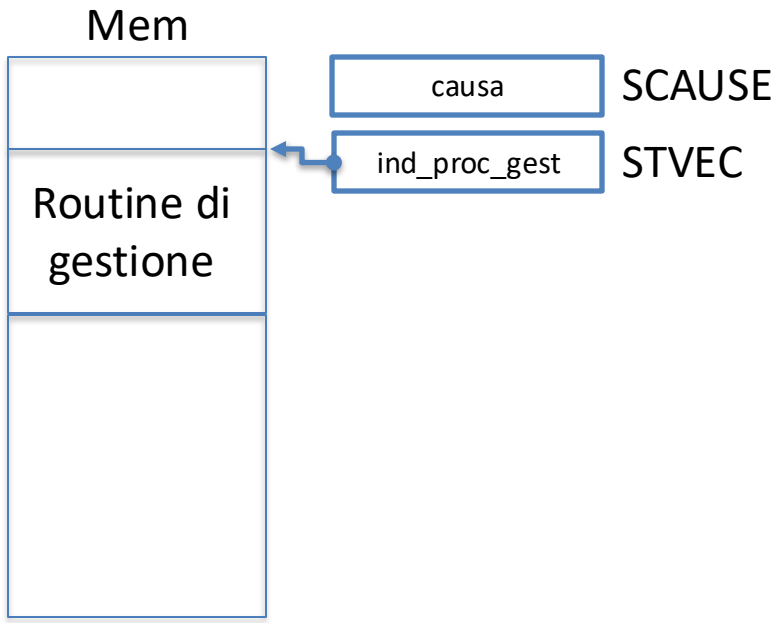
Salto diretto ad indirizzo della routine di gestione

- Salto diretto ad un indirizzo specifico:

  PC <- ind_proc_gest

  - Nel RISC-V
    - ✓ Indirizzo di gestione è contenuto nel registro speciale STVEC
    - ✓ La causa dell'eccezione è memorizzata in registro speciale SCAUSE

- Vantaggi:
  - Non è necessario fare un accesso in memoria per prelevare l'indirizzo della routine di gestione
- Svantaggi
  - Nella routine di gestione occorre analizzare la causa dell'eccezione

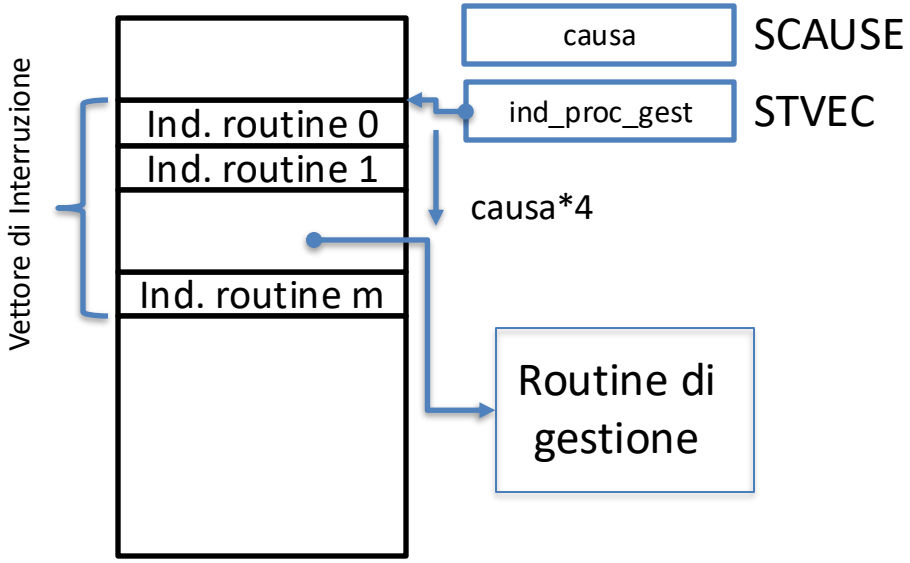


Vettore di interruzione

- Memorizzo in una tabella gli indirizzi delle Routine di Gestione per ogni possible causa:

  PC <- Mem[base + causa*4]

- Nel RISC-V
  - Indirizzo base delle routine di gestione delle eccezioni contenuto in registro speciale STVEC
  - La causa dell'eccezione è memorizzata in registro speciale SCAUSE
- Vantaggi
  - La causa dell'eccezione è nota ed utilizzata per identificare la routine relativa di gestione
- Svantaggi
  - E' necessario accedere alla memoria per prelevare l'indirizzo della routine di gestione



Salvataggio dello stato della macchina al verificarsi di una eccezione

- Vari approcci
  - Salvataggio sullo stack
  - Salvataggio in registri ausiliari (sia visibili che non)
  - Salvataggio in registri speciali

- Nel RISC-V
  - Salvataggio sullo stack e su registri speciali
    - ✓SEPC, SCAUSE, SSTATUS, STVAL (o BadVaddr), …

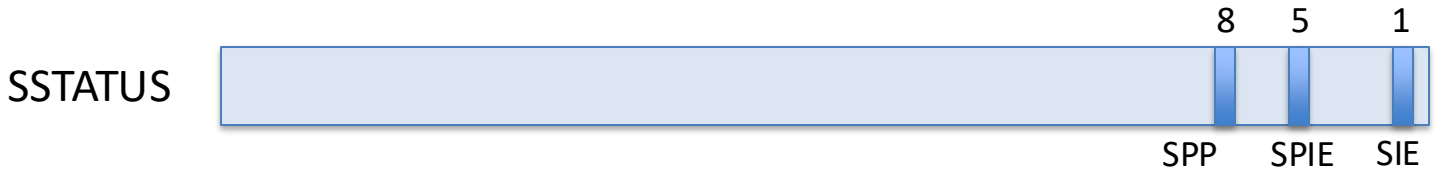
Supporto alla gestione delle eccezioni nel RISC-V

- Il RISC-V ha vari registri a 64 bit
  - SEPC - contiene indirizzo dell'istruzione che ha generato l'eccezione
  - SSTATUS - contiene i bit di abilitazione globale degli interrupt
  - SCAUSE - I bit 63 e [3-0] codificano le possibili sorgenti di eccezione
    - ✓ Illegal instruction = {0, 2}
    - ✓ Breakpoint = {0, 3}
    - ✓ Time interrupt = {1, 5}
    - ✓ External interrupt = {1,9}
    - ✓ …
  - STVAL – Supervisor Trap Value – contiene l'indirizzo al quale si è verificato un riferimento errato alla memoria
  - SIE – Supervisor Interrupt Enable – specifica abilitazioni più fini degli interrupt in attesa
  - SIP – Supervisor Interrupt Pending – monitoraggio degli interrupt in attesa
  - STVEC – Supervisor Trap Vector – Indirizzo base della lista dei vettori di interrupt
  - SSCRATCH – Registro per salvataggi temporanei

- Questi registri si trovano in un banco interno al processore chiamato Control Status Register (CSR)

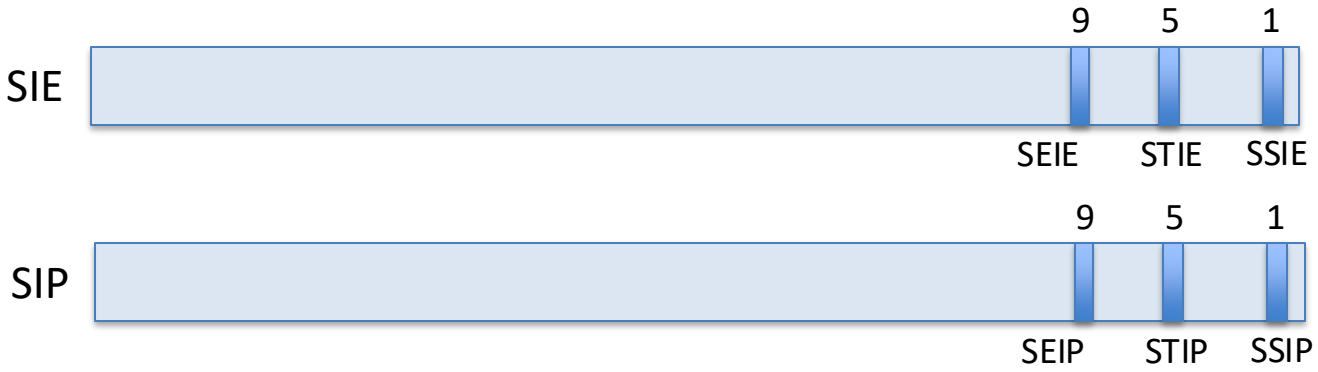
- Al verificarsi di una eccezione la parte di controllo della CPU modifica questi registri
  - Note:
    - ✓ In fase di fetch il PC è aggiornato a PC+4. L'istruzione «colpevole» da memorizzare in SEPC è quindi il PC prima dell'incremento
    - ✓ In PC viene (sovra-)scritto direttamente il nuovo valore (PC <- STVEC)

Status Register



- 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)
- 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 al momento in cui si verifica l'eccezione

Controllo fine degli interrupt

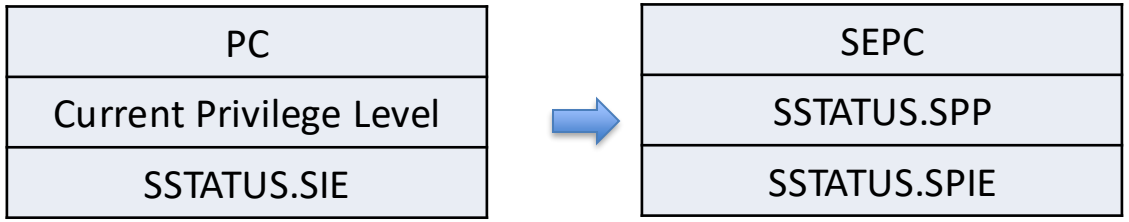


- SxIE sono bit di abilitazione più fine degli interrupt
- SxIP sono bit di indicazione di interrupt pendenti (in attesa)

- x si riferisce alla possibile sorgente dell'eccezione
  - x = E → interrupt esterno
  - x = T → interrupt dal timer
  - x = S → interrupt software (ovvero eccezione)

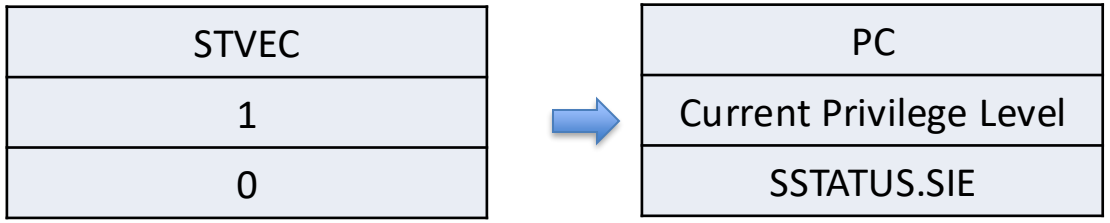
Ingresso ed Uscita in modalità Supervisor

- Al verificarsi di una eccezione



Salvataggio dello stato della macchina: valori correnti nei rispettivi registri dei valori precedenti

E poi…

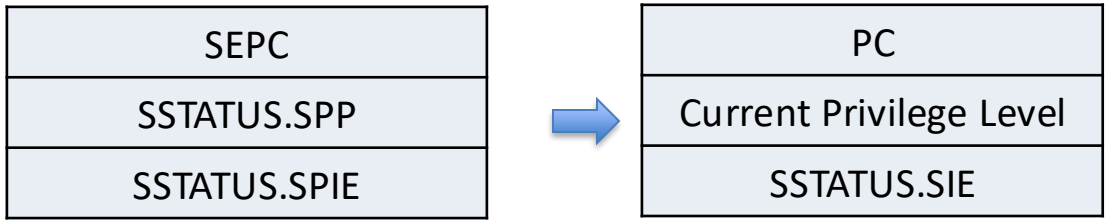


Salto alla routine di gestione

Modalità supervisor

Interrupt disabilitati

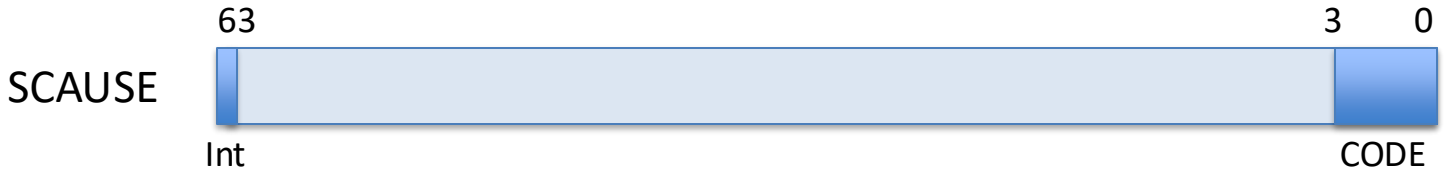
- All'esecuzione della SRET[1]



Ripristino dello stato della macchine: valori precedenti nei rispettivi registri dei valori correnti

[1]Tipicamente si usa SRET per ritornare da una routine di gestione interrupt o eccezione

SCAUSE Register



- Int (1 bit) se vale 1, la sorgente è un interrupt, se vale 0 la sorgente è una eccezione

- Code (4 bits) codifica la ragione dell'eccezione
  - 0 - Instruction address misaligned
  - 2 - Illegal instruction
  - 3 - Breakpoint
  - 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
  - …

Lettura/Scrittura di SEPC, SCAUSE, STVEC, STVAL, …

- Per modificare i contenuti dei registri speciali CSR:
  - CSR Atomic Read & Write (/Immediate): CSRRW rd, csr, rs CSRRWI rd, csr, imm
  - CSR Atomic Read & Set bits (/Immediate): CSRRS rd, csr, rs CSRRSI rd, csr, imm
  - CSR Atomic Read & Clear bits (/Immediate): CSRRC rd, csr, rs CSRRCI rd, csr, imm

- Esempi

```
CSRRW t0,stvec,t1      # t0 ← stvec, stvec ← t1
CSRRW x0,sscratch,t1   # sscratch ← t1          (in questo caso rd = x0: CSR non letto)

CSRRS t0,sstatus,t1    # t0 ← sstatus, sstatus ← sstatus OR t1
CSRRS t0,sstatus,x0    # t0 ← sstatus           (in questo caso rs = x0: CSR non scritto)

CSRRSI t0,sie,2        # t0 ← sie e mette a 1 il bit-1 (2^1) di sie (interrupt software)
CSRRSI t0,sie,512      # NON possibile (l'immediato deve stare su 5 bit)

CSRRC t0,sie,t1        # t0 ← sie, sie ← sie AND NOT(t1)
                       # es. se t1=2^9+2^5=544, azzera i bit 9 e 5 di sie
                       # (disabilita interrupt esterni e interrupt timer)
```

- Nota
  - Se il vecchio valore di CSR non serve, CSRRS/CSRRSI e CSRRC/CSRRCI vengono chiamate con rd=x0

Supporto del RISC-V per la gestione delle eccezioni

- I soli tipi di eccezioni che possono essere generate nell'implementazione della CPU analizzata sono:
  - Esecuzione di una istruzione non valida
  - Malfunzionamenti hardware
- In caso di eccezione il processore deve:
  - Salvare l'indirizzo dell'istruzione che ha generato l'eccezione nel *registro program counter dell'eccezione* (*SEPC - supervision exception program counter register*)
  - Salvare la causa dell'eccezione nel *registro causa dell'eccezione* (*SCAUSE – supervisor cause exception register*)
  - Trasferire il controllo ad un indirizzo specifico del sistema operativo per gestire l'eccezione
  - Terminata la gestione ripristinare lo stato del processore e ritornare all'esecuzione precedente (se possibile)
- Le eccezioni sono trattate nel RISC-V come se fossero degli hazard sul controllo

Supporto del RISC-V per la gestione delle eccezioni (cont.)

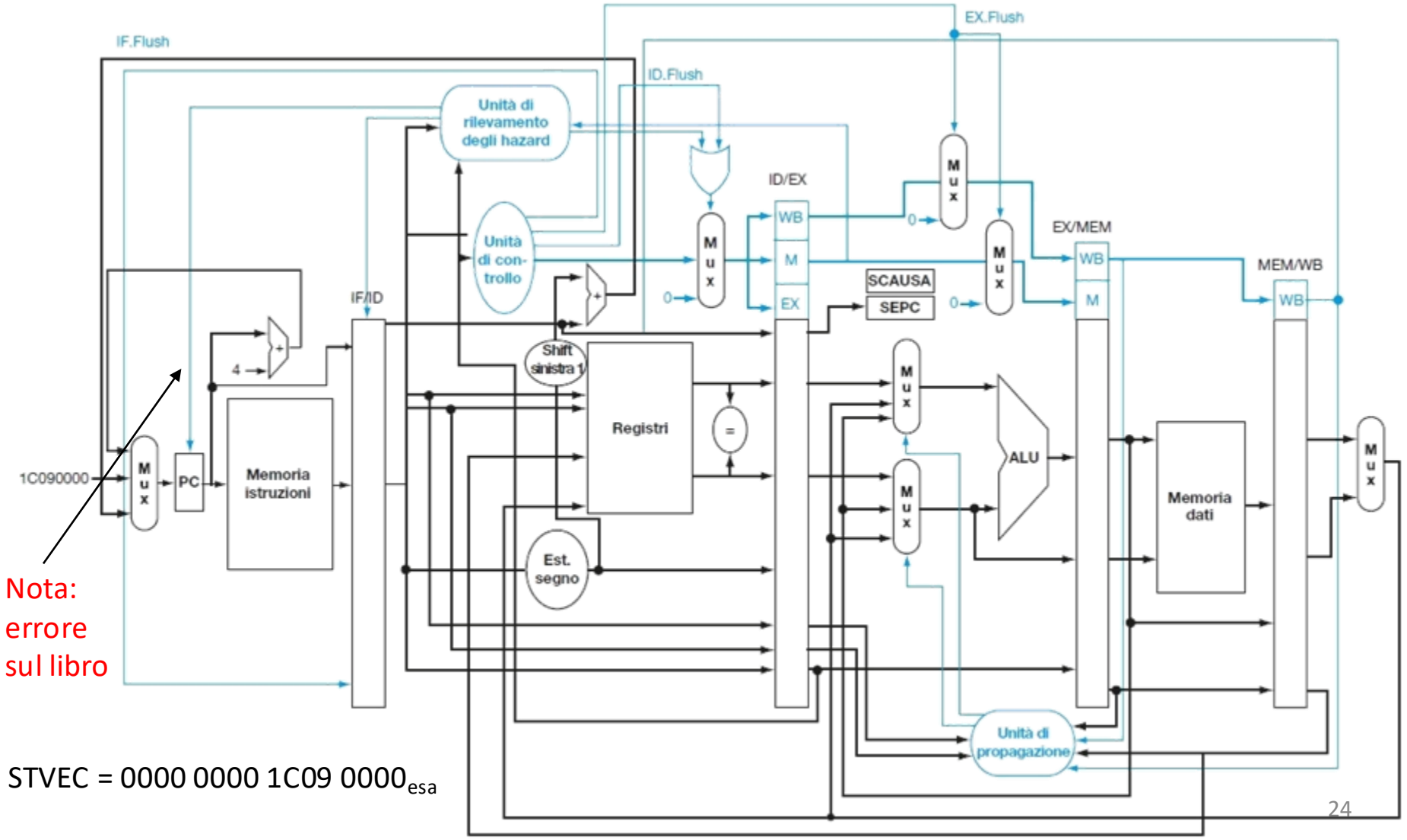
- Per la gestione di una eccezione in IF si usa lo stesso meccanismo di gestione degli errori di predizione per i salti (convertendo l'istruzione in una nop)
- Introduciamo un nuovo segnale di controllo ID.Flush, messo in OR con il segnale di stallo che proviene dall'unità rilevamento hazard, in modo da eliminare l'istruzione dallo stadio ID e realizzare così uno stallo
- Introduciamo un nuovo segnale EX.Flush, che pilota un nuovo multiplexer per mettere a 0 i segnali di controllo di questo stadio
- Assumendo che l'indirizzo della prima istruzione del codice di gestione delle eccezioni sia $0000\ 0000\ 1C09\ 0000_{esa}$, per saltare al codice di gestione dell'eccezione occorre aggiungere una linea che porta il valore $0000\ 0000\ 1C09\ 0000_{esa}$ al multiplexer che seleziona il nuovo valore del PC
- Salviamo infine l'indirizzo dell'istruzione che ha causato l'eccezione nel registro SEPC

Supporto del RISC-V per la gestione delle eccezioni (cont.)

- Problema:
  - Se non si interrompe l'istruzione prima della fine della sua esecuzione non sarà più possibile vedere il valore originale del registro di destinazione (ad es. x1, che potrebbe essere «sporcato» dalla destinazione dell'istruzione che ha generato eccezione)
    - ✓ Se supponiamo che l'eccezione venga riconosciuta nello stadio EX si può utilizzare EX.Flush per prevenire che l'istruzione scriva il registro destinazione nello stadio WB

- Nota:
  - Molte eccezioni richiedono di completare normalmente l'esecuzione dell'istruzione che ha causato l'eccezione
    - ✓ Il modo più semplice per ottenere ciò consiste nell'eliminare l'istruzione e farla eventualmente ripartire dall'inizio dopo che l'eccezione è stata gestita

Supporto del RISC-V per la gestione delle eccezioni (cont.)



Nota: errore sul libro

STVEC = 0000 0000 1C09 0000_{esa}