ONE PIECE

3 - ENIES LOBBY

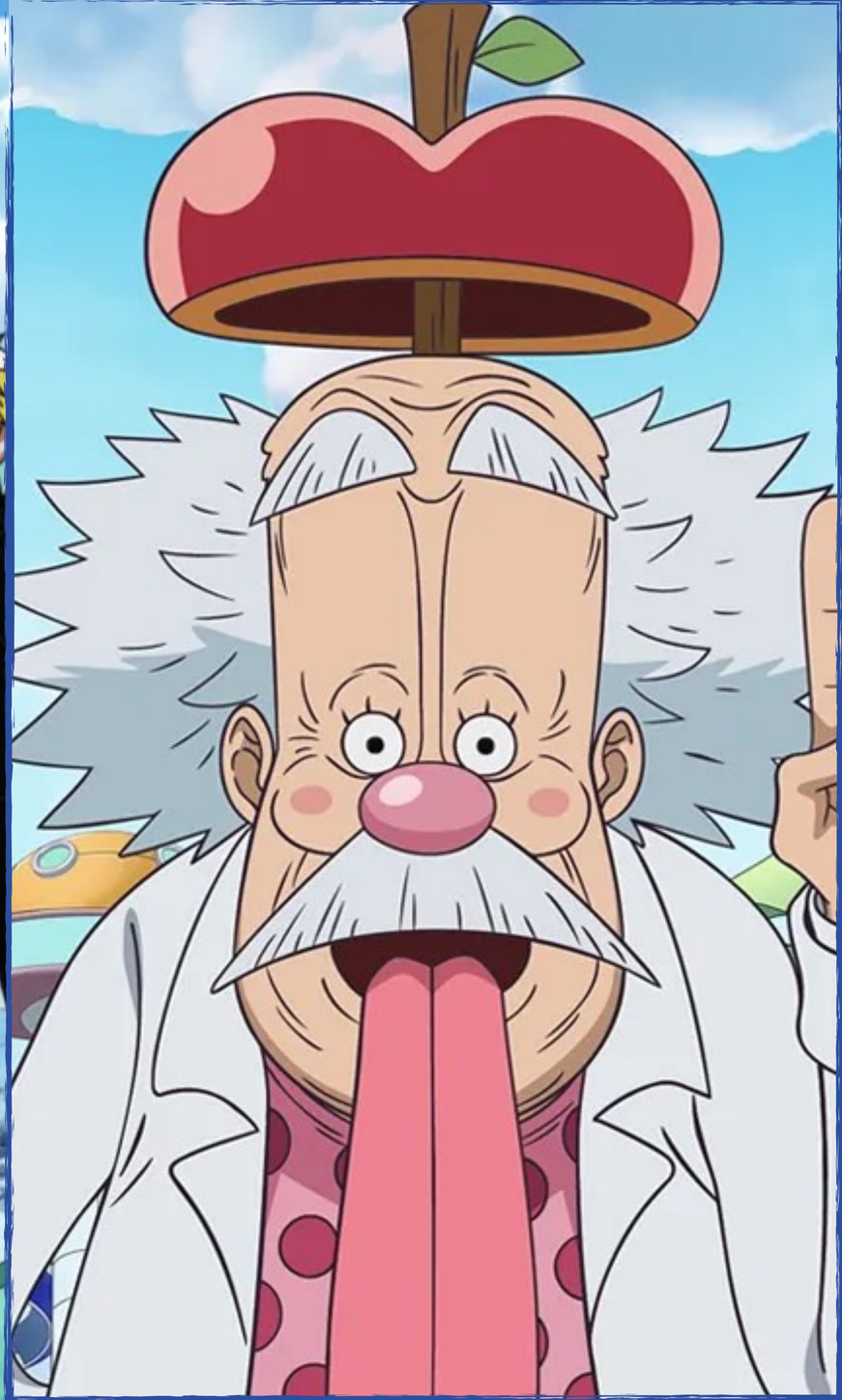
	Gen. - Feb. - Mar.
Enies Lobby	Introduzione
Nico Robin	Storia
Ohara	Architetture
Imperatori	Processi

OHARA

MEMORIA

"Un buon progetto nasce sempre da un compromesso tra obiettivi in conflitto."

(Edsger W. Dijkstra)



Architetture

Sistemi Operativi

Prof. Andrea Bianco

Architetture

Argomenti del capitolo

- CPU e Interrupt
- Dispositivi di I/O e DMA
- System call

CPU e Interrupt

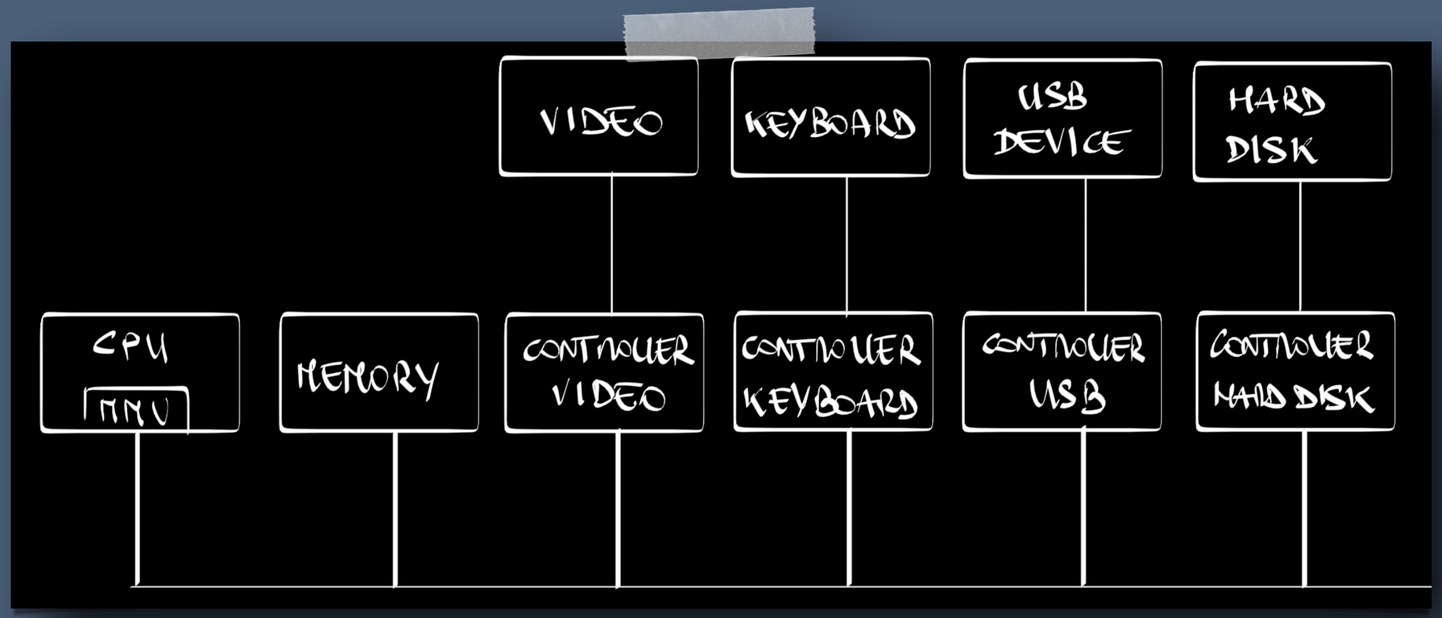
Architetture

Architetture

CPU e Interrupt

Il SO interagisce direttamente con l'hardware.

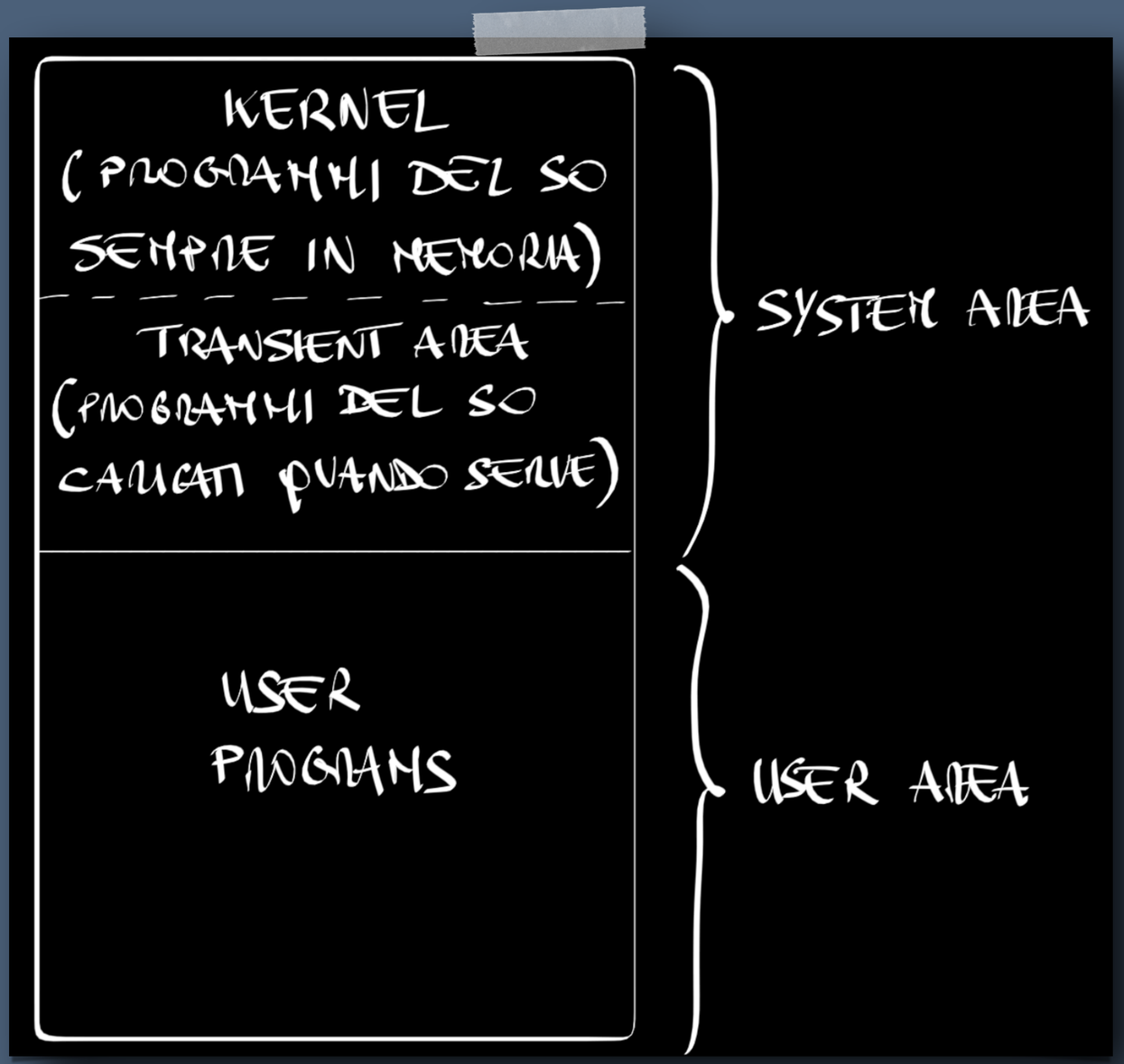
Il progettista del SO deve conoscere l'hardware su cui gira il SO.



Architetture

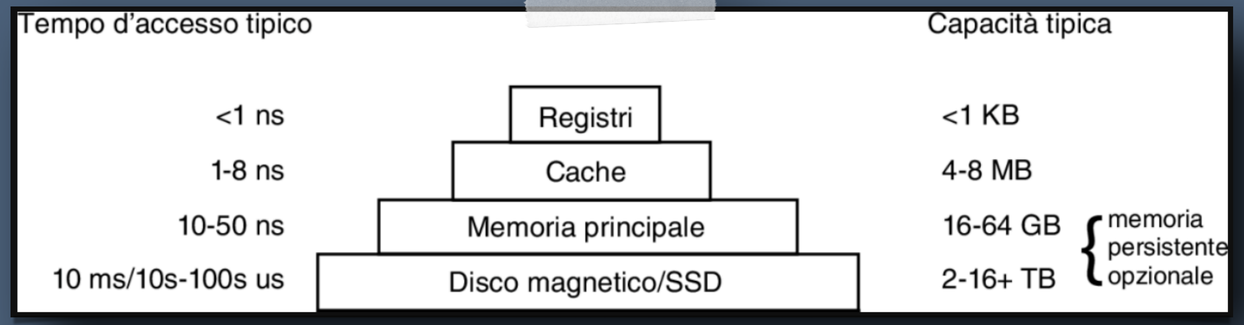
CPU e Interrupt

Organizzazione memoria



Architetture

CPU e Interrupt



CPU

La CPU preleva le istruzioni dalla memoria, le decodifica e le esegue. Ciclo di funzionamento CPU:

- Fetch
- Decode
- Execute
- Goto 1

La CPU si avvale di **registri generali** (general purpose registers), che costituiscono il primo livello della gerarchia di memoria e che memorizzano variabili e risultati temporanei.

La CPU si avvale di **registri di controllo** (control registers), che servono per controllarne il funzionamento.

Architetture

CPU e Interrupt

Registri di controllo

Program Counter (PC): indirizzo istruzione da prelevare.

Stack Pointer (SP): puntatore al top dello stack contenente i frame delle procedure già iniziate ma non ancora terminate.

Program Status Word (PSW):

- Privileged Mode (PM): bit di modalità user/kernel;
- Condition Code (CC): codici di condizione impostati da istruzioni di confronto o codici di proprietà di operazioni aritmetiche;
- Interrupt Mask (IM): bit usati per gestire interrupt;
- Interrupt Code (IC): bit usati per gestire interrupt;
- Memory Protection Information (MPI): informazioni sulla porzione di memoria accessibile.

Architetture

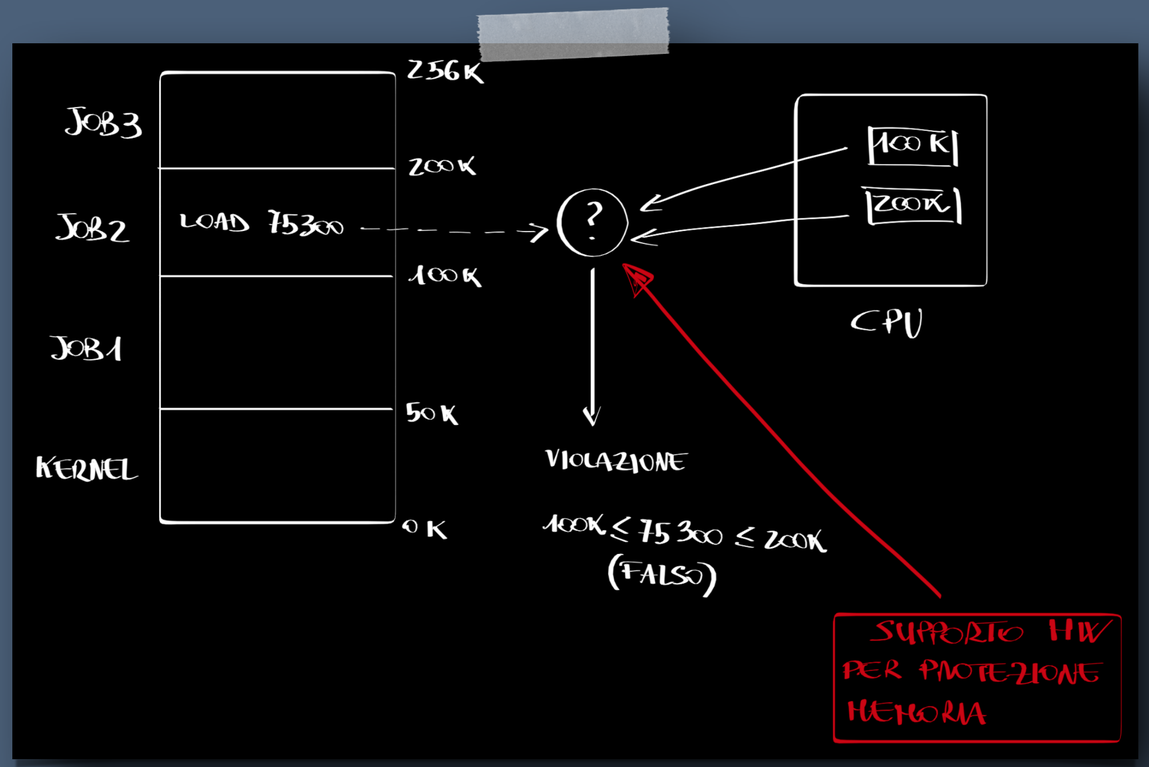
CPU e Interrupt

Memory Protection Information (MPI)

Supporto hardware per costringere i job ad accedere solo al proprio spazio di memoria. Di fatto, impediscono alla CPU di accedere agli indirizzi di memoria che non fanno parte dello spazio di memoria del job in esecuzione.

Il numero ed il significato di tali bit dipende da come è organizzata la memoria.

Qui di fianco, assumiamo allocazione contigua della memoria, no paginazione e/o segmentazione (capitolo Memoria).



Architetture

CPU e Interrupt

Privileged Mode (PM)

Le istruzioni privilegiate possono essere eseguite anche in user mode, purché il programma sia affidabile.

L'accesso alle risorse (memoria, dispositivi) è limitato da un opportuno sistema di protezione (esempio, strutture dati del SO) quando la CPU è in stato user, mentre è illimitato quando la CPU è in stato kernel.

I programmi critici (quelli che svolgono le funzioni del SO) sono eseguibili solo in stato kernel.

BUG

Le istruzioni privilegiate possono essere eseguite anche in user mode, purché il programma sia affidabile.

Correzione:
Le istruzioni privilegiate, cioè le istruzioni che modificano la PSW, che modificano i valori della MMU, che operano sulle porte I/O per comunicare con i dispositivi sono eseguibili solo in stato kernel. (esempio: istruzioni modifica LBR/UBR eseguite al momento del context switch).

Architetture

CPU e Interrupt

Interrupt

Hardware Interrupt

I/O-interrupt: gli interrupt hardware vengono gestiti direttamente dal programma utente che era in esecuzione.

Timer interrupt: usati dal clock per notificare il tick. Si tratta di eventi asincroni rispetto al programma in esecuzione. Il programma non ha modo di prevedere se e quando questi eventi si verificano. Gli eventi vanno gestiti dal SO (esempio, il SO controlla se il tick ha causato l'esaurimento del time slice del programma in esecuzione).

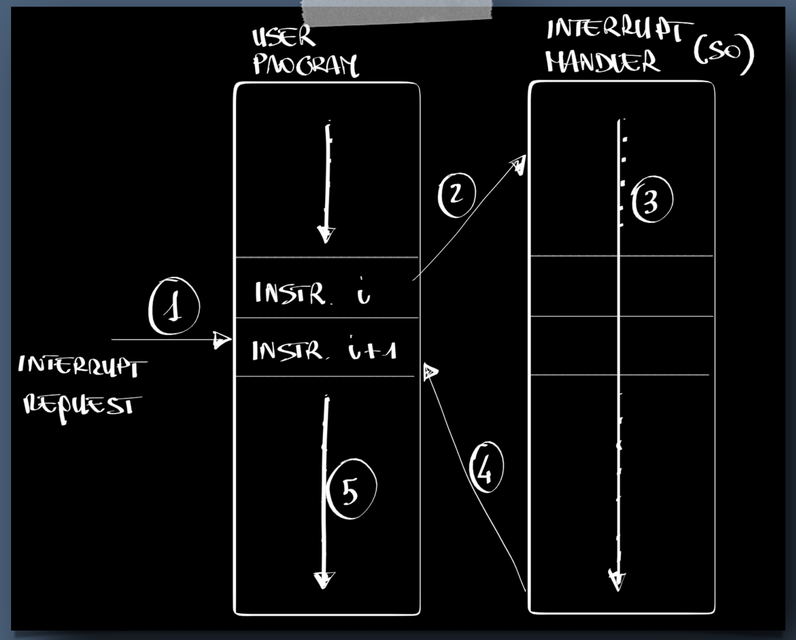
BUG

gli interrupt hardware vengono gestiti direttamente dal programma utente che era in esecuzione.

Correzione:
usati dai dispositivi I/O per notificare eventi alla CPU che richiedono di essere trattati dal SO (esempio, terminazione di un'operazione di lettura/scrittura su disco).

Architetture

CPU e Interrupt



Interrupt

Hardware Interrupt

Fasi:

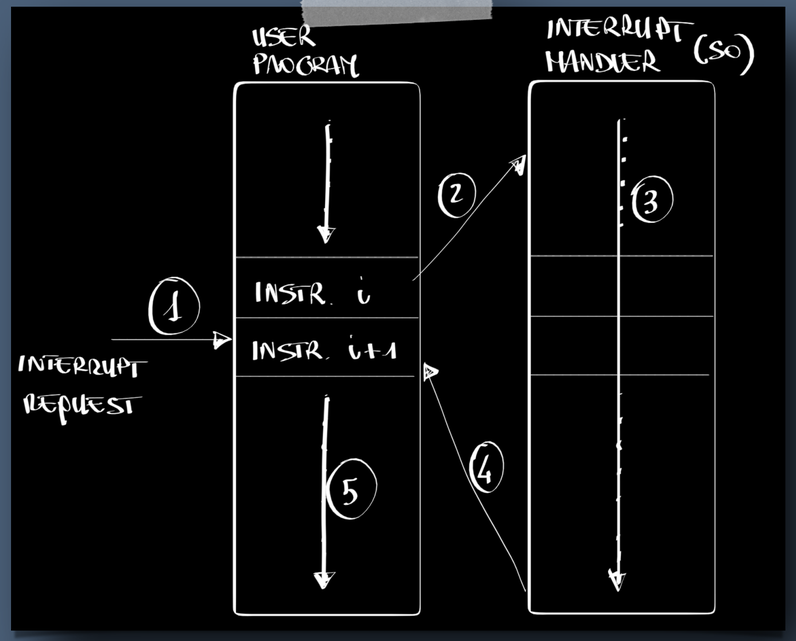
1. La CPU, mentre sta eseguendo l'istruzione i del programma P, riceve un interrupt request su una linea del bus, che verrà presa in considerazione prima di prelevare l'istruzione i+1;
2. terminata l'esecuzione dell'istruzione i, la CPU sospende l'esecuzione di P, cioè non preleva l'istruzione i+1, e salta alla procedura di gestione dell'interrupt, detta interrupt handler;
3. l'interrupt handler, che è parte del SO, gestisce l'interrupt;
4. *l'interrupt handler restituisce il controllo a P;
5. *P riprende l'esecuzione prelevando l'istruzione i+1, come se nulla fosse accaduto.

Architetture

CPU e Interrupt

Interrupt

Hardware Interrupt

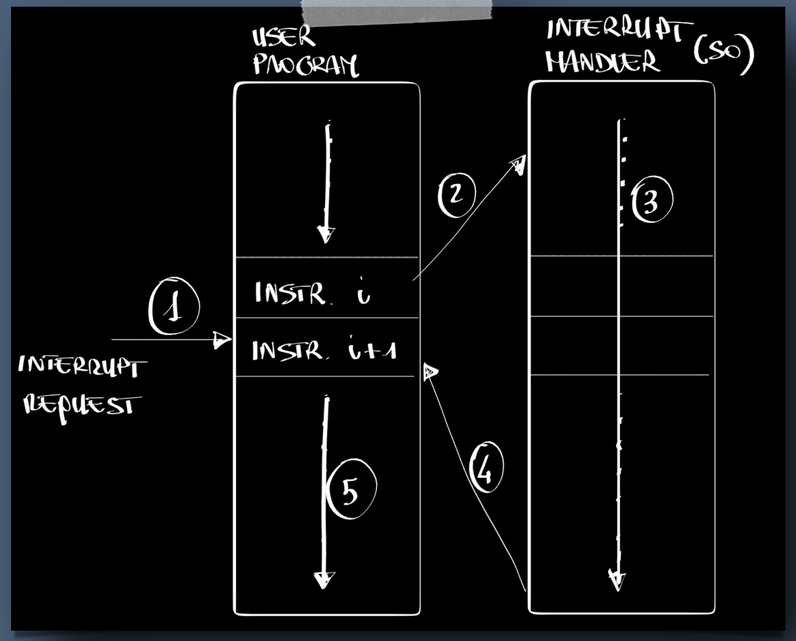


Le fasi 2 e 4 sono tutt'altro che banali, in quanto è immediato notare che il valore dei registri della CPU, ed in particolare del PC, va salvato nella fase 2 e ripristinato nella fase 4 (dettagli più avanti).

*In base alle politiche di scheduling, al punto 4 l'interrupt handler potrebbe restituire il controllo ad un programma P' interrotto in passato, anziché a P.

Architetture

CPU e Interrupt



Interrupt

<u>Hardware Interrupt</u>

Fasi dell'interrupt hardware

1. 2. 3. (Vedi slide precedenti);
4. Al termine dell'interrupt handler, il controllo ritorna sempre allo stesso programma interrotto.
5. Il programma schedulato al punto 4 riprende la propria esecuzione dal punto in cui era stato interrotto.
6. Nel caso venga schedulato P, viene prelevata l'istruzione i

BUG

Al termine dell’interrupt handler, il controllo ritorna sempre allo stesso programma interrotto.

Correzione:
l’interrupt handler restituisce il controllo a P, oppure ad un altro programma P' interrotto in precedenza;

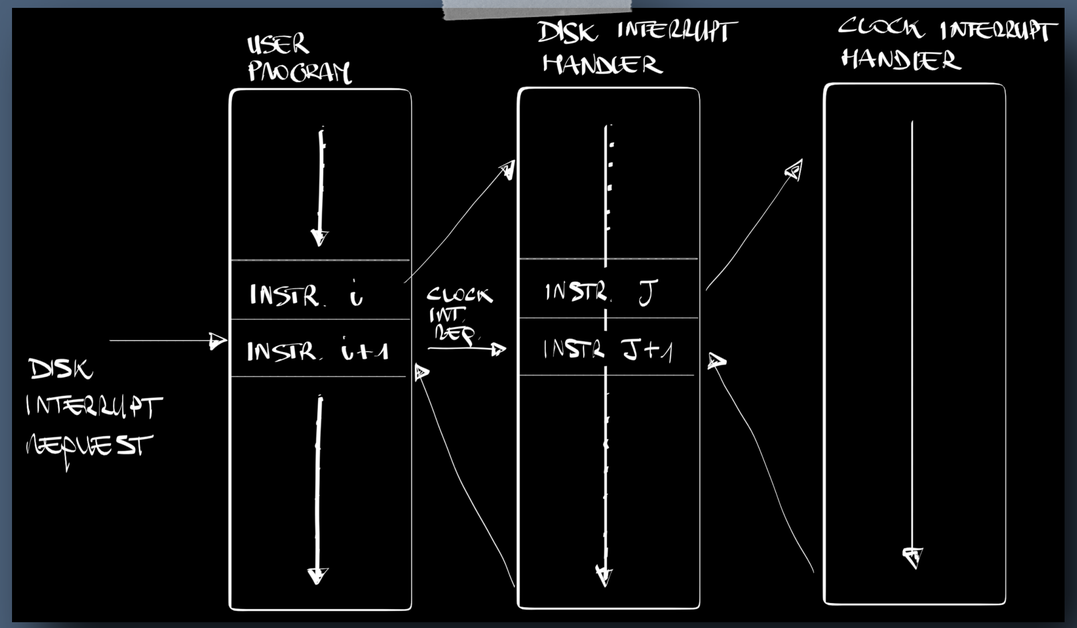
Architetture

CPU e Interrupt

Interrupt

Esempio

Se arriva un interrupt request dal clock mentre è in esecuzione l'interrupt handler per gestire un interrupt generato dall'hard disk, la nuova request può essere soddisfatta come mostrato nella slide seguente, ma servono accorgimenti.



Architetture

CPU e Interrupt

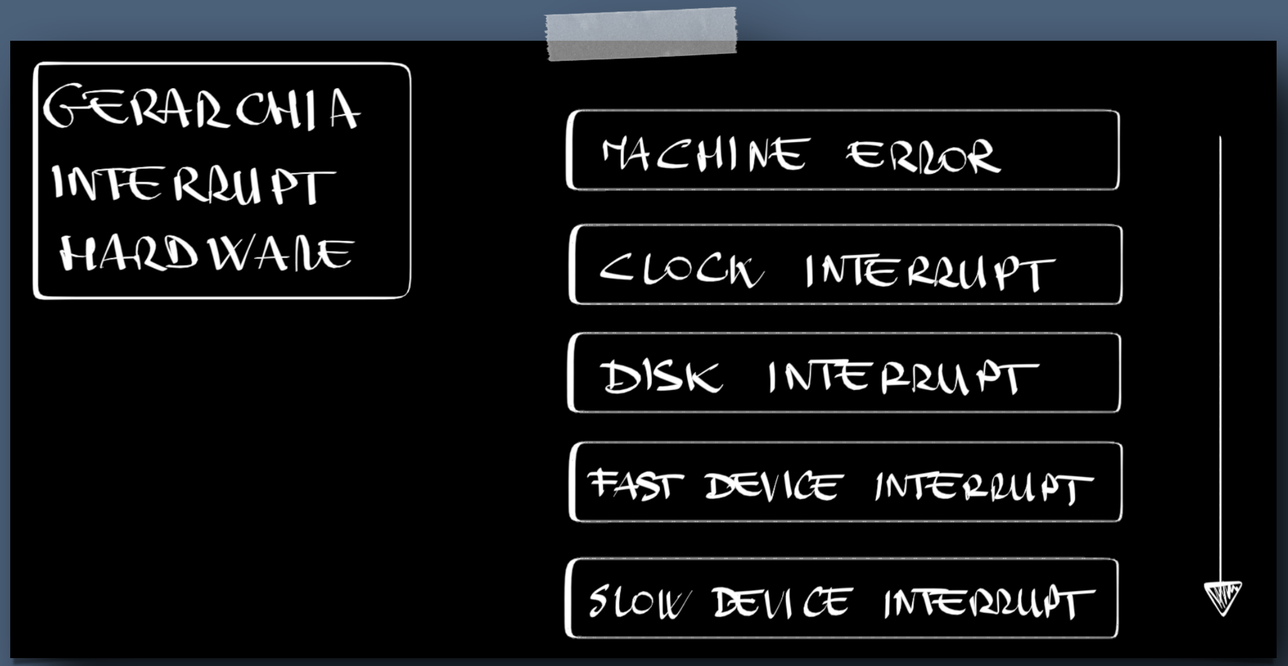
Interrupt

Interrupt a cascata

Gli interrupt vengono organizzati in classi di priorità.

Quando si gestisce un interrupt request si ignorano temporaneamente le richieste di interrupt della medesima classe e delle classi con minor priorità, la cui gestione viene rimandata.

Tali richieste rimangono pendenti.



ANSIA

Cosa si intende per overhead?

ANSIA

Cosa si intende per overhead?

Risposta:
Il tempo utilizzato non per l’esecuzione ma per la gestione

Architetture

CPU e Interrupt

Interrupt

Per ignorare gli interrupt request, PSW contiene alcuni bit che svolgono il ruolo di Interrupt Mask (IM) e che determinano le classi di interrupt abilitate (enabled) e quelle non abilitate (masked off).

Implementazioni possibili:

- l'IM contiene un bit (enabled/masked off) per ogni classe di interrupt;
- l' IM contiene un valore m per abilitare tutti e soli gli interrupt di priorità >= m.

Quando il programma gira in modo user, tutti gli interrupt sono abilitati. Il programma non può modificare la PSW, pertanto non può disabilitare gli interrupt.

Quando il controllo passa ad un interrupt handler, l'IM viene settato in modo opportuno.

Architetture

CPU e Interrupt

Interrupt

Fase 2

Per ogni classe di interrupt abbiamo un **interrupt vector**, memorizzato nella system area, che memorizza:

- l'indirizzo della prima istruzione dell'interrupt handler, da assegnare al PC per attivare l'interrupt handler, cioè per avviare la fase 3.
- il valore da assegnare al registro PSW durante l'esecuzione dell'interrupt handler. Il valore dei bit IM sono tali da abilitare/disabilitare in modo opportuno gli interrupt delle varie classi mentre si esegue l'interrupt handler.

Gli interrupt vector vengono inizializzati in memoria durante la fase di booting. Sono un esempio di struttura dati del kernel, non accessibile ai programmi user.

Architetture

CPU e Interrupt

Interrupt

Fase 2

Per ogni classe di interrupt abbiamo una **Saved Register Information Area (SRIA)**, che serve per memorizzare i valori dei seguenti registri quando la CPU riceve l'interrupt:

- PC
- SP
- PSW

La SRIA può essere allocata (capitolo Processi):

- in una zona di memoria apposita
- nello stack

Architetture

CPU e Interrupt

Interrupt

Fase 2

Cosa accade quando arriva un interrupt abilitato:

- L'hardware imposta l'interrupt code (IC) della PSW, che serve per comunicare al SO le cause dell'interrupt e che verrà esaminato dall'interrupt handler;
- I valori dei registri PC, SP e PSW vengono salvati nell' apposita SRIA. Saranno sfruttati nella fase 4. I registri generali non sono stati ancora salvati.
- I registri PC e PSW vengono impostati in base ai valori memorizzati nell' interrupt vector apposito. L' interrupt handler può pertanto partire. La fase 3 ha inizio.

Architetture

CPU e Interrupt

Interrupt

Fase 3

Cosa fa l'interrupt handler:

- salva i registri generali della CPU. A questo punto i valori di tutti i registri della CPU relativi all'esecuzione del programma P sono stati salvati* e potranno essere ripristinati quando P riprenderà l'esecuzione dalla fase 4 (*capitolo Processi).
- esegue il codice apposito per gestire l'interrupt utilizzando il valore dell'IC (esempio, se abbiamo 2 dischi ed arriva un disk interrupt, l' IC contiene l' indicazione circa il disco che ha causato l' interrupt, informazione indispensabile per l'interrupt handler).
- salta allo scheduler, che seleziona il programma P' (può essere che P' = P) cui assegnare la CPU nella fase 4.

Architetture

CPU e Interrupt

Interrupt

Fase 4

cosa fa lo scheduler dopo che ha selezionato il programma P' che deve riprendere l'esecuzione:

- vanno ripristinati i valori dei registri generali per eseguire il programma schedulato P' che erano stati salvati (non abbiamo ancora detto dove, capitolo Processi).
- vanno ripristinati i valori dei registri di controllo che erano stati salvati nella SRIA. Spesso l'architettura offre l'istruzione Interrupt RETurn (IRET) per ripristinare, tutti insieme, i registri di controllo.

Architetture

CPU e Interrupt

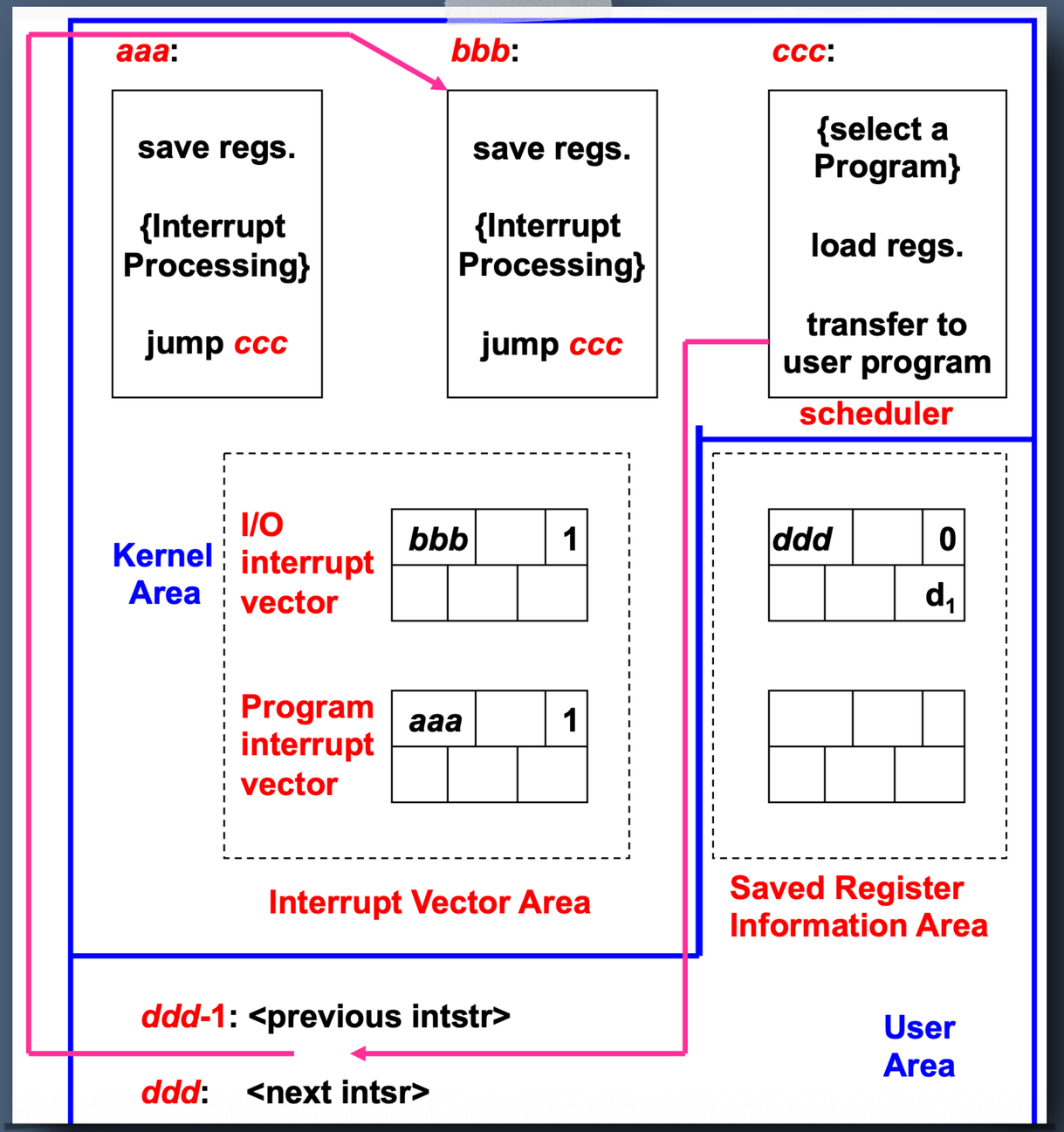
Interrupt

Esempio

Arriva un I/O interrupt dal dispositivo numero 1 prima che il programma utente esegua l'istruzione di indirizzo ddd.

Semplificazioni:

- Unica classe di I/O interrupt
- Mancano alcuni interrupt vector (esempio, clock)
- Sono riportati solo i valori da assegnare a PC e PM.

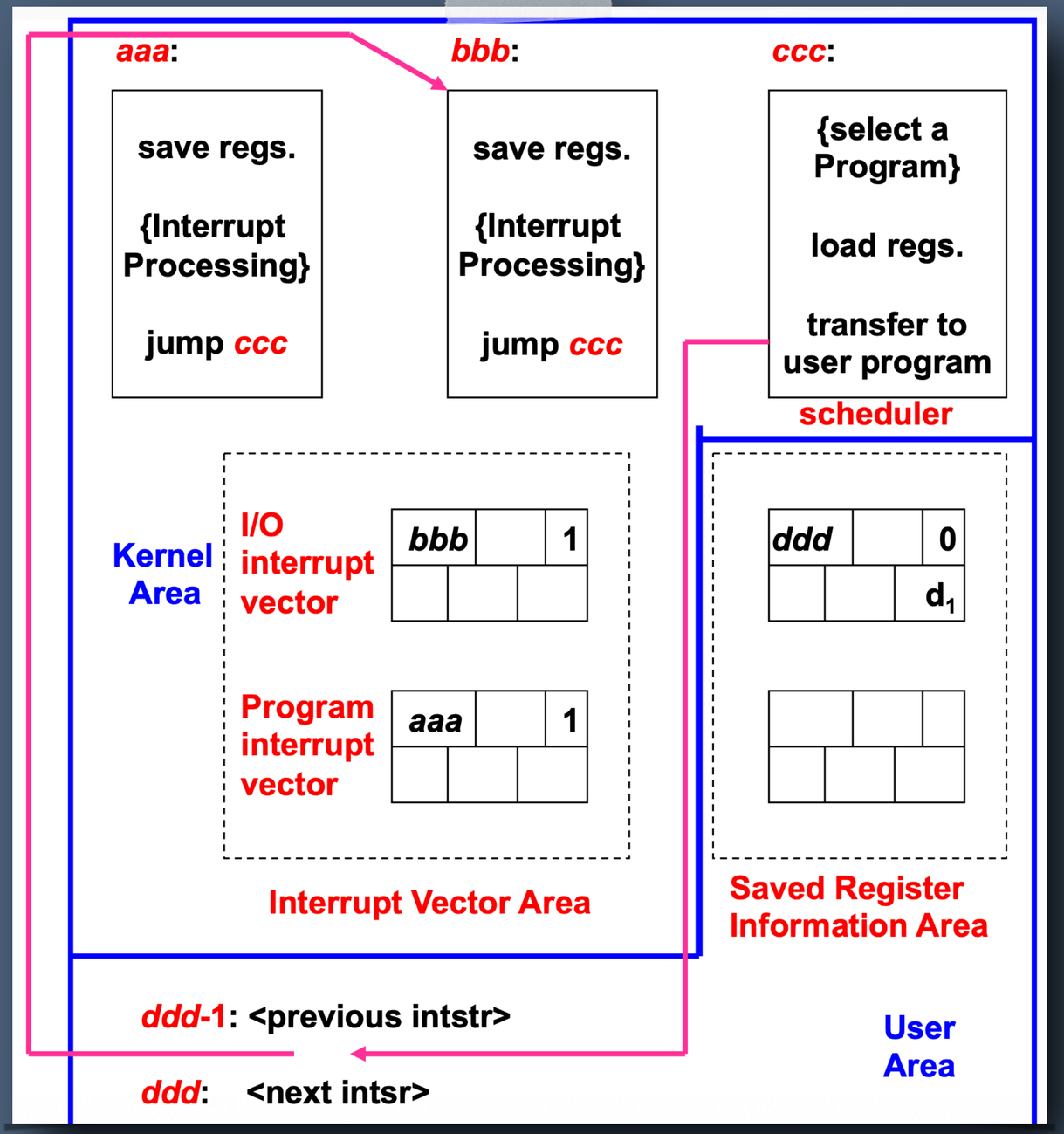


Architetture

CPU e Interrupt

Interrupt

- Salvataggio hw dei registri di controllo:
  - PC = ddd
  - PM = 0 (user)
  - IC = d1 (device numero 1 nella SRIA)
- Settaggio dei registri di controllo sfruttando l' interrupt vector:
  - PC=bbb
  - PM=1 (kernel)

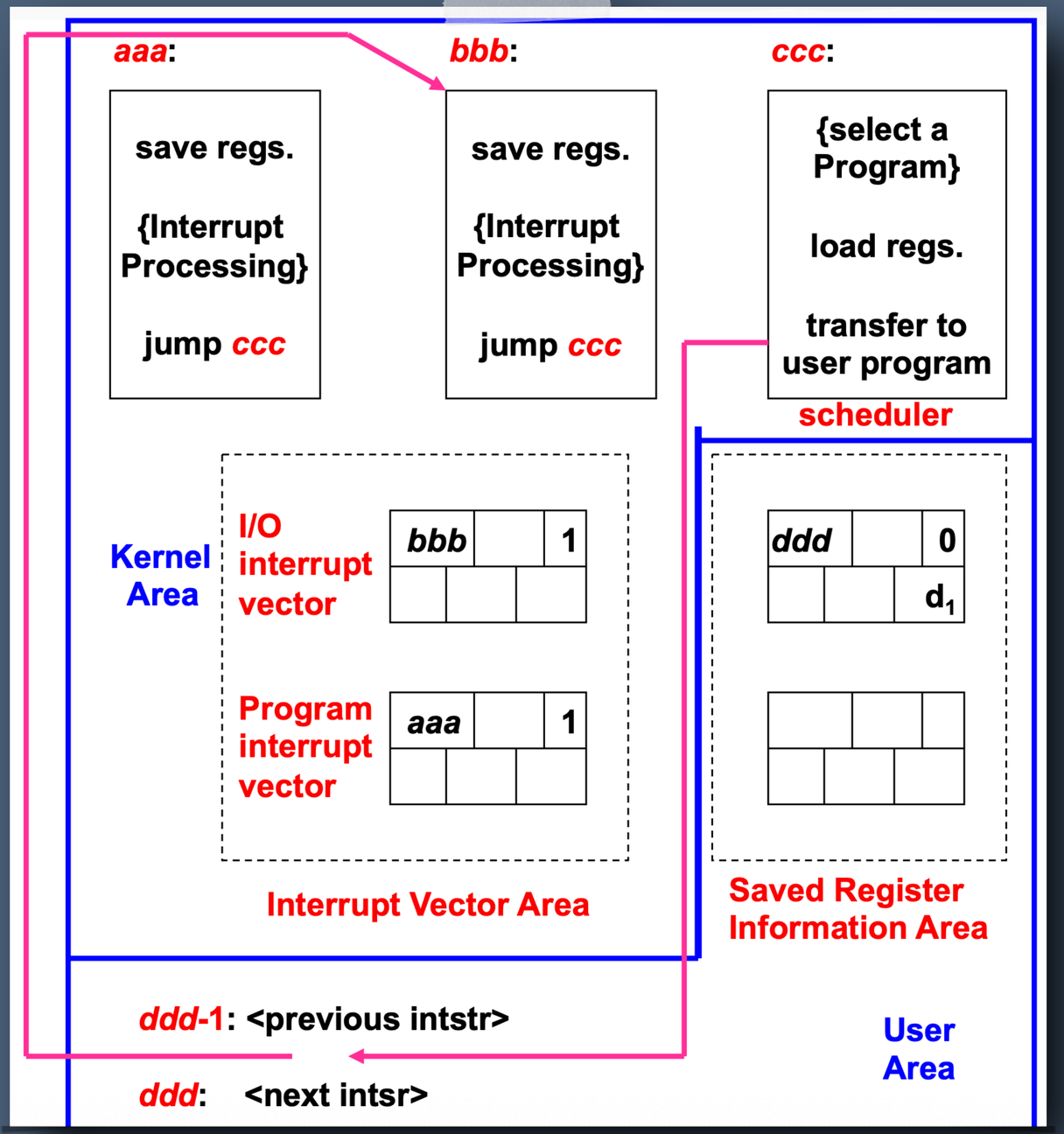


Architetture

CPU e Interrupt

Interrupt

- L'interrupt handler (indirizzo bbb):
  - salva il contenuto dei registri generali (capitolo Processi)
  - controlla l'IC per individuare il device (cioè d1)
  - esegue le sue funzioni
  - invoca lo scheduler (codice all'indirizzo ccc)

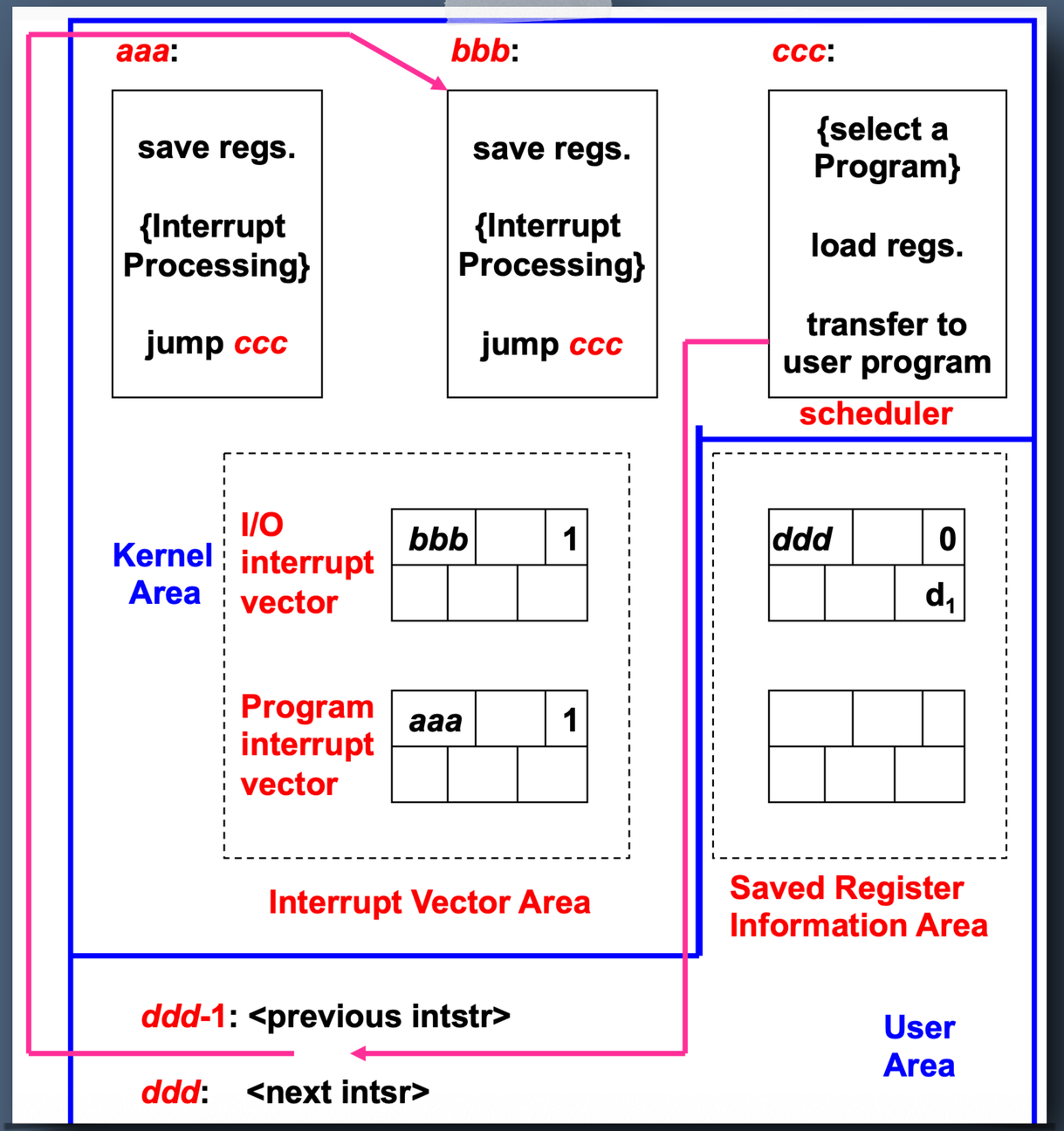


Architetture

CPU e Interrupt

Interrupt

- Se lo scheduler seleziona il programma interrotto:
  - ripristina i registri generali precedentemente salvati
  - ripristina i registri di controllo salvati nella SRIA (istruzione IRET)
  - il PC assume valore ddd



Architetture

CPU e Interrupt

Interrupt

I SO assumono che l'hardware supporti gli interrupt.

Si tratta di un esempio di evoluzione dell'hardware dettata dallo sviluppo dei SO.

Riguardo al salvataggio dei registri della CPU che viene interrotta:

- il PC deve essere salvato via hardware;
- gli altri registri possono essere salvati via software, cioè dall'interrupt handler.

Il salvataggio via software è più lento, ma richiede un hardware meno complicato.

Come compromesso, spesso l' hardware offre la possibilità di salvare tutti i registri di controllo, ma non i registri generali.

Questo è l'approccio discusso in precedenza. Anche la IRET va in questa direzione.

Architetture

CPU e Interrupt

Interrupt

I registri della CPU devono essere salvati anche quando è un interrupt handler ad essere interrotto da un interrupt.

La gestione degli interrupt con classi di priorità fa si che solo interrupt di priorità maggiore possano interrompere un interrupt handler.

Ne consegue che, per ogni programma P, è sufficiente una SRIA per ogni classe di interrupt.

Architetture

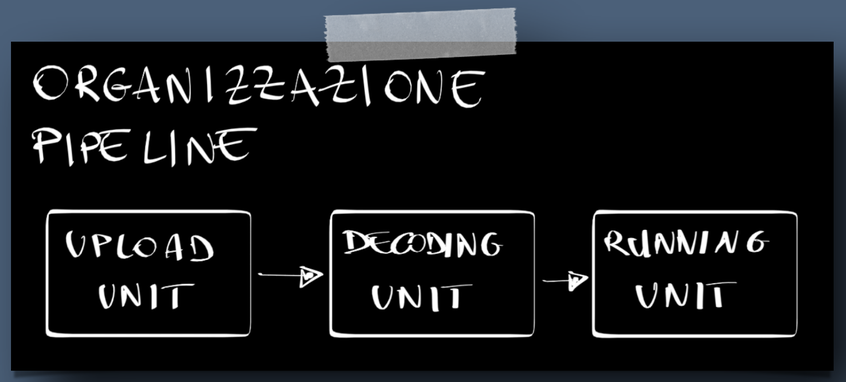
CPU e Interrupt

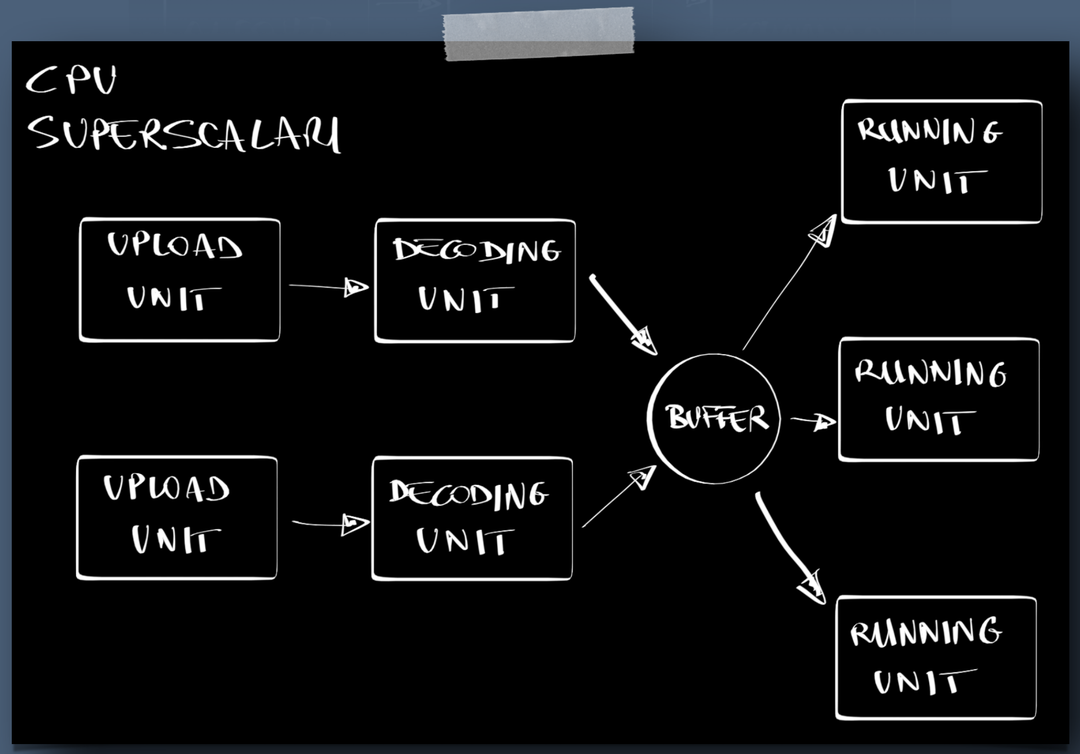
Interrupt

Il ciclo

- Fetch
- Decode
- Execute
- Goto 1

non è più attuale





Architetture

CPU e Interrupt

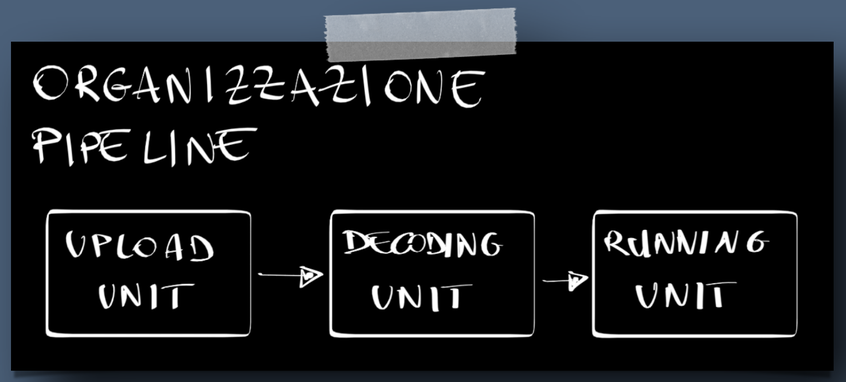
Interrupt

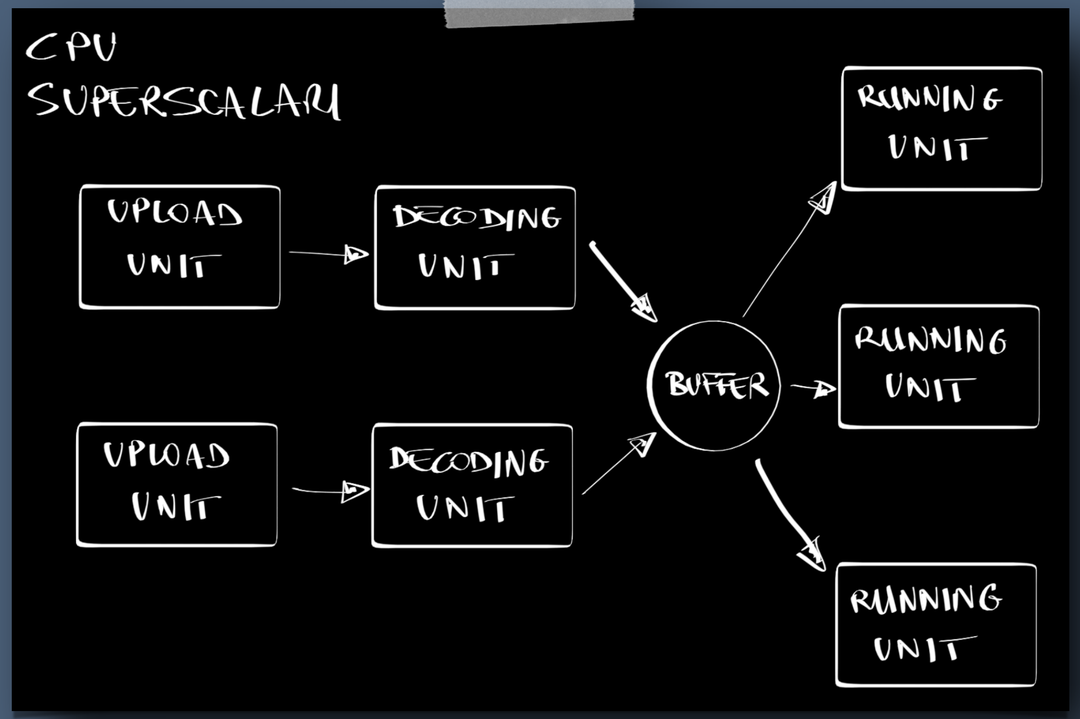
Con queste tipologie di CPU le prestazioni aumentano.

Però, quando arriva l'interrupt:

- possono esserci istruzioni già prelevate ma non ancora eseguite
- nel caso delle CPU superscalari, possono esserci istruzioni eseguite solo parzialmente

Noi considereremo il caso più semplice.





ESERCIZIO

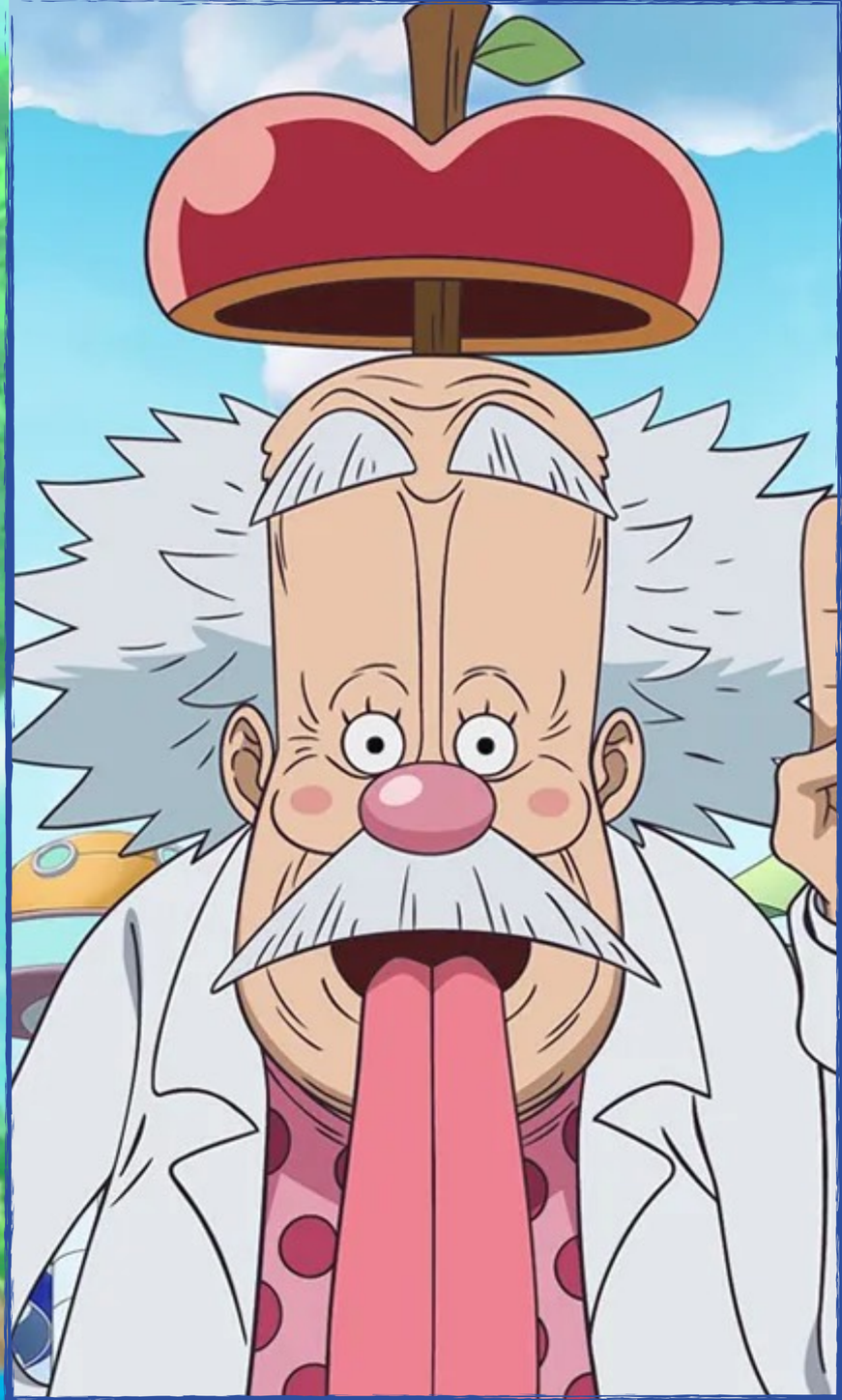
Disegna e descrivi le fasi di gestione di un interrupt hardware, indicando:

- cosa fa l'hardware
- cosa fa l'interrupt handler
- dove interviene lo scheduler
- quando avviene il passaggio user → kernel → user

MEMORIA

"La complessità è il peggior nemico dell'affidabilità."

(Edsger W. Dijkstra)



COMPITI

Spiega in 15 righe massimo:
Perché è necessario distinguere tra user mode e kernel mode dal punto di vista della sicurezza e della stabilità del sistema.

Descrivi un esempio reale (Linux, Windows o macOS) in cui:

- un timer interrupt provoca un context switch
- indica cosa accade al PC e alla PSW

Rispondi: perché il salvataggio dei registri via hardware è più veloce ma meno flessibile rispetto al salvataggio via software?