Sistemi operativi

5. Richiami hardware

Mauro Fiorentini

5. Richiami hardware

Caratteristiche dell'hardware

- Stati del processore
- Coprocessori
- Memoria
- Cache
- Input/Output

Stati del processore

- Un processore ha solitamente almeno due stati:
  - user, per i processi;
  - kernel, per il sistema operativo.
- Solo i processori più semplici, usati nelle applicazioni embedded, hanno uno stato unico.
- Alcuni processori hanno più stati (fino a 4 o 8), uno solo dei quali utilizzato dai processi utente.

Stato user

- Sono vietate alcune operazioni:
  - I/O;
  - accesso ai registri che controllano paginazione e protezioni della memoria;
  - accesso ad alcuni registri di stato, che definiscono i privilegi (user/system mode) e controllano gli interrupt.
- Il tentativo di eseguire una di tali operazioni provoca un trap di istruzione illegale.

Stato kernel

- Libero accesso a tutte le caratteristiche dell'hardware.
  - Le protezioni di memoria restano valide, ma il sistema può modificarle.

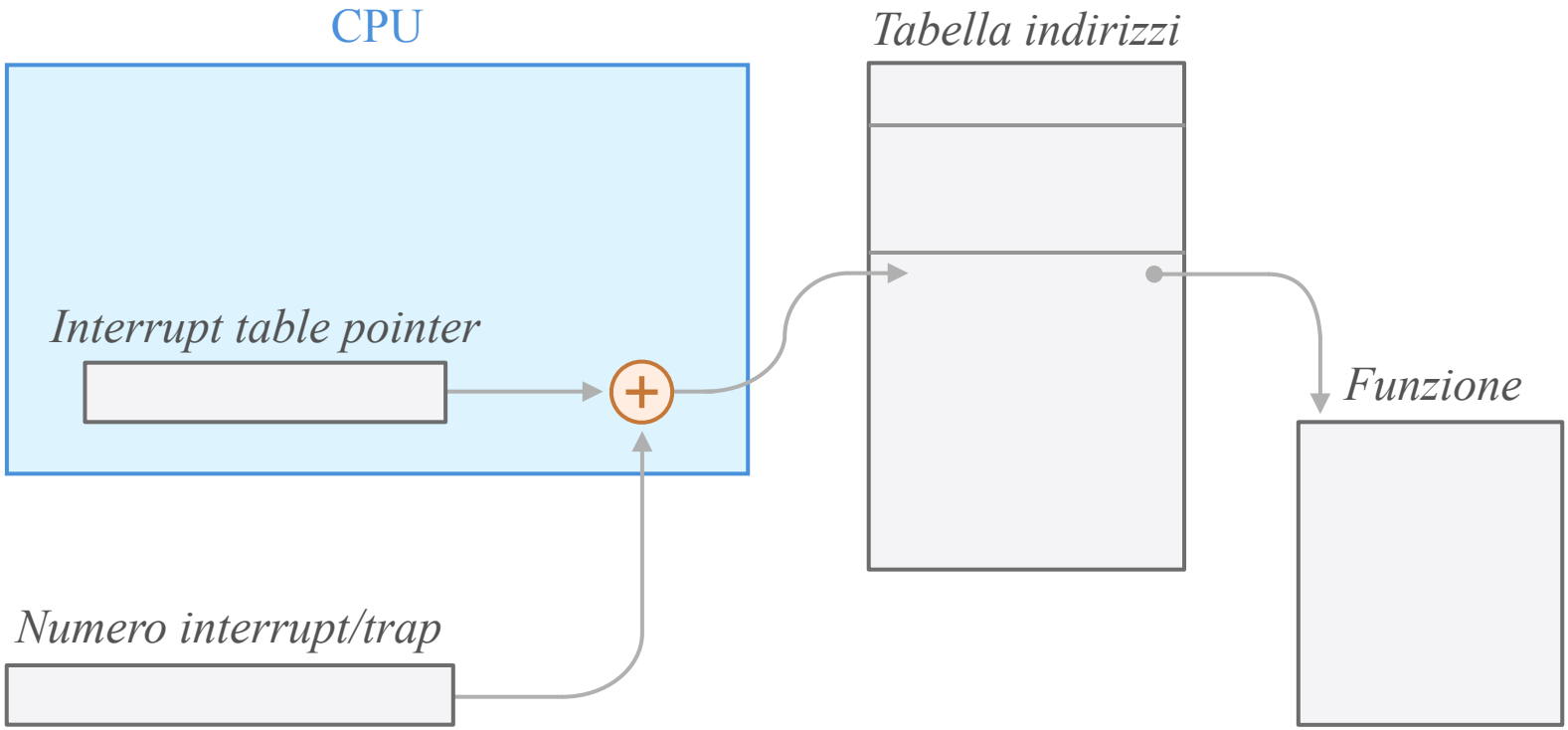
Cambiamento di stato

- Da stato user a stato kernel avviene in 3 casi:
  - interrupt;
  - trap;
  - system call.
- La transizione viene sempre gestita dall'hardware come un interrupt.

Transizione da user a kernel

- La CPU:
  - salva sullo stack di sistema pochi registri (di solito solo program counter e stato);
  - passa in stato kernel;
  - cede il controllo a un indirizzo fisso o a un indirizzo prelevato da una tabella, utilizzando il numero di interrupt o trap come indice.
- Il sistema operativo salva il resto dello stato del processo, inclusi i rimanenti registri.

Indirizzo della funzione gestione interrupt (Motorola)



Transizione da kernel a user

- Avviene al termine della gestione di un interrupt, trap o system call.
- Il sistema operativo ripristina lo stato del processo e poi esegue un'istruzione di "return from interrupt" (eseguibile solo in stato kernel), che ricarica program counter e stato dallo stack.
  - Il cambiamento di stato avviene ricaricando il valore del registro di stato.

Creazione di processi o thread

- Per creare un processo, il sistema crea un'immagine, fittizia, ma coerente, dello stato iniziale:
  - Program Counter che punta alla prima istruzione;
  - registro di stato con modo utente, con interrupt abilitati;
  - Stack Pointer che punta allo stack;
  - altri registri con un valore convenzionale (p. es. 0).
- Quando lo stato verrà "ricaricato", il processo o thread diverrà running.

I coprocessori

- Componenti hardware che estendono l'insieme delle istruzioni eseguibili dalla CPU.
- Generalmente estendono anche l'insieme dei registri visibili.
- Talvolta facoltativi e acquistabili separatamente.

Istruzioni implementate in un coprocessore

- Generalmente rivolte a un ristretto ambito applicativo:
  - aritmetiche/matematiche su numeri in virgola mobile;
  - aritmetiche su interi in notazione decimale;
  - orientate alla grafica;
  - operazioni eseguite su vettori interi;
  - crittografia.
- Possono durare molti cicli (anche più di un migliaio).

Rilevamento di un coprocessore

- Quando la CPU rileva un'istruzione per un coprocessore, invia un apposito segnale.
  - Se il coprocessore risponde entro un ciclo, dichiarandosi abilitato all'esecuzione dell'istruzione, la CPU gli delega l'esecuzione dell'istruzione.
  - Altrimenti genera un trap di istruzione illegale.
    - Il trap può eventualmente innescare un'emulazione software del coprocessore.

Passaggio di informazioni tra CPU e coprocessore

- Nei sistemi più semplici, si usa il bus di memoria.
  - Meno efficiente, più semplice, meno costoso.
  - Il coprocessore può addirittura mantenere copia della coda di prefetch della CPU, per scoprire le istruzioni a lui dirette (es.: Intel 8087).
- In alternativa si usa un bus dedicato.
  - Più veloce, piu costoso.
  - Se CPU e coprocessore si trovano in chip separati, il numero di pin della CPU aumenta.

Sincronizzazione tra CPU e coprocessore (1)

- Nel caso più semplice la CPU attende la fine di ogni istruzione di coprocessore, senza fare altro.
  - Il coprocessore diviene una semplice estensione della CPU, in modo totalmente trasparente.

Sincronizzazione tra CPU e coprocessore (2)

- Con coprocessori evoluti, la CPU e il coprocessore sono in grado di eseguire istruzioni separate simultaneamente.
  - Si realizza un vero parallelismo.
  - Se viene incontrata un'altra istruzione dello stesso coprocessore prima che la precedente sia terminata, la CPU attende.
  - Esiste un'istruzione che forza l'attesa della fine di eventuali istruzioni di coprocessore in esecuzione.

Il problema delle eccezioni

- Se un coprocessore rileva un'eccezione, la segnala alla CPU, provocando un trap.
- La CPU deve memorizzare separatamente l'indirizzo dell'ultima istruzione inviata al coprocessore, per permettere la corretta gestione del trap.
  - Il program counter, infatti, può essere cambiato tra quando l'esecuzione dell'istruzione è iniziata e quando l'eccezione viene segnalata.

Eccezioni asincrone

- Possono essere state eseguite parecchie istruzioni dopo l'inizio dell'istruzione che ha causato il trap.
  - Vietato in alcuni linguaggi.
  - Il compilatore può imporre una sincronizzazione più stretta, forzando l'attesa dopo ogni istruzione di coprocessore, riducendo però le prestazioni.

Il sistema operativo e i coprocessori

- I registri visibili del coprocessore fanno parte dello stato del processo.
- A ogni context switch il sistema deve forzare l'attesa della fine dell'istruzione in corso, poi salvare tutti i registri.
  - Può essere scatenata un'eccezione, che appare come provocata dal sistema stesso e va trattata opportunamente.
  - Il sistema deve sapere quali coprocessori vi siano.

Il salvataggio dei registri del coprocessore

- Può richiedere molto tempo.
  - Molti coprocessori hanno istruzioni apposite per salvare molti registri con una sola istruzione.
- Molti coprocessori hanno un flag che diviene 1 quando eseguono un'istruzione.
  - Il sistema lo azzera quando ricarica lo stato.
  - Se al successivo context switch è ancora zero, non serve salvare lo stato.

La memoria

- Vista dall'unità centrale come un insieme di locazioni, contraddistinte da un diverso indirizzo.
- Rappresentabile come un vettore di celle, con indici generalmente consecutivi.
- Utilizzando n bit si possono distinguere 2^n locazioni ("spazio di indirizzamento").
  - Serve un bus indirizzi di n bit.

Le celle di memoria

- La minima unità indirizzabile può essere:
  - il byte (di solito 8 o 9 bit): macchine byte addressable;
  - la word (da 12 a 64 bit): macchine word addressable.
- L'indirizzamento a word:
  - aumenta il numero di byte indirizzabili;
  - rende scomodo l'accesso ai singoli byte;
  - è tipico di supercalcolatori per scopi scientifici.

Accesso fisico alla memoria

- Avviene in unità che dipendono dal numero di fili del bus dati:
  - un sottomultiplo della lunghezza di parola;
    - più lento, più economico, tipico di microcalcolatori a basso costo;
  - uguale alla lunghezza della parola;
    - tipico di personal computer;
  - un multiplo della lunghezza di parola;
    - più veloce, se in presenza di cache, più costoso, tipico di calcolatore di buone prestazioni.

Accesso logico alla memoria

- Avviene in unità che dipendono dal set di istruzioni (dal byte alla doppia o quadrupla parola).
- Può essere necessario rispettare vincoli di allineamento, se il processore non è in grado di gestire accessi logici che richiedano più accessi fisici alla memoria.

Cache

- Memoria ausiliaria ad alta velocità.
  - Costa più della RAM normale.
  - E' meno densa (meno byte per chip).
  - Consuma molto di più.
- Nello stesso chip della CPU o collegata a esso con un bus dedicato.
- Conserva copie delle ultime parole lette o scritte in memoria.

Cache

Indirizzo	Contenuto	Flag
•	•	•
•	•	•
•	•	•
28 bit	32 bit	2 bit

Un gran numero di elementi identici (linee).

Il confronto degli indirizzi
avviene in parallelo su di un piccolo numero di elementi

I flag indicano:

- se una linea è valida;
- se è stata modificata.

Vantaggi della cache

- Riduce gli accessi alla memoria reale, più lenta.
  - Indispensabile per i processori molto veloci, che perdono troppi cicli per un accesso in memoria.
  - E' normale che oltre il 95% degli accessi si fermi alla cache, senza raggiungere la memoria.

Caratteristiche della cache

- Ogni elemento della cache, detto “linea”, contiene da 1 a 64 parole.
- Da poche centinaia a centinaia di migliaia di linee.
- Estremamente veloce, in grado di rispondere in pochi cicli del processore.
- Gestione interamente hardware.

Organizzazioni di cache

- Esistono due organizzazioni di cache.
  - Ad accesso diretto: ogni indirizzo di memoria corrisponde a una linea;
    - La corrispondenza può essere realizzata con una semplice funzione di hashing o utilizzando alcuni bit dell'indirizzo stesso.
    - Più semplici, più veloci.
  - Associative: per ogni indirizzo esiste un piccolo numero di linee, candidate a contenere quella parola.
    - Più complesse, lievemente più lente, ma con prestazioni migliori.
    - Nelle cache "**fully associative**" tutte le linee disponibili possono essere utilizzate.

Accesso alla memoria con cache

- A ogni accesso in memoria, l'hardware determina le linee candidate, mediante una semplice funzione di hashing sull'indirizzo.
- Gli indirizzi contenuti nelle linee candidate vengono prelevati e confrontati in parallelo.

Lettura in memoria con cache (1)

- Se la parola cercata è presente in cache, (“cache hit”), viene letta.
- Altrimenti (“cache miss”) si procede con un normale accesso in memoria, poi una delle linee candidate viene rimpiazzata con un’intera linea, letta dalla memoria all’indirizzo richiesto.

Lettura in memoria con cache (2)

- Viene sempre effettuata a linee intere.
- Sfrutta la possibilità del bus di trasferire più parole in un unico ciclo di memoria.
- In molte architetture, gli accessi successivi a indirizzi consecutivi sono molto più veloci del primo.
- E' molto probabile che una parola vicina a quella letta venga richiesta dopo breve tempo.

Selezione delle linee da rimpiazzare

- Problema analogo alla selezione delle pagine di memoria da scartare.
- Algoritmo interamente hardware e semplice.
  - Es.: FIFO tra le linee candidate.

Cause di cache miss

- Una parola può non essere mai stata caricata.
- Una parola può essere stata scaricata, dovendone caricare un'altra per mancanza di spazio.
  - Spazio complessivo, nel caso di cache "fully associative".
  - Spazio tra i candidati, nel caso di cache non "fully associative".

Scrittura in memoria con cache

- Se la parola cercata è presente in cache, (“cache hit”), viene selezionata.
- Altrimenti (“cache miss”) una delle linee candidate viene scelta.
- La nuova parola viene scritta nella linea selezionata.
- La memoria viene aggiornata.

Aggiornamento della memoria con cache

- Sono possibili due tecniche di gestione.
  - Write through: la parola viene effettivamente scritta subito in memoria.
    - La CPU attende il completamento dell'operazione.
  - Write back: la linea viene marcata come modificata; verrà scritta quando sarà rimpiazzata.
    - Generalmente viene inserita in una coda di parole da scrivere; verrà scritta appena possibile.
    - Scritture successive sulla stessa linea non comportano accessi in memoria.

Coda di accessi

Piccola coda a gestione FIFO delle scritture in attesa

Indirizzo	Dato

28 bit **32 bit**

Code di accessi

- Le cache mantengono una coda di operazioni di scrittura “pendenti”.
- Alcune gestiscono due code separate, una per le scritture, una per le letture.
- Una richiesta di lettura **ha sempre la precedenza** e viene soddisfatta per prima.
- Prima di accodare una lettura o scrittura, se ne confronta l’indirizzo con quelli delle scritture pendenti.
  - La lettura di una parola nella coda di scrittura non richiede accesso alla memoria.
  - Una scrittura destinata a essere sovrascritta viene annullata.

Ottimizzazioni delle cache

- Sono state escogitate varie tecniche per migliorare le prestazioni, aumentando la complessità dell'h/w.
  - Permettere accessi ad altre linee durante l'attesa di un accesso in memoria ("hit under miss").
  - Richiedere prima la parola richiesta dalla CPU e poi il resto della linea.
  - Verificare se una parola da scrivere è contenuta in un'altra linea già in coda di scrittura.

Problemi creati dalla cache

- Il reale accesso in memoria è asincrono rispetto all'esecuzione del programma.
- Serve un'istruzione, **eseguibile in stato user**, che blocchi il processore finché le scritture pendenti non sono terminate.
- Il sistema deve poter **invalidare l'intera cache**, quando nuove pagine vengono lette dalla memoria.

Cache e paginazione/segmentazione

- I descrittori di pagina o segmento possono identificare un elemento come “non ricercabili in cache”.
  - Serve al sistema operativo per:
    - le periferiche memory mapped;
    - i buffer dove le periferiche leggono e scrivono;
    - eventuale memoria condivisa con altri processori.

Cache e memoria virtuale

- La cache può trovarsi a monte o a valle della traduzione degli indirizzi.
  - Nel primo caso gli indirizzi conservati sono **virtuali**.
  - Nel secondo sono **reali**.

Cache con indirizzi virtuali (1)

- Si accede alla cache **prima della traduzione**, effettuata in parallelo, risparmiando tempo.
- Gli indirizzi conservati e confrontati sono **virtuali**, quindi generalmente servono più bit.
- Bisogna invalidare la cache a ogni context switch.
  - In alternativa l'id del processo può essere memorizzato insieme all'indirizzo virtuale.

Cache con indirizzi virtuali (2)

- E’ molto complesso assicurarne la coerenza in ambiente multiprocessore.

Cache con indirizzi reali

- Si accede alla cache **dopo la traduzione**.
- Gli indirizzi conservati e confrontati sono **reali**.
- Indifferente al context switch.
- E' più semplice assicurarne la coerenza in ambiente multiprocessore.

Coerenza della cache (1)

- Se altre entità (altre CPU, periferiche, DMA controller) possono accedere allo stesso bus, bisogna assicurare la coerenza della cache quando una di esse modifica la memoria.

Coerenza della cache (2)

- Si possono limitare gli accessi da parte di altre entità a una parte della memoria, in pagine marcate come "non caricabili in cache".
- In alternativa la cache può sorvegliare gli indirizzi sul bus, invalidando le linee che contengono parole nelle quali viene effettuata una scrittura da altri ("**bus snooping**").

Coerenza della cache (3)

- In architetture multiprocessore a memoria condivisa servono protocolli complessi per garantire la coerenza, soprattutto per cache write back.
  - Ogni cache deve informare le altre quando modifica un dato che contiene, anche se non lo scrive effettivamente in memoria.
    - Le altre cache devono invalidare la linea, se la contengono.
  - In caso di tentativo di lettura da parte di un altro processore, la memoria deve essere aggiornata, per consentire al richiedente di leggere il valore corretto.

Architetture di cache

- I parametri rilevanti sono:
  - il numero di linee;
  - la dimensione di una linea;
  - l'associatività (numero di candidati per ogni indirizzo);
  - l'algoritmo di scelta della linea da rimpiazzare;
  - write back o write through;
  - indirizzi virtuali o reali.
- Ogni progetto è frutto di compromessi.

Vantaggi e svantaggi

- Aumentando la dimensione della cache aumenta la percentuale di hit, ma aumentano i consumi e il tempo di accesso.
- Aumentando la dimensione della linea migliora l'effetto della località spaziale degli accessi, ma aumenta il ritardo in caso di miss.
- Aumentando l'associatività aumenta la percentuale di hit, ma aumentano i consumi e il tempo di accesso.

Cache a più livelli (1)

- Spesso sono presenti più livelli di cache:
  - uno per ogni core nel chip della CPU, in grado di rispondere velocemente (di solito in 1 o 2 cicli);
    - al massimo alcune decine di kbyte;
  - uno comune a tutti i core, sempre nel chip della CPU, in grado di rispondere cin pochi cicli in più (spesso da 3 a 5);
    - solitamente molte decine di kbyte;
  - uno (interno o esterno), in grado di rispondere al massimo in un paio di decine di cicli aggiuntivi;
    - generalmente vari Mbyte.

Cache a più livelli (2)

- Il **primo livello** fornisce una grande velocità, a spese delle dimensioni.
  - La dimensione tende a restare invariata attraverso generazioni successive del processore, per non aumentare il numero di cicli per accesso.
  - Spesso a indirizzi virtuali, per non attendere la traduzione.
- Il **secondo livello** fornisce una buona velocità con dimensioni medie.
  - Le dimensioni spesso aumentano con l'evolversi del processore.
  - Indirizzi reali o virtuali a seconda dell'architettura.
- Il **terzo livello** fornisce una grande capacità e quindi un'alta percentuale di hit.
  - Le dimensioni spesso aumentano con l'evolversi del processore.
  - Generalmente operano con indirizzi reali.

Cache separate

- A volte la cache di primo livello è suddivisa in zone separate:
  - una per le istruzioni;
  - una per i dati.
- Spesso con dimensioni diverse.

Caratteristiche delle cache a zone separate

- La suddivisione evita che dei dati possano sovrascrivere istruzioni utili in un loop.
- Di fatto, si separano le linee candidate delle istruzioni da quelle dei dati, senza aumentare il numero di candidati per indirizzo.
- Scelta frequente nelle macchine con architettura Harvard (istruzioni e dati in memorie con bus di indirizzamento separati).

Gerarchie di memoria

- Di fatto il disco, grazie al meccanismo di swapping, è il livello inferiore di una gerarchia, che prosegue con RAM e cache.

Prefetch

- Tecnica nata per le istruzioni, che vengono lette sequenzialmente per grandi blocchi.
- Consiste nell'anticipare la lettura, mantenendo le istruzioni in una coda (detta di prefetch), dalla quale la CPU legge senza attendere .
- In alcune macchine la tecnica è stata estesa ai dati.
  - Quando la cache scopre 2-3 accessi a linee consecutive, inizia in anticipo la lettura della successiva.
  - In questo modo l'accesso a vettori avviene senza ritardi.

I/O

- Ogni dispositivo di I/O si interfaccia con la CPU tramite un insieme di registri:
  - **registri dati**, che contengono dati inviati dalla CPU al dispositivo o ricevuti dallo stesso;
  - **registri di configurazione e comando**, che contengono parametri di configurazione e comandi inviati dalla CPU;
  - **registri di stato**, che contengono informazioni sullo stato del dispositivo.

Registri di I/O

- Numero e tipo variano moltissimo da dispositivo a dispositivo.
  - I registri di stato spesso sono una collezione di campi di pochi bit, dai significati differenti.
- Per dispositivi funzionalmente identici, produttori diversi spesso definiscono insiemi di registri differenti.

Esempio di registri di una periferica: una seriale

- Il registro dati contiene il byte da trasmettere o ricevuto.
- I registri di configurazione fissano le caratteristiche del dispositivo: velocità, parità, bit di stop ecc..
- I registri di stato possono indicare pronta, byte arrivato, errore in ricezione.

Uso dei registri di I/O da parte del sistema

- I registri di configurazione vengono scritti dal sistema all'inizializzazione e modificati raramente.
- I registri di comando sono scritti per ogni operazione.
- I registri di stato e dati vengono letti a ogni operazione.

Requisiti hardware dell'I/O

- I registri di I/O possono richiedere:
  - tempi di accesso leggermente differenti da quelli della memoria;
  - un ordine di accesso fissato;
    - un registro può essere letto o scritto solo dopo un altro;
  - una certa granularità di accesso.
    - Molte periferiche richiedono che alcuni registri, raggruppati in un'unica parola, siano scritti in un unico ciclo e non separatamente.

Codifica degli accessi

- I linguaggi ad alto livello generalmente non garantiscono granularità e ordinamento degli accessi.
  - Un accesso potrebbe essere fisicamente spostato rispetto al punto nel quale è richiesto.
  - Spesso non possono neppure garantire che un accesso sia **effettivamente eseguito**.

Requisiti di sistema dell'I/O

- La paginazione deve essere **disabilitata**.
- La cache **non deve essere usata**.
- L'utente **non deve poter accedere ai registri di I/O**.

Accesso a registri di I/O

- Può essere effettuato in due modi:
  - con istruzioni apposite;
  - con accessi a indirizzi collegati ai registri (I/O memory mapped).

Istruzioni di I/O (1)

- A ogni registro di controllo delle periferiche viene assegnato un indirizzo.
  - Di solito lo spazio di indirizzamento di I/O è piccolo rispetto allo spazio di indirizzamento della memoria (es.: 2^{16}).
- Esistono istruzioni simili a istruzioni di lettura/ scrittura in memoria, ma con codici operativi differenti.
- Prassi frequente nei CISC e nelle CPU utilizzate per il controllo di processo.

Istruzioni di I/O (2)

- L'indirizzo del dispositivo viene inviato tramite il bus indirizzi (spesso utilizzato solo in parte).
  - Un apposito segnale del bus di controllo indica che si tratta di accesso ad I/O e non a memoria.
- Gli indirizzi di I/O possono **duplicare** quelli della memoria.
  - Esiste uno **spazio di indirizzamento di I/O**, completamente **separato** da quello della memoria.

Istruzioni di I/O (3)

- La CPU distingue tra accessi in memoria e di I/O e può applicare strategie differenti:
  - traduzione dell'indirizzo disabilitata;
  - cache disabilitata;
  - tempi di accesso diversi.
- Il basso livello di accesso è **necessariamente scritto in assembler**.
  - Nei linguaggi di alto livello è impossibile imporre la generazione delle istruzioni richieste.

Istruzioni di I/O (4)

- Si complica l'hardware.
  - Più istruzioni.
  - Bus complesso.
- Si semplifica leggermente il sistema:
  - paginazione e cache sono automaticamente disabilitate;
  - esecuzione sincrona e granularità di accesso sono garantite;
  - la protezione è garantita, perché le istruzioni sono eseguibili **solo in modo kernel**.

I/O memory mapped (1)

- I registri di controllo delle periferiche equivalgono a parole di memoria.
- La CPU **non distingue** tra accessi in memoria e di I/O.
- Comune nei RISC.

I/O memory mapped (2)

- L'indirizzo del dispositivo viene inviato tramite il bus indirizzi.
  - Nulla indica sul bus che si tratta di accesso ad I/O e non a memoria.
- Gli indirizzi di I/O devono essere **distinti** da quelli della memoria.
  - Esiste un **unico spazio di indirizzamento**.
  - Gli indirizzi usati per l'I/O sono normalmente raggruppati in una zona ristretta dello spazio di indirizzamento, all'inizio o alla fine.

I/O memory mapped (3)

- Si semplifica l'hardware:
  - meno istruzioni;
  - bus più semplice;
  - tempi di accesso diversi rispetto alla memoria possono essere ottenuti solo tramite hardware esterno.
- Si complica leggermente il sistema.
  - La gestione è interamente software.

I/O memory mapped (4)

- Il sistema, utilizzando i meccanismi di paginazione/ segmentazione deve garantire:
  - che gli indirizzi di I/O **non siano tradotti** da virtuali a reali (o siano tradotti in modo fisso);
  - che le celle di memoria corrispondenti **non siano conservate in cache**.
- Inoltre il sistema deve garantire, utilizzando i meccanismi di protezione, che le aree corrispondenti all'I/O **non siano accessibili in modo utente**.

I/O memory mapped (5)

- Il livello inferiore è spesso scritto in assembler.
  - Nei linguaggi di alto livello è generalmente impossibile garantire accessi sincroni e granularità di accesso.
  - Dichiarazioni come `volatile` in C e C++ permettono di garantire **l'ordinamento** degli accessi e **l'effettiva esecuzione** dell'accesso, non la granularità.

Differenti filosofie per l'I/O

- I due metodi possono coesistere.
  - Nulla vieta di utilizzare I/O memory mapped anche in presenza di istruzioni di I/O.
    - Esempio tipico: Intel 80*86.
- L'I/O memory mapped è la scelta normale con processori RISC moderni e con macchine ad alte prestazioni.