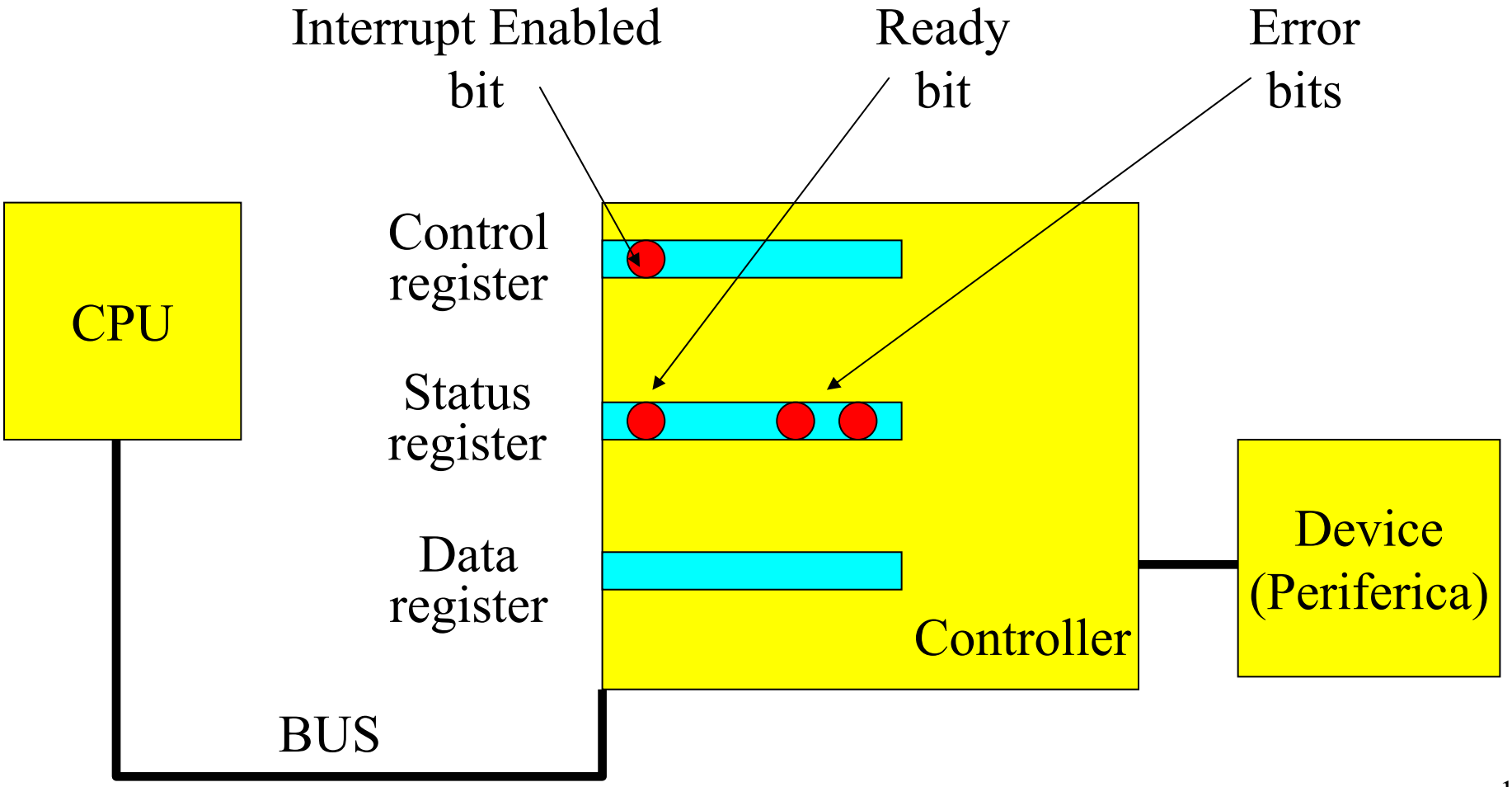
ISA – Input / Output (I/O)

Numerose Periferiche di tanti tipi diversi, collegati alla CPU mediante BUS diversi.

- **Solo Input** (tastiera, mouse), producono dati che la CPU deve leggere.
- **Solo Output** (Schermo), accettano dati da usare (spedire, immagazzinare,..)
- **Input e Output** (Rete, Dischi,..).



ISA – Controller delle Periferiche e Registri del controller

I **Controller delle Periferiche** forniscono alla CPU un'interfaccia comune di accesso a periferiche diverse. Tale interfaccia è formata da Registri dei controller i cui singoli bit hanno funzionalità specifiche diverse.
Esistono tre tipi di registri nei controller:

- **Registri di Controllo** (sola scrittura, accettano comandi e impostazioni).
  - **interrupt Enabled bit**, se settato, fa eseguire un Interrupt quando la periferica cambia il suo stato, cioè quando il Registro di Stato cambia valore viene lanciato un Interrupt per avvisare la CPU che è avvenuto un evento da gestire.
- **Registri Dati** (**Buffer** di lettura e/o scrittura, per scambio dati con periferica).
  Quando le perif. di input producono un dato lo mettono qui in attesa che sia letto.
  Quando si vuole mandare un dato su perif. di output lo si mette qui.
- **Registri di Stato** (sola lettura, danno info sullo stato della periferica, errori, ..)
  - **Ready bit**, settato quando la periferica è pronta ovvero:
    - Periferica di Input, **c'è un dato da leggere nel buffer dati**
      - es: carattere digitato or ora su tastiera da un utente.
    - Periferica di Output, **il buffer è vuoto, può accettare altri dati**
      - es: buffer disposto a ricevere stringa da mandare sullo schermo.
  - **Error bits**, settati quando accade errore.
    - es: sovrascrittura del buffer dati prima che il suo contenuto sia stato utilizzato.

ISA – I/O – Operazioni sul Controller

La tipica operazione sul controller viene svolta in tre parti:
• **Ordine** : viene posto sul registro di controllo un byte (il comando) che ordina al controller di eseguire una determinata operazione, oppure sul registro dati un valore. L'operazione termina quando il valore è memorizzato sul registro.
• **Attesa** : bisogna aspettare che l'ordine venga eseguito, poiche' le periferiche sono distanti e spesso lente. Spesso la durata dell'attesa è impredicibile
• **Verifica** : si legge il registro di stato per verificare se l'operazione è andata a buon fine oppure se c'e' stato un errore.

L'attesa può essere attiva, nel senso che la CPU può verificare ripetutamente il valore del registro di stato, fino ad ottenere l'informazione sull'esito del comando impartito. Ciò comporta un inutile lavoro della CPU.

In alternativa, la CPU può configurare il controller affinchè generi un interrupt per notificare alla CPU che lo stato del controller è cambiato, dopo di che la CPU verifica il registro di stato.

Notare che ordine e verifica vengono effettuati operando su due diversi registri del controller, ciò come vedremo comporterà dei problemi qualora si voglia usare i registri come se fossero delle locazioni di memoria (vedi memory mapped I/O).

ISA – Spazio degli Indirizzi per Operazioni di I/O

Il computer può far vedere i registri di un controller o come se fossero delle locazioni di una periferica (I/O) oppure come se fossero delle locazioni di memoria.

• **separate I/O Address Space** : per accedere al registro bisogna dire al BUS che si sta accedendo ad una periferica di I/O e specificarne l'identificativo.

- Si utilizzano delle particolari istruzioni del livello ISA, dette **IN** (per leggere un valore di un registro) ed **OUT** (per mettere sul registro un valore.
- Tali istruzioni terminano solo dopo che i valori sono stati effettivamente collocati nella destinazione specificata. Sono quindi operazioni lente (come le periferiche) ma assicurano che quando la CPU vede il termine dell'operazione il registro sia stato acceduto fisicamente.

• **Memory Mapped I/O** : si tratta il registro della periferica come se fosse una qualunque locazione di memoria. Al registro viene fatto corrispondere un indirizzo di memoria. Quando si cerca di accedere a quella locazione in realtà si accede al registro.

- L'accesso viene effettuato utilizzando la normale istruzione di accesso a memoria **MOV**, specificando l'indirizzo di memoria a cui corrisponde il registro.
- Poiché tale istruzione mette in moto il meccanismo della cache, l'accesso memory mapped è più veloce dell'accesso I/O vero e proprio, ma l'istruzione MOV per scrivere su un registro può terminare senza che il registro sia stato effettivamente modificato, e la modifica avverrà successivamente.

ISA – Svantaggi del modello Memory Mapped I/O
Inversione dell'ordine delle operazioni

Quando per l' I/O si usa l'istruzione MOV, si usa il modello memory mapped I/O.

L'uso della MOV abilita l'utilizzo della cache che può generare questo problema:

- Se devo mettere un byte sul registro di controllo del controller di una periferica e poi devo verificare il risultato leggendo il registro di stato, posso fare prima una MOV in scrittura sul registro di controllo e poi una MOV in lettura sul registro di stato.

- Purtroppo, i due registri appaiono come due diverse locazioni di memoria, quindi la cache non vede nessun tipo di interazione tra i due registri, quindi la cache potrebbe dilazionare l'accesso in scrittura sul registro di controllo e accedere immediatamente al registro di stato, invertendo di fatto l'ordine delle operazioni.

- Potrebbe cioè avvenire che la lettura del registro di stato per verificare l'effetto del comando messo nel registro di controllo venga fatta prima che quel comando venga posto sul registro di controllo, ottenendo un risultato impredicibile e dipendente dal precedente stato del registro di stato.

ISA – Tipologia di Accesso per I/O (1/2)

Lettura da periferica : Per copiare una word (o un byte) da una periferica verso la memoria, la CPU deve eseguire una istruzioni IN copiando la word (il byte) da un registro dati del controller in un registro della CPU e poi da questo in una data locazione di memoria. Per svolgere tale operazione la CPU deve **aspettare** che la periferica abbia prodotto la word (il byte) e lo abbia collocato nel registro dati e poi abbia settato il bit Ready ad indicare la disponibilità del dato in lettura.

Scrittura su periferica : Per copiare una word (o un byte) dalla memoria in una periferica, la CPU deve copiare ciascuna word in un registro della CPU, poi deve **aspettare** che il buffer dati della periferica sia libero ed abbia settato il bit Ready, dopo di che la CPU eseguirà una istruzioni OUT copiando la word dal suo registro nel registro dati del controller.

Se devono essere trasferite più words questi procedimenti devono essere ripetuti.

Attesa per Disponibilità : riassumendo, per svolgere un'operazione di lettura o scrittura di I/O la CPU deve aspettare che il controller sia disponibile a svolgere l'operazione (lettura: byte nel buffer, scrittura: buffer vuoto) ed abbia notificato la disponibilità settando il proprio bit Ready.
La gestione di questa fase di attesa rappresenta una problematica importante dell' I/O perché impegna la CPU costringendola ad un lavoro ripetitivo finchè non viene individuata la disponibilità a procedere del controller.

ISA – Tipologia di Accesso per I/O (2/2)

Le operazioni di I/O possono essere organizzate in tre diversi modi, senza o con l'ausilio di HW specializzato (il DMA), per ridurre il carico di lavoro della CPU.

• **Attesa Attiva (Busy Waiting) (Programmed)** : la CPU effettua tante letture ripetute dello status register per capire quando il bit Ready segnala che l'operazione può essere svolta. Tali letture impegnano la CPU per tutta la durata dell'attesa, sprecando cicli di clock.

• **Gestione ad Interrupt** : la CPU ordina al controller di abilitare gli interrupt, poi fa altre cose. Quando l'operazione di I/O diventa possibile, il bit Ready viene settato e, poiché lo status register ha cambiato valore, il controller lancia un interrupt che avvisa la CPU. La CPU controlla lo status register, vede settato il bit Ready e fa l'operazione. Non c'e' spreco di risorse di CPU, ma per ogni word da trasferire deve essere la CPU ad ordinare l'operazione IN o OUT. Non c'è passaggio diretto tra memoria ed I/O ma si passa attraverso un qualche registro della CPU che risulta perciò impegnata.

• **Basato su DMA** : Dovendo trasferire più words in I/O, la CPU ordina al DMA di eseguire tutte le operazioni, indicando quante word trasferire, l'indirizzo d'inizio dell'area di memoria in cui mettere/prendere le words e la periferica a cui accedere. Poi la CPU può fare altro, mentre il DMA trasferisce i dati senza impegnare la CPU (e i registri della CPU) ma impegnando il BUS (il DMA ha più priorità della CPU nello accesso al BUS). Finito il trasferimento il DMA lancia un interrupt per avvisare la CPU della conclusione delle operazioni.