I/O : teoria

Efficienza della Gestione

- **Banda passante:** quantità di dati che si può trasferire per unità di tempo, misura di flusso (frazione)
  (B/s) Mbit/s)

- **Latenza:** tempo che intercorre tra l'istanza Ready, che indica che la periferica è pronta, e l'istante effettivo in cui inizia il trasferimento, misura di tempo (ms)

3 Modalità di Gestione I/O:

Problema: la CPU esegue ininterrottamente istruzioni ad una certa frequenza, mentre le periferiche solo in certi momenti ad una certa frequenza.

- **Controllo di programma:** la CPU controlla il valore del registro di stato della periferica e lo copia nello spazio di indirizzamento in cui è mappato ad un registro della CPU.
  Se la periferica è pronta inizia il trasf. dei dati, altrimenti la CPU aspetta che la periferica sia pronta.

+ PRO: banda passante alta, bassa latenza. Economico.
- CONTRO: busy wait: la "CPU" rimane in stallo senza fare altro.

- **Interrupt:** ogni volta che una periferica svolge un'azione viene generato un'interrupt. Nel MIPS32 i reg. periferica sono mappati in memoria e ogni periferica ha due registri:

- control reg: 1 bit per possibilità di generare interrupt, 1 bit ready
- data reg: contiene il dato da trasferire. (8 bit)

Essendo reg. mappati in memoria vi si accede tramite lw/sw

+ PRO: Si evita il busy wait, poca circuiteria aggiunta

- CONTRO: Prestazioni peggiorate rispetto a cont. programma.

- **Dma**: Direct memory Acess: permette di superare i contro del busy wait della gest. programma e i probl. di efficienza dell'interrupt. Con il Dma si rende la cpu libera, che potrà fare altro durante il trasferimento. Sarà il DMA ad occuparsi del trasferimento. Quando esso finirà, verrà mandato un segnale di interrupt alla CPU per informarla della fine del trasferimento.

Il Dma richiede circuiteria apposita, la quale viene realizzata nella periferica. Dunque non ci sarà solo la CPU protagonista, ma anche le periferiche. Servirà quindi un nuovo componente per gestire il controllo del bus di sistema. (DMA Controller)

+PRO: + Efficiente in termini di uso della CPU (gestisce solo inizio e fine)
+ No busy wait. + Prestazioni

+ CONTRO: Costoso a livello di circuiteria

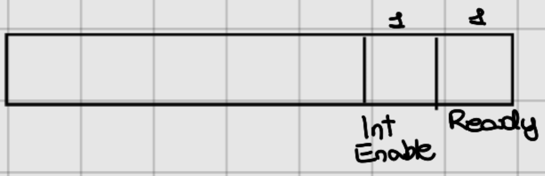
Registri per trasferimento

Per il controllo I/O sono necessari 4 registri memory mapped.

- receiver: riceve l'input dal terminale
- transmitter: trasmette l'input alla console

- **Receiver control**

0xffff0000



Ready bit:

- read-only

- 0 → 1 quando viene digitato un carattere
- resta 1 fino al prelevamento del dato

Int Enable :

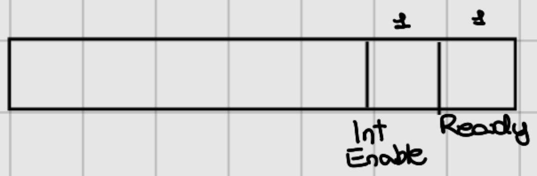
- readable and writeble
- settato a 0
- se il programma lo mette a 1 viene generato interrupt per ogni carattere
- perché abbiano effetto anche nel processore, gli interrupt devono essere abilitati nello Status Register

- **Receiver Data**
  0xffff0004

contiene il codice ASCII dell'ultimo carattere digitato

- read-only
- cambia quando viene digitato un nuovo carattere
- codice ASCII valido se Ready = 1
- La lettura riporta Ready = 0

- **Transmitter Control**
  0xffff0008



Ready bit :

- read-only
- 0 = il Transmitter sta ancora scrivendo
- 1 = il Transmitter è pronto a ricevere un nuovo valore

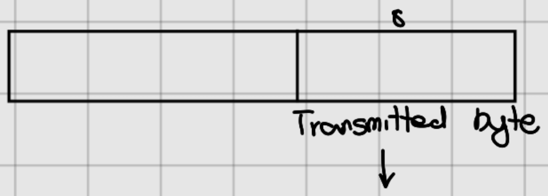
Int Enable :

- readable and writeble

- se 1, terminale richiede interropt se Ready bit = 1
- se 1 ma Ready = 0 il carattere andrá perso.

- **Transmitter Data**

0xffff000c



- bit trasmessi alla console. Ready bit a 0 fino al completamento. Dopo torna 1
- Può essere scritto se Ready = 1, altrimenti i dati vengono ignorati