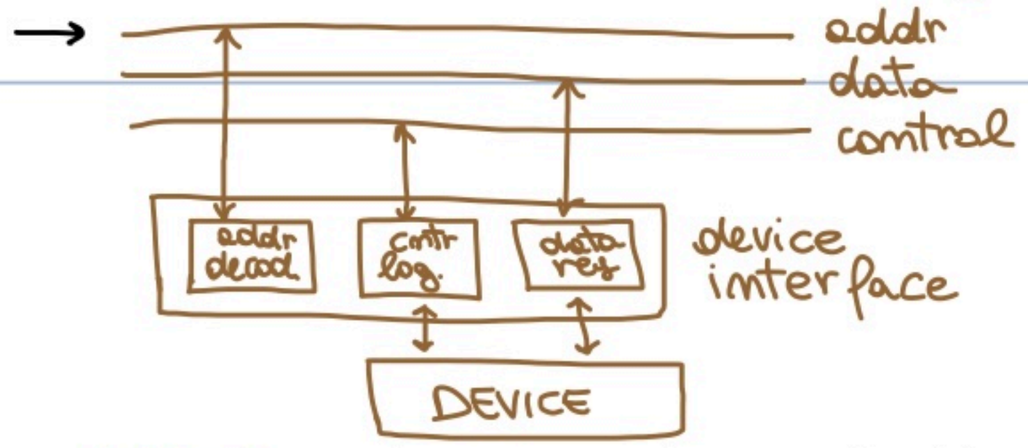
ACCESSO ALLE PERIFERICHE

→ periferiche connesse a microprocessori tramite **bus di sistema**:
↳ possono essere moltep. ma consid. caso 1 bus

- **INTERFACCIA DI UNA PERIFERICA**: per poter scambiare dati con µP la periferica:
  → essere identif tra tutte; (avere indr)
  → specificare tipo di trasf. (Io/SW); (mecc. di cntr)
  → prelevare o scrivere dati; (mem. locale)
- **bus di sistema** → linee raggruppate in: indirizzo, dati, controllo;



- address decoder → decod. indirizzo per dtt. periferica
- control logic → realizza logica di controllo per qp.
- data & status reg → reg che contengono dati e info stato perif.

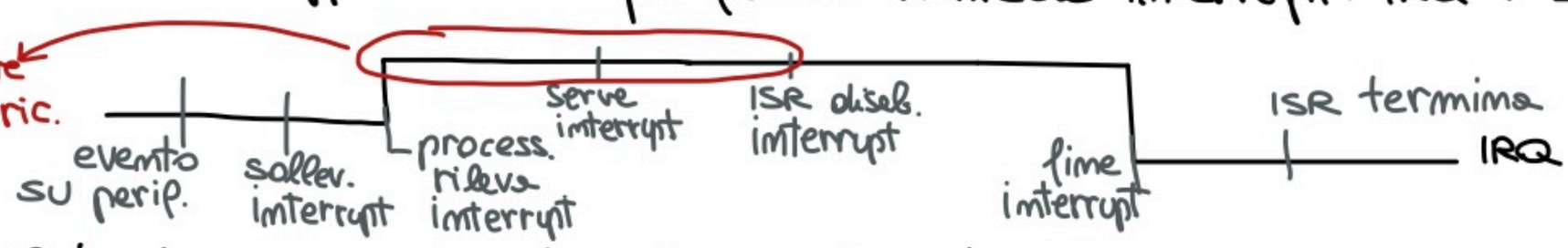
↳ i registri di una periferica sono "acceduti" dal processore come indirizzi normali;

→ **es.** interfaccia di una tastiera: reg dati TTY_D contiene carattere
reg stato TTY_S contiene stato perif.
reg controllo TTY_C permette di contr. la perif.

- **POLLING** → metodo più semplice di accesso ad una periferica:
il programma controlla periodic. il reg di stato nel caso vi siano dati da prelevare;
↳ poca efficienza
- **INTERRUPT** → migliora accesso a periferiche, ovvero il programma consente alle perif. di sollevare una interrupt se necessitano trasmettere dati;
↳ con relativo ISR

ISR: salva PC e status reg → disab. altri interrupt → salva reg. su stack → esec. op. richieste →
↳ ripristina reg. su stack → cancella flag e riab. altri interrupt → **IRET** (ripristina PC e status reg)

→ **logica interrupt**: utilizza 1 linea IRQ (segnale)
↳ non appena una periferica richiede interrupt: IRQ → 1



SOLUZIONE: disabilitazione semi-automatica interrupt;
disabilitazione automatica interrupt;

→ per identificare interrupt e chi l'ha sollevata: → linee di interrupt specifiche;
→ identif. della periferica;
→ daisy-chain;

2→ identific. periferica: quando una perif vuole farsi riconoscere asserisce la linea di interrupt, pone sul bus un codice identif;
NON CONSENTE DI GEST. INTERRUZ. MULT.

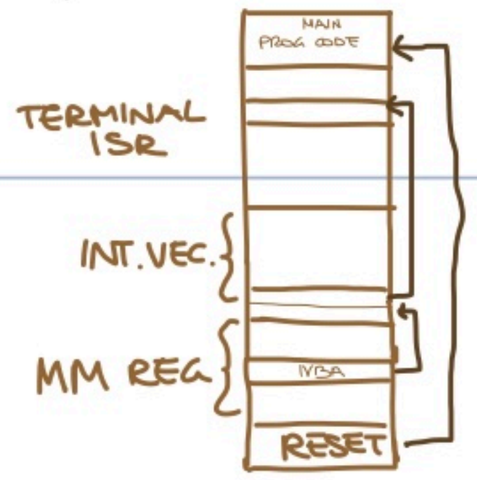
3→ **daisy-chain**: linea aggiuntiva detta IACK utilizzata per propagare una richie. di interrupt alle periferiche. La prima che manda IRQ blocca IACK verso le altre

→ POSIZ. IN BASE A PRIORITÀ
↳ più perif. portano a più linee IRQ/IACK

1 → linee specifiche: ogni device dispone di due linee che finiscono direttamente a un interrupt controller nel μP;

• IDENTIF. ROUTINE DI SERV → il codice sul bus identifica (tramite indirizzo) ISR nella sua tabella "vectored interrupt" la cui posizione è salvata in un registro (memory mapped) IVBA

↳ in questo caso la memoria è composta:



→ reset vector, specificano gli/il entry point dell'appl. o S.O.
→ memory-mapped reg, banco registri periferiche e config. processore
→ interrupt vector, tabella con indirizzi ISR
→ ISR, codice nel caso di interrupt
→ main code, codice appl. e S.O.

→ cosa succede se sto servendo una interrupt e me arriva un'altra?

1 → se il dispositivo non supporta gestione multipla interrupt viene ignorata; (poco flessibile)

2 → ritardo nuova interrupt (delayed interrupt); (poco flessibile)

3 → richiesta supporto interrupt annidati (interrupt nesting) in modo da rispettare priorità; (molto flex, complessa realizzazione)
↳ per gestire priorità differenti: → se ho 1 liv. nesting mi basta gestirle tram 1 registro;
→ più livelli nesting portano ad un meccanismo basato su stack;

• NMI → (non-mask.-inter.) interrupt che non si possono ignorare (debugging o situazioni di errore grave)

→ DIRECT MEMORY ACCESS: tecnica trasmissione dati da/verso periferiche e modulo che fa parte del MIPS nel caso di quantità elevate di dati;
↳ si tratta di un trasferimento diretto alla mem. (processore è libero di fare altro)

→ TRASFERIMENTO:
• processo P richiede operazione in DMA;
• S.O. esegue un nuovo processo mentre P è in attesa;
• DMA agisce da bus master e si interfaccia con disp. specifico e memoria; (formata da diversi registri di interfaccia)
• DMA controller solleva interrupt a fine trasferimento;
• S.O. intercetta interrupt e risveglia P;
• P ottiene i dati dall'op. in DMA;

↳ perché DMA è comodo?
esegue in autonomia trasferimento lasciando libero il processore.
Nel caso in cui vi sia solo 1 bus non risulta comunque ottimale:
sono necessari 2 bus: mem bus e peripheral bus e memoria dual-port