

Corso di Calcolatori Elettronici I
A.A. 2012-2013

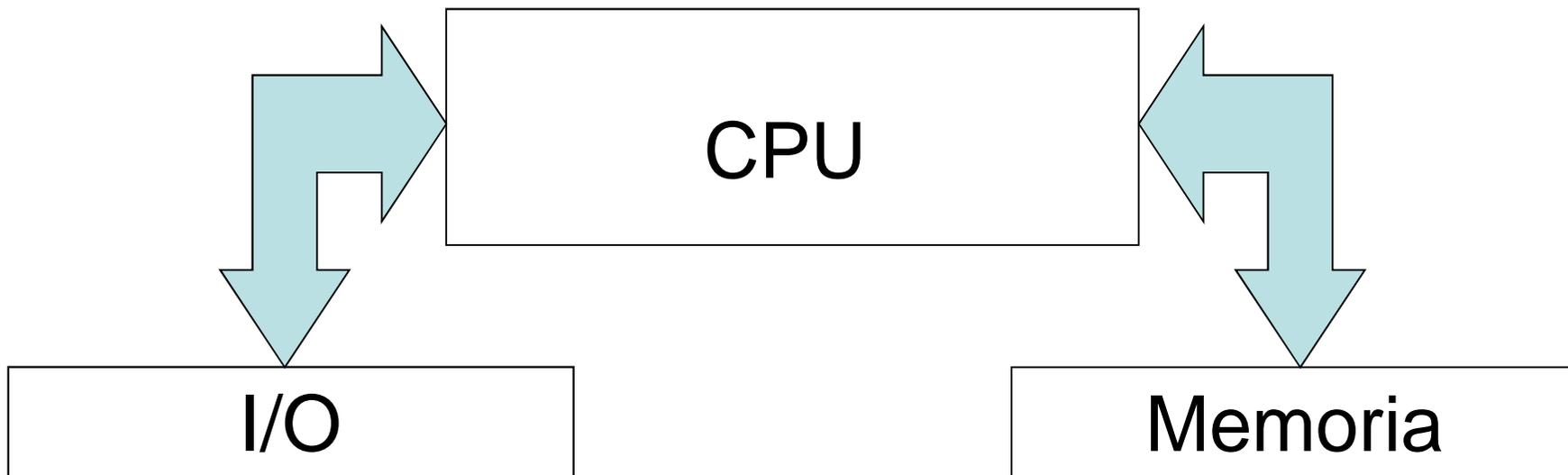
Sottosistema di
Ingresso/Uscita (I/O)

ing. Alessandro Cilaro

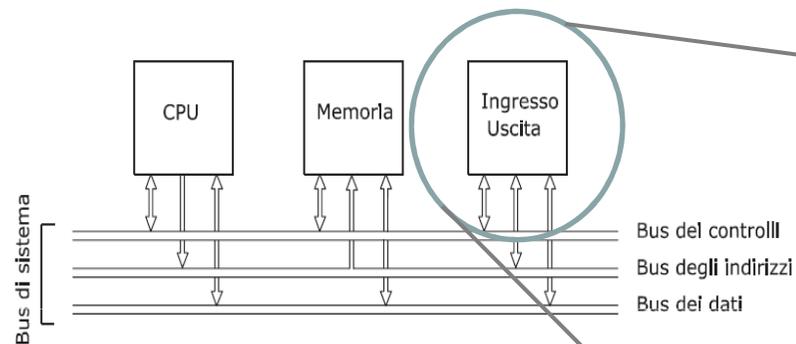
Accademia Aeronautica di Pozzuoli
Corso Pegaso V "GArn Elettronici"

Calcolatore: sottosistemi

- Processore o CPU (*Central Processing Unit*)
- Memoria centrale
- Sottosistema di input/output (I/O)



Sottosistema di I/O



Esempi di componenti di I/O

- tastiera
- monitor
- dispositivi di puntamento
- periferiche USB
- disco rigido (memoria di massa)
- timer
- gestore delle interruzioni
- etc etc



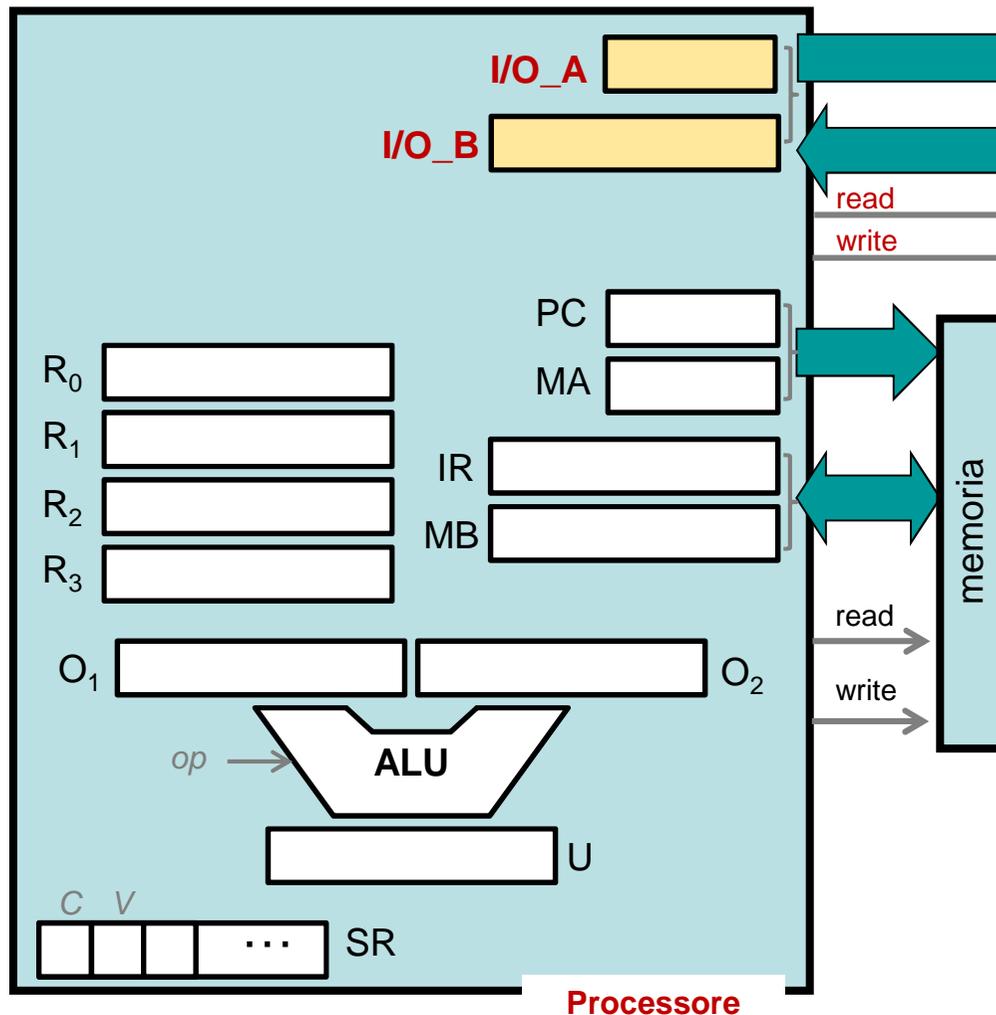
Sottosistema di I/O

- **Periferiche** (*peripherals*) di Ingresso/Uscita, o Input/Output (**I/O**)
 - praticamente tutti i componenti esterni all'interazione diretta processore-memoria
 - Esempi: tastiera, monitor, dispositivi di puntamento, periferiche USB, disco rigido (memoria di massa), timer, gestore delle interruzioni, Controllore del Direct Memory Address (DMA), etc etc

Interfacciamento con I/O

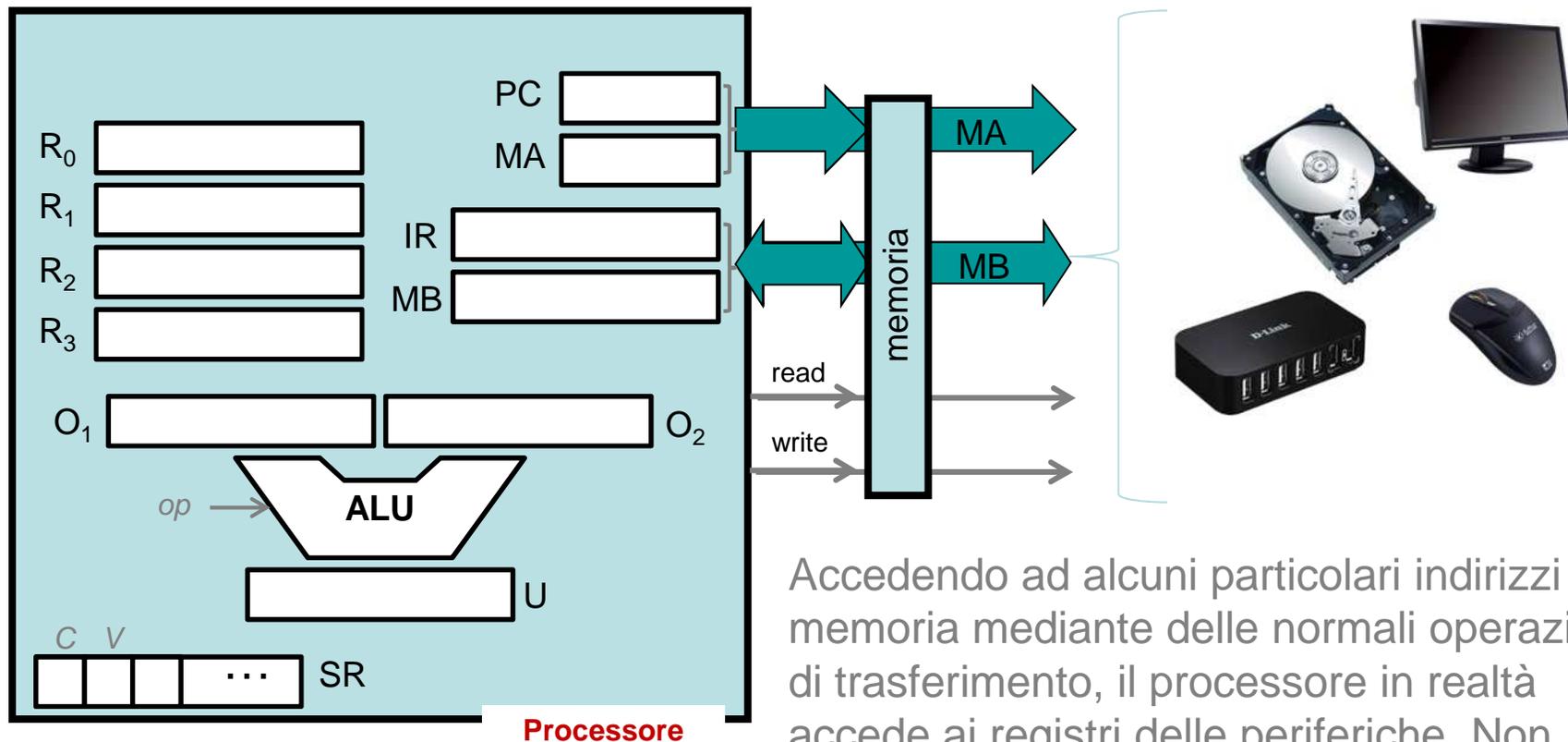
- Due approcci per l'interfacciamento tra processore e dispositivi di I/O
- “I/O mapped” I/O
 - il processore dispone di un'interfaccia indirizzo/dato separato per l'I/O (es. Intel x86)
 - i dispositivi sono in uno spazio di indirizzamento separato e sono presenti istruzioni apposite
- “memory-mapped” I/O
 - i dispositivi sono attivati da alcuni specifici indirizzi,
 - per il processore, gli accessi ai dispositivi di I/O sono visti come operazioni in memoria (es. M68000)

I/O-mapped I/O



il processore dispone di un'interfaccia indirizzo/dato separato per l'I/O (es. Intel x86)
Ad ogni dispositivo è associato un indirizzo, in uno spazio di indirizzamento diverso da quello della memoria
Sono presenti istruzioni apposite che il processore può eseguire per accedere alle periferiche

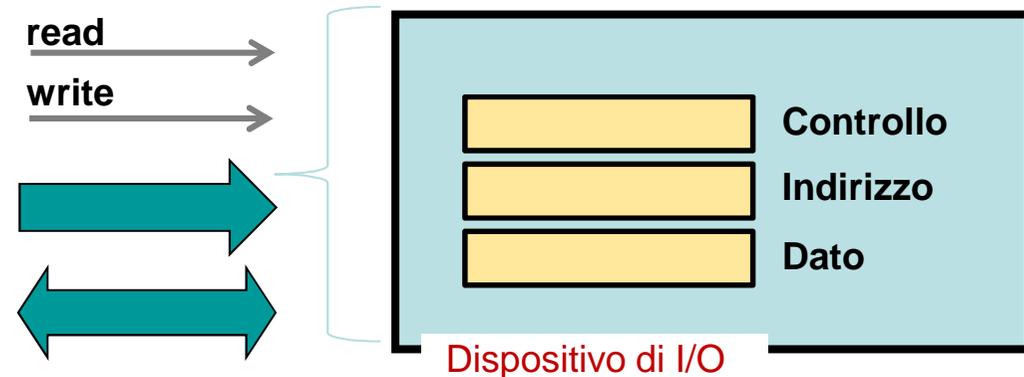
memory-mapped I/O



Accedendo ad alcuni particolari indirizzi di memoria mediante delle normali operazioni di trasferimento, il processore in realtà accede ai registri delle periferiche. Non servono istruzioni specifiche per l'I/O

Interfaccia della periferica

- Tipicamente, l'interfaccia di qualsiasi dispositivo è composta da *registri*
 - locazioni, simili a quelle in memoria, da cui è possibile leggere o su cui è possibile scrivere
- Tre tipi di registri
 - registri di **controllo e stato**
 - registri **indirizzo**
 - registri **dato**



Interfaccia della periferica

- Registri Controllo/Stato
 - **Controllo**: scritti dal processore, determinano mediante opportune codifiche (diverse da periferica a periferica) cosa il dispositivo deve fare
 - **Stato**: scritti dalla periferica e letti dal processore, danno la possibilità al dispositivo di informare il processore circa l'esito delle operazioni effettuate

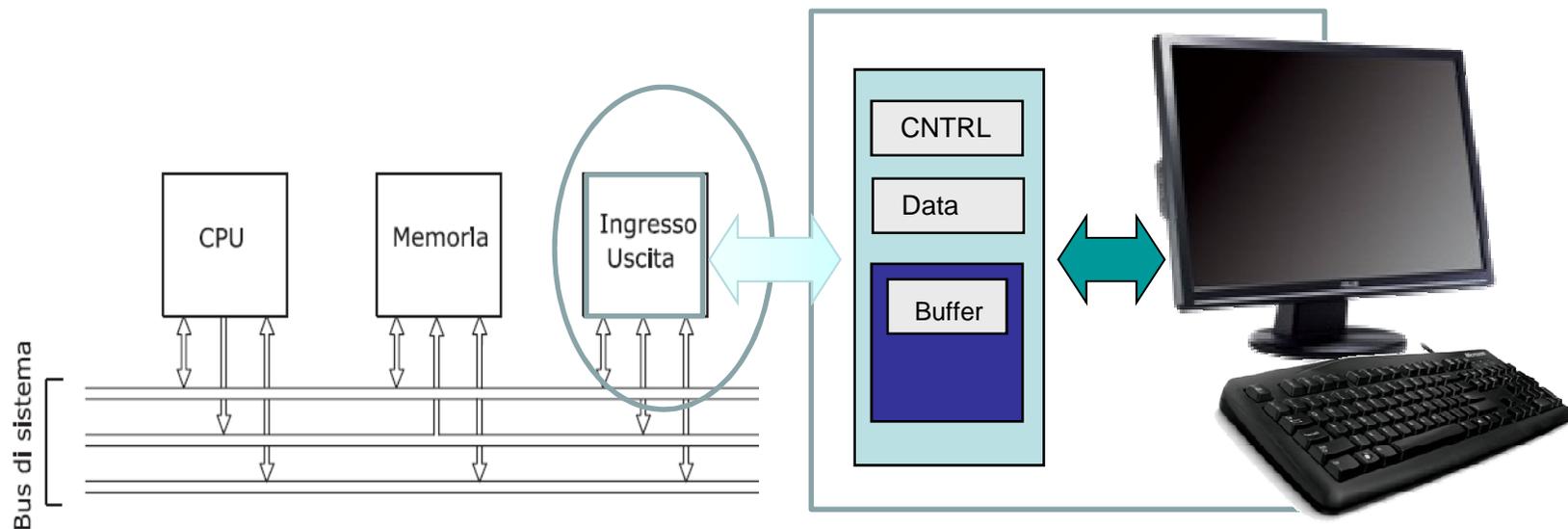
Interfaccia della periferica

- **Registri Indirizzo e Dato**
 - hanno lo stesso significato di quelli usati per la memoria, ma fanno riferimento ad un insieme (tipicamente ridotto) di locazioni fisicamente presenti nella periferica

Esempio: terminale

File: programma024.a68

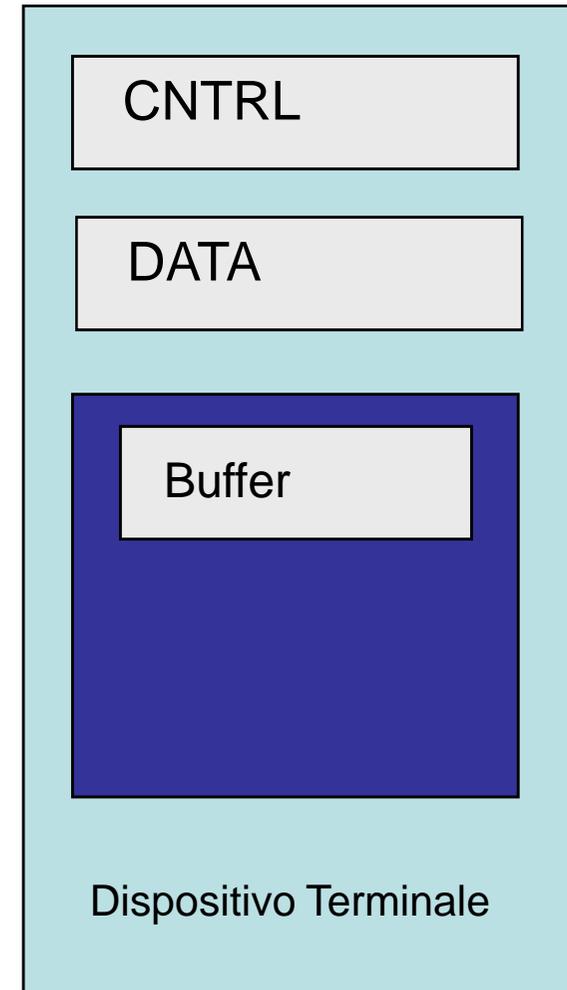
- Terminale
 - rappresenta un dispositivo combinato Monitor/Tastiera (cosiddetta *console*)



Esempio: terminale

File: programma024.a68

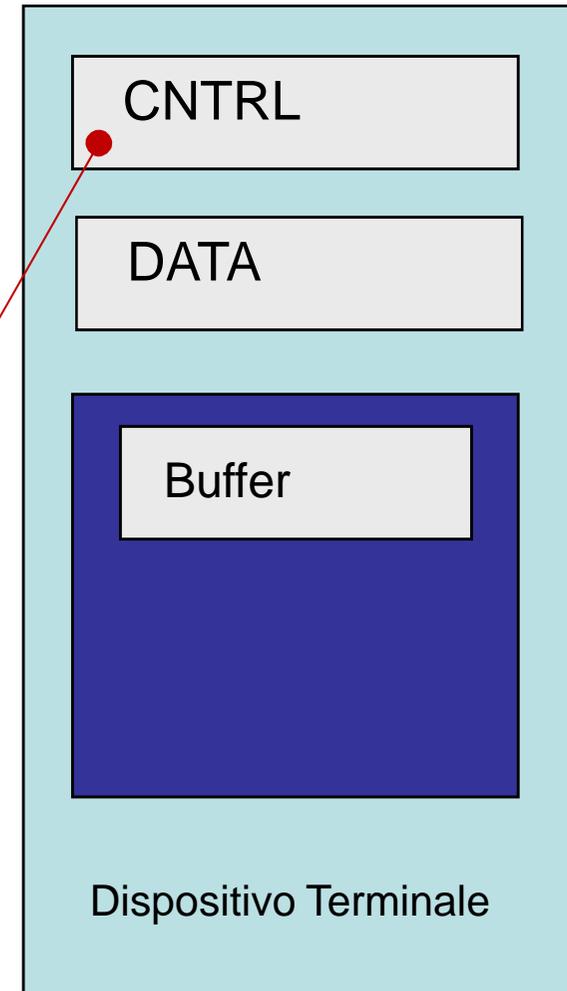
- Terminale: **Interfaccia**
 - un indirizzo combinato di Controllo/Stato (**CNTRL**)
 - un registro dati (**DATA**) su cui scrivere i dati che andranno su schermo o leggere quelli scritti da tastiera
 - contiene un *buffer* interno per contenere i dati immessi da tastiera prima che vengano trasferiti



Esempio: terminale

File: programma024.a68

- **CNTRL** (registro di controllo e stato):
 - **Bit 0**: Abilita interruzione su BUFFER FULL
 - **Bit 1**: Abilita interruzione su ENTER
 - **Bit 2**: Cancella video
 - **Bit 3**: Pulisce buffer tastiera
 - **Bit 4**: Abilita tastiera
 - **Bit 5**: Abilita eco
 - **Bit 6**: Stato di BUFFER FULL
 - **Bit 7**: Stato di ENTER inviato



Esempio: terminale

File: programma024.a68

- **CNTRL** (registro di controllo e stato):

- **Bit 0**: Abilita interruzione su BUFFER FULL
- **Bit 1**: Abilita interruzione su ENTER
- **Bit 2**: Cancella video
- **Bit 3**: Pulisce buffer tastiera
- **Bit 4**: Abilita tastiera
- **Bit 5**: Abilita eco
- **Bit 6**: Stato di BUFFER FULL
- **Bit 7**: Stato di ENTER inviato

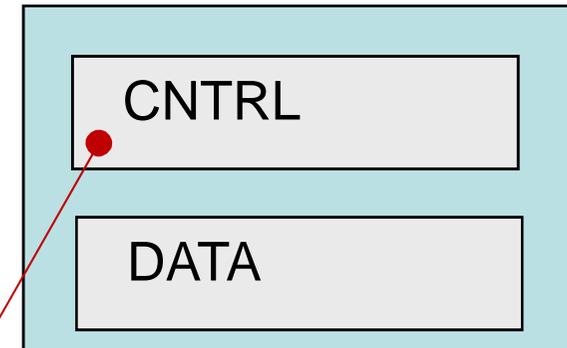
I Bit da **0** a **5** sono bit di controllo (determinano un'azione o una configurazione della periferica)

I Bit **6** e **7** sono bit di stato (indicano al processore cosa sta succedendo nella periferica)

Esempio: terminale

File: programma024.a68

- **CNTRL** (registro di controllo e stato):
 - **Bit 0**: Abilita interruzione su BUFFER FULL
 - **Bit 1**: Abilita interruzione su ENTER
 - **Bit 2**: Cancella video
 - **Bit 3**: Pulisce buffer tastiera
 - **Bit 4**: Abilita tastiera
 - **Bit 5**: Abilita eco
 - **Bit 6**: Stato di BUFFER FULL
 - **Bit 7**: Stato di ENTER inviato

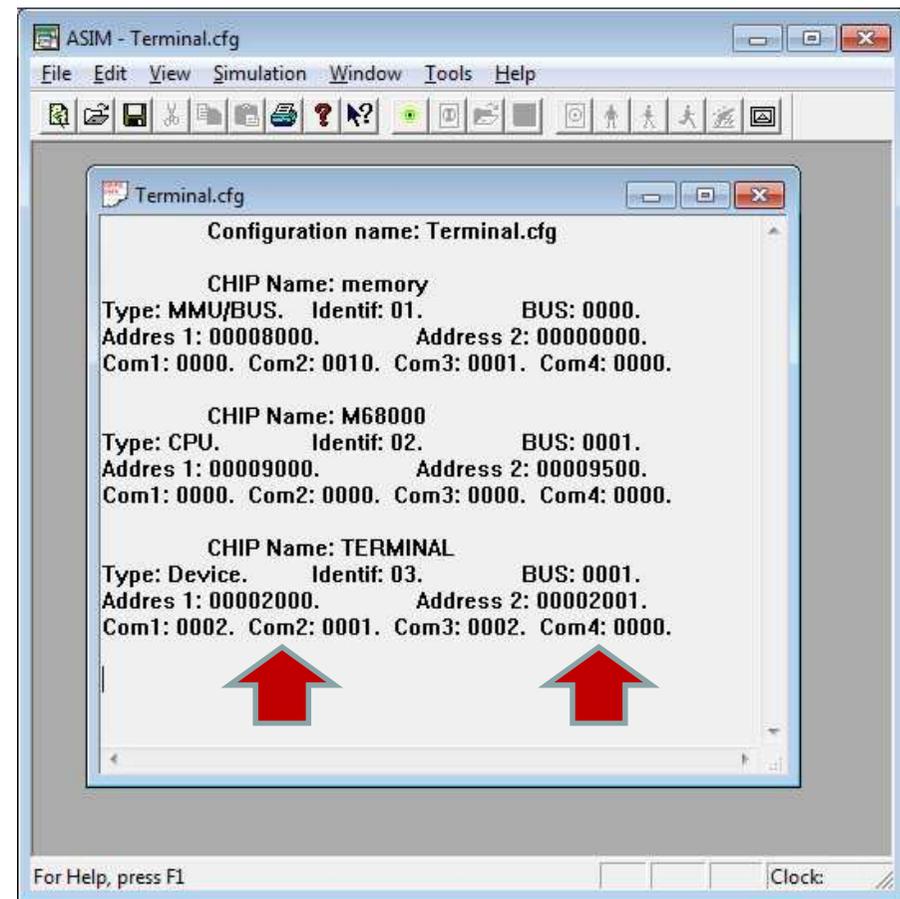


Ad esempio, per chiedere al terminale di **cancellare il contenuto del video**, dobbiamo scrivere il valore **1** nel **bit 2** del registro **CNTRL**.
Se ad esempio **CNTRL** è mappato all'indirizzo **\$2001** dobbiamo eseguire l'istruzione **OR.B #4,\$2001**

Esempio: terminale

File: programma024.a68

- Richiede un file di configurazione in ASIM che comprenda anche la periferica Terminale
 - File `Terminal.cfg`
 - La configurazione descrive un sistema che comprende, oltre alla parte base processore+memoria, anche una periferica *memory-mapped*
 - Sono indicati gli indirizzi cui sono “mappati” i due registri della periferica



Esempio: terminale

File: programma024.a68

```
MEM      ORG $8000
          ORG $8200
TER      EQU $2000
EOS      EQU 0
ENTER    EQU 13
ENTERHIT EQU $80  10000000
TTYCFG   EQU $30  00110000
QUEST    DC.B    'Come ti chiami ? ',0
ANSW     DC.B    'Ciao, ',0

BEGIN    MOVEA.L #TER,A2
          MOVEA.L A2,A0
          ADD.L   #1,A2
          MOVE.B #TTYCFG,(A2)
          LEA.L   QUEST,A1
OUTPUT   MOVE.B  (A1)+,D0
          CMP.B  EOS,D0
          BEQ    CONT
          MOVE.B D0,(A0)
          BCC   OUTPUT
CONT     MOVE.B  (A2),D1
          AND.B #ENTERHIT,D1
          BEQ    CONT

          MOVEA.L #MEM,A1
IN1      MOVE.B  (A0),D0
          MOVE.B D0,(A1)+
          CMP.B  #ENTER,D0
          BNE   IN1
          LEA.L  ANSW,A1
OUT1     MOVE.B  (A1)+,D0
          CMP.B  #EOS,D0
          BEQ   CONT1
          MOVE.B D0,(A0)
          BCC   OUT1
CONT1    MOVEA.L #MEM,A1
OUT2     MOVE.B  (A1)+,D0
          MOVE.B D0,(A0)
          CMP.B  #ENTER,D0
          BNE   OUT2
FINE     BRA    FINE
END      BEGIN
```

Esempio: terminale

File: programma024.a68

```
MEM      ORG $8000
         ORG $8200
TER      EQU $2000
EOS      EQU 0
ENTER    EQU 13
ENTERHIT EQU $80    10000000
TTYCFG   EQU $30    00110000
QUEST    DC.B      'Come ti chiami ? ',0
ANSW     DC.B      'Ciao, ',0

BEGIN    MOVEA.L #TER,A2
         MOVEA.L A2,A0
         ADD.L   #1,A2
         MOVE.B #TTYCFG,(A2)
         LEA.L  QUEST,A1
OUTPUT   MOVE.B (A1)+,D0
         CMP.B  EOS,D0
         BEQ   CONT
         MOVE.B D0,(A0)
         BRA   OUTPUT
CONT     MOVE.B (A2),D1
         AND.B #ENTERHIT,D1
         BEQ   CONT
```

Riserva **512** locazioni (**200** esadecimale) vuote

Indirizzo del registro **DATA** del terminale (**\$2000**)

Maschera che individua il bit di **CNTRL** usato per rilevare quando è stato premuto il tasto **ENTER**

Costante che sarà scritta in **CNTRL** per configurare inizialmente il terminale

Inizializza **A0** con l'indirizzo del registro **DATA** (**\$2000**)

Inizializza **A2** con l'indirizzo del registro **CNTRL**

Scrive **00110000** nel registro di controllo: **Abilita tastiera ed eco**

Carica in **A1** l'indirizzo di partenza della frase da visualizzare

Visualizza i caratteri scrivendo nel registro **DATA** (indirizzo **\$2000**) fino a quando non incontra il carattere terminatore **0** (End of String, **EOS**)

Verifica ripetutamente il registro di controllo **CNTRL** (indirizzo **\$2001**) fino a quando viene premuto **ENTER**

Esempio: terminale

File: programma024.a68

Inizializza **A1** con l'indirizzo di partenza dell'area di memoria precedentemente riservata

Copia i caratteri dal registro **DATA** del terminale all'area di memoria

Interrompe quando viene trovato il carattere **ENTER** nei dati

Inizializza **A1** con l'indirizzo della stringa di risposta da visualizzare

Visualizza i caratteri fino a quando incontra il carattere terminatore **0** (End of String, **EOS**)

Inizializza **A1** con l'indirizzo di partenza dell'area di memoria che contiene i caratteri precedentemente letti dal terminale

Visualizza i caratteri fino a quando viene trovato il carattere **ENTER** (carattere ASCII di codice **13**)

```
IN1  MOVEA.L    #MEM,A1
      MOVE.B   (A0),D0
      MOVE.B   D0,(A1)+
      CMP.B    #ENTER,D0
      BNE     IN1
      LEA.L    ANSW,A1
OUT1  MOVE.B   (A1)+,D0
      CMP.B   #EOS,D0
      BEQ    CONT1
      MOVE.B  D0,(A0)
      BRA    OUT1
CONT1 MOVEA.L    #MEM,A1
OUT2  MOVE.B   (A1)+,D0
      MOVE.B  D0,(A0)
      CMP.B   #ENTER,D0
      BNE    OUT2
FINE  BRA     FINE
      END    BEGIN
```

Esempio: terminale

File: programma024.a68

Programma in
esecuzione

(usare il file
Terminal.cfg
invece del file
base.cfg

per la simula-
zione!)

The screenshot shows the ASIM - Terminal.cfg window with four panes:

- Terminal: M68000 2:** Displays assembly code. The current instruction is `FINE BRA FINE`. Other instructions include `CMP.B #EOS,D0`, `BEQ CONT1`, `MOVE.B D0,(A0)`, `BCC OUT1`, `MOVEA.L #MEM,A1`, `MOVE.B (A1)+,D0`, `MOVE.B D0,(A0)`, `CMP.B #ENTER,D0`, `BNE OUT2`, and `END BEGIN`.
- Terminal: memory 1:** Shows a memory dump with addresses from 00008000 to 0000806C and their corresponding hexadecimal values.
- Terminal: TERMINAL 3:** Shows the program's output: `Come ti chiami ? ale` and `Ciao, ale`. A red arrow points to the output.
- Terminal.cfg:** Shows configuration details for the simulation, including the configuration name (Terminal.cfg), chip names (memory and M68000), and their respective types and identifiers.

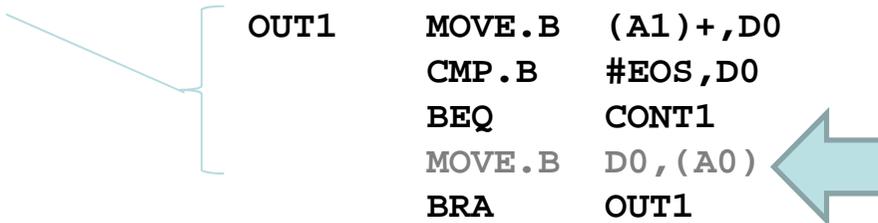
Esempio: terminale

File: programma024.a68

- Osservazione:
 - i dati sono scritti uno dopo l'altro sullo stesso indirizzo
 - Il funzionamento del dispositivo garantisce che valori scritti successivamente sullo stesso indirizzo vengano trasferiti come dati distinti
 - qui l'indirizzo **non** corrisponde fisicamente ad una locazione di memoria!!

Visualizza i caratteri fino a quando incontra il carattere terminatore **0** (End of String, **EOS**), scrivendoli tutti all'indirizzo contenuto in **A0** (che rimane costante nel ciclo e pari all'indirizzo **#TER**, ovvero **\$2000**). L'indirizzo in realtà corrisponde al registro **DATA** del dispositivo terminale

```
BEGIN    MOVEA.L #TER,A2
          MOVEA.L A2,A0
          . . . . .
OUT1     MOVE.B (A1)+,D0
          CMP.B  #EOS,D0
          BEQ   CONT1
          MOVE.B D0,(A0)
          BRA   OUT1
          . . . . .
```



Interazione CPU-periferica

- Osservazione

- il registro di *controllo/stato* **CNTRL** segnala il fatto che sia stato premuto il tasto ENTER, alzando il bit più significativo (che rappresenta un'informazione di *stato*)
- per aspettare fino alla pressione del tasto, il programma principale contiene un ciclo che legge il registro **CNTRL** e itera (bloccandosi nel ciclo) quando il bit di stato è **0**

```
. . . . .  
ENTERHIT EQU $80    10000000  
  
. . . . .  
BEGIN    MOVEA.L #TER,A2  
         MOVEA.L A2,A0  
         ADD.L  #1,A2  
  
. . . . .  
CONT     MOVE.B  (A2),D1  
         AND.B  #ENTERHIT,D1  
         BEQ   CONT
```

Verifica ripetutamente il registro di controllo **CNTRL** fino a quando non viene premuto **ENTER**. Ciò viene segnalato dal fatto che il registro **CNTRL**, posto all'indirizzo **#TER+1**, ha il bit più a sinistra pari ad **1**

Interazione CPU-periferica

- “**Polling**” (o “*I/O controllato da programma*”)
 - tecnica per l’interazione CPU-periferica in cui la CPU accede ripetutamente ai registri di stato fino a quando la periferica è pronta (come nell’esempio precedente)
 - impedisce alla CPU di fare altro durante l’attesa!
 - La CPU si blocca in quello che si chiama *busy-waiting*
- Alternativa: “**Interruzioni**” (*interrupts*)
 - dette anche “**eccezioni**” (*exceptions*)
 - il processore può eseguire altre istruzioni mentre la periferica lavora. Appena questa è pronta, il processore viene interrotto (tramite un’opportuna infrastruttura circuitale fornita dal sistema), e passa ad eseguire la parte di programma che gestisce l’interazione con la periferica