

Input/Output

Alfredo Benso



Introduzione

- Ambiente di lavoro emu8086 (www.emu8086.com)
- Emula un 8086 ed alcune periferiche
 - Stampante
 - Mouse
 - 4 led
 - Un motore
 - 4 semafori stradali
 - 1 robot
 - 1 termometro

Programma tipo (COM)

Un unico segmento che parte all'indirizzo 100h

- `org 100h ;all .COM programs start at 100h`
- `jmp code ;to avoid "executing data"`
- `myArray dw 2, 12, 8, 52, 108`
- `code: mov si, 0`
- `mov ax, myArray[si]`
- `ret`

Programma tipo (EXE)

Segmento dati, stack e codice

- **Data segment**
 - `;add data definition here`
 - `pkey db "Press any key"`
 - `Ends`
- **Stack segment**
 - `dw 128 dup(0)`
 - `Ends`

Programma tipo (EXE)

- `Code segment`
- `Start:`
- `;set segment registers`
 - `mov ax, data`
 - `mov ds, ax`
 - `mov es, ax`
- `;code goes here`
- `lea dx, pkey`
- `mov ah,9`
- `int 21h`
- `;wait for any key`
- `mov ah, 1`
- `int 21h`
- `mov ax, 4c00h`
- `int 21h`
- `ends`

Introduzione

- **Accesso alle risorse di output:**
 - Accesso diretto (IN/OUT)
 - Attraverso l'MSDOS (int 21h)
 - Attraverso il BIOS (int 10h)
- **Accesso alle risorse di input:**
 - Tastiera: tramite MSDOS (int 21h)
 - Mouse: tramite BIOS (int 33h)

Int 21h

- Output:
 - Funzione 2h: Stampa un carattere ed avanza il cursore in avanti di una colonna
 - Funzione 6h: Stampa un carattere
 - Funzione 9h: Stampa una stringa
 - Funzione 40h: Scrive un vettore di caratteri su di un device

Funzione 2h

```
model small
.stack
.data
.code
.startup
    mov ah, 2h ; ah = numero funzione
    mov dl, 'A' ; dl = carattere da stampare
    int 21h ; richiama l'int 21h
.exit
end
```

Funzione 6h

```
model small
.stack
.data
.code
.startup
    mov ah, 6h ; ah = numero funzione
    mov dl, 'A' ; dl = carattere da stampare
    int 21h ; richiama l'int 21h
.exit
end
```

Funzione 9h

```
model small
.stack
.data
    messaggio DB "Hello, World!$"
.code
.startup
    mov ah, 9h
    mov dx, OFFSET messaggio
    int 21h
.exit
end
```

DS:DX = segmento/offset della stringa

Funzione 40h

```
model small
.stack
.data
    messaggio DB "Hello, World!"
.code
.startup
    mov ah, 40h
    mov bx, 1h
    mov cx, LENGTHOF messaggio
    mov dx, OFFSET messaggio
    int 21h
.exit
end
```

Funzione 40h

```
model small
.stack
.data
    messaggio DB "Hello, World!"
.code
.startup
    mov ah, 40h
    mov bx, 1h
    mov cx, LENGTHOF messaggio
    mov dx, OFFSET messaggio
    int 21h
.exit
end
```

BX = device su cui scrivere (1 = console)

Funzione 40h

```
model small
.stack
.data
    messaggio DB "Hello, World!"
.code
    .startup
    mov ah, 40h
    mov bx, 1h
    mov cx, LENGTHOF messaggio
    mov dx, OFFSET messaggio
    int 21h
    .exit
end
```

CX = numero di byte del vettore di caratteri

Funzione 40h

```
model small
.stack
.data
    messaggio DB "Hello, World!"
.code
    .startup
    mov ah, 40h
    mov bx, 1h
    mov cx, LENGTHOF messaggio
    mov dx, OFFSET messaggio
    int 21h
    .exit
end
```

DS:DX = segmento/offset del vettore di caratteri

Int 21h

- Input:
 - Funzione 1h: legge un carattere dallo standard input
 - Funzione 6h: legge un carattere dallo standard input senza attendere
 - Funzione 0Ah: legge una stringa dallo standard input
 - Funzione 0Bh: legge lo stato del buffer di input
 - Funzione 3Fh: legge un vettore di byte da un device

Funzione 1h

```
model small
.stack
.data
    char DB 0
.code
    .startup
    mov ah, 1h ; ah = numero funzione
    int 21h ; richiama l'int 21h
    mov char, al ; al = carattere letto
    .exit
end
```

- La funzione attende finché non è presente un carattere nel buffer di input.
- Il carattere letto viene visualizzato sulla console.

Funzione 6h

```
model small
.stack
.data
    char DB 0
.code
    .startup
    mov ah, 6h ; ah = numero funzione
    mov dl, 0FFh ; dl = FFh
    int 21h ; richiama l'int 21h
    jz skip
    mov char, al ; al = carattere letto
skip:
    .exit
end
```

- La funzione NON attende finché non è presente un carattere nel buffer di input.
- Il carattere letto NON viene visualizzato sulla console.
- Se un carattere è pronto ZF=0.

Funzione 0Ah

- Legge una stringa terminata da Enter
- Il carattere Enter fa parte della stringa
- Struttura di appoggio

```
KEYBOARD STRUCT
    maxInput DB 25 ; # max di byte (0-128)
    inputCnt DB ? ; # byte letti
    buffer DB 25 DUP(?) ; byte letti
KEYBOARD ENDS
```

Funzione 0Ah

```
model small
.stack
.data
    keybdData KEYBOARD <>
.code
    .startup
    mov ah, 0Ah
    mov dx, OFFSET keybdData
    int 21h
    .exit
end
```

Funzione Bh

```
model small
.stack
.data

.code
    .startup
    mov ah, 0Bh ; ah = numero funzione
    int 21h    ; richiama l'int 21h
    cmp al, 0  ; al = 0 -> Nessun carattere
    je skip
    ; Legge il carattere
skip:
    .exit
end
```

Funzione 3Fh

```
model small
.stack
.data
    buffer DB 127 dup(0)
    bufLen DW ?
.code
    .startup
    mov ah, 3Fh
    mov bx, 0h ; bx = device (0=tastiera)
    mov cx, LENGTHOF buffer ; # max byte
    mov dx, OFFSET buffer ; buffer
    int 21h
    mov bufLen, ax ; ax = # byte letti
    .exit
end
```

Int 10h

- Due tipi di modalità video:
 - Testo
 - Grafica

Modalità testo

- Il video è organizzato come una griglia di righe e colonne
- L'angolo in alto e sinistra del video corrisponde a riga = 0, colonna = 0
- I caratteri sono generati utilizzando una tabella residente in memoria

Pagine di testo

- Il testo è organizzato in pagine:
 - Una pagina **attiva** è visualizzata sullo schermo
 - Altre pagine possono essere preparate (scrivendo il testo in esse contenute) e visualizzate quando necessario
 - La pagina video di default è la **pagina 0**

Attributi

- Ogni carattere è descritto tramite 2 byte:
 - Uno memorizza il codice ASCII del carattere da visualizzare
 - Uno definisce gli attributi del carattere
- **Attributi:**
 - **Foreground:** colore del carattere
 - **Background:** colore dello sfondo
- I caratteri sullo schermo possono **lampeggiare**

Attributi

- Bit 7: controllo lampeggio se attivato in precedenza
 - 1=lampeggio attivo
 - 0=lampeggio inattivo
- Bit 6-4: codice colore di background
 - Si possono usare solo i colori a bassa intensità se il lampeggio è attivato
- Bit 3-0: codice colore di foreground
 - Si possono usare tutti i colori possibili

Codici dei colori

- | | |
|--------------------------|-------------------------|
| ■ Bassa intensità | ■ Alta intensità |
| ■ 0h: nero | ■ 8h: grigio |
| ■ 1h: blu | ■ 9h: blu chiaro |
| ■ 2h: verde | ■ Ah: verde chiaro |
| ■ 3h: ciano | ■ Bh: ciano chiaro |
| ■ 4h: rosso | ■ Ch: rosso chiaro |
| ■ 5h: magenta | ■ Dh: magenta chiaro |
| ■ 6h: marrone | ■ Eh: giallo |
| ■ 7h: grigio chiaro | ■ Fh: bianco |

Int 10h

- **Funzione 0h:** definisce la modalità video
- **Funzione 2h:** posiziona il cursore sullo schermo
- **Funzione 3h:** rileva la posizione del cursore sullo schermo
- **Funzione 9h:** scrive un carattere ed il suo attributo nella posizione corrente
- **Funzione 10h:** programma il lampeggio

Funzione 0h

- **Definisce la modalità video**
- **Modi disponibili:**
 - 0h: 40x25, 1 colore
 - 1h: 40x25, 16 colori
 - 2h: 80x25, 2 colori
 - 3h: 80x25, 16 colori
 - 7h: 132x25, 2 colori
 - 14h: 132x25, 16 colori

```
mov ah, 0h
mov al, ModoVideo
int 10h
```

Funzione 2h

- **Posiziona il cursore**

```
mov ah, 2h
mov dh, Riga          ; # della riga
mov dl, Colonna       ; # della colonna
mov bh, Pagina        ; # della pagina video
int 10h
```

- **Esempio**

```
mov ah, 2h
mov dh, 10            ; riga 10
mov dl, 20            ; colonna 20
mov bh, 0             ; pagina attiva
int 10h
```

Funzione 3h

- Legge la posizione del cursore

```
mov ah, 3h
mov bh, Pagina      ; # della pagina video
int 10h
mov posizione, dx   ; dh = riga
                   ; dl = colonna
```

Funzione 9h

- Visualizza un carattere

```
mov ah, 9h
mov al, Carattere   ; carattere da stampare
mov bh, Pagina      ; # della pagina video
mov bl, Attributo   ; blink/colori
mov cx, 1           ; numero di ripetizioni
int 10h
```

Funzione 10h

- Attiva/disattiva il lampeggio

```
mov ah, 10h
mov al, 3h
mov bl, Attributo   ; 1 abilita
int 10h             ; 0 disabilita
```

- Sotto Windows è necessario portare l'applicazione a schermo intero per visualizzare correttamente il lampeggio

Esercizio 1

- Si realizzi un programma che codifichi i caratteri di una stringa mediante una chiave K secondo l'espressione:
 $C' = C \text{ ex-or } K$
- K è un carattere letto da tastiera
- C è l'i-esimo carattere della stringa da codificare, letto dallo standard input
- C' è l'i-esimo carattere della stringa codificata, stampato sulla console

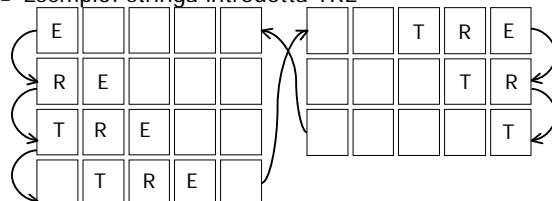
Esercizio 2

- Si realizzi un programma che:
 - Legga da tastiera un testo composto da al più 70 caratteri
 - Visualizzi il testo acquisito da tastiera stampando i suoi caratteri con colori diversi. In particolare, si faccia un modo che caratteri adiacenti siano visualizzati con colori differenti.
- Esempio

Questo è un esempio

Esercizio 3

- Si realizzi un programma che letta da tastiera una stringa composta da non più di 3 caratteri la visualizzi su di un display a scorrimento composto da 5 caratteri per 10 volte.
- Esempio: stringa introdotta TRE



Accesso diretto alle risorse

- Interfaccia della stampante:
 - Porta A: parola di dato
 - Porta B: parola di stato
 - Porta C: porta di controllo

Porta B

- Bit 7: BUSY (la stampante non può accettare nuovi dati) 1=printer not busy
- Bit 6: ACK (ricezione dei dati) 1=data transfer in progress
- Bit 5: Errore Carta (fine carta) 1=no more paper
- Bit 4: On line (Off line or On line) 1=printer on line
- Bit 3: ERR (printer error) 1=no error

Porta C

- Bit 0: STB (impulso per trasferire i dati)
- Bit 1: AUTOFDX (*autofeed*, avanzamento carta di una riga) 1 = line feed automatico
- Bit 2: INIT (inizializza stampante) 0 = inizializza la stampante [segnale opzionale]
- Bit 3: SLCTIN (abilita la stampante ad accettare nuovi dati) 1=printer selected [segnale opzionale]

Stampa di un messaggio (I)

```
prDATA EQU 0378h ; indirizzo LPT1
prSTAT EQU prDATA+1 ; porta B
prCTRL EQU prDATA+2 ; porta C
DELAY EQU 100 ; ritardo
sERR EQU 08h ; attivo BASSO
sSEL EQU 10h ; attivo ALTO
sPE EQU 20h ; attivo ALTO
sACK EQU 40h ; attivo BASSO
sBUSY EQU 80h ; attivo BASSO
cSTB EQU 01h ; attivo ALTO
cAUTO EQU 02h ; attivo ALTO
cINIT EQU 04h ; attivo BASSO
cSEL EQU 08h ; attivo ALTO
```

Stampa di un messaggio (II)

```
STACK segment
DATA segment
    msg DB 'Ciao a tutti',0Dh,0Ah,0
ends

code segment
    MOV BX, OFFSET msg
    XOR SI, SI ;azzera SI
next: MOV AL, [BX][SI]
    CMP AL, 0
    JE done
    CALL pr_al ;stampa un carattere
    INC SI
    JMP next
done: ...
```

Stampa di un messaggio (III)

```
pr_al PROC
    PUSH DX ; salvo i registri nello stack
    PUSH AX
    MOV AH, AL ; salva AL in AH
    MOV DX, prSTAT
pr_n_ready: IN AL, DX ; legge lo stato
    ; della stampante
    TEST AL, sERR
    JZ pr_error ; errore
    TEST AL, sBUSY
    JZ pr_n_ready ; busy
    TEST AL, sSEL
    JZ pr_n_ready ; on line
```

Stampa di un messaggio (IV)

```

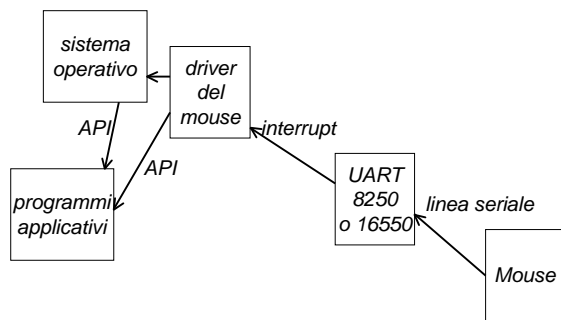
MOV AL, AH
MOV DX, prDATA
OUT DX, AL ; invia il dato
MOV CX, DELAY
ciclo1: LOOP ciclo1
MOV DX, prCTRL
IN AL, DX
OR AL, cSTB ; setta il bit di strobe
OUT DX, AL
MOV CX, 2*DELAY
ciclo2: LOOP ciclo2
AND AL, not cSTB ; resetta lo strobe
OUT DX, AL
MOV CX, DELAY
ciclo3: LOOP ciclo3
    
```

Stampa di un messaggio (V)

```

pr_done: POP AX
          POP DX
          RET
pe_error: JMP pr_done
pr_al    ENDP
          ENDP
    
```

Mouse



Protocollo del mouse

- Il mouse contiene un *trasmettitore seriale* non programmabile. Esso trasmette informazioni ogniqualvolta:
 - Viene premuto o rilasciato un tasto
 - Viene spostato il mouse
- Ogni volta che il mouse deve trasmettere informazioni, spedisce un *pacchetto di dati* contenente:
 - Lo stato attuale dei pulsanti
 - Lo spostamento relativo rispetto all'ultimo pacchetto di dati inviato

Trasmissione seriale

- Il mouse trasmette secondo i seguenti parametri:
 - 1200 baud
 - 8 bit per carattere
 - 1 bit di stop
 - nessuna parità

Formato del pacchetto dati

- Pacchetto di 3 caratteri di 8 bit ciascuno
- Trasmessi in sequenza, e essere letti singolarmente dall'interfaccia seriale

<i>primo byte</i>	1	1	L	R	Y7	Y6	X7	X6
<i>secondo byte</i>	1	0	X5	X4	X3	X2	X1	X0
<i>terzo byte</i>	1	0	Y5	Y4	Y3	Y2	Y1	Y0

La trama

- I bit 7 e 6 indicano la *trama* del pacchetto:
 - Nel primo byte valgono 11
 - Nel secondo e terzo byte valgono 10
- Per leggere dati significativi, occorre:
 - *Sincronizzarsi* con il trasmettitore, aspettando il primo byte con indicatore di trama 11
 - *Attendere* i successivi due caratteri, verificando che abbiano indicatore pari a 10
- In caso di errore occorre *scartare* quanto ricevuto e aspettare il prossimo pacchetto

1	1	L	R	Y7	Y6	X7	X6
1	0	X5	X4	X3	X2	X1	X0
1	0	Y5	Y4	Y3	Y2	Y1	Y0

I tasti

- I bit 5 e 4 del primo byte contengono lo stato dei due tasti del mouse:
 - Il bit 5 "L" è relativo al tasto sinistro
 - Il bit 4 "R" è relativo al tasto destro
 - Ciascun bit vale 1 se il tasto è premuto, 0 se non è premuto
 - Se tutti e due sono a 1, allora è stato premuto il tasto centrale
- Il valore del bit indica lo *stato* attuale del tasto: per rilevare una variazione (pressione o rilascio), il *software* dovrà fare dei confronti con lo stato precedente

1	1	L	R	Y7	Y6	X7	X6
1	0	X5	X4	X3	X2	X1	X0
1	0	Y5	Y4	Y3	Y2	Y1	Y0

Lo spostamento

- I bit da 3 a 0 del primo byte, insieme ai bit da 5 a 0 del secondo e del terzo byte, contengono lo spostamento del mouse rispetto all'ultimo pacchetto trasmesso
- Gli spostamenti sono espressi in complemento a 2 su 8 bit
- Spostamento orizzontale (positivo verso destra): bit 3 e 2 del primo byte e bit da 5 a 0 del secondo byte
- Spostamento verticale (positivo verso il basso): bit 1 e 0 del primo byte e bit da 5 a 0 del terzo byte

1	1	L	R	Y7	Y6	X7	X6
1	0	X5	X4	X3	X2	X1	X0
1	0	Y5	Y4	Y3	Y2	Y1	Y0

Mouse

- Int 33h
 - **Funzione 0:** resetta il mouse e verifica se è disponibile
 - **Funzione 1:** visualizza il puntatore del mouse
 - **Funzione 2:** nasconde il puntatore del mouse
 - **Funzione 3:** legge la posizione e lo stato del mouse
 - **Funzione 4:** definisce la posizione del mouse
 - **Funzione 5:** verifica la pressione dei tasti del mouse
 - **Funzione 6:** verifica il rilascio dei tasti del mouse
 - **Funzione 7:** definisce i limiti orizzontali
 - **Funzione 8:** definisce i limiti verticali

Funzione 0h

- Resetta e legge lo stato del mouse

```
mov ax, 0h
int 33h
cmp ax, 0h
je MouseNonDisponibile
```

Funzione 1h/2h

- **Funzione 1:** Visualizza il puntatore del mouse

```
mov ax, 1h
int 33h
```

- **Funzione 2h:** Nasconde il puntatore del mouse

```
mov ax, 2h
int 33h
```

Funzione 3h

- Legge la posizione e lo stato del mouse
- BX: stato dei bottini del mouse
 - Bit 0 posto a 1: tasto sinistro premuto
 - Bit 1 posto a 1: tasto destro premuto
 - Bit 2 posto a 1: tasto centrale premuto
- CX: coordinate X in pixel
- DX: coordinate Y in pixel

```
mov ax, 3h  
int 33h
```