1 Tecniche di programmazione per l'I/O

I problemi del mondo reale

Quando si devono risolvere problemi reali, non esercizi, le soluzioni devono essere complete.
Vanno tenuti in considerazione tutti i casi possibili e bisogna considerare che l'hardware e l'utente non sono infallibili. Il software deve tenere in conto anche dei possibili errori dell'utente e dei malfunzionamenti dell'hardware.

Anticipazione dei malfunzionamenti dell'hardware

Il programma dovrebbe "anticipare" le condizioni anomale ed operare in modo che, se trattate in tempo, non facciano danno.
Se, per esempio, si guasta un sensore usato in un loop di polling, esso non potrà continuare a girare per sempre, nell'attesa attesa fiduciosa e stupida che il sensore, ormai rotto, risponda.
Sarà necessario che ogni loop di polling preveda un "timeout", ossia un tempo limite oltre il quale il programma esce dal ciclo e entra in una parte in cui gestisce l'emergenza del sensore rotto.
In un programma "reale" ben scritto questo deve essere fatto per tutti i loop di attesa, e per tutte le condizioni anomale che possono capitare.
Vediamo un piccolo esempio in cui si attende l'arrivo di un pezzo meccanico al suo finecorsa. L'attesa in un loop di polling controlla anche l'orologio, per lanciare una procedura d'errore nel caso che il pezzo non arrivi entro un certo tempo limite:

```
AttesaDelFineCorsa:
    MOV AL , [Finecorsa]
    TEST AL, 1       ;  se il bit 0 di AL è a 1 significa
                     ; che il pezzo è arrivato al finecorsa
    JE ProseguiNormalmente
    ; se non è arrivato in fondo, verifica che non siamo in questo loop da troppo tempo:
    CALL CheckTimeout
    ; CheckTimeout mette 1 nel flag di carry se il tempo
    ; destinato a questo loop è esaurito
    JZ AttesaDelFineCorsa
    ; se arriva qui è scattato il TimeOut
    ; spengo la macchina perché non faccia ulteriori danni:
    CALL SpegniTutto
    ; notifico l'errore all'utente:
    CALL AvvertiDellErrore

    MOV AH, 4Ch  ;  fermo il programma (funziona solo se il S.O. è MS-DOS)
    INT 21h

ProseguiNormalmente:
    ..
    .. ; qui il programma continua nelle sue operazioni normali
    ..
```

Watchdog

Un modo più sofisticato di uscire dai loop senza fine, che richiede hardware aggiuntivo ma che funziona sia nel caso di guasti hardware sia in caso di errori nel software, è la tecnica del "**watchdog**" (cane da guardia).
Il watchdog è un contatore che viene incrementato automaticamente per via hardware e che deve essere regolarmente "resettato" dal software. Ciò significa che il nostro programma nel corso del suo svolgimento, deve andare a scrivere periodicamente in un registro del watchdog. Questa scrittura lo fa tornare a zero ed impedisce che il suo conteggio vada in overflow.
La linea di overflow del timer è collegata ad un interrupt della CPU, tipicamente all'interrupt non mascherabile. Se il programma si blocca in un loop senza fine il contatore del watchdog si "riempie" e viene lanciato l'interrupt.
Nella ISR di questo interrupt si metterà in sistema in condizioni di sicurezza, si darà una comunicazione di allarme, si prenderanno le possibili contromisure al malfunzionamento ed eventualmente si ricomincerà tutto daccapo.
Molti microcontrollori odierni includono al loro interno uno o più watchdog.

Comprendere le possibilità fisiche del sistema

La velocità di un computer è spesso sottostimata dal tecnico.

Attuazione che fa rimanere fermo il sistema, accendendolo e spegnendolo.

Dare il tempo ai relay di commutare

1.1 Alcuni esempi di programmi di I/O

1.1.1 Display del valore di contatori

Questo è un facile e istruttivo problema, anche se non del tutto realistico.
Un sistema di controllo di una macchina industriale deve interfacciare un numero N di dispositivi "contacolpi", cioè di contatori che misurano il numero di certi eventi, non ulteriormente specificati. Il problema, da risolvere con un programma Assembly per 8086, riguarda la parte del sistema di controllo che legge il valore corrente dei contatori e lo visualizza su 4 display LED a 7 segmenti.

I contatori sono da 8 bit, N ha il valore massimo di 256, ma può essere minore.

a) La CPU legge il valore corrente dei contatori ad N indirizzi consecutivi in I/O isolato, che partono dall'indirizzo 0. Per ogni contatore il display deve visualizzare contemporaneamente, in esadecimale, il numero del contatore (da 0 ad N) ed il valore del conteggio.
La visualizzazione si deve protrarre per 3 secondi, poi si deve passare al contatore successivo. Dopo il contatore N si deve ricominciare dal primo ed il programma non deve mai concludersi. Il programma non si deve interessare se il conteggio eccede il valore massimo che un contatore può contenere, ma visualizzare solo il contenuto corrente.
La visualizzazione avviene scrivendo un numero di 16 bit all'indirizzo 0FFF00h della memoria ordinaria.
Il valore del conteggio deve essere scritto nelle due cifre meno significative del display, che sono fisicamente collegate al byte di indirizzo 0FFF00h, mentre il numero del contatore deve finire all'indirizzo 0FFF01h.
Scrivere il programma in modo che sia facilmente adattabile ad un numero diverso di contatori. Spiegare cosa si dovrà fare per cambiare il numero di contatori gestito dal programma.

<FILE>
!!!! LED7segmenti.FH5
</FILE>

b) dopo aver risolto il problema precedente descrivere come deve cambiare il sistema e come il programma già scritto se i contatori sono a 12 bit e N massimo è 16 (la domanda NON è opzionale).
c) risolvere il problema precedente visualizzando numeri decimali invece che esadecimali. Da quanti bit devono essere i contatori per poter utilizzare lo stesso display senza "overflow" delle sue cifre?

Problema a)
Il fatto che gli indirizzi dei contatori inizino da zero può essere utile per realizzare un programma un po' più semplice. Dato che i contatori hanno indirizzi da 0 ad N - 1, l'indirizzo di I/O equivale al numero del contatore.
Per semplificare il programma facciamo un ciclo con la loop e scriviamo sul display i valori dei conteggi "a rovescio", utilizzando CX, la variabile del ciclo, anche come "nome" del contatore.
E' importante rimarcare come in questo modo il programma è più semplice ma lega il software all'architettura dell'hardware. Se cambia l'indirizzo hardware del primo contatore il programma deve essere cambiato in modo sostanziale. Realizzare programmi troppo dipendenti dall'architettura dell'hardware di solito non è una buona idea e deve essere fatto solo quando il vantaggio che se ne ricava è considerevole.

Un programma che risolve il punto a) del problema è il seguente:

```
; esempio dei contatori a 12 bit, da visualizzare su un display a 7 segmenti

N EQU 102    ; Numero dei contatori !! deve essere <256 !!

OffsetDisplay EQU 0  ; Offset del primo indirizzo del display,
                     ; mappato in memoria ordinaria
SegDisplay EQU 0FFF0h; Segmento dell'indirizzo del display
Calib EQU 1329   ; Numero che fa fare un'attesa di 1 ms nel
                 ; loop interno della procedura Attesa
CODICE SEGMENT
ASSUME CS:CODICE
Inizio:
    ; sistemazione del registro di segmento per puntare agli indirizzi del display:
    MOV AX, SegDisplay
    MOV ES, AX
DiNuovo:
    MOV CX, N    ; per fare un loop da N giri
    MOV DX, 0    ; indirizzo del primo contatore
PerTuttiIContatori:
    IN AL, DX    ; lettura del conteggio del DX contatore
    MOV ES:[OffsetDisplay], AL    ; scrittura del conteggio nella parte
                                  ; destra del display
    MOV ES:[OffsetDisplay+1], DL ; scrittura del numero del contatore a sinistra
    ; !! Questa ^ funziona solo perché l'indirizzo del contatore è uguale al suo "numero"

    PUSH CX          ; memorizzazione di CX (dobbiamo modificarlo per passare
```

```
                ; il parametro ad Attesa)
    MOV CX,3000      ; 3 secondi (tempo di attesa in ms)
    CALL Attesa
    POP CX           ; ripristino del valore del conteggio della loop
    LOOP PerTuttiIContatori
    JMP DiNuovo

Attesa PROC
Fuori:  ; loop esterno per CX ms
      PUSH CX
      MOV CX, Calib
Dentro: ; loop interno calibrato per durare un ms
        NOP
      LOOP dentro
      POP CX     ; ripristina il CX usato come parametro
      LOOP Fuori
Attesa ENDP

CODICE ENDS
END Inizio
```

La soluzione è molto semplice perché ciò che deve essere scritto nel display si divide esattamente in due byte, che si possono scrivere in memoria facilmente.
Si noti la tecnica usata per accedere all'indirizzo assoluto di memoria 0FFF0h, al quale è mappato il display.

Problema b)

Il problema b) è invece un po' più complicato.
Innanzi tutto bisogna decidere cosa cambiare nel sistema. L'acquisizione di contatori a 12 bit significa che i valori dei conteggi debbono essere letti da due port di I/O e non più da uno solo.
Supponiamo che i 12 bit dei contatori si possano leggere ad indirizzi successivi di I/O, la parte bassa agli indirizzi bassi.
Dato che ora i contatori sono pochi (max 16) bastano 4 bit per rappresentare il numero del contatore. Faremo visualizzare il numero del contatore nella cifra più significativa del display ed il valore del conteggio nelle altre cifre, che richiedono 12 bit.
Ora è necessario "fondere" nel Byte alto i quattro bit della parte alta del conteggio e quelli del numero del contatore.
Dato che ora l'indirizzo del contatore non coincide più con il suo "numero", siamo costretti a modificare il programma.
Visualizziamo i contatori in ordine crescente ed utilizziamo una tecnica che permetta di cambiare facilmente l'indirizzo del primo contatore.
Nel programma che segue si esegue la "fusione" dei nibble della parte alta di AX, prima di scrivere AX sul port del display.
Per contare "in avanti" è necessario un contatore, per il quale viene usato il registro BL. Il valore di quel conteggio deve essere posizionato nel nibble più significativo di AX, perché in quel punto "accenderà" il LED più a sinistra.
Questo si può ottenere con delle shift, oppure con il bel trucco illustrato nel codice che segue.
Vediamo dunque le differenze fra il codice precedente e quello che risolve il problema b)

```
N EQU 10     ; Numero dei contatori !! deve essere <16 !!

; introduciamo un indirizzo iniziale dei contatori, parametrizzato:
IndInizCont EQU 0    ; Indirizzo iniziale dei port che contengono i valori dei conteggi

; .. le altre definizioni uguali alle precedenti

DiNuovo:
    ; scandiamo i contatori "all'avanti", leggendo a partire da IndInizCont:
    MOV DX, IndInizCont  ; partiamo da qui per leggere i valori dei contatori
    ; azzeramento di BL:
    XOR BL, BL   ; !! BL è il numero del contatore
                 ; !! GIA' "SHIFTATO" DI QUATTRO BIT A DESTRA !!
    MOV CX, N
PerTuttiIContatori:
    ; lettura con una sola IN dei 12 bit del conteggio in un numero da 16 bit
    IN AX, DX
    ; posizionamento per la prossima lettura:
    INC DX
    INC DX
    AND AX, 0FFFh    ; pulizia del nibble più alto del numero appena letto
                     ; (per essere sicuri che ci siano degli zeri)
    OR AH, BL        ; "fusione" del due nibble del numero contatore con
                     ; la parte alta del conteggio
    MOV ES:[OffsetDisplay], AX   ; scrittura su display del numero del contatore
                                 ; e del conteggio
```

```
    ; .. chiamata ad Attesa, come prima

    ADD BL, 10000b   ; !! ignobile trucco per far contare solo nel nibble
                     ; !! più significativo di BL: si parte da zero e si
                     ; !! aggiunge sempre un "bit di peso 4" in questo modo nel
                     ; !! nibble più alto c'è il numero di contatore ed
                     ; !! in quello basso sempre 0
    LOOP PerTuttiIContatori
    JMP DiNuovo

; .. il resto come prima
```

Problema c)
!!!! FARE !

1.1.2 Controllo ON/OFF

Per ragioni economiche molti controllori di sistemi continui sono realizzati con attuazioni discrete di tipo ON/OFF. In questi casi l'attuatore[1] o è spento oppure acceso (ON oppure OFF :-).

Dunque si richiede di regolare con un controllo ON/OFF il valore di una grandezza di uscita continua, per esempio si potrebbe regolare la temperatura di un ambiente con un resistore scaldante.
Stabilita la temperatura che si vuole raggiungere (set point) accenderemo il riscaldatore quando la temperatura del sistema è inferiore e lo spegneremo quando è superiore.
Si stabilisce un valore "di soglia" al di sotto del quale l'attuatore è spento, mentre è acceso quando la variabile del sistema è sopra la soglia.
Lo svantaggio di questo semplice approccio è che, se l'attuazione fa "muovere" il sistema rapidamente, quando l'attuatore viene acceso l'uscita del sistema cambia e supera rapidamente la soglia, per cui è necessario spegnerlo subito. Appena l'attuatore viene spento l'uscita cala rapidamente e bisogna riaccenderlo.
Dunque può accadere che l'attuatore venga comandato in rapide sequenze di ON e OFF. Questo modo di operare potrebbe essere dannoso per l'attuatore ed eventualmente anche per il sistema.
Si pensi per esempio ad una caldaia a gas per il riscaldamento domestico. Essa non può essere accesa e spenta ogni secondo, perché si romperebbe dopo pochi giorni; se invece l'attuatore è un resistore scaldante esso può tranquillamente sopportare decine o migliaia di cicli ON/OFF in un secondo.
Quando la frequenza di queste sequenze di ON e OFF è troppo alta, per l'attuatore o per il sistema da controllare, si parla di "effetto ringing" (scampanellìo?), con un nome mutuato dal rumore che fa, prima di rompersi, un attuatore a relais che venga acceso e spento decine di volte al secondo.
Per non avere effetto ringing bisogna che sia vera almeno una delle seguenti condizioni:
1. il sistema è abbastanza "lento" a rispondere per cui, per i suoi ritardi "naturali" passa la soglia solo dopo un tempo sufficientemente lungo
2. il sistema che misura la grandezza di uscita ha una risoluzione sufficientemente bassa; in questo modo il regolatore non si accorge delle variazioni dell'uscita fino a che esse non sono superiori alla risoluzione del sensore, ciò può aumentare il tempo dei cicli ON/OFF.
Nel caso che nessuna di queste condizioni sia rispettata è necessario introdurre due soglie e lavorare "con isteresi".
<FILE>
!!!! Onoff.FH5
</FILE>
Figura 1: Controllo ON/OFF con una e due soglie

Controllori ON/OFF con isteresi
La soluzione definitiva al problema del ringing degli attuatori è la realizzazione di controllori ON/OFF "con isteresi".
Si individuano due soglie e si usa il seguente algoritmo:
1. se l'uscita supera il valore della soglia superiore si mette OFF l'attuatore
2. se l'uscita è minore della soglia inferiore si mette ON l'attuatore
3. se l'uscita ha valore compreso fra le due soglie si lascia l'attuatore così com'è
Assegnando opportunamente le due soglie di riesce ad evitare il ringing.

1.1.3 Controllo PWM

!!!!

1 per queste definizioni si veda il volume precedente

1.1.4 Un progetto completo

In questo capitolo sviluppiamo un progetto completo di un'applicazione di I/O in Assembly.

Si tratta di un progetto, non di un programma; i dettagli sul codice sono intenzionalmente omessi. Il lettore li potrà sviluppare per esercizio, provandone il risultato con il programma di simulazione incluso nel CD ROM allegato a questo volume.
Nel CD ROM è compreso anche lo "scheletro" dell'applicazione, realizzato secondo l'analisi che verrà sviluppata in questo capitolo. Esso potrà essere utile come base per costruire l'applicazione.
Il nostro progetto prende le mosse dall'analisi del funzionamento di una lavatrice.
Al fine di progettare un buon programma dobbiamo trovare parti comuni e differenze. Concentreremo in procedure le funzioni comuni che individueremo, mentre le differenze costituiranno i parametri di quelle procedure.

Passiamo dunque ad illustrare il problema da risolvere:

Una lavatrice deve essere controllata da un microcontrollore compatibile con l'8086, collegato ai tre port di I/O indicati dalla Figura 2: Port di I/O per il controllo della lavatrice, con i sensori ed attuatori illustrati nelle figure.

Port di indirizzo BASE, OUTPUT

Peso del bit	7	6	5	4	3	2	1	0
Funzione	/	/	/	Riscalda-tore 1= scalda	Scarico 1= aperta	Prelavag-gio 1= aperta	Lavaggio 1= aperta	Risciac-quo 1= aperta

Port di indirizzo BASE + 1, INPUT

Peso del bit	7	6	5	4	3	2	1	0
Funzione	Livello Alto 1 = sopra	Porta Aperta 1 = aperta	Livello Basso 1 = sopra	Bottoniera H	Bottoniera L	/	/	/

Port di indirizzo BASE + 2

Peso del bit	7	6	5	4	3	2	1	0
Funzione	/	/	/	/	Termosta-to 1 = sopra	Accens. Motore 1 = ON	Direz. Motore 1=orario	Veloc. Motore 1= veloce
Direzione					INPUT	OUTPUT	OUTPUT	OUTPUT

Figura 2: Port di I/O per il controllo della lavatrice

Il significato dei bit di I/O è il seguente
- Risciacquo comanda un'elettrovalvola che apre un flusso d'acqua che attraversa la vaschetta di risciacquo e poi finisce nel cestello di lavaggio
- Lavaggio comanda un'elettrovalvola, analoga alla precedente, che usa la vaschetta per di detersivo di lavaggio
- Prelavaggio elettrovalvola per il detersivo di prelavaggio
- Scarico elettrovalvola di scarico; se è aperta l'acqua esce dal cestello

Tutte le elettrovalvole sono aperte (l'acqua passa) quando il valore logico del bit del port BASE è 1.
- Riscaldatore un resistore scaldante, percorso da corrente, provvede al riscaldamento dell'acqua di lavaggio. Se il bit è 1 esso scalda, se 0 no.

???? bottone "niente centrifuga (panni delicati) ???? !!!! cambiare il programma !!!!

Il termostato dà un valore logico TTL, diverso se la temperatura è maggiore o minore di quella impostata dall'utente. In modo analogo funzionano i due sensori di livello: se l'acqua li bagna avrà un valore logico, l'altro se è asciutto. Stabilire e documentare la logica di funzionamento dei sensori e degli attuatori e scrivere il programma in modo coerente a quanto stabilito.

La bottoniera di controllo della lavatrice sia fatta in modo che può essere in una sola di quattro posizioni, tramite le quali l'utente indica che tipo di lavaggio vuole ottenere:
1. lavaggio completo (con prelavaggio)
2. senza prelavaggio
3. solo risciacquo
4. solo centrifuga

La bottoniera codifica le quattro posizioni nei due bit BottL e BottH, all'indirizzo BASE + 1, secondo la seguente tabella:

	Bottoniera Bit alto (H)	Bottoniera Bit basso (L)
lavaggio completo	0	0
ciclo senza prelavaggio	0	1
solo risciacquo	1	0
solo centrifuga	1	1

Il significato dei segnali indicati in Figura 2: Port di I/O per il controllo della lavatrice è ulteriormente illustrato dalla Figura 3: Architettura della lavatrice.

!!!!snapshot del programma lavatrice!!!!

Figura 3: Architettura della lavatrice

Il problema da risolvere è l'analisi del sistema e la progettazione del software dell'elettrodomestico.
Pensiamo dunque al funzionamento della lavatrice ed individuiamo i fattori comuni e quelli diversi. Dobbiamo trovare dei "mattoni" elementari, con i quali costruire il programma che controlla la lavatrice.
Definiamo anche una "nomenclatura", un modo specifico di indicare le cose, inventato per risolvere questo problema.
Osservando una lavatrice al lavoro vediamo che essa effettua una serie di rotazioni del cestello e di pause e che una rotazione in un senso è sempre seguita da una rotazione nell'altro senso.
Chiamiamo "rotazione" ciò che succede quando il cestello gira in uno stesso senso e "ciclo" la sequenza di una rotazione, seguita da una pausa e da una rotazione nel senso opposto.
Il riparatore di lavatrici interpellato dall'Autore afferma che per avere un miglior bucato si deve iniziare a far scaldare l'acqua contemporaneamente all'inizio dei cicli di lavaggio e solo dopo che il cestello è stato riempito di acqua. La nostra lavatrice dovrà seguire questa regola. Il programma dovrà perciò accendere e spegnere il riscaldatore, in base a ciò che legge dal termostato, mentre esegue le rotazioni o le pause.
Chiamiamo "fase" un certo numero di "cicli" ed individuiamo le seguenti quattro fasi del processo di lavaggio: prelavaggio, lavaggio, risciacquo e centrifuga.
Facciamo ora uno schema nel quale comprendiamo tutte le analogie riscontrate nei vari momenti del funzionamento della lavatrice:
- "immissione": ogni "fase" inizia con il riempimento del cestello, attraverso la giusta vaschetta
- "scarico": ogni "fase" termina con lo scarico di tutta l'acqua contenuta nel cestello
- "rotazione": periodo di tempo in cui il cestello viene fatto ruotare nello stesso verso
- "pausa": periodo di tempo nel quale il cestello rimane fermo
- "ciclo": sequenza rotazione
- "fase": sequenza "immissione", "cicli", "scarico"

A ciascuno di questi elementi comuni faremo corrispondere una procedura.
Individuiamo ora ciò che distingue gli elementi comuni nei vari momenti del processo:
- immissione: l'acqua potrà essere caricata da uno qualsiasi dei cassetti
- rotazione: possono essere diverse:
  - durata della rotazione
  - velocità della rotazione
  - presenza o meno della regolazione di temperatura
- pausa: durante le pause possono essere diverse:
  - durata delle pause (durante il risciacquo le pause sono più lunghe)
  - presenza o meno della regolazione di temperatura
- ciclo
  - durata delle rotazioni
  - velocità delle rotazioni
  - presenza o meno della regolazione di temperatura
- fasi
  - vaschetta di alimentazione acqua
  - numero dei cicli
  - durata delle rotazioni
  - velocità delle rotazioni
  - presenza o meno della regolazione di temperatura

Assumiamo che la porta dell'oblò del cestello sia sempre chiusa prima di iniziare il programma: un bottone elettromeccanico impedirà al programma di funzionare se la porta è aperta.

E' utile definire alcune maschere per i singoli bit dei sensori, il programma sarà più autodocumentato e più facile da modificare:

```
mANDrisciacquo EQU 00000001b     ; maschera per AND sul bit vaschetta risciacquo
mORrisciacquo EQU 11111110b      ; maschera per OR sul bit vaschetta risciacquo
mANDriscaldatore EQU 00010000b   ; maschera per AND
mORriscaldatore EQU 11101111b    ; maschera per OR
```

procedura Inizializza
E' utile metterne sempre una. In questa procedura metteremo tutte le cose che si devono fare una sola volta e che valgono per tutto il programma, come per esempio la programmazione del chip di I/O. Qualora sia necessario aggiungere al programma altre inizializzazioni si saprà subito dove intervenire.

Si può individuare un nucleo comune in tutte le modalità di funzionamento della lavatrice. Infatti durante il suo funzionamento la lavatrice compie un certo numero di "cicli" nei quali tiene per un certo tempo

Definiamo "ciclo" una sequenza di eventi come la successiva:

```
    gira in senso orario        fermo        gira in senso antiorario
_____________________________________________________________________>

    TempoRotazione              TempoStop    TempoRotazione
```

Parametri della procedura Fase

TempoRotazione in millisecondi
nCicli
Vaschetta
    Codifica:
        1 = usa vaschetta prelavaggio
        2 = usa vaschetta lavaggio
        3 = usa vaschetta risciacquo
        4 = non usare nessuna vaschetta
Riscalda: flag
        0 = non riscaldare
        1 = riscalda alla temperatura indicata dal termostato (termometro)

```
    MOV AX,
    CALL

Fase PROC
```

Si lascia come esercizio al lettore il completamento del programma, che potrà essere provato con la simulazione "lavatrice", inclusa nel CD ROM allegato a ???? questo ???? libro (o al volume XXXX di questo libro).

Curiosità

Guai alla Stazione Termini