

## Sistemi di elaborazione

Un *“sistema di elaborazione”*, è una struttura logica in grado di eseguire su *“dati”* presentati in ingresso, operazioni logiche od aritmetiche, fornendo in uscita dei *“risultati”*.

Le operazioni da effettuare sono individuate attraverso codici numerici detti *“istruzioni”*.

Un sistema di questo tipo si può facilmente immaginare scomposto in due strutture, una parte *“operativa”* che **esegue le sequenze di operazioni previste per trasformare i “dati”,  $X$  nei “risultati”,  $Z$**  ed una parte di *“controllo”* che **interpreta le “istruzioni” trasformandole in “comandi”,  $A$  che sono funzione anche delle “condizioni”  $B$**  attraverso le quali la rete operativa comunica a quella di controllo lo stato di avanzamento della elaborazione.

Entrambe le parti del sistema sono **reti sequenziali sincrone** il cui funzionamento è cadenzato dallo stesso *“clock”*.

## Sviluppo di una istruzione

Una *“istruzione”* è una operazione del sistema esprimibile in una sequenza di **microoperazioni** (*moperazioni*), detta *“msequenza”*

Ogni *moperazione* può essere costituita da un insieme di operazioni singole da eseguire in un certo intervallo di tempo.

Una *msequenza* esprime la successione delle azioni elementari della parte *“operativa”* del sistema in cui la parte di *“controllo”* traduce ciascuna istruzione.

La parte *“operativa”* del sistema è composta da moduli combinatori, ciascuno in grado di eseguire un certo repertorio di *moperazioni* logiche, aritmetiche e/o di trasferimento. L'esecuzione di una *moperazione* da parte dell'unità operativa del sistema richiede in generale alla parte di **controllo** la generazione di una sequenza di segnali di **comando** sul cui sviluppo agiscono le *“condizioni”* che, ad ogni periodo di clock, vengono trasmesse dalla parte operativa. Tutte le operazioni sono abilitate da appositi bit ma sincronizzate dal clock.

L'insieme dei comandi necessari per eseguire una *moperazione* è detto *“mistruzione”*.

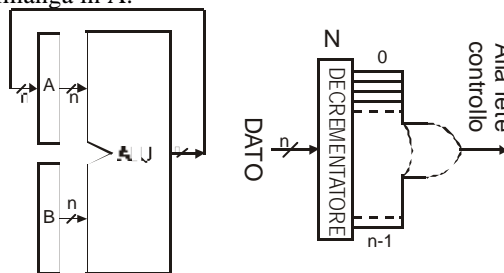
## Sviluppo di una istruzione (2)

La successione di comandi costituenti una **istruzione** gestirà il funzionamento dei moduli combinatori della parte operativa coinvolti nella **operazione**, come ALU, multiplexer, registri a scorrimento, ecc..., abilitando i dati in uscita dai registri sorgente, selezionando l'operazione da effettuare ed abilitando in lettura il registro di destinazione, il tutto scandito dal periodo di clock.

I **"comandi"** non sono altro che i codici numerici che abbiamo visto associati a ciascuna operazione logica, aritmetica e di trasferimento nei vari esempi di reti di interconnessione tra registri e blocchi logici combinatori che abbiamo usato nel modulo A per introdurre il linguaggio RTL.

## La parte "operativa"

Supponiamo che la **parte di controllo** di un sistema debba eseguire una **sequenza** che preveda il caricamento, in un registro N, di un numero intero detto DATO e che si voglia calcolare il numero  $A+NB$ , dove A e B sono i numeri contenuti in due omonimi registri, ed il risultato si vuole che rimanga in A.



La **operazione**:  $N \leftarrow \text{DATO}$ , interpretata dal **controllo** genererà un primo **comando** di caricamento del registro N e poi un secondo per abilitare le uscite di N verso la funzione OR del contenuto.

## Compiti della parte di controllo

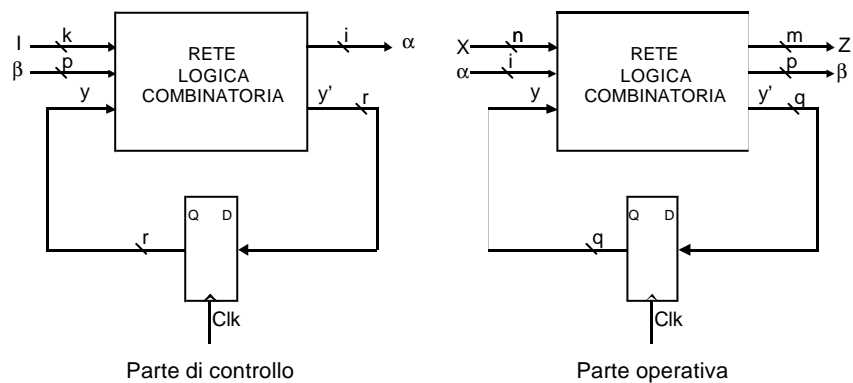
Poiché inizialmente avremo certamente  $N \neq 0$ , questa “*condizione*” inviata alla parte di *controllo* genererà la successione di *comandi* necessari per l’esecuzione della *operazione*:  $A \leftarrow A + B$

I comandi saranno generati in questo ordine, prima quello di **abilitazione alle uscite dei due registri A e B**, poi verrà generato il **codice numerico che seleziona l’operazione di somma nel repertorio della ALU**, e, quindi, sarà fornito il **livello che abilita il caricamento del risultato in A**.

Contemporaneamente il controllo del sistema avrà anche generato il **comando di decremento del registro N**, in modo che tutto sia pronto per la successiva iterazione.

Quando **dopo N cicli**, alla parte di *controllo* del sistema arriverà la *condizione*  $N = 0$ , essa bloccherà la successione di *comandi* per l’operazione aritmetica ed **il contenuto di A sarà pari ad  $A + NB$**

## Struttura di parte di controllo e parte operativa



## Un primo esempio semplice

### Sistema senza istruzioni e senza condizioni

Per focalizzare meglio il funzionamento di una rete di *controllo* ci possiamo riferire ad una situazione molto semplificata (**verrà successivamente integrata**) in cui essa non riceve *istruzioni I* dall'esterno né *condizioni C* dalla parte operativa.

La situazione ipotizzata, nonostante le drastiche semplificazioni, consente ugualmente di capire alcuni meccanismi importanti.

Un esempio pratico corrispondente alle nostre ipotesi e già visto nelle ultime lezioni del modulo A, è:

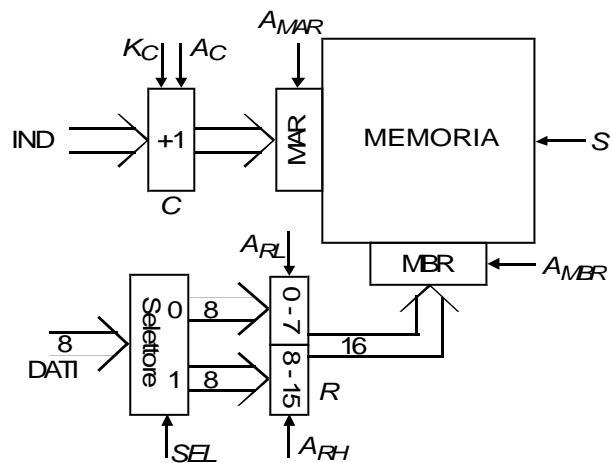
- un sistema che contiene una memoria M in cui ciascuna locazione contiene 16 bit.
- Il sistema riceve, in sincronismo con un clock (*clk*), dati da 8 bit (1 byte) che vanno caricati, a due a due, in locazioni consecutive della memoria, a partire da un indirizzo fornito, nell'istante iniziale, dal canale esterno IND.

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

7

## Parte operativa



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

8

## *sequenza e istruzioni*

La **sequenza** che descrive il funzionamento della parte **operativa** è:

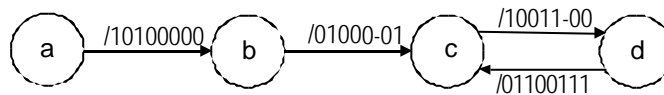
1.  $R(0:7) \leftarrow \text{DATI}, C \leftarrow \text{IND};$
2.  $R(8:15) \leftarrow \text{DATI}$
3.  $\text{MBR} \leftarrow R, \text{MAR} \leftarrow C, R(0:7) \leftarrow \text{DATI};$
4.  $M[\text{MAR}] \leftarrow \text{MBR}, C \leftarrow \text{INCR}(C), R(8:15) \leftarrow \text{DATI}$
5. goto 3

Le **istruzioni** del sistema in cui gli 8 segnali di **comando** sono stati indicati con  $a_1, a_2, \dots, a_8$  sono indicati nella seguente tabella:

	$A_{R(0:7)}$	$A_{R(8:15)}$	$A_C$	$A_{\text{MAR}}$	$A_{\text{MBR}}$	$K_C$	S	SEL	
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$	$a_6$	$a_7$	$a_8$	
1:	1	0	1	0	0	0	0	0	
2:	0	1	0	0	0	-	0	1	
3:	1	0	0	1	1	-	0	0	
4:	0	1	1	0	0	1	1	1	goto 3

La parte di controllo emetterà in una prima fase, nell'ordine i codici (**istruzioni**) 1, 2, 3, 4 e poi ciclicamente 3 e 4.

## Progettazione della parte di controllo



Stato	P.S.	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$	$y_1 y_2$	$y'_1 y'_2$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$
a	b,	1	0	1	0	0	0	0	0	00	01,	1	0	1	0	0	0	0	0
b	c,	0	1	0	0	0	-	0	1	01	11,	0	1	0	0	0	-	0	1
c	d,	1	0	0	1	1	-	0	0	11	10,	1	0	0	1	1	-	0	0
d	c,	0	1	1	0	0	1	1	1	10	11,	0	1	1	0	0	1	1	1

## Progettazione della parte di controllo (2)

$y_1$	$y_2$	$y'_1$	$y'_2$	$y_1$	$y_2$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$
0	0	0	1	0	0	1	0	1	0	0	0	0	0
0	1	1	1	0	1	0	1	0	0	0	-	0	1
1	1	1	0	1	1	1	0	0	1	1	-	0	0
1	0	1	1	1	0	0	1	1	0	0	1	1	1

$$y'_1 = \overline{y_1} \cdot y_2 + y_1 \cdot \overline{y_2} + y_1 \cdot y_2$$

$$y'_2 = \overline{y_1} \cdot \overline{y_2} + \overline{y_1} \cdot y_2 + y_1 \cdot \overline{y_2}$$

$$a_1 = \overline{y_1} \cdot \overline{y_2} + y_1 \cdot y_2$$

$$a_2 \equiv a_8 = \overline{y_1} \cdot y_2 + y_1 \cdot \overline{y_2}$$

$$a_3 = \overline{y_1} \cdot \overline{y_2} + y_1 \cdot y_2$$

$$a_4 \equiv a_5 = y_1 \cdot y_2$$

$$a_6 \equiv a_7 = y_1 \cdot \overline{y_2}$$

$$y'_1 = y_1 + y_2$$

$$y'_2 = \overline{y_1} + \overline{y_2}$$

$$a_1 = \overline{y_1} \cdot \overline{y_2} + y_1 \cdot y_2$$

$$a_2 \equiv a_8 = \overline{y_1} \cdot y_2 + y_1 \cdot \overline{y_2}$$

$$a_3 = y_2$$

$$a_4 \equiv a_5 = y_1 \cdot y_2$$

$$a_6 \equiv a_7 = y_1 \cdot \overline{y_2}$$

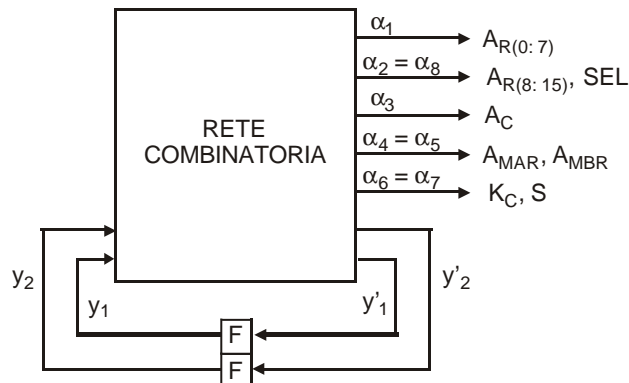
Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

11

## Struttura della parte di controllo

E' costituita dalla **rete combinatoria** che implementa le **funzioni di comando (uscita)** e dalle **due funzioni del prossimo stato** ed ha **due flip-flop di tipo D** per il trasferimento delle **variabili interne al prossimo stato**.



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

12

## Parte di controllo con istruzioni

Eliminiamo, ora, la **prima delle ipotesi semplificative** del sistema considerato, supponendo che la stessa parte di controllo del sistema che abbiamo sintetizzato, abbia degli ingressi a cui sono applicate **dei valori booleani con significato di istruzioni**.

**Manteniamo, invece, l'ipotesi che la parte operativa non invii condizioni.**

Per limitare al massimo le complicazioni supponiamo che la **variabile d'ingresso sia solo una**, che indichiamo con  $I$  e che quando essa è uguale ad "1" il sistema lavori e quando vale "0" il sistema si arresti.

Per quanto riguarda la **parte operativa** tutto rimane come in precedenza.

Per la **parte di controllo**, invece, la tabella di flusso deve ora prevedere due colonne, una relativa alla condizione  $I = 0$  ed una relativa alla condizione  $I = 1$ .

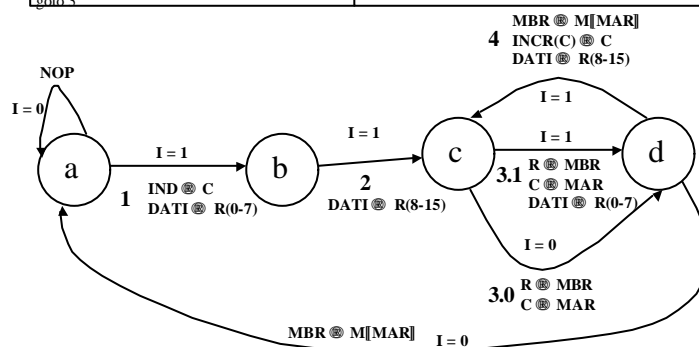
Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

13

## Diagramma di flusso

$I=1$	$I=0$
Wait : If $I = 0$ Goto wait	Wait : If $I = 0$ Goto wait
Else	Else
1. $R(0:7) \leftarrow \text{DATI}, C \leftarrow \text{IND};$	1. $R(0:7) \leftarrow \text{DATI}, C \leftarrow \text{IND};$
2. $R(8:15) \leftarrow \text{DATI}$	2. $R(8:15) \leftarrow \text{DATI}$
3.1 $\text{MBR} \leftarrow R, \text{MAR} \leftarrow C, R(0:7) \leftarrow \text{DATI}$	3.0 $\text{MBR} \leftarrow R, \text{MAR} \leftarrow C$
4. $M[\text{MAR}] \leftarrow \text{MBR}, C \leftarrow \text{INCR}(C), R(8:15) \leftarrow \text{DATI}$	4. $M[\text{MAR}] \leftarrow \text{MBR}$
$\leftarrow \text{DATI}$	goto wait
goto 3	



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

14

## La tabella di flusso

Stato	$I = 0$								$I = 1$										
a	a ,	0	0	0	0	0	-	0	-	b ,	1	0	1	0	0	0	0	0	0
b	- ,	-	-	-	-	-	-	-	-	c ,	0	1	0	0	0	-	0	1	
c	d ,	0	0	0	1	1	-	0	-	d ,	1	0	0	1	1	-	0	0	
d	a ,	0	0	0	0	0	-	1	-	c ,	0	1	1	0	0	1	1	1	
	P.S.	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$	P.S.	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$	

Quando  $I = 0$ , la rete rimane nello stato iniziale **a** generando una **mistruzione che non abilita nessun registro**, tutti i comandi **a** sono "0", salvo  $\alpha_6$ , (comando di incremento del contatore **C**), ed  $\alpha_8$ , (bit che indirizza il dato nella parte bassa, oppure in quella alta del registro **A**) che sono, invece, segnali non definiti.

Quando  $I = 1$  la rete passa dallo stato **a** a quello **b** e poi da **b** a **c** e così via, generando i comandi indicati nella **parte destra della tabella di flusso che è identica a quella del sistema precedente** nel quale non era previsto l'ingresso per l'istruzione.

## Alcune considerazioni sistemistiche

**Si osservi che la distinzione tra livello 0, inteso come livello non attivo, e segnale non definito, può perdere significato, quando non si usano driver tristate.**

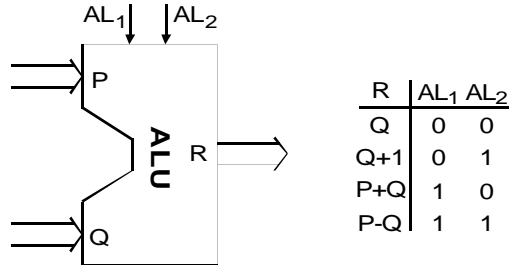
Quando l'ingresso  $I$  viene messo a "0", **l'attività del sistema deve cessare**. E' ragionevole supporre che ciò debba avvenire dopo che l'ultimo byte è stato trasferito dal canale DATI alla parte bassa del registro **A**, e cioè, **quando la parte di controllo si trova nello stato c**.

Quando ciò avviene il prossimo stato deve essere letto nella colonna di sinistra della tabella di flusso che indica ancora **d**, ma con un set di comandi diverso, perché, in questo caso  $\alpha_1$  non deve abilitare il caricamento di un nuovo byte e la selezione non è definita. Il passo successivo deve essere solamente la scrittura in memoria dell'ultima parola ed il trasferimento nello stato iniziale.

Il prossimo stato, partendo dallo stato **b**, quando  $I = 0$ , non è definito, così come non sono definiti i comandi, perché questa combinazione tra stato della rete ed ingresso non si può verificare.

## Un nuovo esempio di sistema

Verrà utilizzata una ALU con le caratteristiche riassunte in figura.



Vogliamo progettare un sistema costituito da un registro P a complementazione, un registro Q ed una ALU del tipo descritto, che, operando su numeri interi, sia in grado di eseguire le seguenti operazioni.

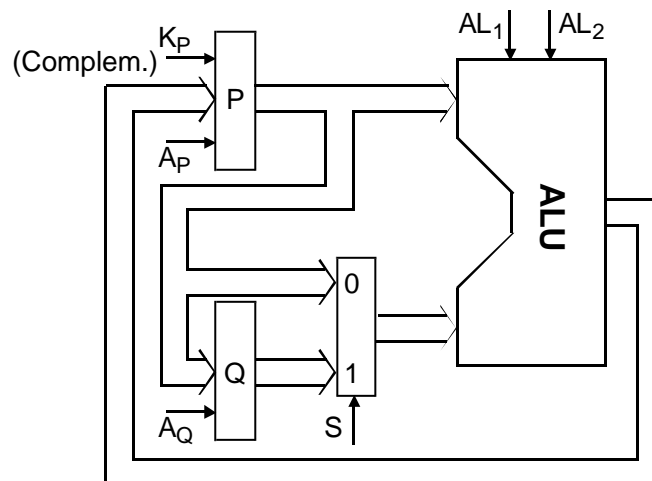
I <sub>0</sub>	I <sub>1</sub>	Operazione
0	0	ALT
0	1	$-P - Q \text{ @ } P$
1	1	$Q - P \text{ @ } Q$

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

17

## Parte operativa



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

18

## ALU con 3 istruzioni senza condizioni: flowchart

00 0: NOP goto 0

1:  $P+Q \oplus P$

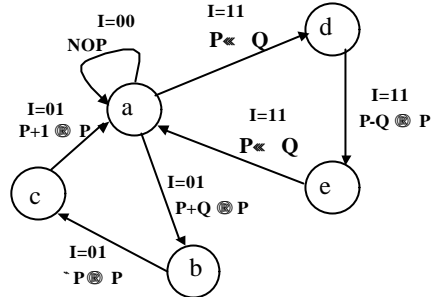
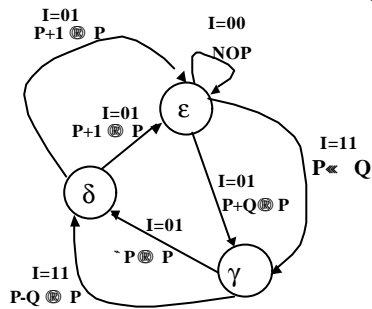
01 2:  $\sim P \oplus P$

3:  $P+1 \oplus P$  goto 1

1:  $P \ll Q$

11 2:  $P-Q \oplus P$

3:  $P \ll Q$  goto 1



Stati risultanti

$\epsilon = (a)$

$\gamma = (b, d)$

$\delta = (c, e)$

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

19

## msequenze e mistruzioni (1)

Istruzioni	$\mu$ sequenze	$A_P$	$A_Q$	$K_P$	$AL_1$	$AL_2$	$S$
		$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$
0 0	$f$	0	0	-	-	-	-
0 1	1: $P+Q \oplus P$ ;	1	0	0	1	0	1
	$COMP(P) \oplus P$ ;	1	0	1	-	-	-
	$P+1 \oplus P$ , goto 1	1	0	0	0	1	0
1 1	1: $P \oplus Q, Q \oplus P$ ;	1	1	0	0	0	1
	$P-Q \oplus P$ ;	1	0	0	1	1	1
	$P \oplus Q, Q \oplus P$ , goto 1	1	1	0	0	0	1

### istruzione con codice 01:

il calcolo dell'espressione  $-P-Q$  si effettua trasformandola in:  $-(P+Q)$ , ove il cambio di segno è eseguito secondo le regole del complemento a 2 e, cioè, complementando il numero ed aggiungendo 1. Quest'ultima operazione viene effettuata con la ALU, immettendo il contenuto di  $P$  nell'ingresso inferiore, dal momento che la ALU può effettuare l'incremento solo sull'ingresso  $Q$ .

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

20

### µsequenze e istruzioni (2)

Istruzioni	µsequenze	A <sub>P</sub> α <sub>1</sub>	A <sub>Q</sub> α <sub>2</sub>	K <sub>P</sub> α <sub>3</sub>	AL <sub>1</sub> α <sub>4</sub>	AL <sub>2</sub> α <sub>5</sub>	S α <sub>6</sub>
0 0	<b>f</b>	0	0	-	-	-	-
0 1	1: P + Q @ P;	1	0	0	1	0	1
	COMP(P) @ P;	1	0	1	-	-	-
	P + I @ P, goto 1	1	0	0	0	1	0 goto 1
1 1	1: P @ Q, Q @ P;	1	1	0	0	0	1
	P - Q @ P;	1	0	0	1	1	1
	P @ Q, Q @ P, goto 1	1	1	0	0	0	1 goto 1

**istruzione con codice 11:**

una delle possibili soluzioni offerte dalla struttura della rete è di scambiare il contenuto dei registri per effettuare la differenza Q - P perché la ALU effettua automaticamente la differenza tra l'ingresso P e quello Q, quando i due bit di controllo sono 1 1. Il trasferimento Q @ P viene effettuato attraverso la ALU. La terza operazione scambia di nuovo il contenuto dei registri perché il risultato dell'operazione si trovi in Q come richiesto.

### µsequenze e istruzioni (3)

Istruzioni	µsequenze	A <sub>P</sub> α <sub>1</sub>	A <sub>Q</sub> α <sub>2</sub>	K <sub>P</sub> α <sub>3</sub>	AL <sub>1</sub> α <sub>4</sub>	AL <sub>2</sub> α <sub>5</sub>	S α <sub>6</sub>
0 0	<b>f</b>	0	0	-	-	-	-
0 1	1: P + Q @ P;	1	0	0	1	0	1
	COMP(P) @ P;	1	0	1	-	-	-
	P + I @ P, goto 1	1	0	0	0	1	0 goto 1
1 1	1: P @ Q, Q @ P;	1	1	0	0	0	1
	P - Q @ P;	1	0	0	1	1	1
	P @ Q, Q @ P, goto 1	1	1	0	0	0	1 goto 1

**Chiusura delle operazioni:**

**goto 1** significa che l'istruzione continua ad essere eseguita finché il corrispondente codice è presente sugli ingressi della parte di controllo. E' ovvio che la µsequenza non può essere interrotta e, quindi, qualora il codice fosse tolto prima della conclusione dell'istruzione, essa deve essere completata. La parte di controllo del sistema presupporrà che il cambio di istruzione si possa effettuare solo ad operazione conclusa e, cioè solo quando la rete si trova nello stato di ALT.

**In pratica il passaggio dall'istruzione con codice 01 a quella 11 o viceversa si può fare solo passando per il codice 00.**

Stato	$I_0 I_1$	
	0 0	0 1
a	a, 00----	b, 100101
b	-, -----	c, 101----
c	-, -----	a, 100010
d	-, -----	-, -----
e	-, -----	a, 110001

## Sintesi della rete di controllo

Stato	Prossimo stato, $\alpha_1, \dots, \alpha_6$		
	0 0	0 1	1 1
$\varepsilon: (a)$	$\varepsilon, 00----$	$\gamma, 100101$	$\gamma, 110001$
$\gamma: (bd)$	-, -----	$\delta, 101----$	$\delta, 100111$
$\delta: (ce)$	-, -----	$\varepsilon, 100010$	$\varepsilon, 110001$

Classi di compatibilità  
(a), (bd), (be), (cd), (ce)

**Tabella minima per tentativi**

Dalla tabella di flusso si può passare a quella di transizione e da questa estrarre le due funzioni che danno i valori delle due variabili interne  $y_1'$  ed  $y_2'$  nel prossimo stato e le 6 funzioni d'uscita.

**Le otto funzioni compongono la rete combinatoria che occorre per sintetizzare la rete sequenziale sincrona di controllo che utilizza due flip-flop di tipo D per memorizzare il prossimo stato.**

## Sistema senza istruzioni esterne I, ma con una condizione b.

Non rimane, a questo punto che concentrarsi sull'ultimo elemento che resta per completare il quadro di un sistema completo, le "condizioni **B**".

Nel caso più generale di sistema di elaborazione, infatti, la parte operativa manda alla parte di controllo, durante l'esecuzione di una sequenza di operazioni, dei segnali detti "condizioni".

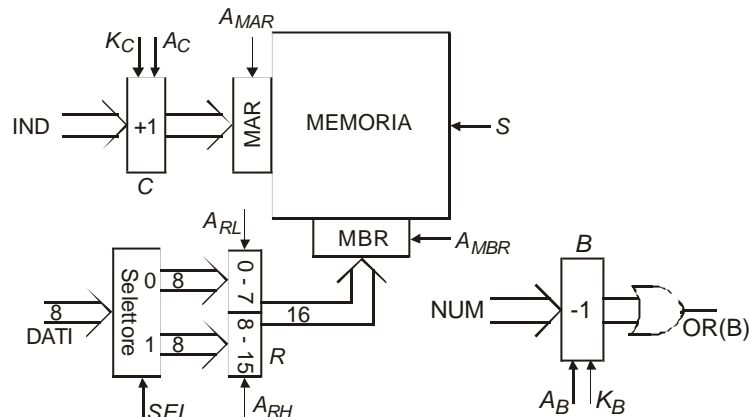
Un semplice esempio può essere fornito dal sistema visto prima, quello che scriveva in memoria dei dati senza ricevere ne' istruzioni ne' condizioni.

Possiamo, adesso, rimuovere la seconda ipotesi. **Si può immaginare che, contemporaneamente al caricamento dell'indirizzo di partenza in memoria, venga fornito da un canale, detto NUM, il numero di parole da 16 bit da registrare in memoria.**

Quando il numero di parole sarà stato raggiunto il sistema dovrà arrestarsi.

## Aggiornamento parte operativa

In queste nuove ipotesi, alla parte operativa precedente va aggiunto un contatore,  $B$ , che sarà decrementato di una unità ad ogni scrittura in memoria. Le uscite del registro  $B$  sono collegate ad un OR che darà valore "0" quando il numero di parole stabilite sarà stato raggiunto.

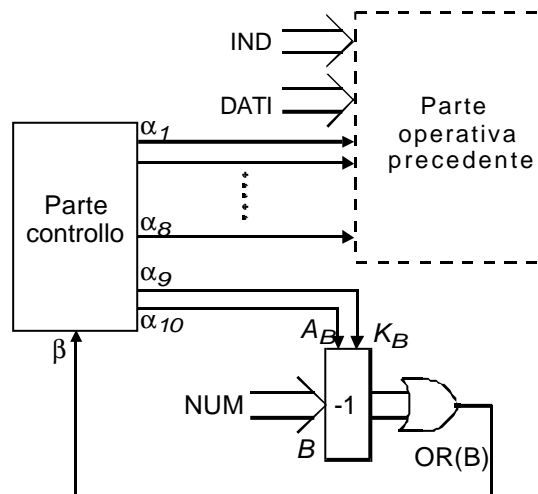


Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

25

## Aggiornamento della parte di controllo



La parte di controllo dovrà gestire altri due comandi  $\alpha_9$  ed  $\alpha_{10}$  che servono per caricare inizialmente il contatore e per decrementarlo ad ogni ciclo di scrittura e dovrà ricevere il segnale  $\beta = 0$  quando i cicli previsti sono stati eseguiti ed il sistema si deve fermare.

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

26

## Microsequenza operativa

```

1: IND → C, DATI → R(0-7), NUM → B
2: DATI → R(8-15), DECR(B) → B
3: if OR(B) = "1"
    then R → MBR, C → MAR, DATI → R(0-7)
        4: MBR → M[MAR], INCR(C) → C, DATI → R(8-15), DECR(B) → B, goto 3
    else R → MBR, C → MAR
        5: MBR → M[MAR]
    fi
  
```

L'associazione tra i comandi  $\alpha$  e la condizione  $\beta$  ed i segnali della parte operativa è la seguente:

$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$\alpha_5$	$\alpha_6$	$\alpha_7$	$\alpha_8$	$\alpha_9$	$\alpha_{10}$	$\beta$
$A_{RL}$	$A_{RH}$	$A_C$	$A_{MAR}$	$A_{MBR}$	$K_C$	$S$	$SEL$	$A_B$	$K_B$	$OR(B)$

In pratica i comandi fino ad  $a_8$  hanno lo stesso significato della rete precedente.

Si aggiungono solo due nuovi comandi  $a_9 = A_B$  ed  $a_{10} = K_B$  che comandano il caricamento ed il decremento del registro B.

Marzo 2002

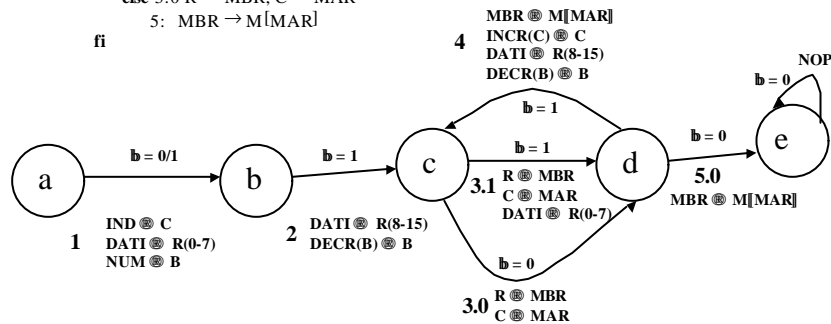
Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

27

## Diagramma di flusso

```

1: IND → C, DATI → R(0-7), NUM → B
2: DATI → R(8-15), DECR(B) → B
3: if OR(B) = "1"
    then 3.1: R → MBR, C → MAR, DATI → R(0-7)
        4: MBR → M[MAR], INCR(C) → C, DATI → R(8-15), DECR(B) → B, goto 3
    else 3.0 R → MBR, C → MAR
        5: MBR → M[MAR]
    fi
  
```



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

28

## La tabella di flusso della parte di controllo

Stato	$\beta$	
	0	1
a	b, 1 0 1 0 0 0 0 0 1 0	b, 1 0 1 0 0 0 0 0 1 0
b	-, - - - - - - - - - -	c, 0 1 0 0 0 - 0 1 1 1
c	d, 0 0 0 1 1 - 0 - 0 -	d, 1 0 0 1 1 - 0 0 0 -
d	e, 0 0 0 0 0 - 1 - 0 -	c, 0 1 1 0 0 1 1 1 1 1
e	e, 0 0 0 0 0 - 0 - 0 -	-, - - - - - - - - - -

Prossimo stato,  $\alpha_1, \dots, \alpha_{10}$

Quando comincia a funzionare, la rete è nello stato **a** e la condizione **b** può essere 0 o 1, per effetto di elaborazioni precedenti. Quando, però, si carica il registro **B** con NUM (> 0), **b** diventa sicuramente 1 ed allora si passa nello stato **b** e poi la  $\mu$ sequenza si sviluppa secondo la colonna di destra della tabella di flusso (**b=1**).

E' facile rendersi conto che la tabella traduce in termini booleani la  $\mu$ sequenza di  $\mu$ operazioni vista in precedenza.

## Completamento della sintesi

Stato	$\beta$	
	0	1
a	b, 1 0 1 0 0 0 0 0 1 0	b, 1 0 1 0 0 0 0 0 1 0
b	-, - - - - - - - - - -	c, 0 1 0 0 0 - 0 1 1 1
c	d, 0 0 0 1 1 - 0 - 0 -	d, 1 0 0 1 1 - 0 0 0 -
d	e, 0 0 0 0 0 - 1 - 0 -	c, 0 1 1 0 0 1 1 1 1 1
e	e, 0 0 0 0 0 - 0 - 0 -	-, - - - - - - - - - -

Prossimo stato,  $\alpha_1, \dots, \alpha_{10}$

La tabella di flusso può essere ridotta perché **b**  $\gg$  **e**.

Dalla tabella ridotta si sintetizza la rete sequenziale corrispondente con due sole variabili interne, essendo gli stati ridotti a quattro. Dalla tabella di transizione dell'automa si estraggono le tre tabelle di transizione delle funzioni  $z$ ,  $y_1'$  ed  $y_2'$ , che, dopo eventuale minimizzazione con le mappe di Karnaugh, possono essere sintetizzate e vanno a costituire la parte combinatoria della rete sequenziale sincrona di controllo, secondo la solita struttura con due flip-flop "D".

## Un esempio di sistema completo (con “*istruzioni*” e “*condizioni*”)

Supponiamo di dover progettare un sistema costituito da **due registri a scorrimento verso destra**,  $P$  e  $Q$ .

Il sistema lavora su numeri interi ed è in grado di eseguire le due istruzioni da 1 bit:

$I = 0$ : ALT

$I = 1$ : **if** ( $P + Q$  non da “*supero*”) **then**  $P + Q \rightarrow P$   
**else**  $P/2 + Q/2 \rightarrow P$

**Prima di affrontare il problema principale chiariamo il significato della parola “*supero*”.**

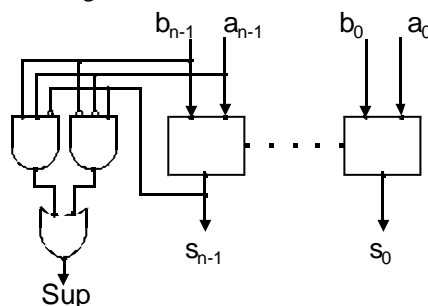
## Significato di “*supero*”

Il “*supero*” o “*overflow*”, in inglese, si ha quando, in una operazione di somma algebrica nella aritmetica complemento a due, si ha un riporto che eccede la capacità del registro che deve ricevere il risultato.

Il “*supero*” si può avere solo sommando due numeri dello stesso segno.

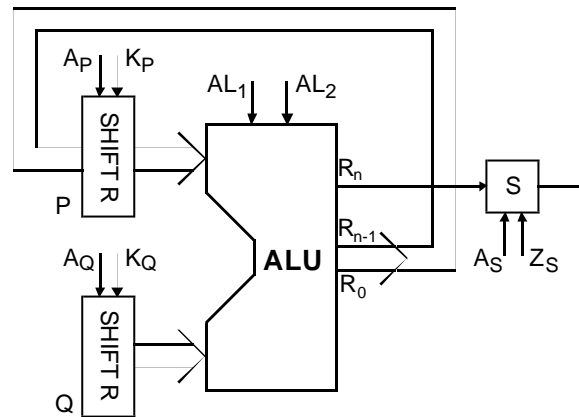
Il trabocco del riporto dai bit assegnati al numero verso il bit segno farà assumere al risultato segno opposto rispetto ai due operandi.

Basterà, quindi, che il sommatore sia dotato di un semplice circuito accessorio per essere in grado di rilevare la condizione di *supero*.



## Rete operativa del sistema

Supponiamo che la ALU su cui si basa la parte operativa del sistema che stiamo progettando abbia un sommatore dotato dell'uscita di "supero".



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

33

## Considerazioni sulla parte operativa

L'uscita  $R_n$  del sommatore è quella di "overflow" e viene caricata in un registro da 1 bit,  $S$  il cui stato sarà successivamente verificato per riconoscere una eventuale situazione di supero.

Il flip-flop  $S$  deve avere un ingresso di "azzeramento", "clear" in inglese.

Il segnale di azzeramento  $Z_s$  opera quando è attivo anche il segnale di abilitazione  $A_s$ . Per evitare incertezze, supponiamo che nello stato iniziale  $S$  sia a 0 e, pertanto, **provvederemo ad azzerarlo prima della chiusura delle operazioni.**

E' evidente che bisogna supporre che quando la rete sarà messa in funzione per la prima volta una apposita "procedura di **inizializzazione**" provvederà ad effettuare questa operazione.

Siamo ora in grado di sviluppare le **sequenze** corrispondenti ai **codici d'istruzione**

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

34

## ALU con 2 istruzioni e condizione: flowchart

0: NOP goto 0

1:  $(P+Q)n \oplus S$

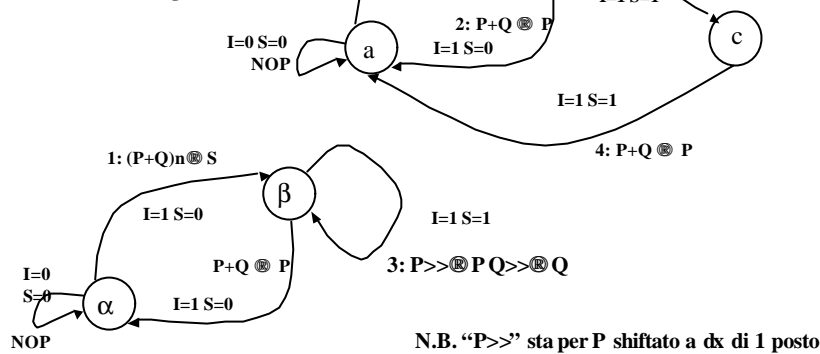
if  $S=0$  then

2:  $P+Q \oplus P$  goto 1

else

3:  $P \gg \oplus P \gg \oplus Q$

4:  $P+Q \oplus P$  goto 1



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

35

## *μ*sequenze, *μ*operazioni e *μ*istruzioni

Istruzioni	<i>μ</i> sequenze
0	<b>f</b>
1	1: $(P+Q)n \rightarrow S$ 2: <b>if</b> $S=0$ <b>then</b> $P+Q \rightarrow P$ , <b>goto</b> 1 <b>else</b> 3: $SD(P) \rightarrow P$ , $SD(Q) \rightarrow Q$ , '0' $\rightarrow S$ 4: $P+Q \rightarrow P$ , <b>goto</b> 1
	<b>fi</b>

<i>μ</i> operazioni	<i>μ</i> istruzioni							
	$A_P$ $\alpha_1$	$A_Q$ $\alpha_2$	$A_S$ $\alpha_3$	$K_P$ $\alpha_4$	$K_Q$ $\alpha_5$	$Z_S$ $\alpha_6$	$AL_1$ $\alpha_7$	$AL_2$ $\alpha_8$
$\phi$	0	0	0	-	-	-	-	-
1:	0	0	1	-	-	0	1	0
2: <b>if</b> $S = "0"$ <b>then</b>	1	0	0	0	-	-	1	0
3: <b>else</b>	1	1	1	1	1	1	-	-
4:	1	0	0	0	-	-	1	0
							<b>goto</b> 1	<b>fi</b>

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

36

## La tabella di flusso

L'espressione "goto 1" tra le istruzioni da eseguire, indica che il sistema deve passare ad eseguire nuovamente la operazione 1 e ciò avverrà ripetutamente, finché il codice d'istruzione  $I = 1$ , non viene cambiato in  $I = 0$ .

Se supponiamo che il cambio dell'istruzione possa avvenire solo quando l'esecuzione dell'istruzione in corso è terminata, la tabella di flusso della parte di controllo è quella di figura 18, dove la mezza tabella relativa alla condizione  $I = 0$  è specificata solo per lo stato  $a$ .

Stato \ I, S		0 0		1 0		1 1		1 0	
		a, 0 0 0		-, -		-, -		b, 0 0 1 -- 0 1 0	
a		a, 0 0 0		-, -		-, -		b, 0 0 1 -- 0 1 0	
b		-, -		-, -		c, 1 1 1 1 1 1		a, 1 0 0 0 -- 1 0	
c		-, -		-, -		-, -		a, 1 0 0 0 -- 1 0	

Prossimo stato,  $\alpha_1, \dots, \alpha_8$

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

37

## Riduzione della tabella di flusso

E' ovvio che i valori logici dei comandi  $\alpha$  per ciascuno stato vanno messi in relazione con la parte operativa tenendo presente il significato di ciascuno.

La tabella deve essere studiata attentamente per capire il perché di tanti dettagli.

Per esempio la colonna  $I = 1, S = 1$  ha lo stato  $a$  completamente non specificato perché, avendo supposto che nello stato iniziale  $S = 0$ , la combinazione non può verificarsi. Anche lo stato  $c$  della colonna 1 1 non può essere raggiunto perché dallo stato  $b$  si passa a  $c$  azzerando il flip-flop  $S$  ( $\alpha_3$  ed  $\alpha_6$  sono ad 1).

Dall'esame della tabella di flusso è abbastanza ovvio che  $b \gg c$  e la tabella può essere ridotta a:

Stato \ I, S		0 0		1 0		1 1		1 0	
		$\gamma, 0 0 0$		-, -		-, -		$\epsilon, 0 0 1 -- 0 1 0$	
(a): $\gamma$		$\gamma, 0 0 0$		-, -		-, -		$\epsilon, 0 0 1 -- 0 1 0$	
(b,c): $\epsilon$		-, -		-, -		$\epsilon, 1 1 1 1 1 1$		$\gamma, 1 0 0 0 -- 1 0$	

Prossimo stato,  $\alpha_1, \dots, \alpha_8$

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

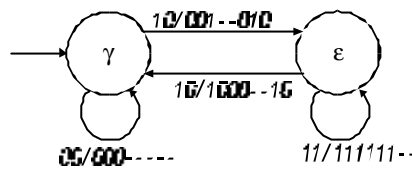
38

## Diagramma degli stati

Stato	$I, S$	0 0	1 0	1 1	1 0
(a): $\gamma$		$\gamma, 000$	$\epsilon, 000$	$\epsilon, 111111$	$\epsilon, 001$
(b,c): $\epsilon$		$\gamma, 100$	$\epsilon, 111111$	$\epsilon, 111111$	$\gamma, 100$
Prossimo stato, $\alpha_1, \dots, \alpha_8$					

Questa tabella di flusso ha caratteristiche molto interessanti perché pur avendo solo due stati consente una  $\mu$ sequenza a tre passi, perché dopo un ciclo di permanenza nello stato  $\epsilon$ , l'automata rimane ancora, per un ciclo, nello stesso stato, dopo l'azzeramento del registro S.

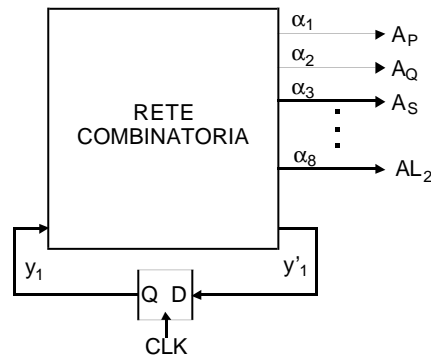
Il diagramma a stati dell'automata è:



## Sintesi della rete sequenziale di controllo

Dalla tabella di flusso ridotta, assegnato ad esempio il codice  $y_1 = 0$  allo stato  $\epsilon$  e quello  $y_1 = 1$  allo stato  $\gamma$  si ricavano le tabelle di transizione della funzione del prossimo stato,  $y_1'$  e quelle delle funzioni d'uscita  $\alpha_1, \dots, \alpha_8$ . Le relative espressioni sono:  $y_1' = y_1 * S + y_1 I$ ,  $\alpha_1 = y_1 S$ ,  $\alpha_2 = y_1 I$ , ..... e così via.

Lo schema a blocchi della parte di controllo è il seguente:



## Sistemi con controllo microprogrammato .

I sistemi che abbiamo esaminato finora sono sistemi che effettuano operazioni semplici e la loro parte di controllo è sempre stata sintetizzata come minima rete sequenziale sincrona del relativo automa, ma, **quando i sistemi aumentano di complessità**, il processo di **sintesi della parte di controllo**, che è già abbastanza laborioso nei casi semplici, diventa decisamente **poco gestibile**.

Poiché abbiamo già visto che per **realizzare funzioni combinatorie** complesse si può far ricorso alla tecnica della *“look-up table”*, che rinuncia a priori alla possibilità di minimizzare la rete, ma consente, in cambio, una grande semplicità di sviluppo del progetto, **la parte di controllo di un sistema complesso può essere basata su una rete combinatoria implementata attraverso una “tabella funzione” memorizzata in una Read Only Memory (ROM)**.

In questo caso il controllo del sistema si dice **“microprogrammato”**

Il trasferimento della struttura della rete sequenziale di controllo nella nuova tecnica è, in linea di principio, molto semplice.

## ROM di microprogrammazione

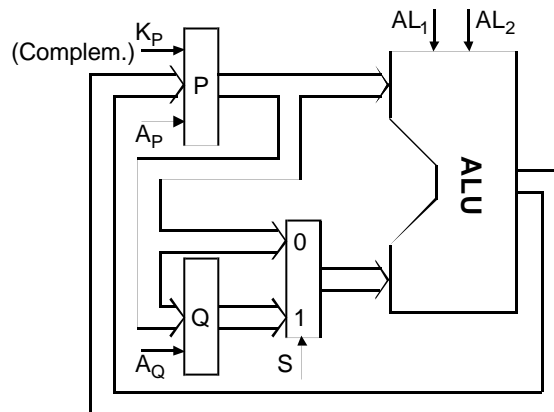
Sappiamo che la rete combinatoria della parte di controllo del sistema riceve in ingresso, dall'esterno, i codici di istruzione ***I***, dalla parte operativa, una combinazione di valori booleani delle condizioni ***b*** e, dai flip-flop di stato, la combinazione delle variabili interne nel prossimo stato.

- L'unione dei codici binari in cui l'**istruzione** (codice operativo) costituisce la **parte più significativa**, le condizioni quella intermedia e le variabili di stato *y*, quella **meno significativa**, viene usata come **indirizzo di lettura** della ROM di  $\mu$ programmazione
- Nella corrispondente **locazione della ROM** sono memorizzate la **istruzione** (bit di comando) da eseguire e la **combinazione delle variabili interne *y***, che specificano il prossimo stato della rete sequenziale.

Il numero di bit per locazione è la somma del numero dei comandi e quello delle variabili interne

## Un primo esempio

Per chiarire con un caso pratico il significato di controllo  $\mu$ programmato, possiamo usare uno degli esempi di parte di controllo, sintetizzati in precedenza con la tecnica della rete sequenziale minima. La parte operativa dell'esempio è:



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

43

## Tabella di flusso/transizione

La parte operativa è costituita da una ALU che esegue operazioni aritmetiche sul contenuto di due registri P e Q e la tabella di flusso minima è riproposta in figura.

**Accanto agli stati è stato inserito il corrispondente codice.**

		$I_0 I_1$		
$Y_1 Y_0$	Stato	0 0	0 1	1 1
0 0	$\epsilon$ : (a)	$\epsilon$ , 00 - - - -	$\gamma$ , 100101	$\gamma$ , 110001
0 1	$\gamma$ : (bd)	-, - - - - - -	$\delta$ , 101 - - -	$\delta$ , 100111
1 0	$\delta$ : (ce)	-, - - - - - -	$\epsilon$ , 100010	$\epsilon$ , 110001

Prossimo stato,  $\alpha_1, \dots, \alpha_6$

Si tratta di una struttura semplice con istruzioni a due bit e nessuna condizione.

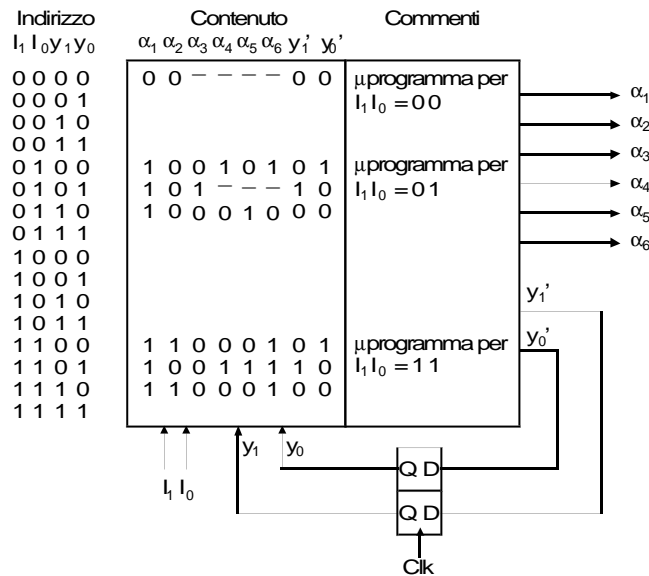
La parte combinatoria della rete sequenziale di controllo può essere realizzata con semplicità con una ROM

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

44

## Il controllo microprogrammato



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

45

## Contenuto della ROM

I codici d'istruzione assumono il ruolo di bit più significativi dell'indirizzo. Non essendo utilizzato il codice 10 il relativo blocco di locazioni è vuoto.

Gli ultimi due bit di tutte le locazioni contengono il codice del prossimo stato ed **i corrispondenti bit del registro d'uscita della memoria sono direttamente connessi con gli ingressi dei due flip-flop D sincronizzati dal clock**. Questo significa che se si sta eseguendo, per esempio, la prima  $\mu$ istruzione del  $\mu$ programma relativo all'istruzione 11, che comincia alla locazione 1 1 0 0 (C in esadecimale, 12 in decimale), i comandi saranno  $A_p = 1$ ,  $A_Q = 1$ ,  $K_p = 0$ ,  $AL_1 = 0$ ,  $AL_2 = 0$ ,  $S = 1$  e la prossima  $\mu$ istruzione da eseguire sarà all'indirizzo 1 1 0 1 ( $D_{16} / 13_{10}$ )

**La capacità della ROM necessaria è di 16 locazioni da 8 bit.**

E' evidente, che essendo alcuni comandi non specificati ed uno dei codici di stato non utilizzato, **l'uso della memoria è scarso**, ma questo è ampiamente compensato dal fatto che **le memorie sono i componenti che meglio sfruttano la superficie del cristallo di silicio** ed hanno, pertanto, un costo per bit molto basso.

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

46

## Considerazioni sulle “memorie a sola lettura”

Le ROM sono componenti il cui contenuto deve essere programmato in fase di costruzione e, una volta prodotte, non possono più essere cambiate. Poiché la costruzione di una ROM con un prefissato contenuto, richiede costi di programmazione piuttosto elevati, questa soluzione è riservata a sistemi destinati ad essere prodotti in molte migliaia di esemplari.

Per fortuna, però, esistono vari tipi di *memorie a sola lettura* che possono essere programmate direttamente in laboratorio, con apparecchiature abbastanza economiche. Ciò consente di estendere la tecnica della  $\mu$ programmazione anche alla realizzazione di sistemi da riprodurre in pochi esemplari ed ai prototipi della produzione di serie.

A seconda del meccanismo di programmazione e della possibilità di essere cancellate e riscritte queste memorie a sola lettura si chiamano “*Programmable Read Only Memory*” (**PROM**), “*Erasable Read Only Memory*” (**EPROM**), “*Electrically Alterable Read Only Memory*” (**EAROM**).

## $\mu$ programmazione con istruzioni e condizioni

Supponiamo di dover realizzare un sistema con tre registri, P, Q, R, contenenti interi positivi, in grado di eseguire le seguenti istruzioni:

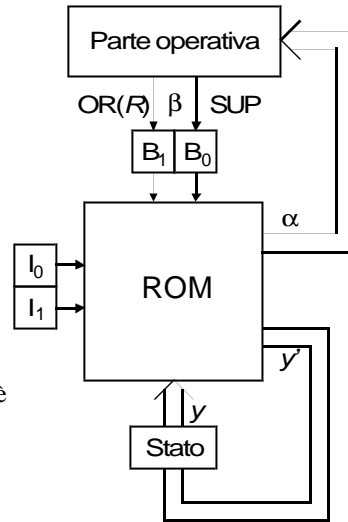
$$\begin{array}{lll} I_1 & I_0 & \\ 0 & 0 & \text{alt} \\ 0 & 1 & \text{if } (P + Q \text{ non da supero}) \text{ then } P + Q \rightarrow P \\ & & \text{else } P / 2 + Q / 2 \rightarrow P \\ 1 & 0 & P / 2^R \rightarrow P \end{array}$$

Tralasciamo di descrivere nei dettagli la parte operativa del sistema, la cui struttura sarà basata su una ALU con circuito di rilevamento di una eventuale situazione di supero. Avrà i due registri P e Q con possibilità di scorrimento a destra per la divisione per due ed un registro R con possibilità di conteggio all'indietro per l'operazione di decremento. Un circuito di OR sui bit di contenuto di R servirà a rilevarne l'azzeramento ( $OR(R) = 0$ ) al raggiungimento del numero R di iterazione della divisione per 2 del contenuto di P.

## Schema generale

La condizione,  $\beta_1 = \text{OR}(R)$  e quella,  $\beta_0 = \text{SUP}$ , sono memorizzate nei flip-flop  $B_1 B_0$  che costituiscono il **registro di condizione**. In maniera perfettamente simile il codice d'istruzione è memorizzato nei flip-flop  $I_1 I_0$  che sono il **registro istruzione**. In pratica i due registri costituiscono insieme il **registro indirizzo della ROM di programmazione**.

Lo schema generale del sistema è riportato nella figura, nella quale non è disegnata la rete che provvede alla distribuzione del segnale di sincronizzazione o "clock" a tutti i registri.



## μsequenze e μistruzioni

$I_1 I_0$	μsequenze	Segnali $\alpha$
0 0	$\mathcal{F}$	$\mu_1$
0 1	1: $\text{SUP} \rightarrow B_0$ ; <b>if</b> $B_0 = "0"$ <b>then</b> $P + Q \rightarrow P$ , goto 1; <b>else</b> $\text{SD}(P) \rightarrow P$ , $\text{SD}(Q) \rightarrow Q$ ; $P + Q \rightarrow P$ , "0" $\rightarrow B_0$ , goto 1 <b>fi</b>	$\mu_2$ $\mu_3$ $\mu_4$ $\mu_5$
1 0	1: $\text{OR}(R) \rightarrow B_1$ ; <b>if</b> $B_1 = "0"$ ; <b>then</b> $\mathcal{F}$ , goto 1 <b>else</b> $\text{SD}(P) \rightarrow P$ , $\text{DECR}(R) \rightarrow R$ , goto 1 <b>fi</b>	$\mu_6$ $\mu_7$ $\mu_8$

Le μsequenze sono state formulate nell'ipotesi che i registri di stato e di condizione siano inizialmente azzerati. Le μistruzioni sono indicate globalmente col simbolo  $\mu_1, \mu_2, \dots, \mu_1$  senza specificare i valori booleani dei comandi  $\alpha$  che possono essere dedotti analizzando le singole operazioni da effettuare, in relazione ad una struttura dettagliata della parte operativa.

## Considerazioni sulla tabella di flusso

La rete sequenziale di controllo da realizzare ha quattro ingressi (**2 bit d'istruzione e 2 di condizioni**) e **tre stati**, per cui le variabili interne y necessarie sono solo due.

La corrispondente tabella di flusso avrebbe tre righe e sedici colonne, perché tante sono le possibili combinazioni d'ingresso.

E' opportuno notare, però, che delle **sedici colonne della tabella molte sarebbero completamente non specificate** perché sono utilizzate solo tre delle combinazioni d'istruzione e solo tre delle combinazioni delle condizioni. La possibilità che entrambi i bit del registro delle condizioni,  $B_0$  e  $B_1$  siano ad "1", è esclusa per costruzione.

## Il contenuto della ROM

INDIRIZZO						CONTENUTO		
$I_1$	$I_0$	$B_1$	$B_0$	$y_1$	$y_0$	$\alpha$	$y_1'$	$y_0'$
0	0	0	0	0	0	$\mu_1$	0	0
0	1	0	0	0	0	$\mu_2$	0	1
0	1	0	0	0	1	$\mu_3$	0	0
0	1	0	1	0	1	$\mu_4$	1	0
0	1	0	1	1	0	$\mu_5$	0	0
1	0	0	0	0	0	$\mu_6$	0	1
1	0	0	0	0	1	$\mu_7$	0	0
1	0	1	0	0	0	$\mu_8$	0	1
1	0	1	0	0	1	$\mu_9$	0	0

I  $\mu$ programmi corrispondenti alle varie istruzioni cominciano tutte alle locazioni  $I_0 I_1 0 0 0 0$ , perché si è supposto che i registri contenenti le variabili y e  $\beta$  siano sempre azzerati preliminarmente.

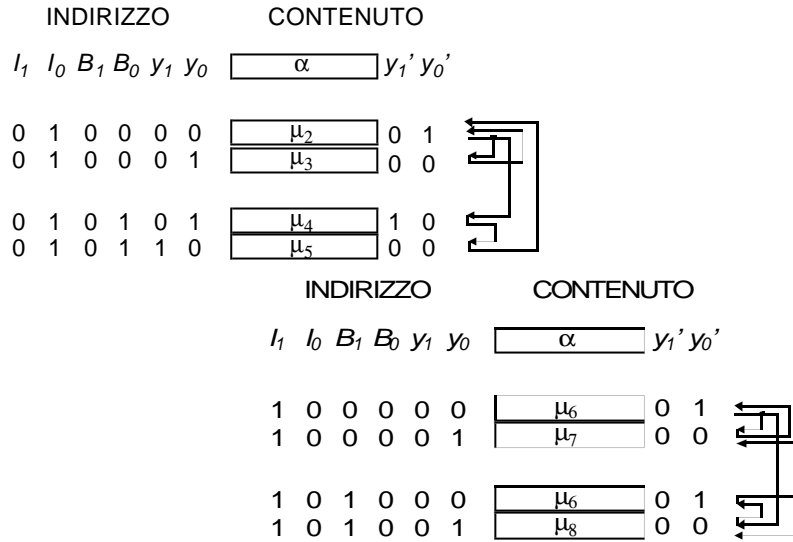
## Esecuzione delle istruzioni (1)

- L'istruzione con codice 0 0 richiede il congelamento di ogni operazione. La combinazione di comandi  $\mu_1$  blocca il sistema.
- L'istruzione 0 1 può dar luogo a due diverse evoluzioni a seconda che la prima  $\mu$ istruzione  $\mu_2$  abbia caricato il valore 0 od 1 nel registro  $B_0$ . La successiva  $\mu$ istruzione, fermo restando che il prossimo stato deve avere il codice 0 1, può essere all'indirizzo 0 1 0 0 0 1, oppure a quello 0 1 0 1 0 1 che differiscono solo per il bit  $B_0$ . Nel primo caso dopo la  $\mu$ istruzione  $\mu_2$  viene eseguita quella  $\mu_3$ , nell'altro viene eseguita quella  $\mu_4$  seguita dalla  $\mu_5$ .

## Esecuzione delle istruzioni (2)

- La struttura del  $\mu$ programma relativo all'istruzione 10 è un po' più complessa. La prima  $\mu$ istruzione  $\mu_6$  può caricare nel registro  $B1$  uno 0 oppure un 1 e ciò fa sì che nel primo caso la prossima  $\mu$ istruzione sia la  $\mu_7$  che si trova all'indirizzo 1 0 0 0 0 1. Questa  $\mu$ istruzione non esegue nessuna operazione (è identica alla  $\mu_1$ ). Se  $B1 = 1$ , invece, viene eseguita la  $\mu_8$  che si trova all'indirizzo 1 0 1 0 0 1. Alla fine dell'esecuzione, però, poiché l'operazione non altera il contenuto di  $B1$ , non si torna all'indirizzo di partenza ma si va all'indirizzo 1 0 1 0 0 0 che contiene anch'esso la  $\mu$ operazione 6 per il caricamento di  $B1$ .

### Esecuzione delle istruzioni (3)

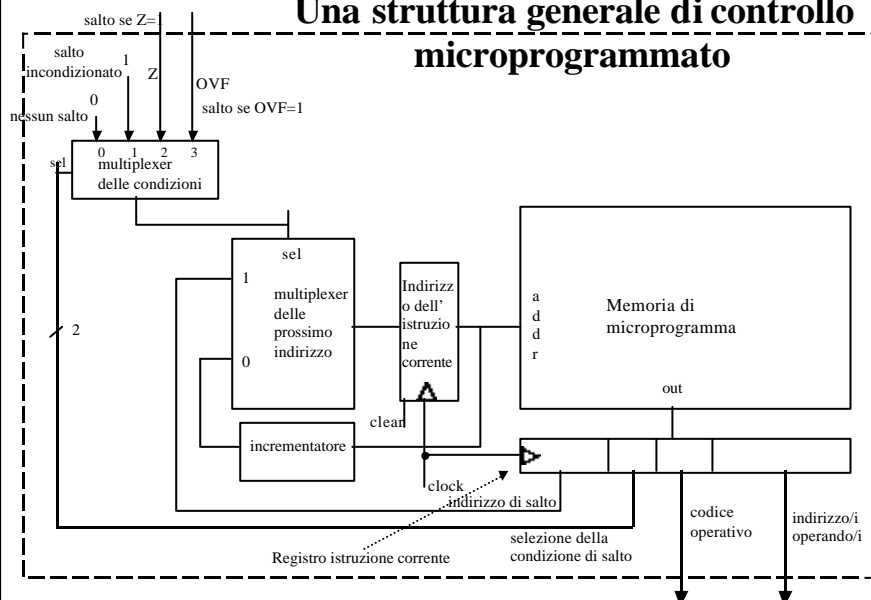


Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

55

### Una struttura generale di controllo microprogrammato

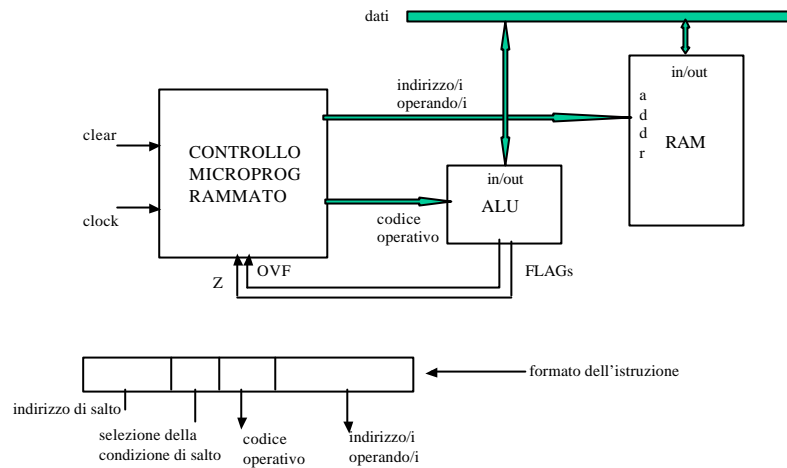


Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

56

## Il sistema microprogrammato completo



Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

57

## Considerazioni conclusive

In un sistema di elaborazione, la cui parte di controllo sia *microprogrammata*, l'esecuzione di una istruzione avviene separando il **codice operativo dell'istruzione** ( $I_1 I_0$  nel semplice esempio appena visto), detto anche "*macrooperazione*" dagli alti campi di bit previsti dal *formato* d'istruzione e trasferendolo in un registro.

Il contenuto di questo registro in "*unione*" con i bit di "*condizione*" e con quelli della **combinazione delle variabili interne** che individua il "*prossimo stato*" della rete sequenziale, viene usato come indirizzo di accesso della ROM di microprogrammazione.

Marzo 2002

Architettura degli elaboratori - Mod. B - 1. Sistemi di elaborazione

58