Macchine dedicate
e
Macchine programmabili

Corso di Calcolatori Elettronici A

2007/2008

Sito Web:http://prometeo.ing.unibs.it/quarella

Prof. G. Quarella

prof@quarella.net

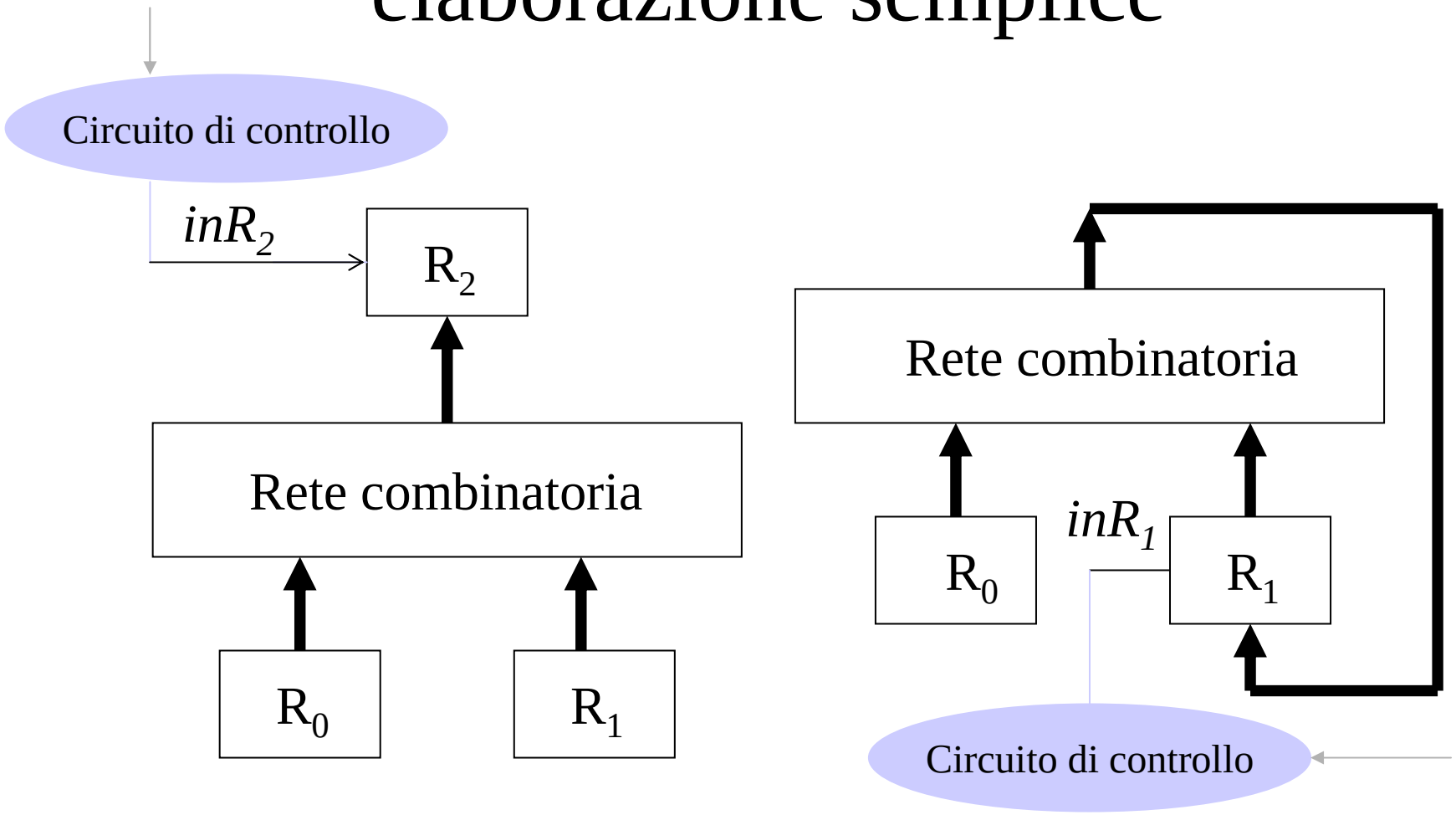
Obiettivo

- Progettare circuiti per permettano di:
  1. Trasferire l'informazione da un punto a un altro del calcolatore
  2. Elaborare l'informazione
- Le reti combinatorie e sequenziali ci permetteranno di costruire circuiti complessi per lo svolgimento delle operazioni di cui sopra

Macchine dedicate per elaborazioni logico/aritmetiche

- Cominciamo con l'analizzare ***elaborazioni semplici*** aventi le seguenti caratteristiche:
  - Agiscono su **una o due parole** e producono come risultato una **singola parola**
  - Possono essere realizzate con reti combinatorie di profondità limitata, e quindi l'intero calcolo può essere svolto in **un ciclo di clock**
- Assumiamo che le parole siano contenute in registri (***registri sorgente***) e il risultato venga memorizzato anch'esso in un registro (***registro destinazione***).

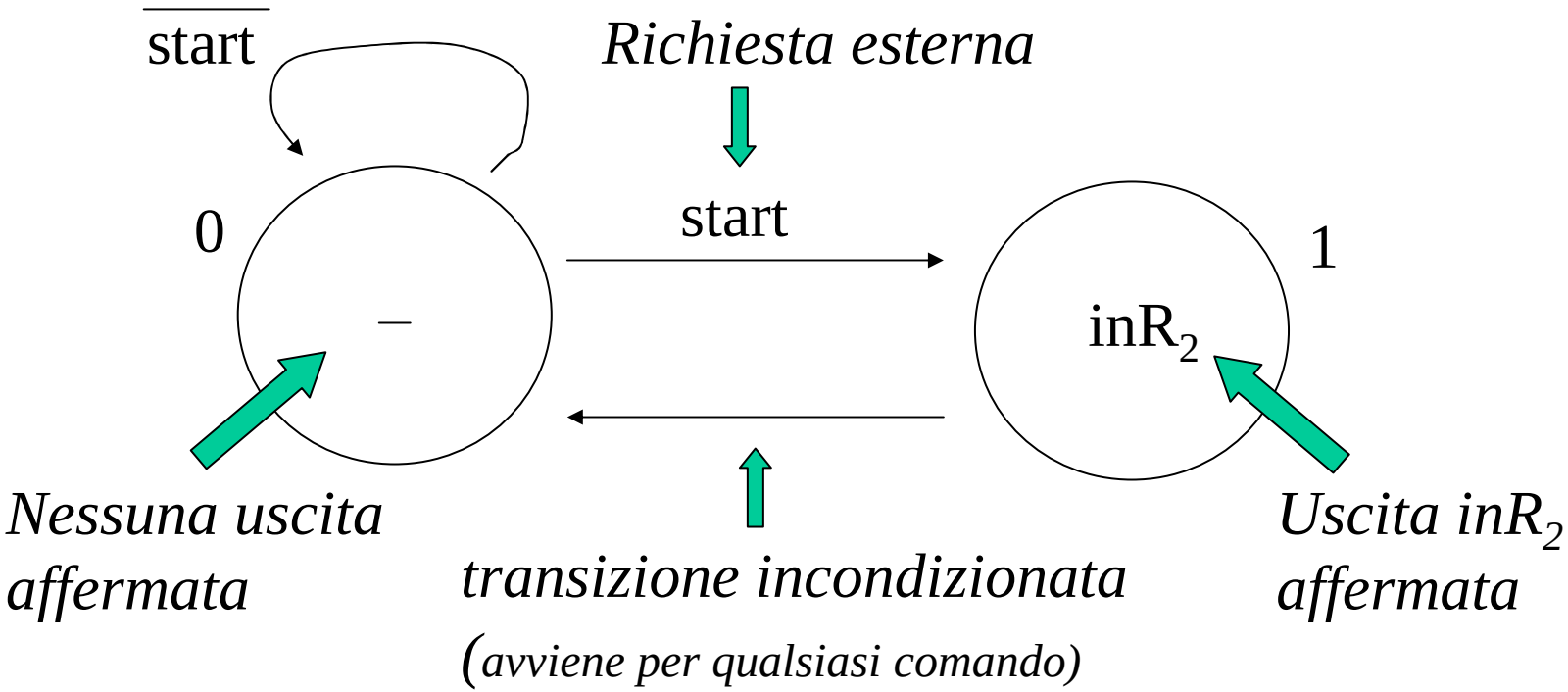
Schemi di esecuzione di una elaborazione semplice



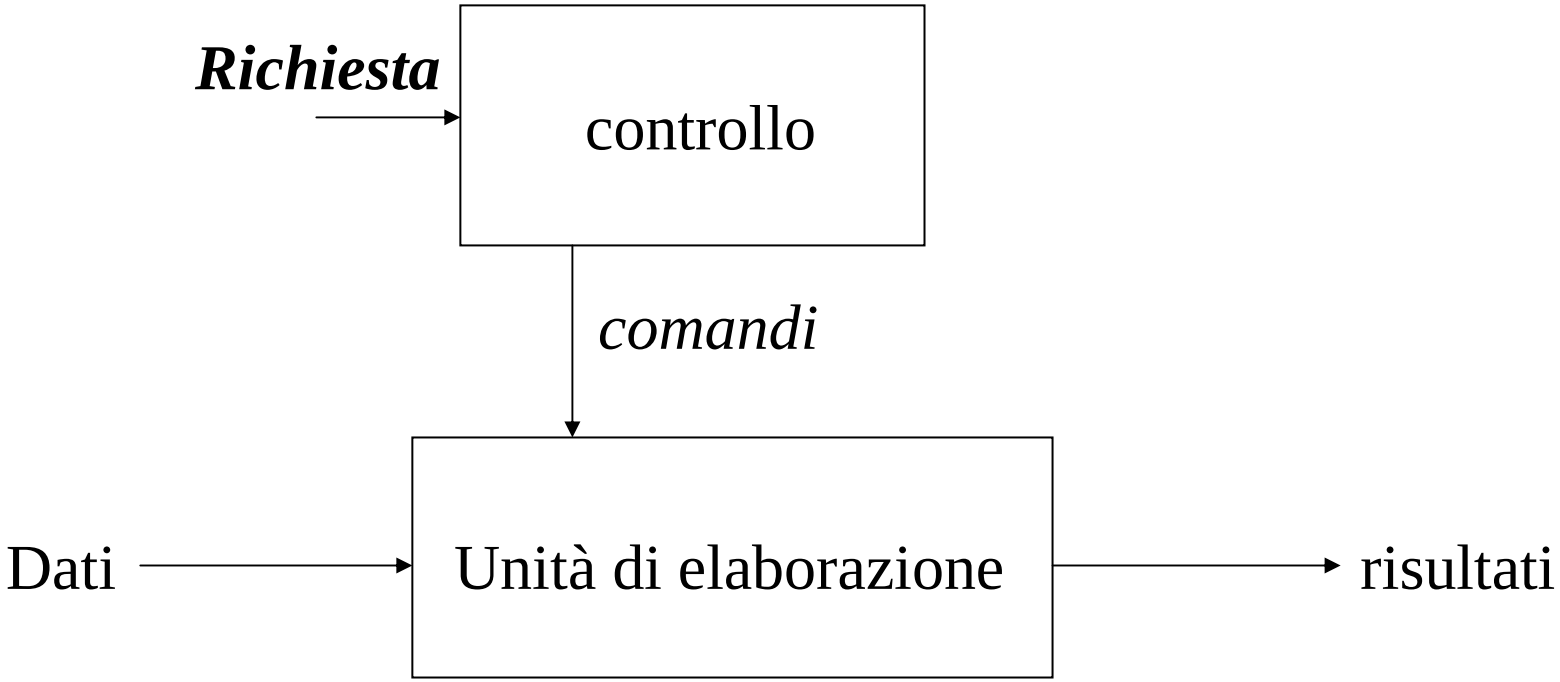
L'orologio non è rappresentato esplicitamente ma sincronizza la rete

Macchina sequenziale per una elaborazione semplice

Automa di Moore a 2 stati (0 e 1): permette di attivare il segnale di controllo inR_2 per la scrittura nel registro R_2



La Macchina completa



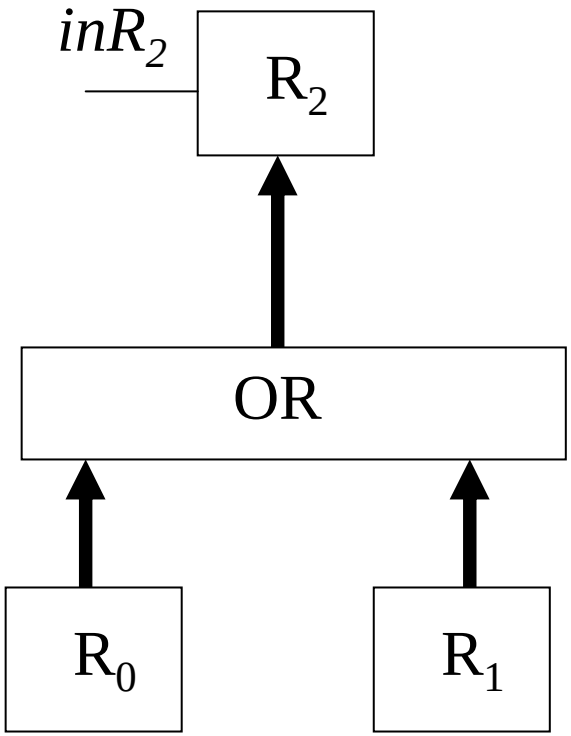
Il modello

Il modello visto è composto da 2 componenti:

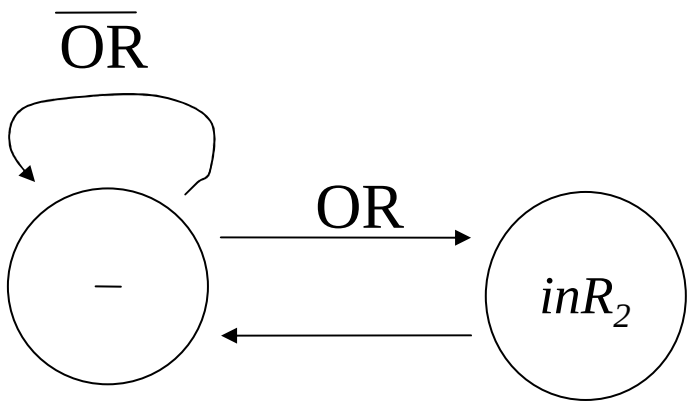
1. ***Sistema di calcolo***: lo schema con registri, interconnessioni e reti combinatorie
2. ***Sistema di controllo***: la macchina sequenziale in grado di emettere i segnali di controllo che pilotano le diverse fasi del sistema di calcolo

Esempio: OR

Sistema di calcolo

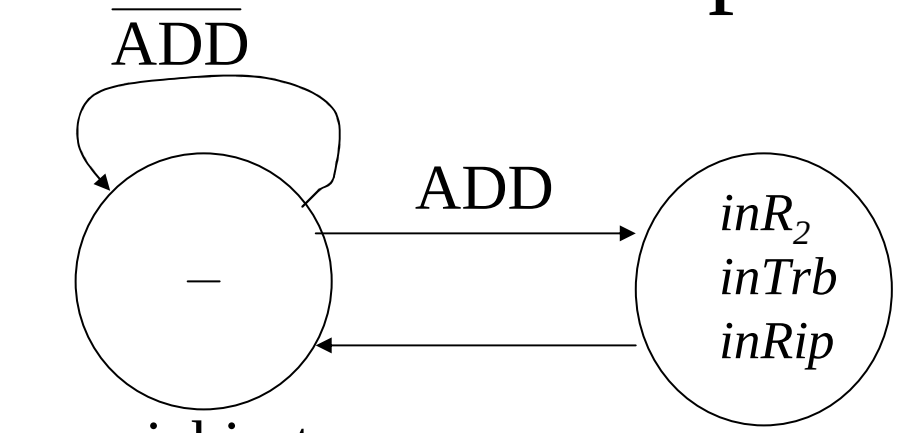


Sistema di controllo



- Il blocco OR di logica combinatoria sarà composto di *n* porte OR (se R_0 e R_1 sono a *n* bit) che ricevono in ingresso i bit dei registri R_0 e R_1

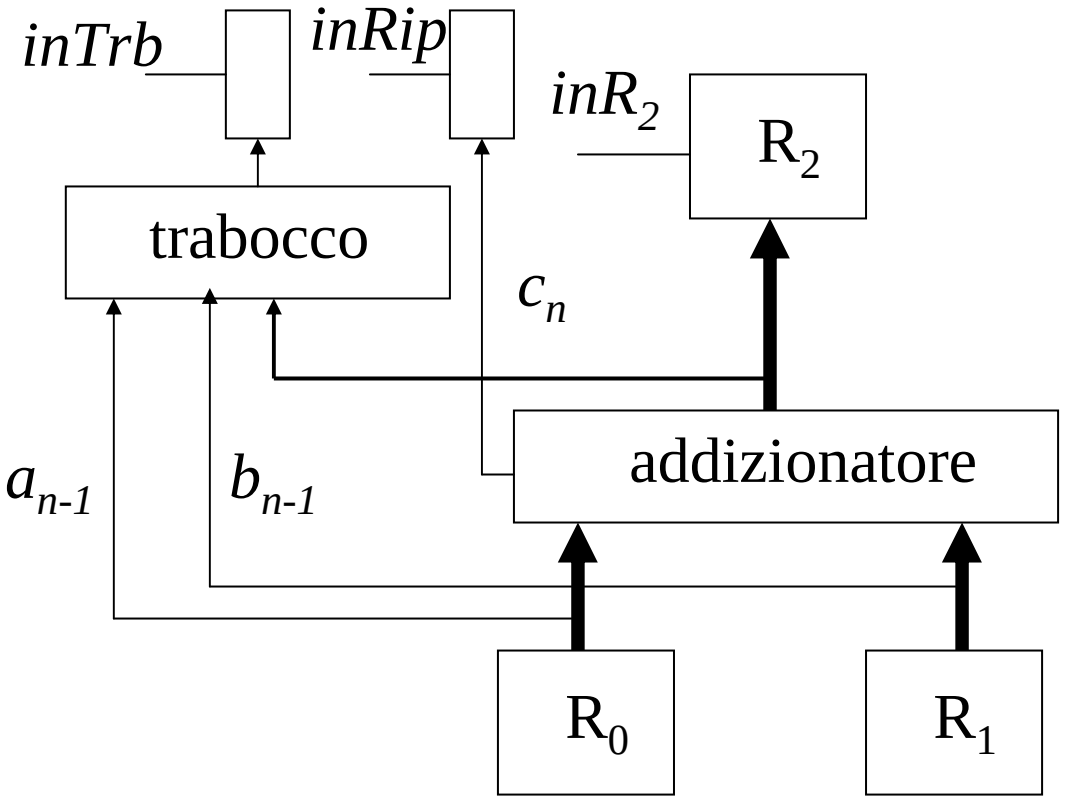
Esempio: addizione(1)



richiesta
$\{\overline{ADD}, ADD\}$
uscite
$\{-, inR_2, inTrb, inRip\}$

Sistema di controllo

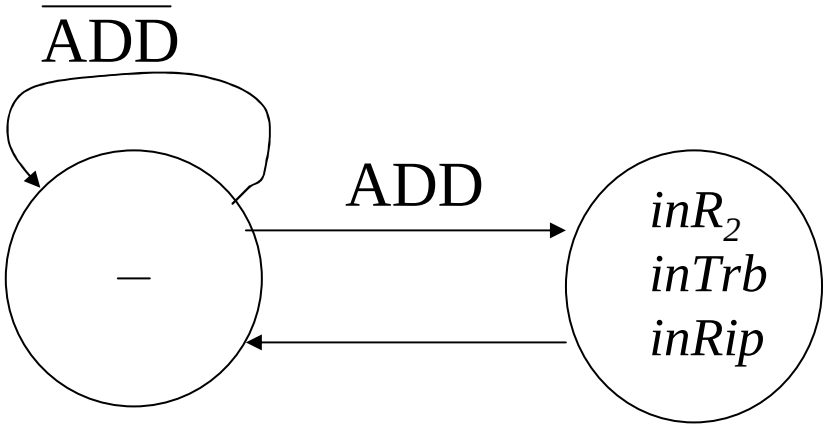
Sistema di calcolo

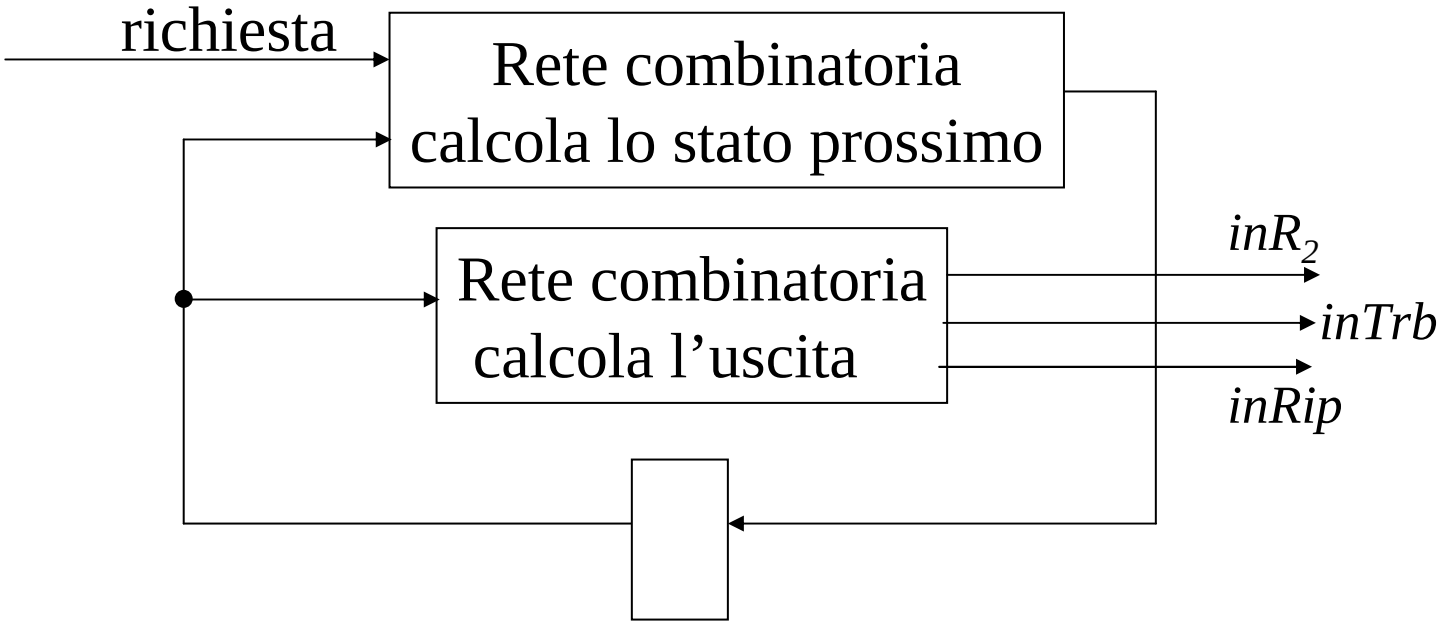


Esempio: addizione(2)

come realizzo il controllo

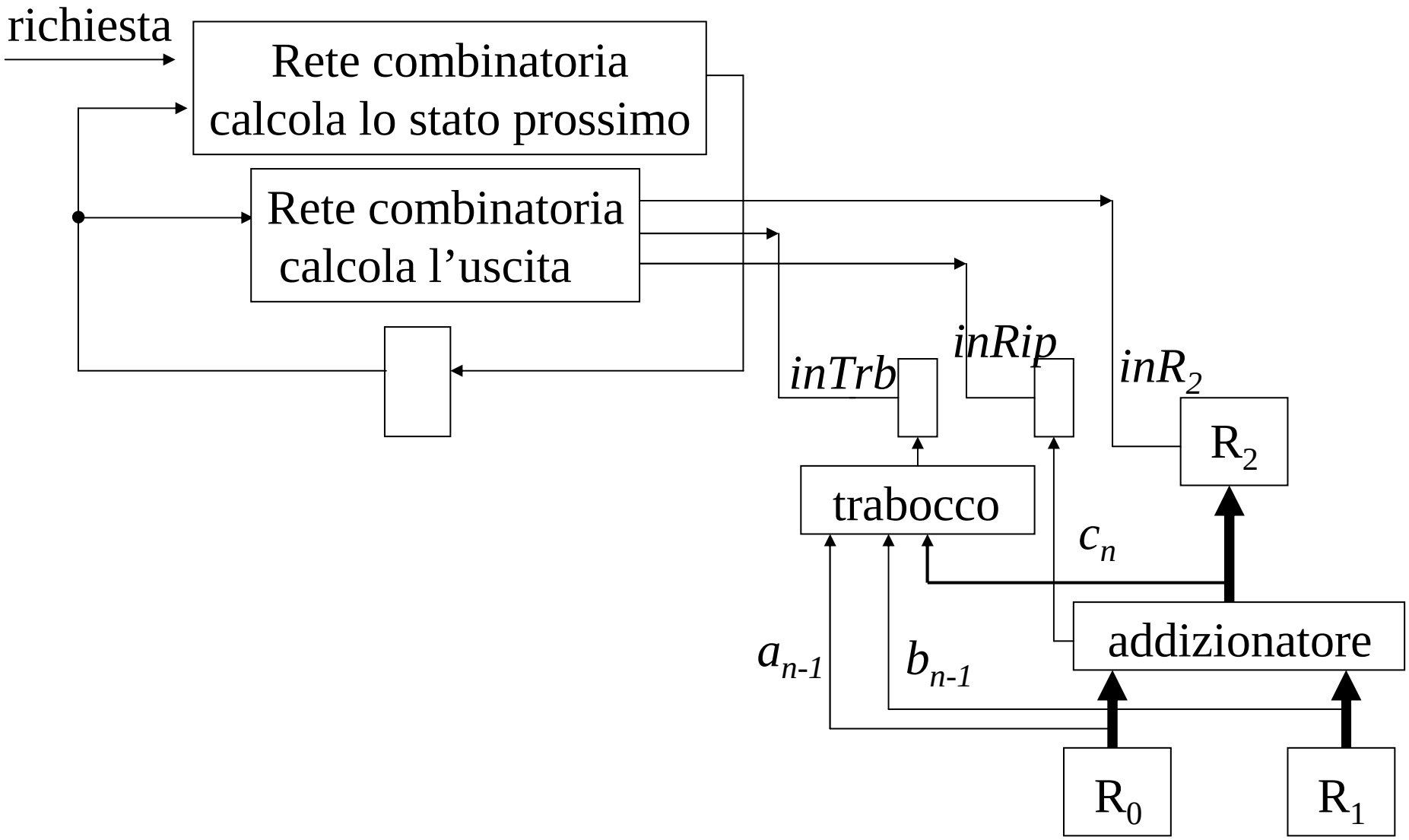
Sistema di controllo





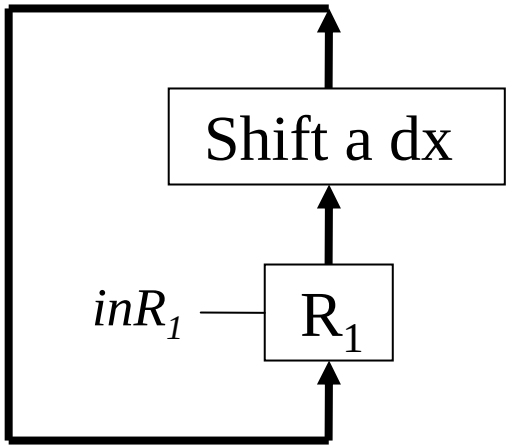
Esempio: addizione(3)

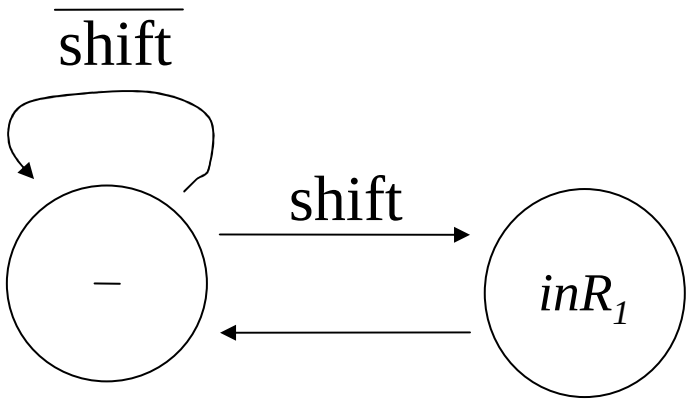
il circuito completo



Esempio: scorrimento aritmetico

- Elaborazione che agisce sugli *n* bit di un registro facendoli scorrere di una posizione verso sinistra o verso destra





- Le elaborazioni di scorrimento aritmetico consentono di realizzare operazioni di moltiplicazione o divisione per 2
- Nello scorrimento a sinistra si inserisce uno 0 nel bit meno significativo
- Nello scorrimento a destra si inserisce uno 0 nel bit più significativo (se il numero è con segno si inserisce 0 o 1 a seconda del segno - *estensione del segno* -)

Elaborazioni complesse

- Un'***elaborazione complessa*** ha le seguenti caratteristiche:
  - Agisce su un **numero finito di parole** e produce come risultato **una singola parola**
  - L'elaborazione viene suddivisa in una **sequenza di calcoli semplici**, ognuno dei quali può essere svolto in un ciclo di clock
- Supporremo quasi sempre che R_0 sia il registro destinazione e R_1, R_2, …, R_N i registri sorgente
- Prevediamo l'uso di altri registri di appoggio T_i

Sistema di calcolo e sistema di controllo per elaborazioni complesse

- Il **sistema di calcolo** include: i registri sorgente, destinazione e di appoggio; le reti di interconnessione e le reti combinatorie richieste dalle elaborazioni semplici utilizzate
- Il **sistema di controllo** è più complesso di quello visto: non più 2 soli stati, ma anche decine o centinaia
- Consideriamo sempre **sistemi sincroni**: con un cambiamento di stato ad ogni ciclo di clock

Esempio: macchina per il calcolo del valor medio di 4 interi

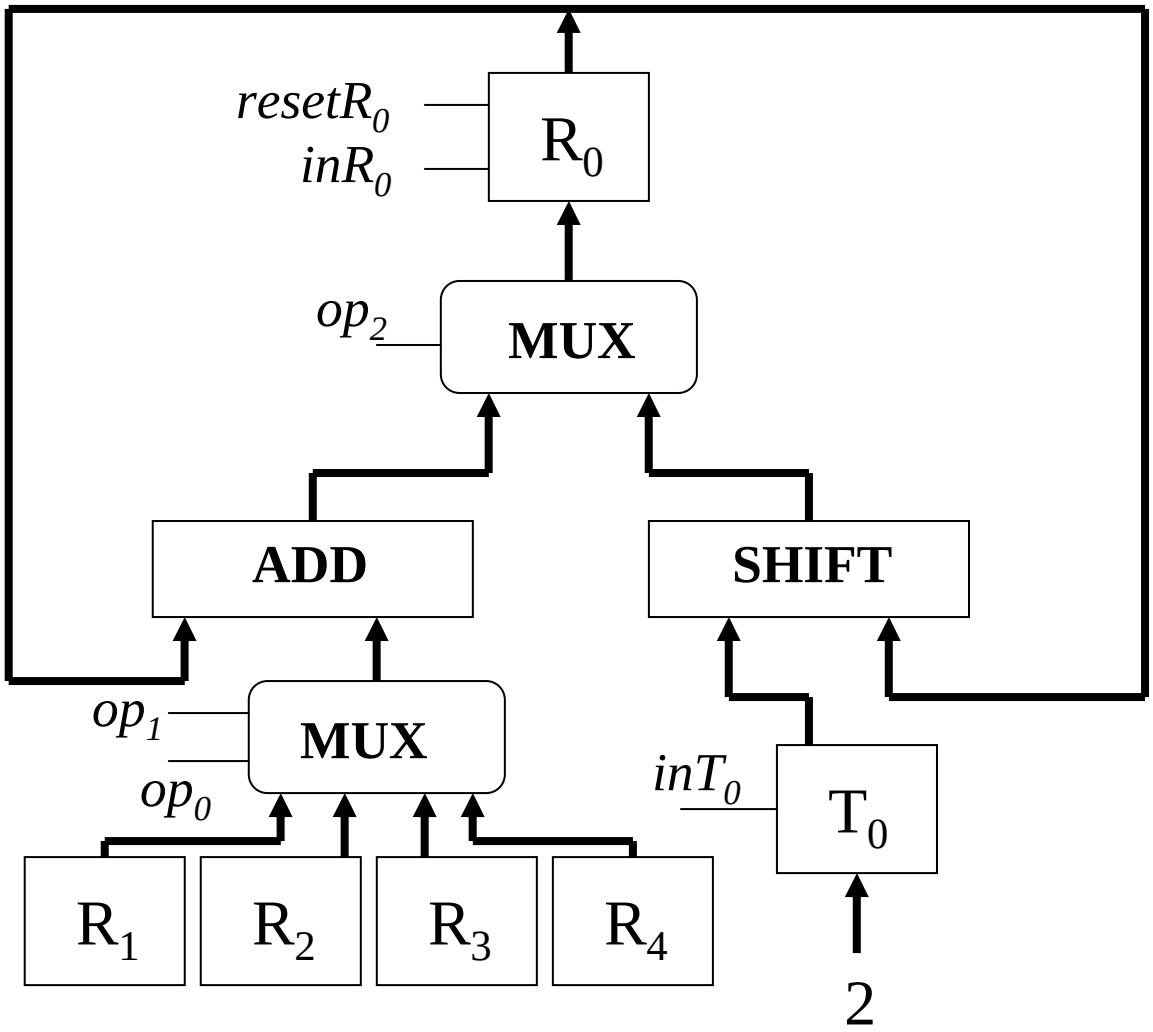
- I 4 interi sono contenuti in R_1, R_2, R_3 e R_4 e il risultato verrà scritto in R_0
- L'addizione dei 4 interi sarà realizzata con una sequenza di addizioni successive di 2 addendi
- Ricorriamo alle reti combinatorie viste in precedenza: addizionatore di 2 addendi e scorrimento aritmetico
- T_0 verrà usato per memorizzare la variabile di scorrimento pari a 2 (per la divisione per 4)

Algoritmo per il calcolo del valor medio di 4 interi

1. Reset R_0, $2 \rightarrow T_0$ (R_0 contiene 0 e T_0 contiene 2)
2. $R_0 + R_1 \rightarrow R_0$ (R_0 contiene R_1)
3. $R_0 + R_2 \rightarrow R_0$ (R_0 contiene R_1+R_2)
4. $R_0 + R_3 \rightarrow R_0$ (R_0 contiene $R_1+R_2+R_3$)
5. $R_0 + R_4 \rightarrow R_0$ (R_0 contiene $R_1+R_2+R_3+R_4$)
6. $R_0/4 \rightarrow R_0$ (R_0 contiene il valor medio)

- Questa elaborazione richiede **6 cicli di clock**
- Consideriamo prima il sistema di calcolo poi quello di controllo

Sistema di calcolo per il valor medio

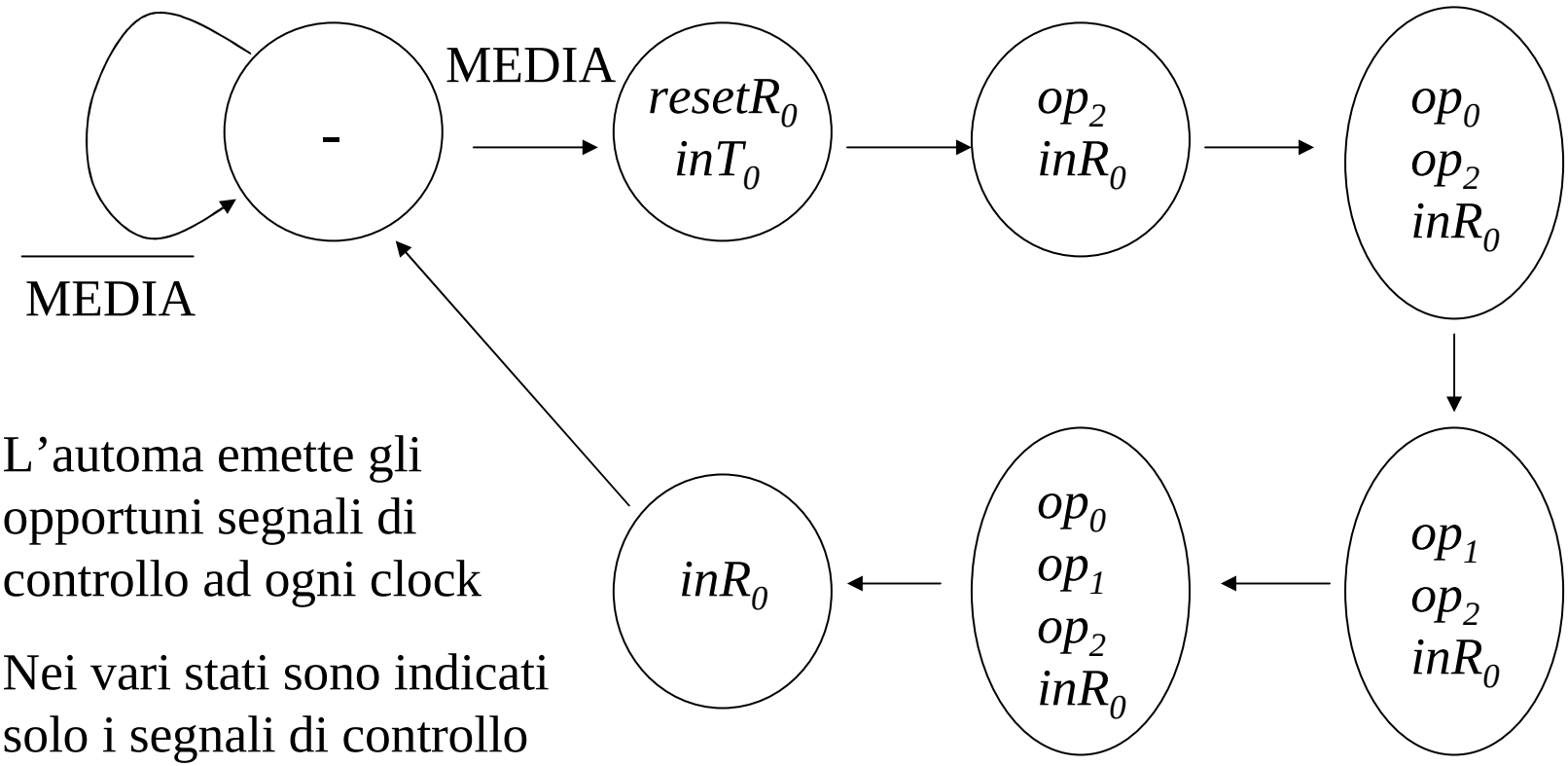


Chiarimenti sulla figura

- I registri R_i contenenti gli operandi sono collegati all'addizionatore attraverso un **multiplexer a 4 vie**, con 2 linee di controllo op_0 e op_1
- Le 4 combinazioni 00, 01, 10, 11 per op_0 e op_1 permettono al mux di selezionare rispettivamente R_1, R_2, R_3 e R_4
- Le uscite delle reti combinatorie che realizzano l'addizionatore e lo shifter sono collegate a un ulteriore **multiplexer (a 2 vie)** che, a seconda del valore della linea di controllo op_2, seleziona l'uscita di una delle due reti come ingresso per R_0 (=1 da ADD e =0 da SHIFT)
- I **segnali di controllo** sono quindi complessivamente 6:

  op_0, op_1, op_2, inT_0, inR_0 e $resetR_0$

Sistema di controllo incondizionato per il valor medio



- L'automa emette gli opportuni segnali di controllo ad ogni clock
- Nei vari stati sono indicati solo i segnali di controllo che assumono valore 1 (segnali affermati)

Esempio: macchina per il calcolo del logaritmo intero in base 2

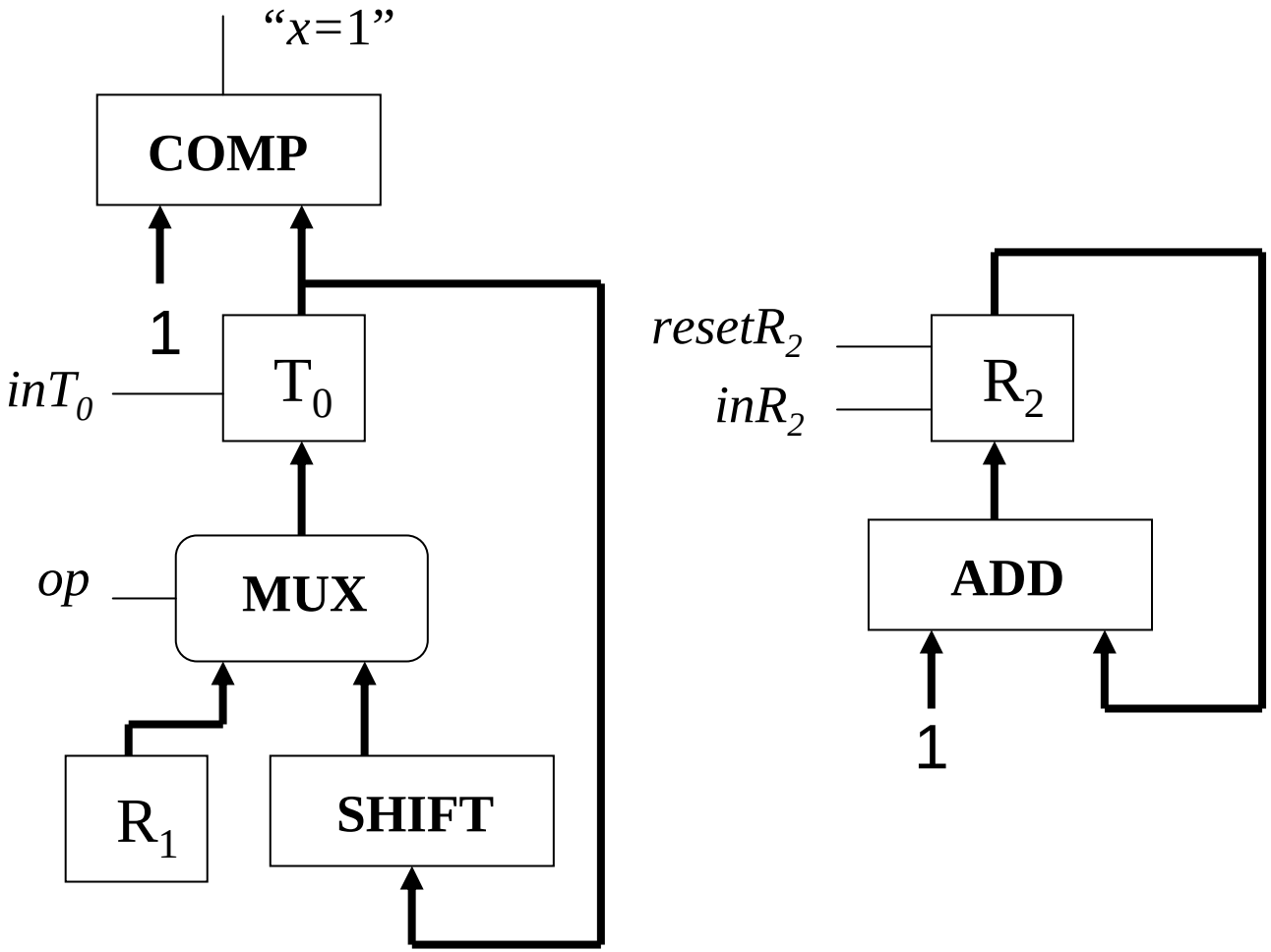
- Si vuole calcolare $\lfloor \mathbf{log_2(}\boldsymbol{x}\mathbf{)} \rfloor$: basta dividere ripetutamente per 2 il numero, finché non assume il valore 1, incrementando contemporaneamente un contatore che alla fine avrà il valore logaritmo cercato
- Il registro sorgente R_1 contiene x
- Il registro destinazione R_2 conterrà il risultato
- Il registro di appoggio T_0 contiene inizialmente x, poi viene man mano diviso per 2 (tramite scorrimento)

Algoritmo per il calcolo del logaritmo intero in base 2

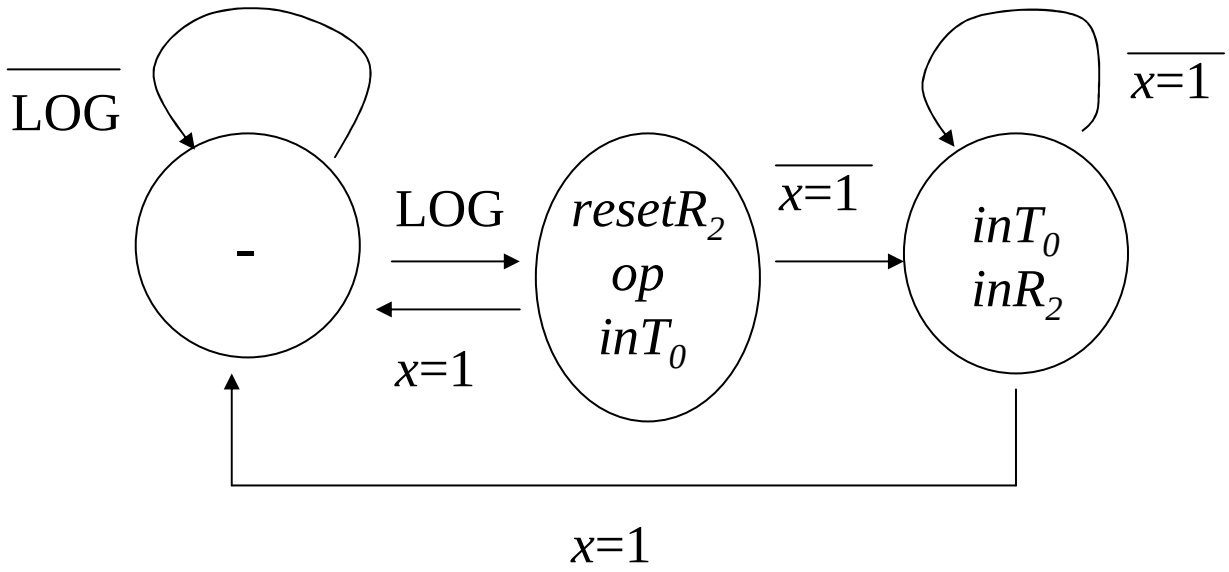
1. Reset R_2 (poni R_2 a 0); $R_1 \rightarrow T_0$ (copia R_1 in T_0)
2. Se $T_0 = 1$ termina, altrimenti vai al passo successivo
3. $T_0/2 \rightarrow T_0$ (dividi T_0 per 2); $R_2 + 1 \rightarrow R_2$ (incrementa il contatore)
4. Vai al passo 2

Il numero dei cicli di clock richiesti per questa elaborazione è variabile

Sistema di calcolo per il logaritmo



Sistema di controllo condizionato per il logaritmo



- Viene usato un segnale "x=1" proveniente dall'uscita del comparatore che vale 1 se il contenuto di T_0 è 1, e 0 se è maggiore di 1.
- A seconda del valore di "x=1", l'automa torna nello stato iniziale oppure passa allo stato in cui avviene la divisione di T_0 per 2 e l'incremento del contatore.

Moltiplicazione

moltiplicando 1101 $\times$ $x=1101_2=13_{10}$

moltiplicatore 1011 = $y=1011_2=11_{10}$

1101	+	shift a sx di 0 pos. = molt..ando per 2^0
1101	+	shift a sx di 1 pos. = molt..ando per 2^1
0000	+	
1101	+	shift a sx di 3 pos. = molt.ando. per 2^3
10001111		

$$z = 1101_2 \times 2^0 + 1101_2 \times 2^1 + 1101_2 \times 2^3 = 13+26+104 = 143_{10}$$

Algoritmo per la moltiplicazione

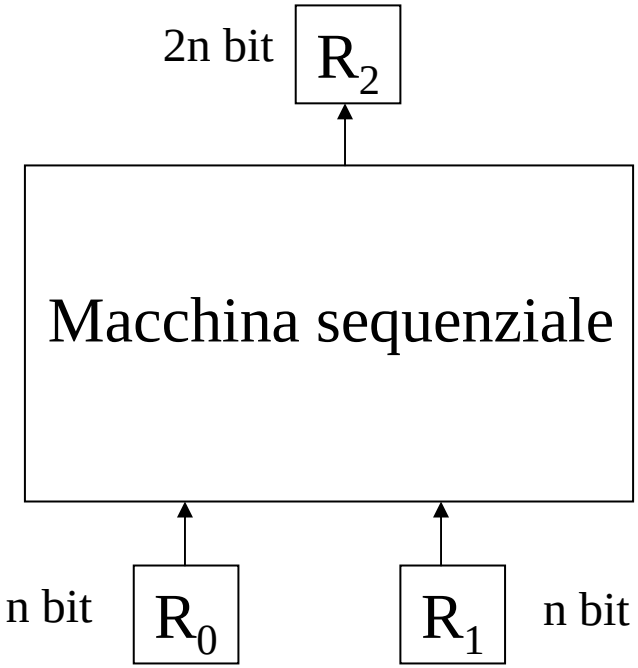
Assumiamo:

moltiplicando in $\mathbf{R_0}$ (n bit),
moltiplicatore in $\mathbf{R_1}$ (n bit),
prodotto in $\mathbf{R_2}$ ($2n$ bit)

1. Reset T_2; $R_0 \rightarrow T_0$; $R_1 \rightarrow T_1$; $j \leftarrow 1$
2. Se $T_1[0]=1$ vai al passo 3, altrimenti vai al passo 4
3. $T_0+T_2 \rightarrow T_2$
4. Shift_dx(T_1)$\rightarrow T_1$; Shift_sx(T_0) $\rightarrow T_0$; $j \leftarrow j+1$
5. Se $j = n$ vai a 6, altrimenti vai a 2
6. $T_2 \rightarrow R_2$
7. Halt

Algoritmo per la moltiplicazione

Realizzabile con un programma o <u>hardware</u>

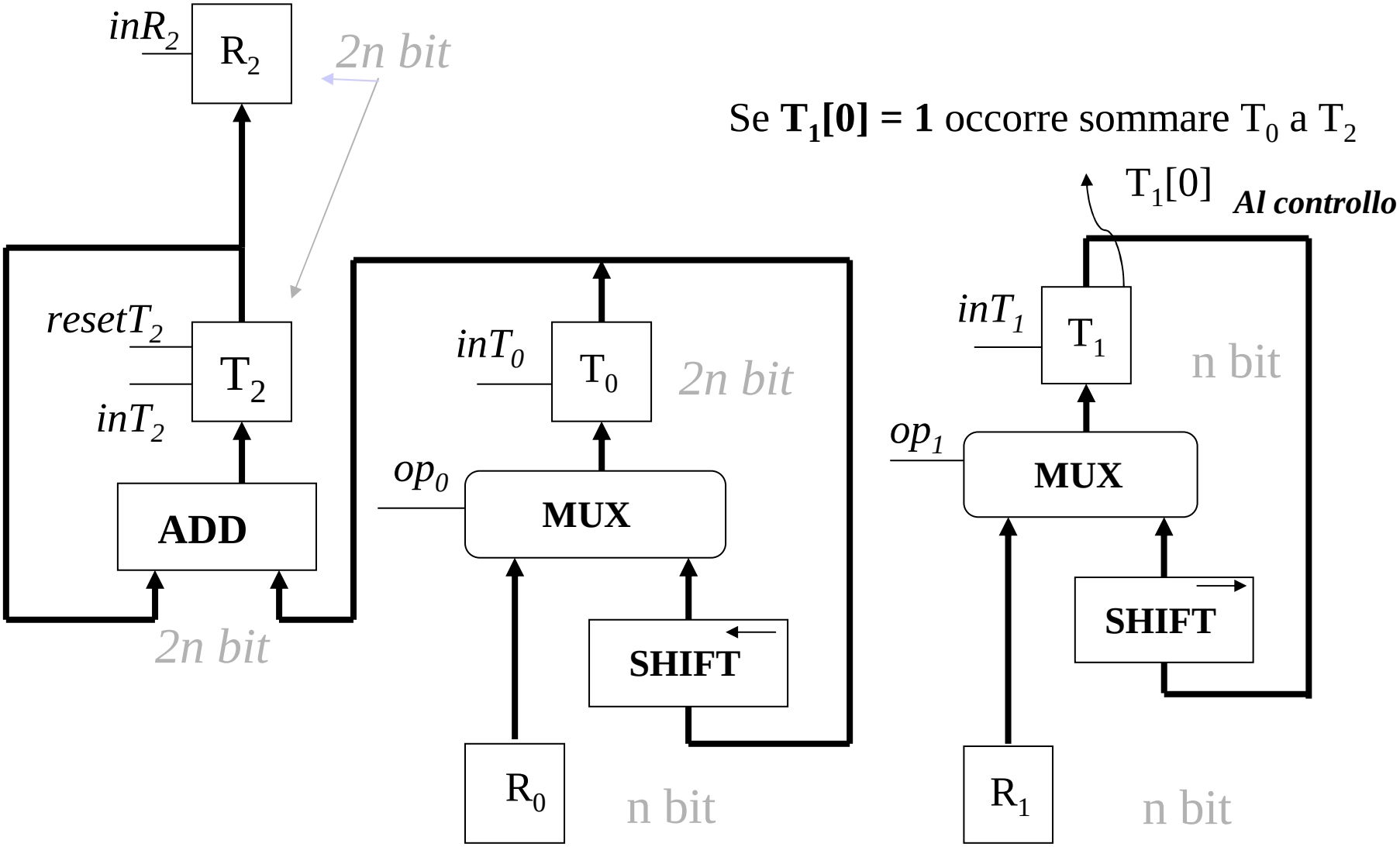


I calcolatori presentano molte organizzazioni diverse.

Oggi, in genere si hanno due registri di uscita, uno che memorizza la parte alta ed uno la parte bassa del risultato.

MIPS ha due registri e due Istruzioni.

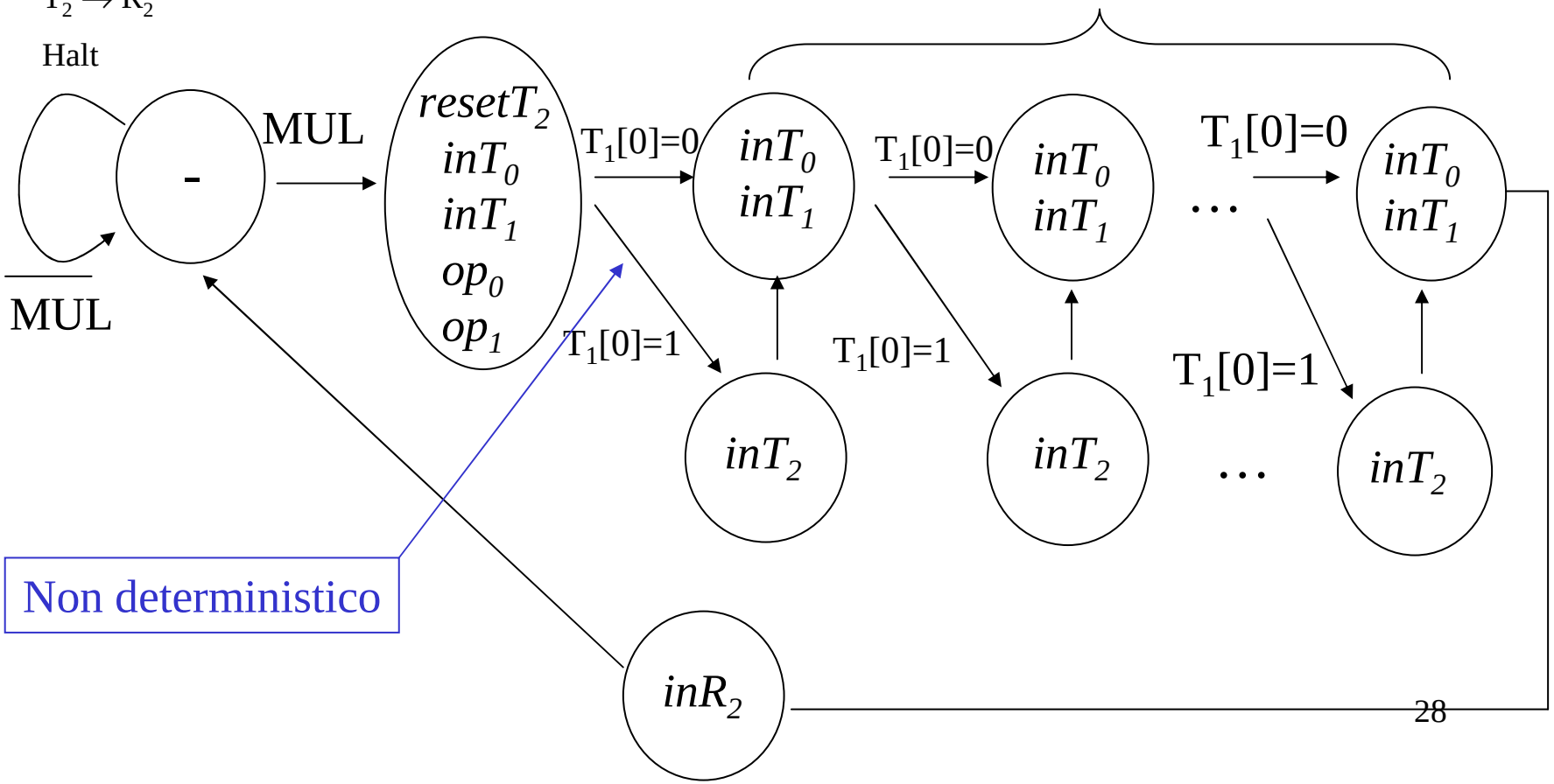
Sistema di calcolo per la moltiplicazione



Sistema di controllo per la moltiplicazione

1. Reset T_2; $R_0 \rightarrow T_0$; $R_1 \rightarrow T_1$; $j \leftarrow 1$
2. Se $T_1[0]=1$ vai al passo 3 altrimenti vai al passo 4
3. $T_0+T_2 \rightarrow T_2$
4. $T_1/2 \rightarrow T_1$; $T_0*2 \rightarrow T_0$; $j \leftarrow j+1$
5. Se j = *n* vai a 6 altrimenti vai a 2
6. $T_2 \rightarrow R_2$
7. Halt

n coppie di stati che dipendono dal controllo sul bit meno significativo di T_1



Il problema del segno

	moltiplicando	moltiplicatore	
→	>0	>0	>0
	>0	0<	0<
→	0<	>0	0<
	0<	0<	>0

Divisione

Dividendo: $110011_2 = 51_{10}$

Divisore: $101_2 = 5_{10}$

```
  1 1 0 0 1 1        qi
- 1 0 1              1
-------------
    0 1 0
-   0 0 0            0
-------------
      1 0 1
-     1 0 1          1
-------------
          0 0 1
-         0 0 0      0
-------------
r   =     0 0 1
```

Quoziente = $1010_2 = 10_{10}$

Resto = $001_2 = 1_{10}$

Divisione con modifica

Dividendo: $110011_2 = 51_{10}$
Dividendo di $2n = 6$ bit

Divisore: $101_2 = 5_{10}$
Divisore di $n = 3$ bit

1 1 0 0 1 1	
- 1 0 1 0 0 0	Divisore * 2^3 (sh a sx di 3 posizioni)
0 0 1 0 1 1	$q_3 = 1$
(- 0 1 0 1 0 0)	Divisore * 2^2 (sh a sx di 2 posizioni)
0 0 1 0 1 1	$q_2 = 0$
- 0 0 1 0 1 0	Divisore * 2^1 (sh a sx di 1 posizioni)
0 0 0 0 0 1	$q_1 = 1$
(- 0 0 0 1 0 1)	Divisore * 2^0 (nessuno shift)
r = 0 0 0 0 0 1	$q_0 = 0$

$$\mathbf{q = 1010_2}$$

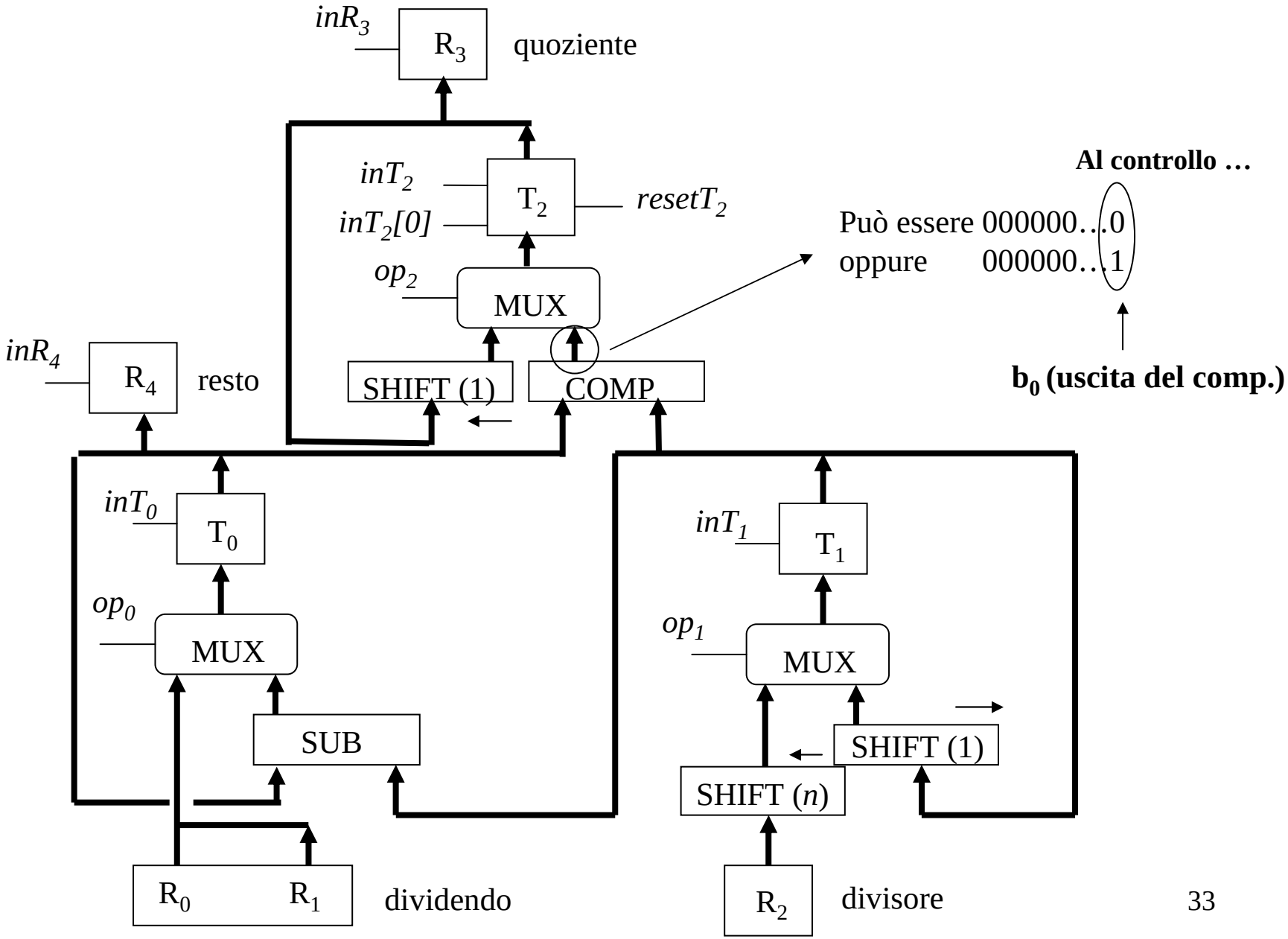
Algoritmo per la divisione

I 2*n* bit del **dividendo** sono in $\mathbf{R_0}$ **e** $\mathbf{R_1}$; il **divisore** è in $\mathbf{R_2}$. Alla fine $\mathbf{R_3}$ conterrà il **quoziente** e $\mathbf{R_4}$ conterrà il **resto**. Sono usati i registri di appoggio $\mathbf{T_0}$ che contiene inizialmente i 2*n* bit del dividendo, $\mathbf{T_1}$ che contiene inizialmente il divisore moltiplicato per 2^n e $\mathbf{T_2}$ (di 2*n* bit) per memorizzare man mano il quoziente

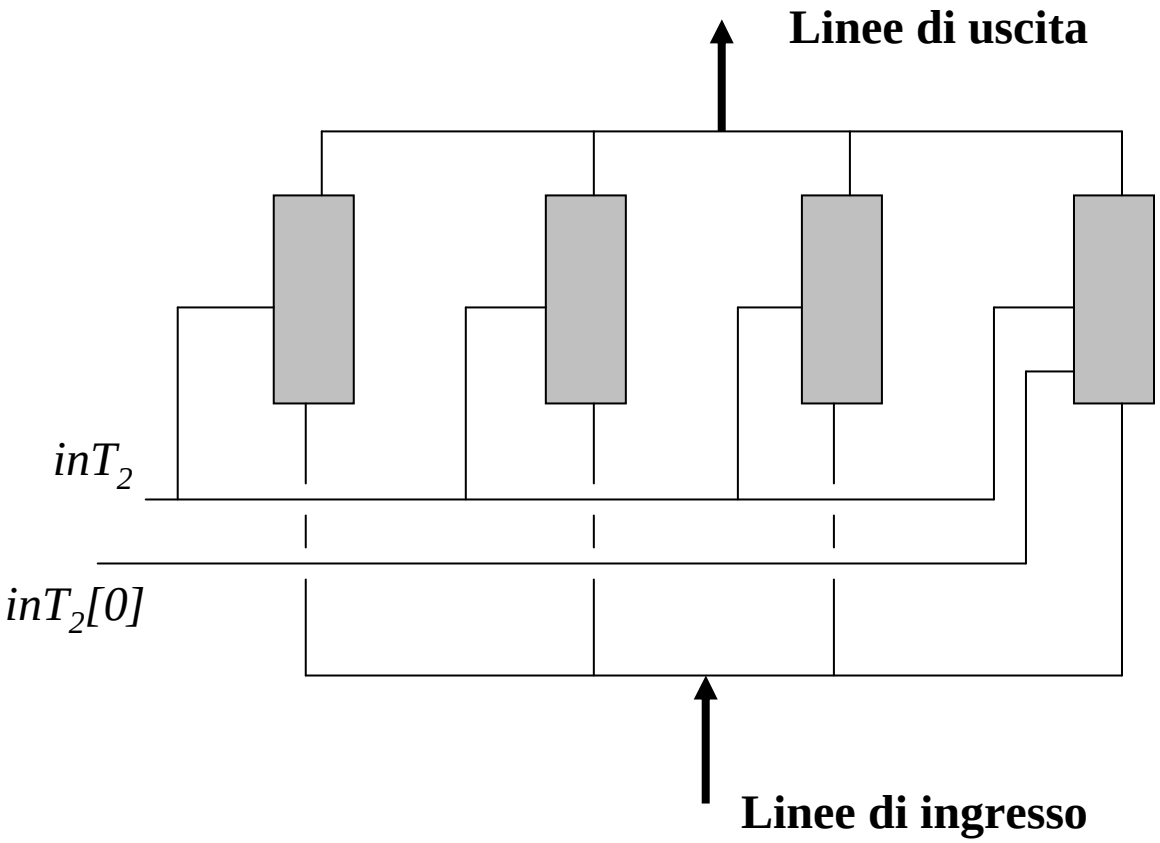
1. $R_0 \rightarrow T_0$; $R_1 \rightarrow T_0$; $R_2 * 2^n \rightarrow T_1$; Reset T_2
2. $(T_0 >= T_1) \rightarrow T_2[0]$ (poni in $T_2[0]$ il bit meno significativo del risultato del confronto fra T_0 e T_1 … cioè 1 se $T_0 >= T_1$ e 0 altrimenti)
3. Se $T_0 >= T_1$, allora $T_0 - T_1 \rightarrow T_0$
4. Shift a dx di T_1 (divisore); Shift a sx di T_2 (quoziente)
5. Se sono state fatte *n* iterazioni, vai al passo 6 altrimenti vai al passo 2
6. $(T_0 >= T_1) \rightarrow T_2[0]$ (ultimo bit meno significativo del quoziente)
7. Se $T_0 >= T_1$, allora $T_0 - T_1 \rightarrow T_0$ (calcolo del resto)
8. $T_2 \rightarrow R_3$; $T_0 \rightarrow R_4$

ciclo

Sistema di calcolo per la divisione

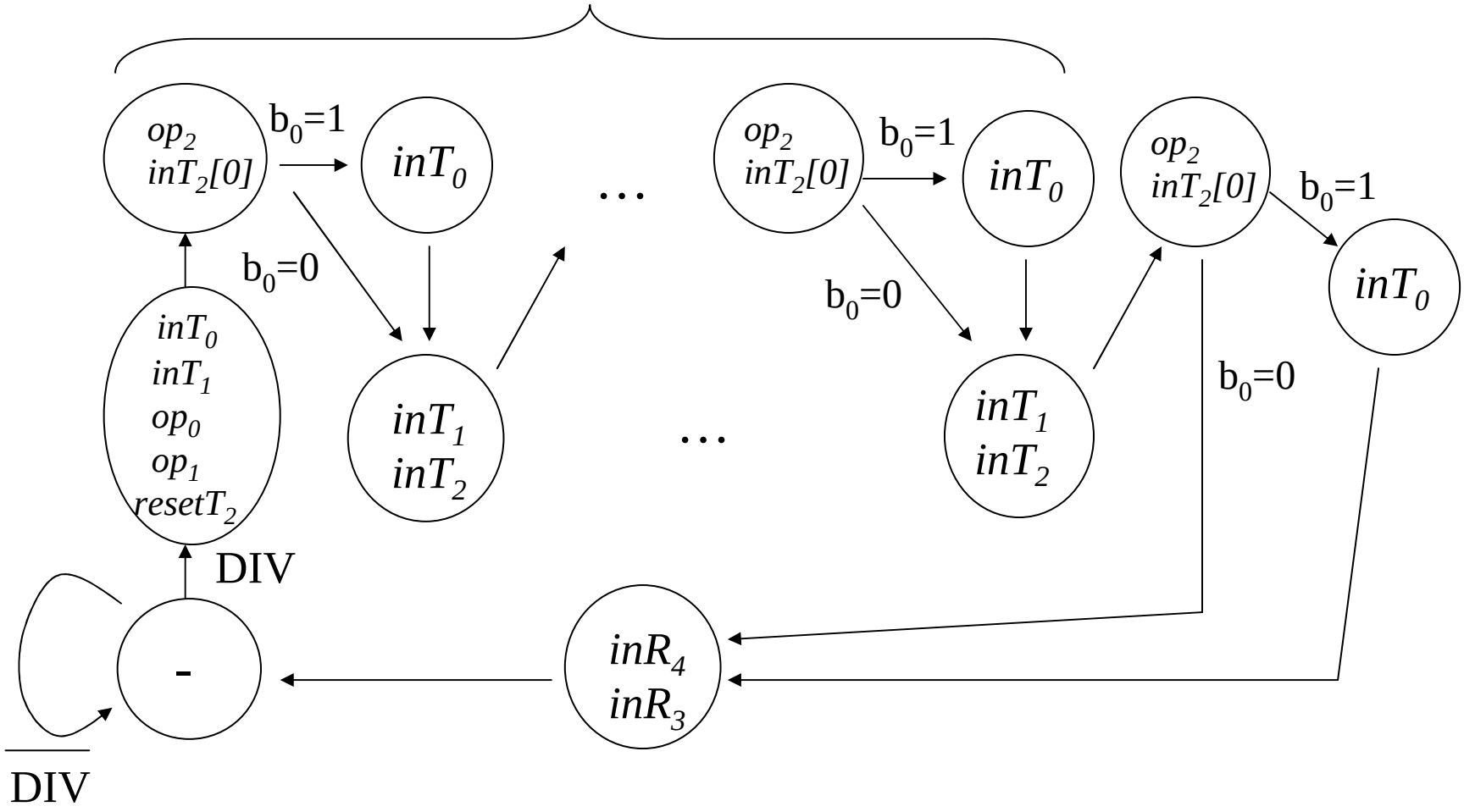


Schema per la scrittura del registro T_2



Sistema di controllo per la divisione

n terne di stati che dipendono dal confronto fra T_0 e T_1
(se $T_0 >= T_1$ allora $b_0 = 1$ altrimenti $b_0 = 0$)

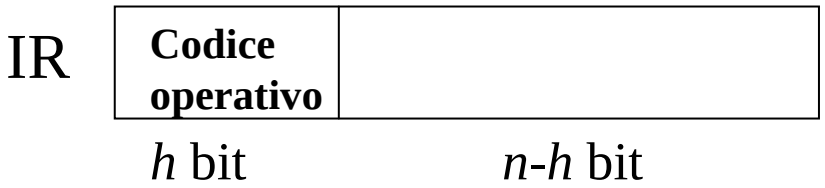


Macchine programmabili

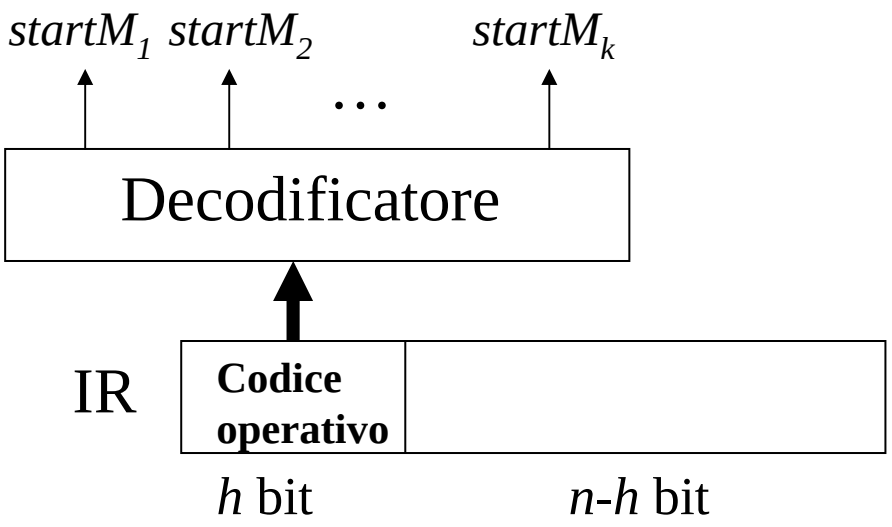
- Quelle che abbiamo visto finora sono macchine “dedicate” in quanto il loro sistema di controllo è progettato per svolgere una **specifica elaborazione.**
- Una data macchina non è in grado di svolgere nessun’altra elaborazione, se non quella per cui è stata prevista.
- Una **macchina programmabile** è invece una macchina in cui le operazioni da svolgere sono descritte da informazioni **registrate in una memoria** .
- Vedremo con un esempio la specifica di un sistema di controllo in grado di **eseguire una generica operazione contenuta in un registro** (registro istruzione).
- Una macchina programmabile è una macchina in grado di eseguire in modo automatico **programmi** (sequenze di istruzioni) di lunghezza arbitraria.

Codici operativi e macchine dedicate

- Supponiamo di disporre di alcune macchine sequenziali dedicate M_1, M_2…, M_k, ognuna delle quali in grado di svolgere un compito specifico (ad es. l'addizione, la moltiplicazione, lo scorrimento, …)
- Ogni macchina può essere identificata da un gruppo di bit, che chiameremo ***codice operativo***
- Utilizziamo un registro **IR** (***registro istruzione***) di *n* bit per registrare il codice operativo della macchina che si intende attivare



Decodifica del codice operativo



I **segnali di controllo** che servono ad attivare una delle macchine sequenziali incluse nel sistema di calcolo si ottengono come uscite di un decodificatore ad *h* ingressi

Esempio con 4 macchine

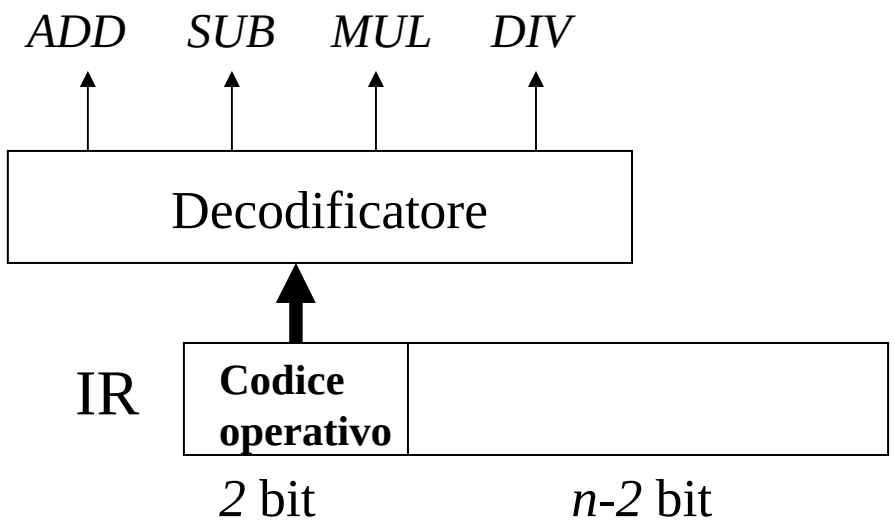
Codice operativo per le 4 macchine

00 → addizione

01 → sottrazione

10 → moltiplicazione

11 → divisione



Codice operativo		*DIV*	*MUL*	*SUB*	*ADD*
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

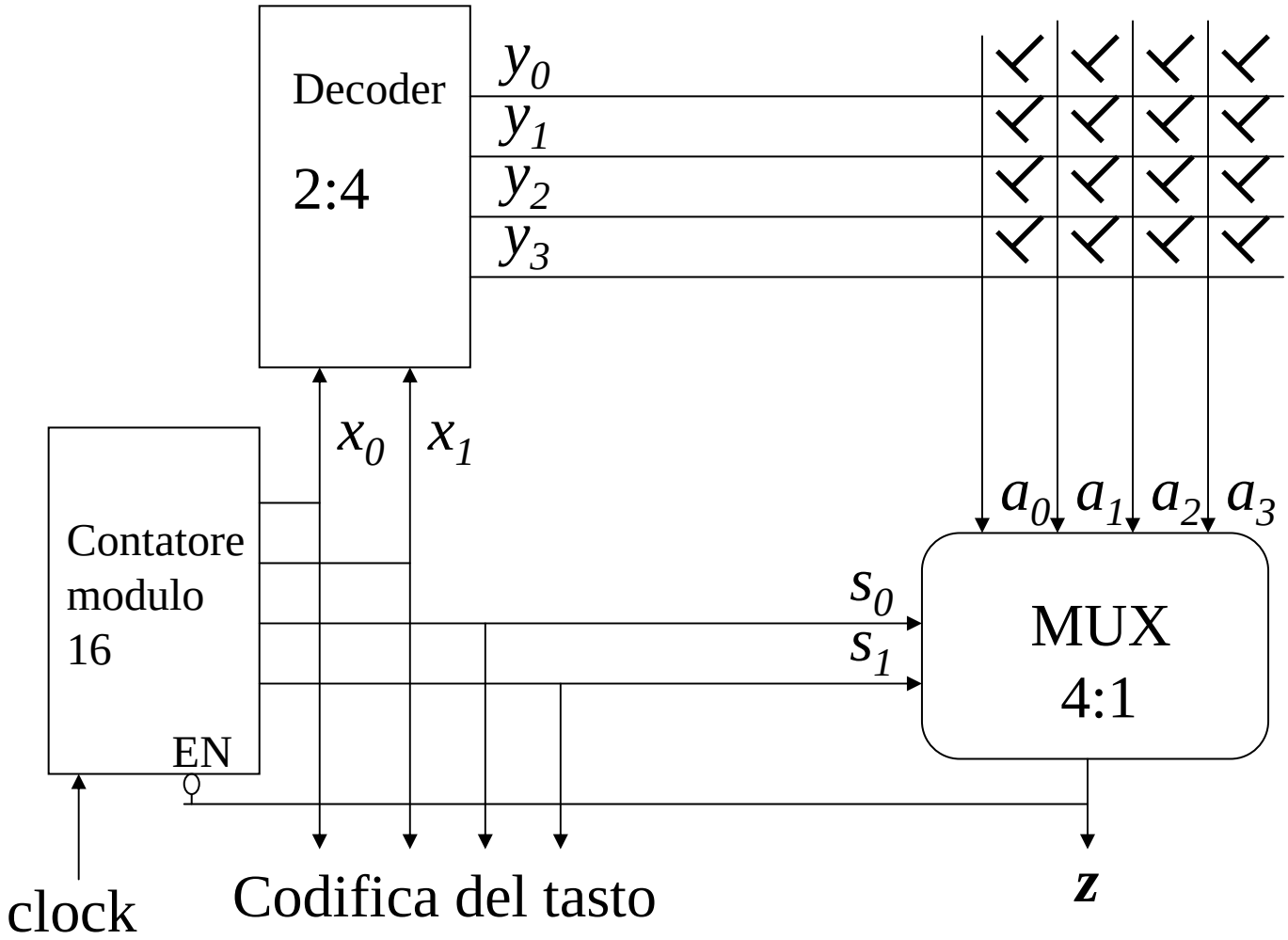
Esempio: Tastiera

Supponiamo di disporre di una tastiera con 16 tasti

1. Vogliamo progettare il circuito che rileva la pressione di un tasto e trasmette il codice corrispondente. Costruiamo il circuito a partire da 3 blocchi funzionali
   - Un contatore
   - Un decodificatore
   - Un multiplexer

2. Vediamo poi come è fatto il sistema di controllo e come opera in base alla codifica del tasto premuto che viene memorizzata in un registro istruzione

Implementazione della tastiera



Chiarimenti sulla figura

- Le 4 uscite del contatore (4 bit codificano 16 stati) sono usate per scorrere la matrice 4x4 dei tasti:
  - I primi 2 bit costituiscono l'ingresso del decodificatore da cui uscirà una combinazione di 4 bit dove un solo bit è posto a 1 (tale bit identificherà la **riga** della matrice che si sta esaminando)
  - Gli altri 2 bit sono invece gli ingressi di selezione del multiplexer e codificano la **colonna** da esaminare
- Se il tasto in corrispondenza della coppia (riga, colonna) esaminata è stato premuto, allora il multiplexer restituisce ***z*=1**. Contemporaneamente, la **codifica del tasto** (che corrisponde allo stato del contatore) viene **salvata in un registro** per una successiva conversione.
- Quando $z = 1$ il conteggio viene fermato finché il tasto non viene rilasciato.

Una tastiera con 16 operazioni

Codifiche dei tasti date da $x_0 x_1 s_0 s_1$

+ → 0000

- → 0001

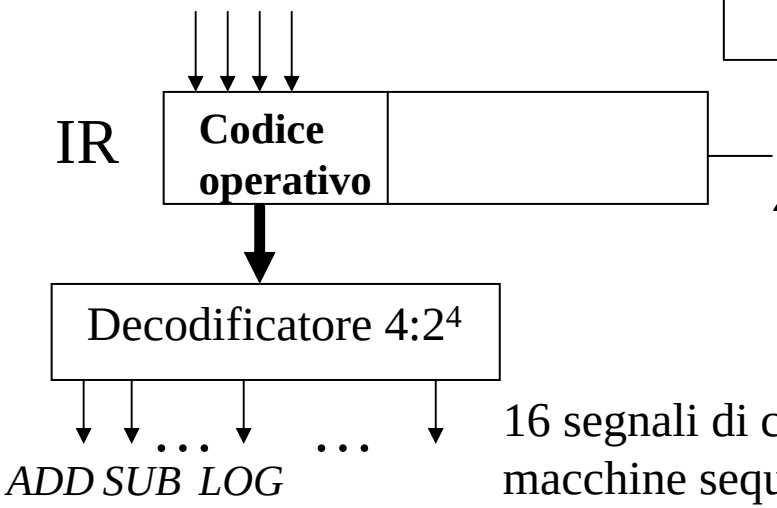
… …

log → 0100

…. …

00	+	-	*	/
01	log	ave	exp	x^2
10	x^3	x^y	x!	1/x
11	$\sqrt{x}$	sin	cos	tan
	00	01	10	11

Codifica del tasto

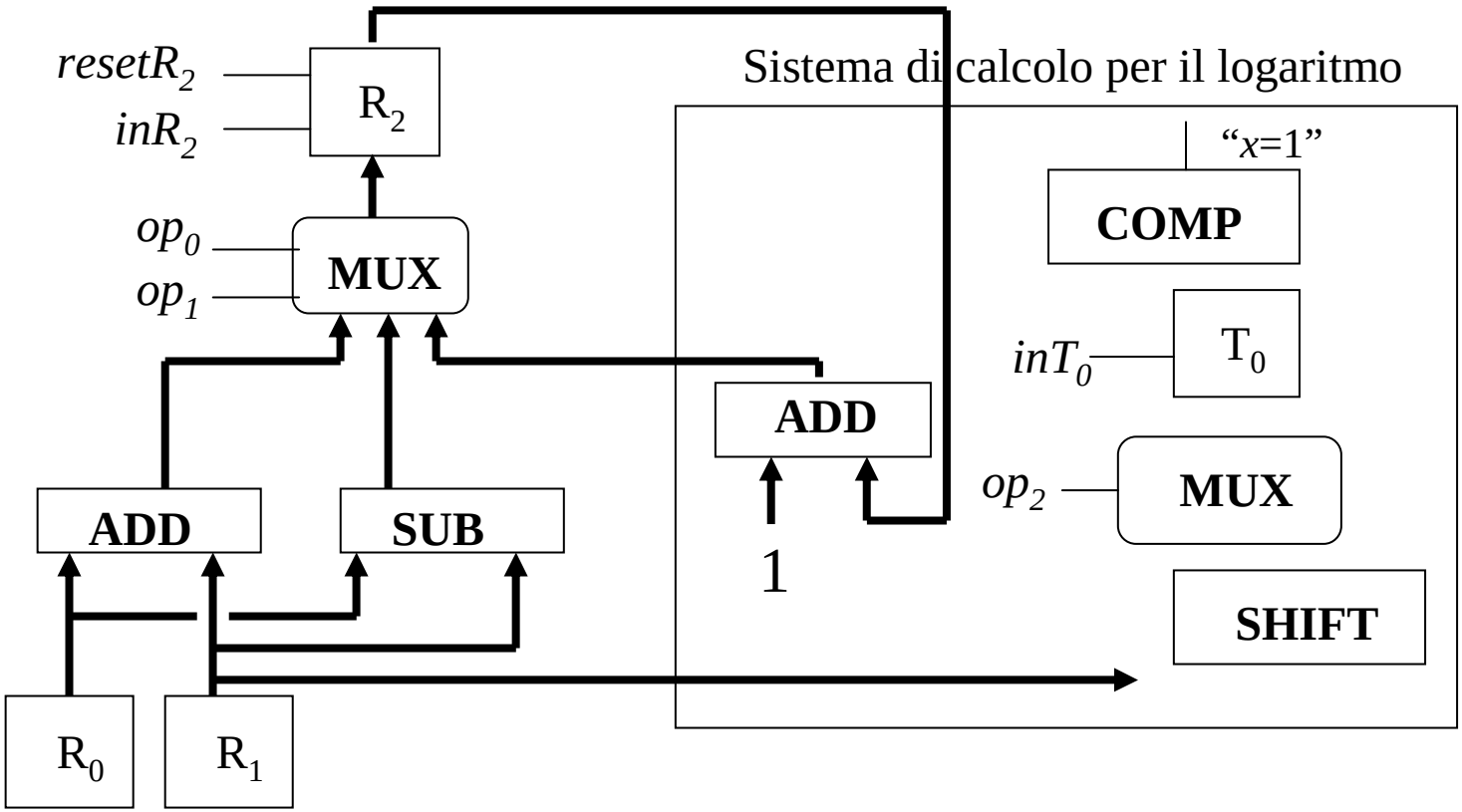


z (cioè z potrebbe essere il segnale di selezione del registro IR, che abilita la scrittura di IR e fa partire tutto il processo)

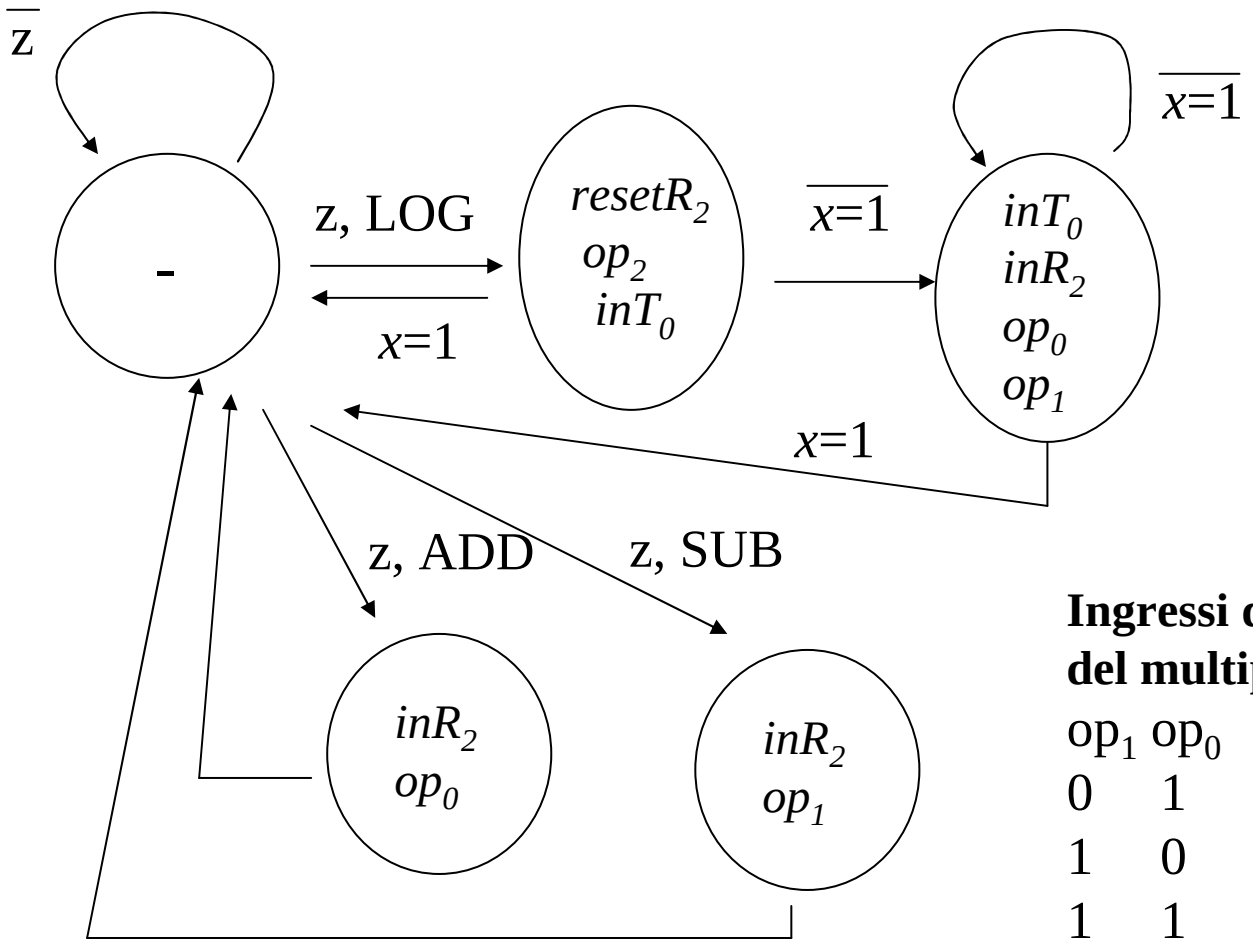
16 segnali di controllo che fanno partire una delle macchine sequenziali associata al tasto premuto

Una parte del sistema di calcolo associato alla tastiera

- Consideriamo le operazioni di **addizione, sottrazione** e **logaritmo**
- Assumiamo che R_0 e R_1 siano i registri sorgente (solo R_1 per il logaritmo) ed R_2 il registro destinazione



Una parte del sistema di controllo associato alla tastiera



Ingressi di selezione del multiplexer

op_1	op_0	
0	1	ADD
1	0	SUB
1	1	LOG

In conclusione…

- Si è detto che una macchina programmabile è in grado di eseguire programmi di lunghezza arbitraria registrati in una memoria
- Quello che abbiamo visto è solo **l'esecuzione di una generica istruzione**
- Si veda: progetto della CPU e del suo controllo