Calcolatori Elettronici A
a.a. 2008/2009

LIVELLO ORGANIZZAZIONE: SCHEMI DI BASE

Massimiliano Giacomin

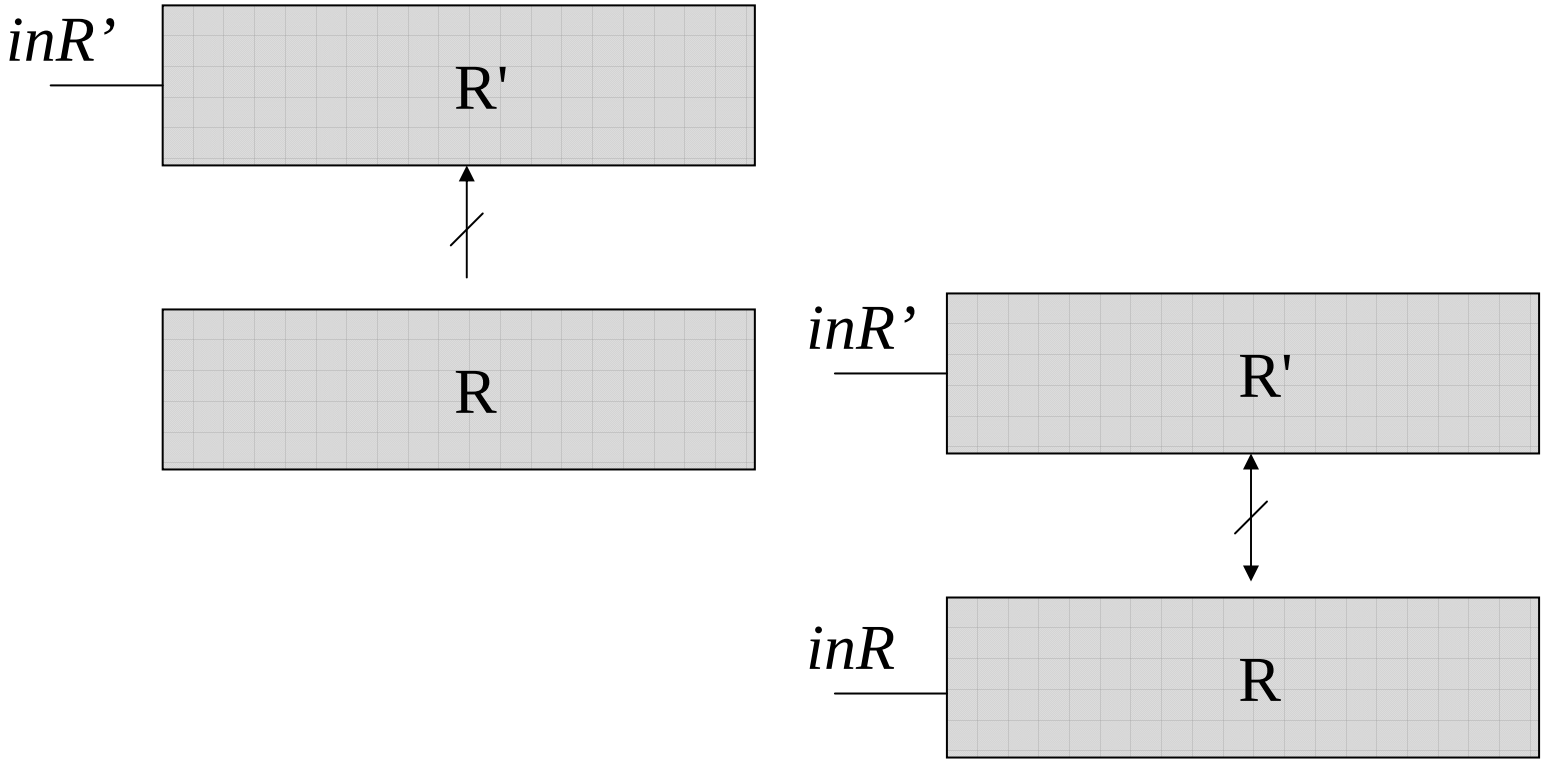
DUE ASPETTI

- Progettare circuiti per permettano di:
  1. Trasferire l'informazione da un punto a un altro del calcolatore
  2. Elaborare l'informazione
- Vedremo:
  - le principali modalità di interconnesione
  - l'organizzazione di base, a partire da quella delle macchine dedicate (non programmabili) per arrivare a quella del calcolatore

Reti combinatorie per il trasferimento di informazione

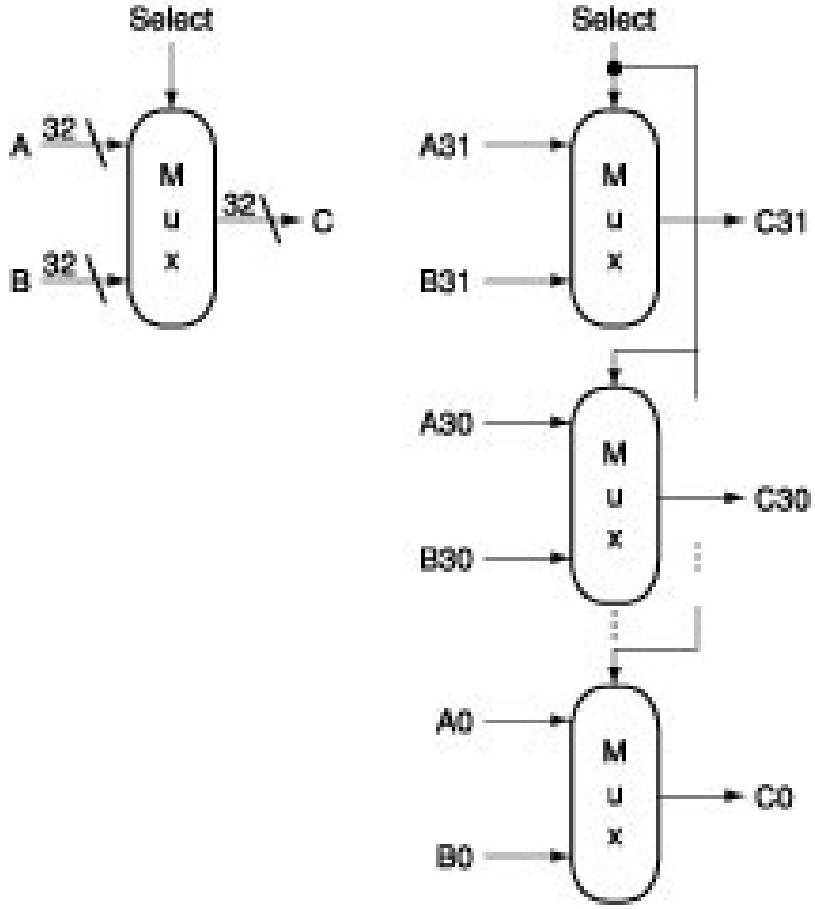
- Esistono 4 tipi di reti combinatorie per realizzare l'***interconnessione*** fra più registri:
  - **Punto-a-punto**: da registro sorgente prefissato a registro destinazione prefissato
  - **Multiplexer**: da registro sorgente variabile a registro destinazione prefissato
  - **Con decodificatore**: da registro sorgente prefissato a registro destinazione variabile
  - **Tramite bus**: da registro sorgente variabile a registro destinazione variabile

Interconnessione punto-a-punto

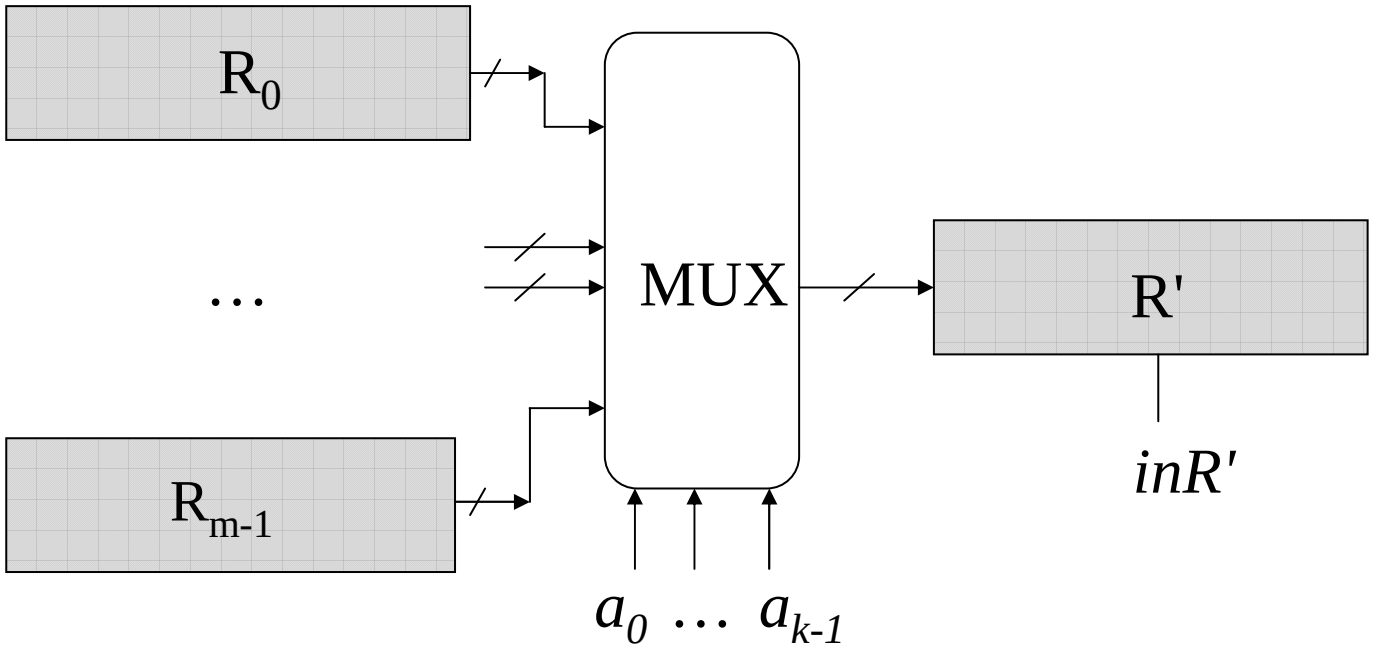


Interconnessione Multiplexer

- Si usano multiplexer che selezionano tra più bus

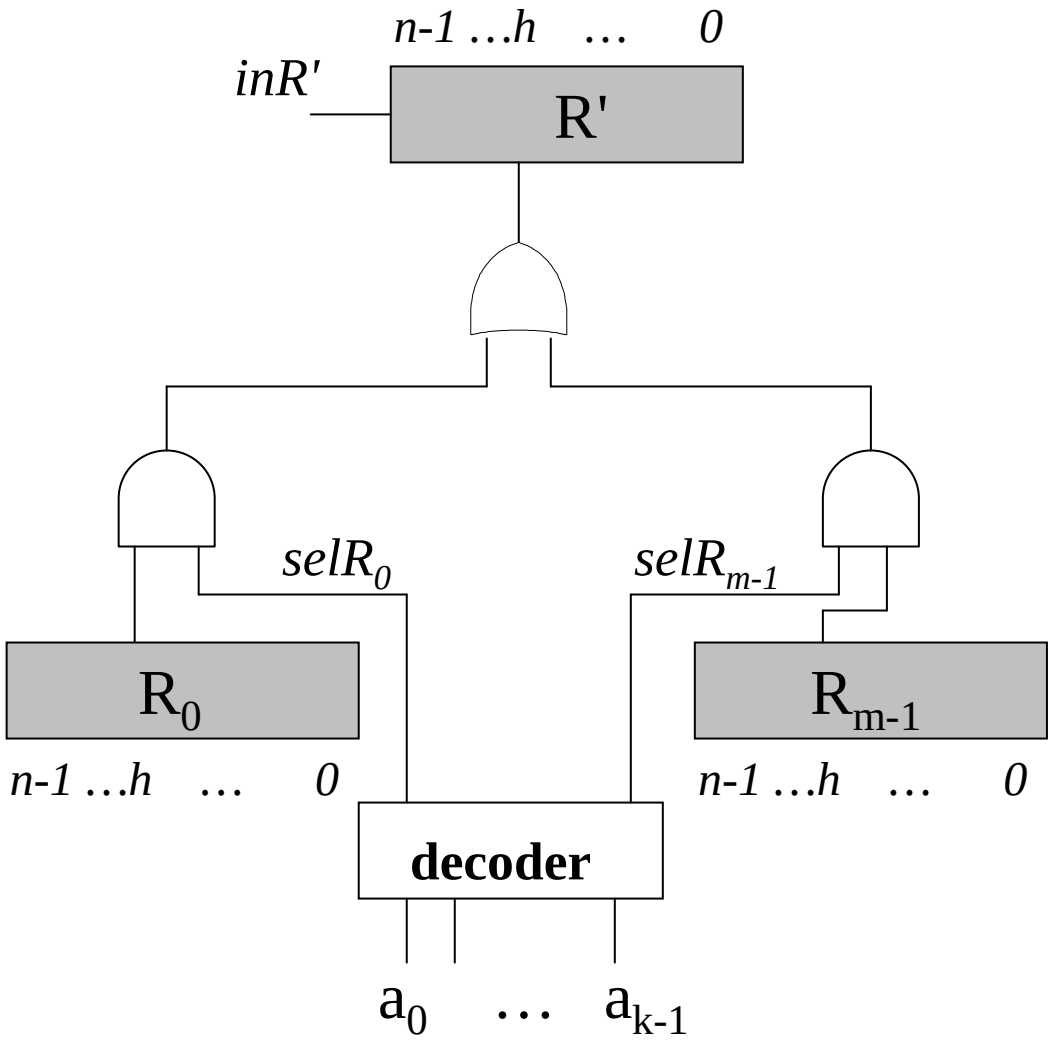


Interconnessione Multiplexer



- Il multiplexer ha come **ingressi** *m* fasci di *n* linee corrispondenti ai valori dei registri e $k = \lceil \log(m) \rceil$ linee $a_0, \ldots, a_{k-1}$ per codificare gli *m* possibili indici dei registri
- Quando *inR'* vale 1, il contenuto del registro il cui indice è codificato dalle *k* linee di selezione $a_0, \ldots, a_{k-1}$ è copiato in R'

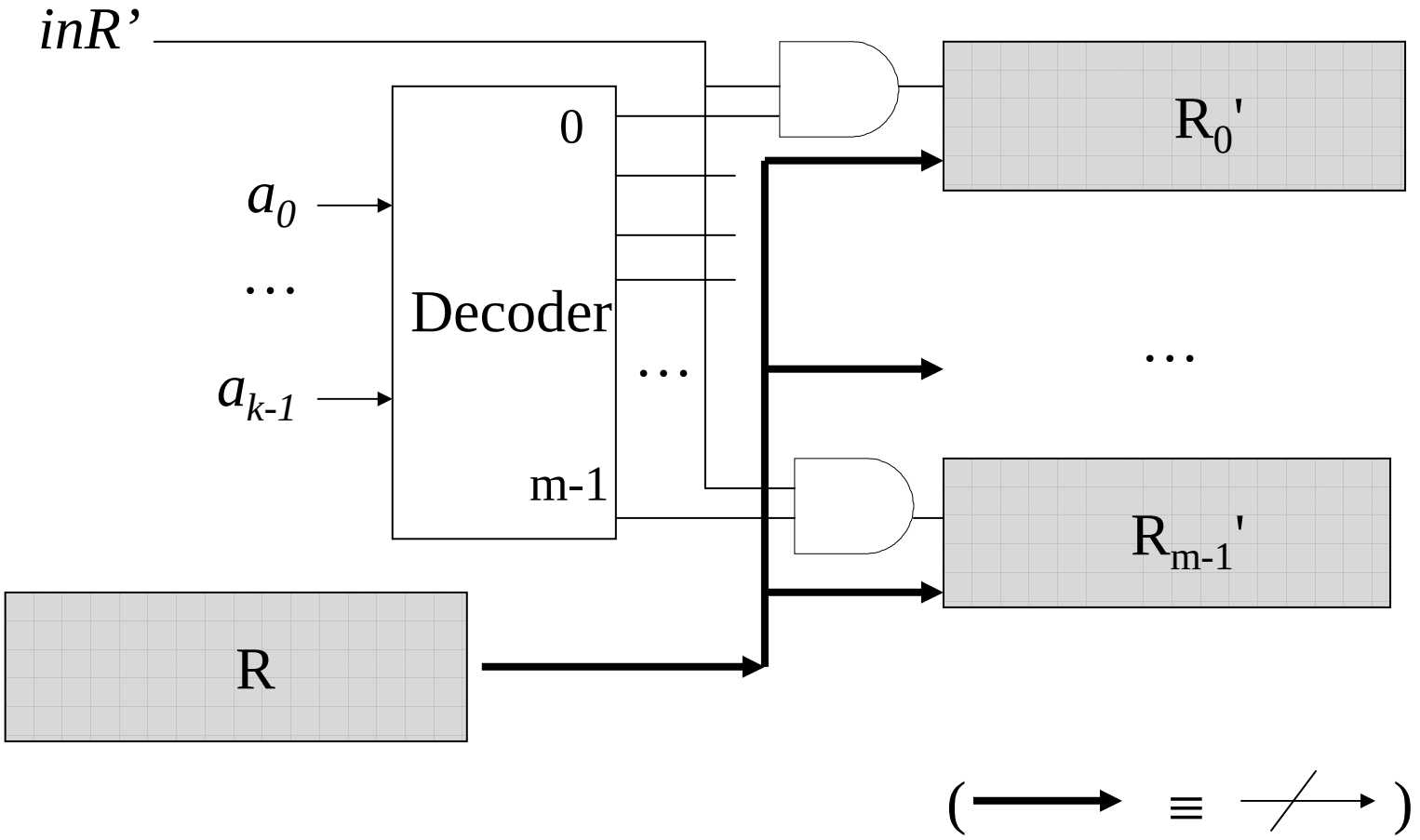
Realizzazione con un decoder



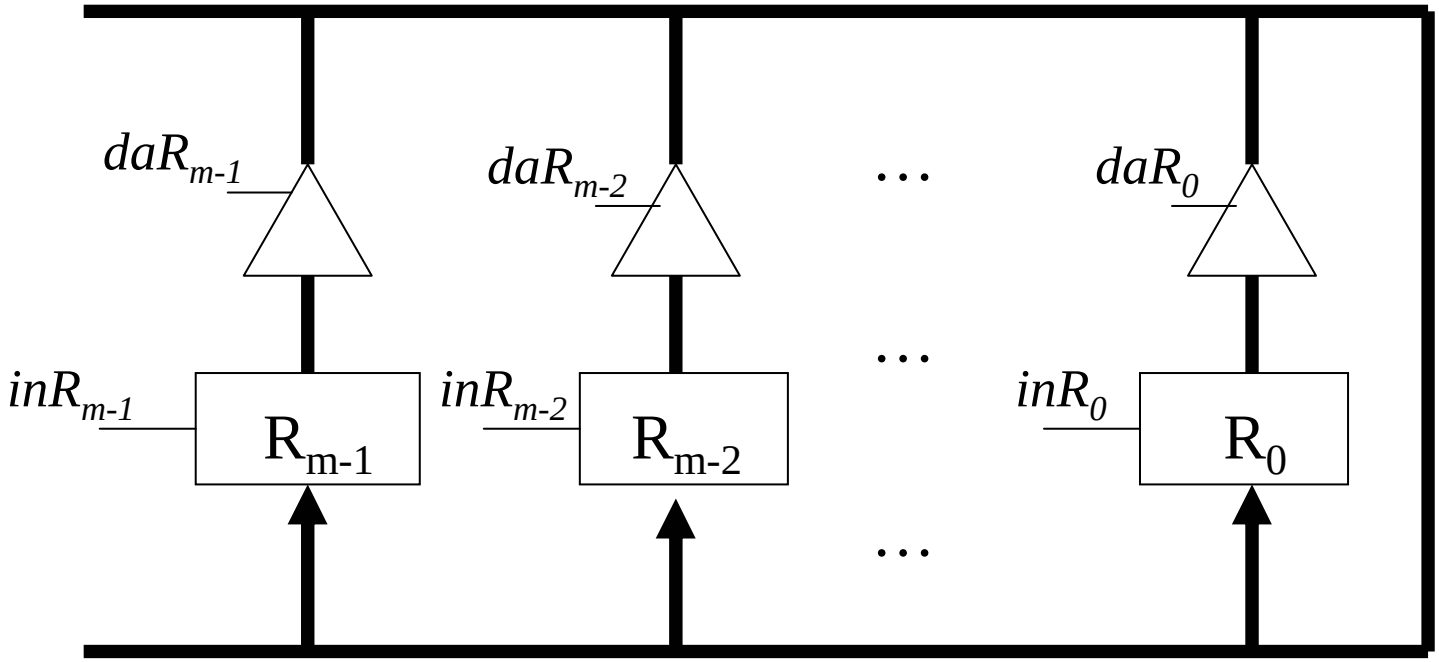
La figura illustra la rete relativa all'ingresso di un generico elemento (contenente il bit *h*-esimo) del registro destinazione R'

Il decodificatore a k ingressi è usato per creare una linea di selezione ($selR_i$) per ognuno degli *m* registri sorgente

Rete 1-a-m con decodificatore

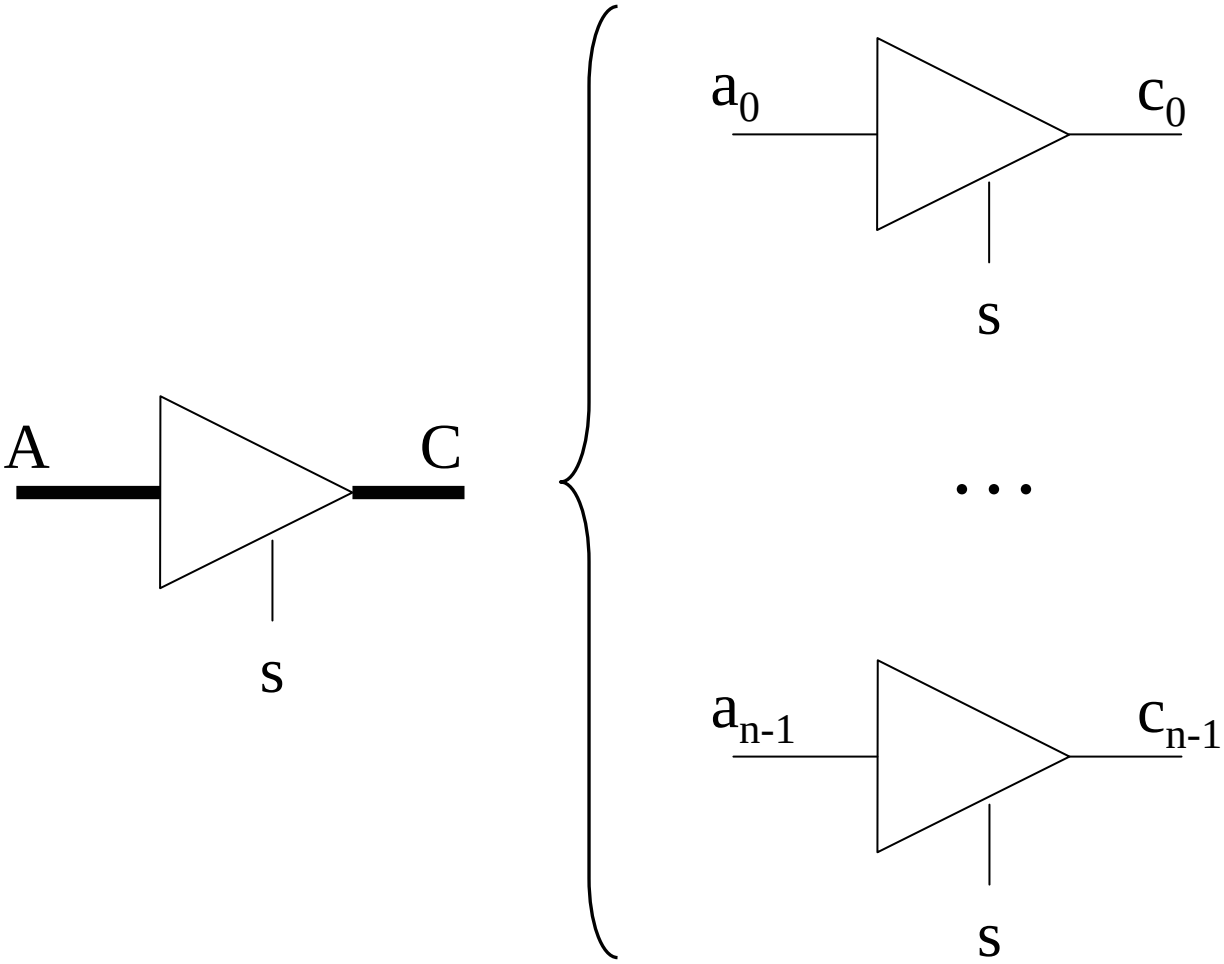


Interconnessione tramite bus



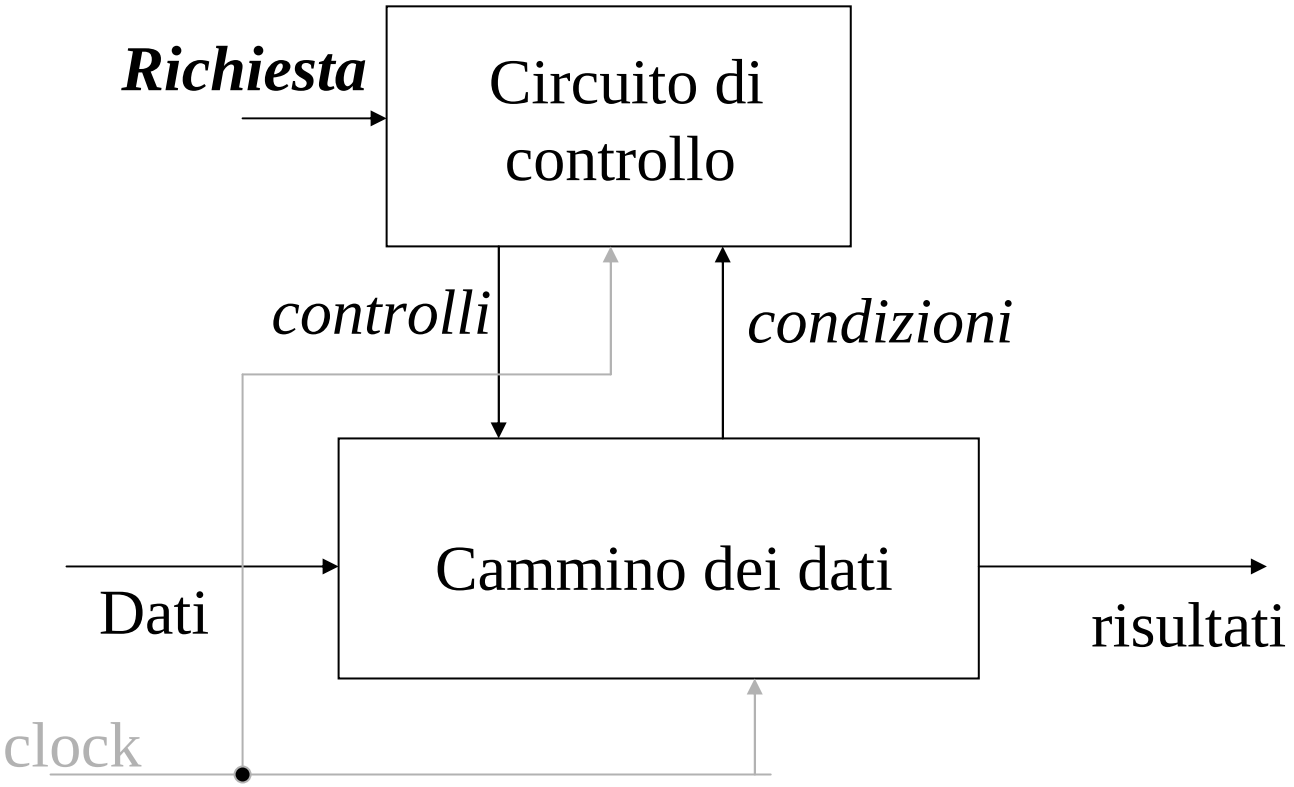
- **Trasferimento da R_i a R_j:** si attiva la linea di selezione dell'i-esimo buffer tri-state (daR_i) e quella di selezione del registro R_j (inR_j)
- I bus non consentono di effettuare **trasferimenti simultanei** nello stesso ciclo di clock (occorrono bus specializzati che collegano tra loro gruppi di registri logicamente indipendenti)

NB: come al solito, abbiamo indicato un elemento (il buffer tri-state) capace di gestire un fascio di linee



MACCHINE DEDICATE
(NON PROGRAMMABILI)

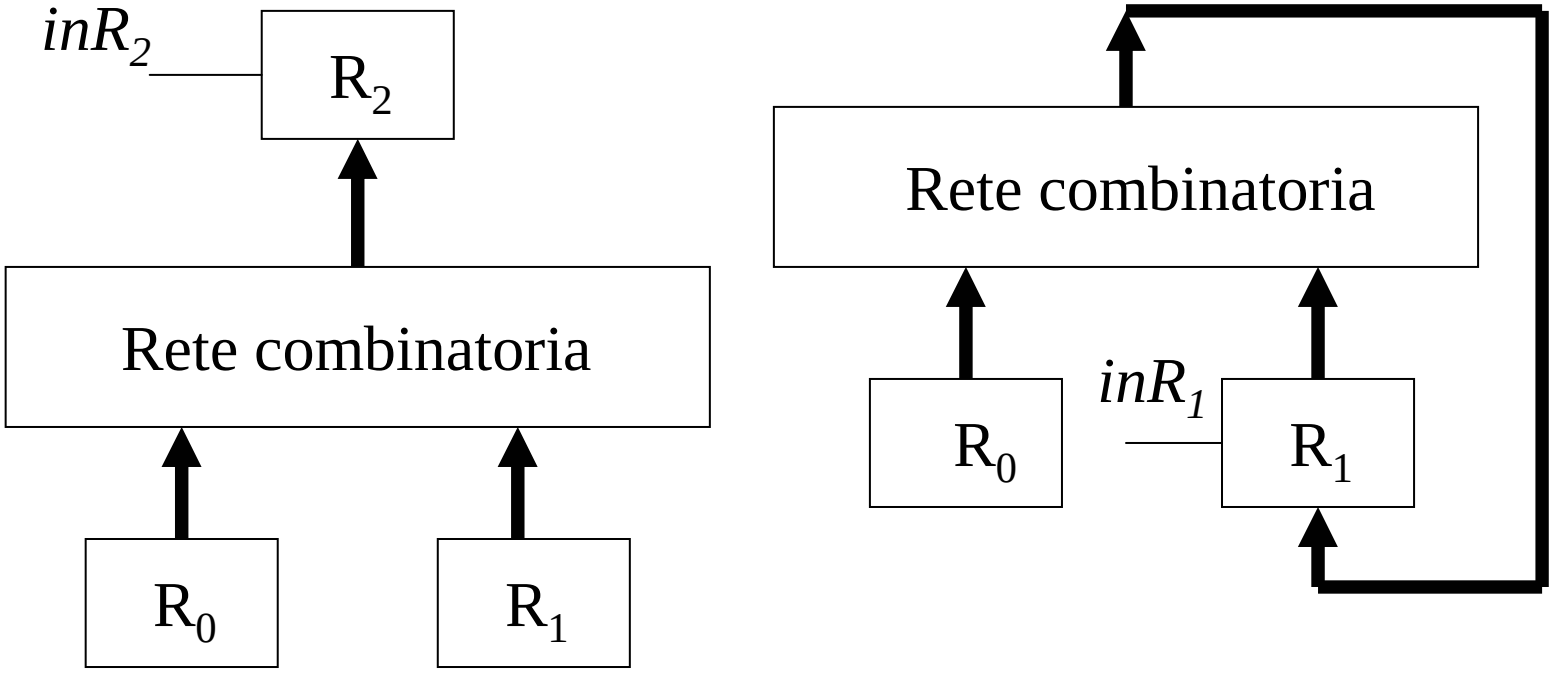
Il modello


Richiesta
Circuito di controllo
controlli
condizioni
Cammino dei dati
Dati
risultati
clock

- Assumiamo che i dati da elaborare siano contenuti in ***registri sorgente*** e il risultato sia memorizzato in un ***registro destinazione***
- Il **sistema di calcolo** include:
  - i registri sorgente e destinazione
  - le reti di interconnessione e le reti combinatorie
    che effettuano “elaborazioni semplici” (1 ciclo di clock)
  - eventuali registri di appoggio
    (mantengono risultati intermedi tra diversi cicli di clock)
- Il **sistema di controllo** è una rete sequenziale sincrona
  (aggiorna il proprio stato ad ogni ciclo di clock)

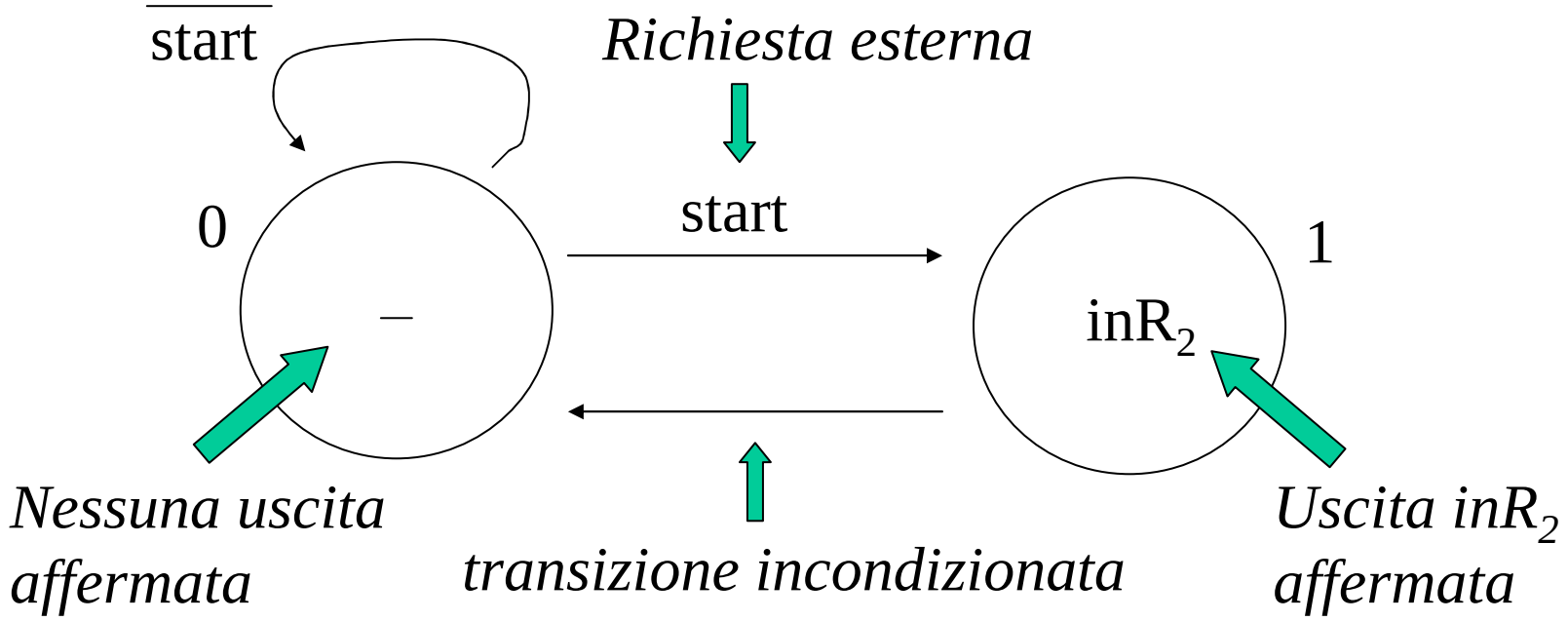
Per elaborazioni “semplici” (richiedono reti combinatorie di profondità limitata: 1 ciclo di clock): sistema di calcolo e di controllo semplici

Circuito di calcolo (elaborazioni semplici)



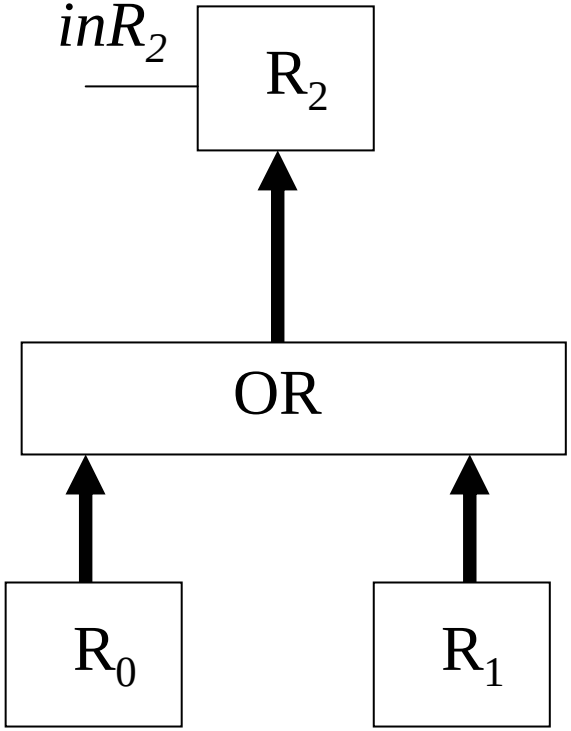
Circuito di controllo (elaborazioni semplici)

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

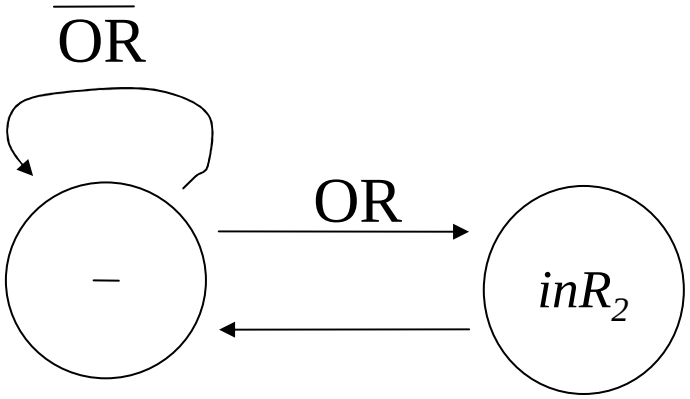


Esempio: OR

Sistema di calcolo

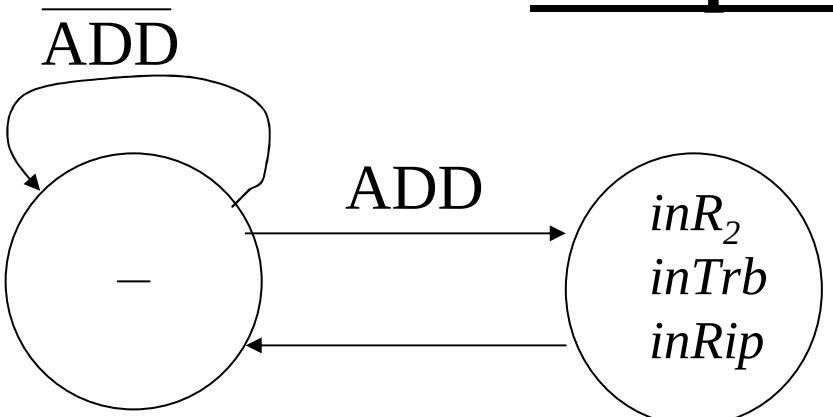


Sistema di controllo



- 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

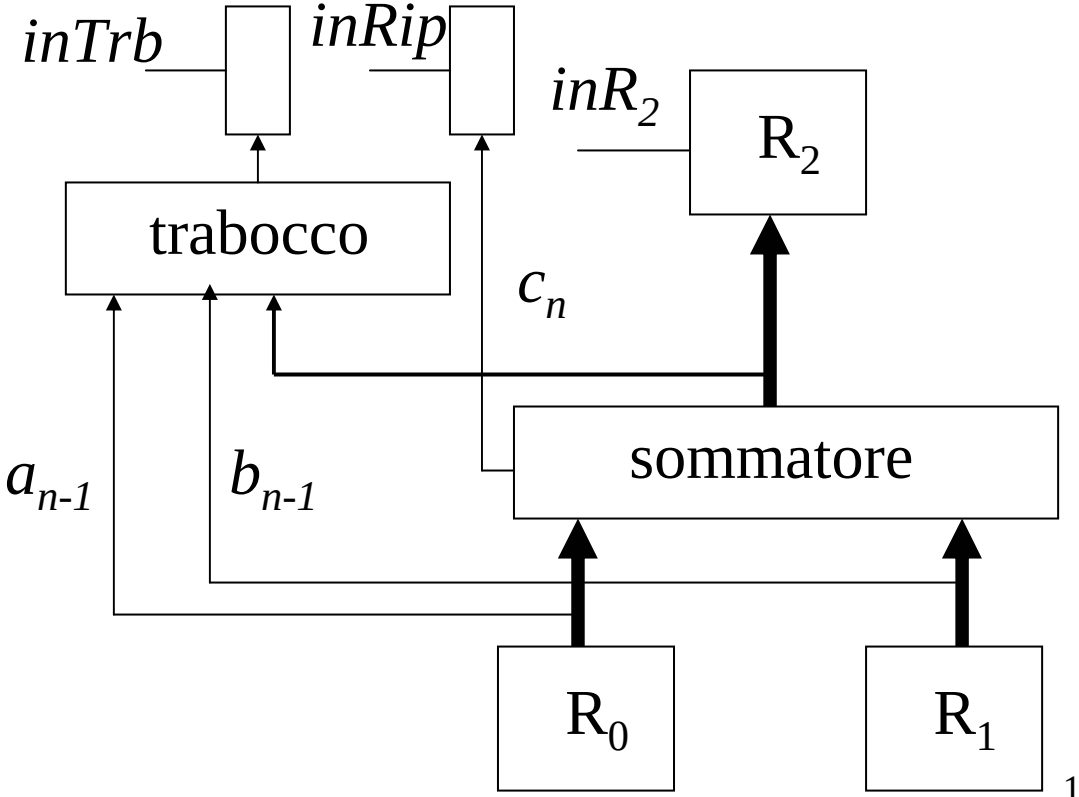
$\{\overline{ADD}, ADD\}$

uscite

$\{-, inR_2, inTrb, inRip\}$

Sistema di controllo

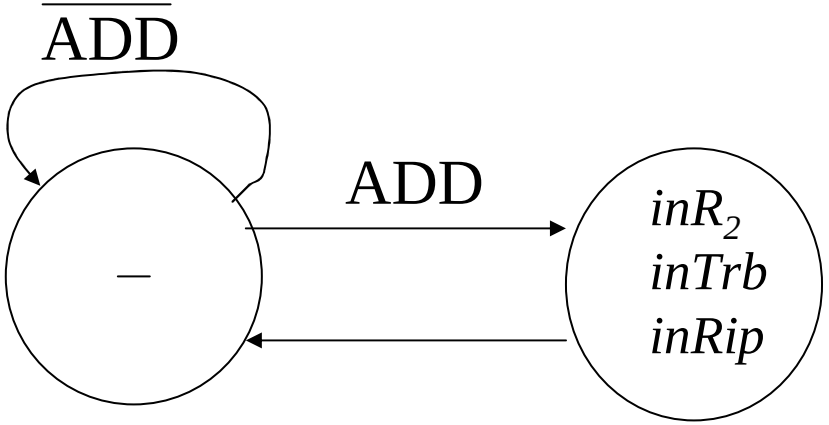
Sistema di calcolo

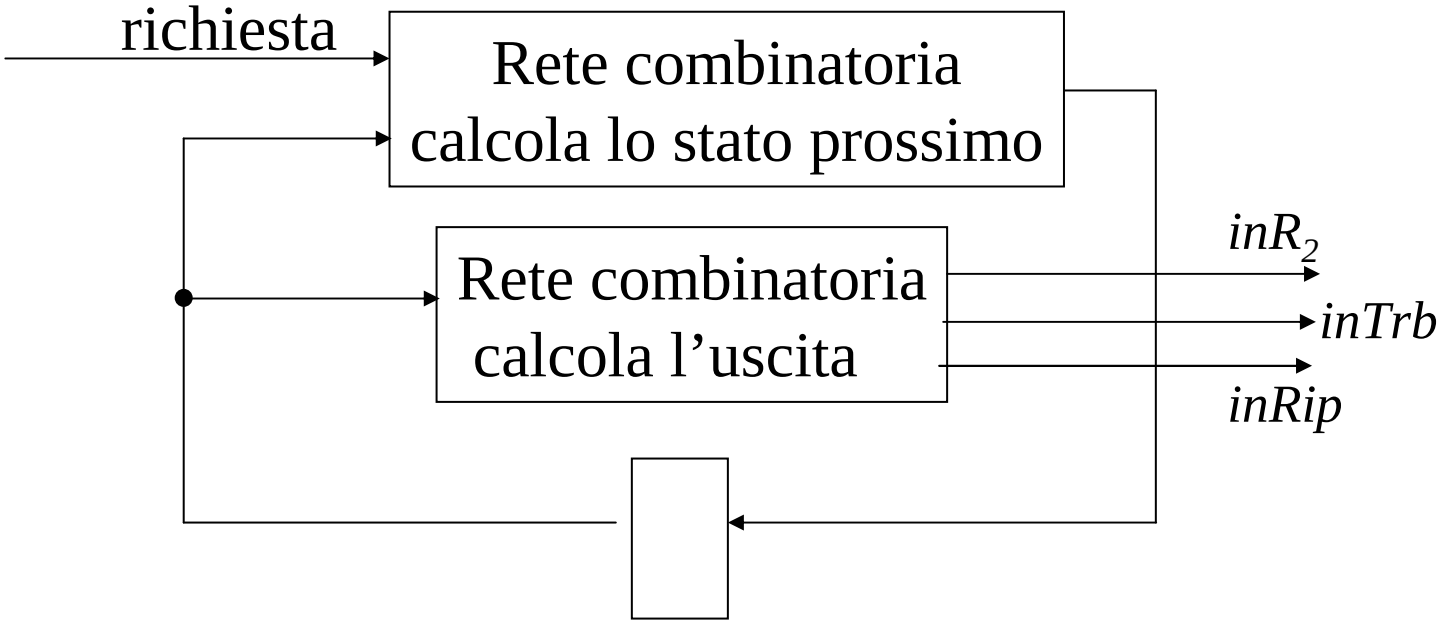


Esempio: addizione(2):

come realizzo il controllo

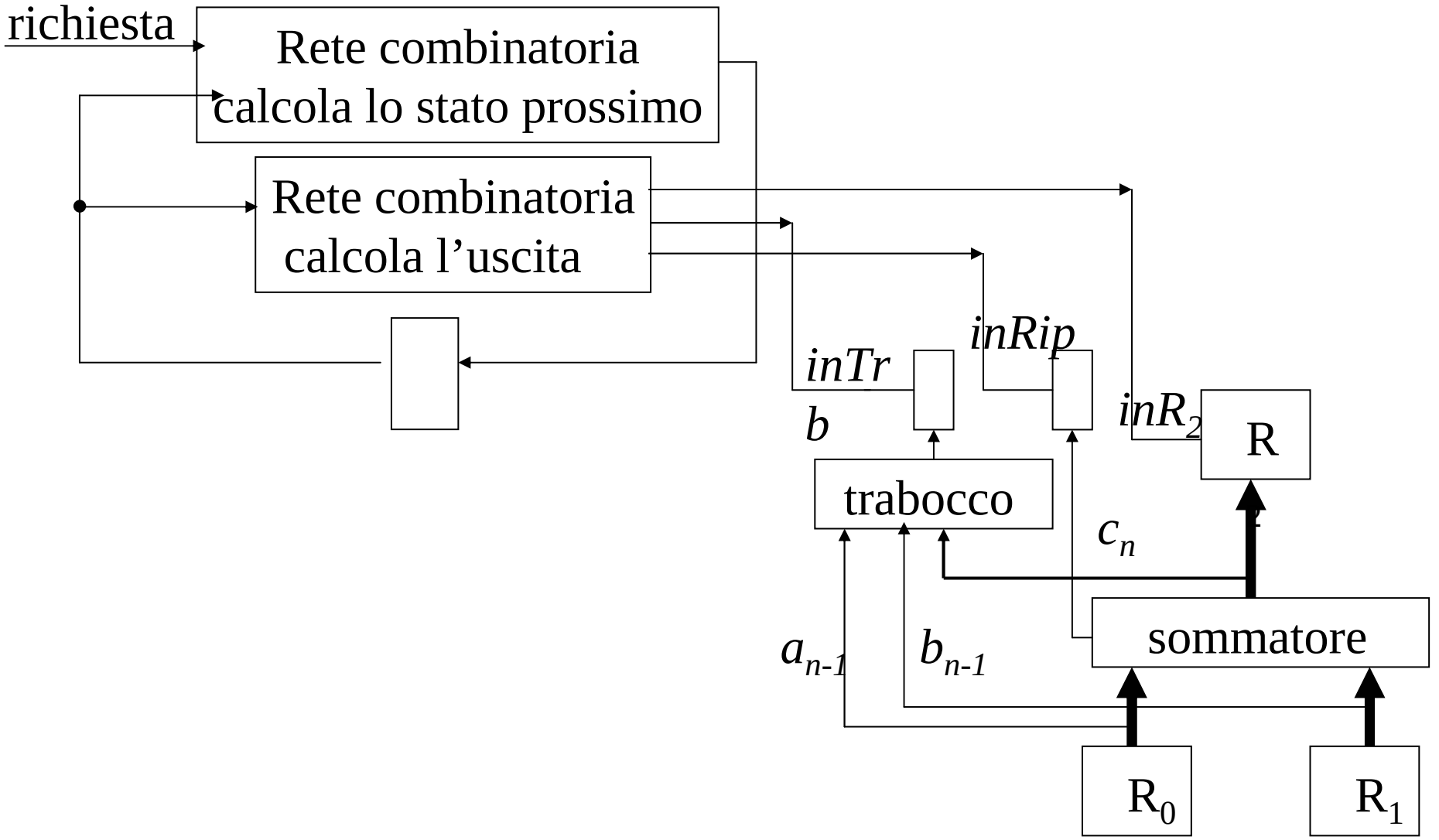
Sistema di controllo





Esempio: addizione(3)

il circuito completo



Un altro esempio: la moltiplicazione

moltiplicando 1101 ×

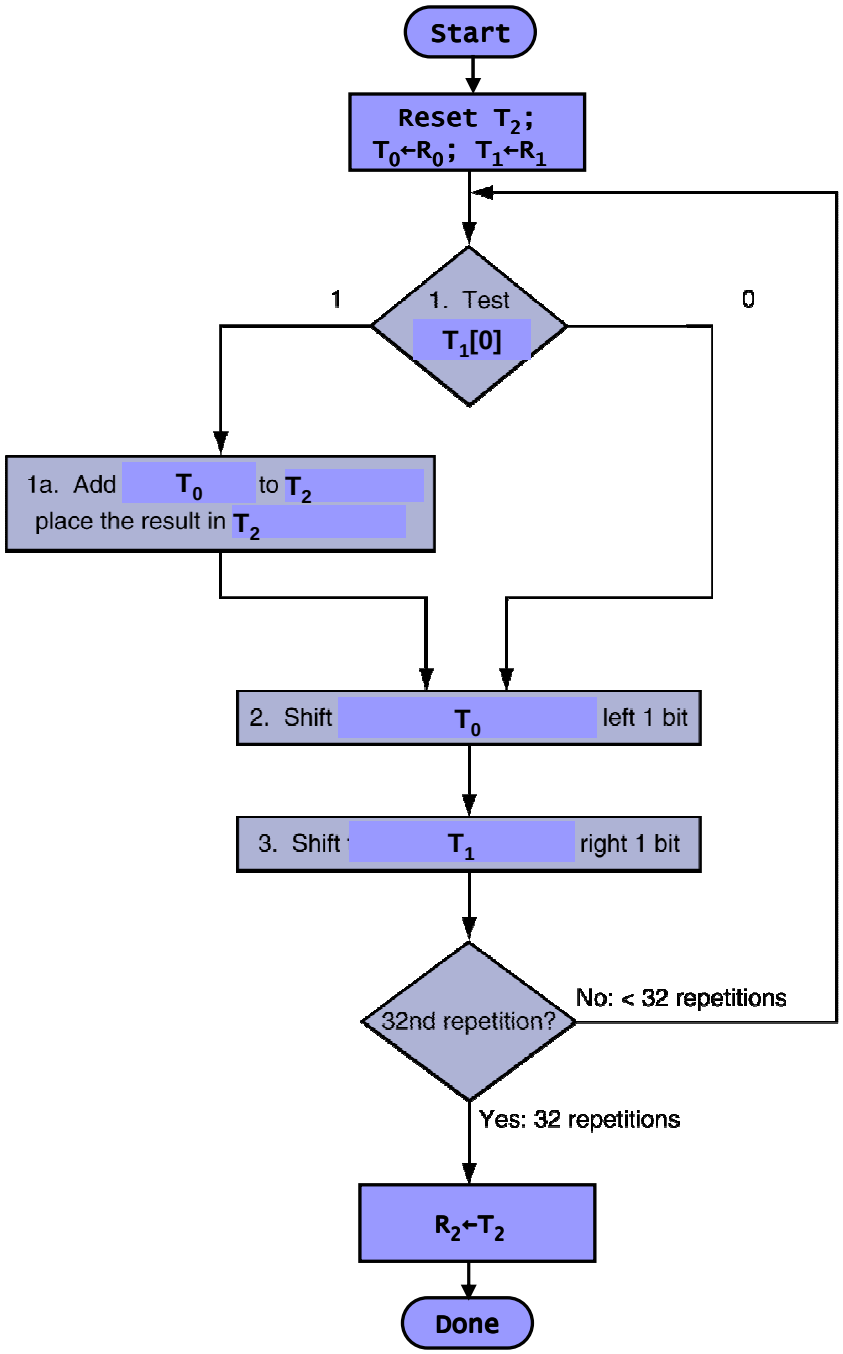
moltiplicatore 1011 =

$x=1101_2=13_{10}$

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

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		

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



Assumiamo:

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

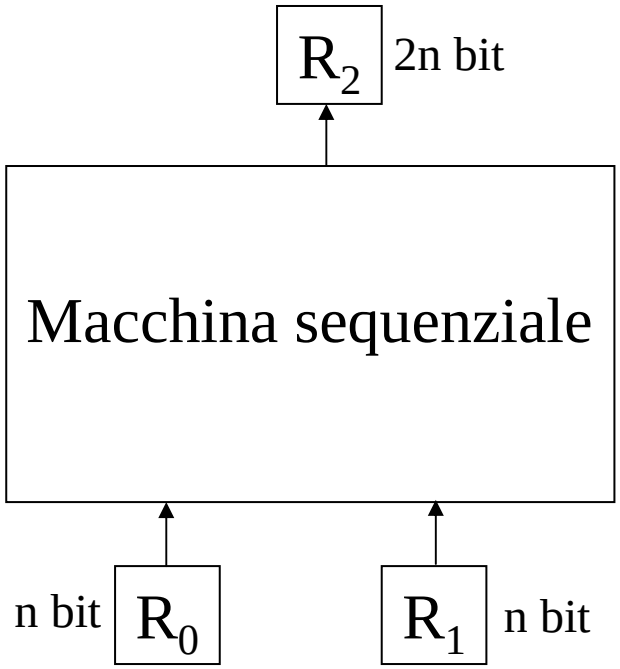
Usiamo:

- T_0 per elaborare moltiplicando
- T_1 per il multiplicatore
- T_2 per il prodotto

(NB: n=32 in questo caso)

L’algoritmo precedente è realizzabile con:

- un programma
- hardware

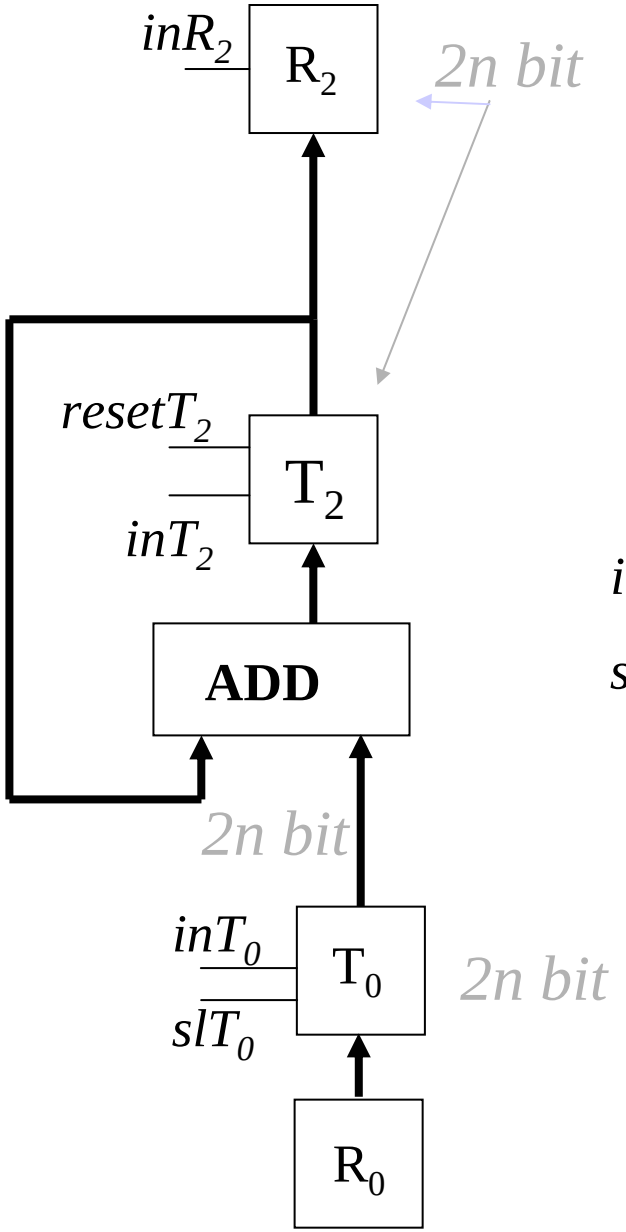


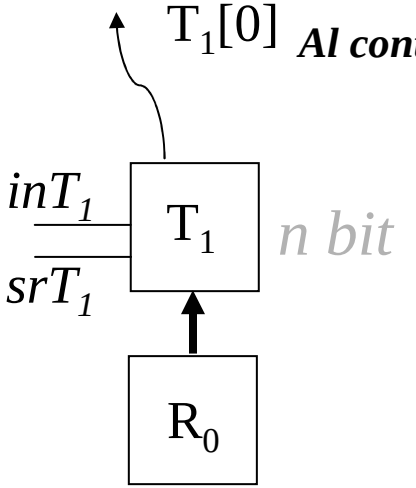
I calcolatori presentano molte organizzazioni diverse.

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: *Hi* e *Lo* e due istruzioni: *mflo* e *mfhi*

Sistema di calcolo per la moltiplicazione



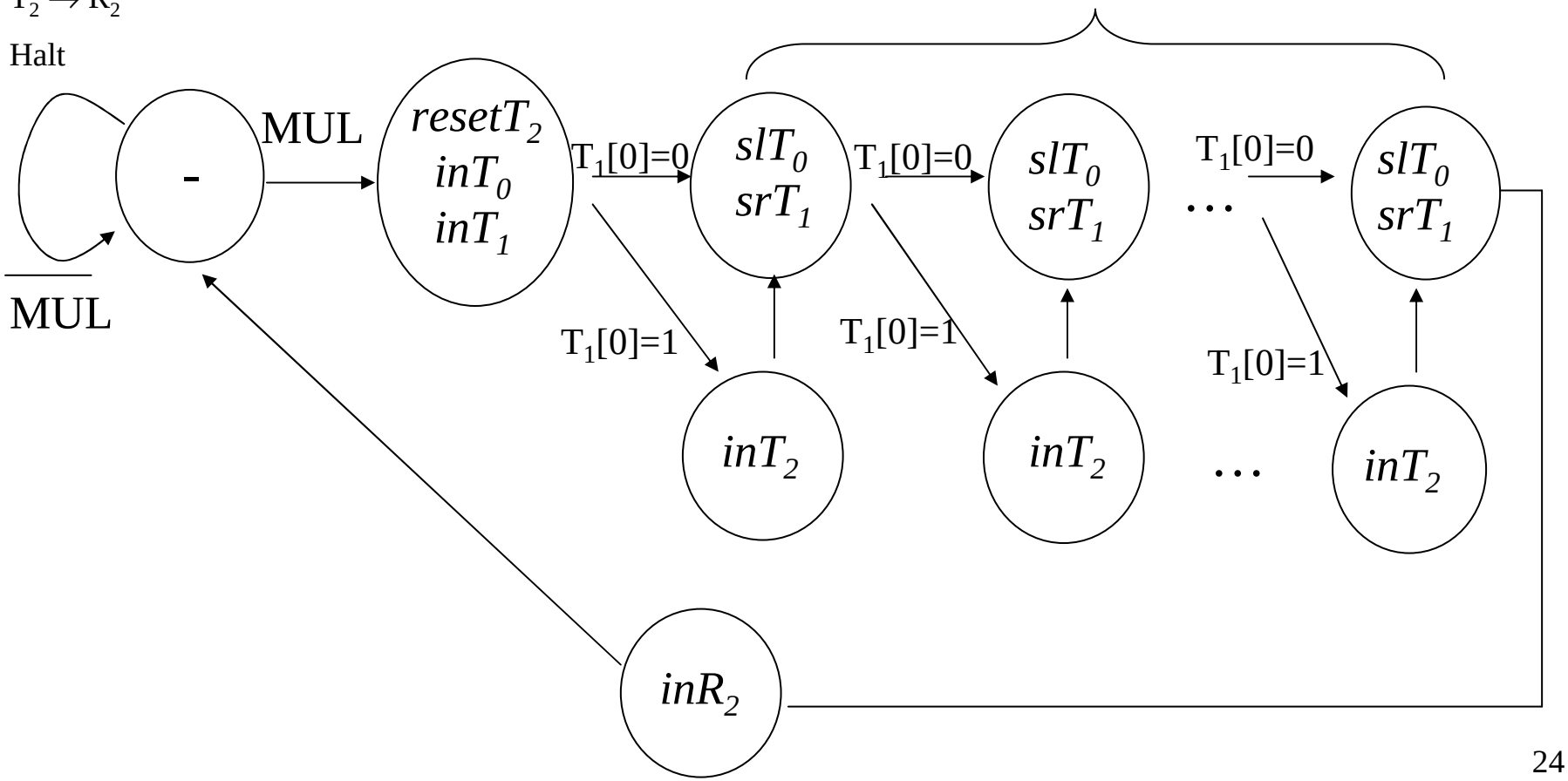


[se **T_1[0] = 1** occorre sommare T_0 a T_2]

Sistema di controllo per la moltiplicazione

1. Reset T_2; $R_0 \rightarrow T_0$; $R_1 \rightarrow T_1$
2. Se $T_1[0]=1$ vai al passo 3 altrimenti vai al passo 4
3. $T_0+T_2 \rightarrow T_2$
4. $T_1/2 \rightarrow T_1$; $T_0*2 \rightarrow T_0$,
5. Se n-esima ripetiz 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

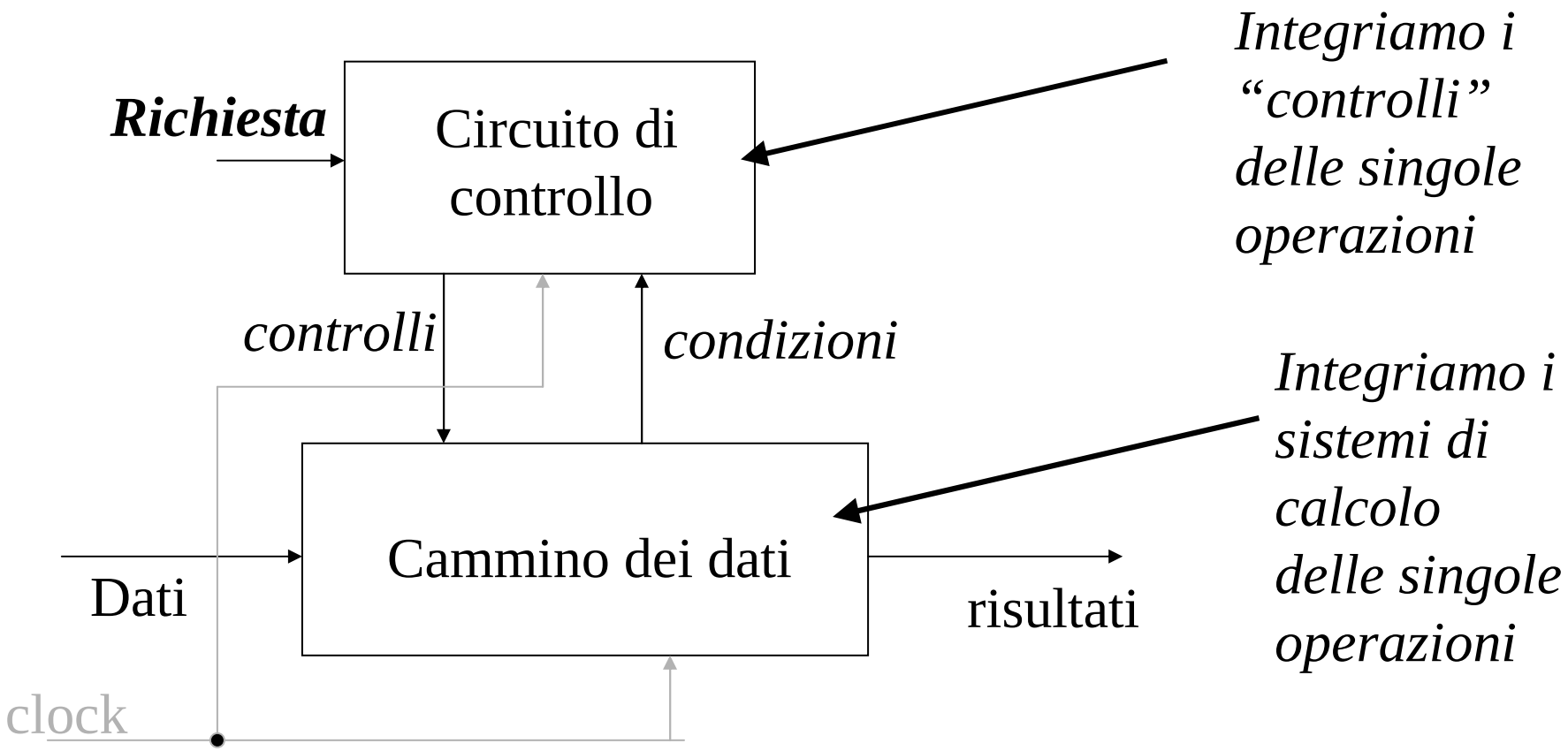


Note:

- occorre tener conto del segno (un modo è convertire i numeri in valore assoluto e alla fine stabilire il segno a seconda di quello degli operandi)
- esistono implementazioni hardware più efficienti (si usano sommatori per parallelizzare le somme)

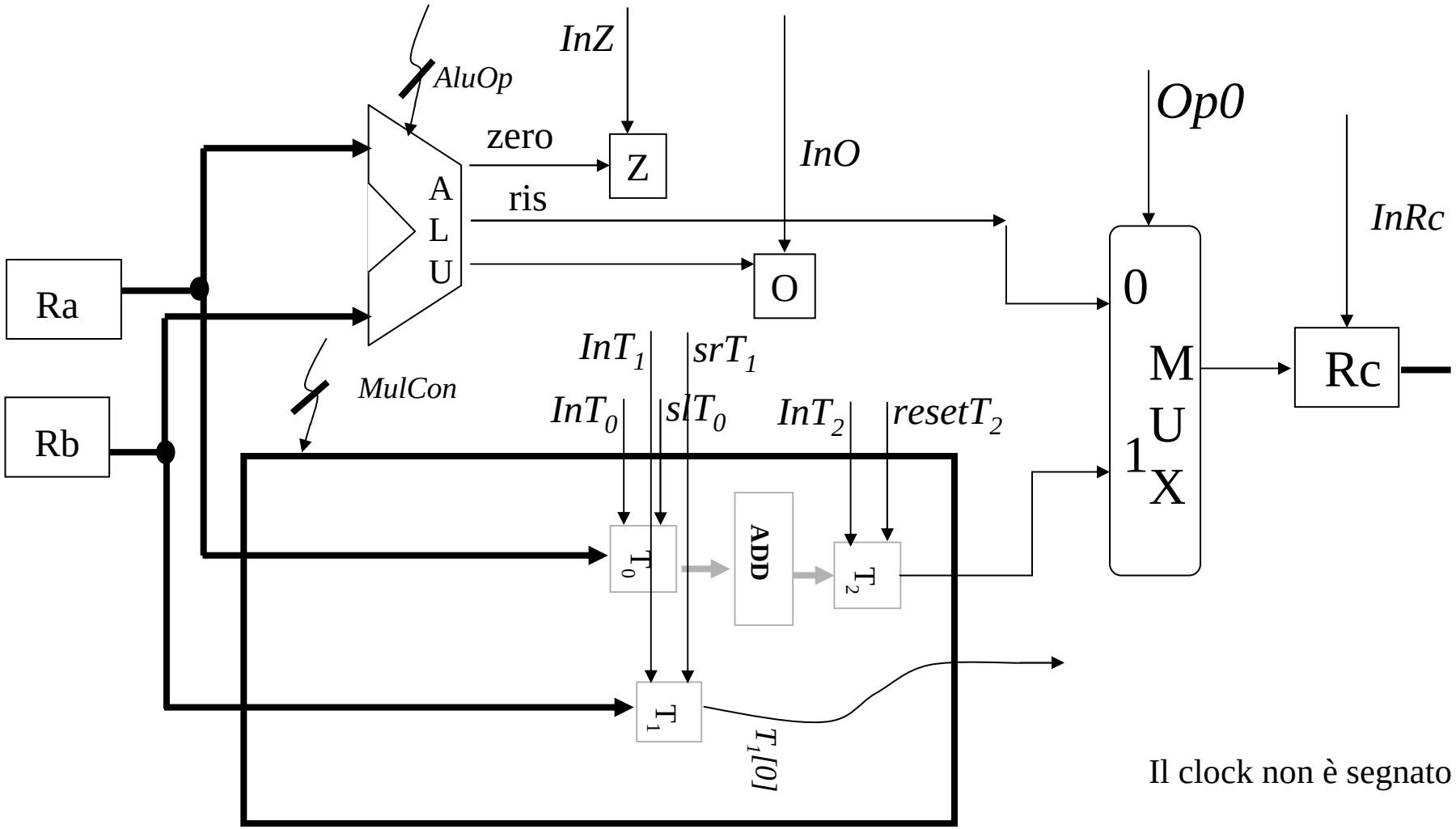
Una calcolatrice

capace di eseguire +,-, =,<,∨,∧ e x

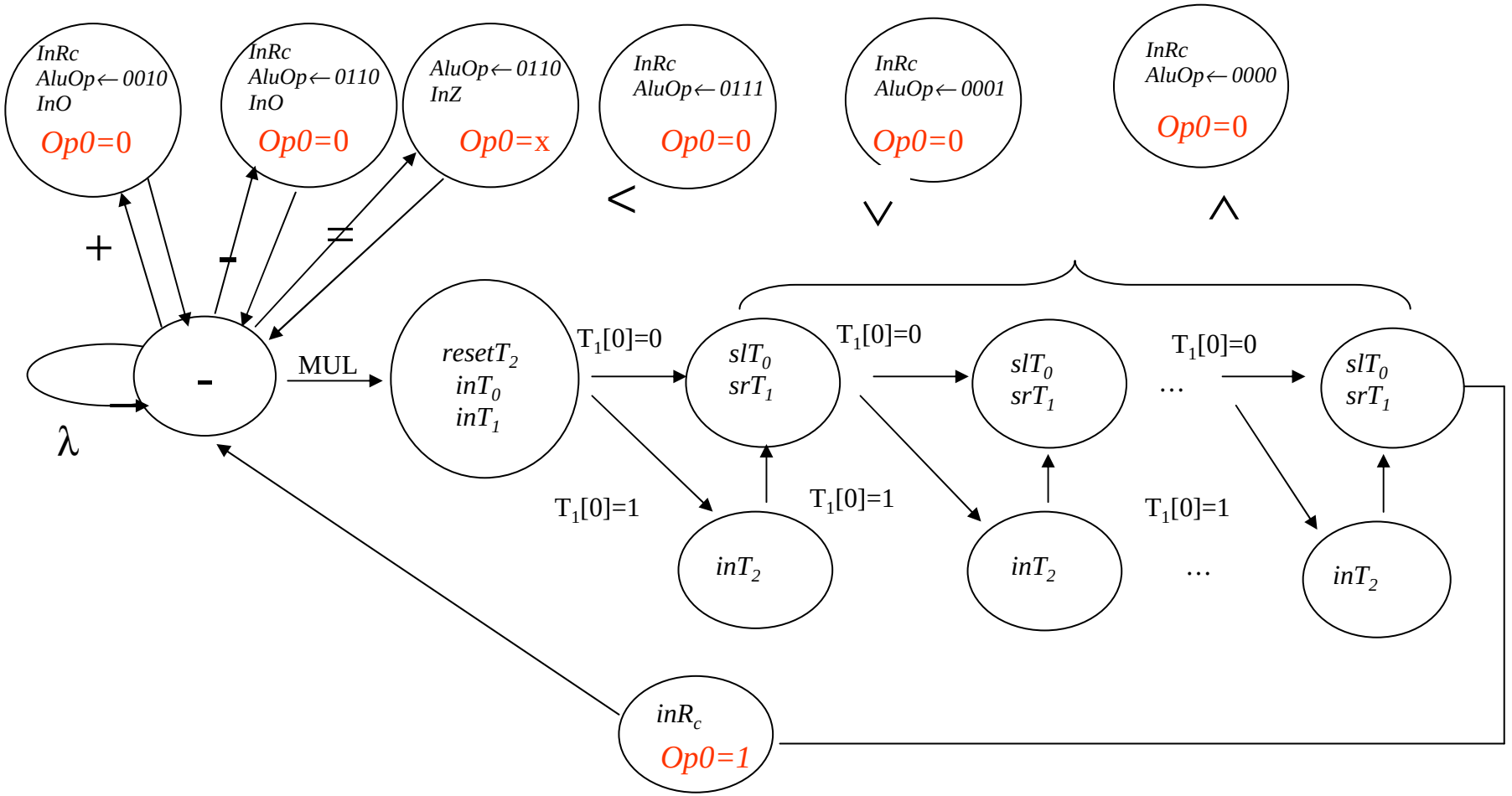


Sistema di calcolo

- tutti i segnali di controllo vengono da circuito di controllo
- $T_1[0]$: condizione che va al circuito di controllo



Il clock non è segnato



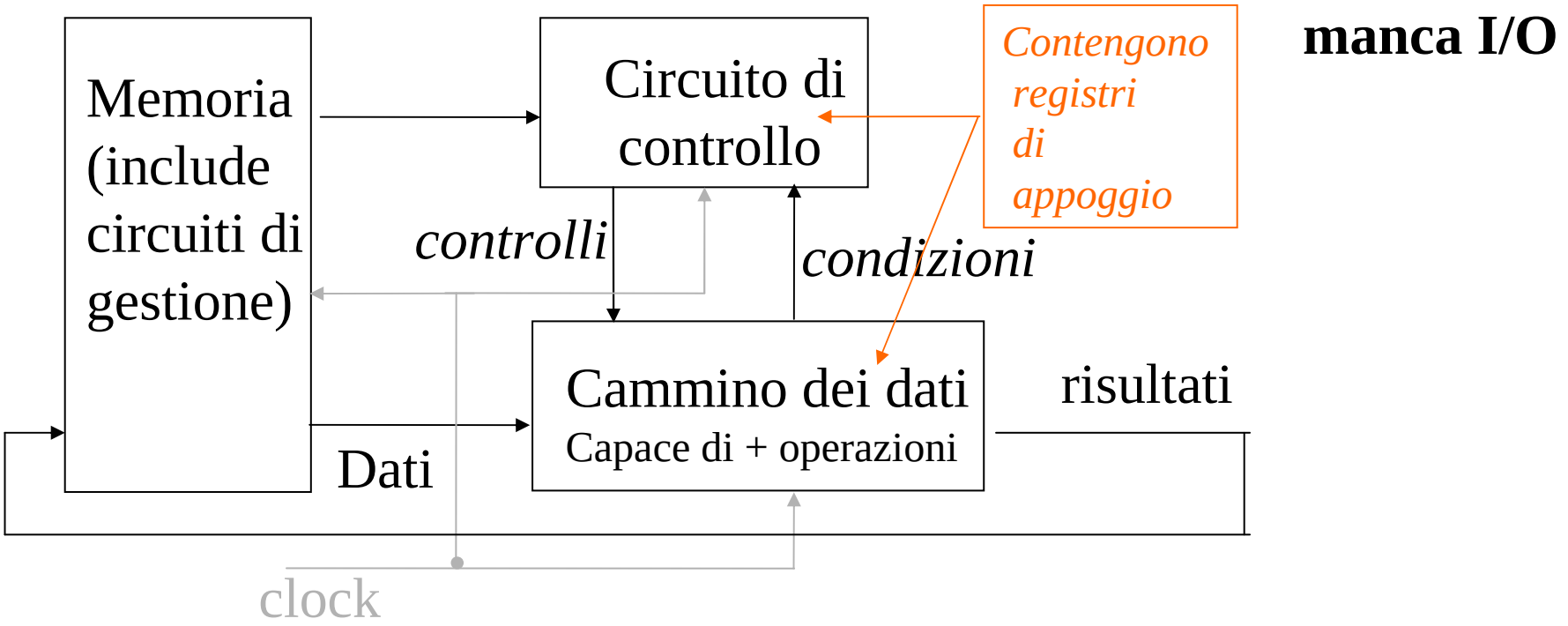
NB: tipicamente si usa la convenzione

- segnale di scrittura non segnato: significa 0
- controllo MUX non segnato: significa don’t care

IL CALCOLATORE:
UNA MACCHINA PROGRAMMABILE

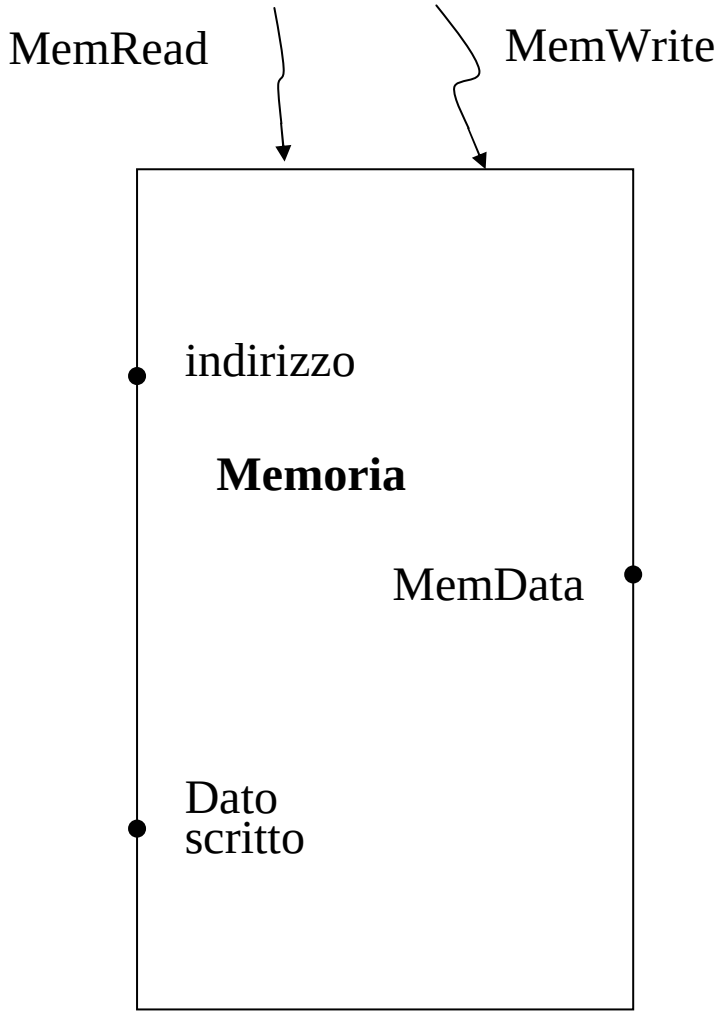
- Quelle che abbiamo visto finora sono macchine “dedicate” in quanto il loro sistema di controllo è progettato per svolgere una **specifica elaborazione** (oppure: una elaborazione di un insieme prefissato)
- Un calcolatore è invece una **macchina programmabile**:
  - le operazioni da svolgere sono descritte da una sequenza di istruzioni **registrate in una memoria** di lunghezza arbitraria (**programma**)
- Vediamo l’organizzazione di base (in seguito verrà dettagliata)

Lo schema di base



- i contenuti dei registri di memoria possono essere interpretati come dati o come istruzioni
- il circuito di controllo decodifica le istruzioni e comanda la loro esecuzione
- le istruzioni vengono alimentate automaticamente:
  necessità di un meccanismo che calcoli il flusso delle istruzioni

La memoria



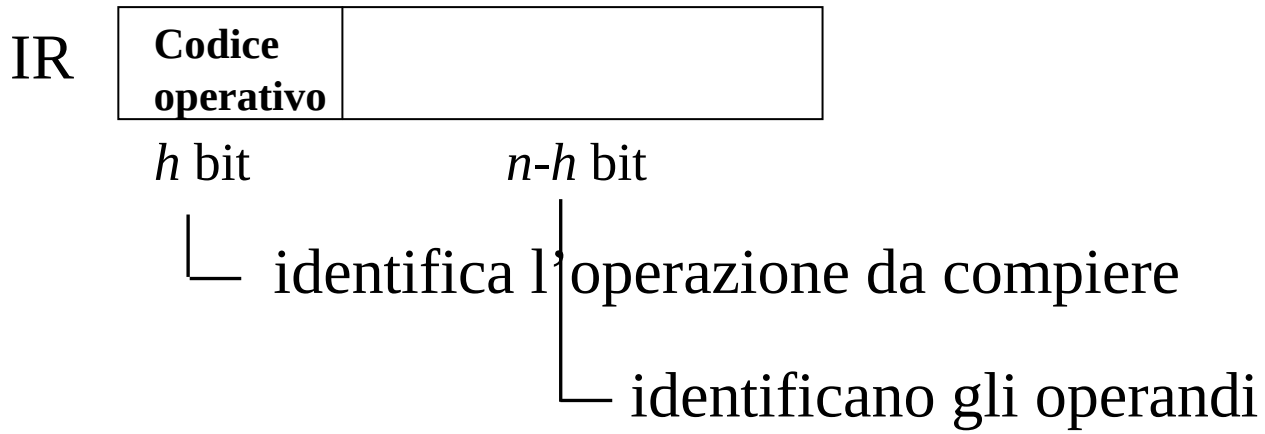
Come trovo un'istruzione?

Utilizzo un registro PC che mantiene l'indirizzo dell'istruzione corrente

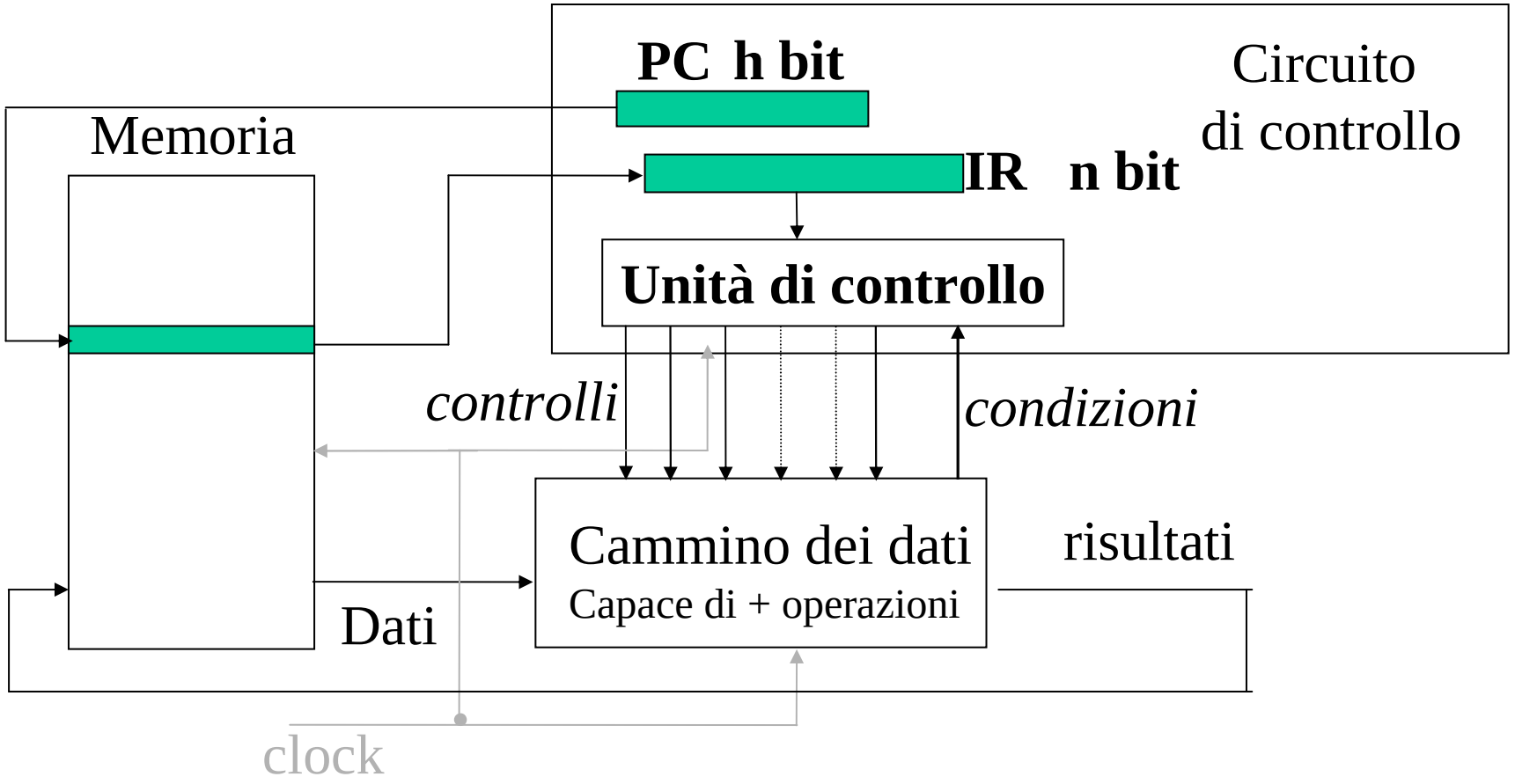
Dove memorizzo un'istruzione?

Utilizzo un registro IR che mantiene l'istruzione corrente
(NB: anche se vedremo un'implementazione che ne fa a meno!)

Interpretazione di una istruzione

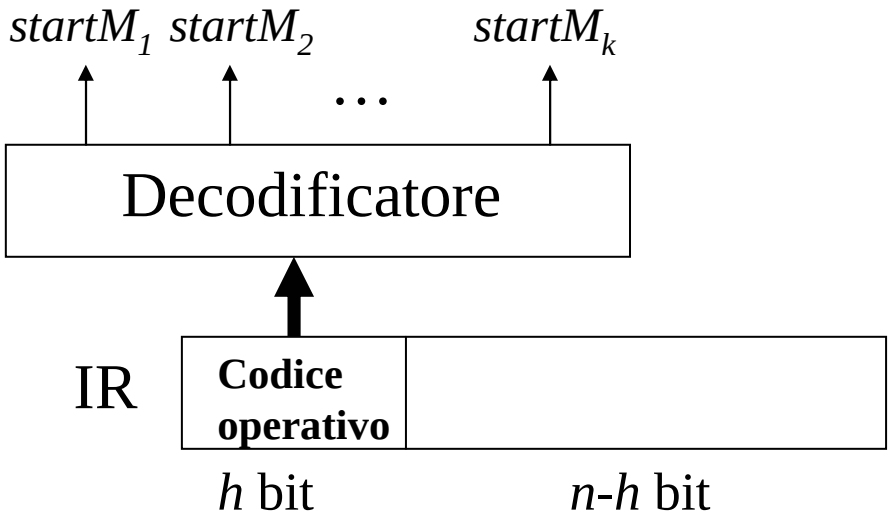


Lo schema di base (2)



Una semplice possibilità…

- 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 possiamo identificare con il ***codice operativo***



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

ESEMPIO CON 4 MACCHINE

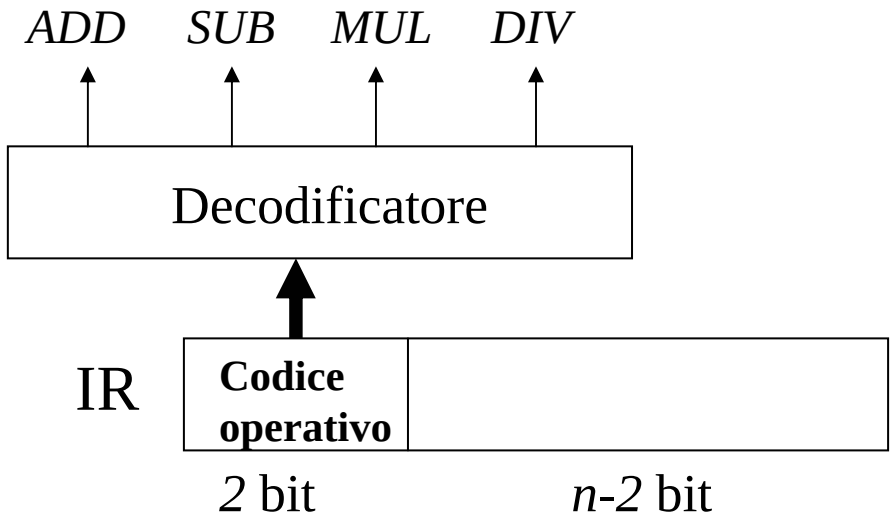
Codice operativo per le 4 macchine

00 → addizione

01 → sottrazione

10 → moltiplicazione

11 → divisione

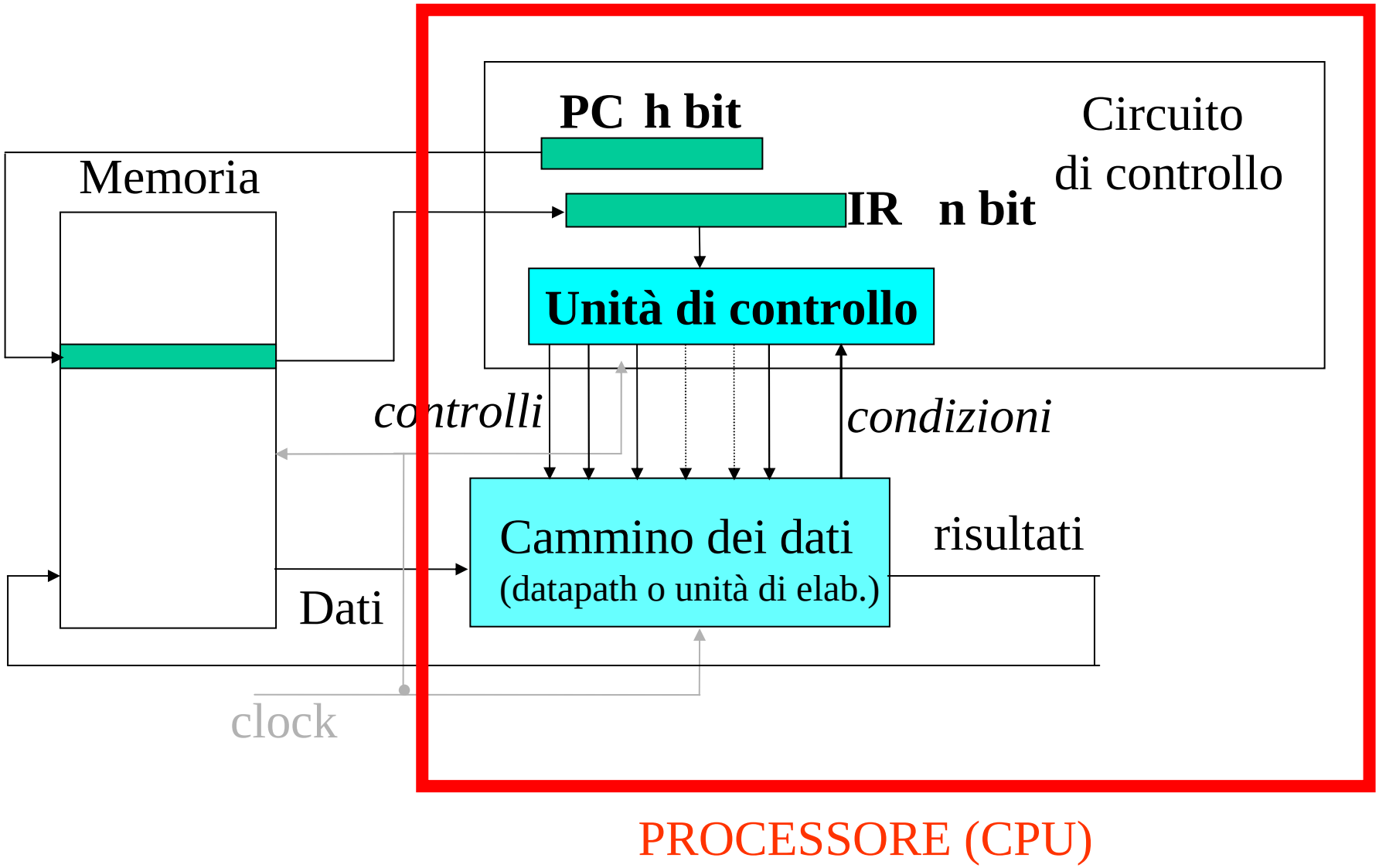


Codice operativo		*DIV*	*MUL*	*SUB*	*ADD*
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

IN REALTA’

- Non si usano macchine separate per le singole istruzioni, ma si cerca di utilizzare la stessa unità funzionale per più istruzioni
- Istruzioni diverse possono avere parti (stati) in comune

⇨ Progetto della CPU e del suo controllo



PROCESSORE (CPU)