Dalle istruzioni alle microoperazioni

SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Pellegrini

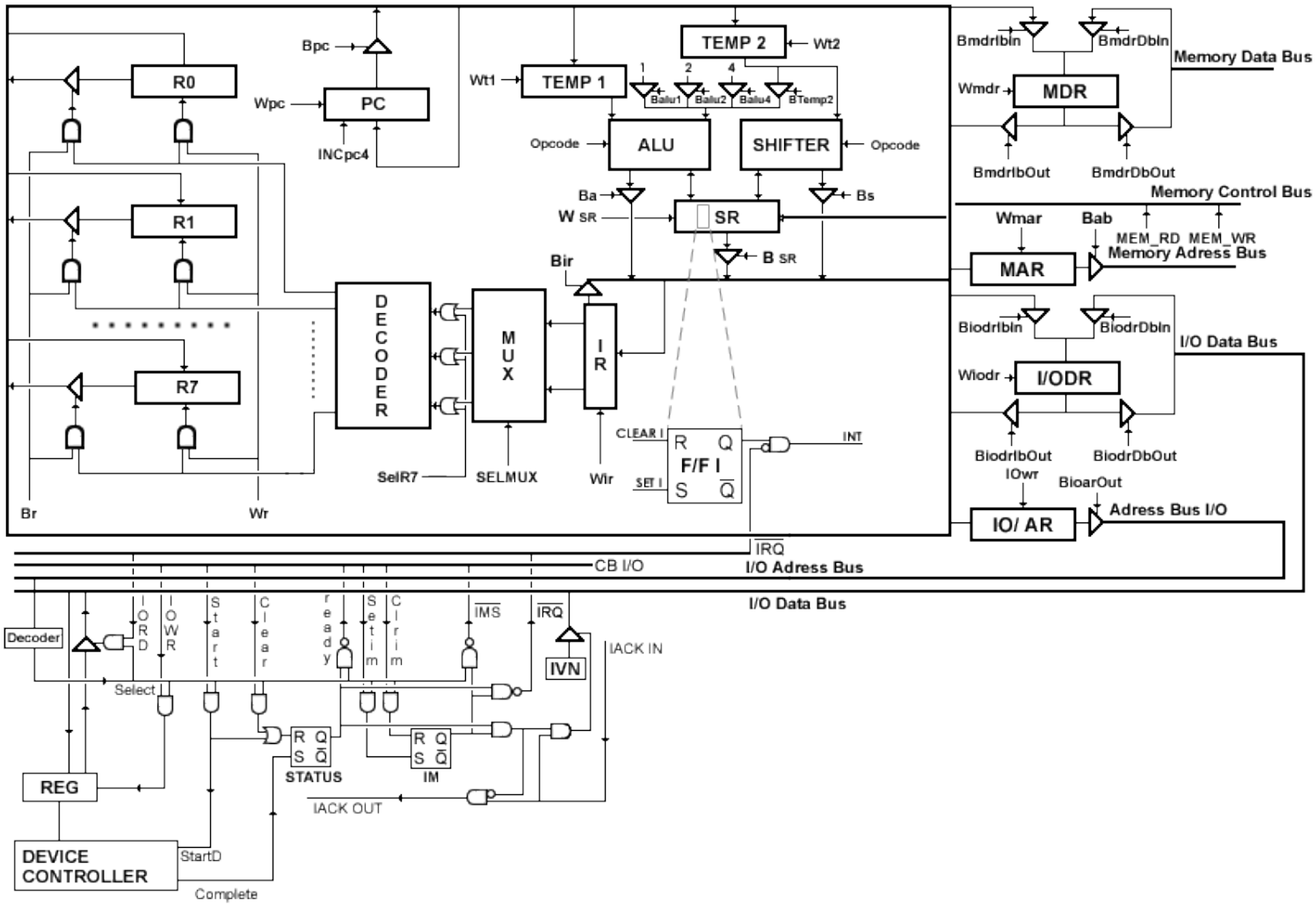
Calcolatori Elettronici
Sapienza, Università di Roma

A.A. 2012/2013

Le microoperazioni

- La fase di esecuzione di un'istruzione può essere divisa in più fasi
  - In un singolo ciclo macchina, il processore può non essere in grado di eseguire tutte le operazioni associate
  - Ad esempio, un'istruzione `ADD` richiede il movimento dei dati verso la ALU, il calcolo del risultato e la riscrittura del risultato
- Il SCO della CPU implementa un'istruzione tramite una serie di microoperazioni, ciascuna eseguita in un ciclo macchina

PD32: Architettura completa

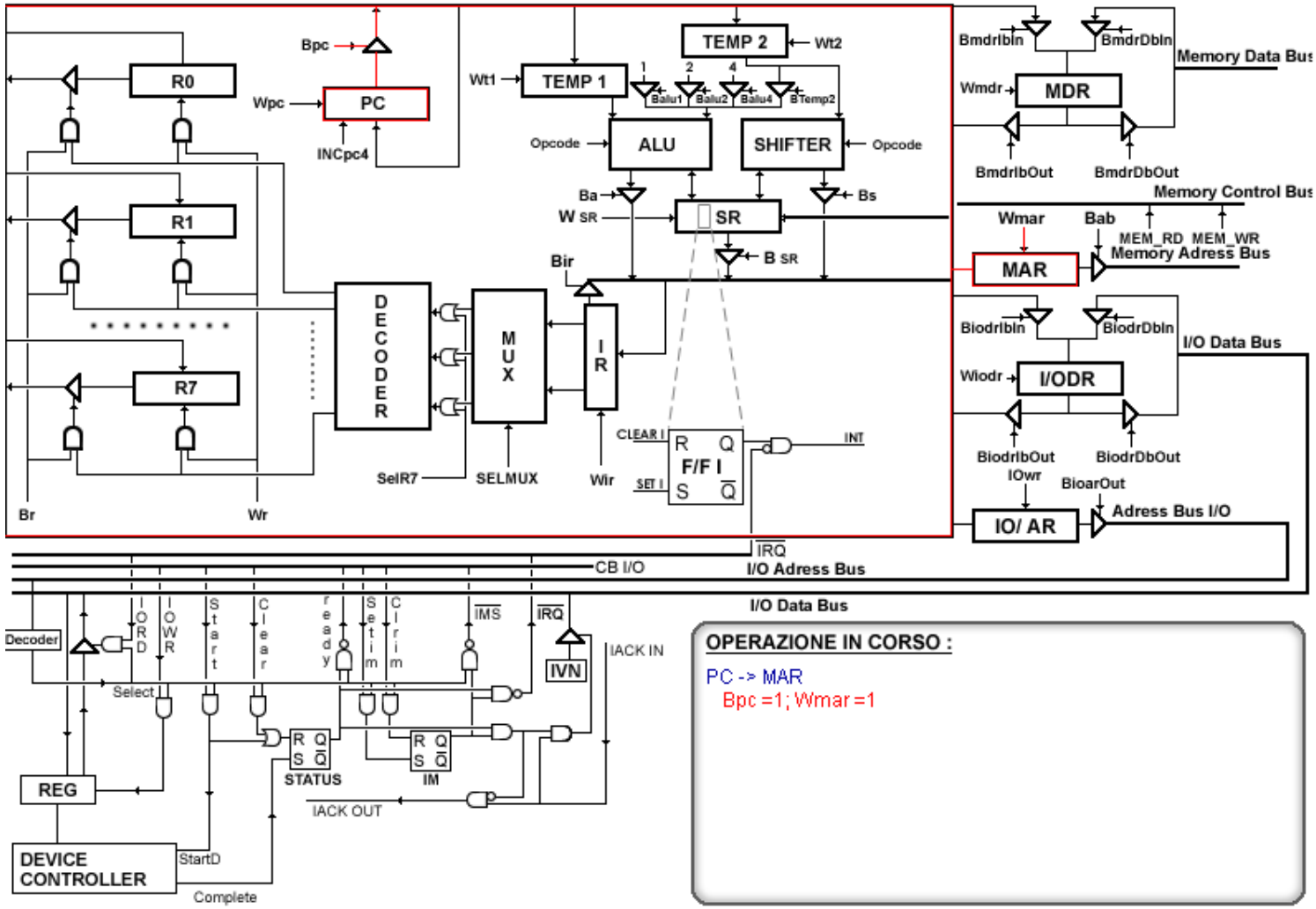


La fase di Fetch

- Ogni operazione incomincia con la fase di fetch
- Le microoperazioni associate alla fase di fetch sono:
  - $\texttt{PC} \rightarrow \texttt{MAR}$
  - $(\texttt{MAR}) \rightarrow \texttt{MDR}; \texttt{PC} + 4 \rightarrow \texttt{PC}$
  - $\texttt{MDR} \rightarrow \texttt{IR}$

- In questo modo, l'istruzione successiva viene caricata nel registro IR (così da poterla interpretare ed eseguire) e il valore di PC viene incrementato (così da puntare alla prossima istruzione/dato)

La fase di Fetch

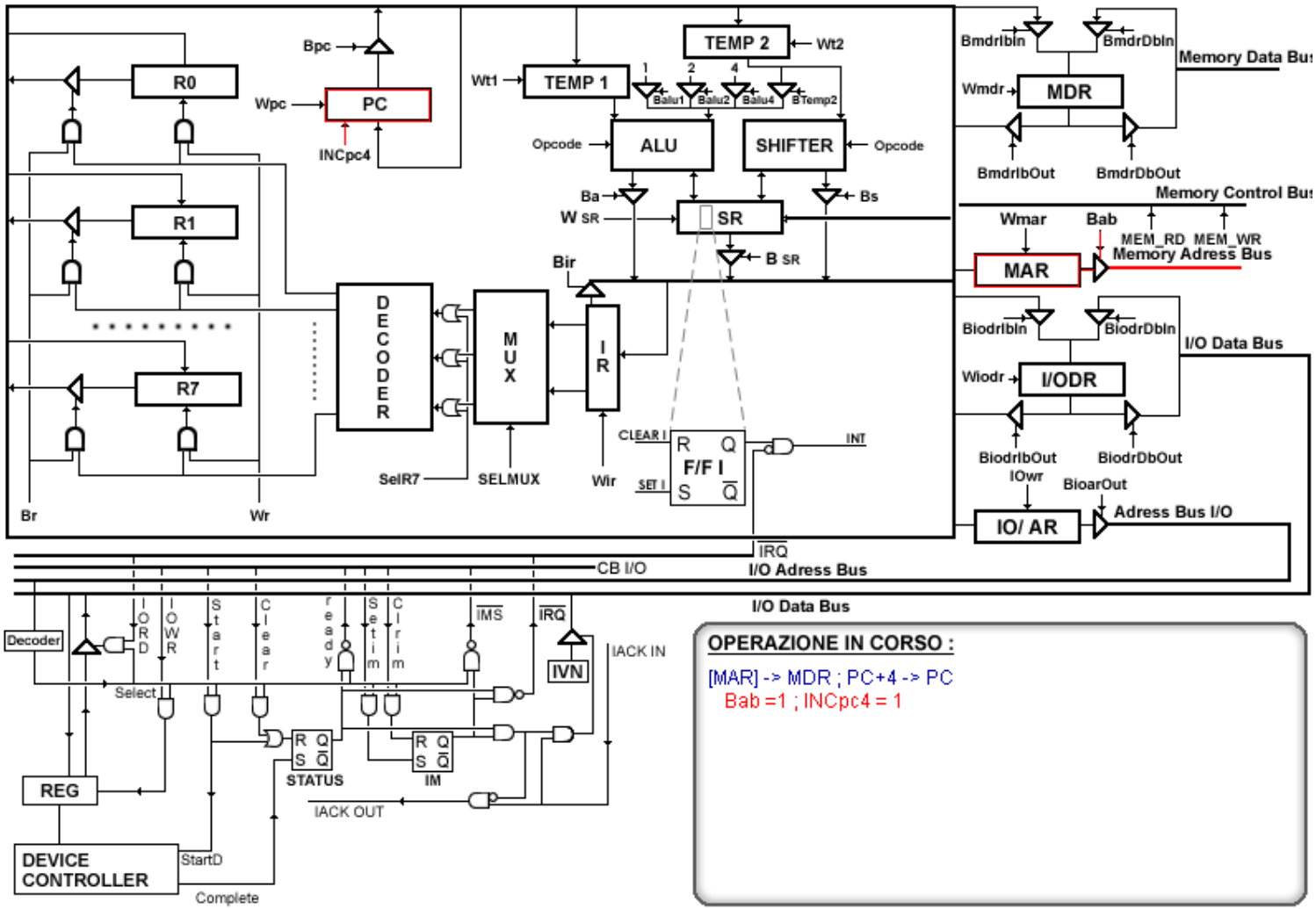


OPERAZIONE IN CORSO :

PC -> MAR

Bpc =1; Wmar =1

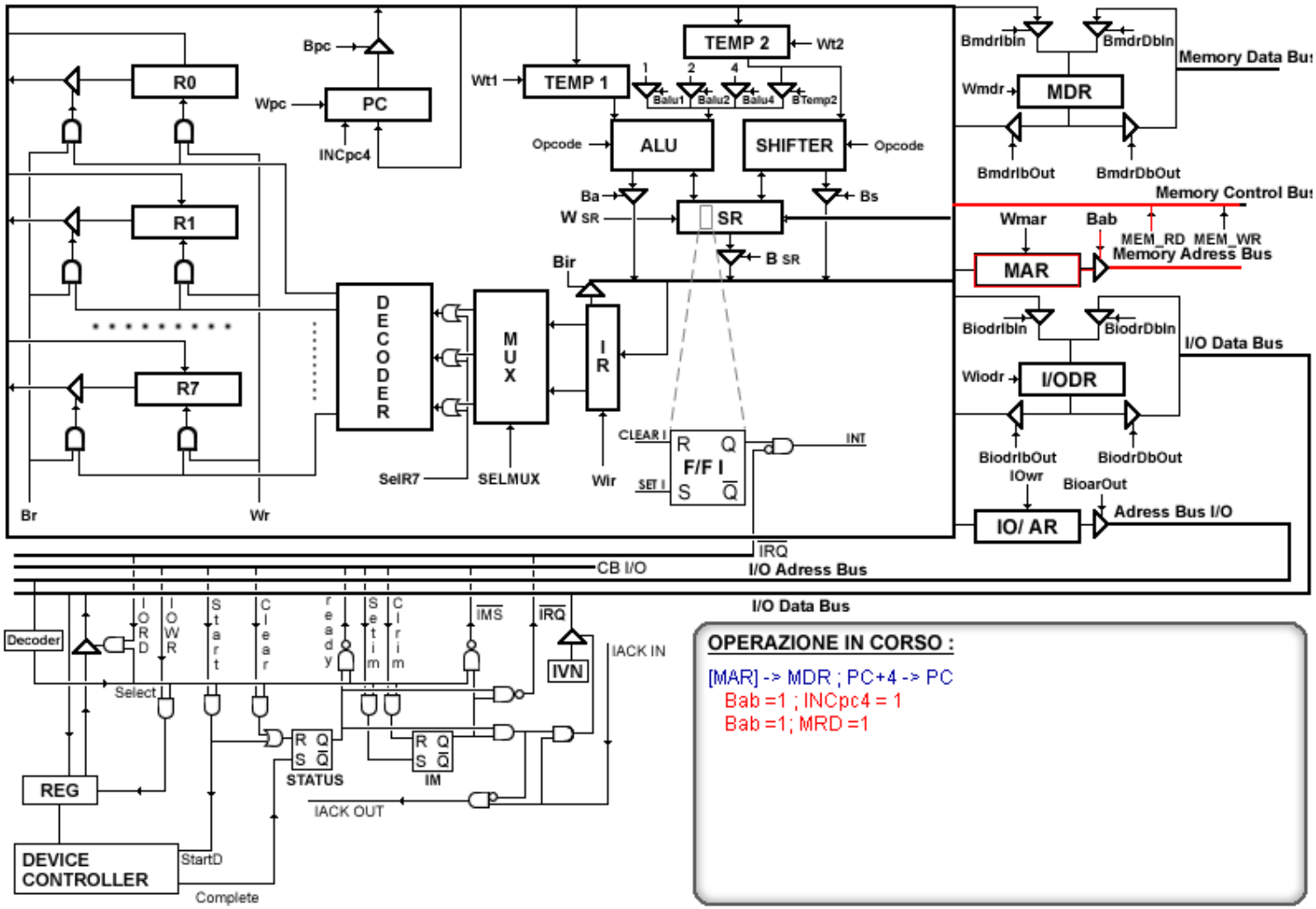
La fase di Fetch



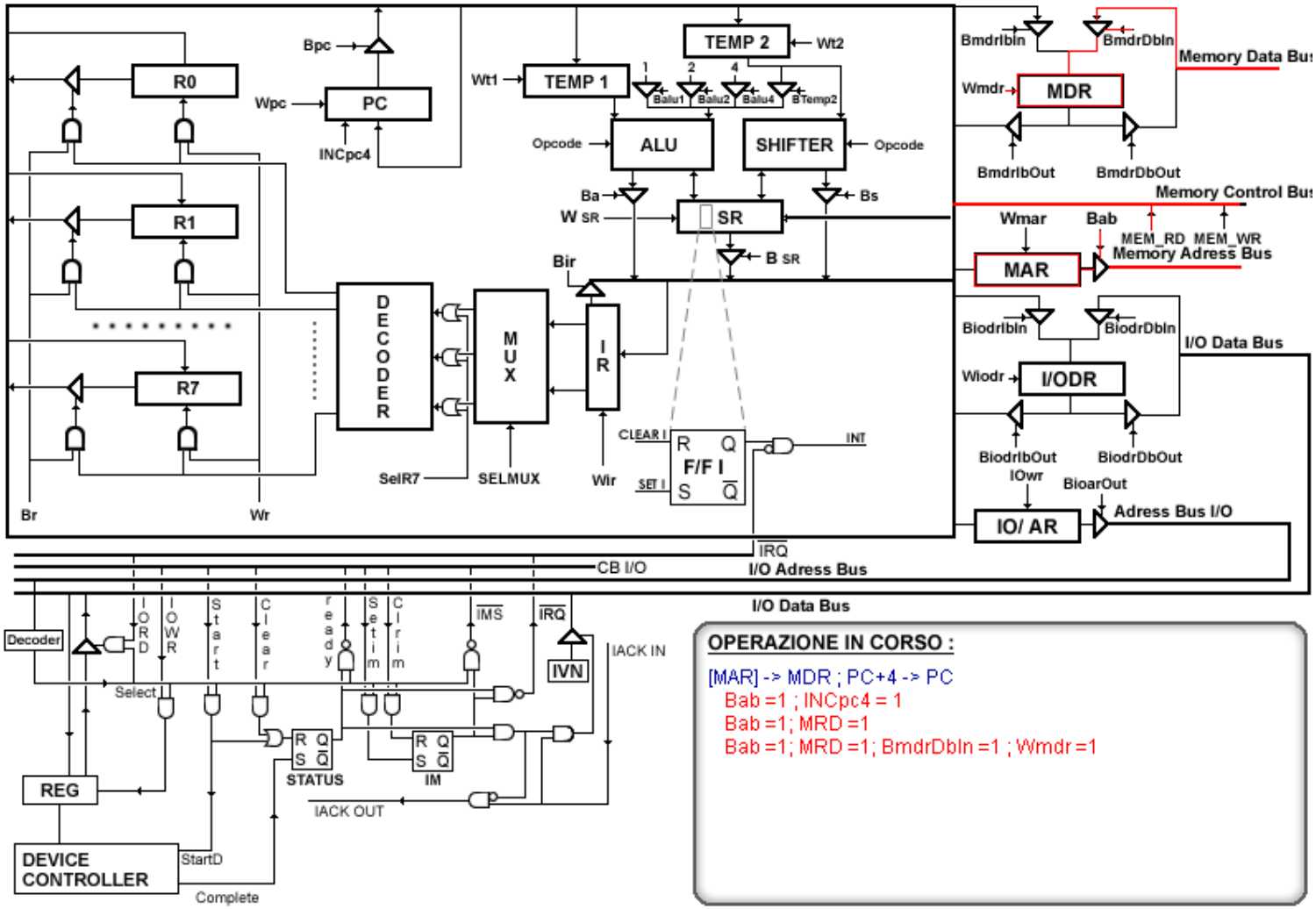
OPERAZIONE IN CORSO :

[MAR] -> MDR ; PC+4 -> PC
Bab =1 ; INCpc4 = 1

La fase di Fetch



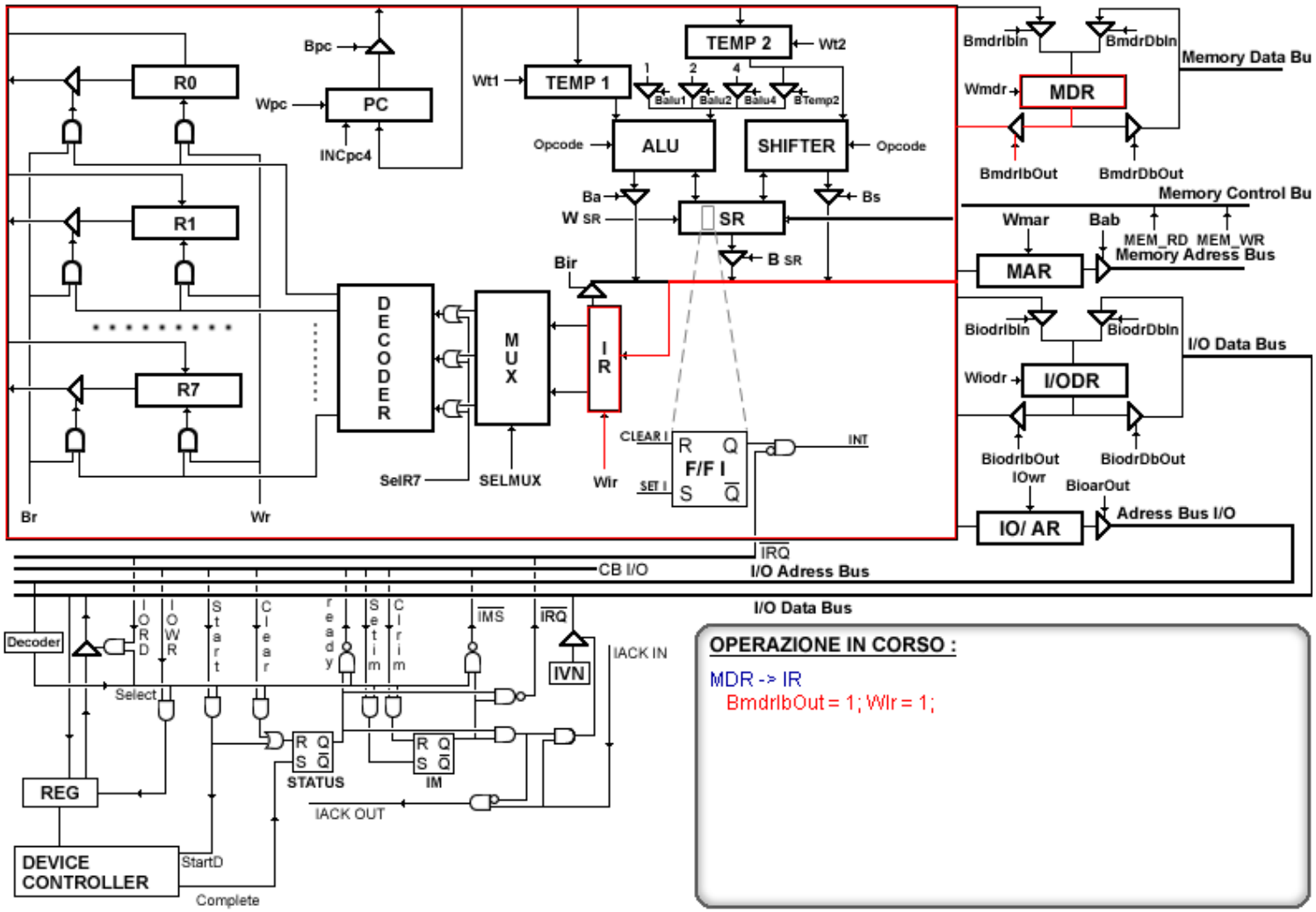
La fase di Fetch



OPERAZIONE IN CORSO :

[MAR] -> MDR ; PC+4 -> PC
Bab =1 ; INCpc4 = 1
Bab =1; MRD =1
Bab =1; MRD =1; BmdrDbIn =1 ; Wmdr =1

La fase di Fetch



OPERAZIONE IN CORSO :

MDR -> IR

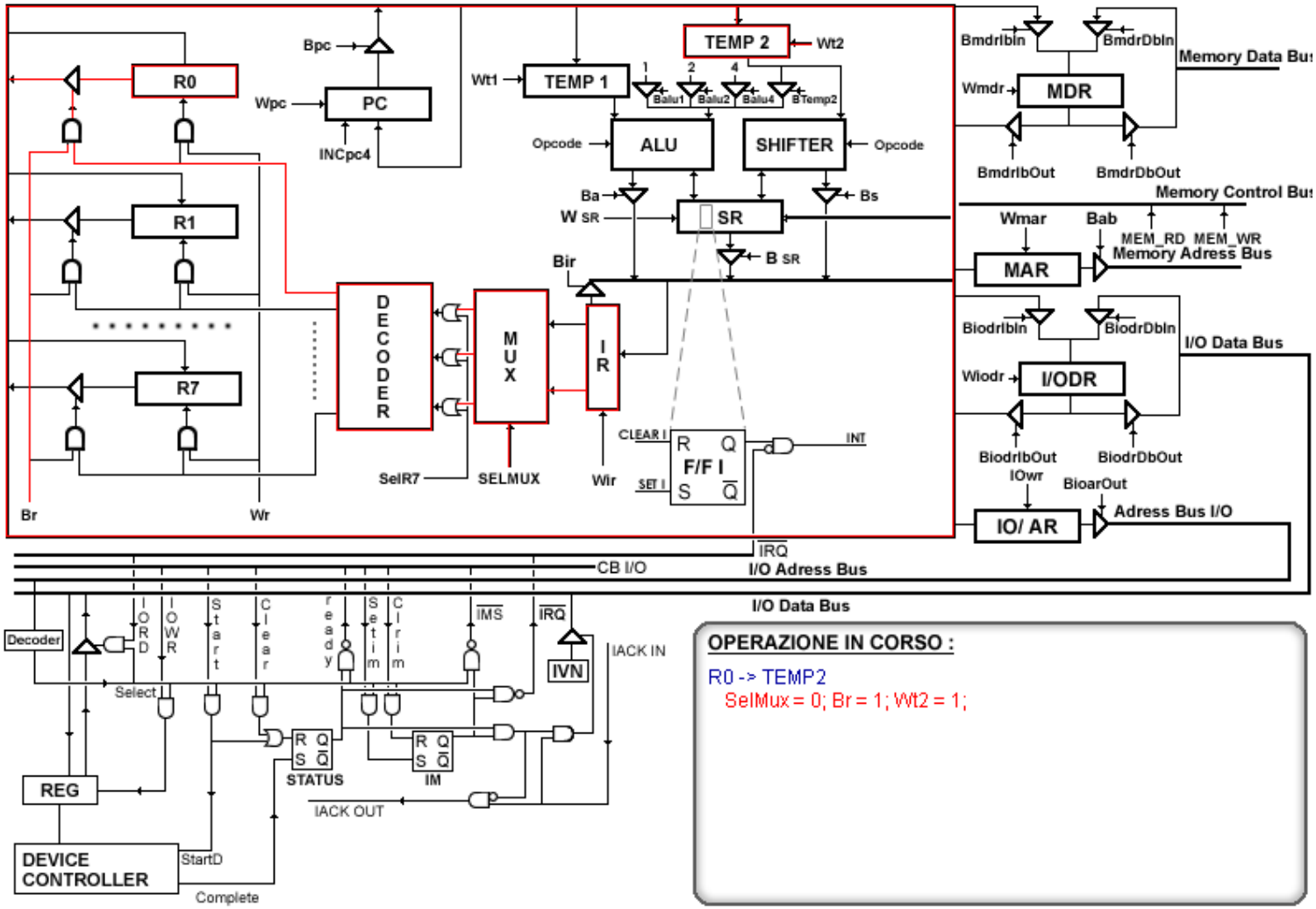
BmdrIbOut = 1; WIr = 1;

Istruzioni di movimento dati

- Le microoperazioni associate al movimento dati dipendono dalla modalità di indirizzamento utilizzato

- `MOV Rx, Ry`:
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR; PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `IR`
  - `Rx` $\rightarrow$ `TEMP2`
  - `TEMP2` $\rightarrow$ `Ry`

Istruzioni di movimento dati

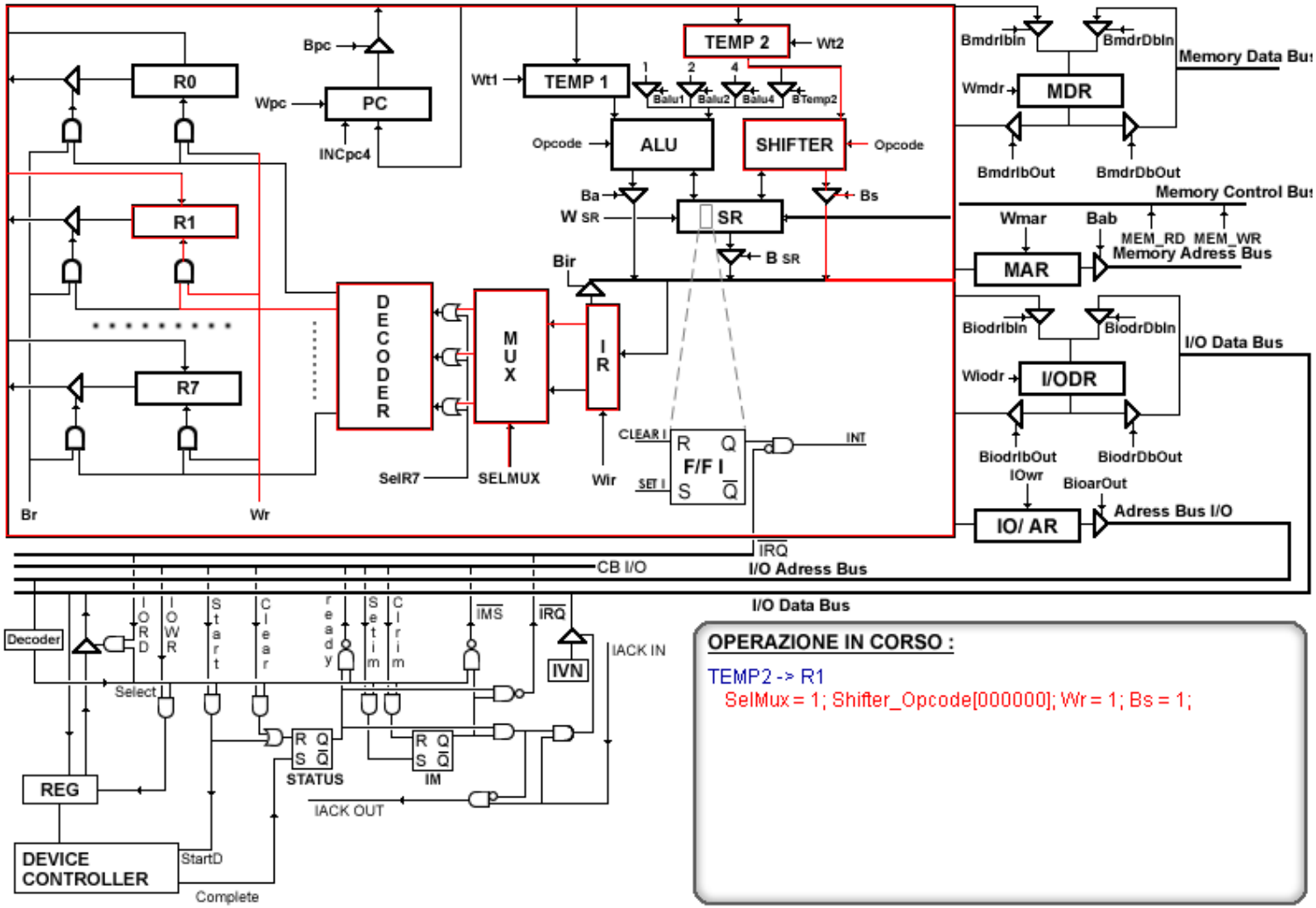


OPERAZIONE IN CORSO :

R0 -> TEMP2

SelMux = 0; Br = 1; Wt2 = 1;

Istruzioni di movimento dati



OPERAZIONE IN CORSO :

TEMP2 -> R1

SelMux = 1; Shifter_Opcode[000000]; Wr = 1; Bs = 1;

Istruzioni di movimento dati

- `MOV #label, Ry`:
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR; PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `IR`
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR`
  - `PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `Ry`

Istruzioni di movimento dati

- `MOV address, Ry`:
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR; PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `IR`
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR`
  - `PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR`
  - `MDR` $\rightarrow$ `Ry`

Istruzioni Aritmetiche e Logiche

- L'esecuzione di una determinata operazione aritmetica o logica dipende dall'opcode passato alla ALU

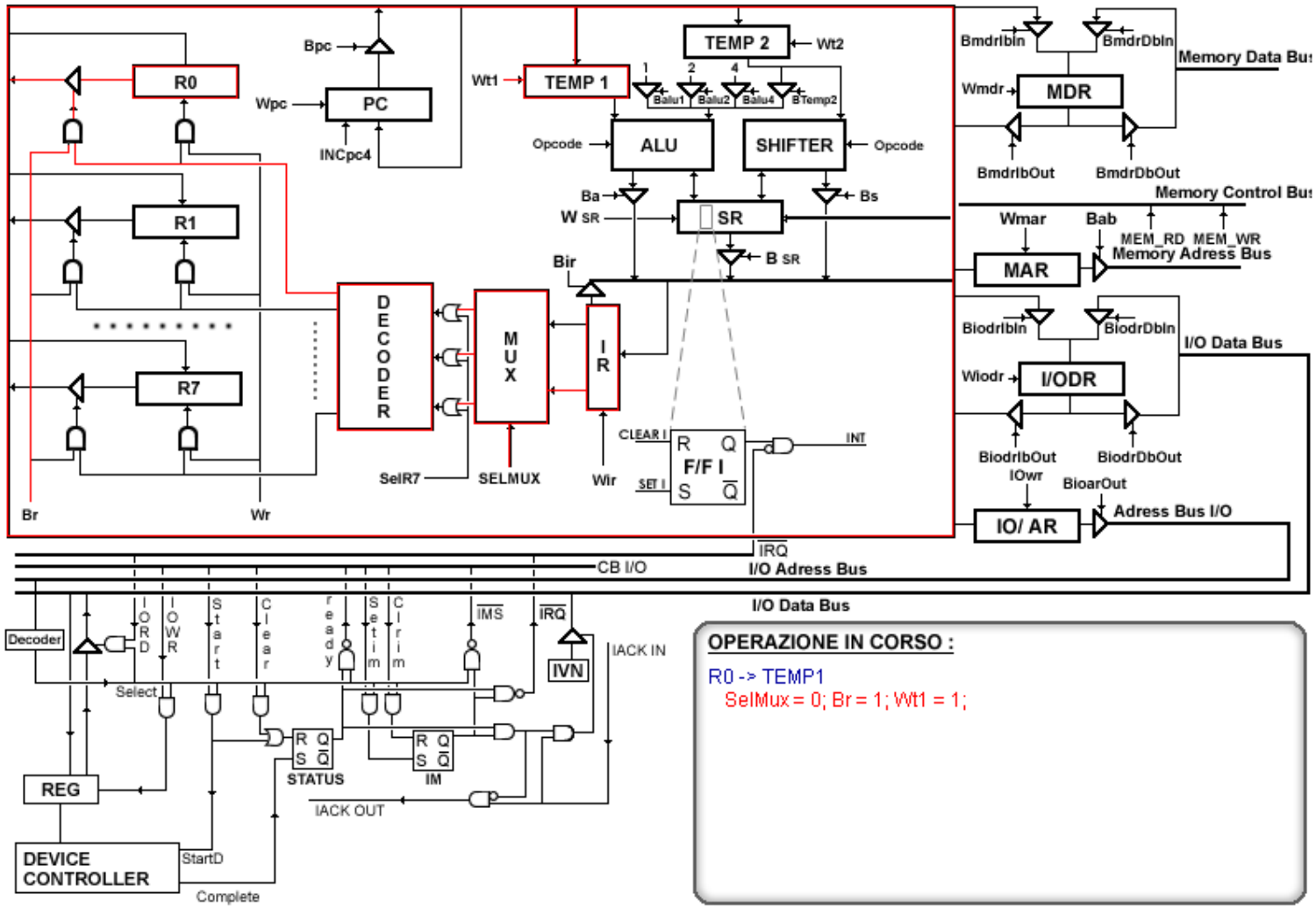
- `ADD Rx, Ry`:
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR; PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `IR`
  - `Rx` $\rightarrow$ `TEMP1`
  - `Ry` $\rightarrow$ `TEMP2`
  - `ALU_OUT[ADD]` $\rightarrow$ `Ry`

Istruzioni Aritmetiche e Logiche

- L'esecuzione di una determinata operazione aritmetica o logica dipende dall'opcode passato alla ALU

- `AND Rx, Ry`:
  - `PC` $\rightarrow$ `MAR`
  - `(MAR)` $\rightarrow$ `MDR; PC + 4` $\rightarrow$ `PC`
  - `MDR` $\rightarrow$ `IR`
  - `Rx` $\rightarrow$ `TEMP1`
  - `Ry` $\rightarrow$ `TEMP2`
  - `ALU_OUT[AND]` $\rightarrow$ `Ry`

Istruzione di Somma

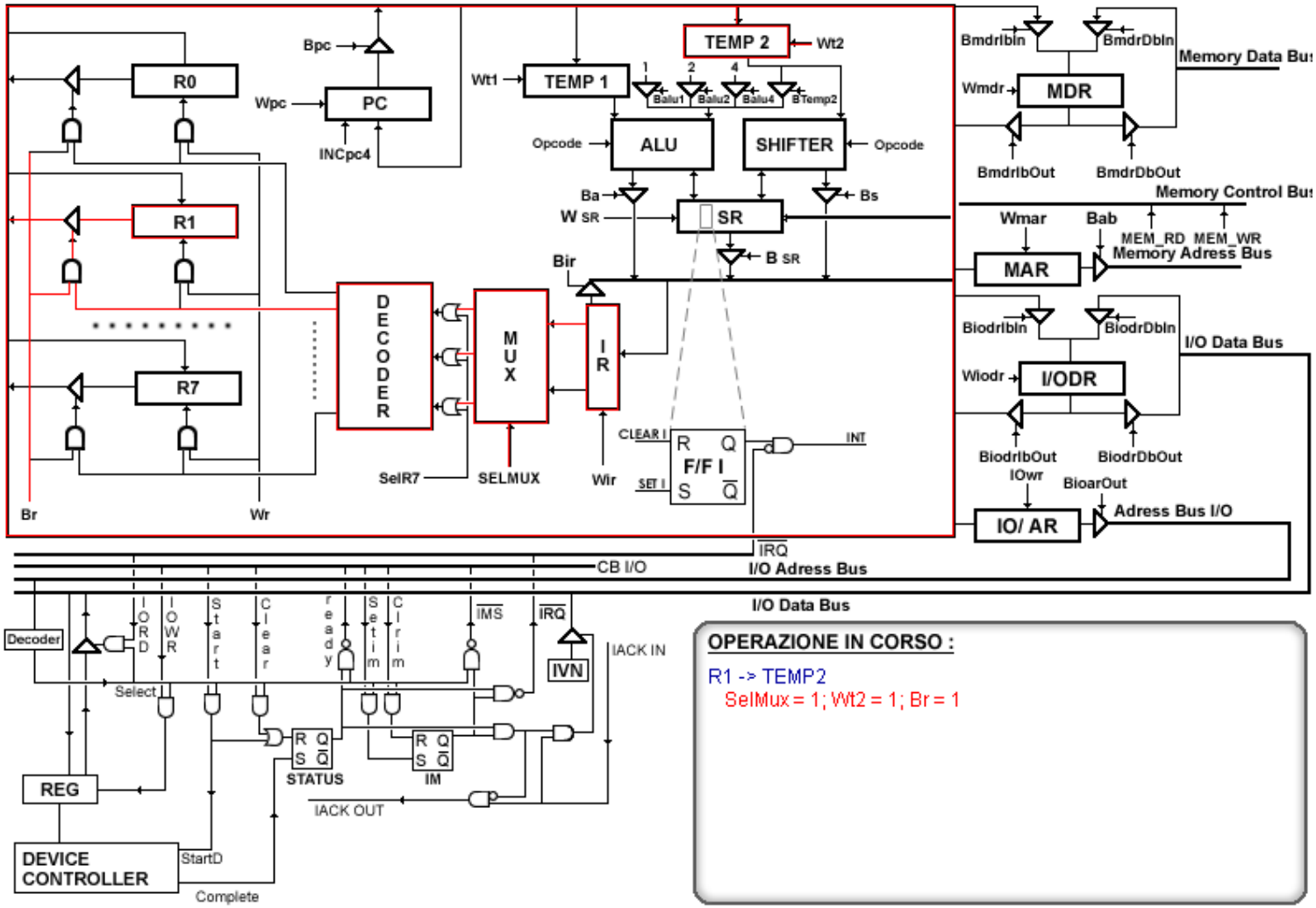


OPERAZIONE IN CORSO :

R0 -> TEMP1

SelMux = 0; Br = 1; Wt1 = 1;

Istruzione di Somma

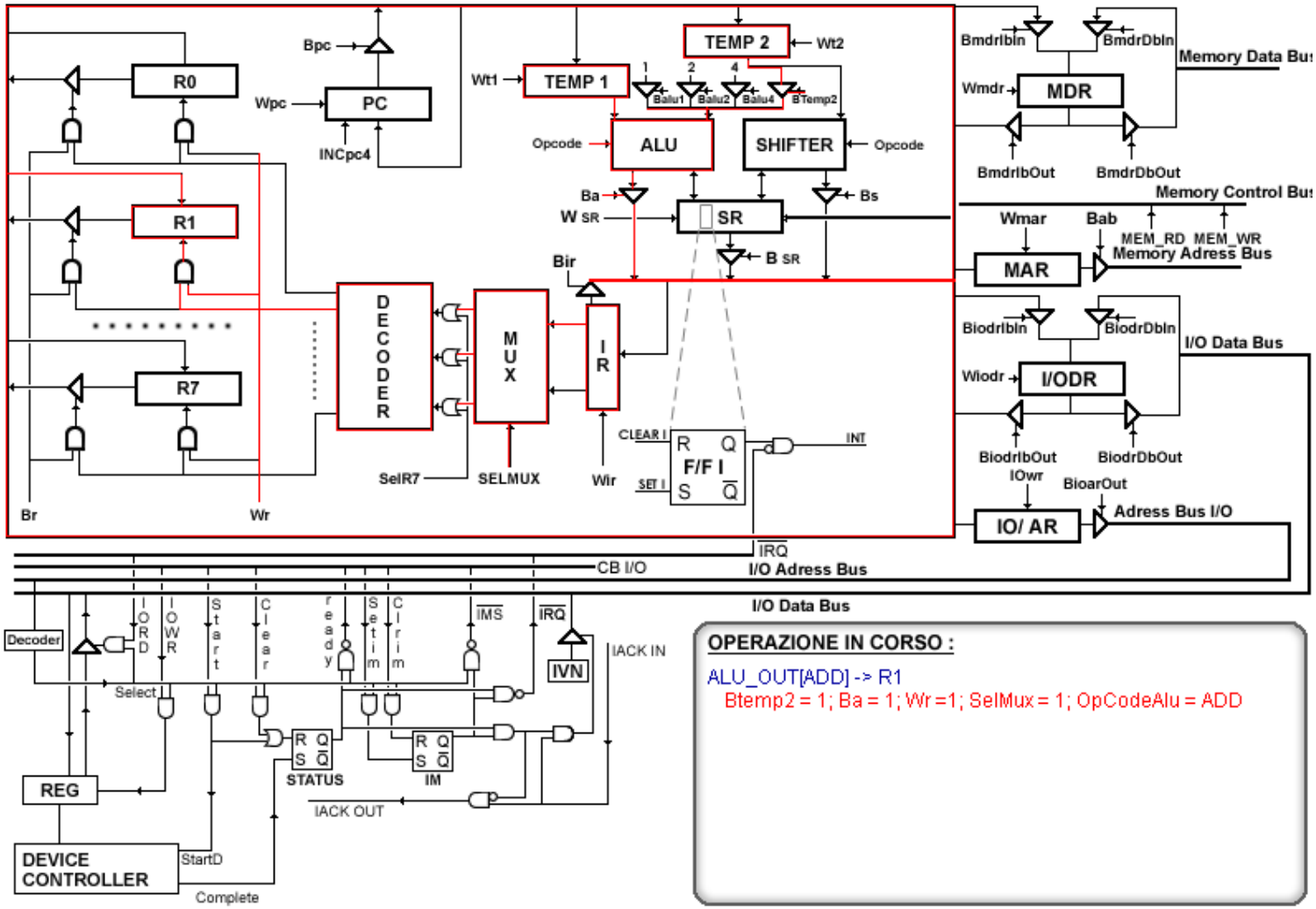


OPERAZIONE IN CORSO :

R1 -> TEMP2

SelMux = 1; Wt2 = 1; Br = 1

Istruzione di Somma

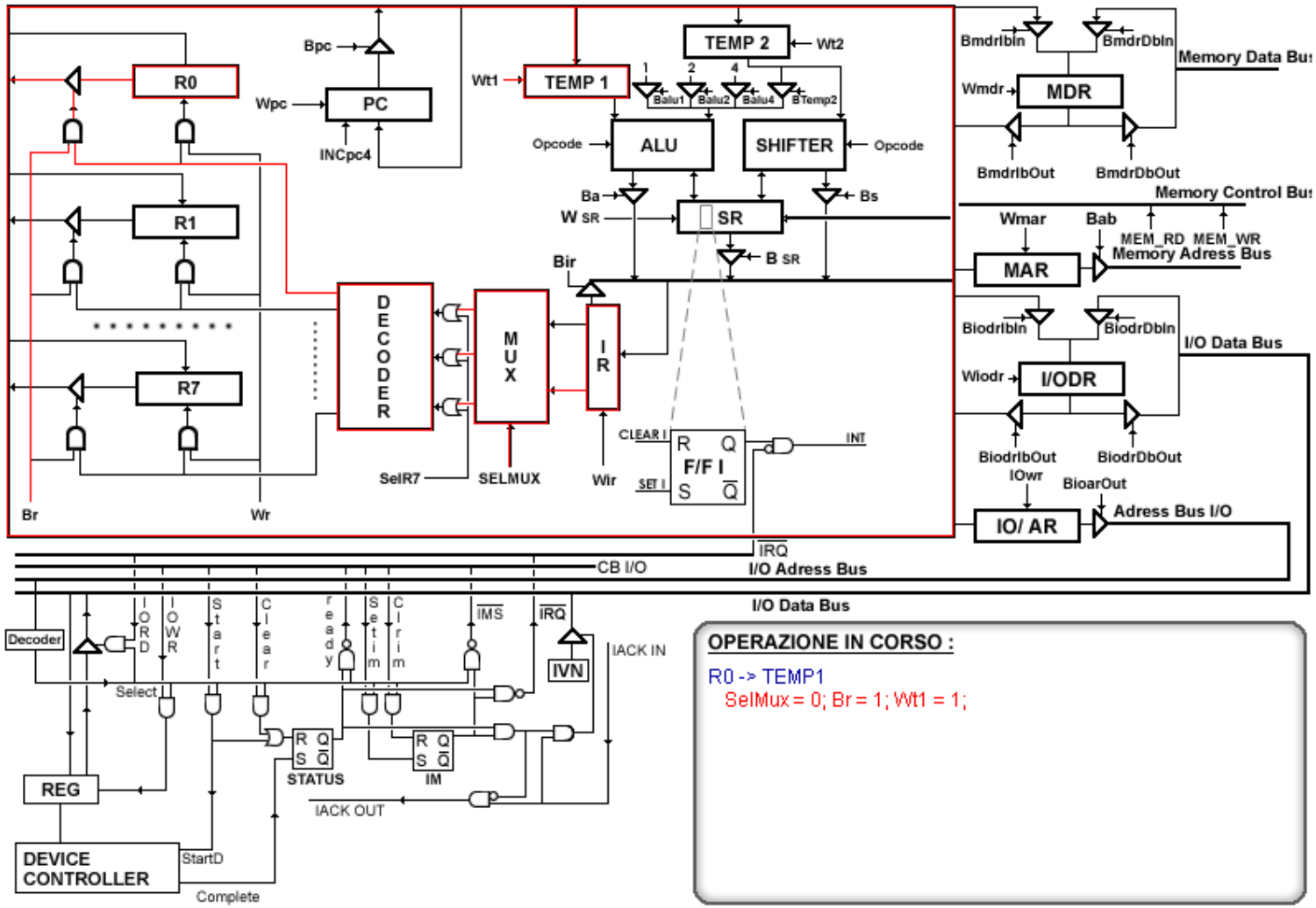


OPERAZIONE IN CORSO :

ALU_OUT[ADD] -> R1

Btemp2 = 1; Ba = 1; Wr =1; SelMux = 1; OpCodeAlu = ADD

Istruzione di And logico

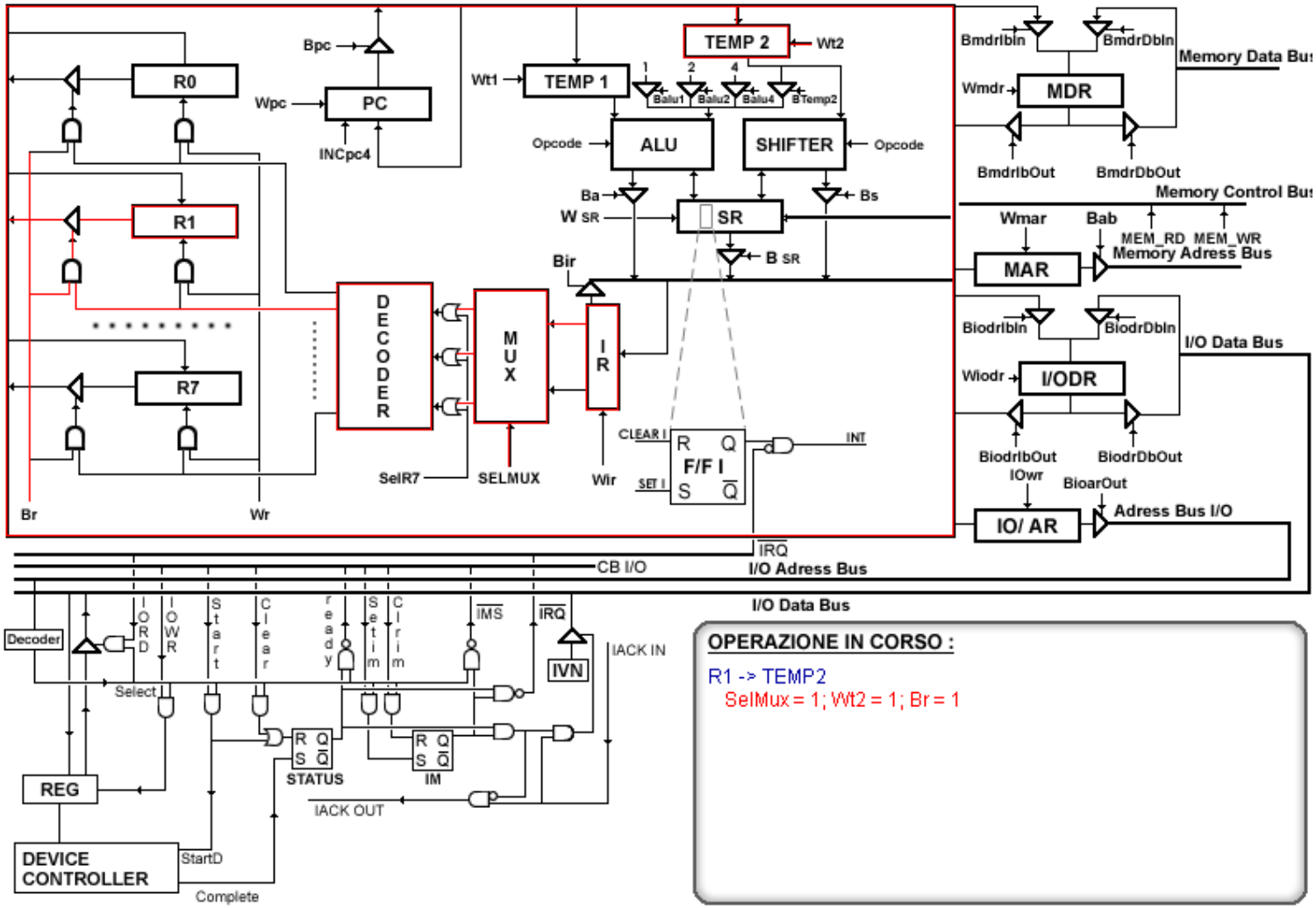


OPERAZIONE IN CORSO :

R0 -> TEMP1

SelMux = 0; Br = 1; Wt1 = 1;

Istruzione di And logico

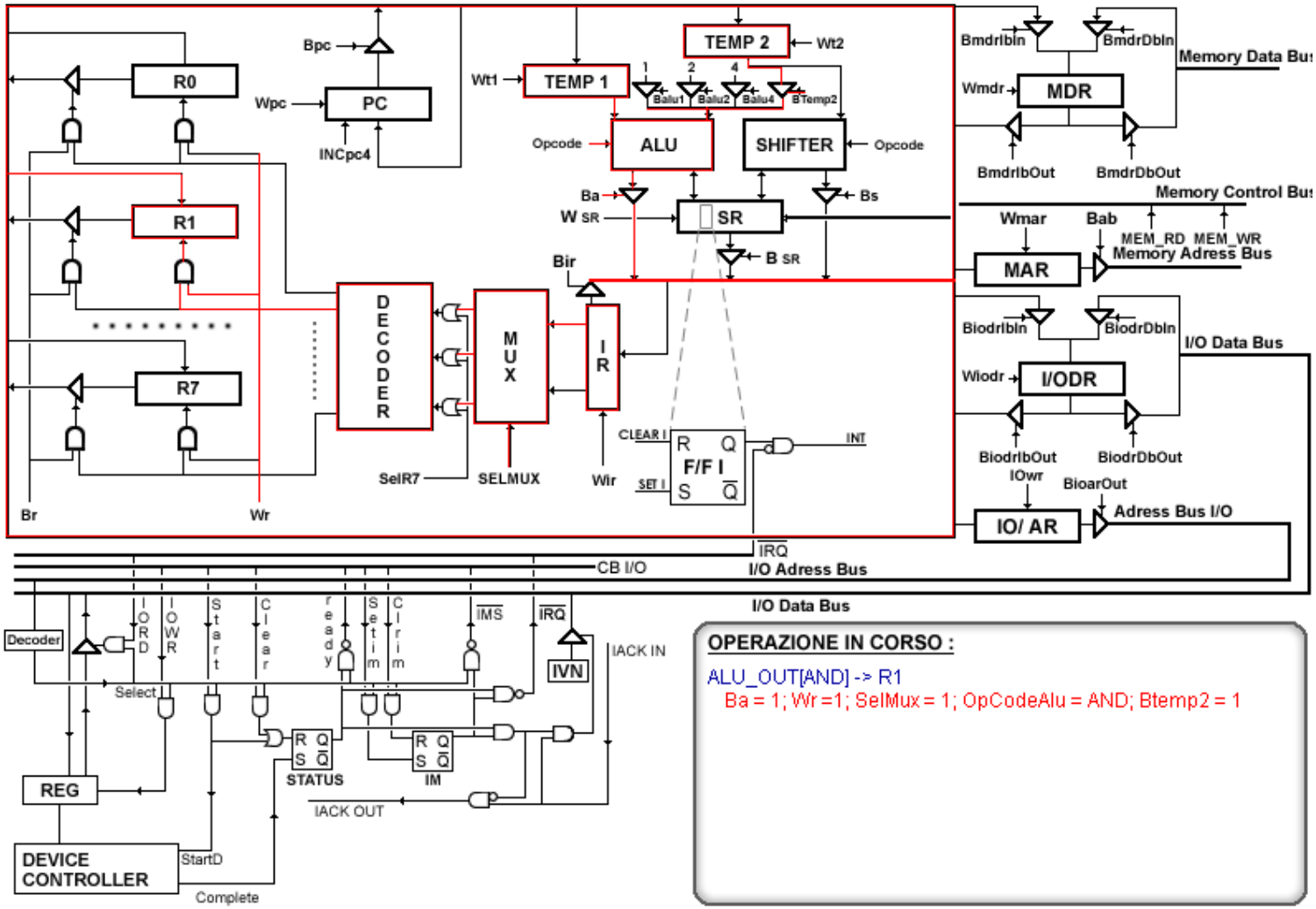


OPERAZIONE IN CORSO :

R1 -> TEMP2

SelMux = 1; Wt2 = 1; Br = 1

Istruzione di And logico



OPERAZIONE IN CORSO :

ALU_OUT[AND] -> R1

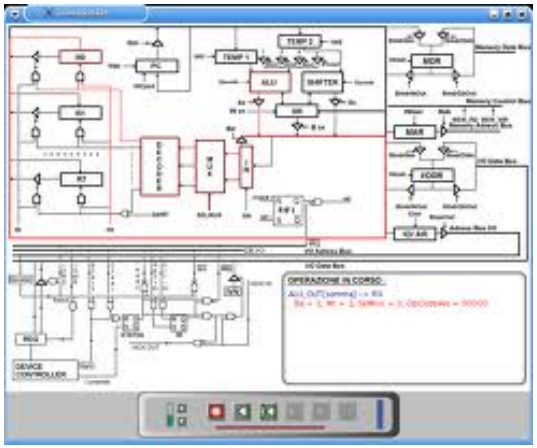
Ba = 1; Wr =1; SelMux = 1; OpCodeAlu = AND; Btemp2 = 1

Istruzioni di controllo

- JZ ADDRESS:
  - PC $\rightarrow$ MAR
  - (MAR) $\rightarrow$ MDR; PC + 4 $\rightarrow$ PC
  - MDR $\rightarrow$ IR
  - IF SR[Z] == 1 THEN
  - PC $\rightarrow$ MAR
  - (MAR) $\rightarrow$ MDR
  - MDR $\rightarrow$ PC
  - ELSE
  - PC + 4 $\rightarrow$ PC

Generatore di Microoperazioni

Per generare le microoperazioni associate a tutto l'instruction set del PD32, si può utilizzare MicroOpGen:



`http://www.dis.uniroma1.it/~ciciani/microopgen/`