



La Sapienza

Università degli Studi di Roma

Dipartimento di Informatica e Sistemistica

CALCOLATORI ELETTRONICI

CPU: Istruzioni - Indirizzamento

Emiliano Trevisani

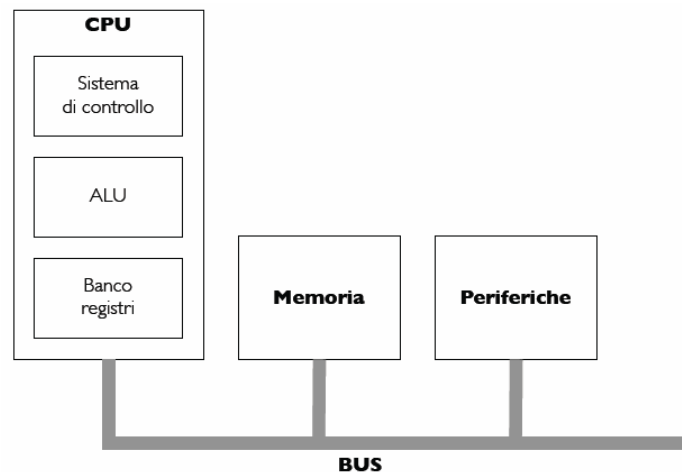
trevisani@dis.uniroma1.it

A.A. 2007/2008

CPU: Istruzioni - Indirizzamento



- ❑ Con riferimento al modello di Von Neumann già discusso:
 - Un programma è costituito da istruzioni e dati, entrambi in memoria centrale
 - L'esecuzione delle istruzioni del programma avviene in modo sequenziale:
 - Eseguire un'istruzione equivale a completare i seguenti 3 task seriali [ciclo istruzione FDE] :
 - **Fetch**: l'istruzione viene prelevata dalla memoria
 - **Decode**: l'istruzione viene interpretata ed i relativi dati preparati
 - **Execute**: l'istruzione viene eseguita sui dati predisposti nella fase di decode; il risultato (se previsto), viene scritto in memoria.
 - Terminato un ciclo FDE, si passa all'istruzione successiva.



CPU: Istruzioni - Indirizzamento



- ❑ Ogni CPU offre un **instruction set** ossia l'insieme di istruzioni supportate
- ❑ In genere le istruzioni possono essere classificate in:
 - Istruzioni di movimento dati registro ↔ memoria e registro ↔ registro
 - Istruzioni aritmetico / logiche
 - Istruzioni di controllo [della macchina o dell'esecuzione; es: salti]
 - Istruzioni di I/O
- ❑ Ogni istruzione è **codificata** in una stringa di bit:
 - In questo formato è:
 - memorizzabile in memoria
 - interpretabile dall'unità di controllo della CPU che attiva la sequenza di attività elementari (segnali / temporizzazioni) per la sua esecuzione
 - La stringa generalmente definisce:
 - il tipo di istruzione
 - i registri o gli indirizzi di memoria degli operandi
 - il registro o l'indirizzo di memoria destinazione
 - altre info

CPU: Istruzioni - Indirizzamento



- ❑ Esempio: istruzioni codificate a 32 bit

Tipo istruzione	Sorgente1	Sorgente2	Destinazione	Info
-----------------	-----------	-----------	--------------	------

- ❑ Abbiamo già visto come l'esecuzione di una istruzione possa essere implementata tramite:

- **Esecuzione diretta**

- HW dedicato: efficienza e velocità
- Set di istruzioni limitato

- **Interpretazione**

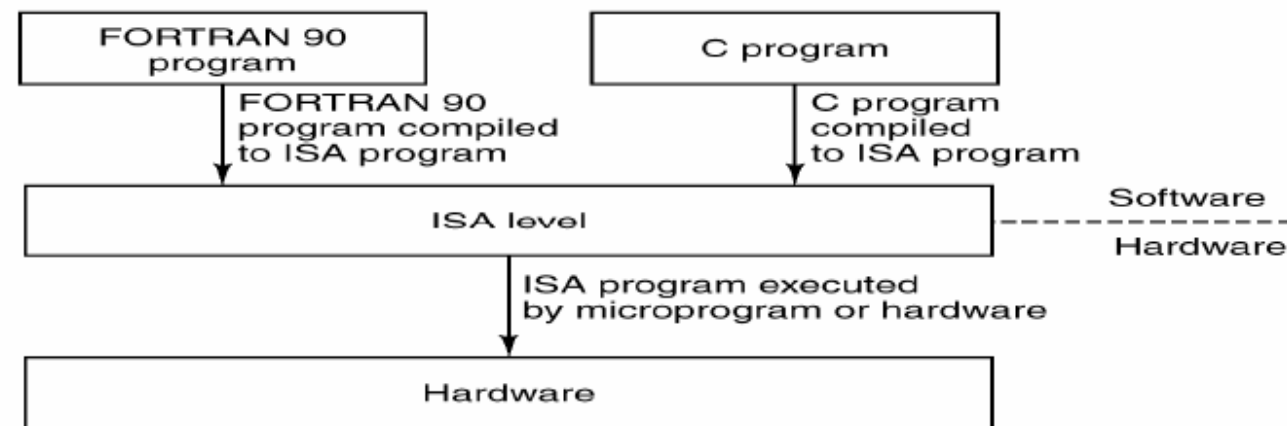
- Interpretazione effettuata dal sottosistema di controllo [**SCO**] della CPU
- Set di istruzioni in grado di offrire un insieme di operazioni elementari
- Istruzioni complesse vengono scomposte in una successione di operazioni elementari [μ -programma]

CPU: Il livello ISA



□ Instruction Set Architecture [ISA]

- Il livello ISA è l'interfaccia tra HW e SW e rappresenta il livello più basso al quale il processore è programmabile
- Criteri implementativi:
 - Semplicità di implementazione
 - Efficienza della microarchitettura
 - Semplicità di generazione del codice
 - Compatibilità con il passato



CPU: Il livello ISA



□ Instruction Set Architecture [ISA]

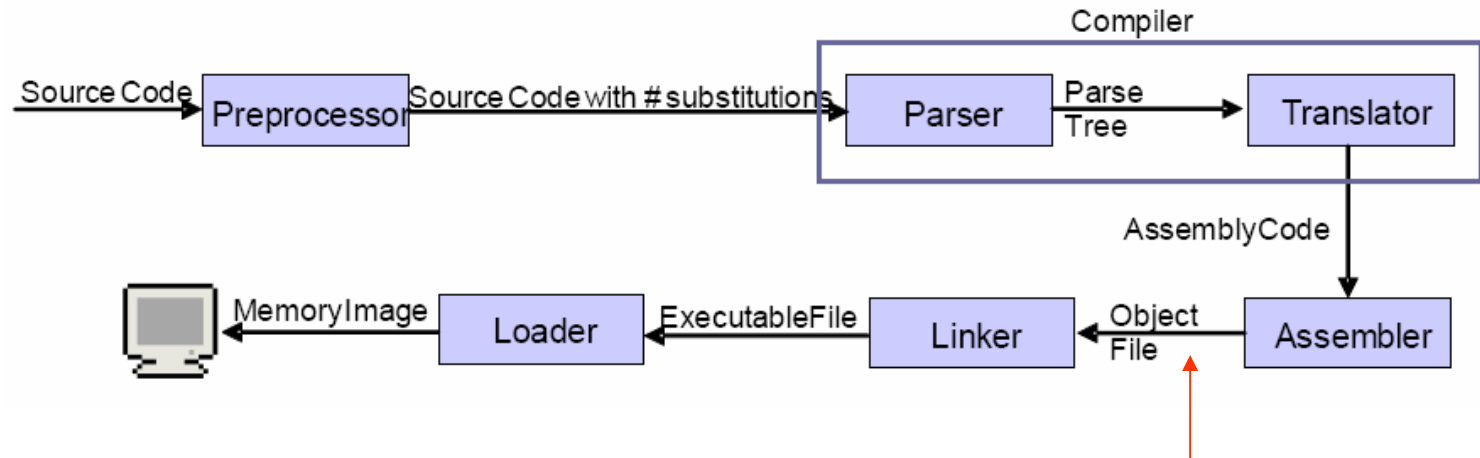
- Caratteristiche principali:
 - **Memoria:** organizzazione e modalità di indirizzamento
 - **Registri:** quali registri sono visibili al livello ISA e quali funzioni hanno
 - **Indirizzamento:** modalità con cui le istruzioni fanno riferimento ai propri operandi
 - **Istruzioni:** repertorio delle istruzioni macchina
 - **Modi di funzionamento:** [esempio: user mode / kernel mode; supporto HW alle funzionalità del Sistema Operativo]



□ Linguaggi di programmazione

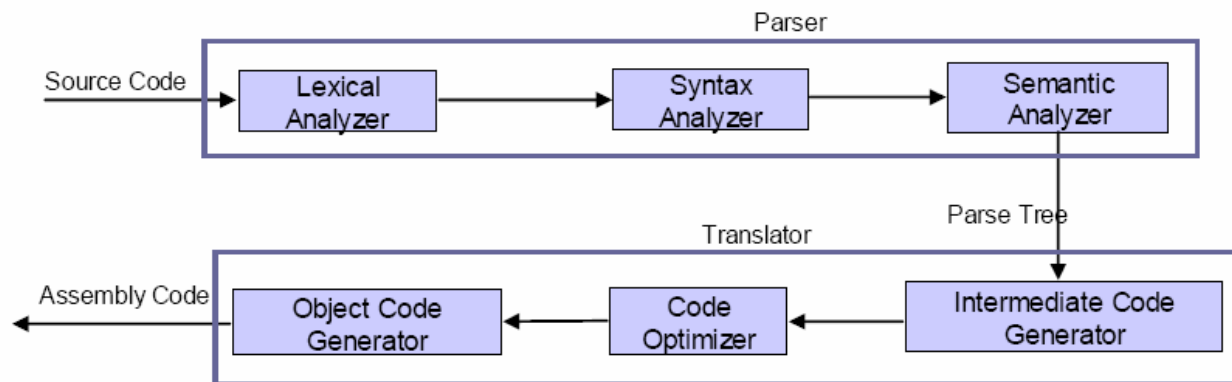
- Classificazione:
 - **M-Code:** linguaggio macchina o Machine Code;
 - stringhe di bit che rappresentano le istruzioni eseguibili dall'HW
 - **Low-Level:** [es. Assembly]
 - Livello di astrazione più elevato rispetto a M-Code
 - Al programmatore è richiesta la conoscenza dell'architettura della macchina
 - Output del compilatore
 - **High Level:** [es. Java, C++,,..]
 - Livello di astrazione più elevato rispetto all'HW
 - Non è richiesta al programmatore la conoscenza dettagliata dell'architettura della macchina
 - Input del compilatore

□ Struttura di un sistema di compilazione

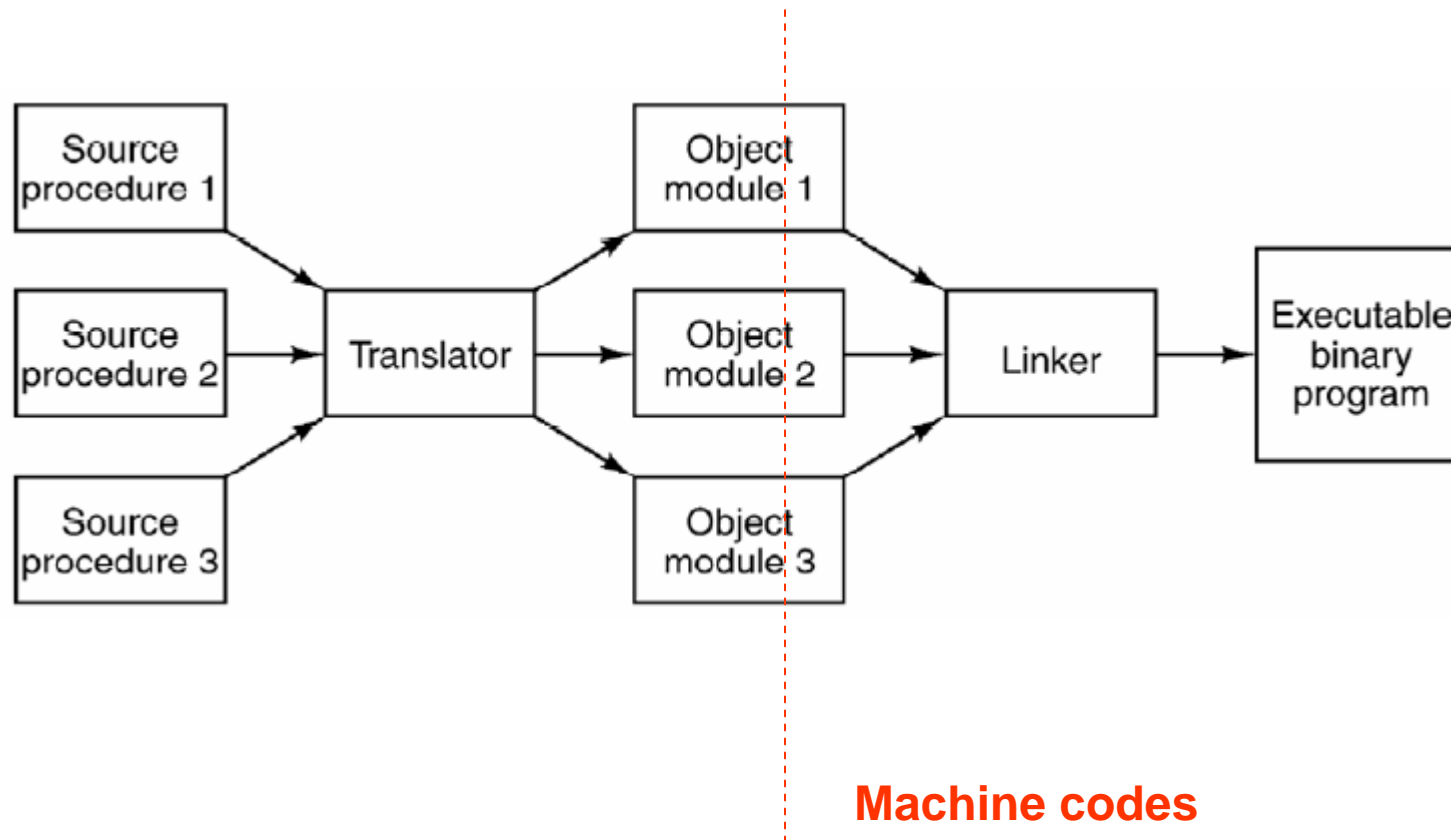


Code Machine

□ Struttura interna di un compilatore



□ Schema generale di linking



CPU: Esempi di cicli FDE

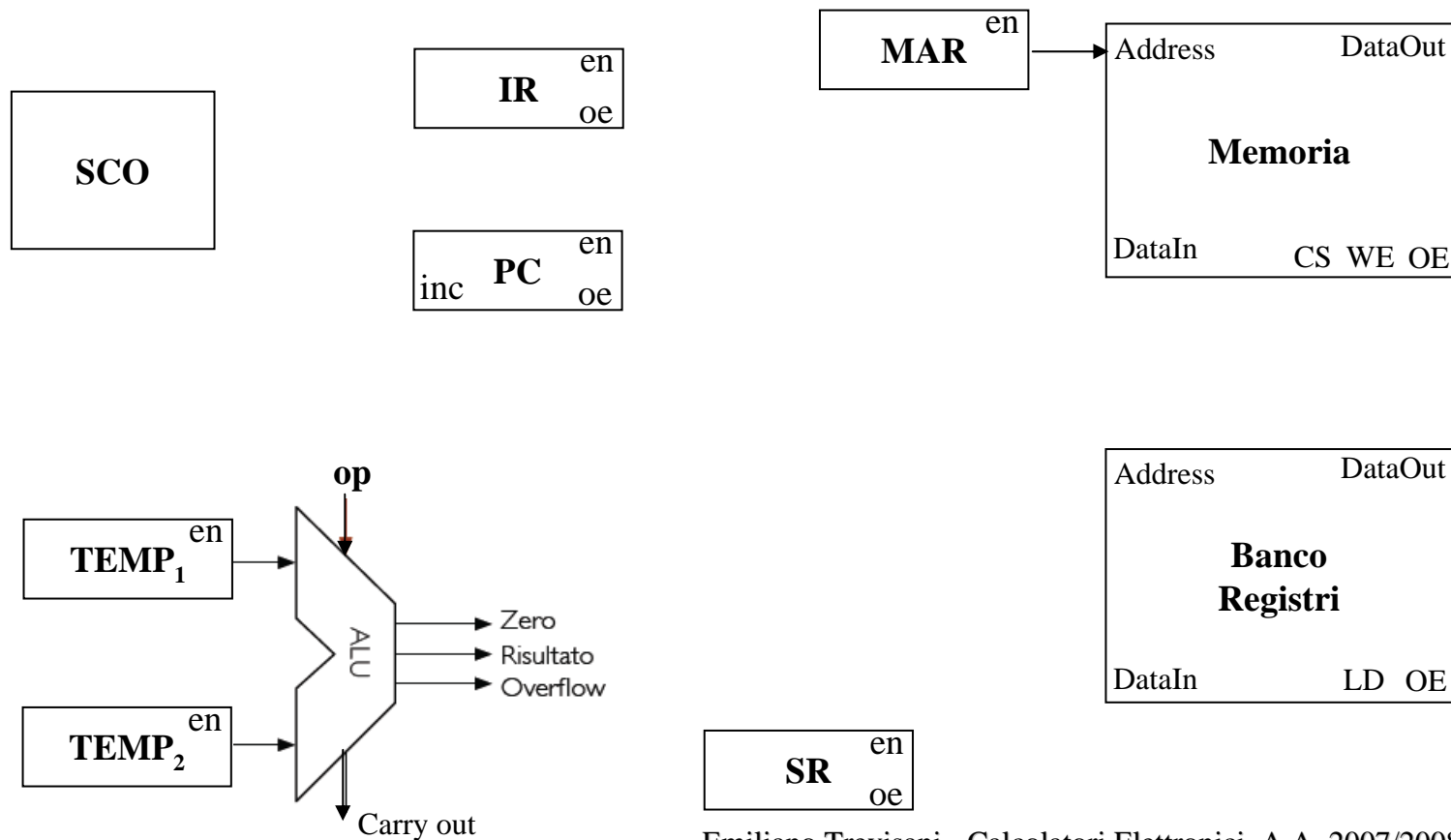


- ❑ Nel seguito saranno illustrati alcuni esempi di cicli istruzione [Fetch-Decode-Execute] per alcune istruzioni “tipiche”
 - ADD [operazione di somma tra 2 dati contenuti in 2 registri – risultato in un registro]
 - LOAD [lettura di un dato in memoria - dato memorizzato in un registro]
 - STORE [scrittura in memoria del dato contenuto in un registro]
 - JMP [salto incondizionato ad una locazione di memoria]
- ❑ Utilizzeremo un’architettura di calcolo semplificata:
 - ALU
 - Instruction Register [IR], Program Counter [PC], Status Register [SR]
 - Banco registri BR [bus indirizzi IR-BR dedicato]
 - Bus unico [dati-indirizzi della memoria] \Rightarrow Registri temporanei [TEMP₁,TEMP₂, MAR]
 - Memoria [lettura / scrittura 2 cicli di clock: indirizzo sul bus \rightarrow MAR, dato sul bus]
 - Sottosistema di controllo [SCO]

CPU: Esempi di cicli FDE



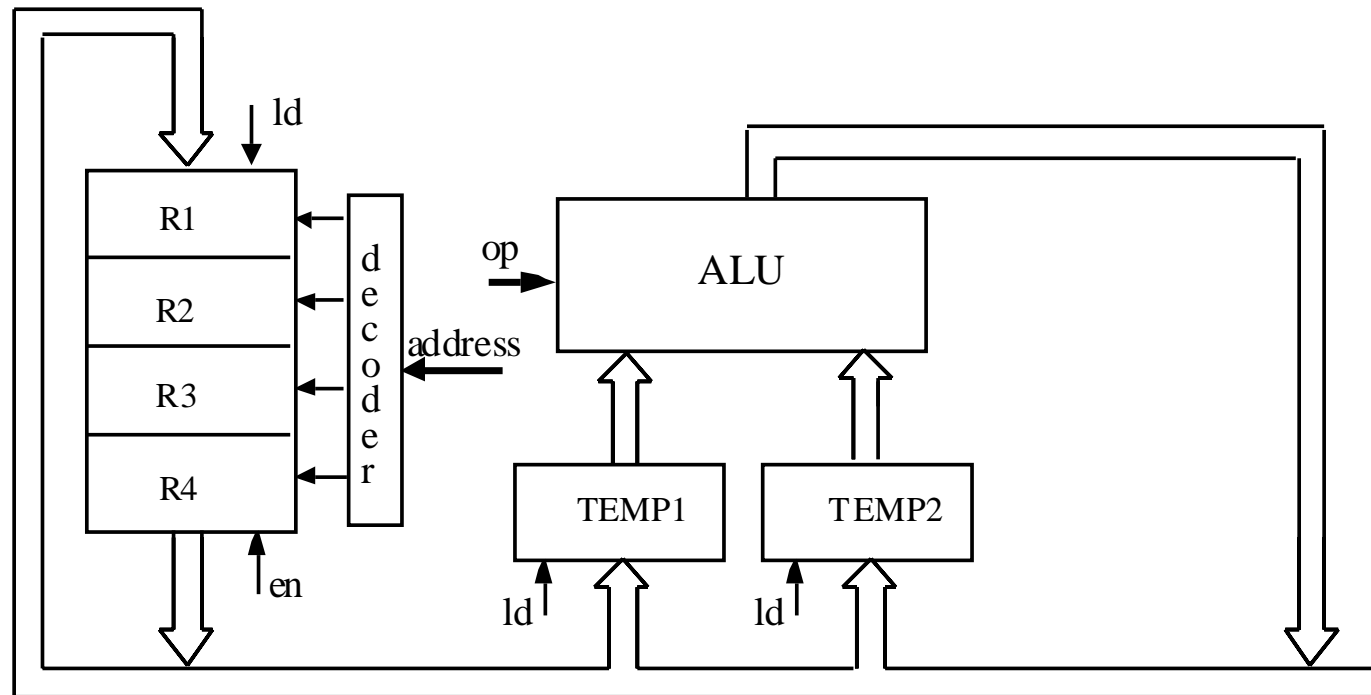
- ❑ Architettura di riferimento: per semplicità non sono riportati il clock e tutti i segnali di controllo; questi ultimi saranno evidenziati caso per caso
- ❑ In **rosso** i segnali attivati da SCO
- ❑ In **blu** il data path logico



CPU: Esempi di cicli FDE



- Architettura di riferimento: il **modello** di interconnessione è quello a bus singolo già descritto con le ovvie estensioni per supportare i registri aggiuntivi e l'interfaccia con la memoria



CPU: Esempi di cicli FDE



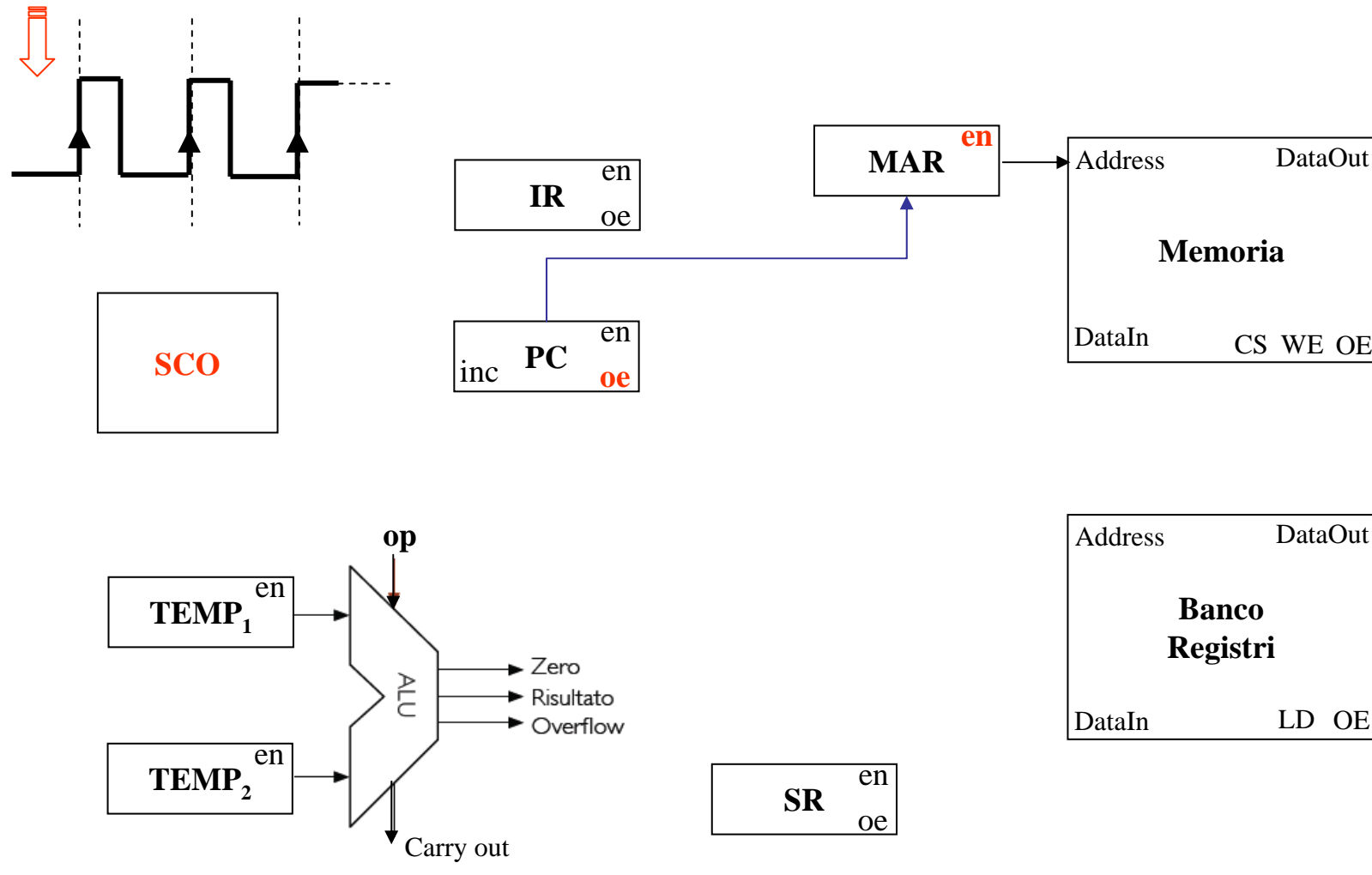
□ Esempi:

- **ADD, #01, #02, #03,:** Somma contenuto dei registri R1 ed R2; risultato in R3
- **LOAD, #0FA0542A,,#04,:** Il dato contenuto nella locazione di memoria 0x0FA0542A viene copiato in R4
- **STORE, #04,,#0FA0542A,, :** Il dato contenuto in R4 viene copiato nella locazione di memoria 0x0FA0542A
- **JMP, #01,,,: Istruzione di salto incondizionato all'istruzione il cui indirizzo è il valore corrente di PC + contenuto di R1**

CPU: Esempi di cicli FDE



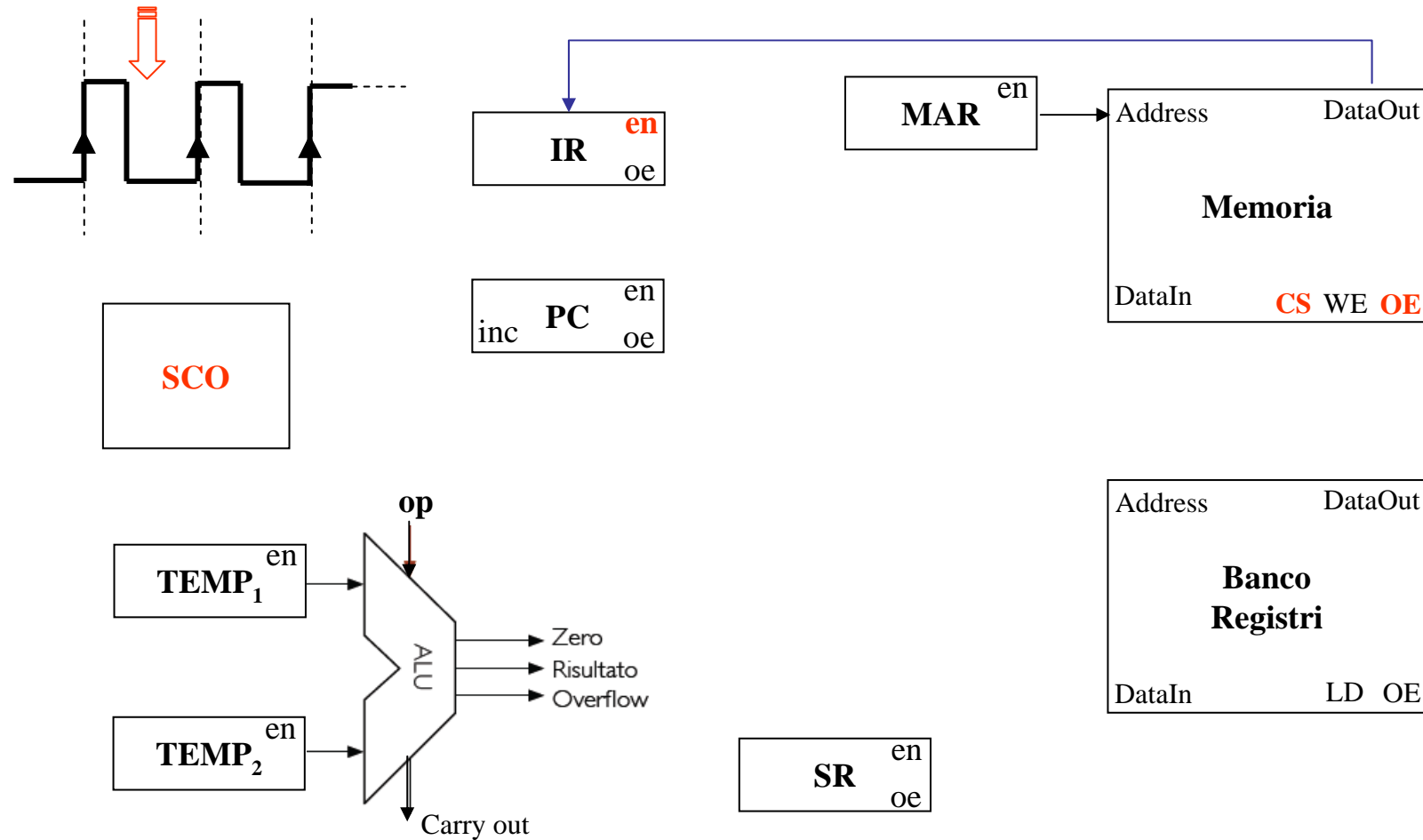
□ ADD, #01, #02, #03: fase di **FETCH** 1/3



CPU: Esempi di cicli FDE



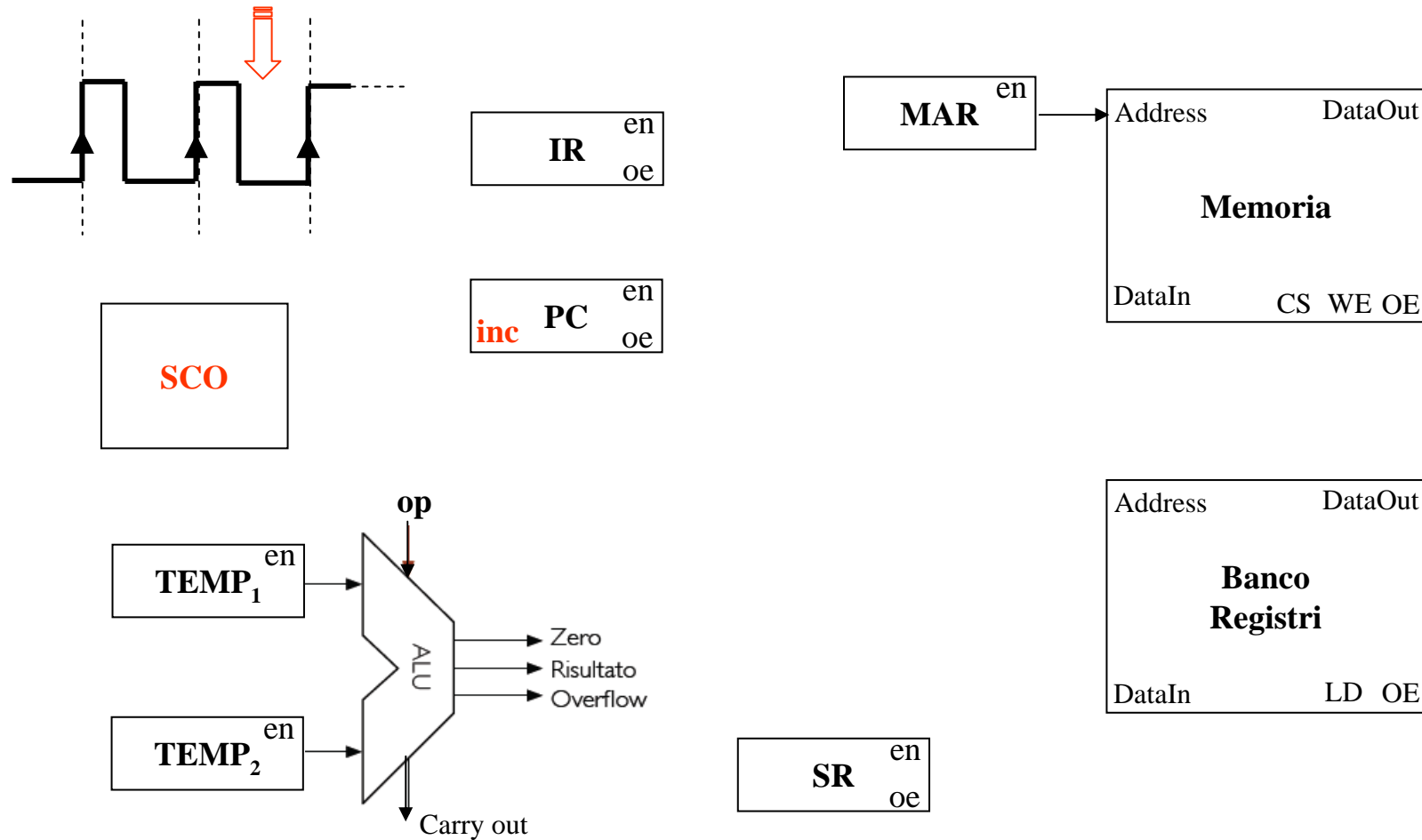
□ ADD, #01, #02, #03: fase di **FETCH 2/3**



CPU: Esempi di cicli FDE



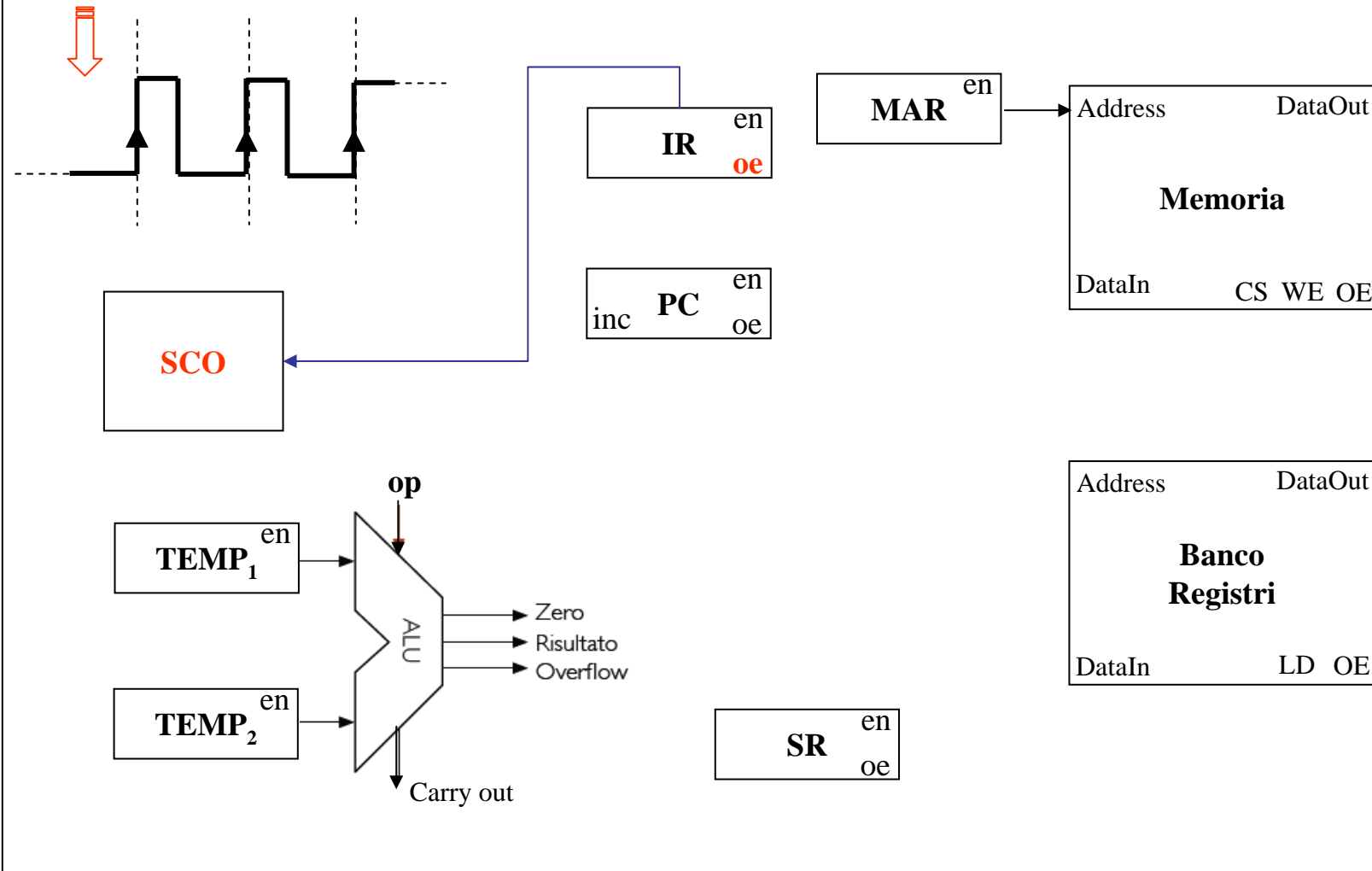
□ ADD, #01, #02, #03: fase di **FETCH** 3/3



CPU: Esempi di cicli FDE



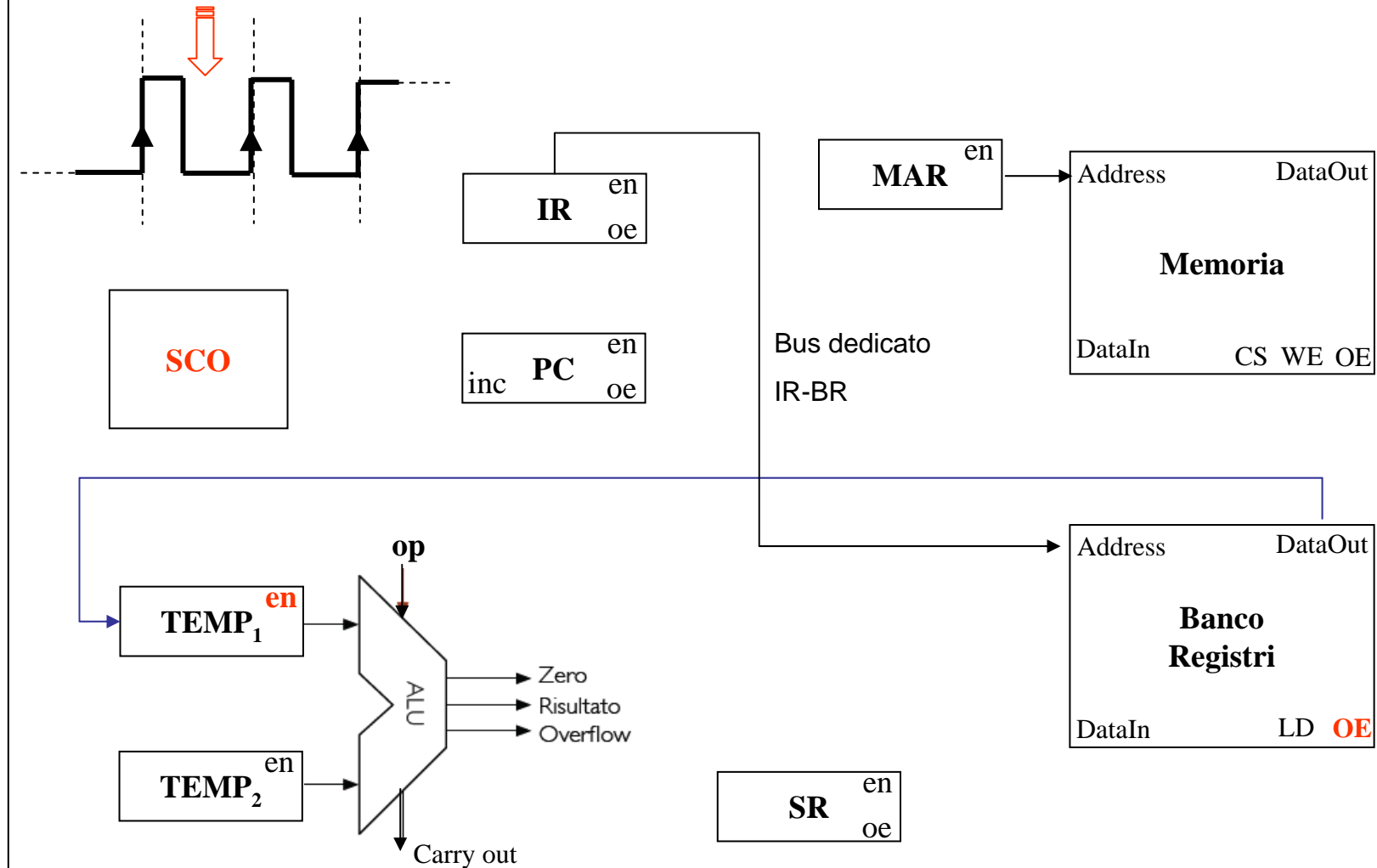
□ ADD, #01, #02, #03: fase di **DECODE 1/3**



CPU: Esempi di cicli FDE



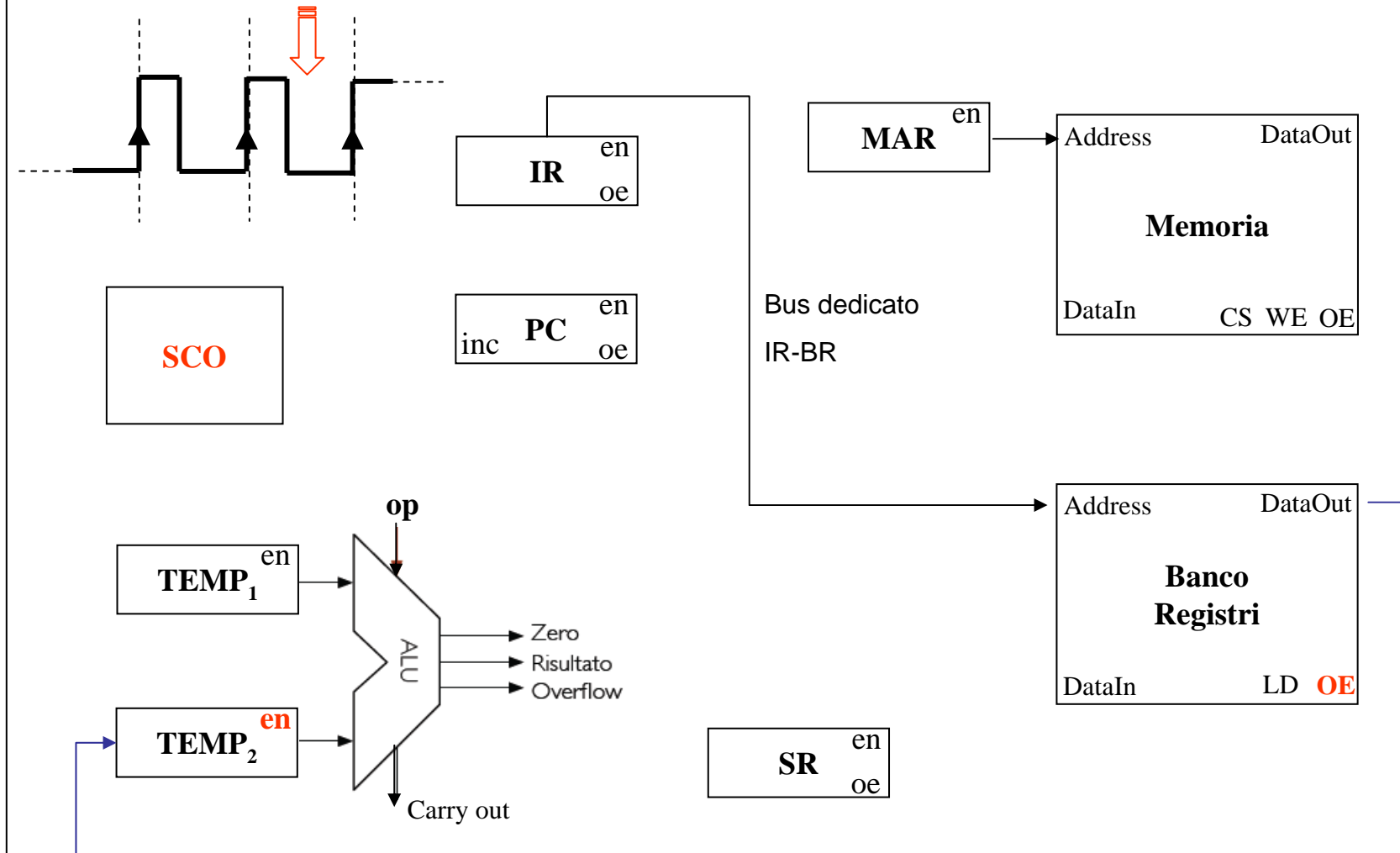
□ ADD, #01, #02, #03: fase di **DECODE 2/3**



CPU: Esempi di cicli FDE



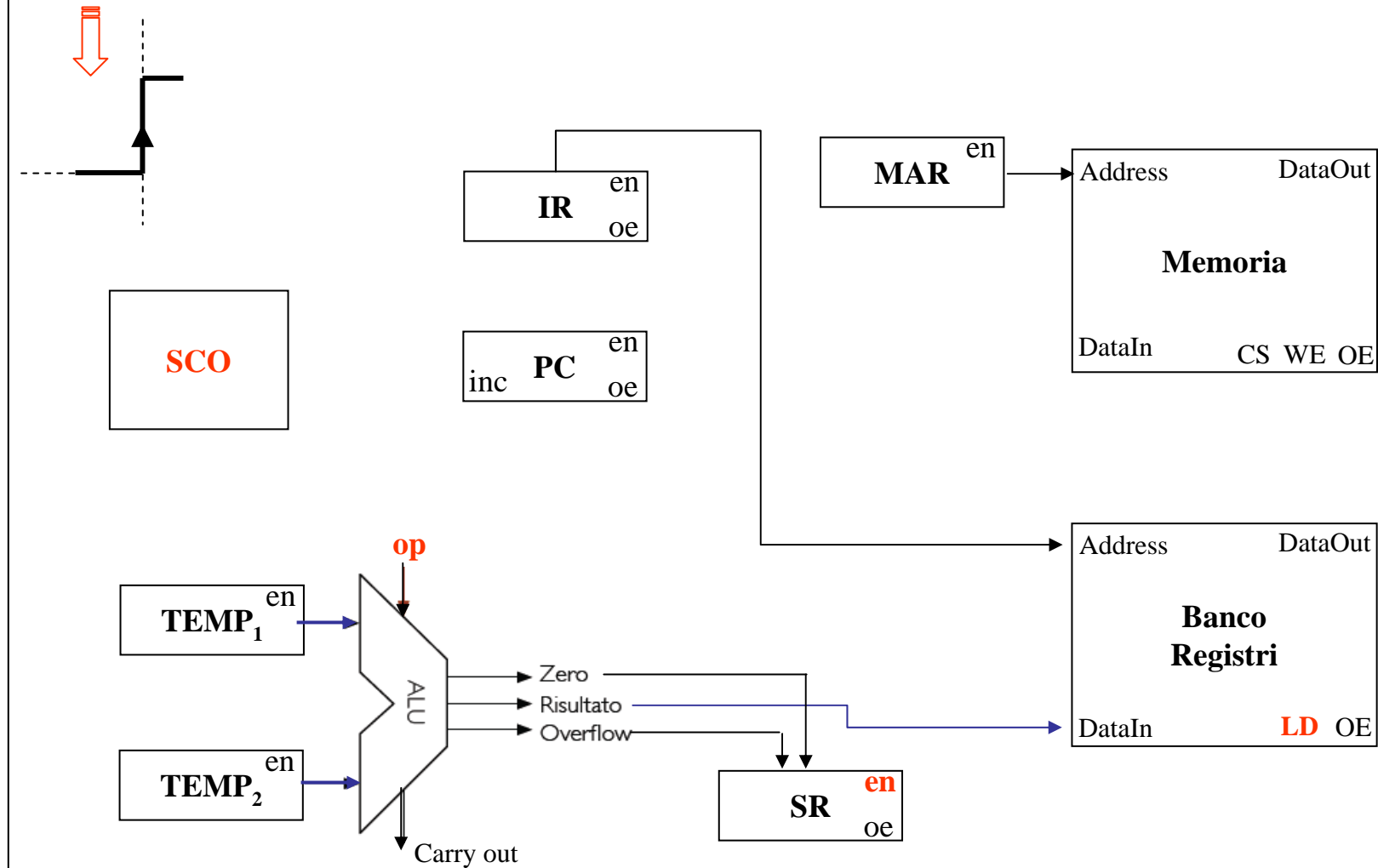
□ ADD, #01, #02, #03: fase di **DECODE 3/3**



CPU: Esempi di cicli FDE



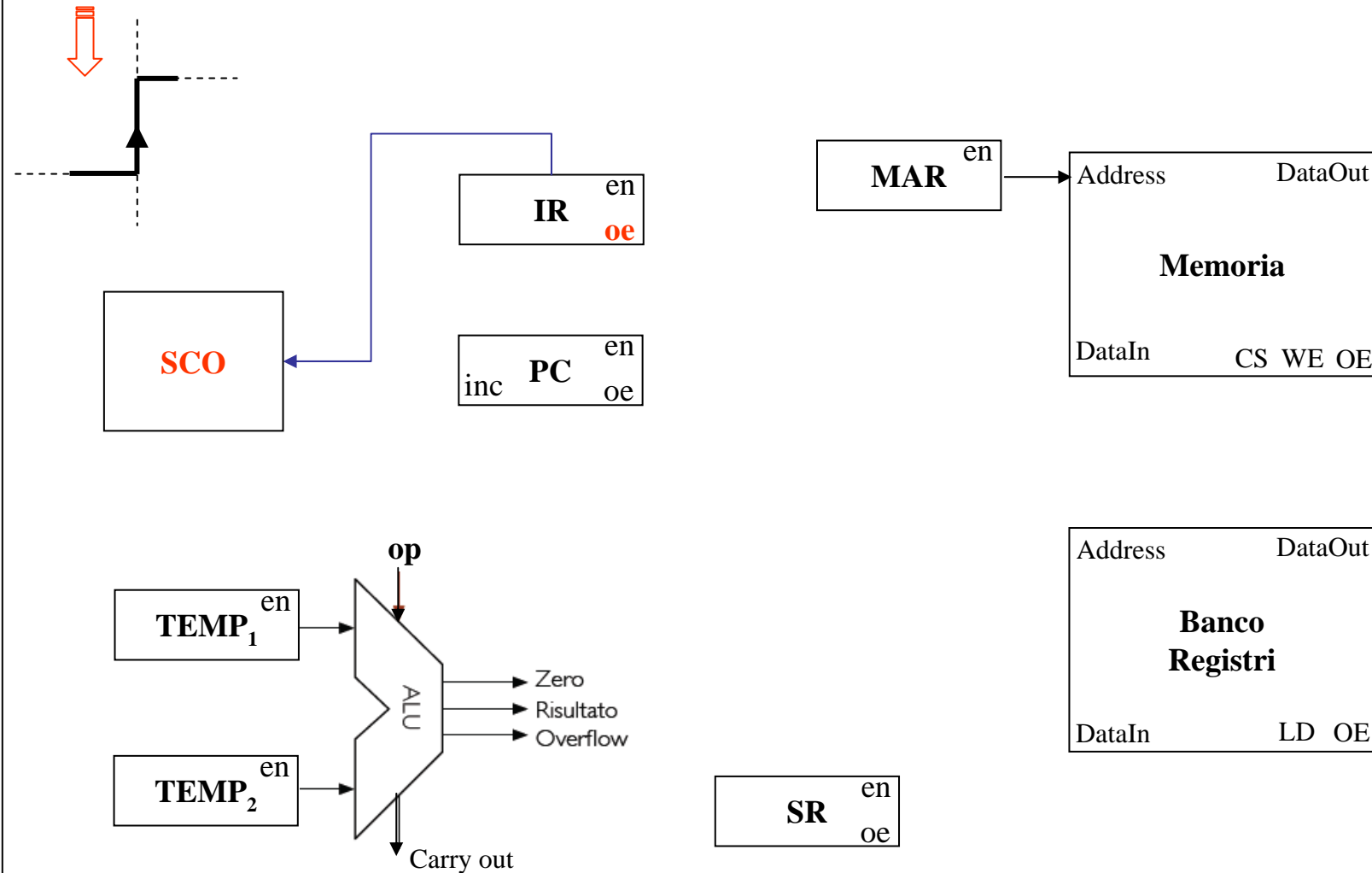
□ ADD, #01, #02, #03: fase di **EXECUTE 1/1**



CPU: Esempi di cicli FDE



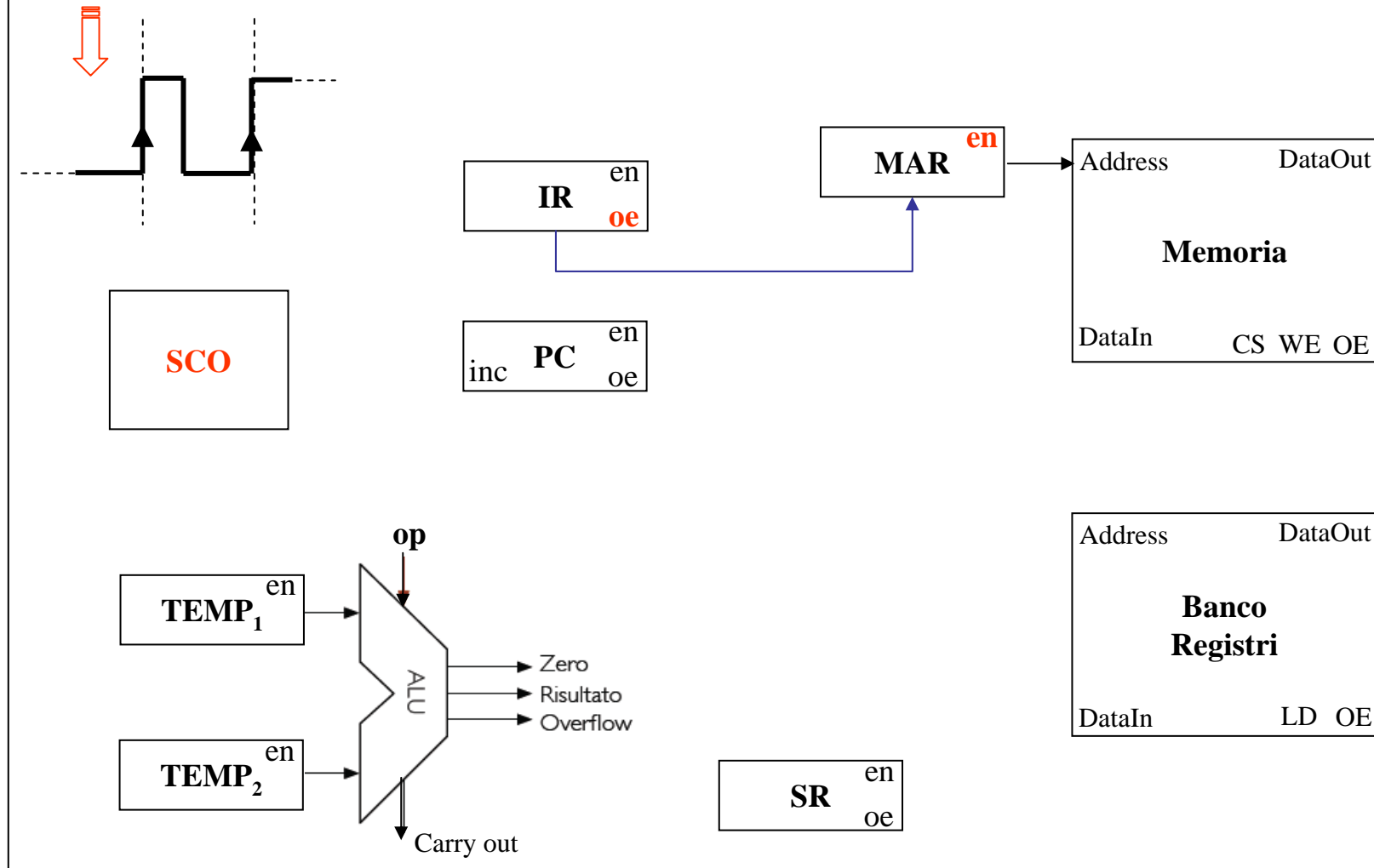
□ LOAD, #0FA0542A, #04 : fase di **DECODE 1/1**



CPU: Esempi di cicli FDE



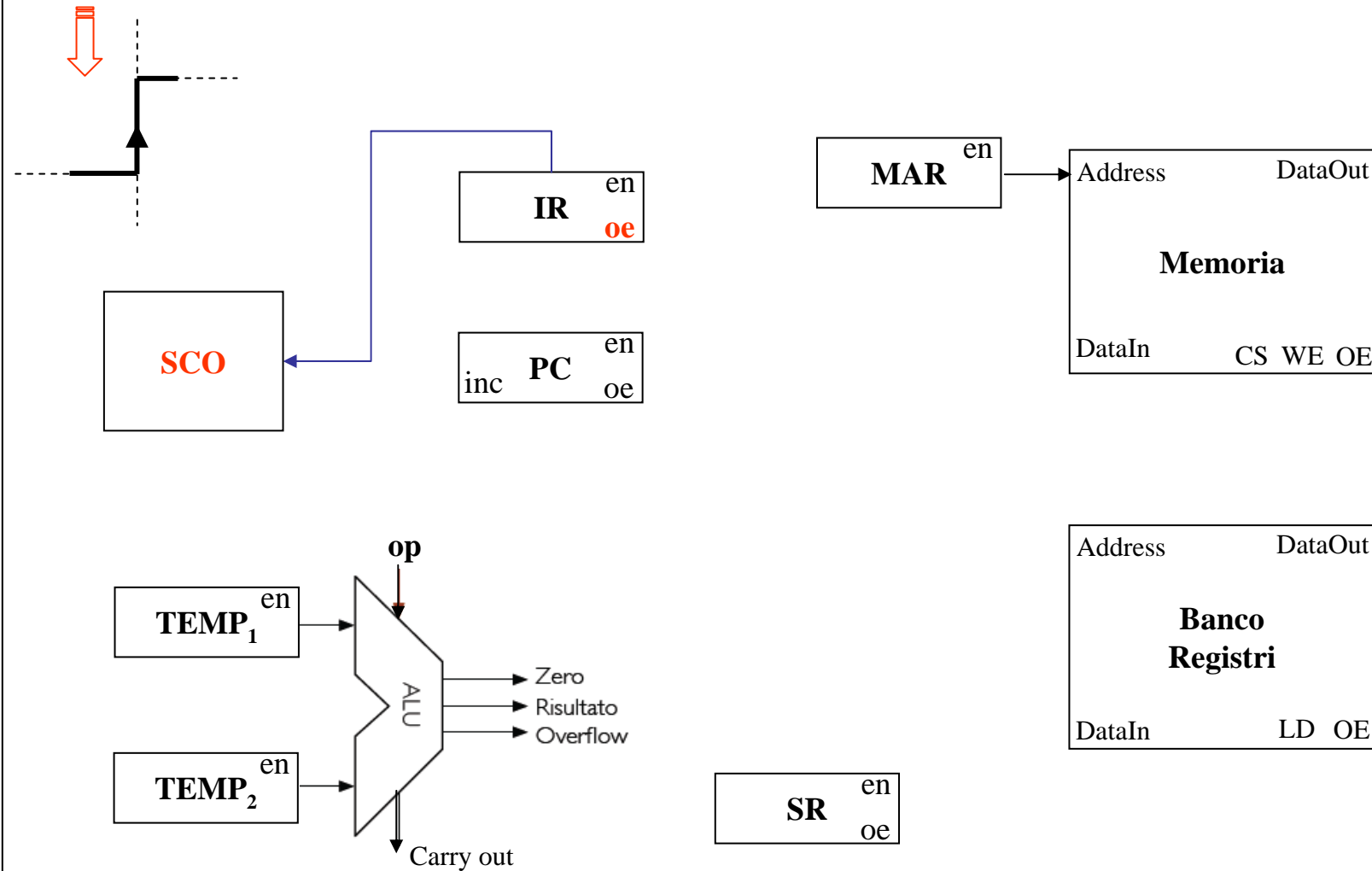
□ LOAD, #0FA0542A, #04 : fase di **EXECUTE 1/2**



CPU: Esempi di cicli FDE



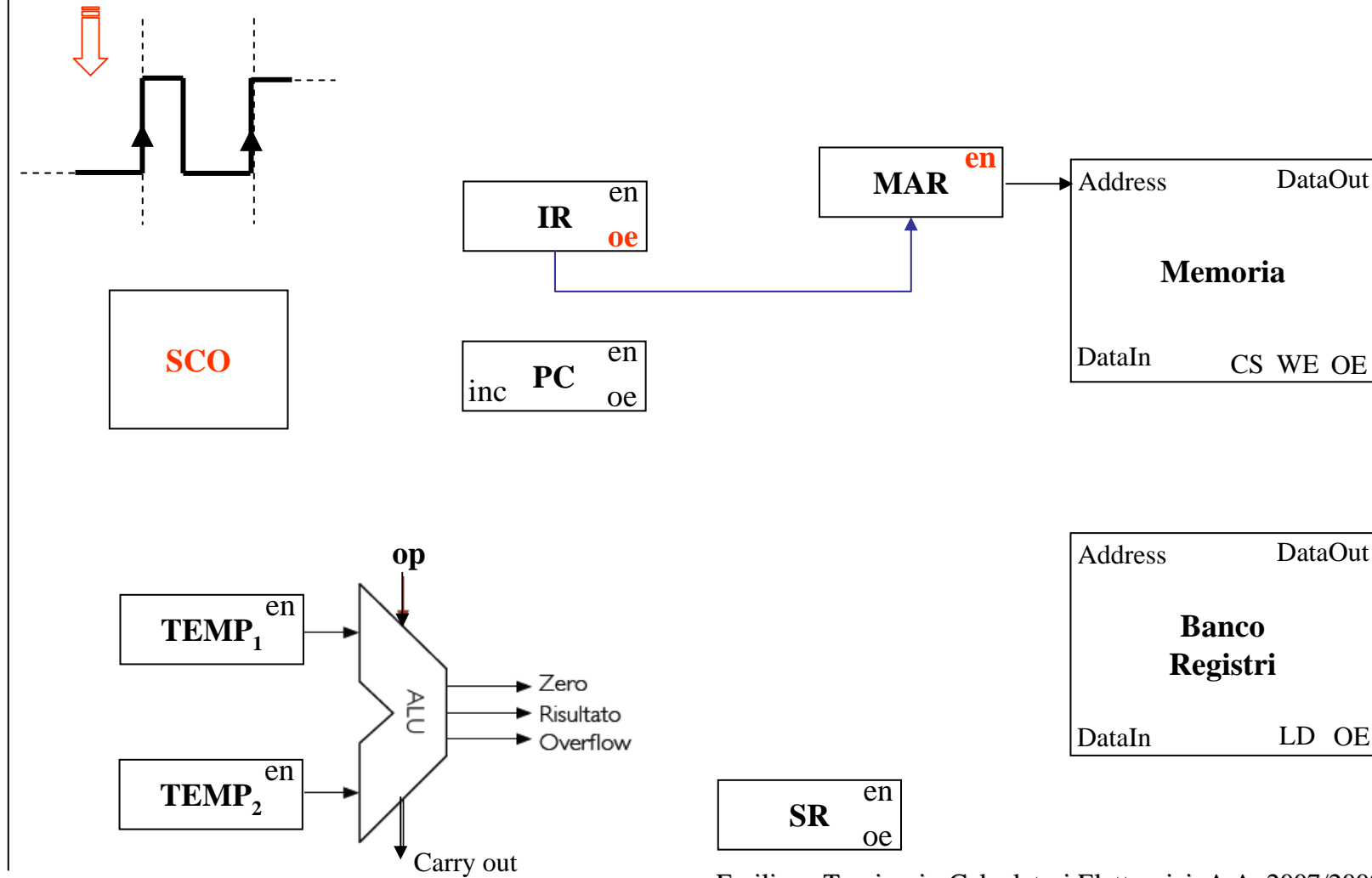
□ STORE, #04, #0FA0542A : fase di **DECODE 1/1**



CPU: Esempi di cicli FDE



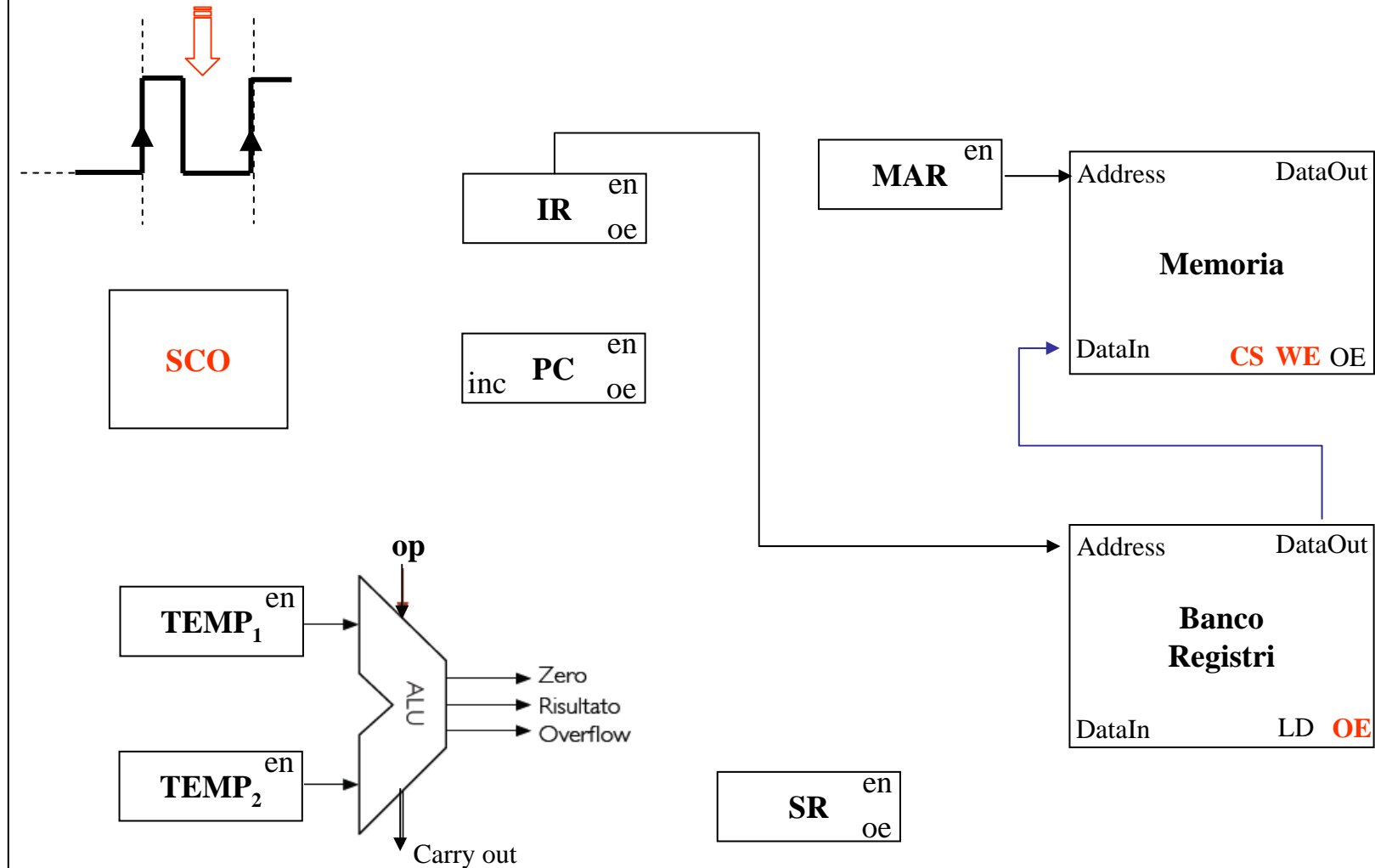
□ STORE, #04, #0FA0542A : fase di **EXECUTE 1/2**



CPU: Esempi di cicli FDE



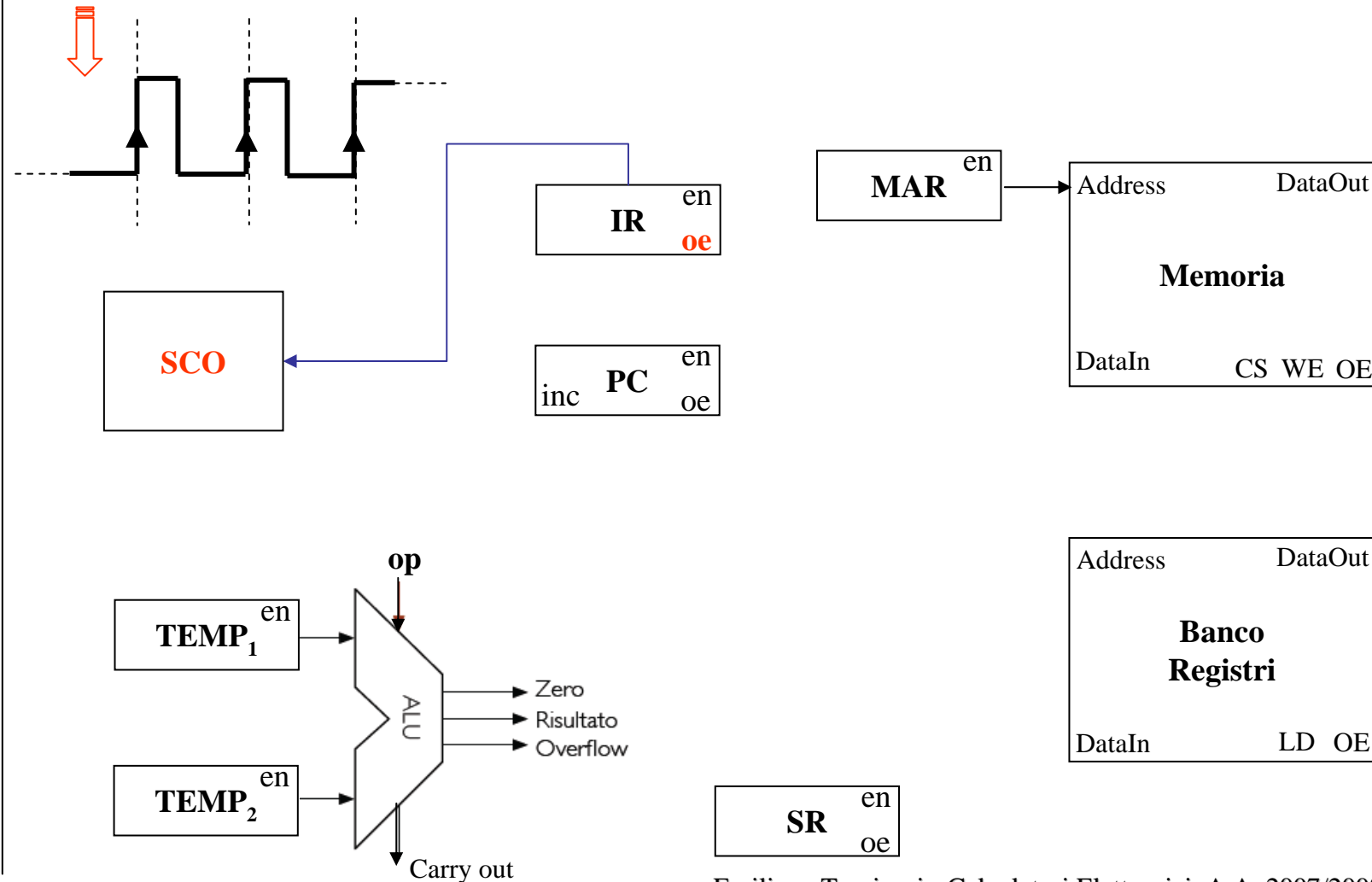
□ STORE, #04, #0FA0542A : fase di **EXECUTE 2/2**



CPU: Esempi di cicli FDE



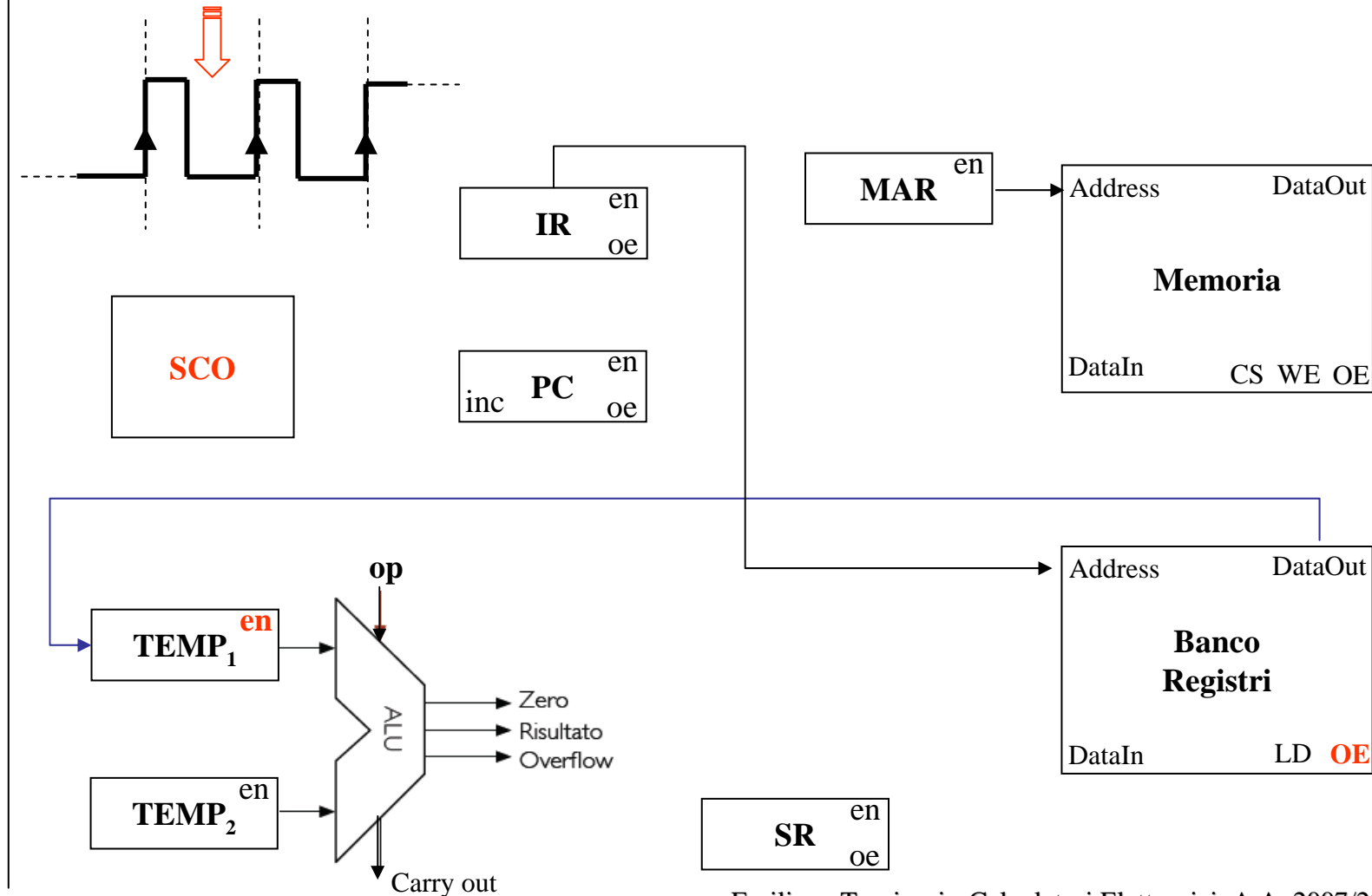
□ JMP, #01,,,: fase di **DECODE** 1/3



CPU: Esempi di cicli FDE



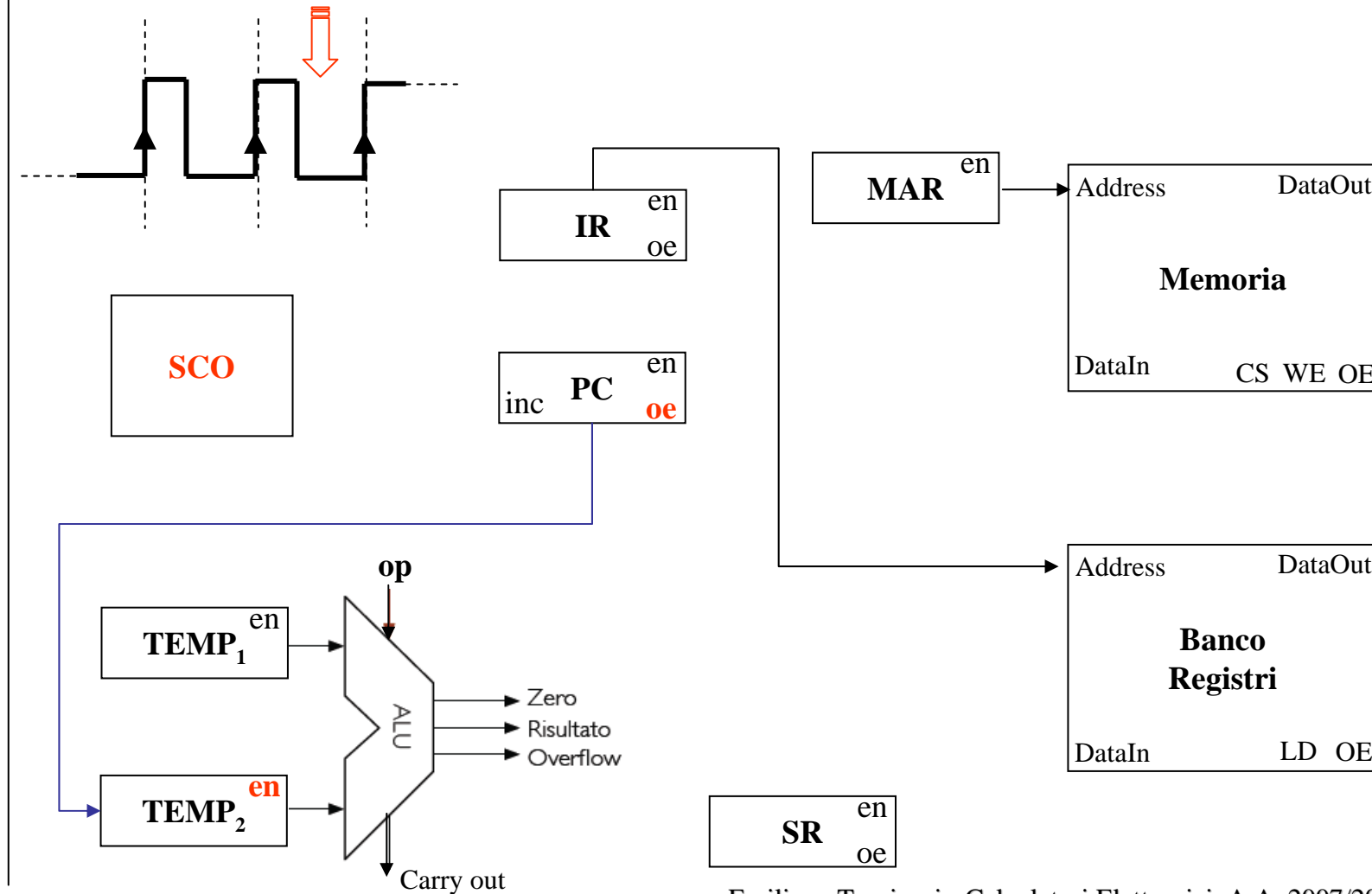
□ JMP, #01,,,: fase di **DECODE 2/3**



CPU: Esempi di cicli FDE



□ JMP, #01,,,: fase di **DECODE 3/3**



CPU: Esempi di cicli FDE



□ JMP, #01,,,: fase di **EXECUTE 1/1**

