Esempio di esecuzione di un programma in linguaggio macchina

Analizziamo ora più in dettaglio il comportamento della CPU durante l'esecuzione delle istruzioni

per avere un'idea più precisa di come la control unit coordina il processore.

REGISTRI SPECIALI

- **PC - program counter**:

  contiene **l'indirizzo** in memoria centrale **dell'istruzione che deve essere eseguita**.

  - Ad ogni istruzione eseguita il PC viene modificato per contenere l'indirizzo della istruzione successiva

REGISTRI SPECIALI

- **IR (*Instruction Register), registro istruzione corrente.***

  Contiene, istante per istante, **istruzione** che è attualmente **in esecuzione**.

- **PS - Program status o registro di stato**

  Fornisce informazioni sul risultato dell'ultima operazione eseguita dalla ALU (zero, carry, segno …)

REGISTRI SPECIALI

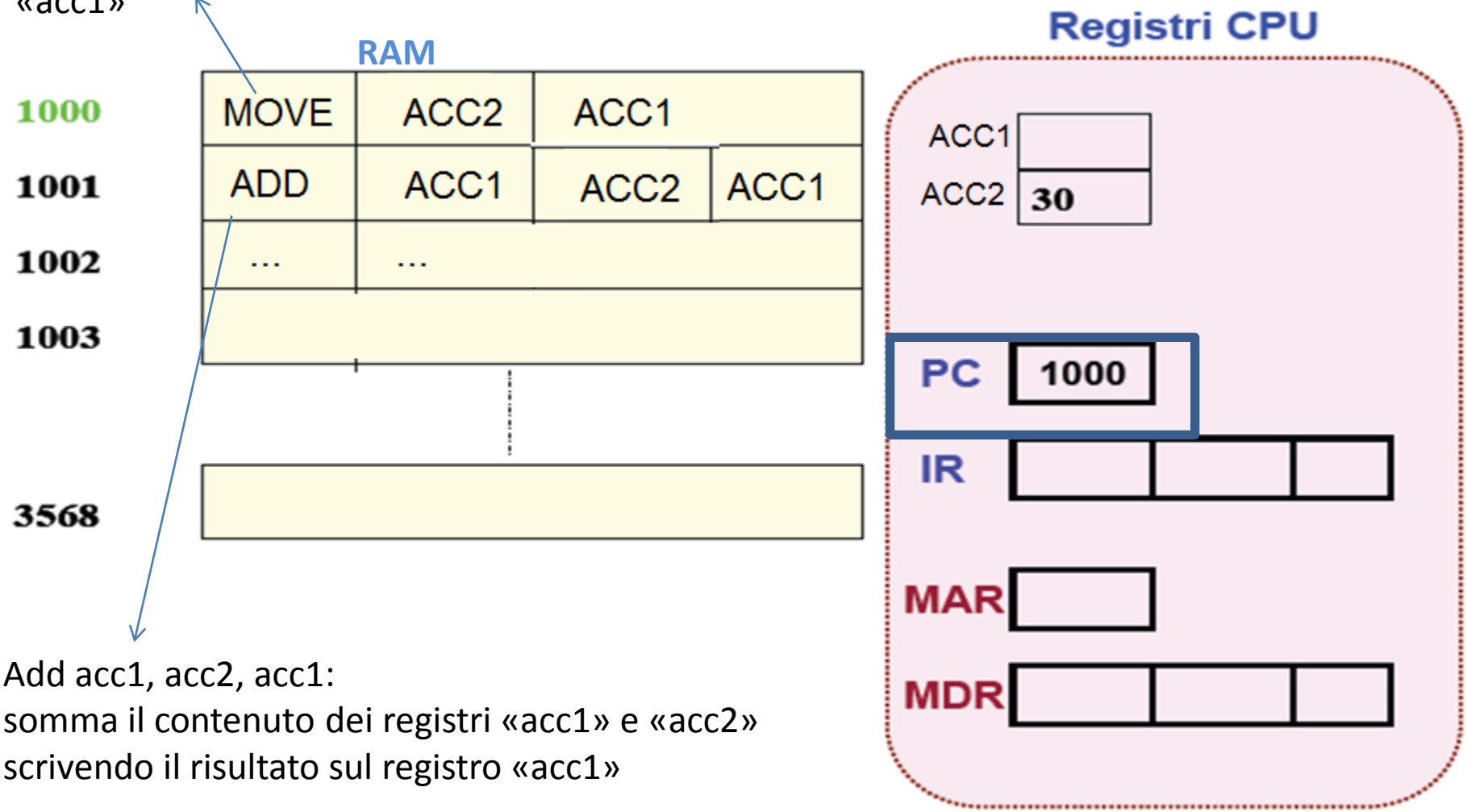
- il **registro Stack Pointer (*SP),*** utilizzato per la gestione di un **segnale di interruzione;**
- **MAR – Memory Address Register o Registro Indirizzi Memoria**
  - ✓ Contiene l'indirizzo della cella da cui leggere o in cui scrivere un dato. → lo stesso di PC
  - ✓La dimensione di MAR determina l'ampiezza dello spazio di memoria fisica essendo legato al bus indirizzi (es: a 32 bit)

REGISTRI SPECIALI

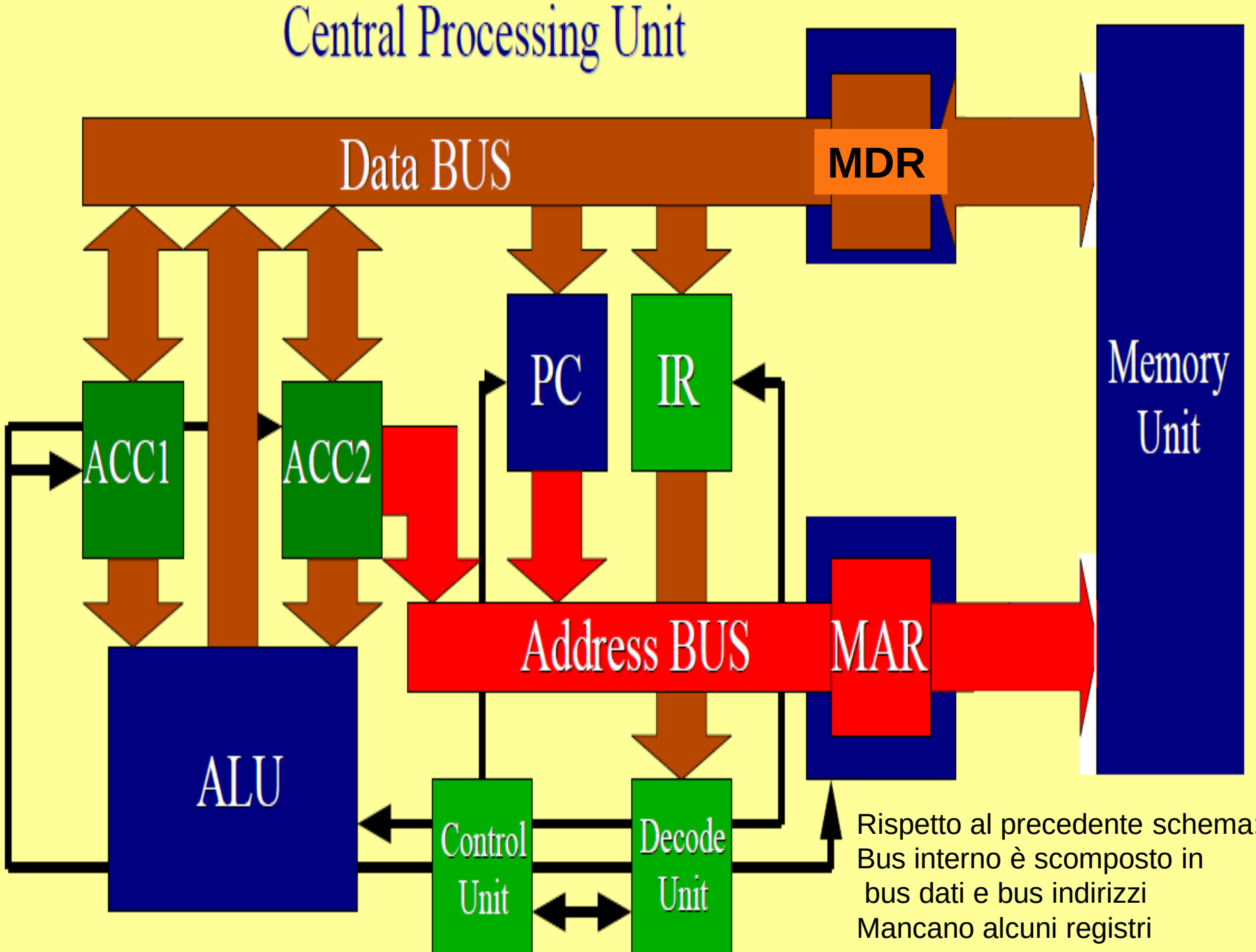
- **MDR – Memory Data Register Registro dati Memoria**
  - ✓registro attraverso il quale viene scambiata l'informazione tra la memoria e la CPU
  - ✓Contiene il dato letto dalla memoria o da scrivere in memoria
  - ✓dà la misura del grado di parallelismo (n° di bit di dati che possono essere letti o ricevuti in una singola operazione) della macchina (8, 16, 32, 64 bit)

Esecuzione del programma macchina

Move acc2, acc1: copia di un dato dal registro interno «acc2» a quello «acc1»



Add acc1, acc2, acc1:
somma il contenuto dei registri «acc1» e «acc2» scrivendo il risultato sul registro «acc1»



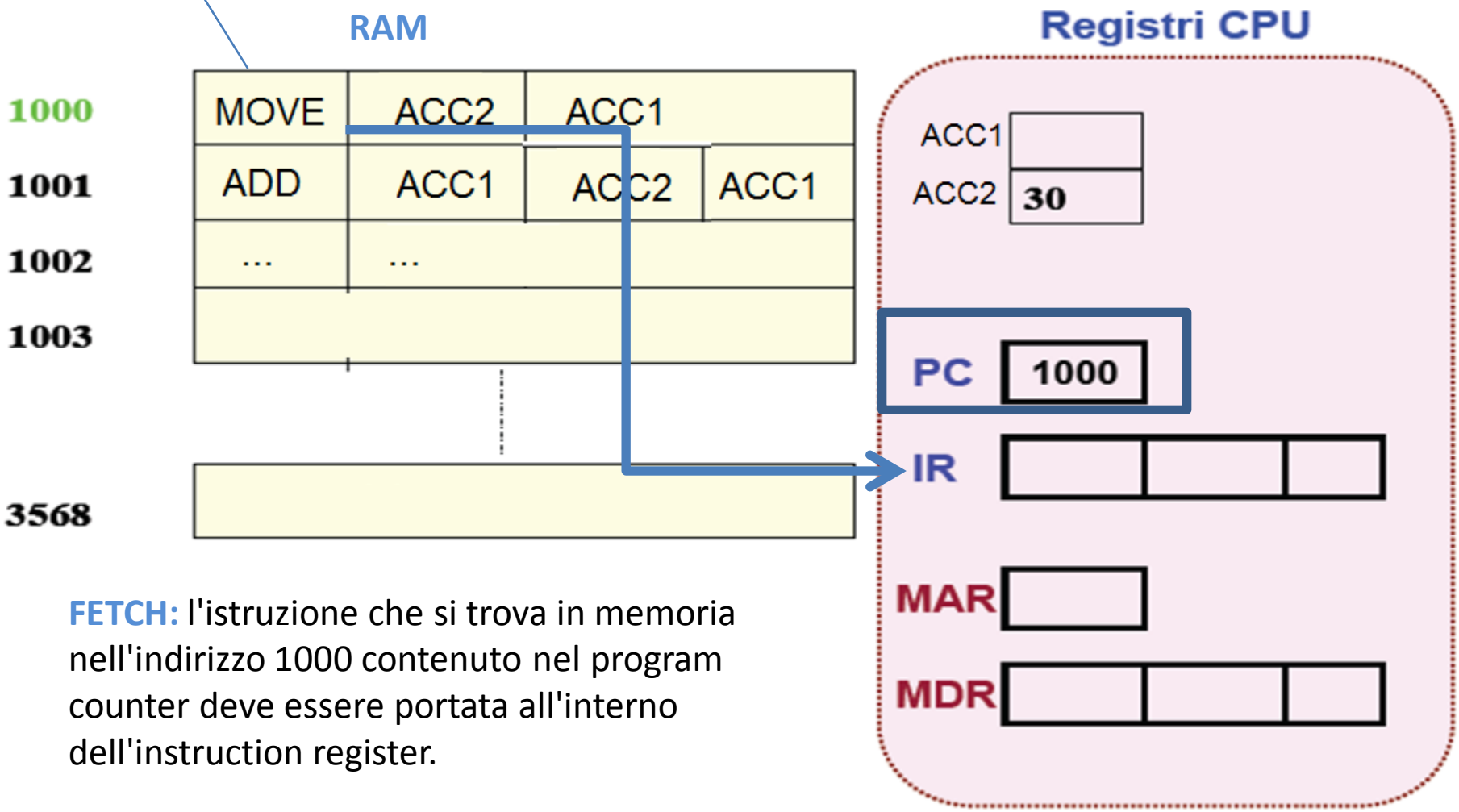
Rispetto al precedente schema:
Bus interno è scomposto in
bus dati e bus indirizzi
Mancano alcuni registri

1° FASE: FETCH

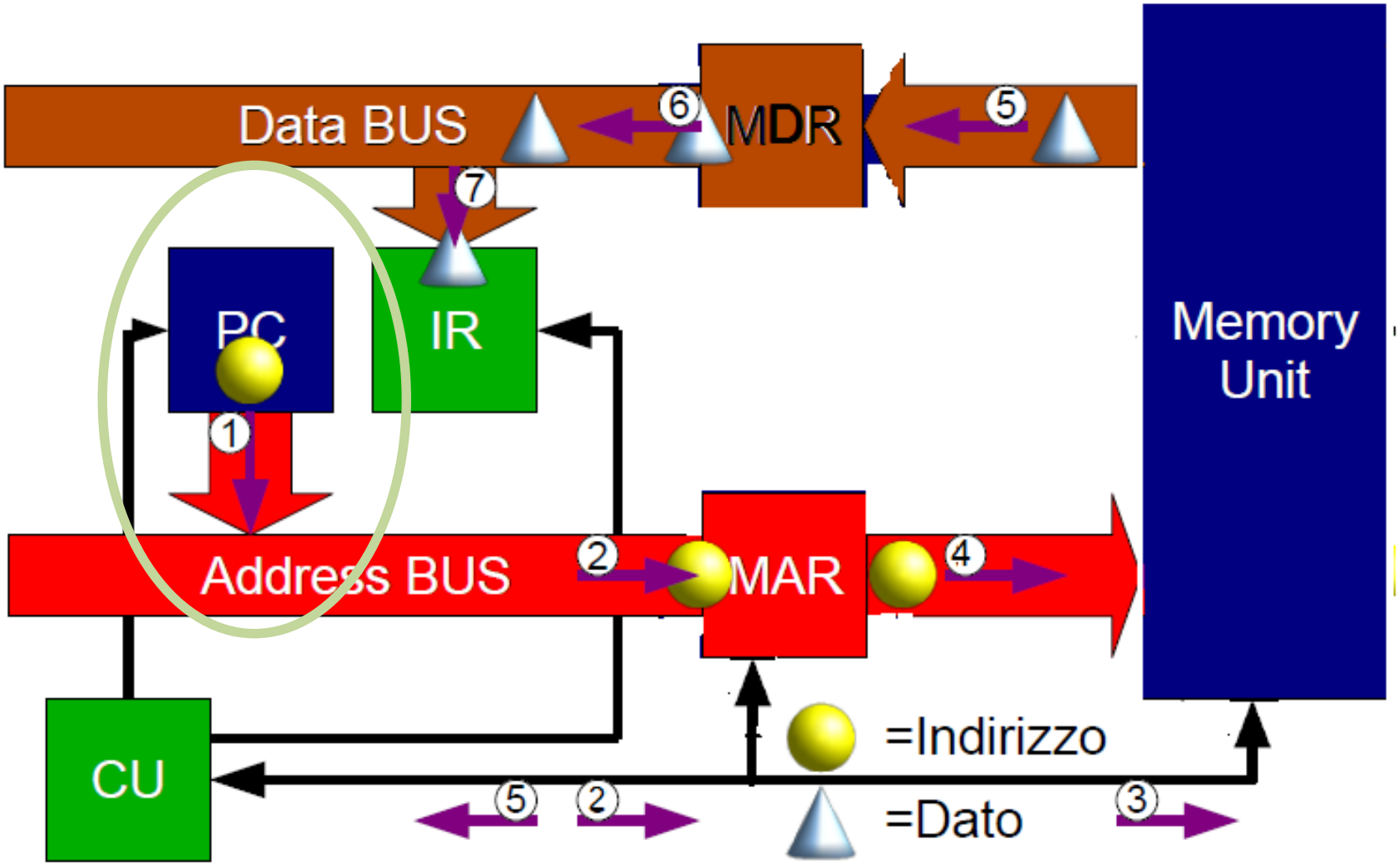
In questa fase l'istruzione che si trova in memoria nell'indirizzo contenuto nel program counter deve essere recuperata e portata all'interno dell'instruction register.

Esecuzione istruzione 1000

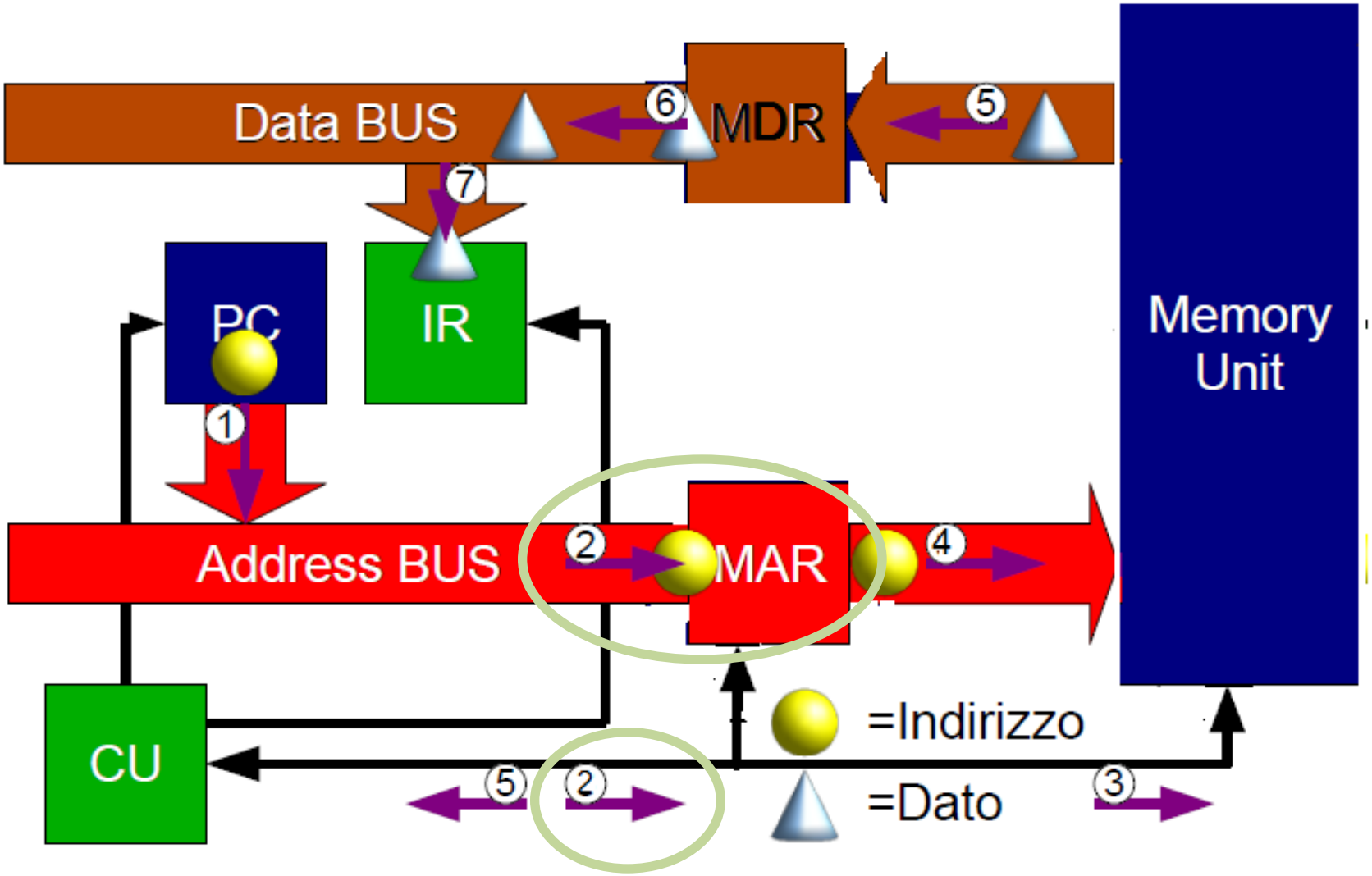
Move acc2, acc1: copia di un dato dal registro interno «acc2» a quello «acc1»



FETCH: l'istruzione che si trova in memoria nell'indirizzo 1000 contenuto nel program counter deve essere portata all'interno dell'instruction register.

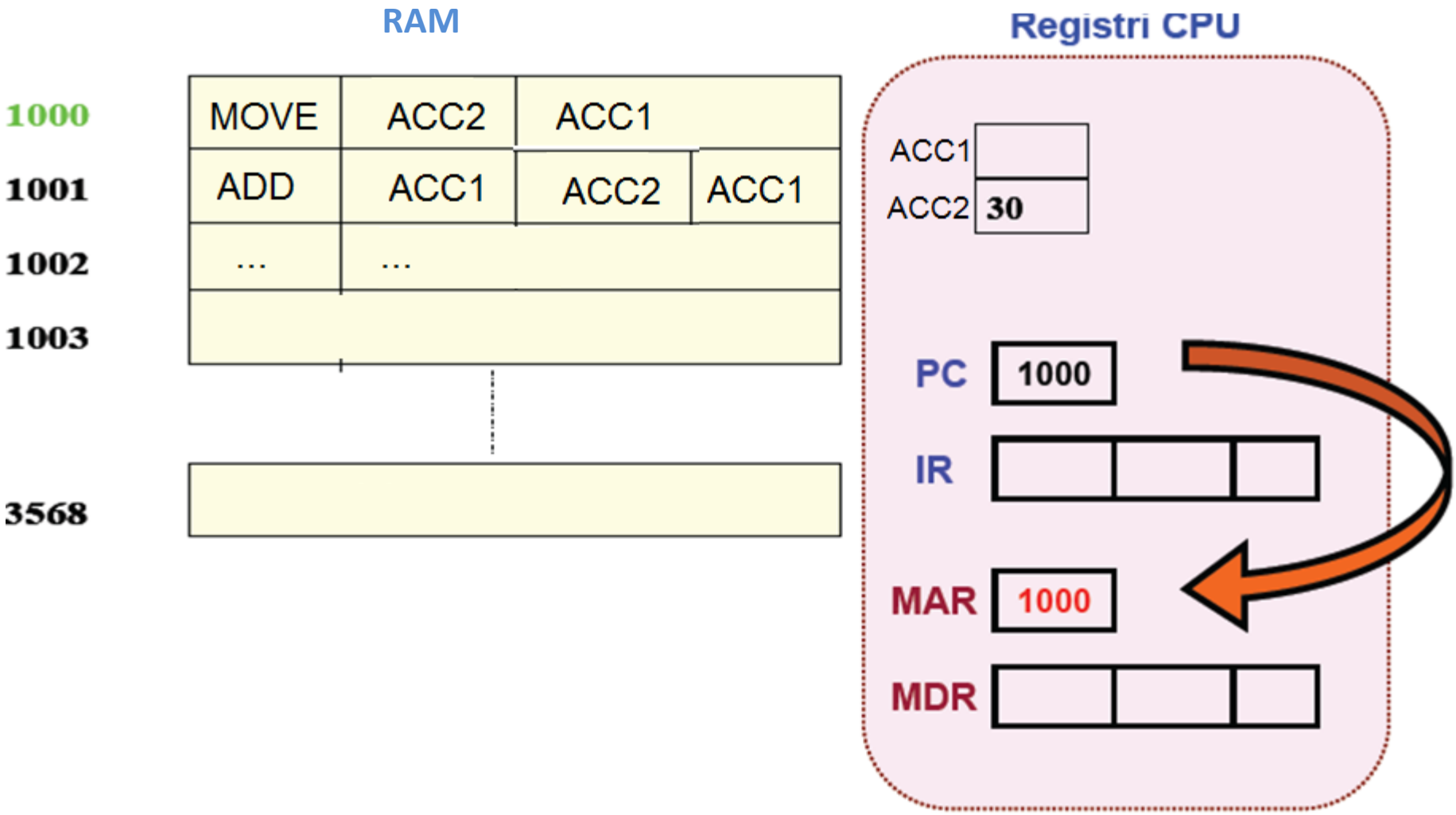


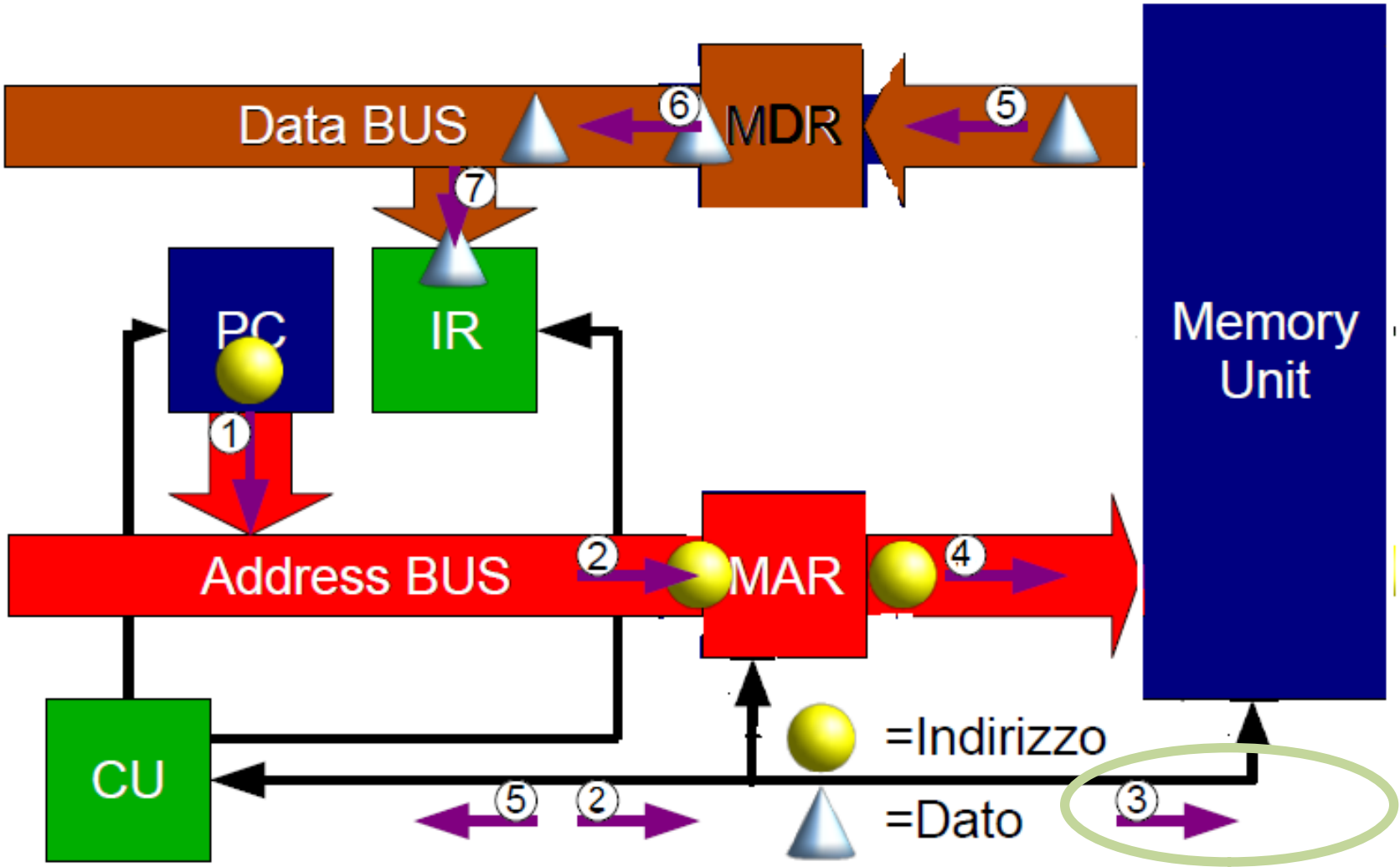
1. La CU ordina al PC di copiare il proprio contenuto sull'address bus (ossia l'indirizzo della prossima istruzione, istruzione 1000, in figura la pallina gialla)



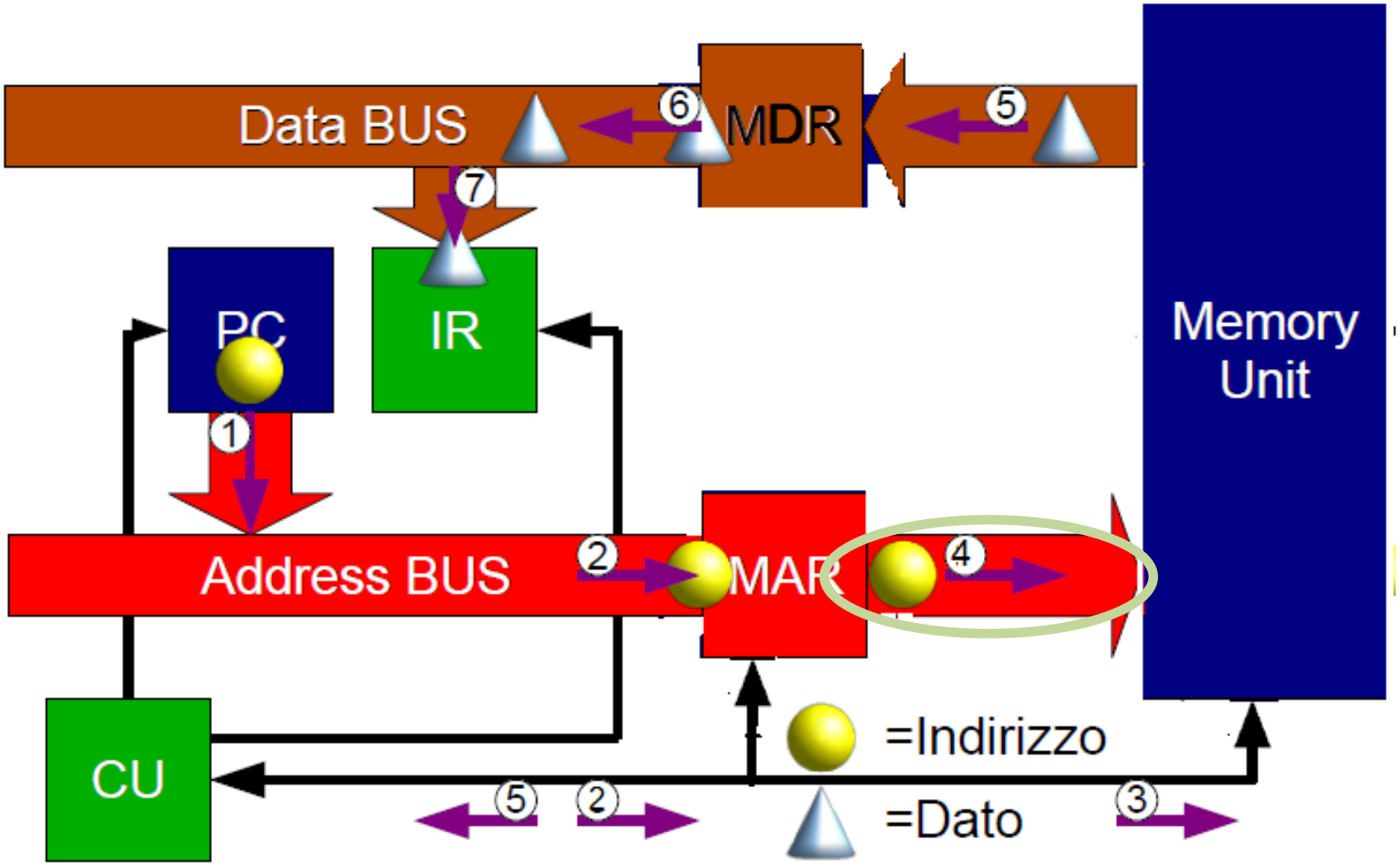
2. La CU ordina al MAR di immagazzinare l'indirizzo presente nell'address bus

FETCH(1)

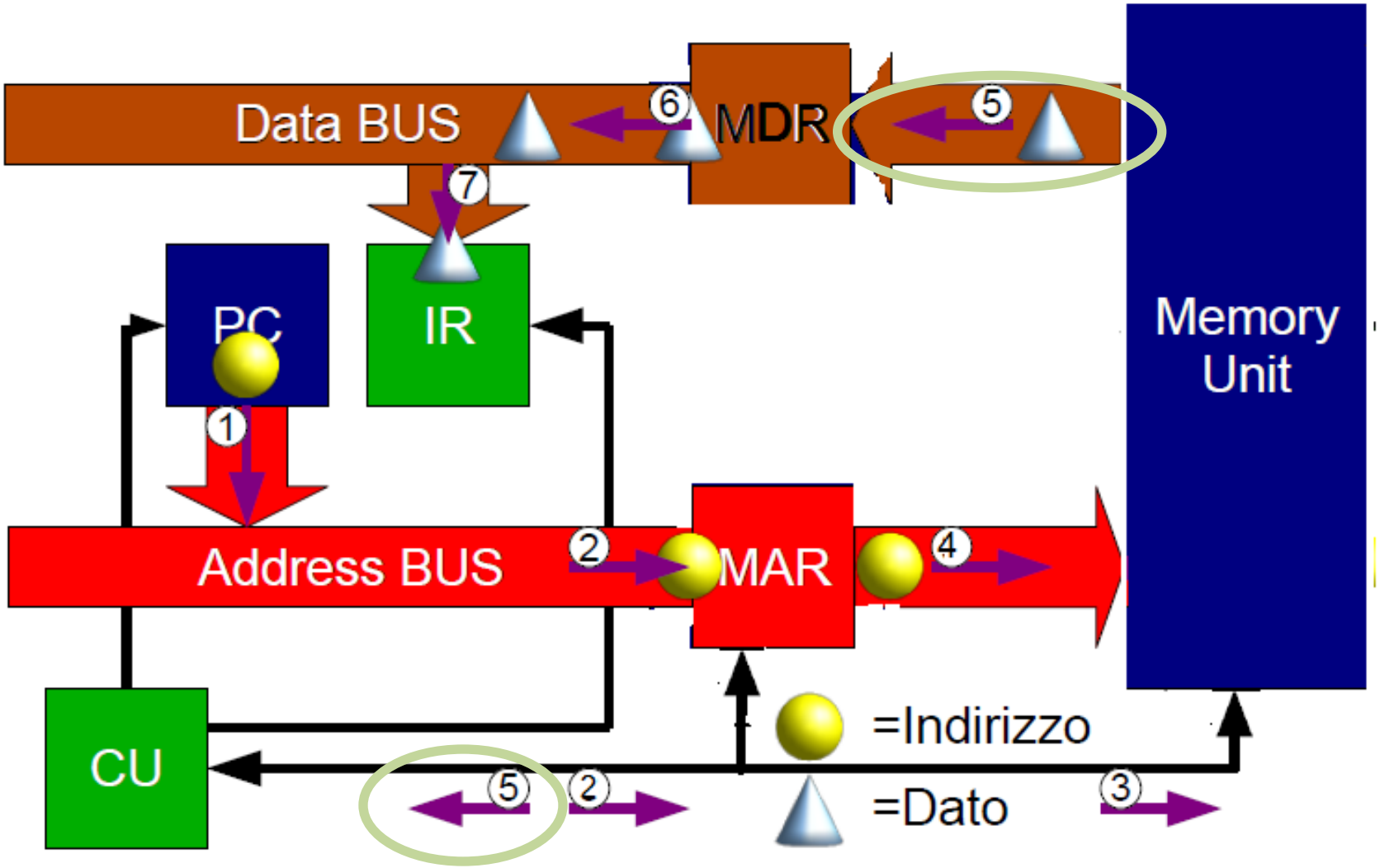

RAM
Registri CPU
1000
1001
1002
1003
3568
MOVE
ACC2
ACC1
ADD
ACC1
ACC2
ACC1
...
...
ACC1
ACC2
30
PC
1000
IR
MAR
1000
MDR



3. La CU ordina alla memoria di effettuare una operazione di lettura

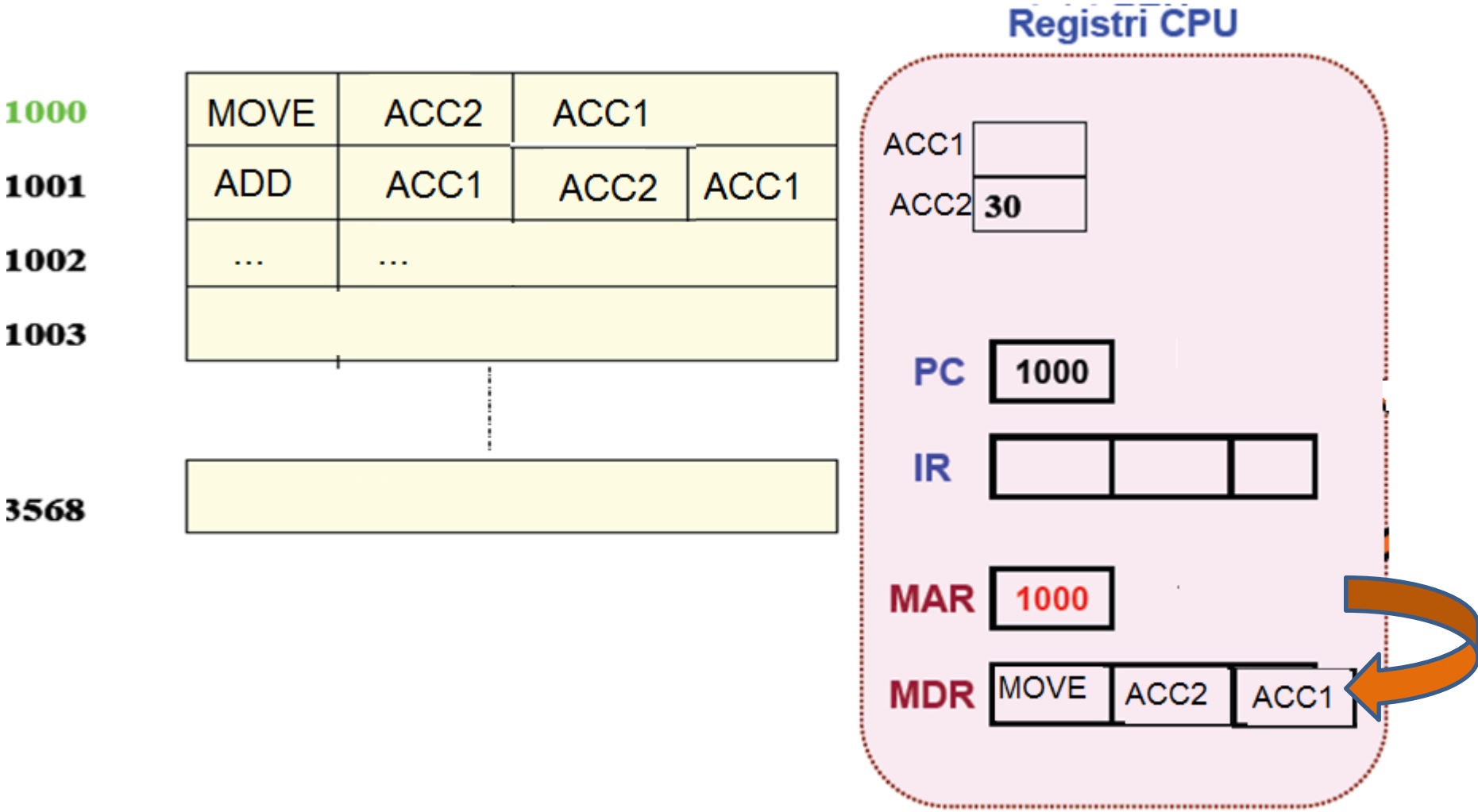


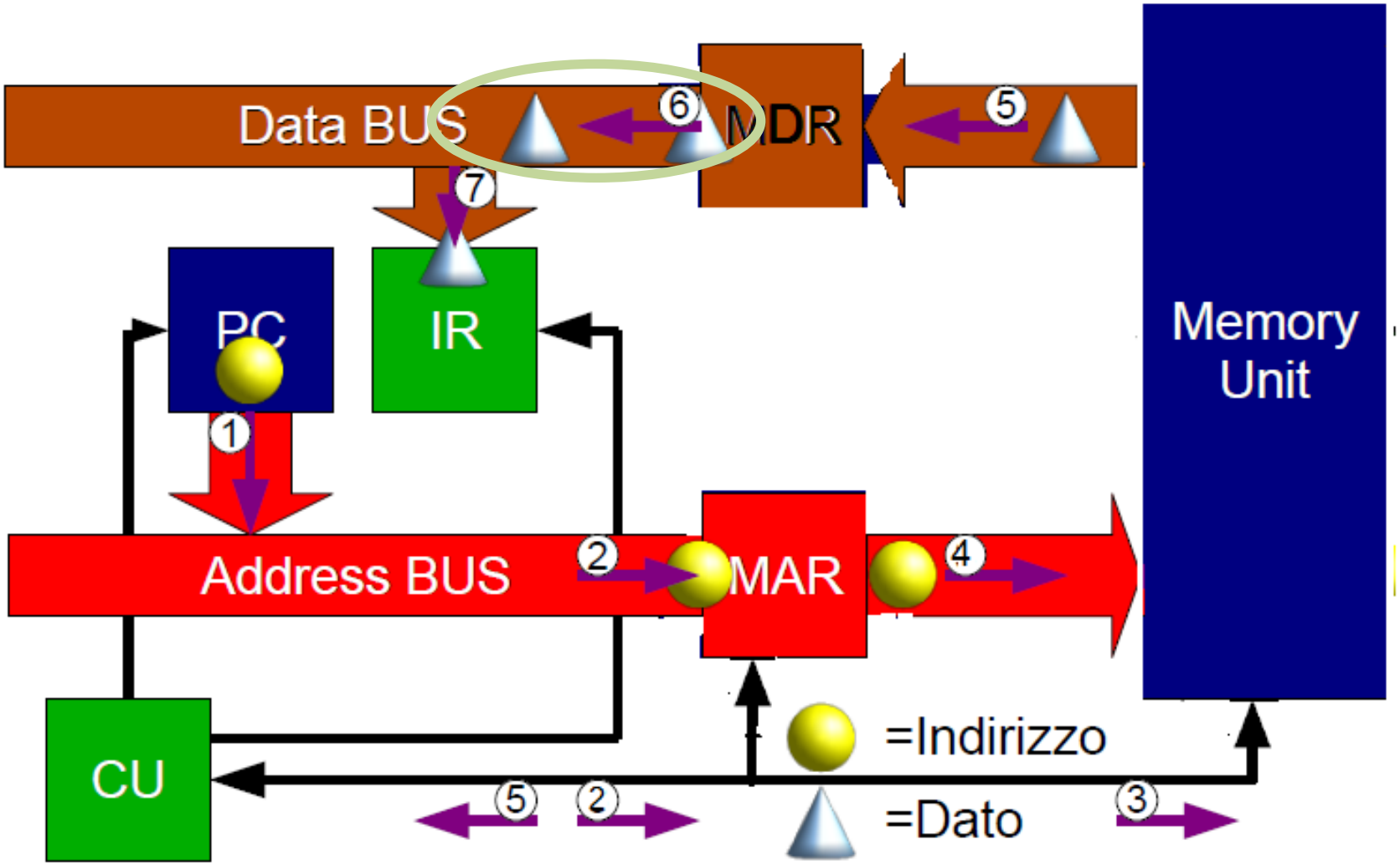
4. L'indirizzo del MAR viene copiato sull'address bus esterno



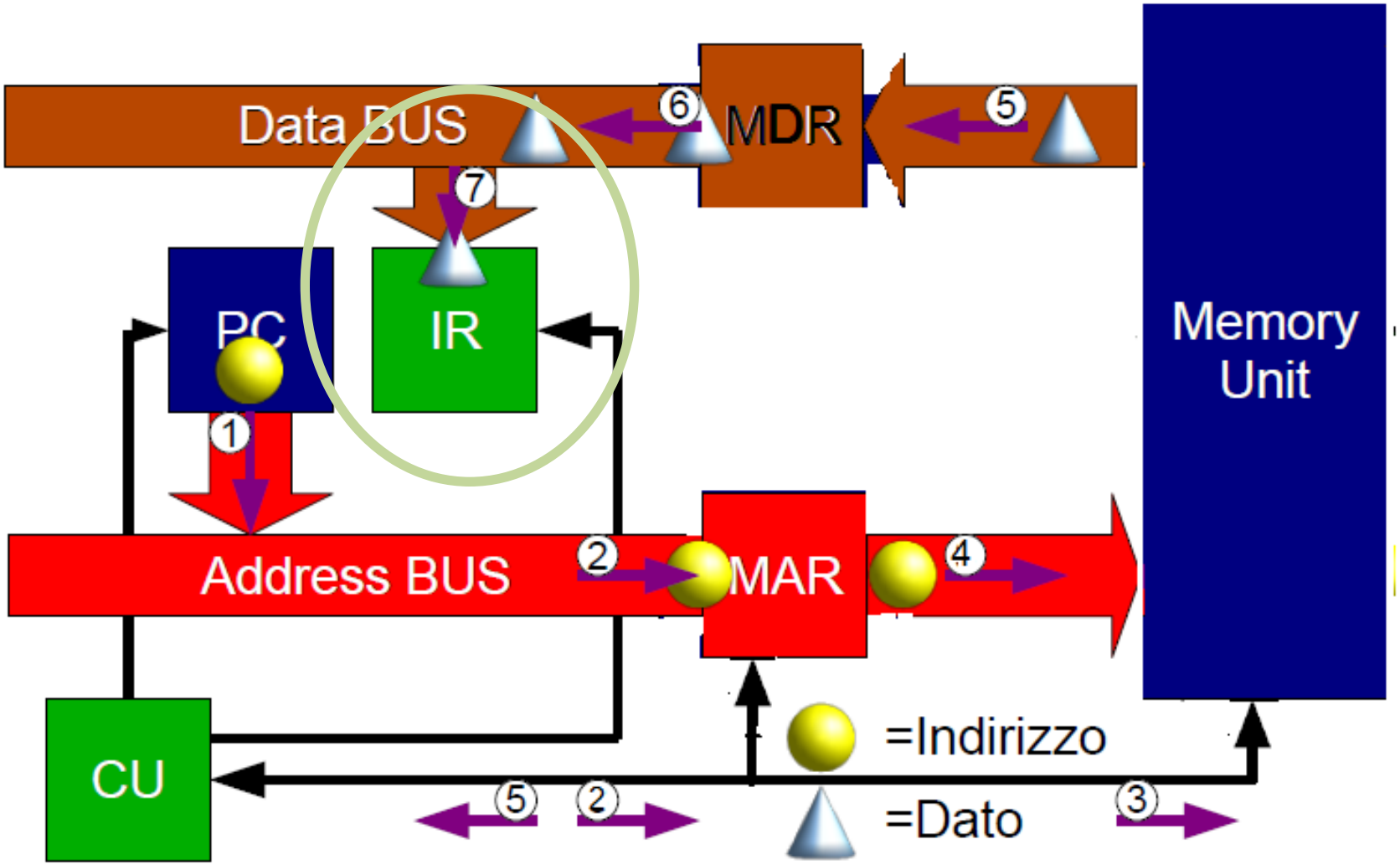
5. La memoria recupera il dato (ossia l'istruzione da eseguire, triangolino azzurro), lo copia nell'MDR, e avverte la CU che il dato è stato recuperato tramite il control bus

FETCH(2)


1000
1001
1002
1003
3568
MOVE
ACC2
ACC1
ADD
ACC1
ACC2
ACC1
...
...
Registri CPU
ACC1
ACC2
30
PC
1000
IR
MAR
1000
MDR
MOVE
ACC2
ACC1



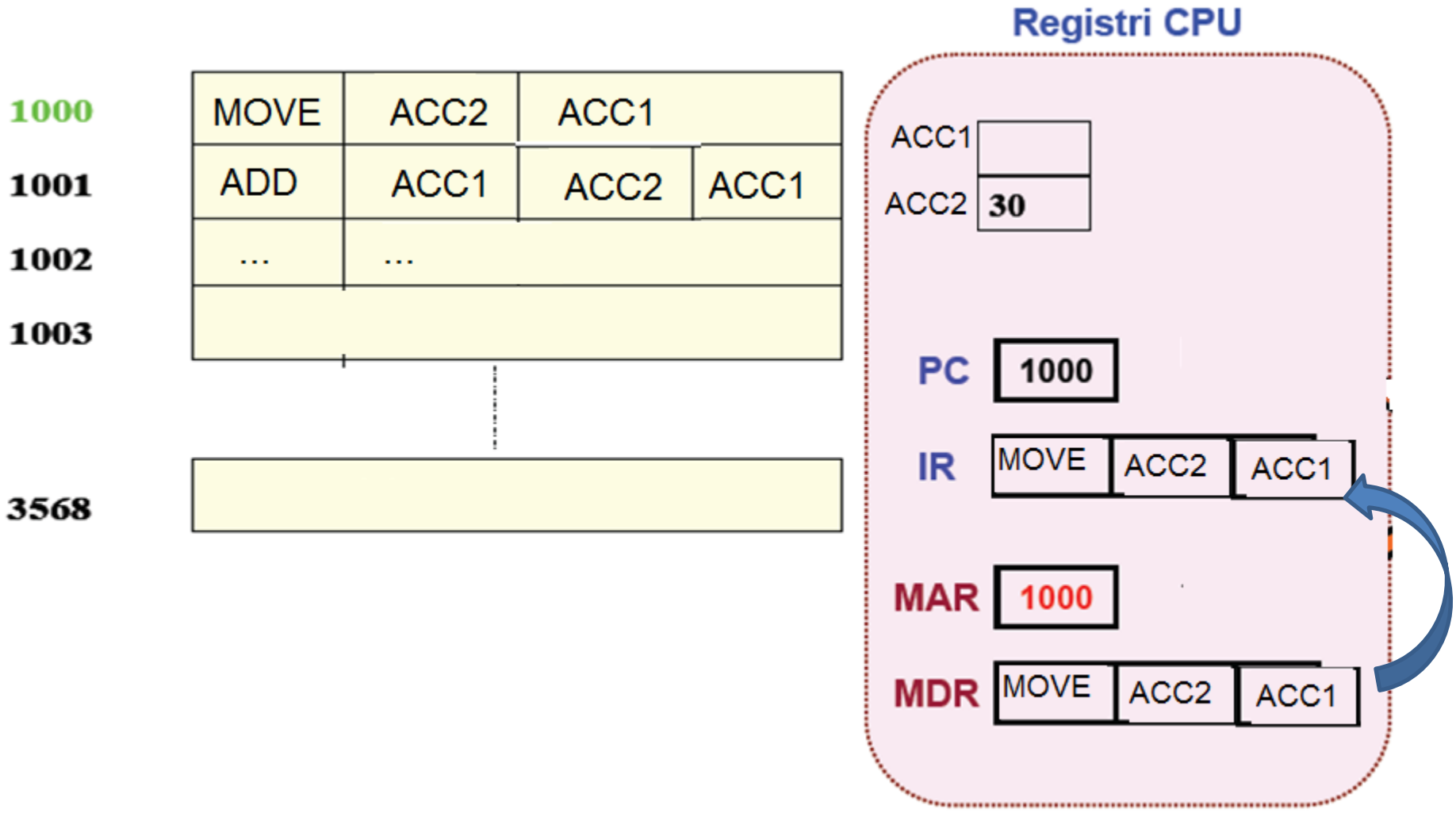
6. La CU ordina all'MDR di copiare il dato nel data bus interno



7. La CU ordina all'IR di copiare il dato contenuto nell'MDR

FETCH(3)

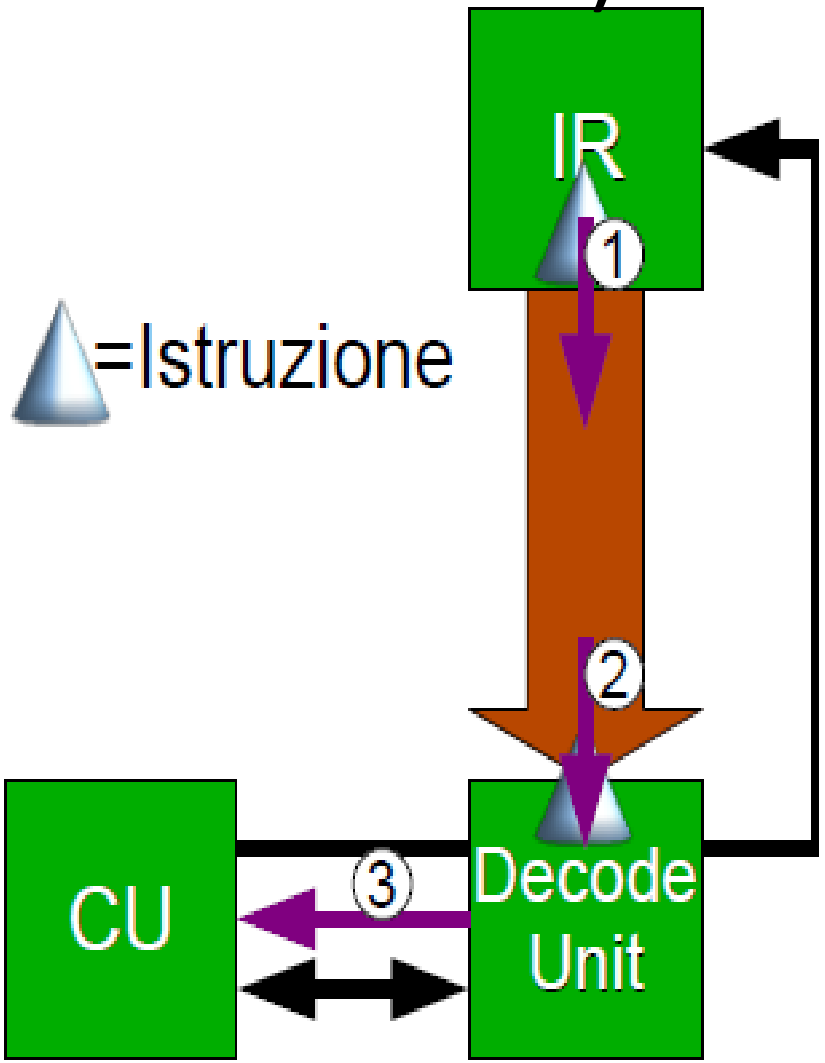
A questo punto l'istruzione da eseguire è all'interno della CPU nel registro IR e la prima fase dell'elaborazione è terminata.



Fase 2: Decode (Decodifica dell'istruzione)

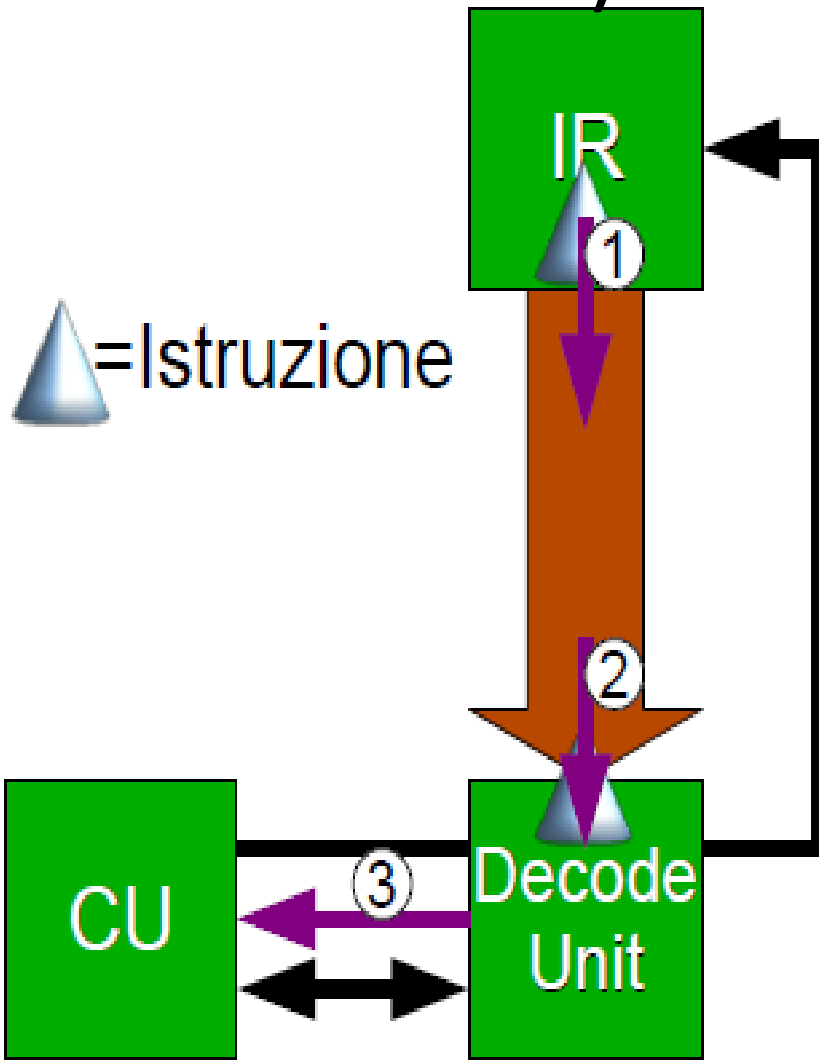
La figura mostra le unità coinvolte nella decodifica dell'istruzione:

si tratta di una fase in cui la CU necessita di sapere con che istruzione ha a che fare in modo da poter proseguire opportunamente l'elaborazione.



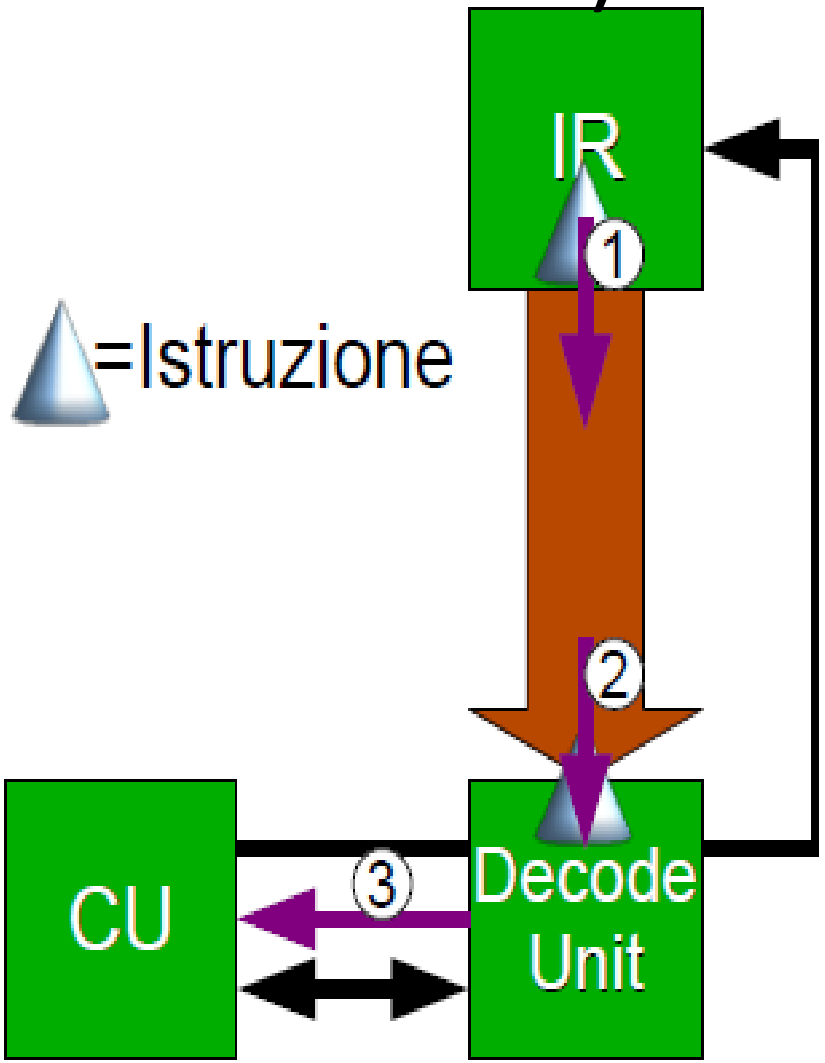
Fase 2: Decode
(Decodifica dell'istruzione)

Il processo è piuttosto semplice, è operato per la maggior parte dal decodificatore, e può essere riassunto nelle seguenti fasi:



Fase 2: Decode
(Decodifica dell'istruzione)

1. La CU ordina all'IR di spedire l'istruzione caricata al decodificatore
2. La CU ordina al decodificatore di analizzare l'istruzione e si mette in attesa del risultato
3. A decodifica avvenuta, il decodificatore comunica alla CU di che istruzione di tratta

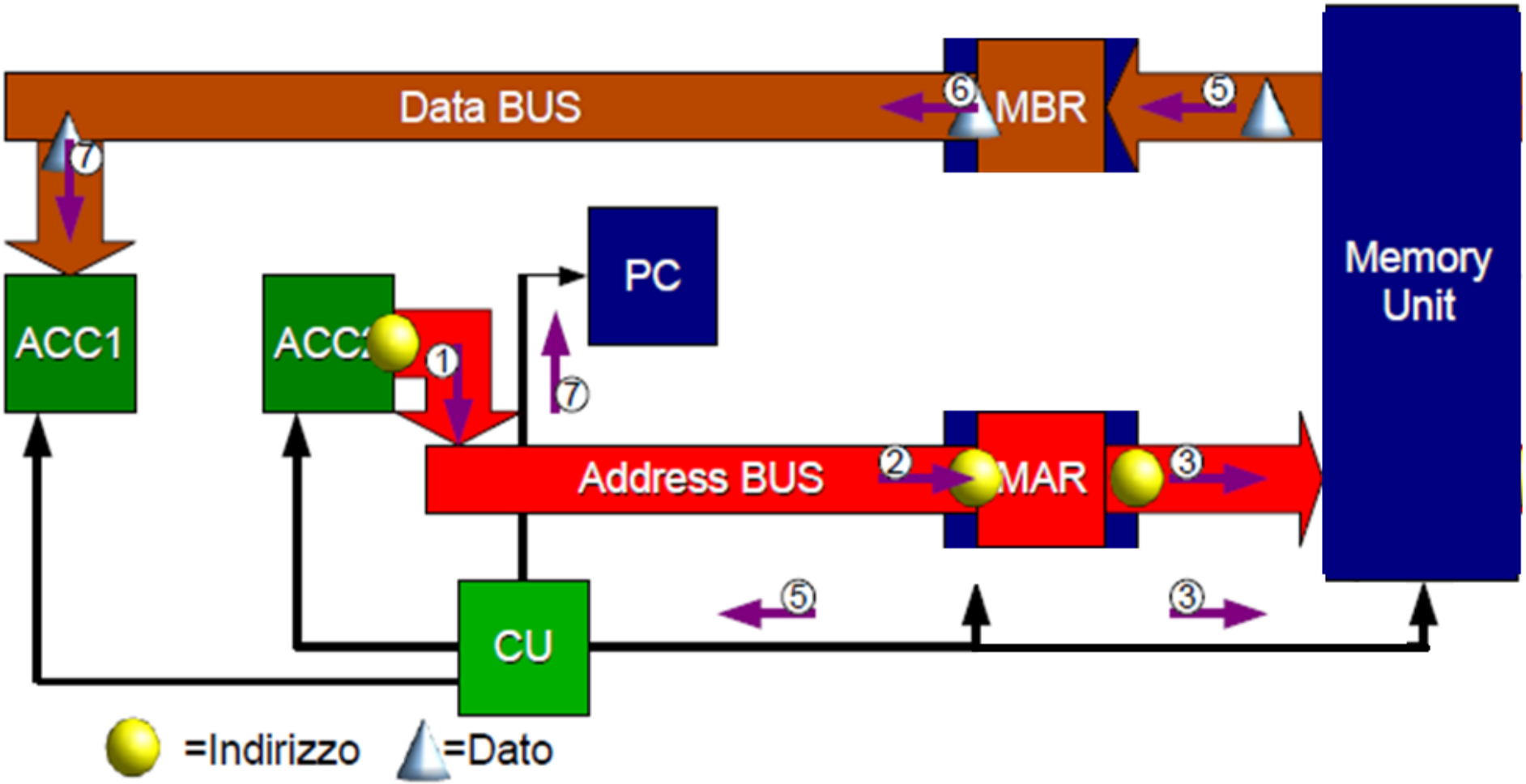


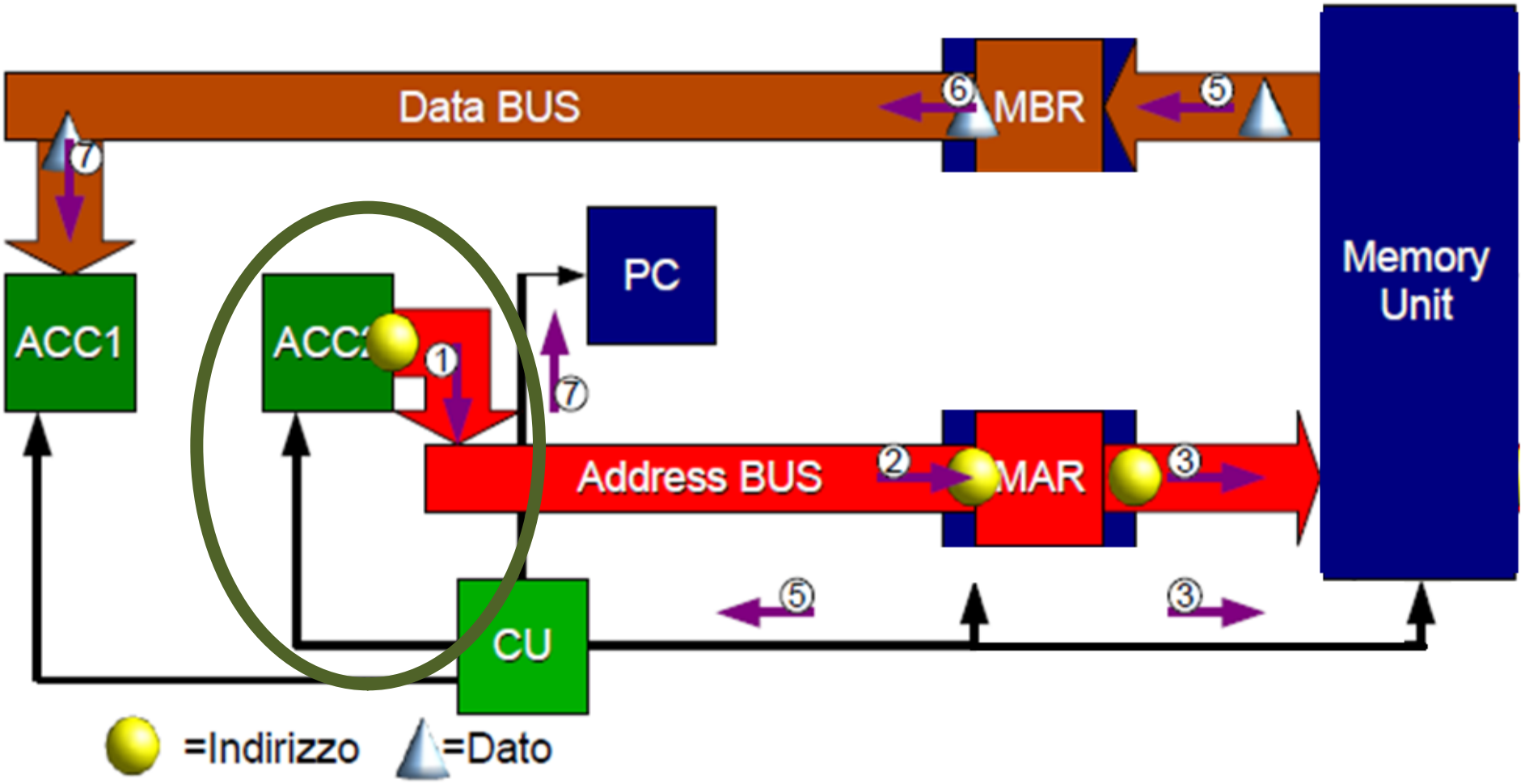
Fase 3: Execute

- A questo punto la CU conosce tutte le informazioni necessarie per procedere con la fase 3, ossia con l'esecuzione dell'istruzione vera e propria.
- Il modo con cui eseguire le istruzioni dipende ovviamente dal tipo di istruzione;
- analizzeremo il caso di tre istruzioni : trasferimento dati, somma e salto

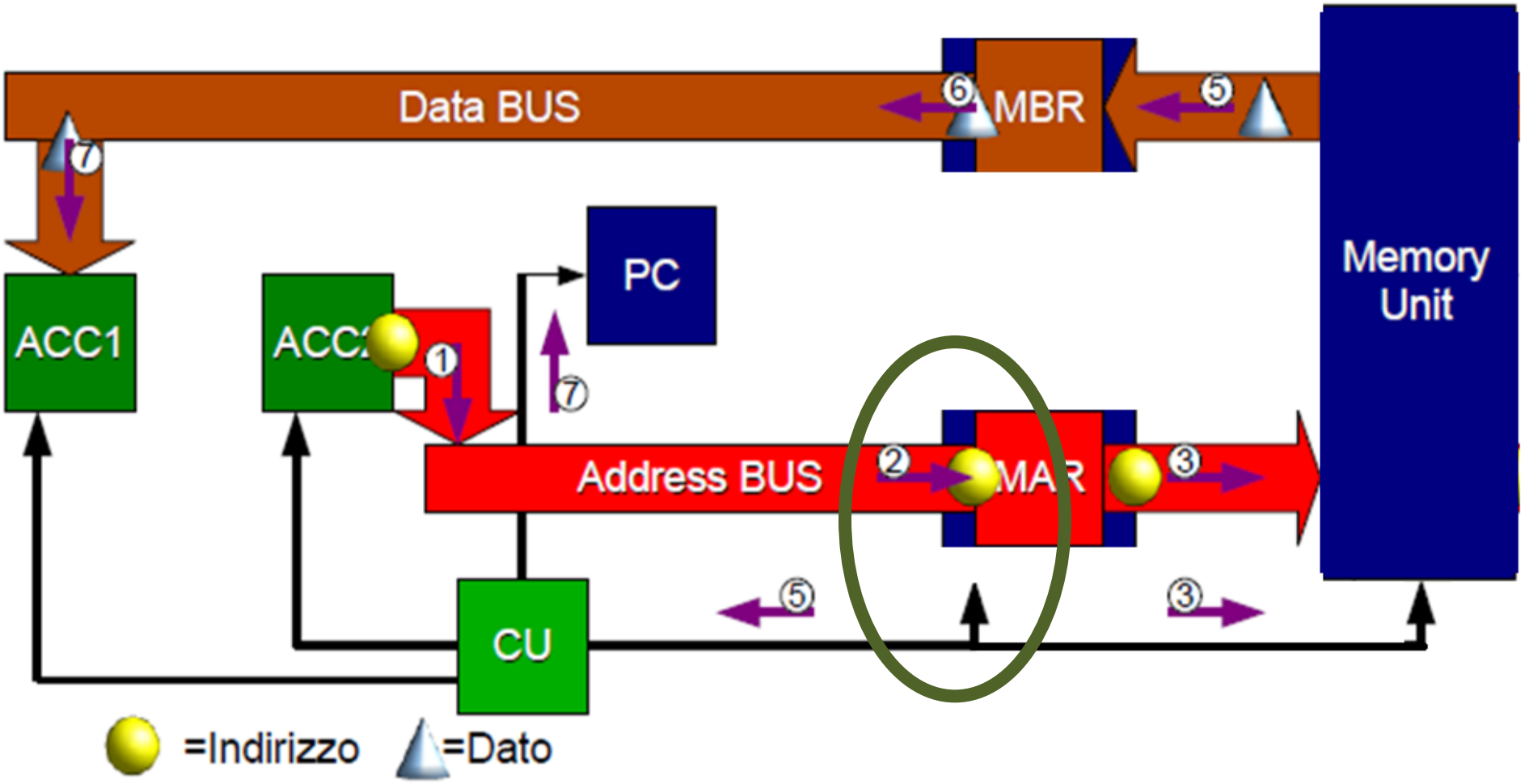
EXECUTE: TRASFERIMENTO DI DATI

Esecuzione dell'istruzione di memoria «Caricamento su ACC1 del dato il cui indirizzo è su ACC2»

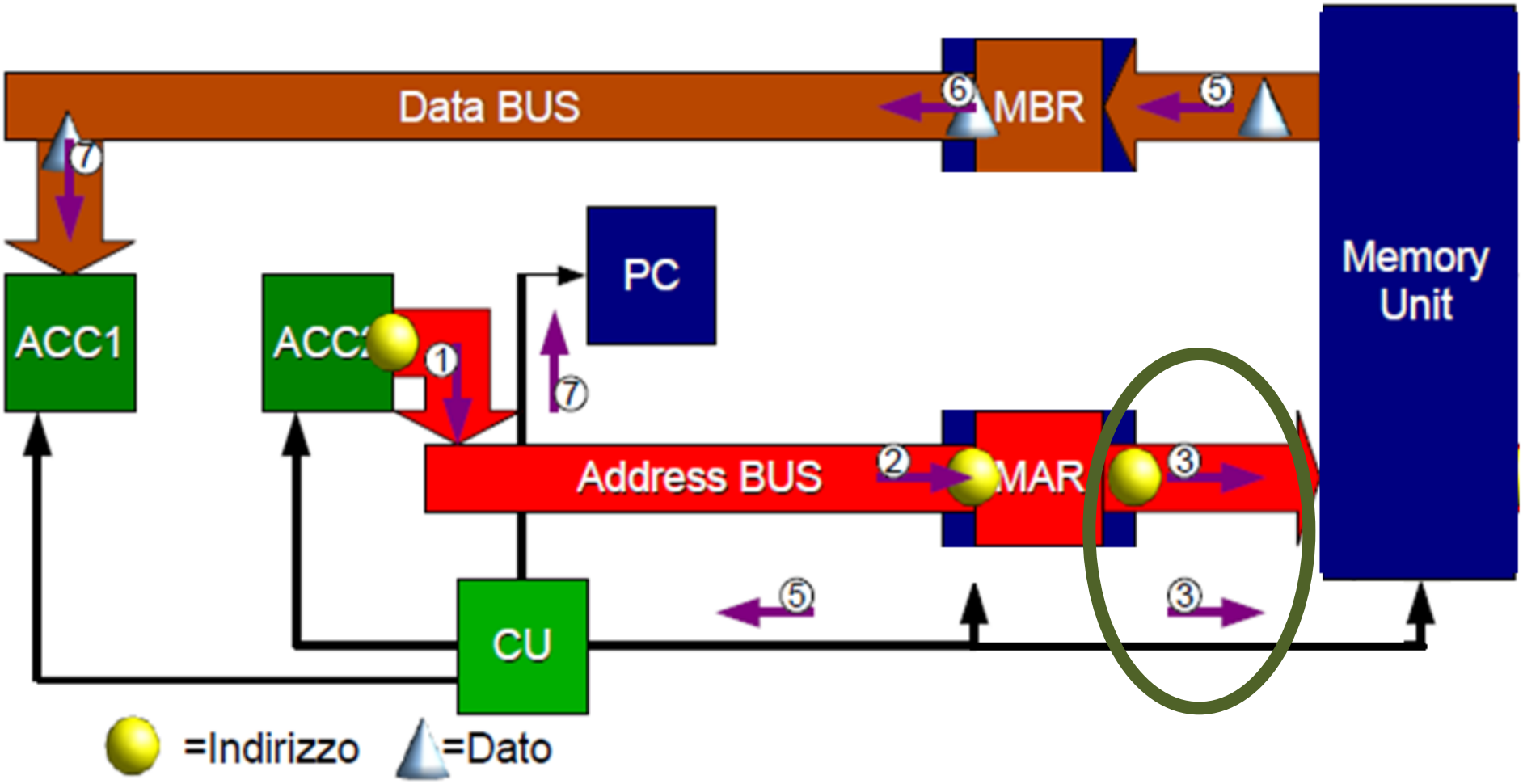




1.La CU ordina ad ACC2 di copiare il proprio contenuto sull'address bus (ossia l'indirizzo della prossima istruzione)

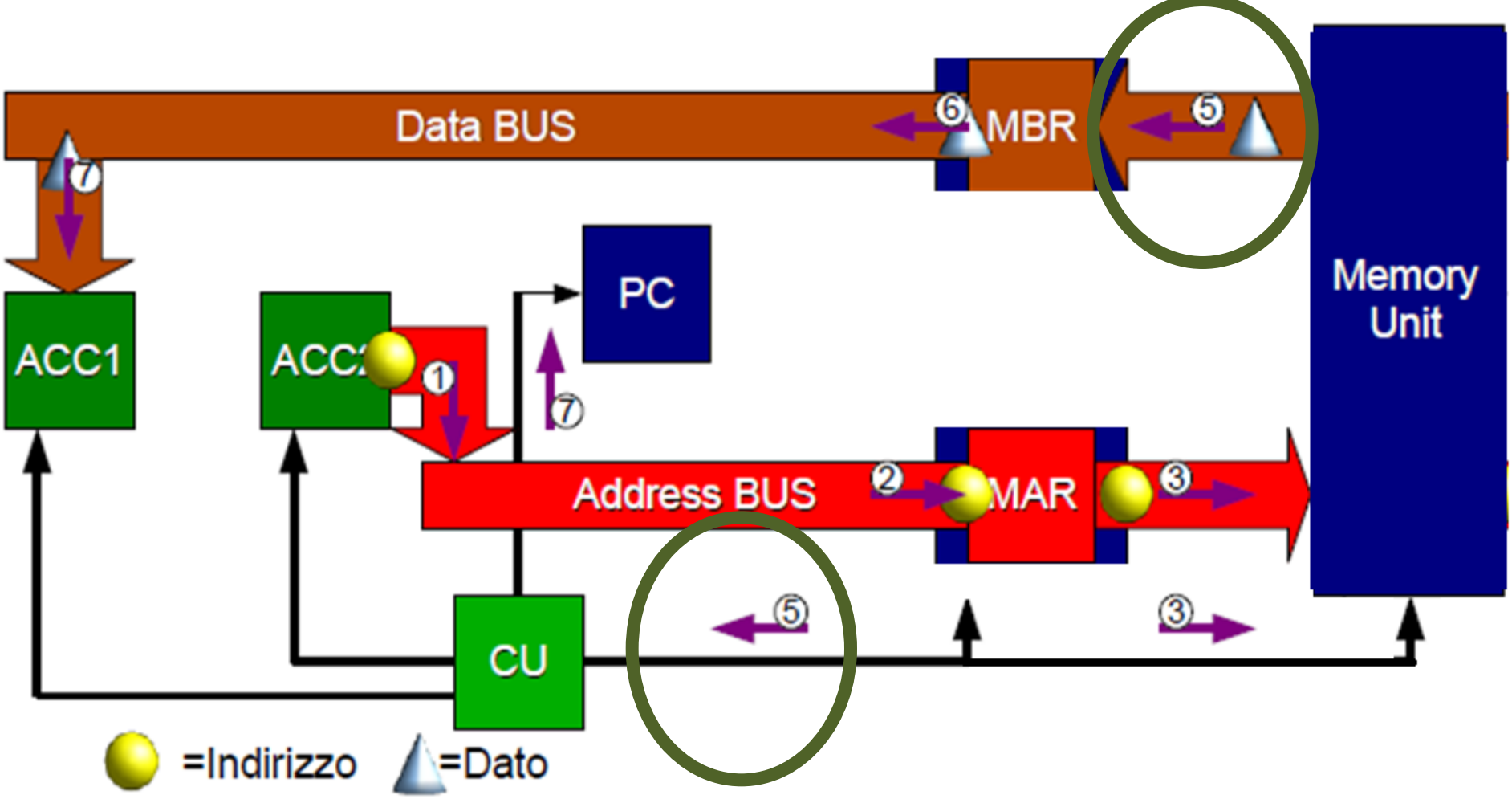


2. La CU ordina al MAR di immagazzinare l'indirizzo presente nell'address bus

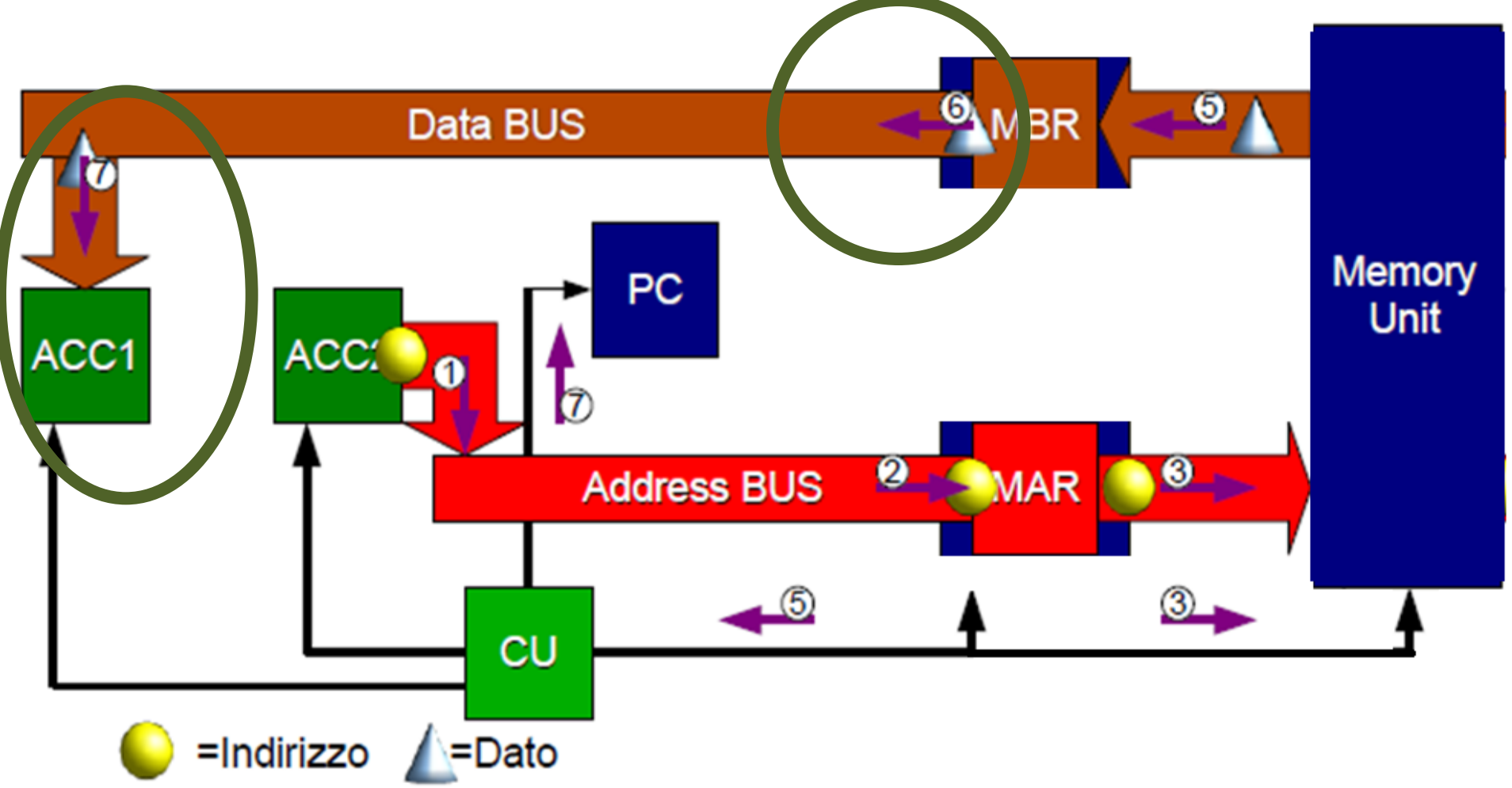


3. La CU ordina all'unità di memoria di effettuare una lettura in memoria

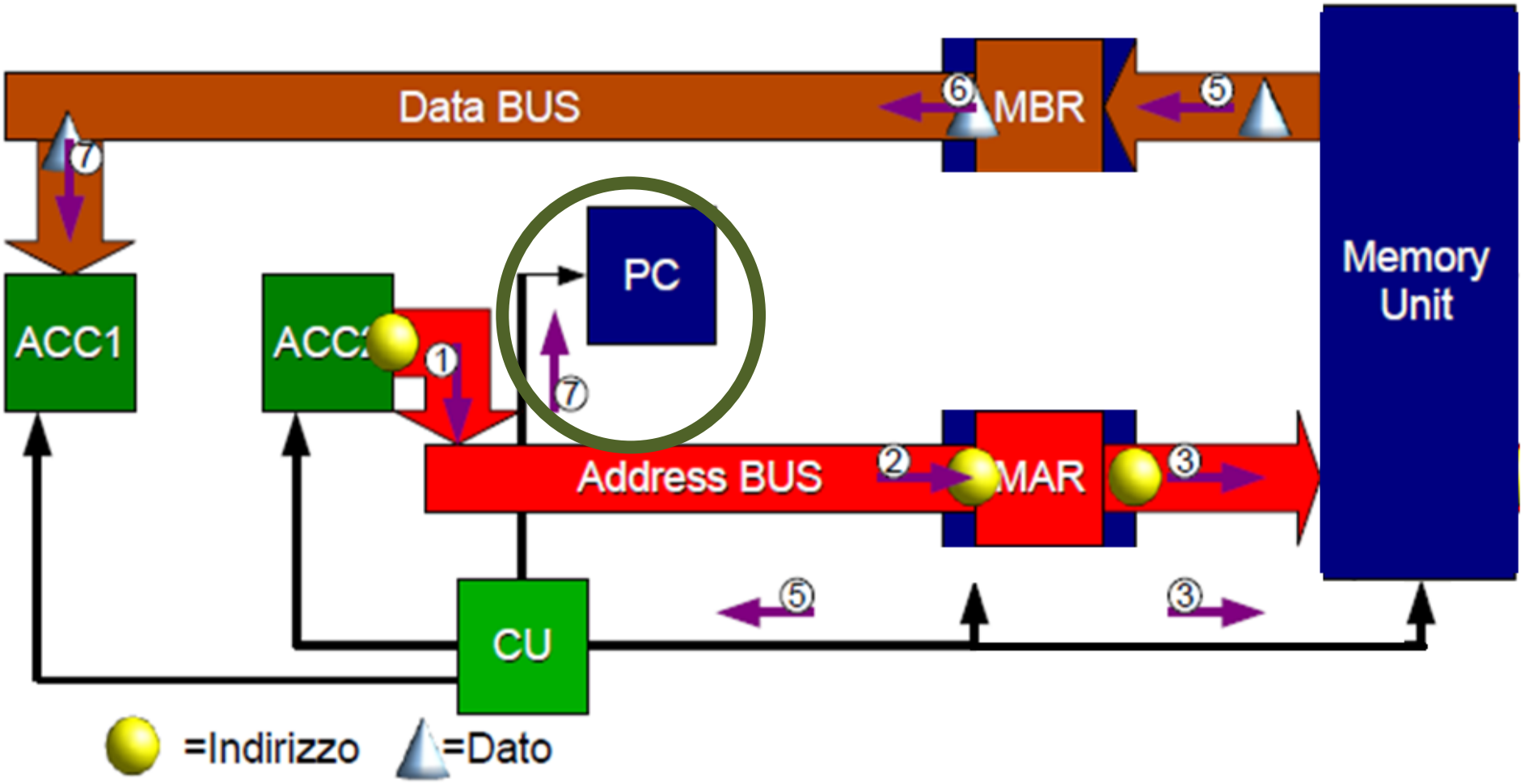
3. La memoria copia l'indirizzo del MAR sull'address bus esterno



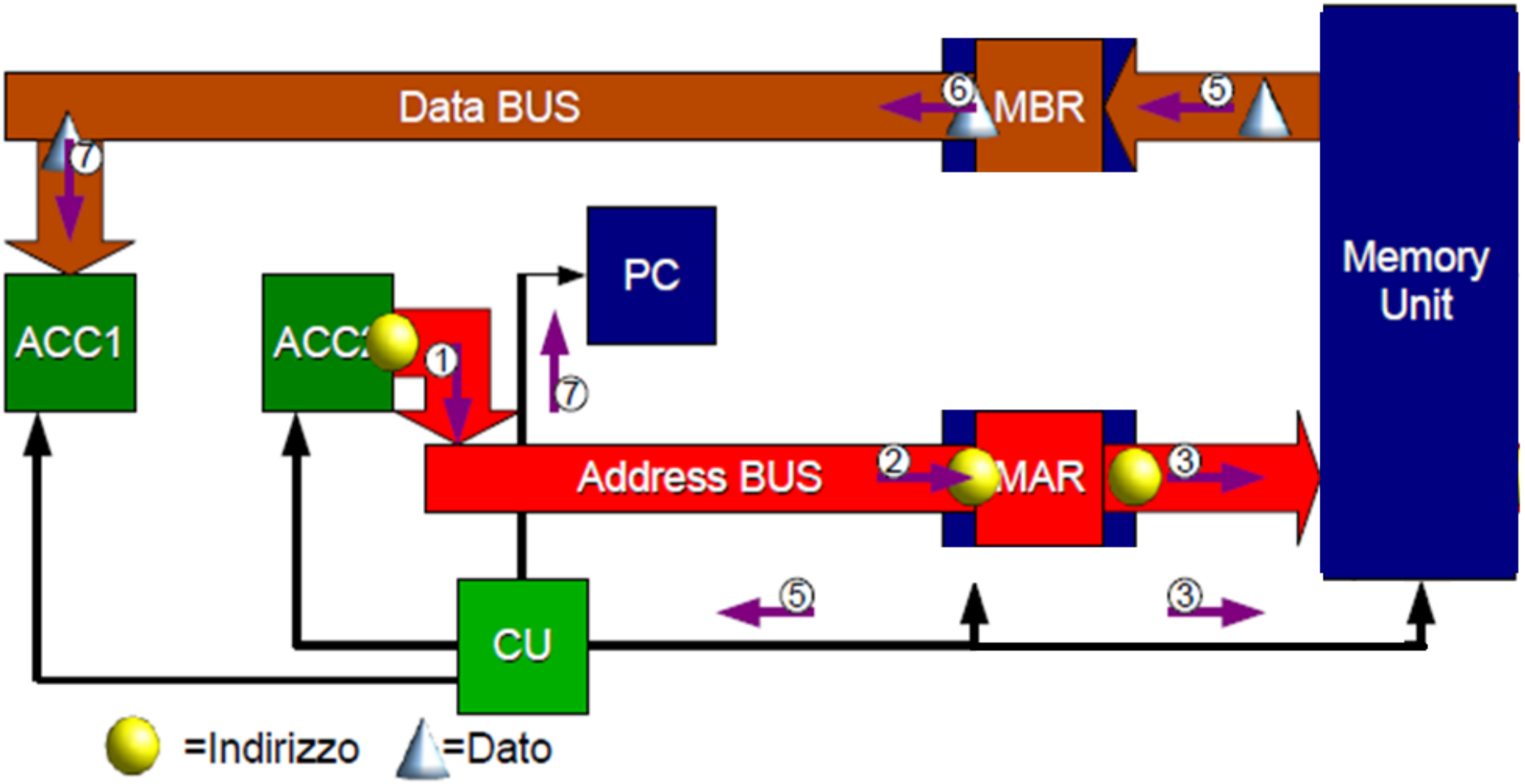
5. Quando la memoria ha recuperato il dato (in figura rappresentato da un triangolino azzurro), lo copia nell'MDR, e avverte la CU che il dato è stato recuperato



6. La CU ordina all'MDR di copiare il dato nel data bus interno
7. La CU ordina all'ACC1 di copiare il dato contenuto nell'MDR

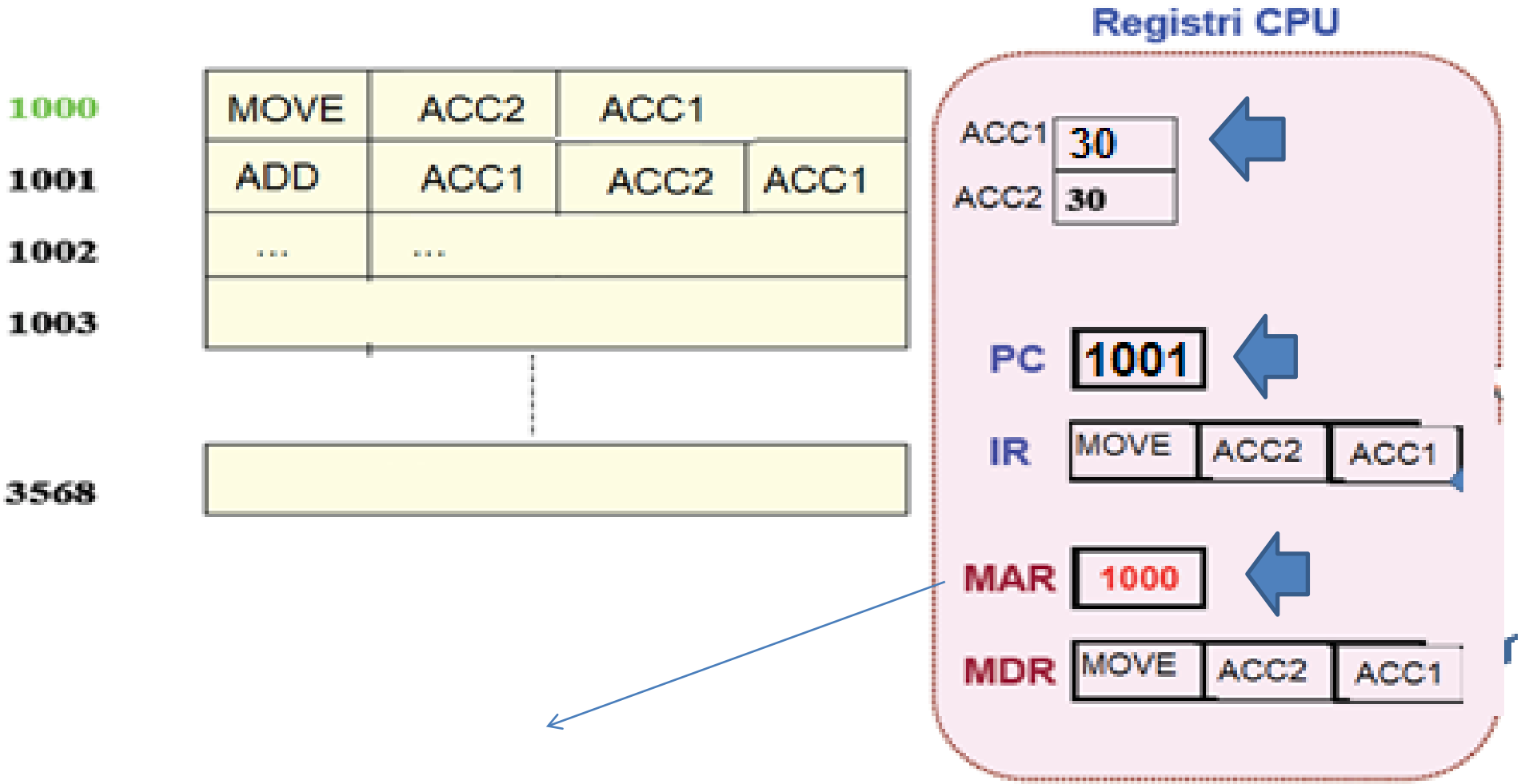


8. La CU ordina al PC di incrementare il proprio contenuto di fatto puntando alla prossima istruzione



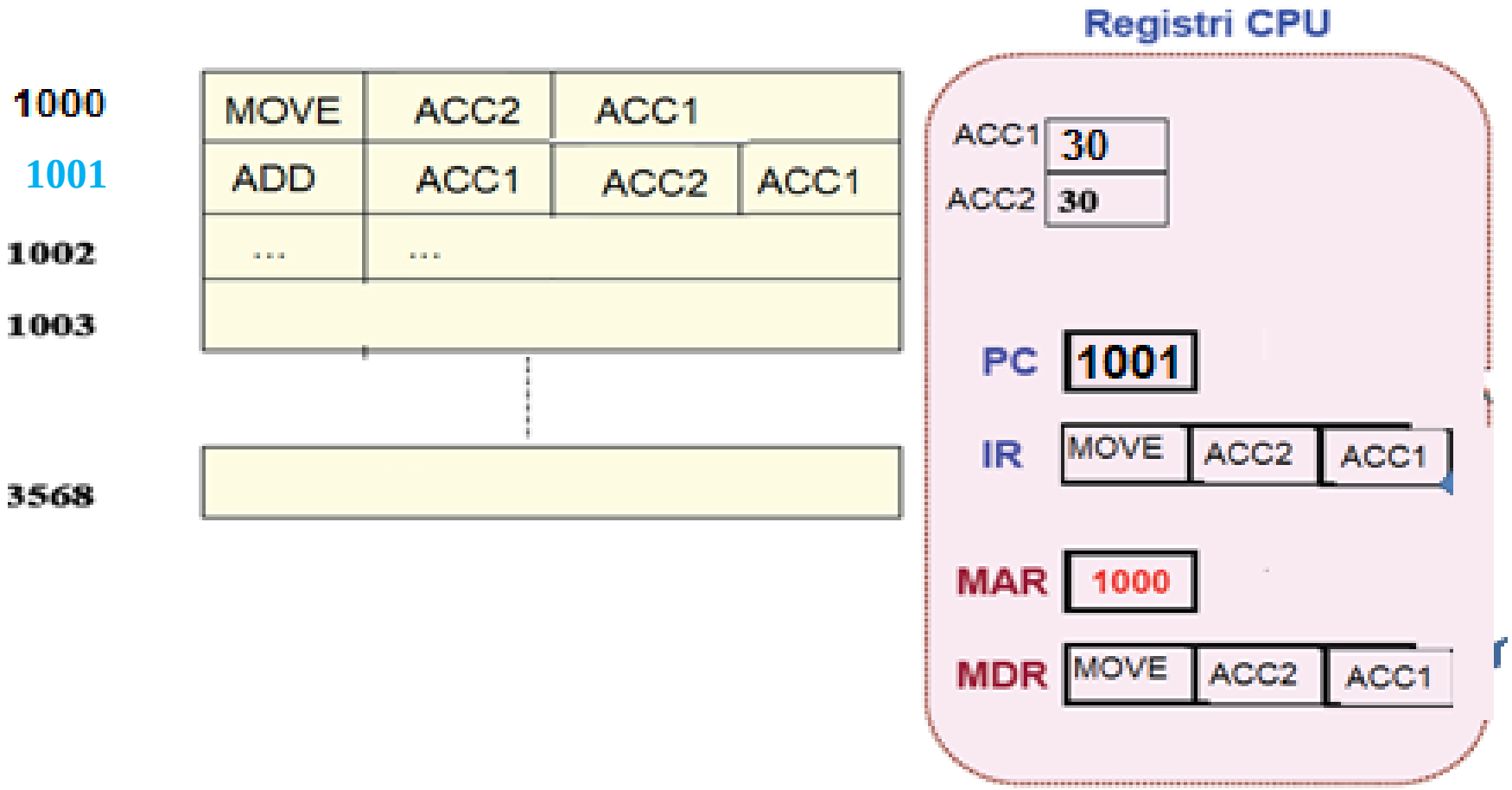
A questo punto l'operando è caricato in ACC1, l'istruzione è stata eseguita e il PC contiene l'indirizzo della successiva istruzione.

DECODE + EXCUTE



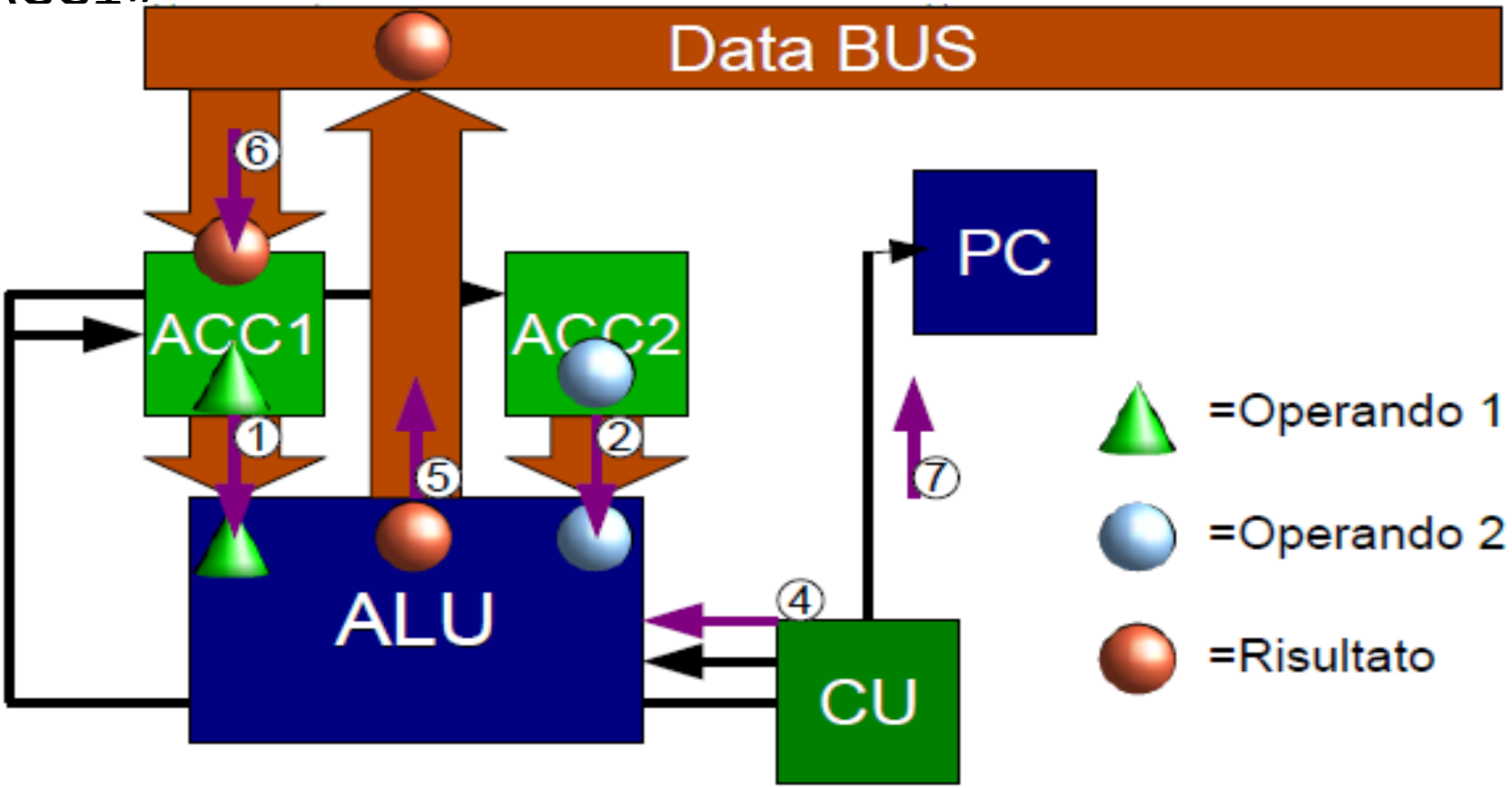
Nota: MAR viene aggiornato nella fase di fetch dell'istruzione successiva

Esecuzione istruzione ***1001***

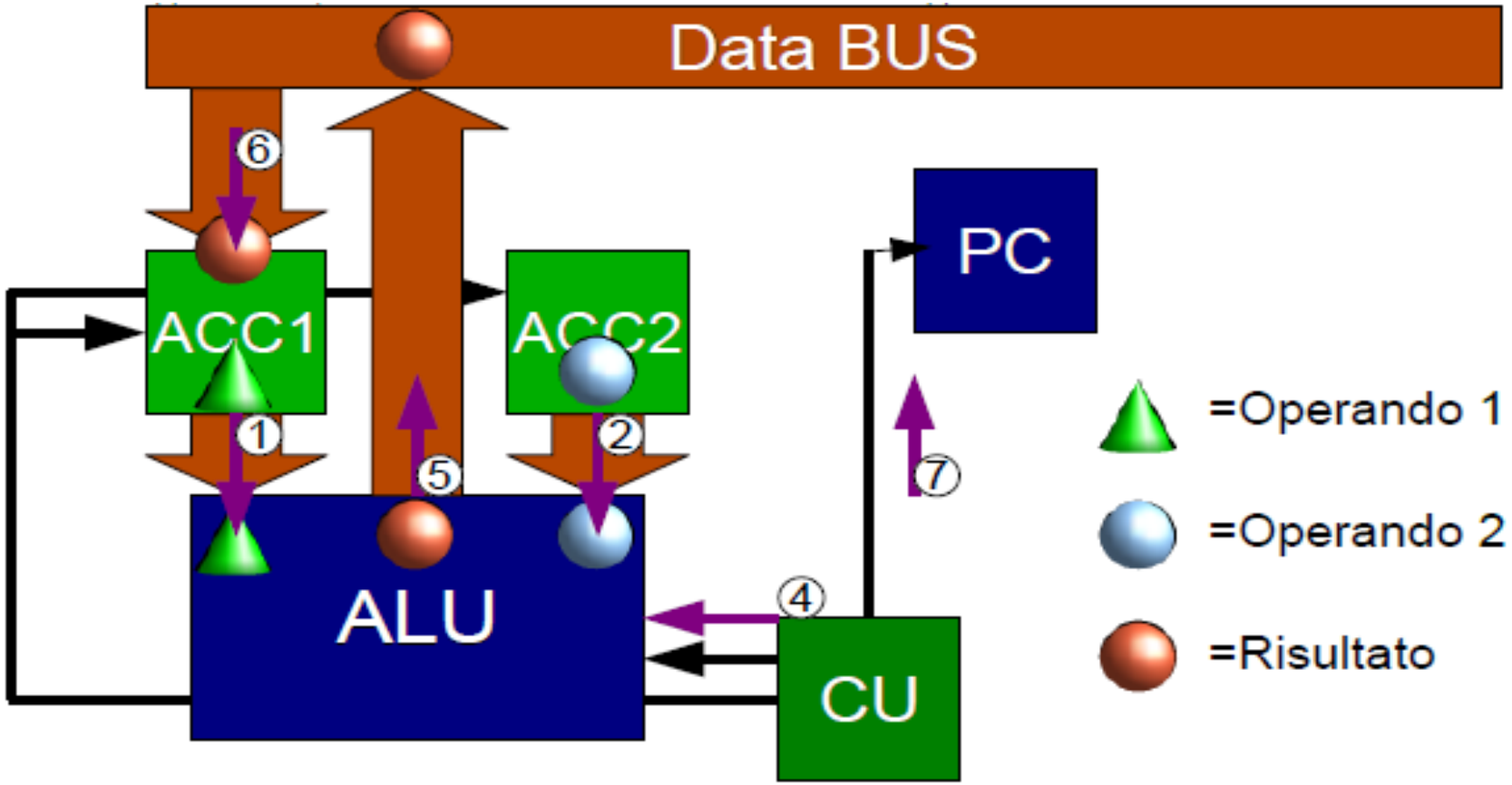


EXECUTE: SOMMA

Esecuzione dell'istruzione di calcolo «Somma tra il contenuto di ACC1 e ACC2 e scrittura del risultato su ACC1»

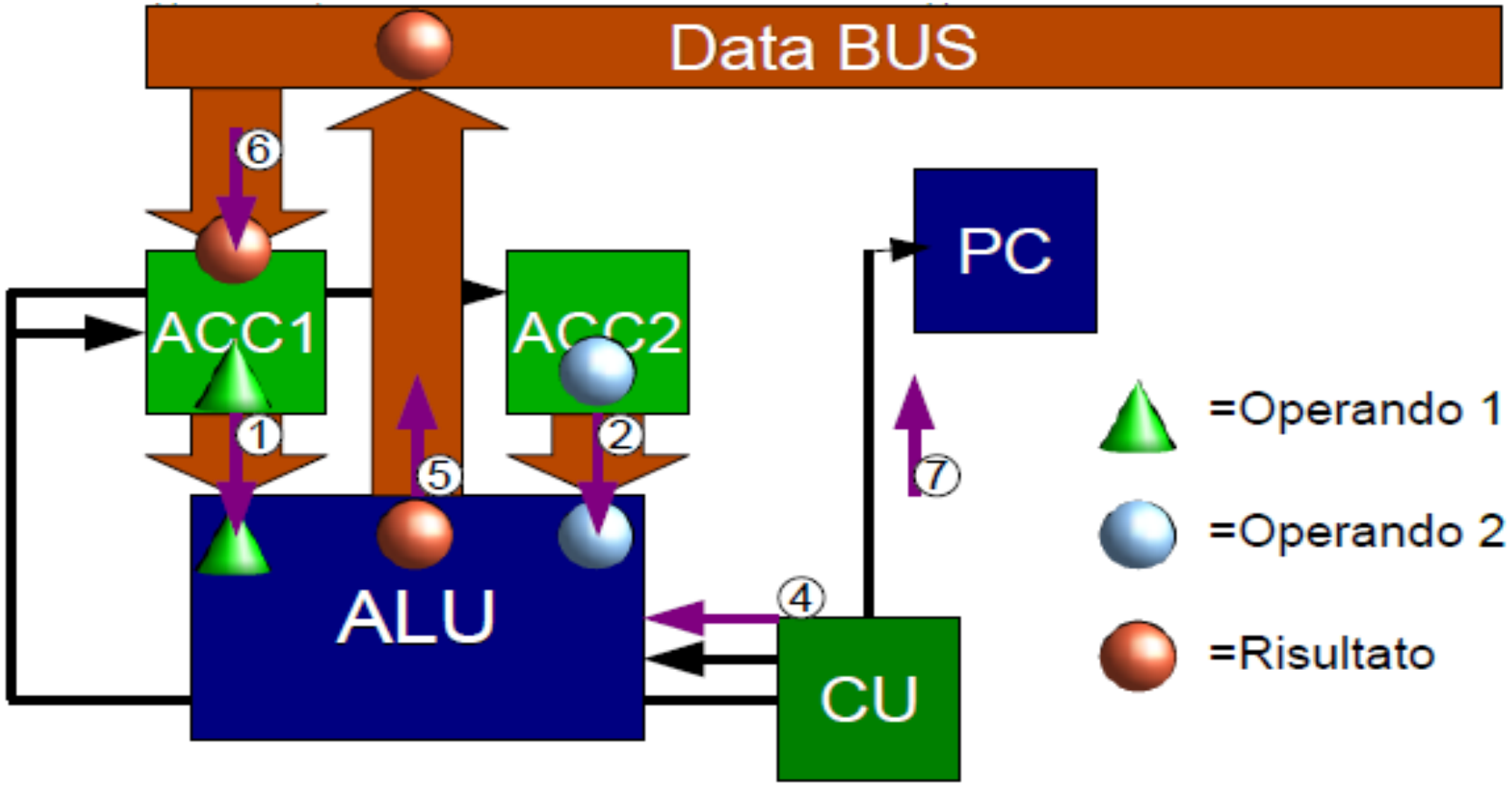


EXECUTE: SOMMA



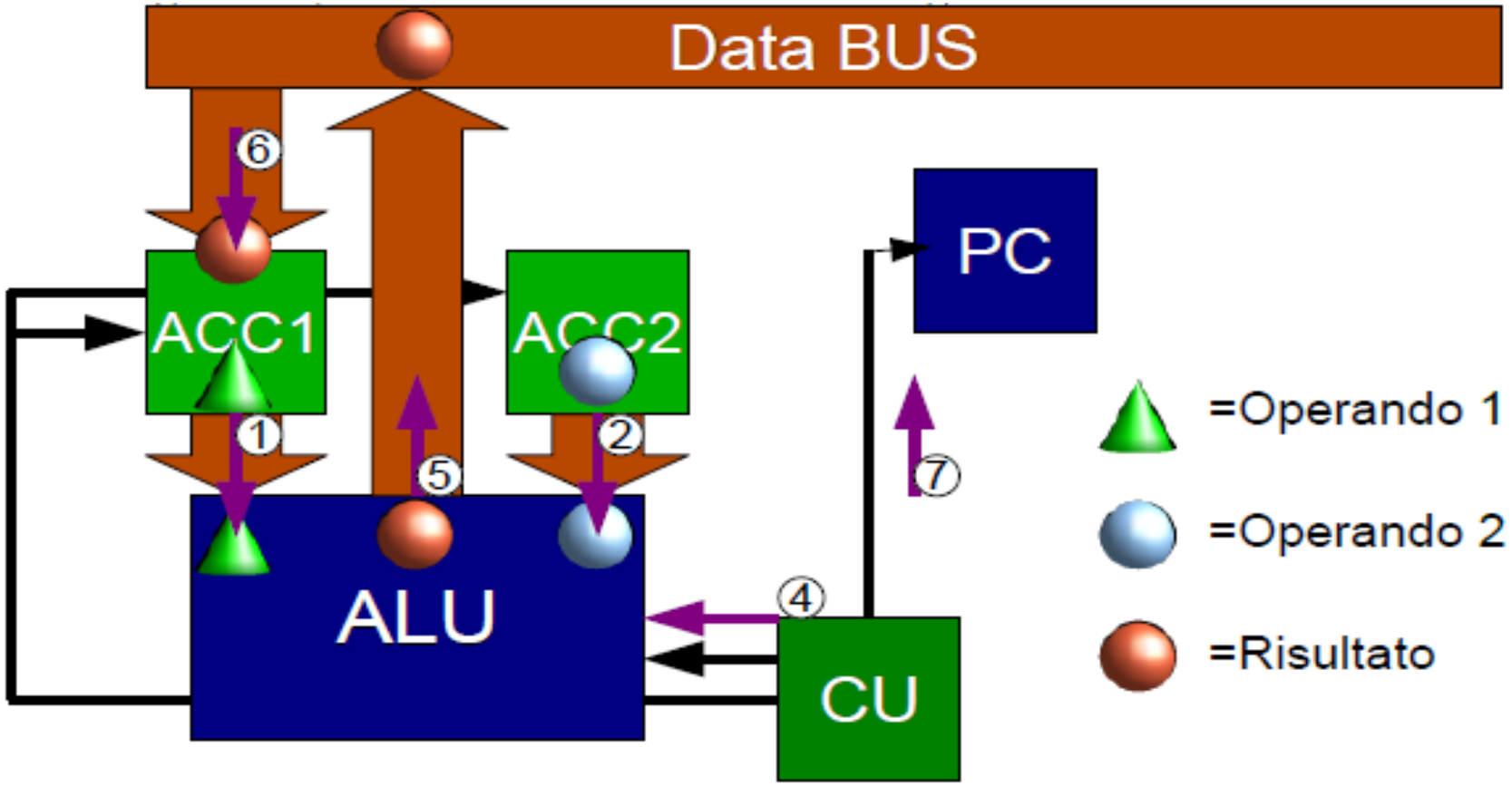
1. La CU ordina ad ACC1 di passare il proprio operando alla ALU
2. La CU ordina ad ACC2 di passare il proprio operando alla ALU

EXECUTE: SOMMA



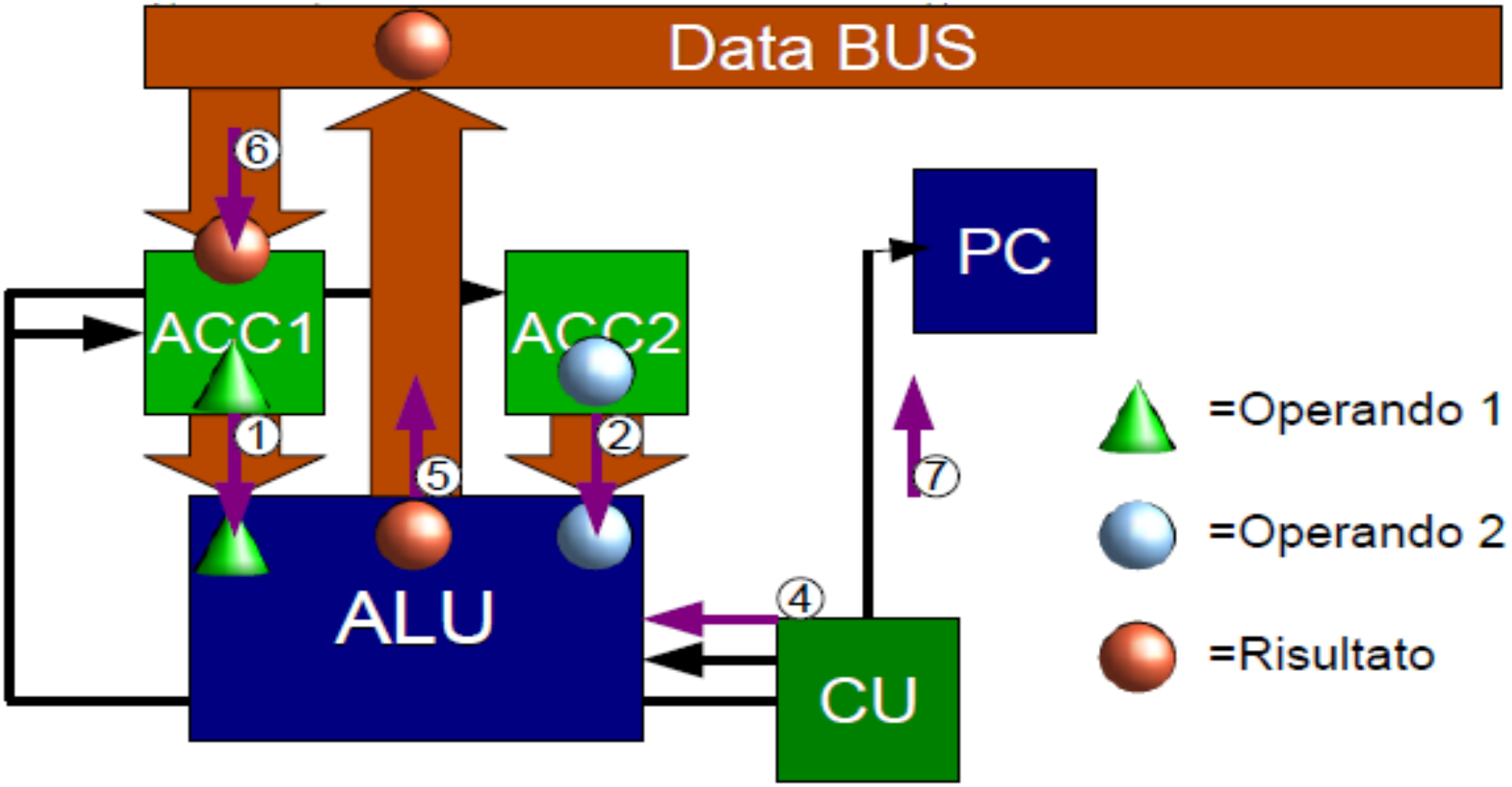
3. La CU ordina alla ALU di leggere i due operandi da sommare
4. La CU ordina alla ALU di sommare gli operandi

EXECUTE: SOMMA



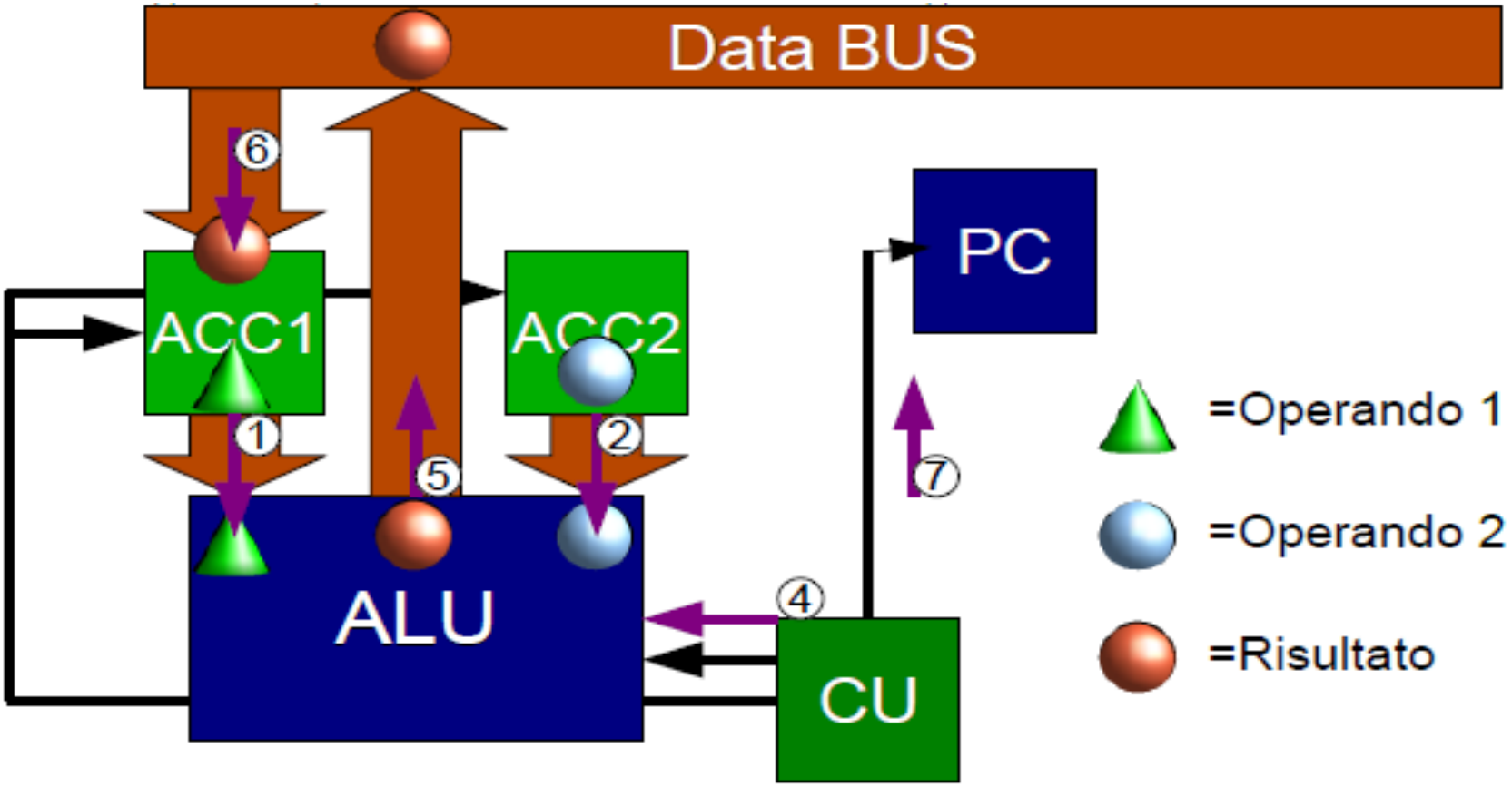
5. A risultato pronto la ALU avvisa la CU che in uscita è pronto il risultato
6. La CU ordina a ACC1 di memorizzare il dato in uscita dalla ALU

EXECUTE: SOMMA



7. La CU ordina al PC di incrementare il proprio contenuto di fatto puntando alla prossima istruzione

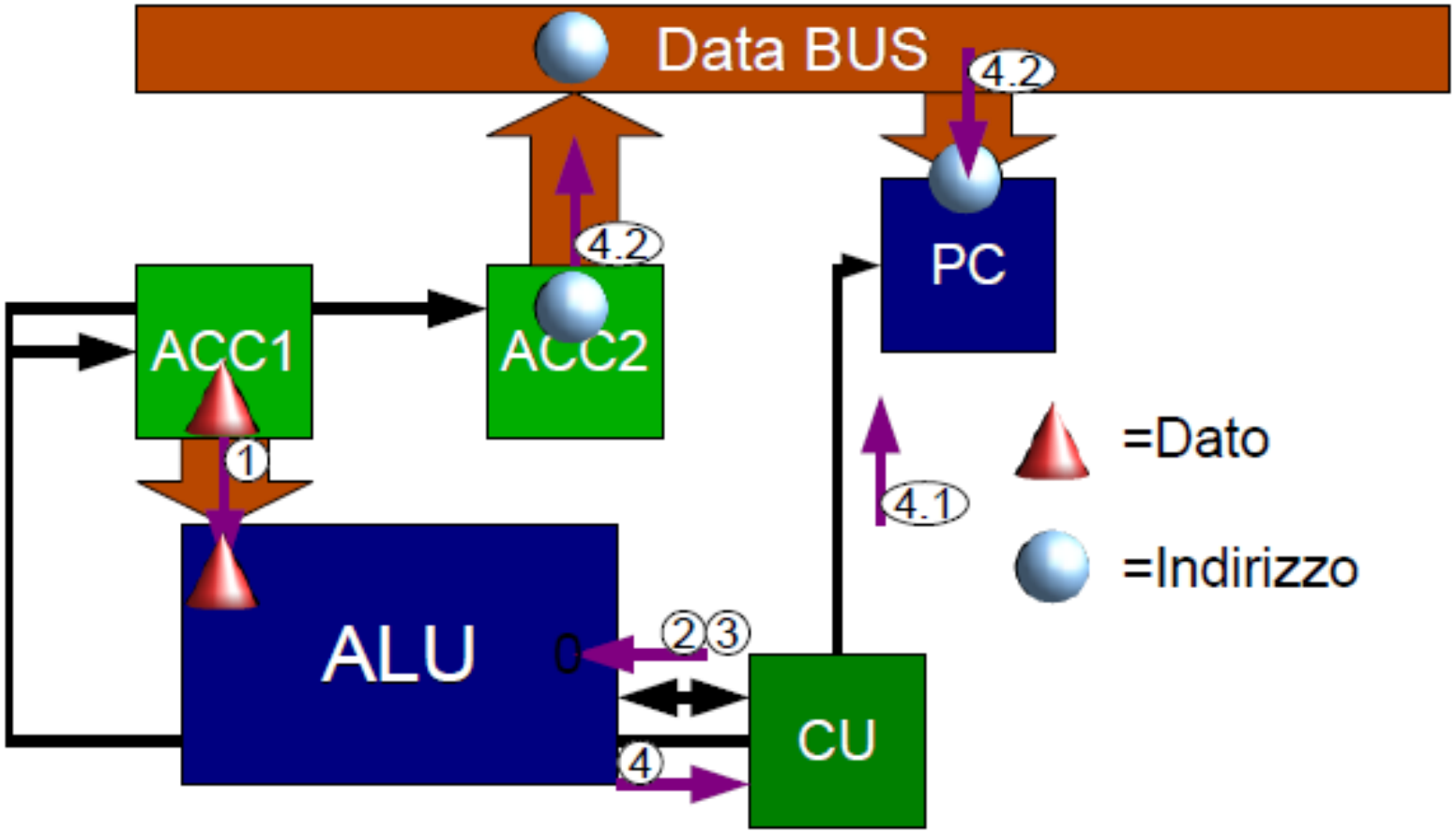
EXECUTE: SOMMA

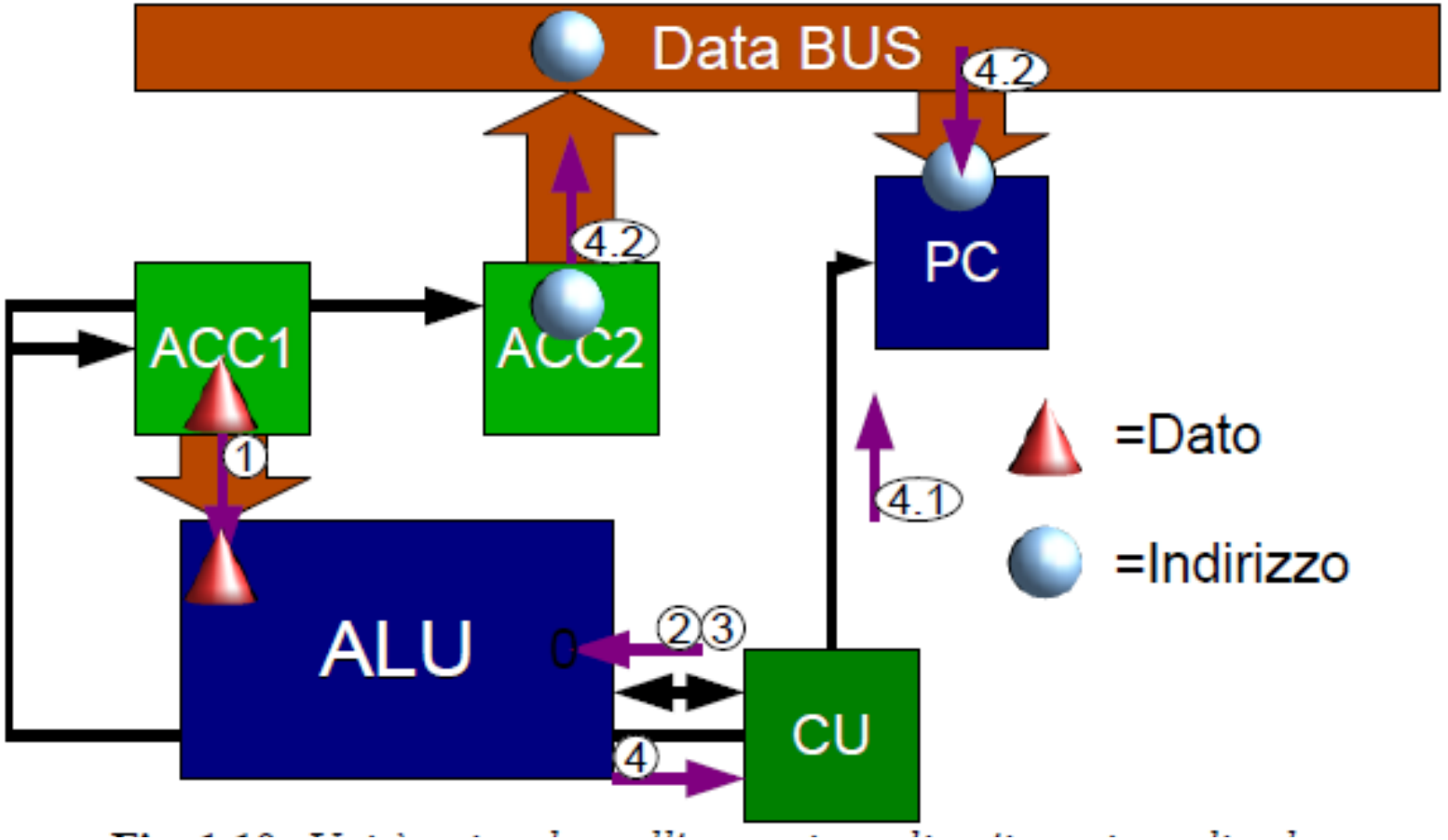


A questo punto l'istruzione è stata eseguita e il PC contiene l'indirizzo della successiva istruzione.

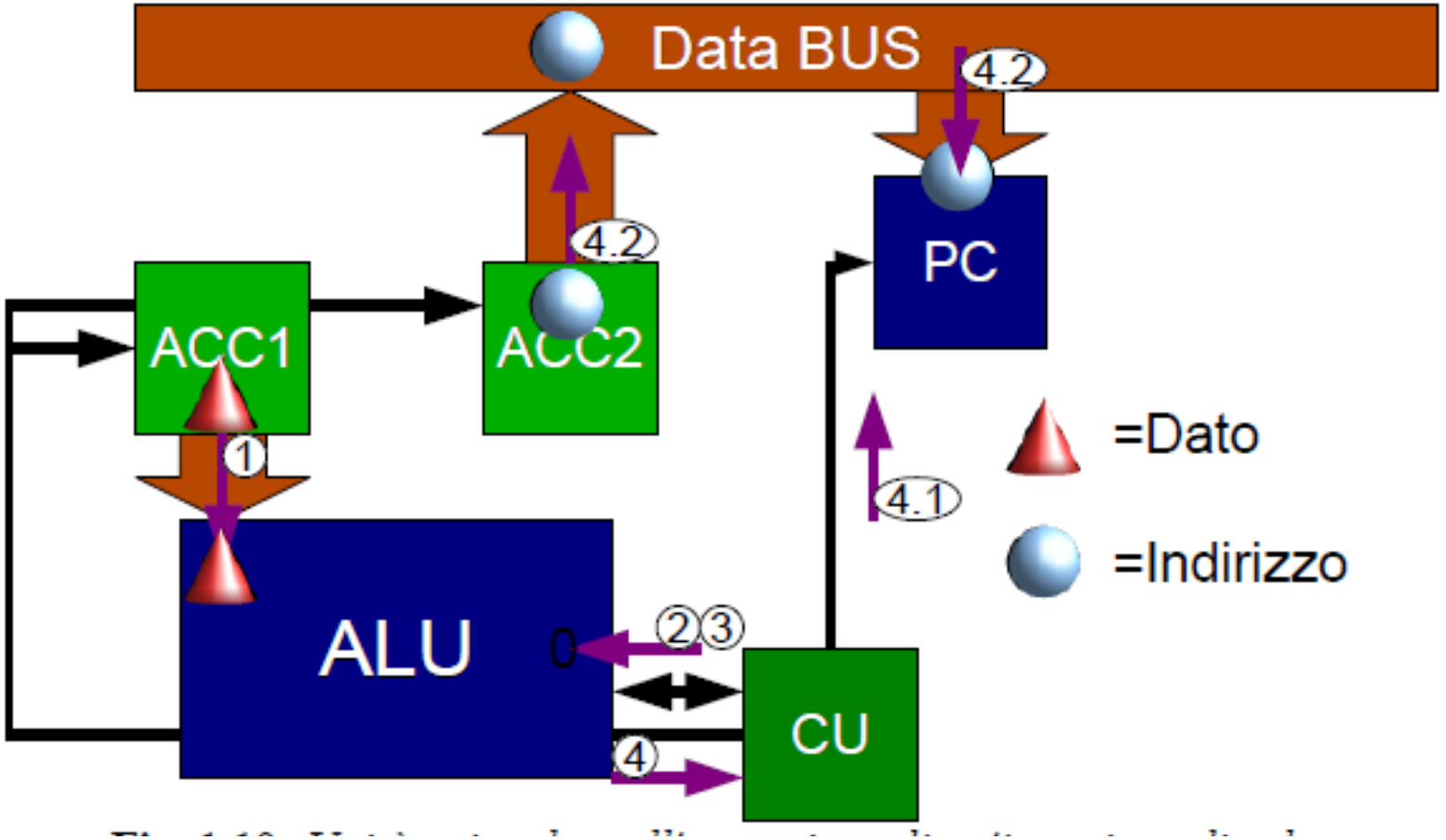
EXCUTE: SALTO

Esecuzione dell'istruzione di salto condizionato
«Se ACC1 = 0 salta all'indirizzo specificato da ACC2»

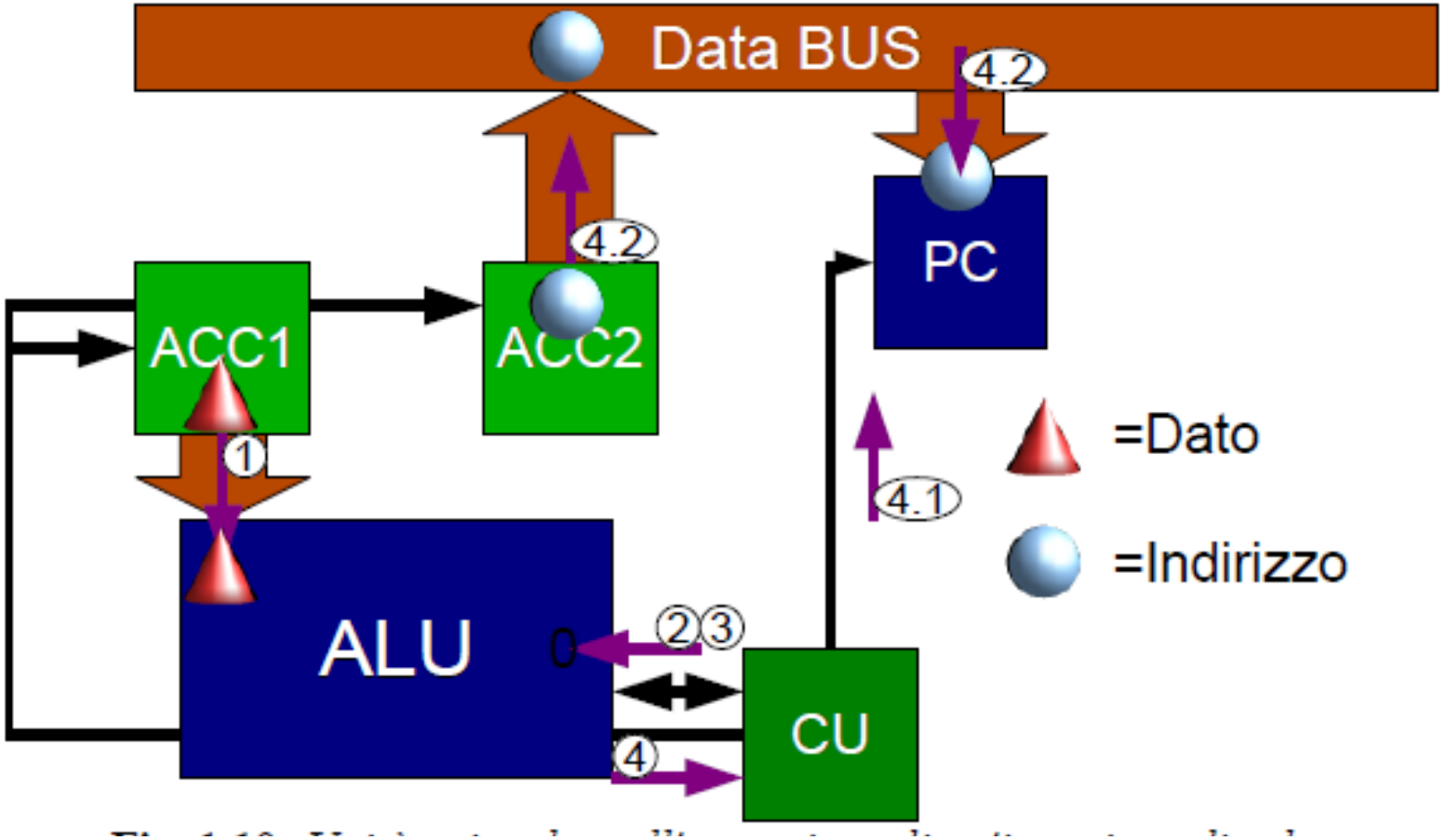




1. La CU ordina ad ACC1 di passare il proprio operando alla ALU
2. La CU ordina alla ALU di leggere da ACC1 e passa 0 come secondo operando
3. La CU ordina alla ALU di confrontare i due operandi



4. A risultato pronto la ALU lo comunica alla CU
1. Se il risultato è ACC1 ≠ 0 allora la CU ordina al PC di incrementare il proprio contenuto di fatto puntando alla prossima istruzione
2. Le il risultato è ACC1 = 0 allora la CU ordina ad ACC2 di copiare il l'indirizzo che contiene sul data bus e poi al PC di caricare l'indirizzo presente sul data bus



A questo punto l'istruzione è stata eseguita e il PC contiene l'indirizzo della successiva istruzione.