Il Ciclo Fetch-Decode-Execute

Linguaggio Assembler

- Op – code (Mnemonico)
- Operando

```
#assembly code program

0000 LDA-24       #loads
0001 ADD-25       #adds
0002 HLT          #stops

#end of program
```

Registro Accumulatore

- Registro piu' importante della CPU
- Spesso indicato con la lettera A

LDA – 24

		Operando
L D A	mnemonico	- 24
LoaD	Metti	il contenuto
A	nell'accumulatore	della cella 24

Registro Accumulatore

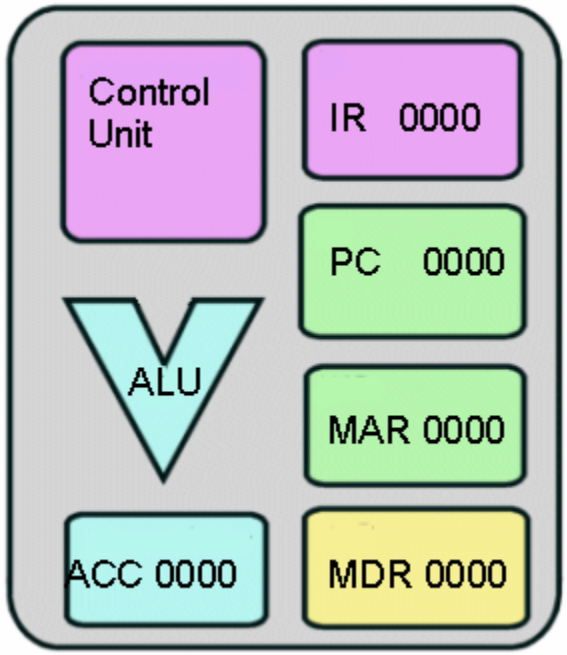
Spesso Registro partner obbligato nelle operazioni.
Uno dei due operandi deve stare in A

ADD – 25

		Operando
ADD	mnemonico	- 25
ADD	Addiziona	con il contenuto
	l'accumulatore	della cella 25

Registri della CPU

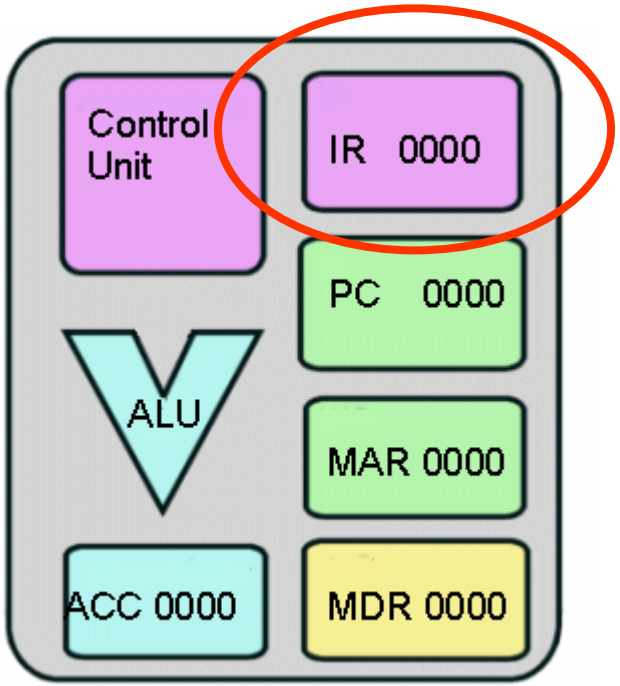
- ACC accumulatore
- IR registro istruzione
- PC program counter



- MAR ???
- MDR ???

IR Instruction Register

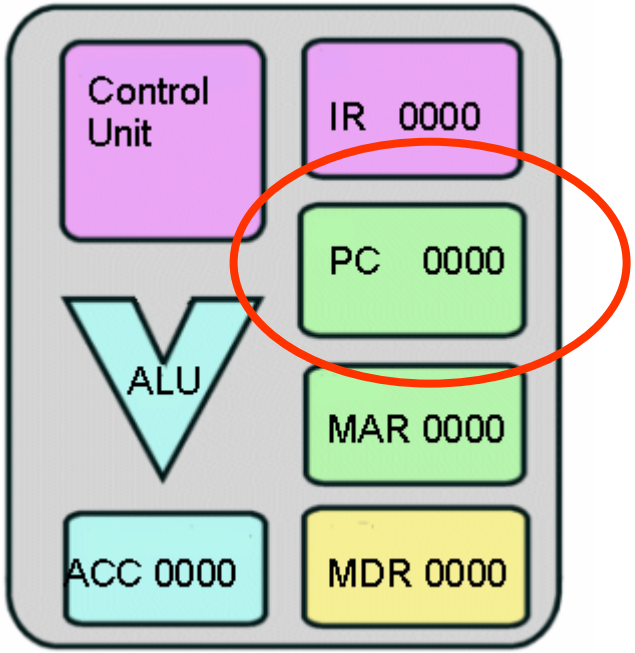
Il codice della istruzione da eseguire deve stare in IR per poter essere decodificato dalla unita' di controllo



PC Program Counter

Detto anche IP (Instruction Pointer)

Contiene l'indirizzo della istruzione da eseguire



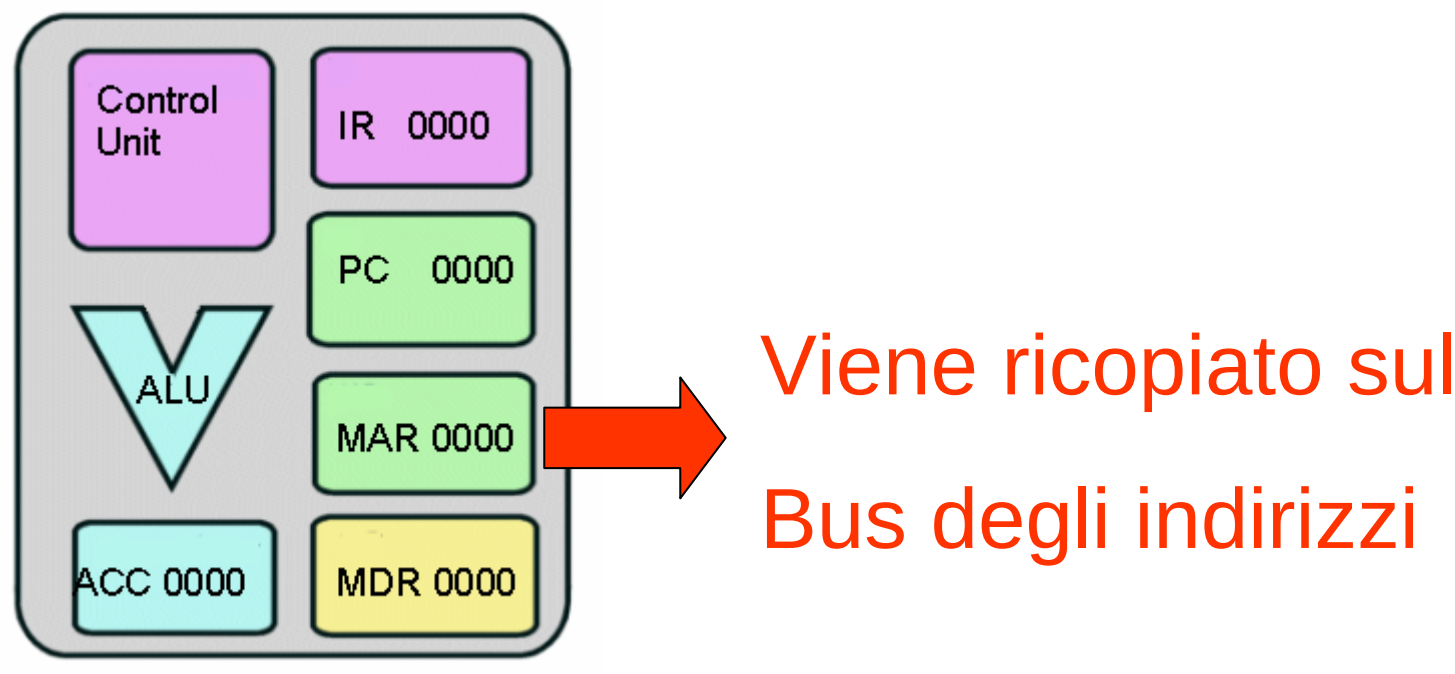
Alla accensione contiene l'indirizzo deciso dal costruttore della CPU

Aumenta automaticamente di 1 puntando alla cella successiva

(sequenza)

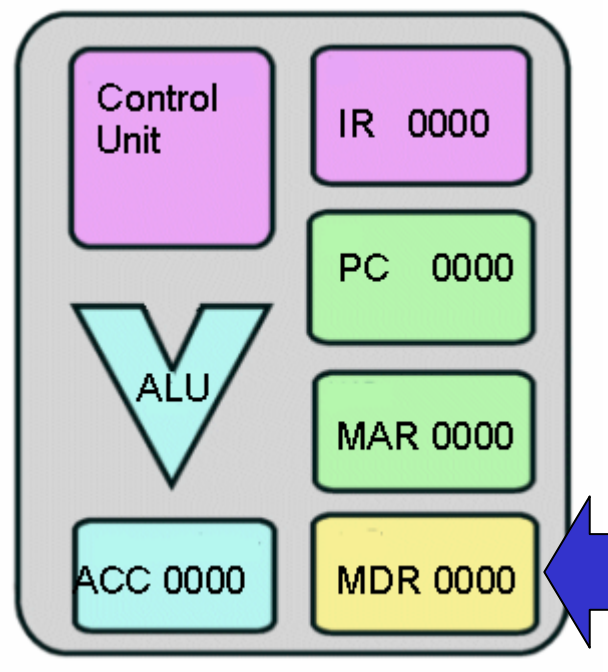
MAR

- Memory Address Register



MDR

- Memory Data Register



Viene ricopiato sul

Bus dati

Le istruzioni sono nella RAM

In un processore CISC, le istruzioni sono di lunghezza variabile. In un RISC hanno tutte la stessa dimensione.

RAM

Assumiamo che siano tutte della stessa lunghezza

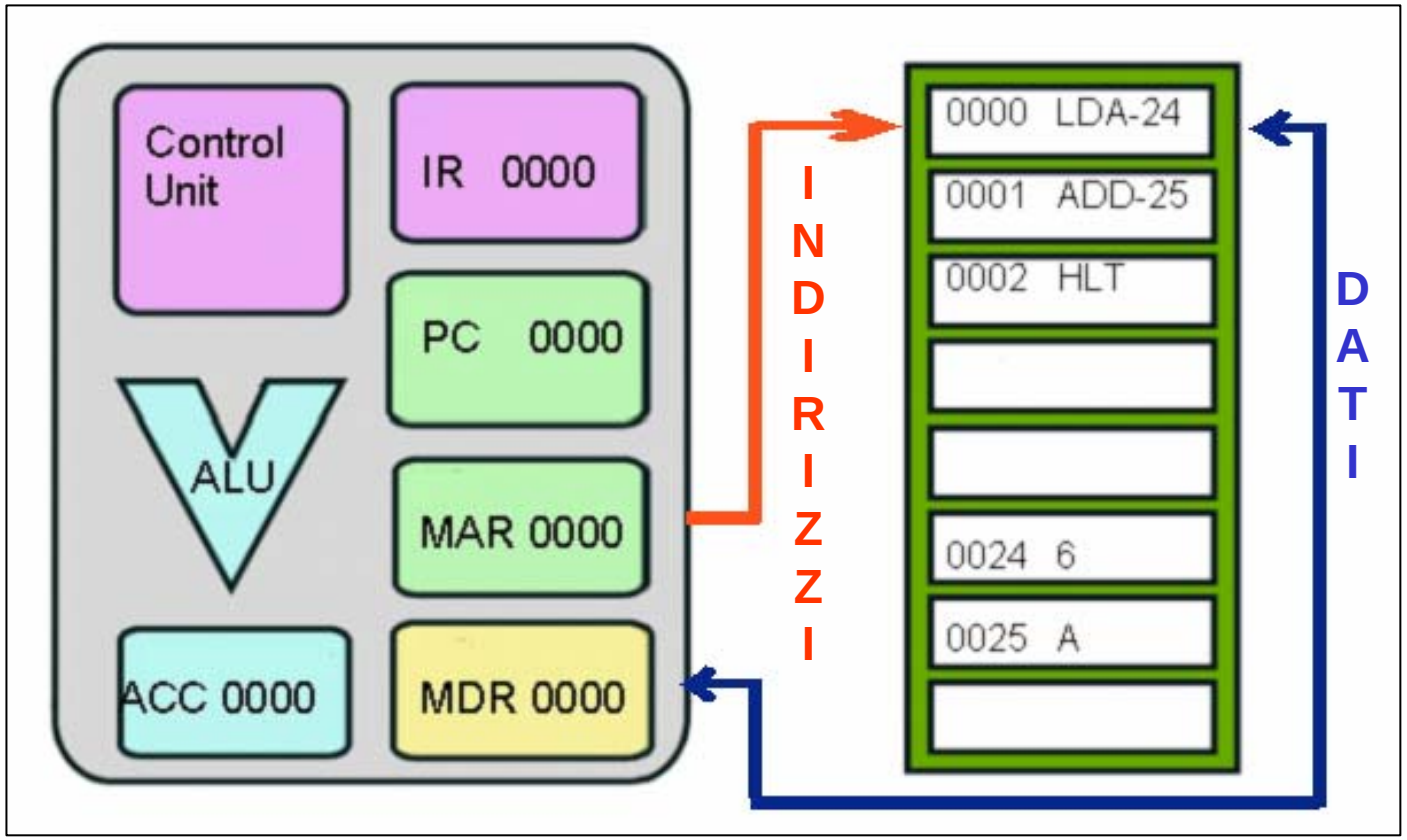
Ognuna occupa una cella di RAM.

Anche i dati sono nella RAM

(Architettura Von Neuman)

FETCH-DECODE-EXECUTE cycle

La CPU per ogni istruzione usa il bus indirizzi e dati per fare le operazioni di fetch decode ed execute



Fasi del ciclo FDE

FETCH - la CPU legge le istruzioni dalla RAM

DECODE – la Control Unit capisce cosa deve fare

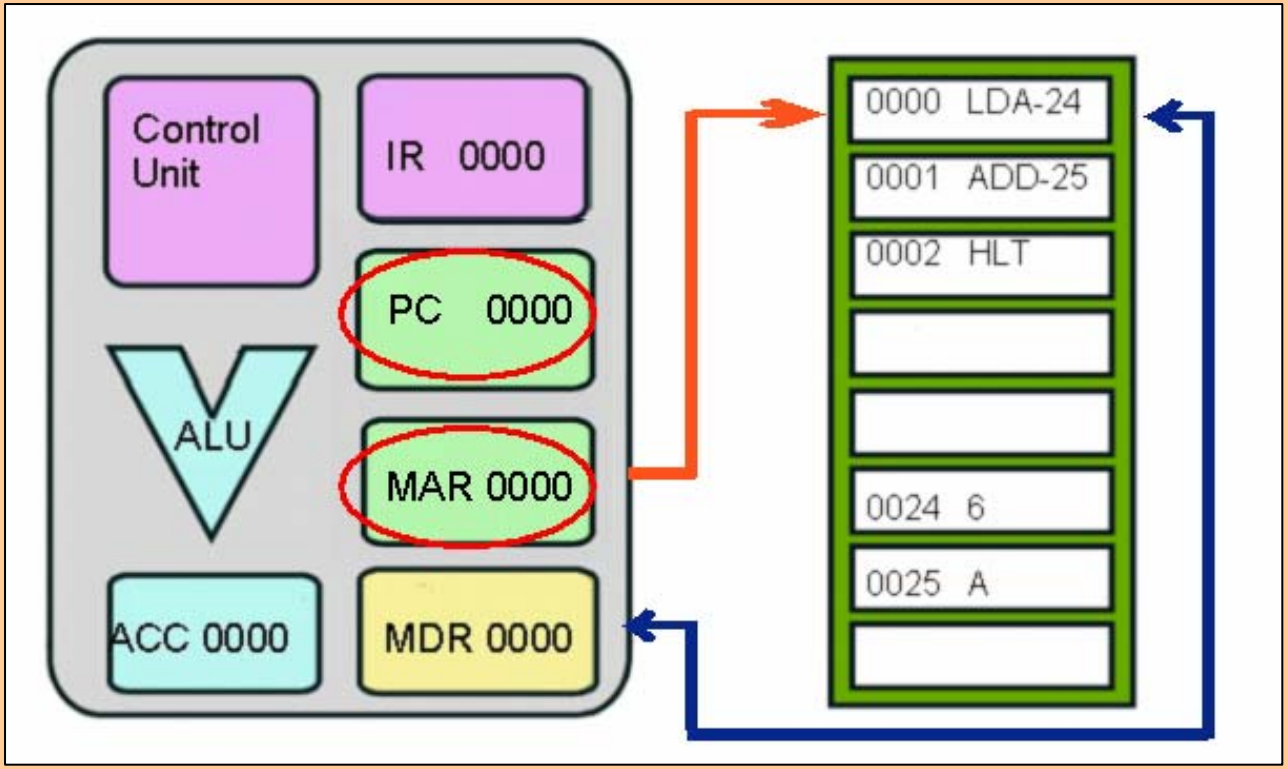
EXECUTE - la istruzione viene eseguita.

Le fasi di FETCH and DECODE non cambiano mai. La fase di EXECUTE è diversa di volta in volta perchè dipende dalla istruzione da eseguire

In questo esempio assumiamo che la macchina stia facendo un “cold boot”
tutti i registri interni sono a 0.

FETCH

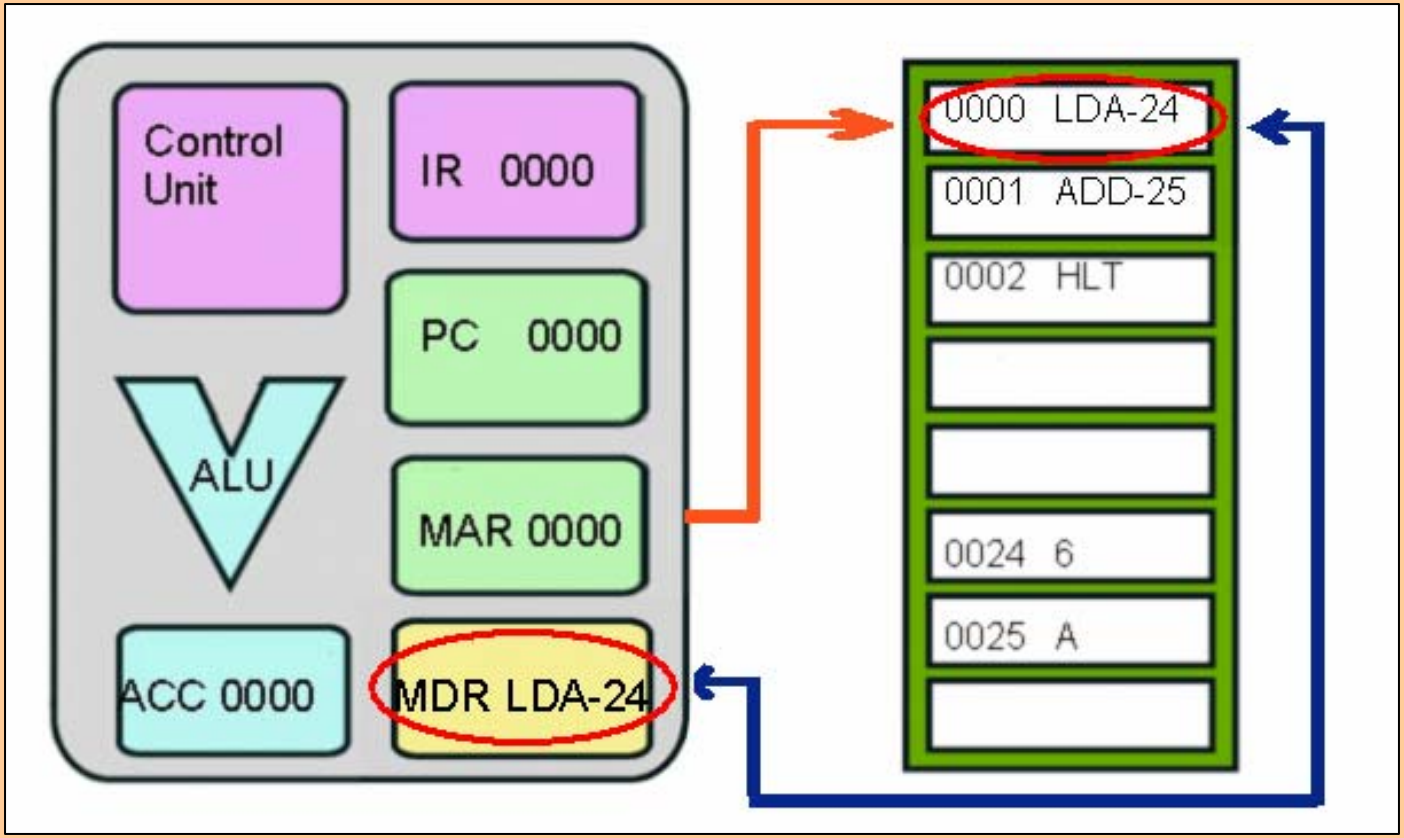
Il Program Counter viene copiato nel MAR.



Quando un nuovo valore entra nel MAR, questo viene ricopiato nel bus degli indirizzi

Così in questo caso stiamo puntando alla locazione 0000.

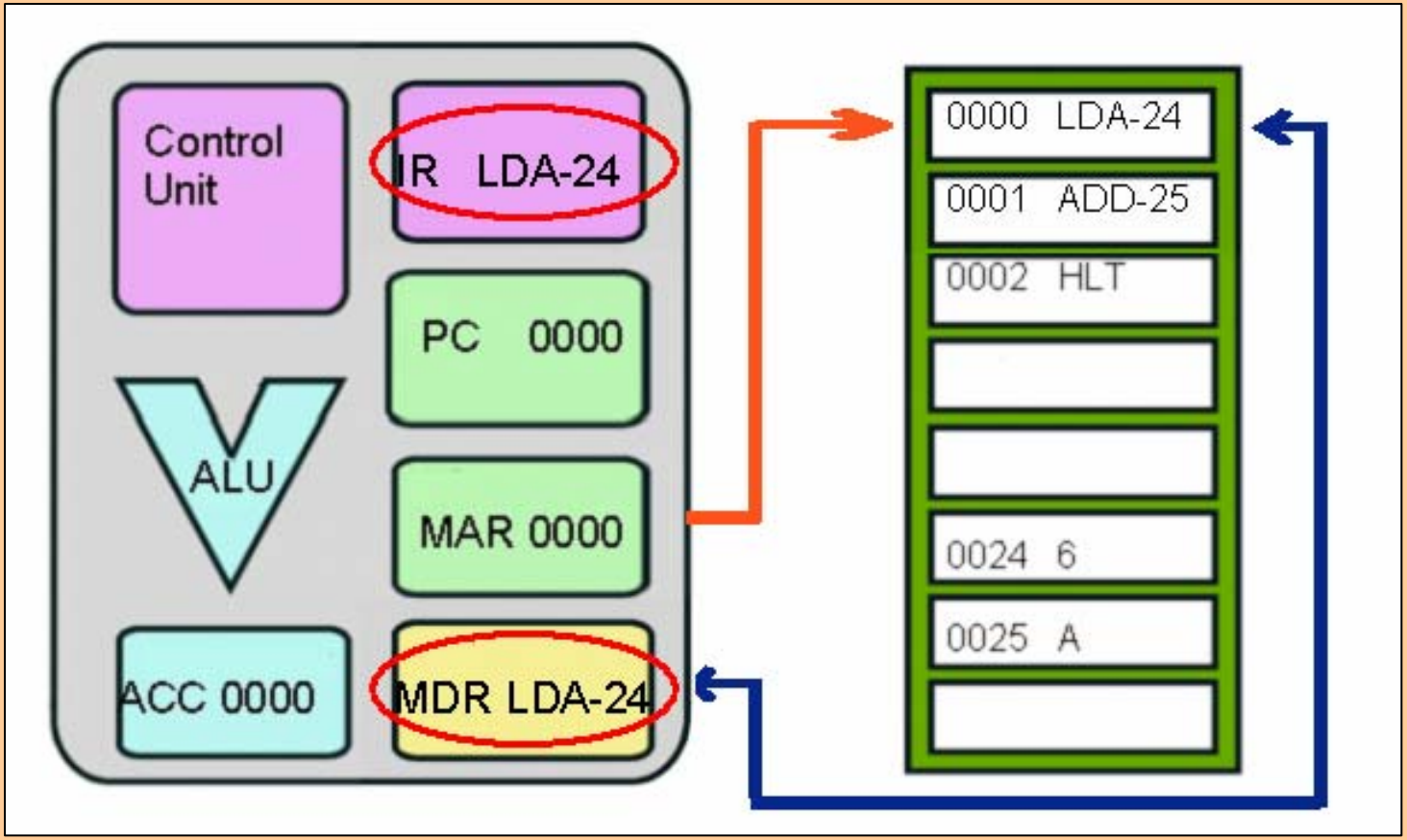
FETCH



Quando la RAM riceve il segnale di lettura da un indirizzo specificato, copia il contenuto della cella “puntata” sul Data Bus. Il contenuto del data bus viene copiato nell’ MDR.

L’ MDR conterrà LDA-24.

FETCH



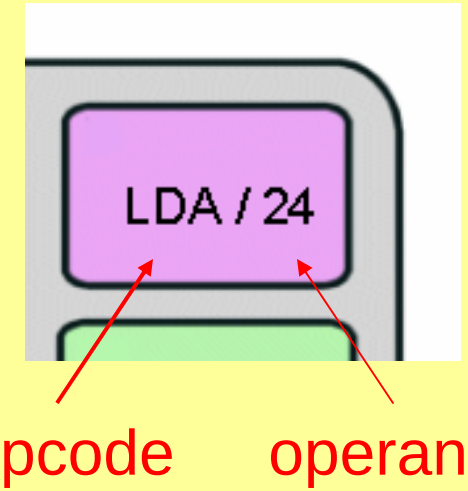
Il contenuto dell' MDR viene trasferito all' Instruction Register, pronto ad essere decodificato.

Questa è la fine della fase di FETCH

DECODE

Il decoder nella unità di controllo prende la istruzione e la divide nelle sue due parti:

opcode e **operandi**.



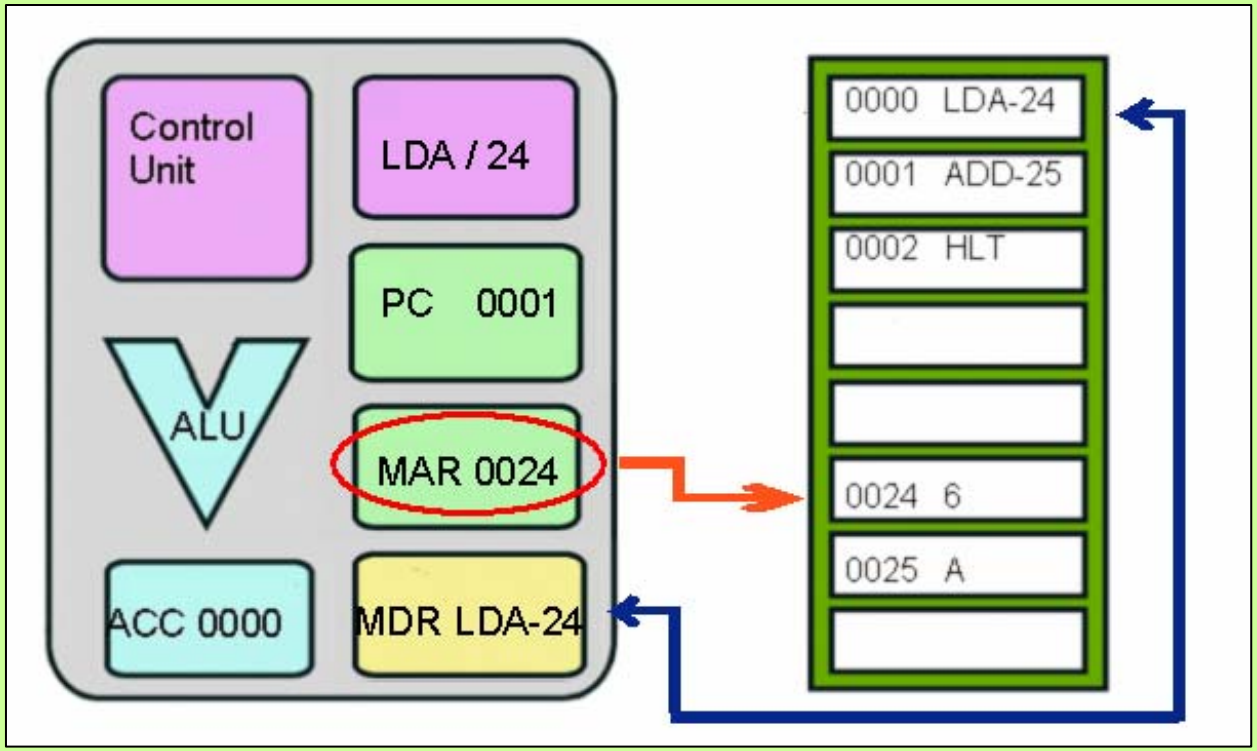
In questo esempio l' opcode *LDA* significa carica l'accumulatore con un dato

L'operando *24* dice qual è l'indirizzo del dato da caricare .

Nel frattempo il PC viene incrementato di 1.

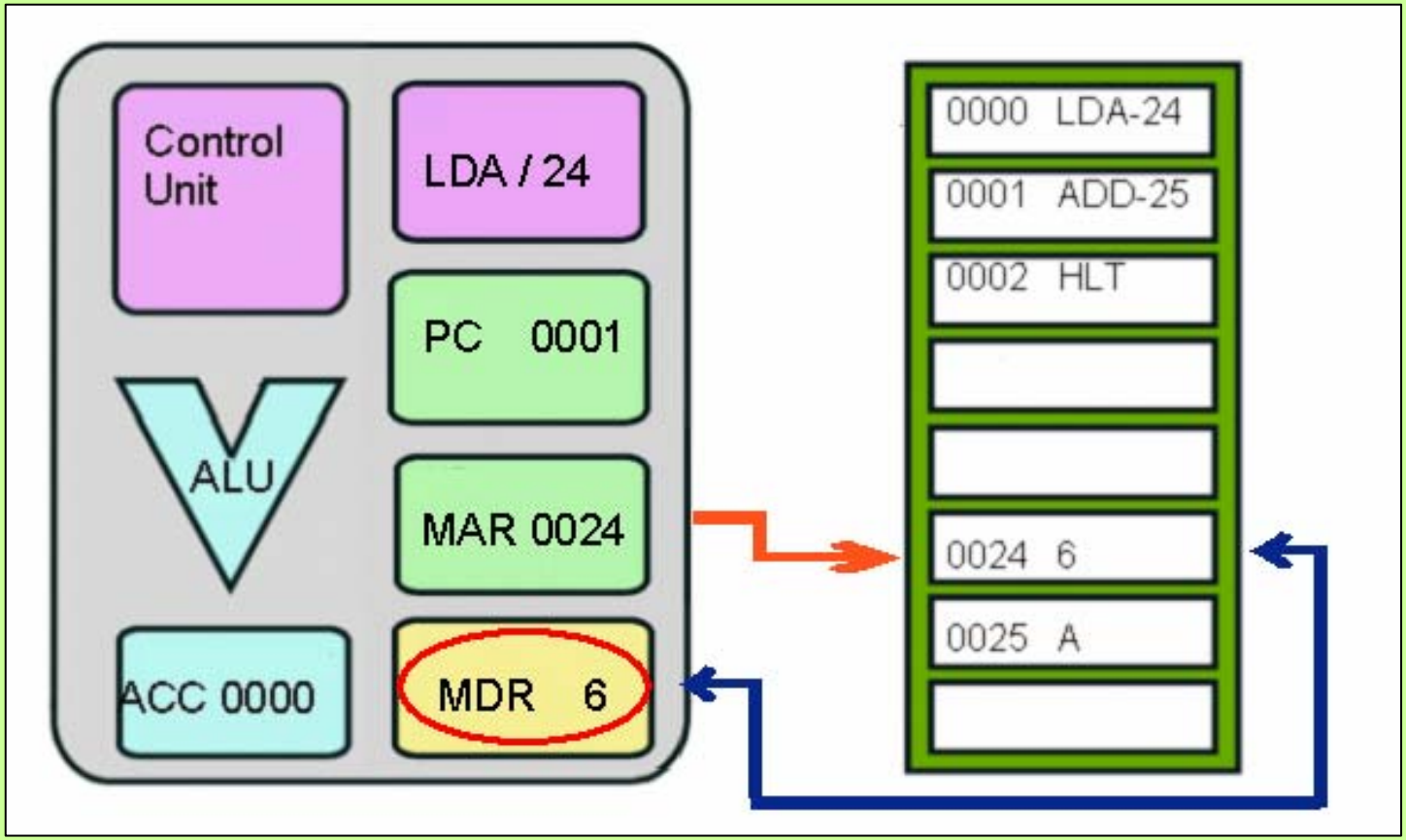
EXECUTE

Dipende dall'istruzione. Nel nostro caso stiamo caricando l'accumulatore con un valore preso dalla RAM



Per leggere il dato bisogna porre nel registro MAR l'indirizzo della cella da leggere in modo da "puntare" alla L'indirizzo 24 e' messo nelo registro MAR indicando alla RAM che siamo interessati a quella cella

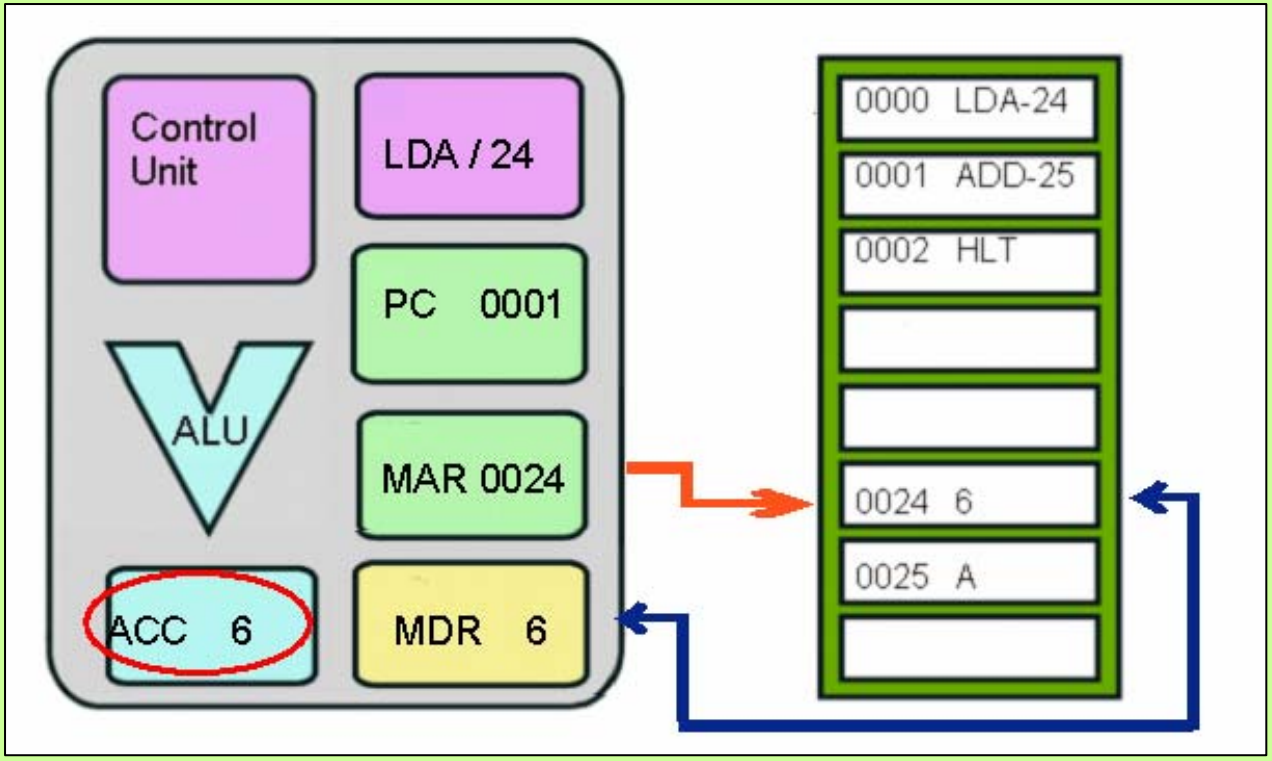
EXECUTE



Vengono inviati sul bus controllo i segnali di lettura della RAM . La RAM invia sul bus dati il contenuto della cella 24 La CPU mette il valore 6 letto dal bus dati in MDR.

EXECUTE

La istruzione imponeva di copiare il valore letto in ACC.
Il valore presente in MDR viene copiato nell'accumulatore.
La istruzione e' completata e la CPU e' pronta ad iniziare la lettura della istruzione successiva.



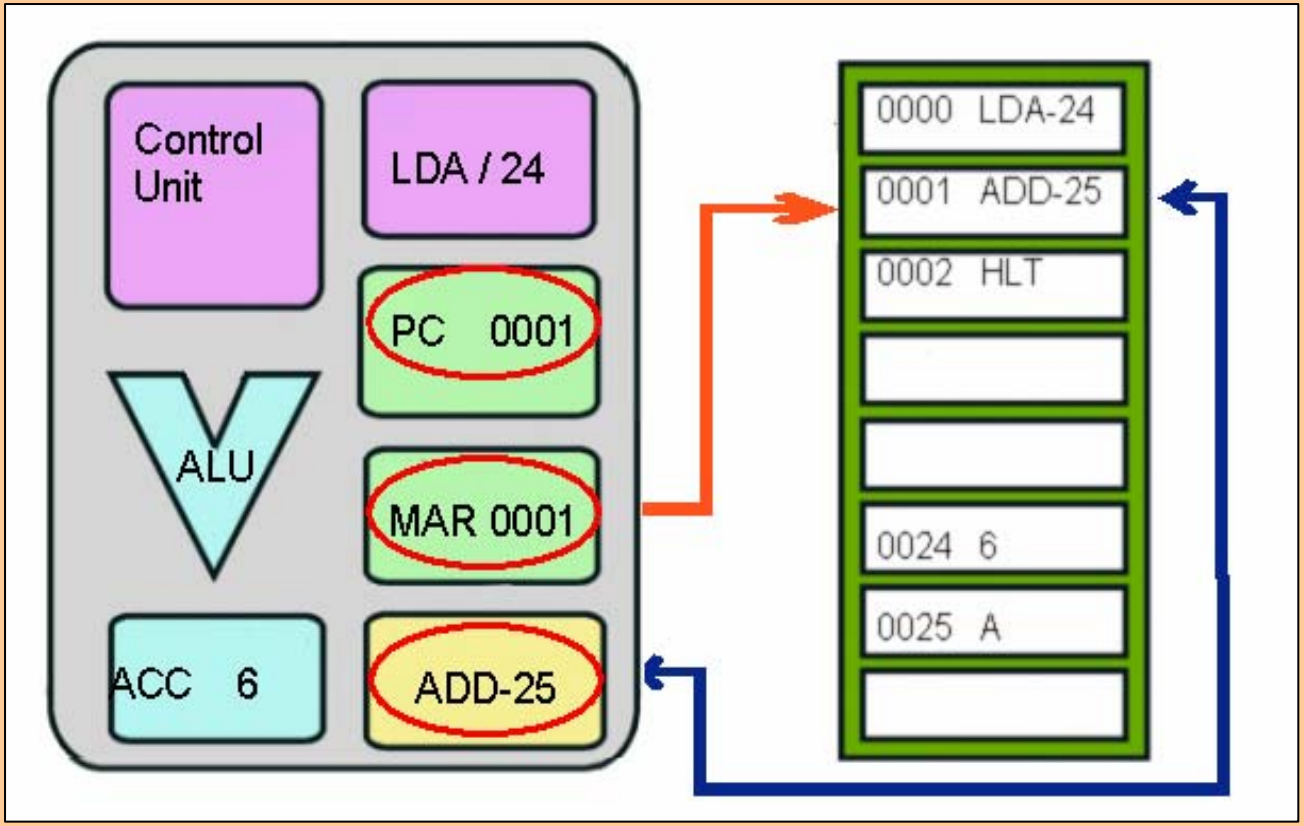
Nota:
I valori dei registri cambiano quando arriva un nuovo dato al loro interno

FETCH

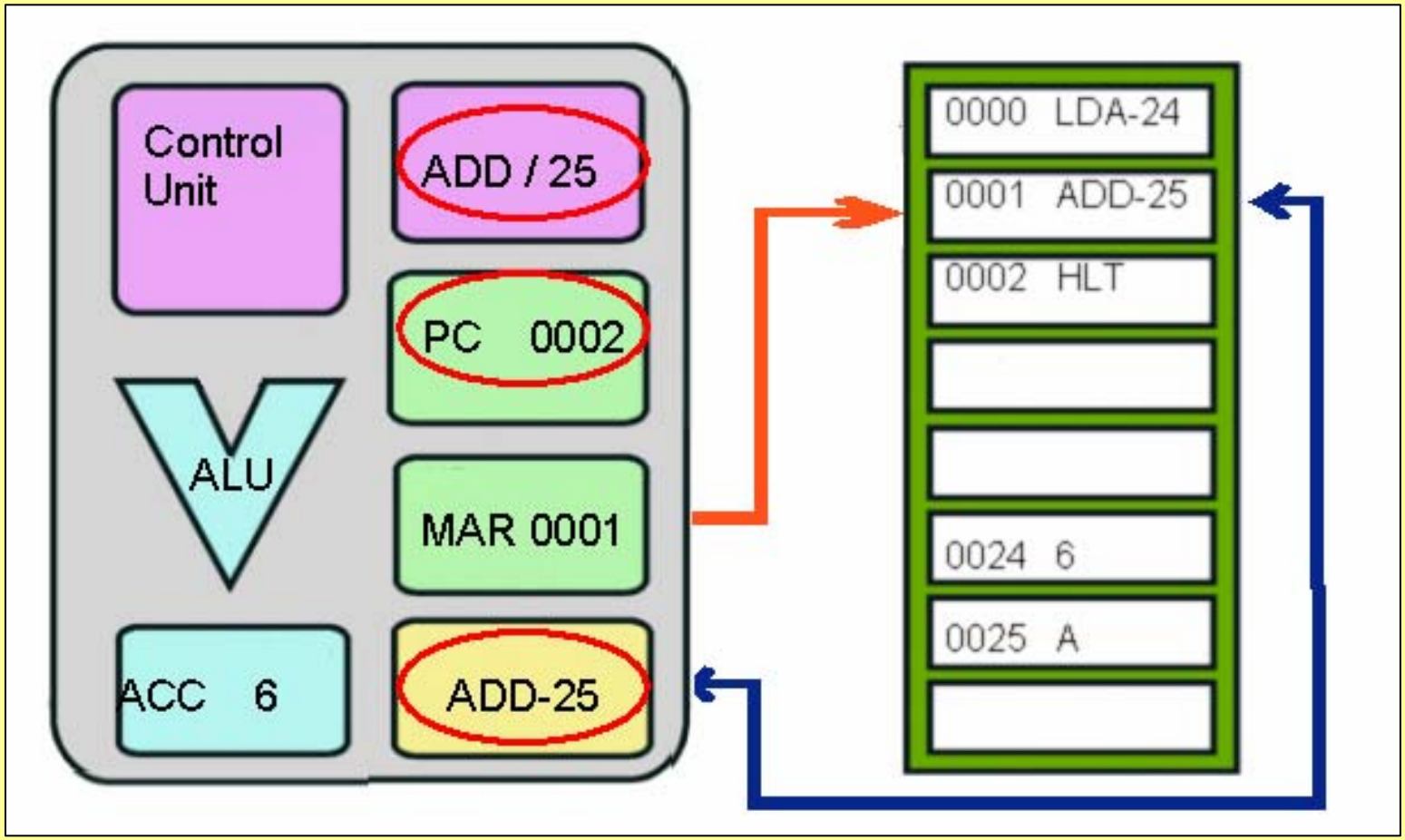
Il PC viene ricopiato nel MAR. Il bus degli indirizzi punta alla cella di memoria di indirizzo 0001

Viene effettuata la lettura della RAM

L'istruzione (ADD-25) viene messa sul Data Bus e ricopiata in MDR.



FETCH + DECODE



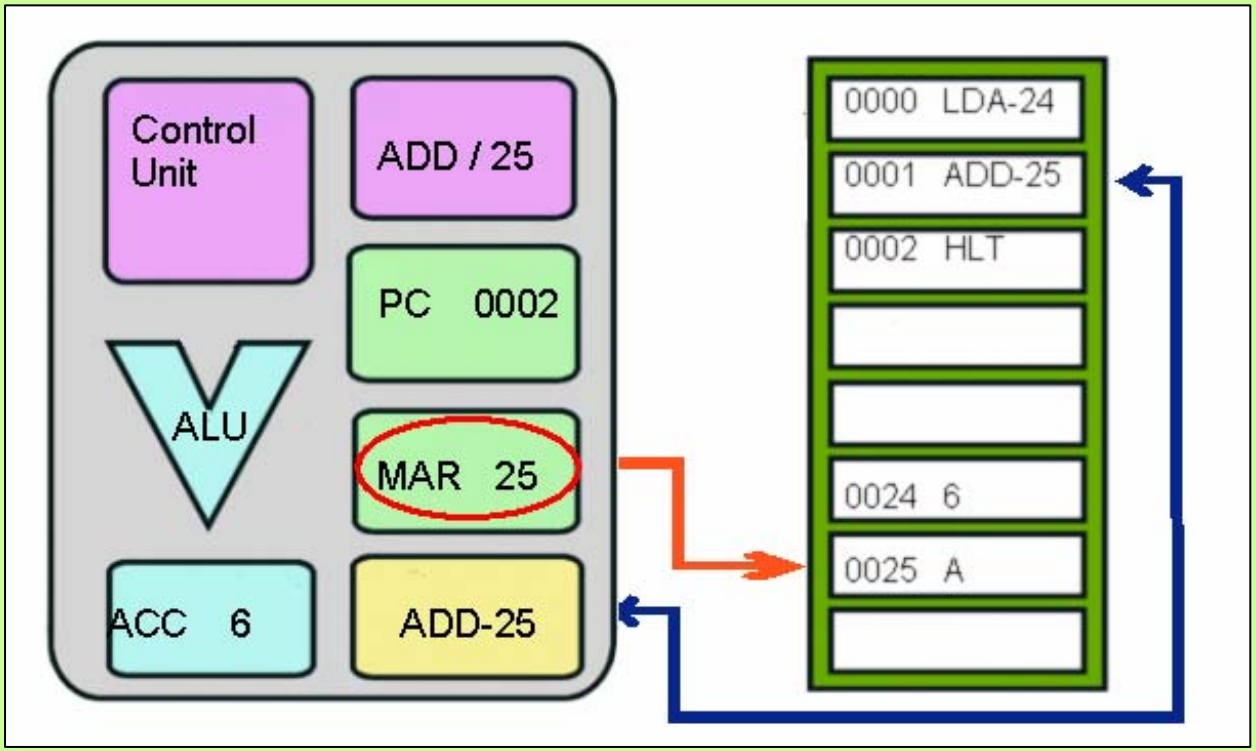
Da MDR l'istruzione ADD-25 e' copiata nell'Instruction Register, dove i circuiti di decodifica la dividono in ADD e 25.

Nel frattempo il Program Counter viene incrementato di 1

EXECUTE

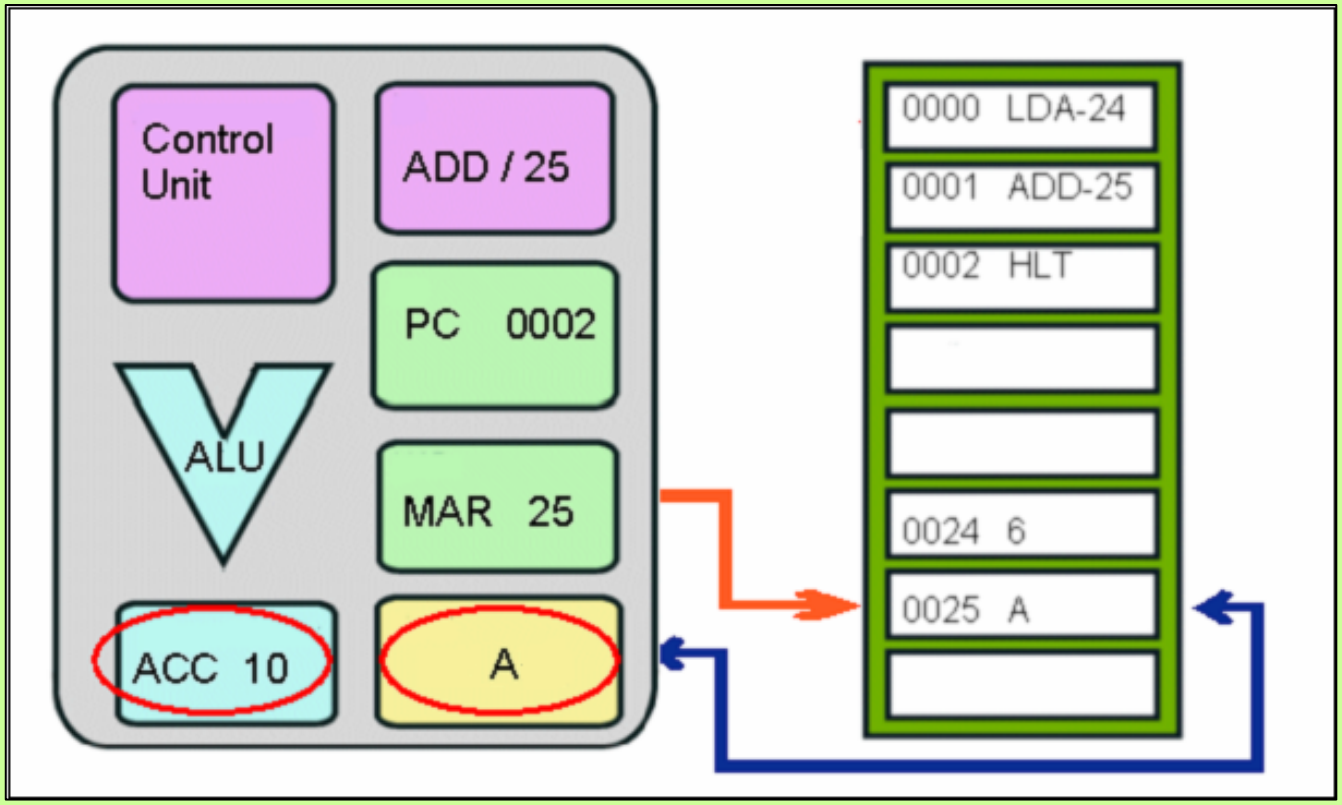
In questo caso bisogna Addizionare il contenuto della RAM all'indirizzo 25, al valore gia' presente nell' Accumulatore.

Poiche' dobbiamo recuperare il contenuto della cella 25 Because dobbiamo puntare il MAR a 25.



EXECUTE

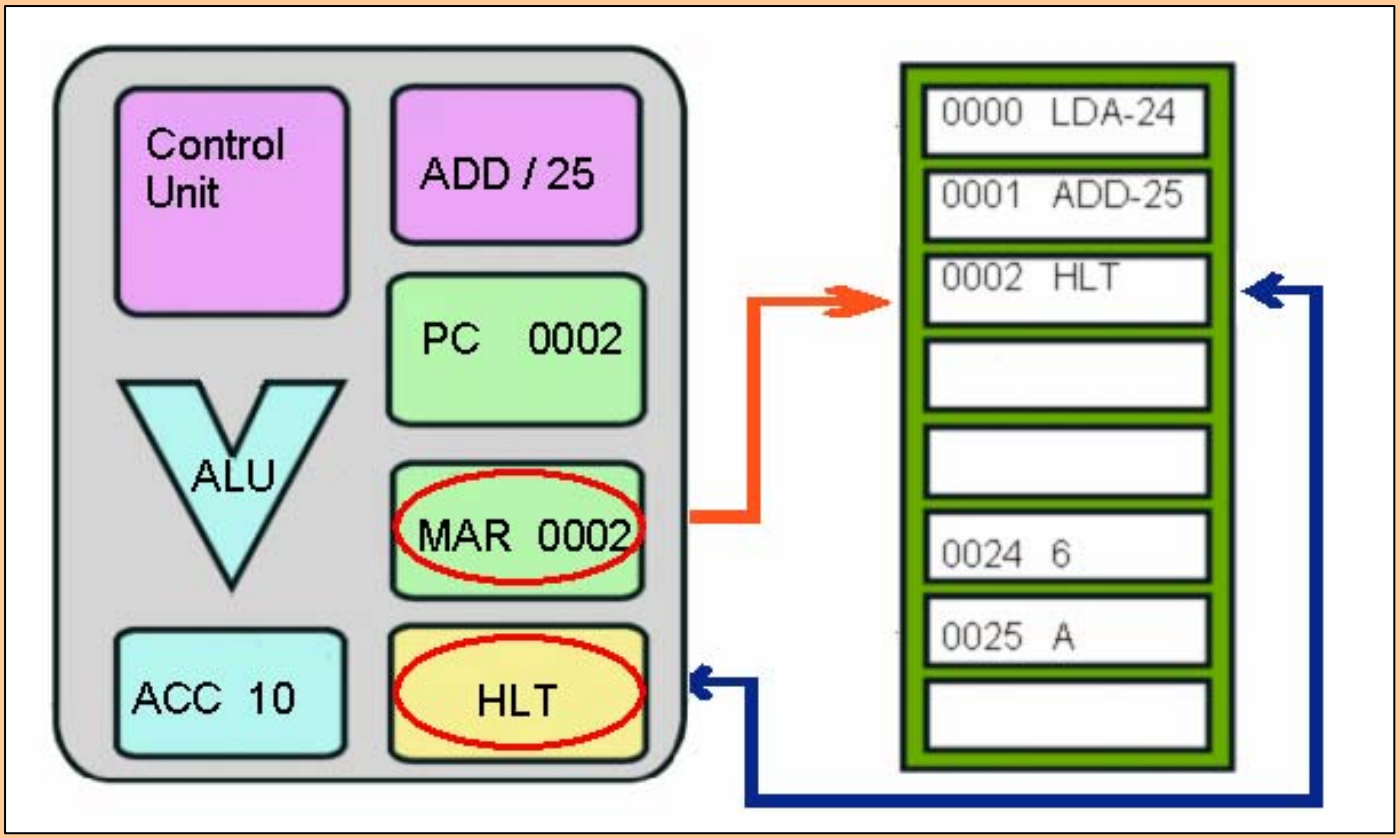
Viene effettuata la lettura della cella prescelta. Il contenuto della cella 25 (A) viene copiato sul Data Bus and quindi si ritrova in MDR.



L'ALU somma il valore in MDR con quello in Accumulatore.
In esadecimale 6 + A = 10 L'ALU mette il risultato in ACC

FETCH

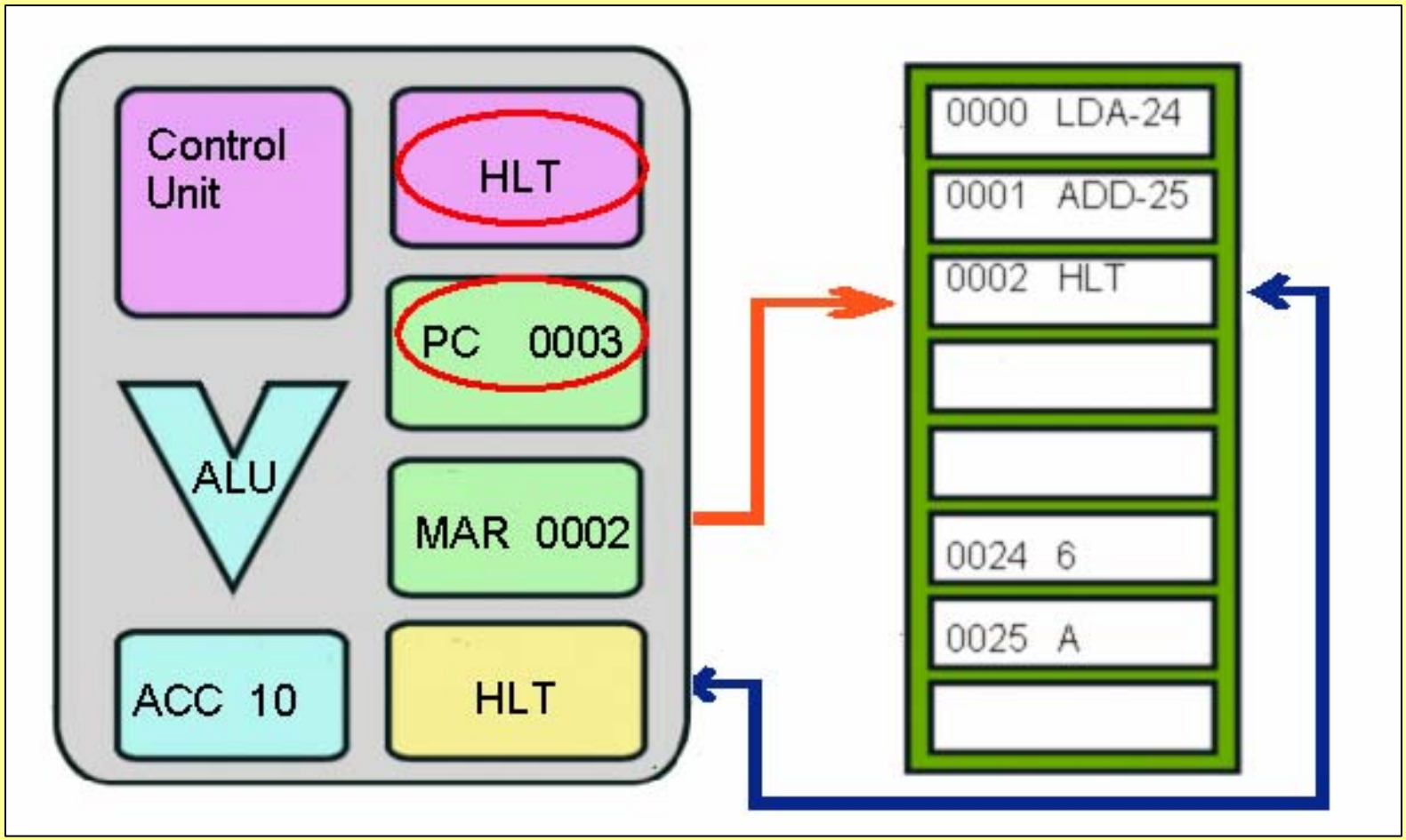
Il PC viene copiato nel MAR, e viene effettuata la lettura della locazione 0002.



Il contenuto di 0002 e' ricopiato sul Data Bus and trasferito in MDR.

FETCH + DECODE

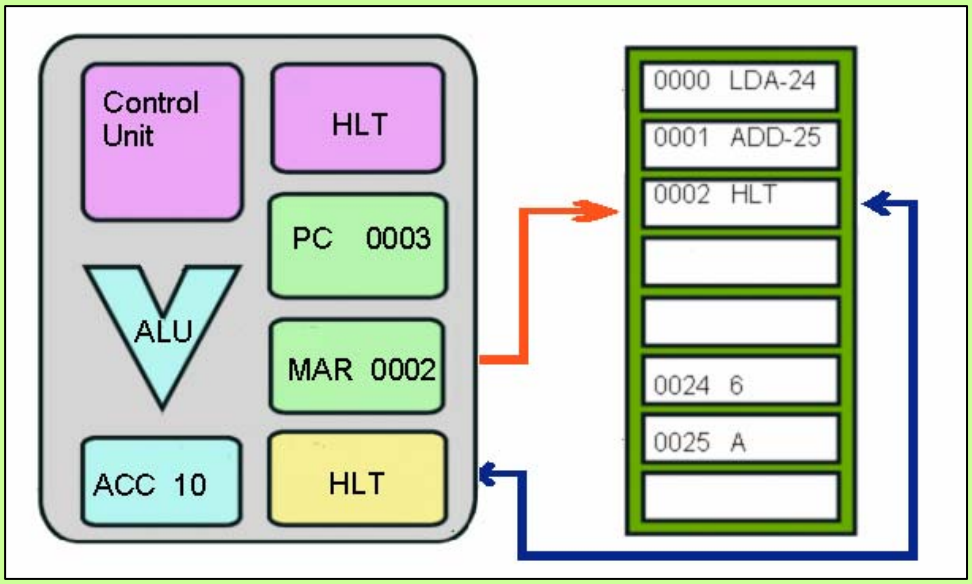
Il contenuto di MDR e' copiuato in IR per essere decodificato. Il PC viene incrementato di 1.



EXECUTE

In questo caso l'istruzione non ha operando, solo Opcode, cosi' il MAR non cambia.

L' instruzione HLT significa "stop the program" cosi' niente altro avverra' d'ora in poi.



Lo spazio che il programma occupa nella RAM sara' sovrascritto dal successivo programma che verra' caricato ed eseguito.

Riepilogo [1]

- Le istruzioni del programma da eseguire devono essere copiate nella RAM
- La CPU fa fetch, decode e execute di ciascuna istruzione una alla volta.
- Le fasi Fetch e Decode sono sempre le stesse ma la fase di Execute variera' in funzione della istruzione

Riepilogo [2]

FETCH

- PC -> MAR. MAR punta al corretto indirizzo di RAM
- Il Contenuto dell'indirizzo puntato arriva in MDR.
- Il Contenuto di MDR e' copiato in IR per la decodifica.

DECODE

- L'Istruzione e' divisa in Opcode e Operando.
- PC e' incrementato di 1.

EXECUTE

- In questa fase MAR puo' cambiare di nuovo, per caricare o scrivere dei valori in memoria