CPU: registro Instruction Pointer (**IP**); registro generale (**AL**, 8 bit); registro indice (**SI**, 16 bit); registri interni (**IR**,**T**); ALU

RAM: capacità di 3072 celle di memoria da 1 byte ciascuna, dall'indirizzo $0400_{hex}$ a $0FFF_{hex}$

ROM: capacità di 768 celle di memoria da 1 byte, da $0100_{hex}$ a $03FF_{hex}$

I/O: 128 indirizzi di input, da $00_{hex}$ a $7F_{hex}$, e 128 indirizzi di output, da $80_{hex}$ a $FF_{hex}$

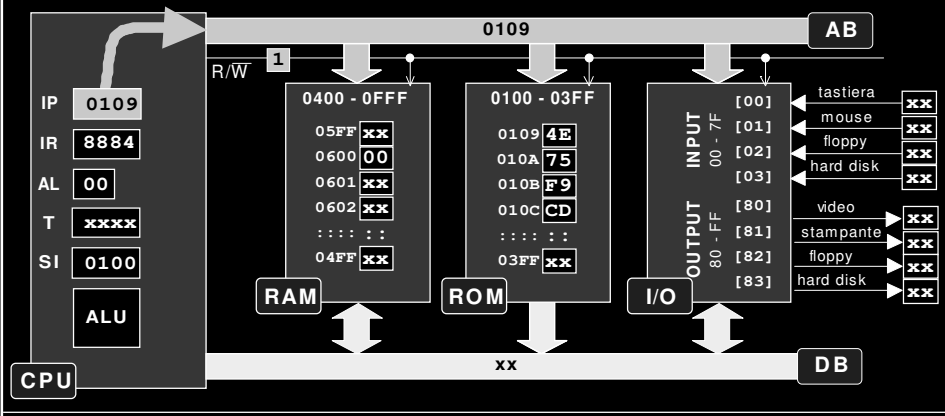
**Esempio**:

```
0100    BE 00 01      MOV SI,0100
0103    B0 00         MOV AL,00
0105    88 84 00 05   MOV [SI+0500],AL
0109    4E            DEC SI
010A    75 F9         JNZ 0105
010C    CD 20         INT 20
```

*Mediante addizione in complemento a 2*

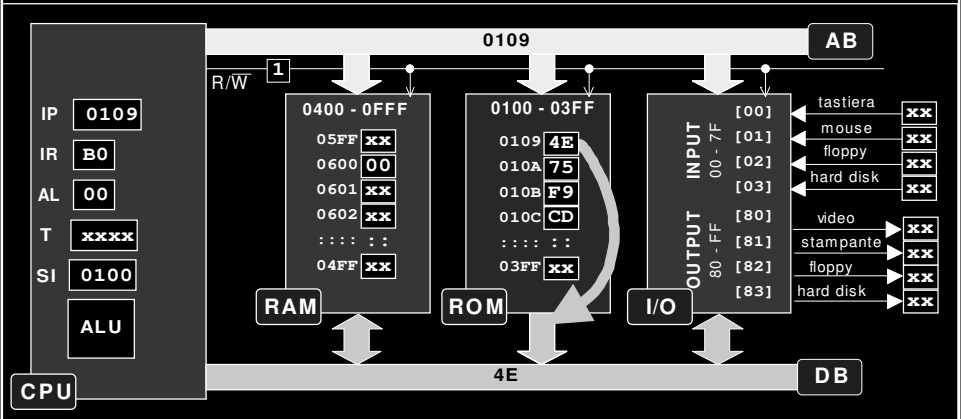


# Architettura semplificata




**Fetch (1)**: la CPU emette in AB il contenuto di IP e pone la linea R/W a 1 per chiedere la lettura della prossima istruzione da eseguire

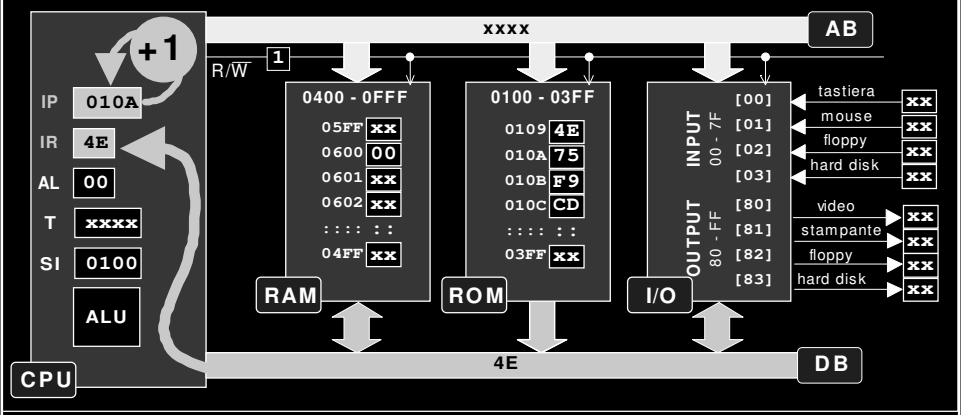
# Architettura semplificata




**Read memory**: la ROM, attivata dall'indirizzo presente in AB, seleziona la sua cella di indirizzo 0109 e, dato il valore della linea R/W (=1), pone il contenuto della cella in DB



# Architettura semplificata




**Fetch (2)**: la CPU legge il valore presente in DB, lo pone in IR e si prepara per la prossima parola di istruzione incrementando IP

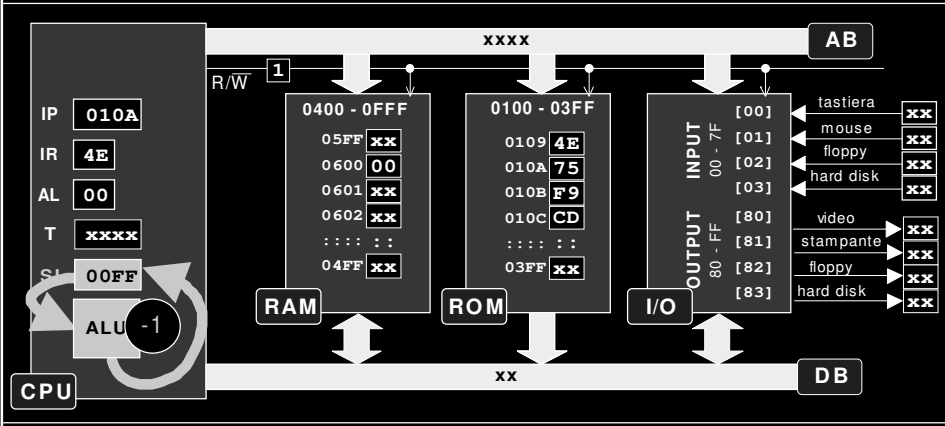
# Architettura semplificata

**Decodifica**: il codice operativo dell'istruzione corrente (**4E**) informa la CPU che essa deve decrementare di 1 il contenuto di SI.

L'istruzione è lunga un solo byte, che ne rappresenta il **codice operativo**.



# Architettura semplificata




**Execute (1)**: la CPU invia all'ALU il contenuto di SI (0100) ed impone l'operazione di decremento di un'unità.

Il risultato (00FF) viene salvato nello stesso registro SI.

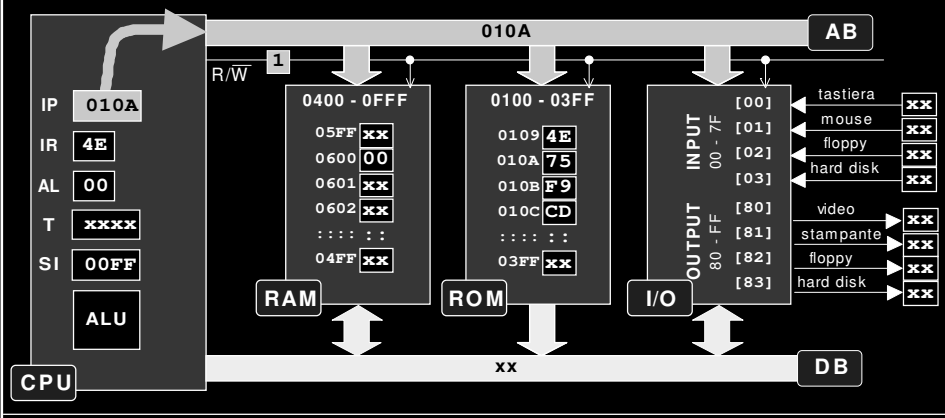
# Architettura semplificata

L'istruzione DEC SI è stata completata mediante :

- **Fetch** (fase 1): emetti IP in AB
  - *Richiesta di lettura in memoria*
- **Fetch** (fase 2): scrivi in IR il valore fornito da DB, incrementa IP, decodifica l'istruzione in IR
- **Execute** (fase 1): decrementa SI
  - *In questo caso non sono necessarie altre sottofasi*

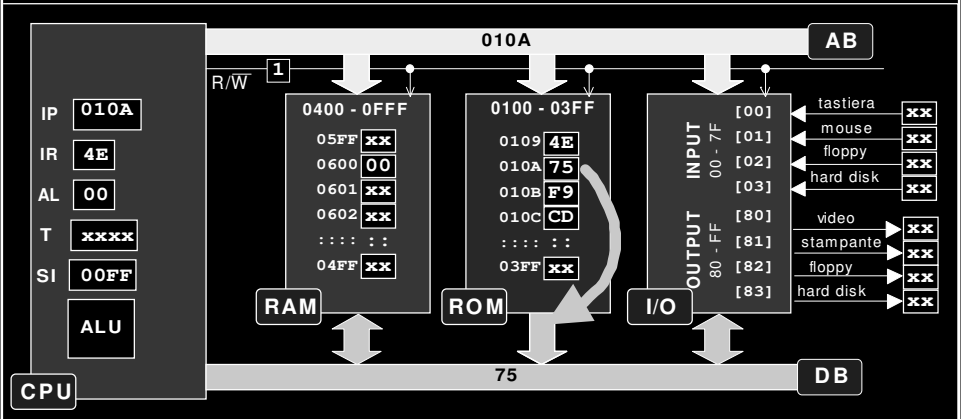


# Architettura semplificata




**Fetch (1)**: la CPU emette in AB il contenuto di IP e pone la linea R/W a 1 per leggere la prossima istruzione da eseguire

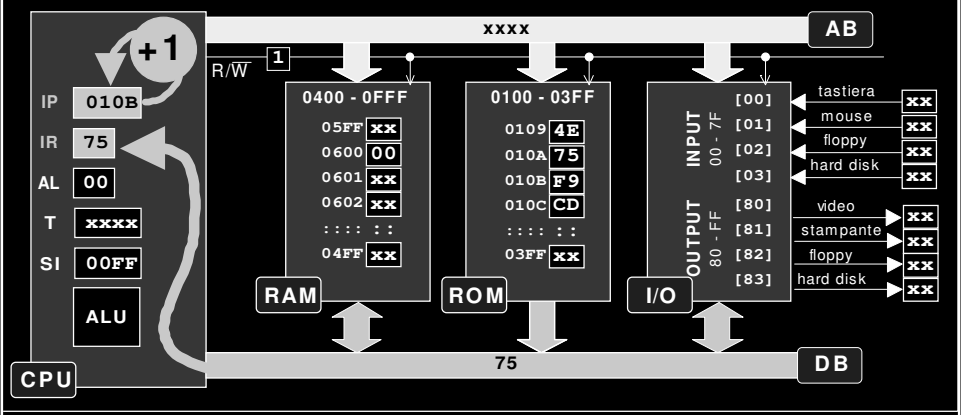
# Architettura semplificata




**Read memory**: la ROM, attivata dall'indirizzo presente in AB, seleziona la sua cella di indirizzo 010A e, dato il valore della linea R/W (=1), pone il contenuto della cella in DB



# Architettura semplificata




**Fetch (2)**: la CPU legge il valore presente in DB, lo pone in IR e si prepara per la prossima parola di istruzione incrementando IP

## Architettura semplificata

**Decodifica**: il codice operativo dell'istruzione corrente (**75**) informa la CPU che l'istruzione è condizionata dal valore del bit di zero (**flag Z**).

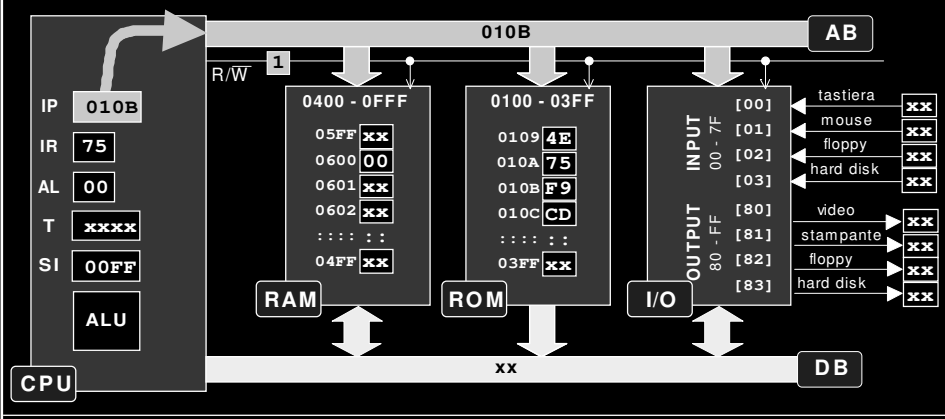
**Z**=1 → prosegui normalmente (*jump if **n**ot zero*)

**Z**=0 → somma l'operando (*displacement*) ad IP

L'operazione si può effettuare su 16 bit in complemento a 2 (oppure sugli 8 bit della parte bassa dei dati trascurando l'eventuale riporto)

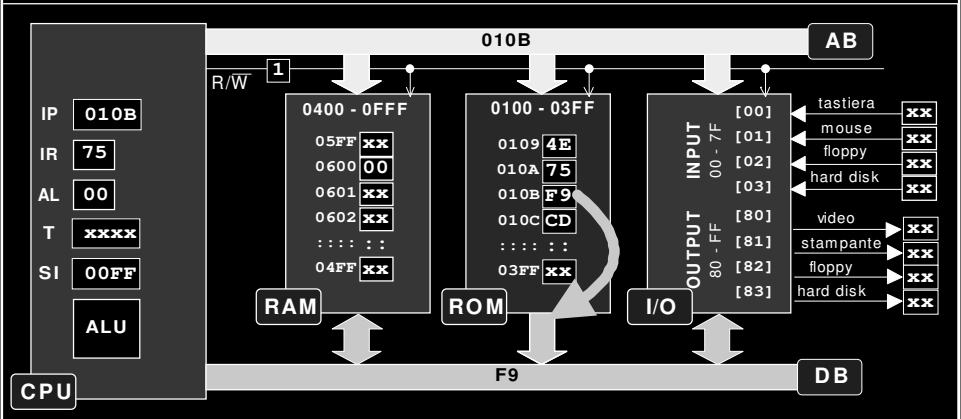


## Architettura semplificata




**Execute (1)**: l' istruzione ha 1 operando, dunque la CPU emette il contenuto di IP in AB e pone la linea R/W a 1, per leggere l'operando ( displacement)

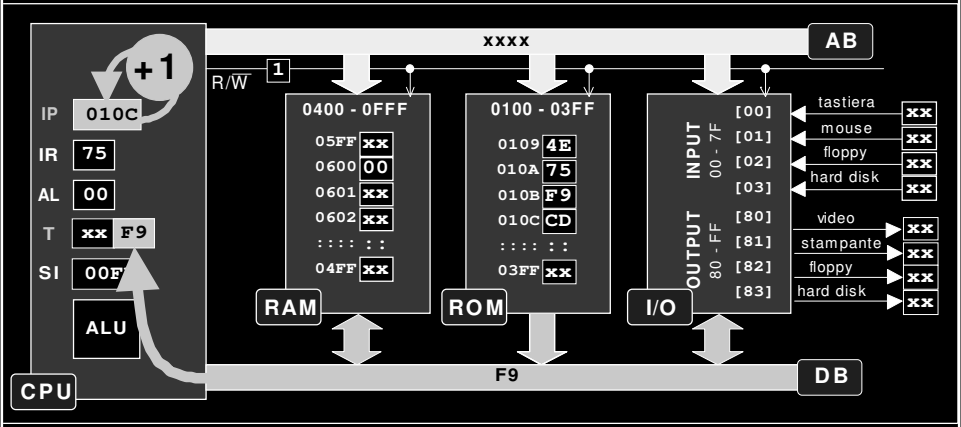
# Architettura semplificata




**Read memory**: la ROM, attivata dall'indirizzo presente in AB, seleziona la sua cella di indirizzo 010B e, dato il valore della linea R/W (=1), pone il contenuto della cella in DB

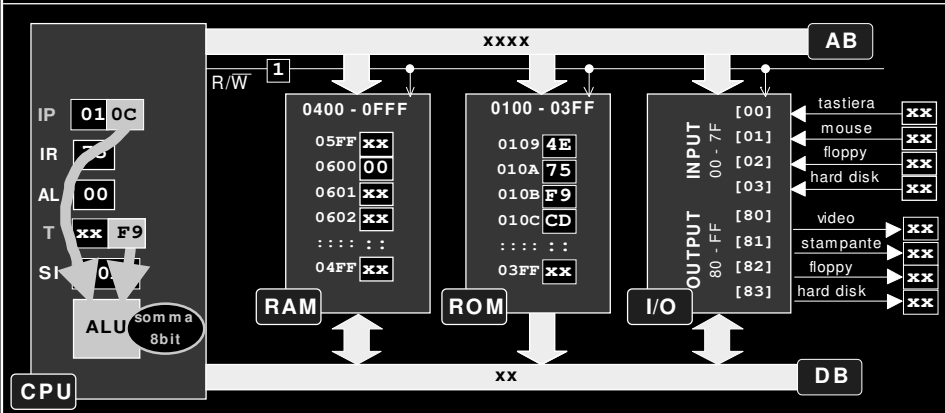


# Architettura semplificata




**Execute (2)**: la CPU legge il valore presente in DB, lo pone nella parte bassa del registro T (*dei valori temporanei*), e si predispone ad acquisire la prossima parola di istruzione incrementando IP

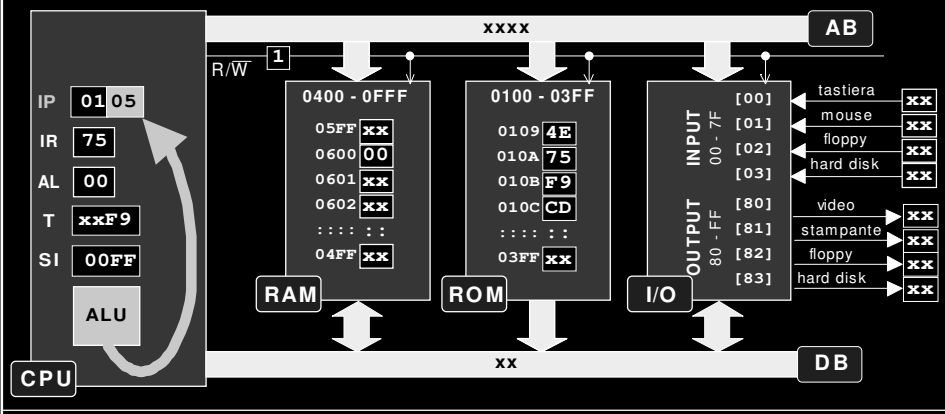
# Architettura semplificata




**Execute (3a)**: per Z=0, la CP richiederebbe all'ALU la somma tra (la parte bassa di) IP e (quella di) T



# Architettura semplificata




**Execute (3b)**: la CPU poi scriverebbe il risultato nel(la parte bassa del) registro IP (trascurando eventuale riporti), determinando cosi' l'indirizzo non consecutivo dell'istruzione successiva (JUMP, salto)