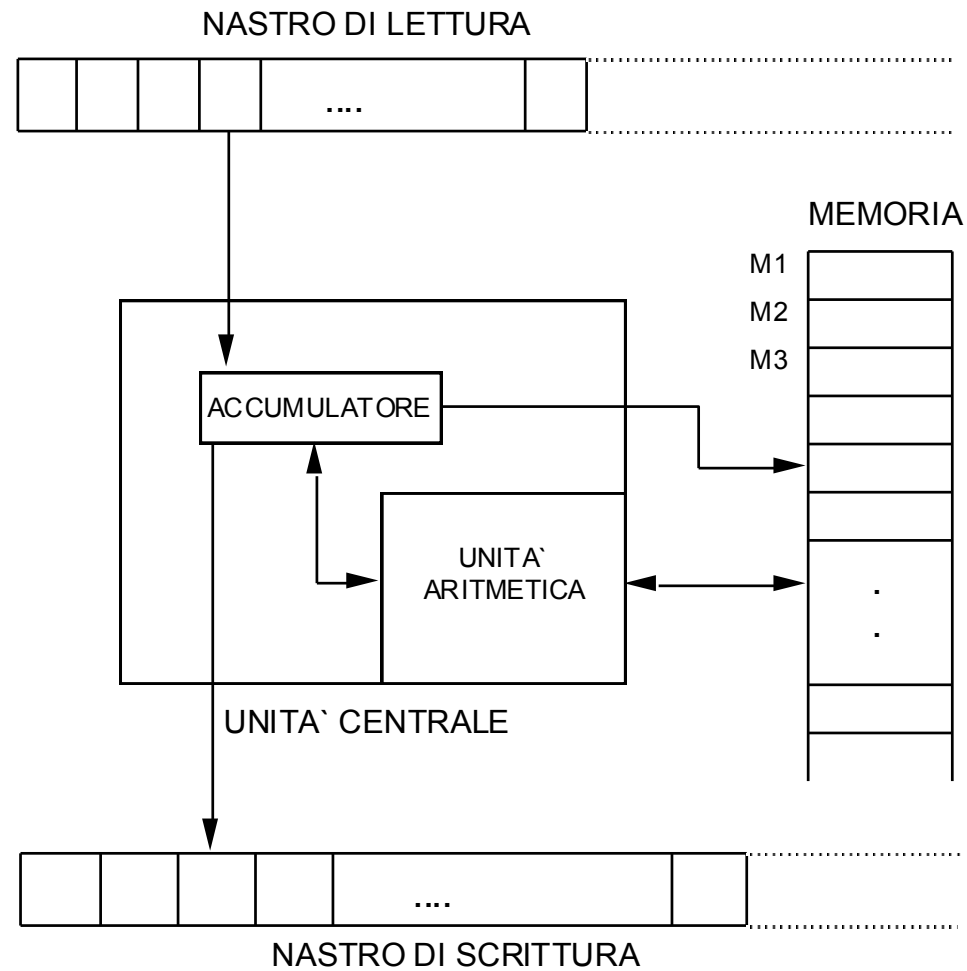


Macchina di von Neumann

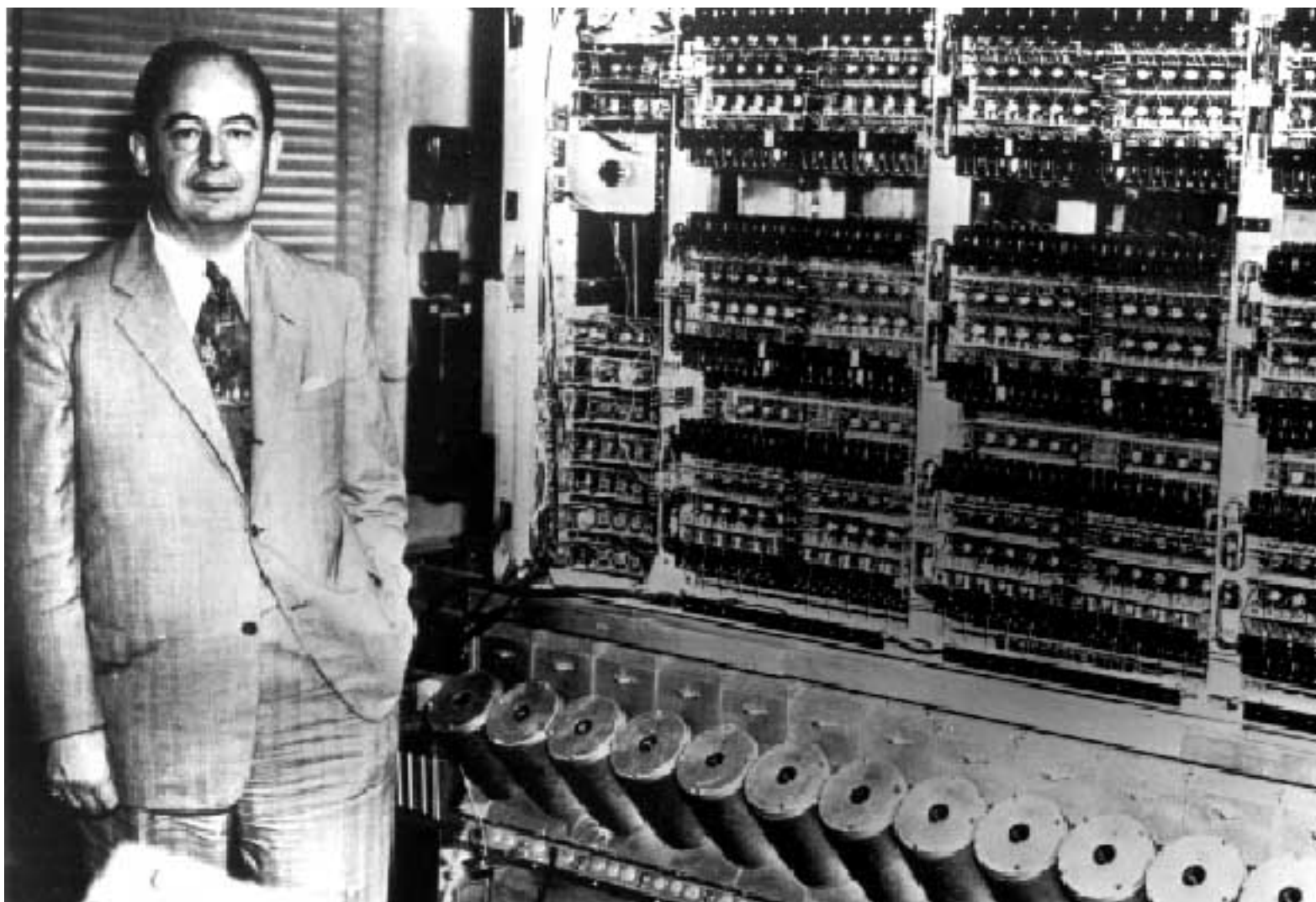
Calcolatore/Sistema Informatico

- Altro non è che un enorme “tritabit”
- Solo che i bit sono organizzati in “pacchetti di informazioni”
 - byte, words, e via di seguito...
- e ... viaggiano all'interno del sistema!
- Cerchiamo di capire a grandissime linee come ciò accade per poter meglio fornire disposizioni al sistema (algoritmi) su come elaborare l'informazione che esso riceve

Macchina di von Neumann



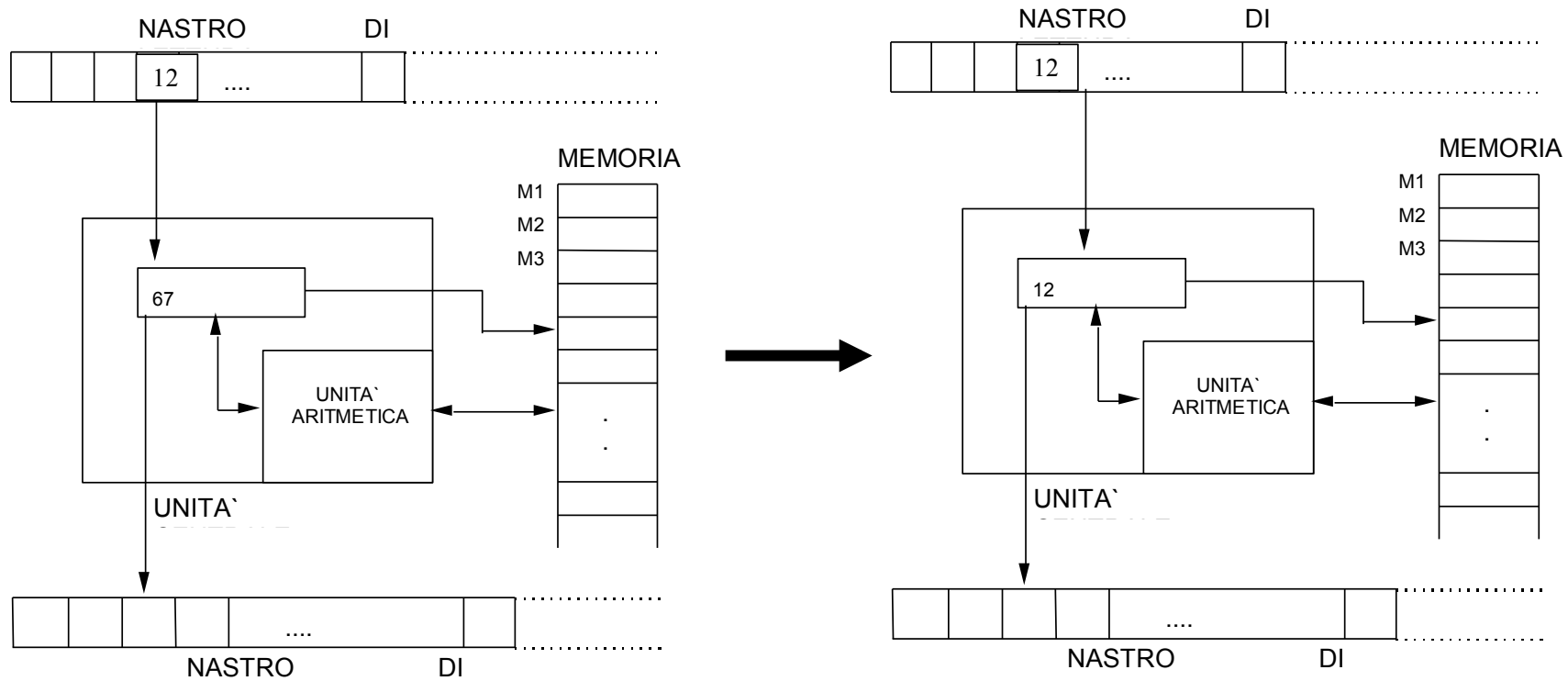
Macchina di von Neumann



Programma

- Per comunicare (tra uomini o macchine) occorre un **linguaggio**
- L' uomo deve usare un linguaggio accessibile alla macchina, ossia rigoroso, preciso: **non** il linguaggio naturale
- Il linguaggio della macchina di von Neumann:
 - un **programma** è una sequenza di istruzioni
 - un' **istruzione** è costituita da un campo **codice operativo** (CO) e da un (eventuale) **campo operando**
 - il CO dice alla macchina che operazione deve eseguire
 - l'operando dice a che cosa va applicata l'operazione

READ



Istruzioni di trasferimento

- READ
- WRITE
- LOAD `xxx`: il contenuto della cella `xxx` viene trasferito nell'accumulatore
- STORE `xxx`: il contenuto dell'accumulatore viene trasferito nella cella `xxx`
- Un primo programma (legge due numeri e li riscrive in ordine inverso):
 - 1) READ
 - 2) STORE 30
 - 3) READ
 - 4) WRITE
 - 5) LOAD 30
 - 6) WRITE

Operazioni aritmetiche

- ADD, SUB, MULT, DIV (divisione intera)
- ADD x : $[ACC] + [x] \rightarrow [ACC]$
- Un secondo programma (legge due numeri e ne stampa la somma):

```
1) READ
2) STORE    30
3) READ
4) ADD      30
5) WRITE
```


Flusso d'esecuzione

- Una caratteristica fondamentale degli algoritmi è la possibilità di decidere il da farsi in funzione dei dati
- Nella macchina di von Neumann questa funzionalità è ottenuta mediante istruzioni di salto:
 - BR x : la macchina salta **direttamente** ad eseguire l'istruzione x -esima, indipendentemente dall'istruzione attuale
 - BEQ x : salto condizionato: la macchina salta direttamente ad eseguire l'istruzione x -esima, **se il contenuto dell'accumulatore è 0**, altrimenti prosegue con l'istruzione successiva
 - BGE, BG, ... sono altre forme di salto condizionato dipendenti dal contenuto dell'accumulatore

Indirizzamento

- **Indirizzamento immediato:**
 - ADD= 13: viene sommato il valore 13 al contenuto dell'accumulatore, non il contenuto della cella 13!
 - LOAD= 1: il valore 1 viene caricato nell'accumulatore, non il contenuto della cella 1!

Sommatoria di n numeri

1) READ

2) STORE 101

3) LOAD= 0

4) STORE 102

5) LOAD 101

6) BEQ 14

Usiamo la cella 101 per tenere traccia di quante termini dobbiamo ancora sommare, che troviamo come primo input sul nastro

La cella 102 contiene la somma parziale, inizialmente a zero.

Se la cella 101 porta il valore di zero, abbiamo terminato la sommatoria!

...continua...

7) READ

8) ADD 102

9) STORE 102

...altrimenti, leggiamo il prossimo valore in ingresso dal nastro, e gli aggiungiamo la somma parziale fin'ora.

10) LOAD 101

11) SUB= 1

12) STORE 101

Avendo considerato un nuovo termine, decrementiamo di uno il contatore dei termini rimanenti nella cella 101.

...e finisce!

13)BR

5

Torniamo alla riga 5 per verificare se esistono altri termini da sommare.

14)LOAD

102

15)WRITE

16)END

Se siamo qui, significa che la cella 101 conteneva zero, quindi la sommatoria è conclusa: scriviamo il risultato in uscita!

Altro esempio

- Riscrittura di una sequenza di numeri in ordine inverso
 - i numeri sono diversi da 0
 - la sequenza è “chiusa” da uno 0
- Serve qualcosa di nuovo: **indirizzamento indiretto**
 - `LOAD@ x`: viene trasferito in accumulatore il contenuto della cella il cui indirizzo è a sua volta contenuto nella cella x

Una possibile soluzione

- Cella 101: contatore numeri letti
- Cella 102: indirizzo celle contenenti i numeri letti
 - indirizzo di partenza: 110

1)	LOAD=	0
2)	STORE	101
3)	LOAD=	110
4)	STORE	102

Inizializziamo il contatore nella cella 101 al valore zero!

Inizializziamo la cella "puntatore" all'indirizzo di partenza (110).

...continua...

5) READ
6) BEQ 15
7) STORE@ 102

Se abbiamo letto zero, abbiamo finito la sequenza in ingresso; altrimenti, memorizziamo il dato nella cella “puntata” dalla cella 102.

8) LOAD 101
9) ADD= 1
10) STORE 101

Incrementiamo il contatore dei dati letti in ingresso.

11) LOAD 102
12) ADD= 1
13) STORE 102

Incrementiamo l'indirizzo a cui punta la cella 102.

14) BR 5

Ripetiamo finchè non troviamo uno zero in ingresso.

...e finisce!

- Una volta terminata la lettura, cioè dopo aver letto il dato 0, iniziamo la fase di scrittura, sostanzialmente procedendo “a ritroso”
 - il contatore nella cella 101 indica il numero di dati ancora da scrivere, e ci serve per sapere quando fermarsi
 - la cella 102 procede a ritroso per rileggere al contrario i dati che arrivano dal nastro di input

15) LOAD 101

16) BEQ 26

Se il contatore in cella 101 è zero, abbiamo finito.

17) LOAD 101

18) SUB= 1

19) STORE 101

Stiamo per scrivere un dato in uscita, decrementiamo il contatore in cella 101.

20) LOAD 102

21) SUB= 1

22) STORE 102

Decrementiamo l'indirizzo in cella 102 per trovare il dato da scrivere in uscita.

23) LOAD@ 102

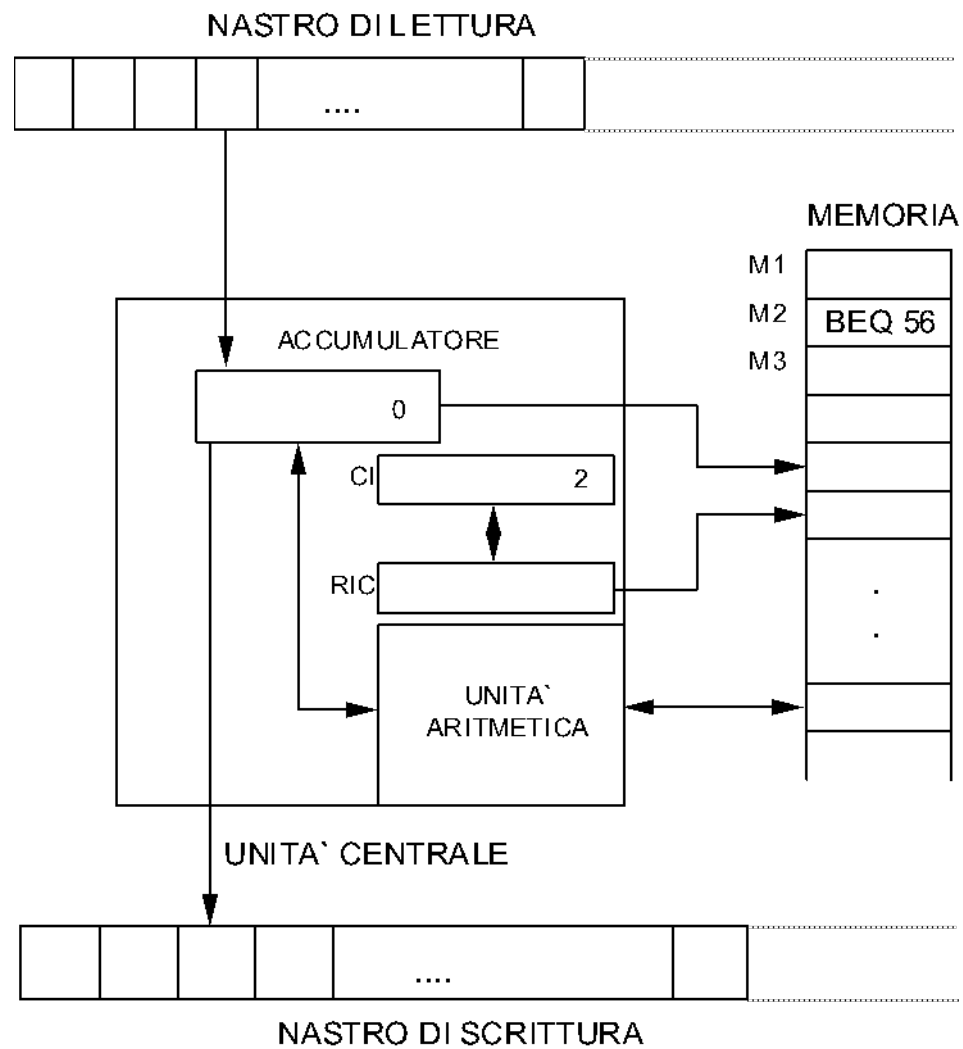
24) WRITE

25) BR 15

26) END

Scriviamo il dato in uscita e ripetiamo finchè il contatore in cella 101 non è zero.

Nulla di magico



In realtà

0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 leggi un valore in ingresso e ponilo nella cella numero 16 (variabile a)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 leggi un valore e ponilo nella cella numero 17 (variabile b)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 leggi un valore e ponilo nella cella numero 18 (variabile c)
0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 leggi un valore e ponilo nella cella numero 19 (variabile d)
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 carica il registro A con il contenuto della cella 16 (valore di a)
0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 carica il registro B con il contenuto della cella 17 (valore di b)
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 somma i contenuti dei registri A e B
0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 copia il contenuto del registro A nella cella numero 20 (risultato parziale)
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 carica il registro A con il contenuto della cella 18 (valore di c)
0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 carica il registro B con il contenuto della cella 19 (valore di d)
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 somma i contenuti dei registri A e B
0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 carica il registro B con il contenuto della cella 20 (risultato parziale)
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 moltiplica i contenuti dei registri A e B
0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 copia il contenuto del registro A nella cella numero 20 (risultato)
0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 scrivi in output il contenuto della cella numero 20 (risultato)
1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 arresta l'esecuzione del programma
..... spazio per la variabile a (cella 16)
..... spazio per la variabile b (cella 17)
..... spazio per la variabile c (cella 18)
..... spazio per la variabile d (cella 19)
..... spazio per il risultato (cella 20)

Esercizio

- Scrivere un programma per la macchina di von Neumann che dati 4 interi strettamente positivi in ingresso ne stampi la media sul nastro di uscita

Una possibile soluzione

- Domandiamoci:
 - cosa devo memorizzare?
 - di quante celle di memoria ho bisogno?
 - quali sono i macro-passi dell'algoritmo?

Ragioniamo

- Cosa devo memorizzare?
 - i quattro interi di cui devo calcolare la media?
 - la loro somma parziale?
 - ...altro?
- Di quante celle di memoria ho bisogno?
 - quattro? una?
- Quali sono i macro-passi dell'algoritmo?
 - calcolo la somma
 - divido per 4
 - stampo in uscita (non scordatevelo!)

Un possibile codice

- La cella 100 memorizza la somma parziale
- La cella 101 conta il numero di dati letti in ingresso

1) LOAD= 0
2) STORE 100
3) LOAD= 0
4) STORE 101

Inizializzo la somma parziale e il contatore di dati letti in ingresso.

5) READ
6) ADD 100
7) STORE 100

Leggo il primo dato e aggiorno la somma parziale.

8) LOAD 101
9) ADD= 1
10) STORE 101

Aggiorno il contatore del numero di dati letti in ingresso.

...continua...

11)READ

12)ADD 100

13)STORE 100

Leggo il secondo
dato e aggiorno la
somma parziale.

14)LOAD 101

15)ADD= 1

16)STORE 101

Aggiorno il contatore del
numero di dati letti in ingresso.

...continua...

17)READ

18)ADD 100

19)STORE 100

Leggo il terzo dato
e aggiorno la
somma parziale.

20)LOAD 101

21)ADD= 1

22)STORE 101

Aggiorno il contatore del
numero di dati letti in ingresso.

...continua...

23)READ

24)ADD 100

25)STORE 100

Leggo il quarto
dato e aggiorno la
somma parziale.

26)LOAD 101

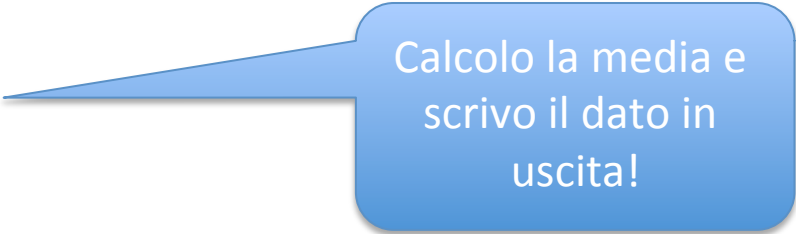
27)ADD= 1

28)STORE 101

Aggiorno il contatore del
numero di dati letti in ingresso.

...e finisce!

29)LOAD	100
30)DIV	101
31)WRITE	
32)END	



Calcolo la media e
scrivo il dato in
uscita!

- Questa soluzione è un po' stupida...
 - Come migliorarla?
 - Come renderla più **generica**, ad esempio gestendo un numero arbitrario di dati di input, terminati da uno zero?

Perché tutto questo?

- Per **guidare** un'auto non vi serve capire come funziona il motore
 - ...anche se alle volte sarebbe comunque utile
- Non vale lo stesso se vi viene chiesto di **progettare** un'auto!
- La macchina di von Neumann è (uno dei) motori più semplici
 - ...partire dalle cose semplici per arrivare a quelle sofisticare!

