# Linguaggio Assembler

**Andrea Payaro**

***Consulente Certificato da ELA***
***(European Logistics Association)***

andrea@payaro.it

## *Andrea Payaro*

- Ph.D. in Business Management at University of Padova
- Committee member of AILOG
- Technical Committee Member of RELOADER
- Consultant and teacher of Supply Chain Management at University of Padova
- Certified by ELA (European Logistics Association) – Strategic Level

## Le istruzioni

- 1) trasferimento tra RAM e registri di calcolo della CPU
- 2) operazioni aritmetiche: somma, differenza, moltiplicazione e divisione
- 3) operazioni di controllo: confronto, salto e stop



## Trasferimento

- Da RAM a Registro
- LOAD R0, TEMP;

Mettere il Punto e Virgola alla fine dell'istruzione

- Carica nel registro R0 il contenuto della locazione di memoria RAM indicizzata con l'etichetta TEMP.
- Tale istruzione ha bisogno di una dichiarazione:
- TEMP INT 88;

## La giusta sequenza

- TEMP INT 88;
- LOAD R0,TEMP;
- STOP;

Fine del programma e
Dell'esecuzione delle istruzioni



## Stoccaggio / Memorizzazione

- Da Registro a RAM
- STORE R1, TEMP;
- Carica nella locazione di memoria RAM indicizzata con l'etichetta TEMP il contenuto del registro R1
- Tale istruzione ha bisogno di una dichiarazione:
- TEMP INT 88;

- TEMP INT 88;
- LOAD R0,TEMP;
- STORE R1, TEMP;
- STOP;

Nella CPU considerata ci sono 7 registri



## *Le operazioni aritmetiche*

- SOMMA
- ADD Ri,Rj;
- Somma al contenuto di Ri il contenuto di Rj e metti il risultato in Ri.

- SOTTRAZIONE
- SUB Ri,Rj;
- Sottrai al contenuto di Ri il contenuto di Rj e metti il risultato in Ri.



- MOLTIPLICAZIONE
- MULT Ri,Rj;
- Moltiplica Ri per Rj e il risultato lo si mette in Ri
- DIVISIONE
- DIV Ri,Rj;
- Dividi Ri per Rj e il risultato lo si mette in Ri

- RESTO
- MOD Ri,Rj;
- Calcola il resto della divisione tra Ri e Rj e il risultato viene messo in Ri



## *Confronto*

- COMP Ri, Rj;
- se Ri < Rj memorizza -1 nel registro RC
- se Ri = Rj memorizza 0 in RC
- se Ri > Rj memorizza 1 in RC
- RC è il registro del confronto

## Salto

- Permette di "saltare" ad un’altra istruzione del programma a seconda del contenuto del registro RC (cioè a seconda del risultato di un confronto)

- BRLT (RC=-1);  /*less than
- BREQ (RC=0);  /*Equal
- BRGT (RC=1);  /*great than
- BRANCH;  /* salto incondizionato



## Attenzione

- Prima di un’istruzione di salto e’ necessario un confronto tranne nel caso di un salto incondizionato.

## Esempio

```
ZERO    INT    0;
PARI    INT;
DISPARI INT;
NUM1    INT;
NUM2    INT    2;
        LOAD       R0, NUM1;
        LOAD       R1, NUM2;
        MOD        R0, R1;
        COMP       R0, ZERO;
        BREQ       R_PARI;
        STORE      1, DISPARI;
```



```
        STORE      0, PARI;
        BRANCH     FINE;
R_PARI  STORE      0, DISPARI;
        STORE      1, PARI;
FINE    STOP;
```

- Il programma scrive 1 sulla locazione PARI (0 in DISPARI) se NUM1 è pari e 1 in DISPARI (0 in PARI) se NUM1 è dispari.

## Andrea Payaro

# Thanks for Your Attention

andrea@payaro.it
Via Monte Bianco 16
35020, Ponte San Nicolo' PD