# Esercitazione 05

## Esercizi in Assembly

Gianluca Brilli
*gianluca.brilli@unimore.it*

# Esercizio 1 – Loop e branch

- Realizzare un programma che vada ad implementare un **ciclo for**, all'interno del quale andremo a stampare i caratteri ASCII tra 0 e 9.
- Ragioniamo su come realizzare la struttura del ciclo tramite le operazioni di branch.

# Esercizio 2 – Chiamate a Funzione

- Realizziamo un programma in grado di calcolare il **fattoriale** in maniera **ricorsiva**:
  - ***n! ← n x (n – 1)!***
  - ***n! ← 1 if n = 0 or n = 1***
- Ripassiamo come effettuare chiamate a funzione in assembly:
  - Passare i parametri nei registri da x10 a x17
  - Trasferire controllo alla funzione e svolgere calcoli
  - Scrivere i risultati nei registri per il chiamante
  - Trasferire controllo al chiamante

# Esercizio 3 – Operazioni su Array

- Realizzare un programma che vada a calcolare il **prodotto scalare** tra due array a e b, memorizzati ad esempio nel **segmento dati**.
- Si ricorda che il prodotto scalare può essere calcolato per mezzo della seguente formula:

$$\begin{bmatrix} A_x & A_y & A_z \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = A_x B_x + A_y B_y + A_z B_z = \vec{A} \cdot \vec{B}$$

# Esercizio 4 – Maschere di bit

- Utilizziamo le maschere di bit per realizzare un meccanismo di controllo errori, basato sul **calcolo della parità**.
- In pratica quando viene trasmesso un dato, viene aggiunto un bit che è settato a 1 se il numero di uni nel dato da trasmettere è dispari ( **parità pari** ).
- Realizzare un programma che vada a settare il bit di parità per una dato a 7 bit.

# Esercizio 4 – Maschere di bit

- Supponiamo il seguente funzionamento:

| 7 bit di dati | Byte con bit di parità | |
|---|---|---|
| | Bit di parità pari | Bit di parità dispari |
| 1101001 | **0**1101001 | **1**1101001 |
| 1111111 | **1**1111111 | **0**1111111 |

- Il bit più significativo del nostro byte rappresenta il bit di parità (come in figura).