

Il linguaggio del calcolatore: linguaggio macchina e linguaggio assembly

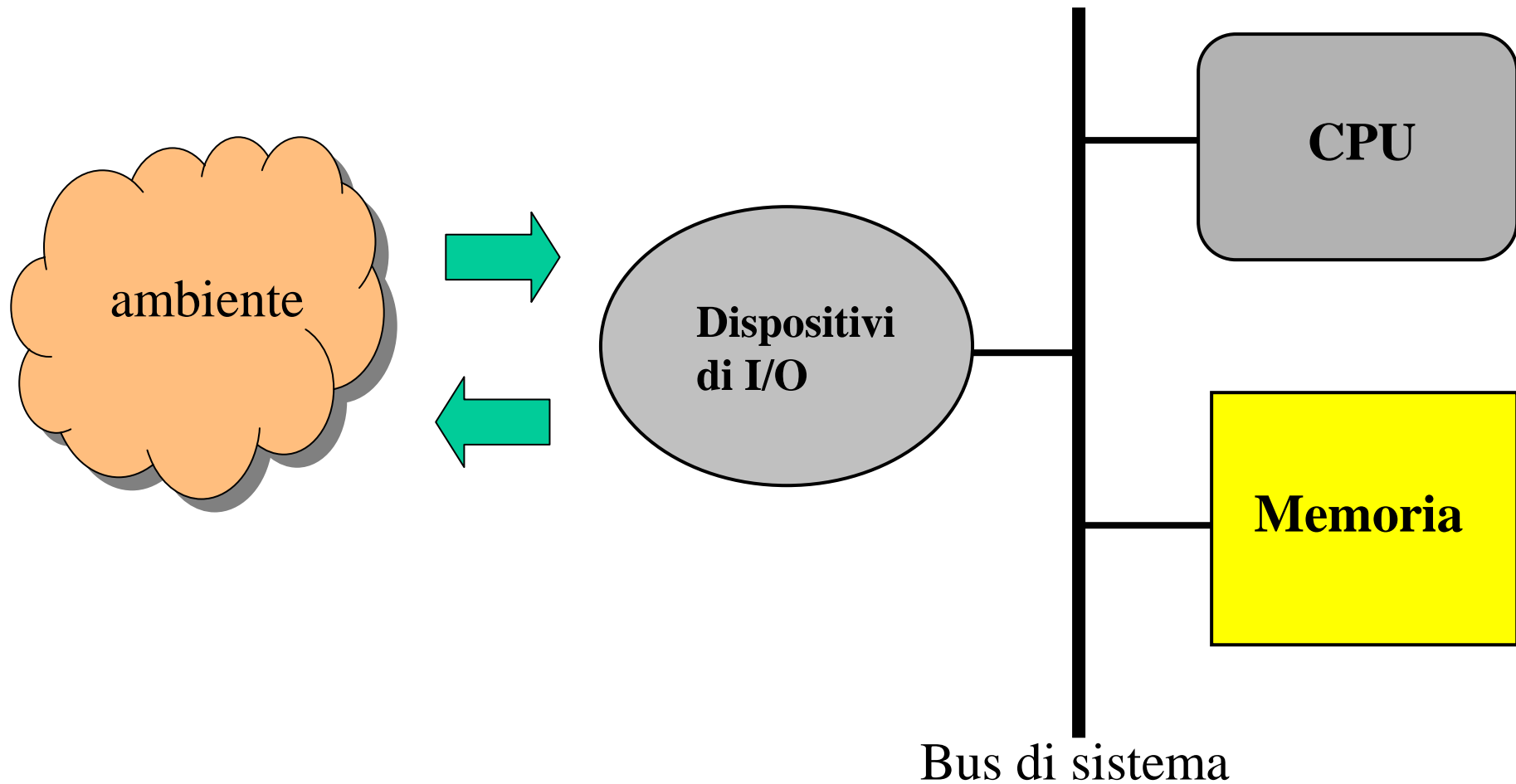
Percorso di Preparazione agli Studi di Ingegneria

Università degli Studi di Brescia

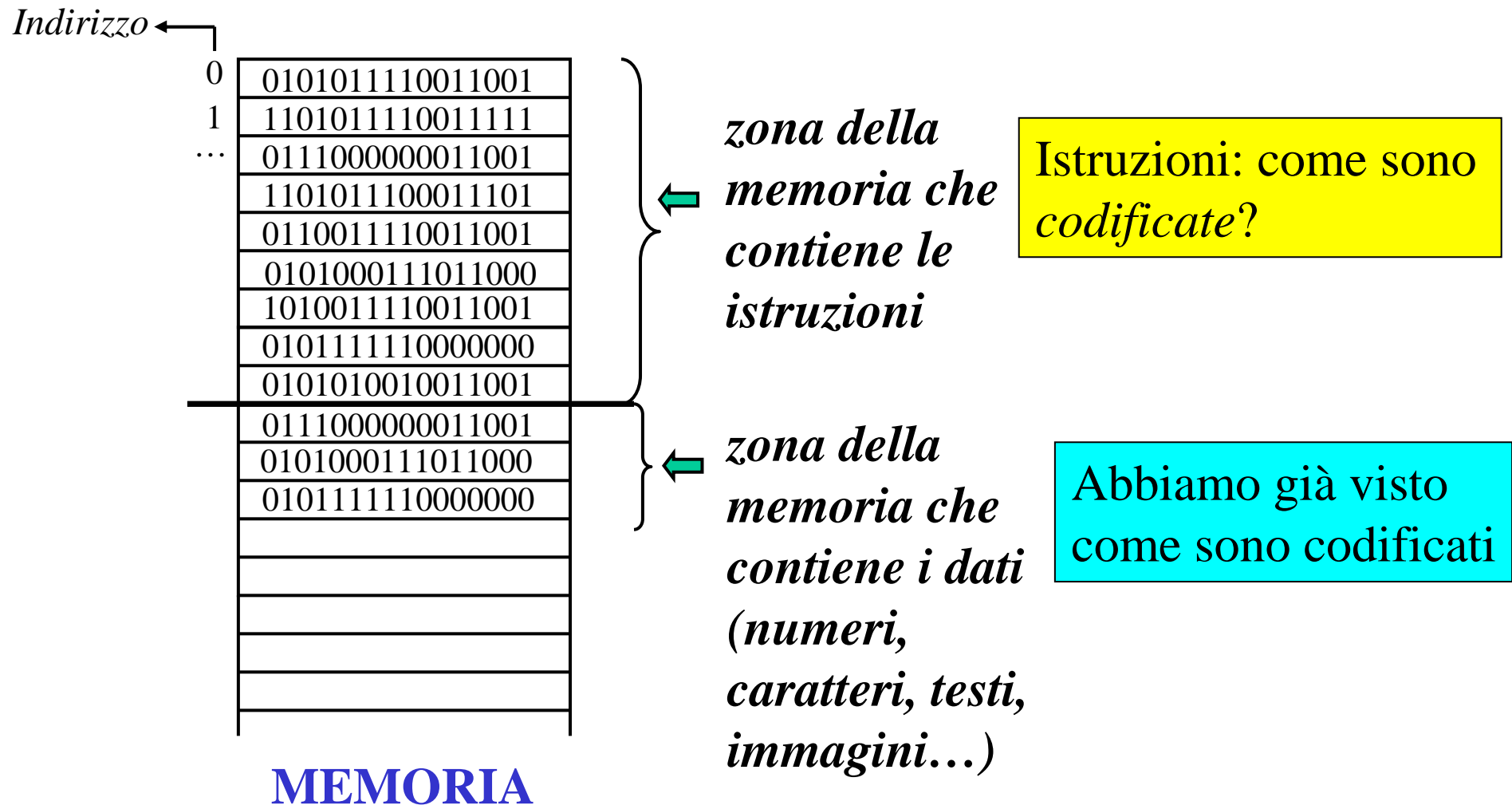
Docente: Massimiliano Giacomini

Richiamo sull'architettura del calcolatore

CPU = *Central Processing Unit (Unità centrale)*
detta oggi Microprocessore o processore



Programma e dati in memoria: rivisitazione



Il linguaggio macchina (richiami)

- Linguaggio macchina: costituito da **istruzioni macchina**, eseguite dalla CPU
- Ogni CPU ha un proprio linguaggio macchina (**ISA – Instruction Set Architecture**): per esempio, le istruzioni dei processori Intel X86 sono diverse da quelle del processore MIPS
 - esistono CPU di marca diversa con diversa struttura fisica che risultano **compatibili** (es. Intel e AMD)
- Le istruzioni del linguaggio macchina sono costituite da stringhe di bit, suddivise in:
 - **Codice operativo** → tipo istruzione
 - **Operandi** → indicano i dati su cui l'istruzione opera (sorgenti) e dove memorizzare il risultato (destinazione)



Codice operativo

- Il processore è in grado di eseguire molti tipi di istruzioni:
add, sub, lw, sw, j, beq, ...
- Ad ogni tipo di istruzione viene assegnato un codice binario



0000 add

0001 sub

1000 lw

1001 sw

- Quanti bit sono necessari per il codice operativo?
Dipende dal numero delle istruzioni disponibili!

Operandi

- Ogni istruzione “opera” su un certo numero di operandi
- Ciascun operando può essere *operando sorgente* o *destinazione*

ESEMPI

add \$r1, \$r3, \$r4

↑ ↑ ↑
 sorgente sorgente
 ↑
 ↑
destinazione

addi \$r1, \$r3, 100

↑ ↑ ↑
 sorgente sorgente
 ↑
 ↑
destinazione

lw \$r1, \$r2, 100

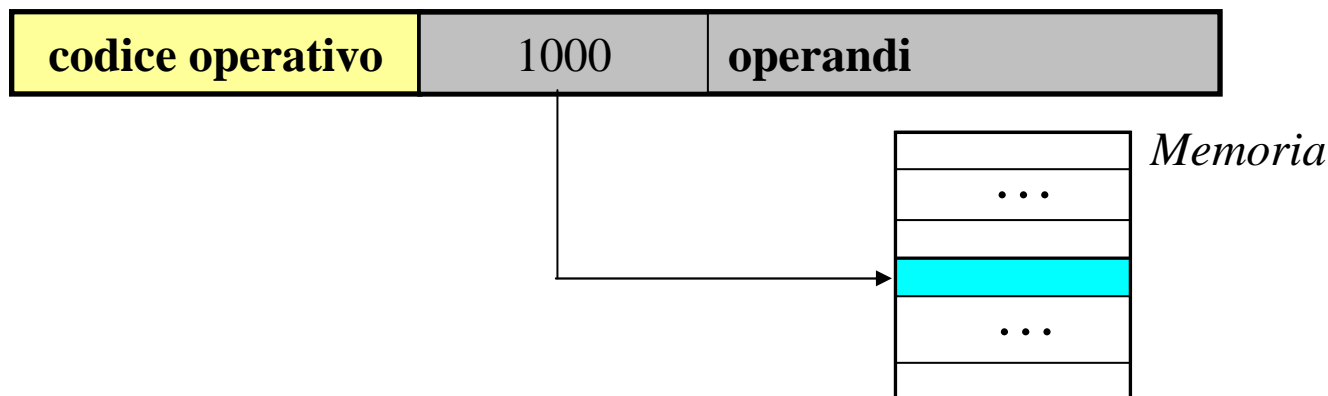
↑ ↑ ↑
 sorgente sorgente
 ↑
 ↑
destinazione

- Un operando può essere:

- Un valore indicato nel campo dell'istruzione (*immediato*)



- Una parola della memoria centrale (*operando in memoria*)




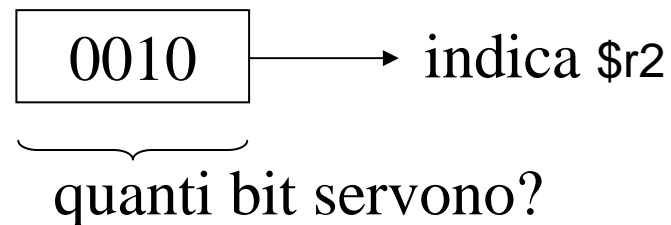
- Un registro dell'unità centrale (*operando registro*)

- Codifica degli operandi registro:

\$r0	4	→	0
\$r1	3	→	1
\$r2	5	→	2
\$r3	10	→	3
\$r4	2	→	4
...			...

Ognuno è indicato
da un numero univoco


 Per indicare un registro (in un campo): codifica binaria



- Codifica degli operandi registro:

\$r0	4	→	0
\$r1	3	→	1
\$r2	5	→	2
\$r3	10	→	3
\$r4	2	→	4
...			...

Ognuno è indicato
da un numero univoco

 Per indicare un registro (in un campo): codifica binaria

0010

 → indica \$r2

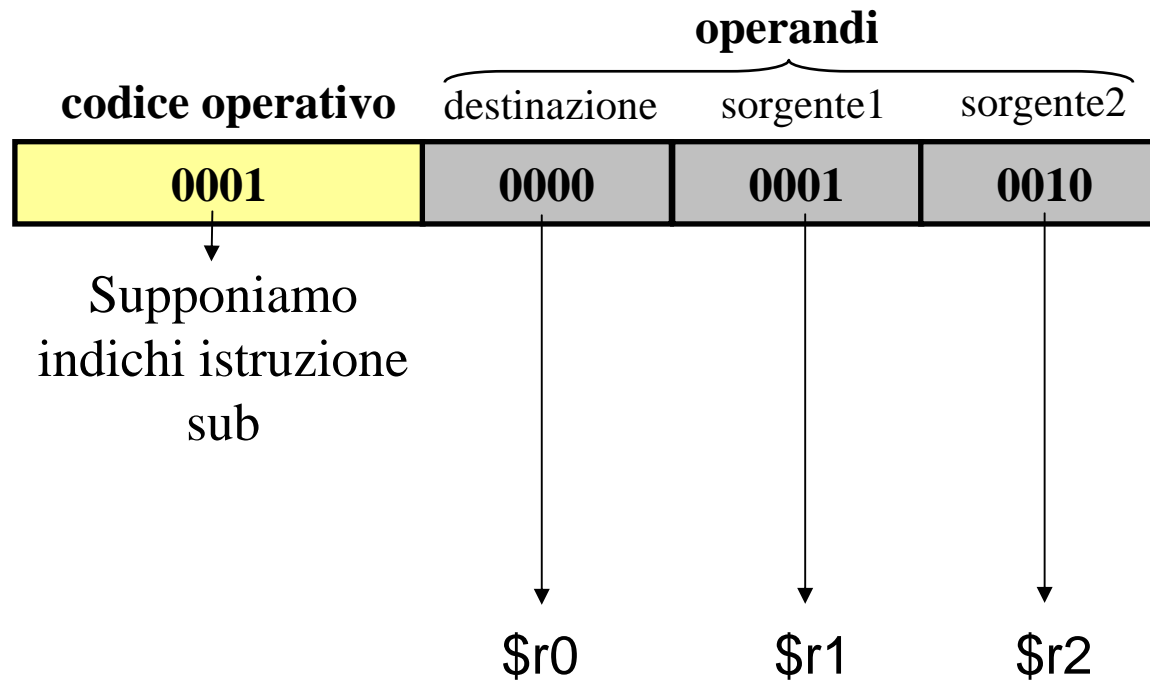
⏟
quanti bit servono?

Esempio: CPU con 32 registri

⇒ campi per gli operandi registro
saranno di 5 bit

Un esempio ipotetico di codifica (con operandi registro)


sub \$r0, \$r1, \$r2 \\ \$r0 = \$r1-\$r2



Linguaggio macchina vs. Linguaggio assembly

**Codice macchina di una procedura
che calcola e stampa la somma dei
quadrati degli interi fra 0 e 100**

```
0010011110111101111111111111100000
10101111110111111100000000000010100
1010111111010010000000000000100000
1010111111010010100000000000100100
101011111101000000000000000011000
101011111101000000000000000011100
100011111101011100000000000011100
100011111101110000000000000011000
000000011100111000000000000011001
00100101110010000000000000000001
00101001000000010000000001100101
101011111101010000000000000011100
00000000000000000111100000010010
00000011000011111100100000100001
00010100001000001111111111110111
101011111101110010000000000011000
00111100000001000001000000000000
100011111101001010000000000011000
00001100000100000000000011101100
00100100100001000000010000110000
100011111101111110000000000010100
001001111101111010000000000100000
0000001111100000000000000001000
00000000000000000001000000100001
```


Traduzione:
programma
chiamato
“assemblatore”

**Codice assembly di una procedura
che calcola e stampa la somma dei
quadrati degli interi fra 0 e 100**

```
addiu    $29, $29, -32
sw        $31, 20($29)
sw        $4, 32($29)
sw        $5, 36($29)
sw        $0, 24($29)
sw        $0, 28($29)
lw        $14, 28($29)
lw        $24, 24($29)
multu    $14, $14
addiu    $8, $14, 1
slti     $1, $8, 101
sw        $8, 28($29)
mflo     $15
addu     $25, $24, $15
bne      $1, $0, -9
sw        $25, 24($29)
lui      $4, 4096
lw        $5, 24($29)
jal      1048812
addiu    $4, $4, 1072
lw        $31, 20($29)
addiu    $29, $29, 32
jr        $31
move     $2, $0
```

Il linguaggio assembly

- Usare direttamente il formato binario per scrivere (e leggere) programmi sarebbe impraticabile \Rightarrow si usa il *linguaggio assembly* (o *assembler*)
- Il linguaggio assembly è la *rappresentazione simbolica* della codifica binaria usata dal calcolatore (linguaggio macchina)
- L'assembly è più leggibile:
 - utilizza *codici operativi simbolici* (anziché bit) che richiamano direttamente il significato di una istruzione (p.es. ADD al posto di 0001)
 - permette l'utilizzo di *etichette* per identificare gli indirizzi di parole di memoria che contengono istruzioni o dati (in questo caso le etichette possono essere indicate come “nomi di variabili”)
- *Assemblatore*: traduce linguaggio assembler in linguaggio macchina

La traduzione in linguaggio macchina binario

- Traduzione dei codici simbolici in corrispondenti codici binari:
mediante una tabella dei codici
(es: ad *add* corrisponde il codice operativo 00..1)
- Traduzione degli indirizzi simbolici (etichette) in indirizzi effettivi:
 - decisione di dove memorizzare il programma e i dati
 - individuazione, per ogni etichetta e nome di variabile, del corrispondente indirizzo effettivo
 - traduzione delle etichette simboliche nei corrispondenti indirizzi in binario