UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

ISA e indirizzamento

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 21 marzo 2025

RECAP

MEMORIA

«Indirizzamento al byte»

ALLINEAMENTO A WORD
a gruppi di 4 byte (=1word)

Le **istruzioni** sono già «naturalmente» di 32 bit, quindi vengono scritte in 4 celle di memoria consecutive

0	0	1	0	1	1	0	0	**0**
1	0	0	1	1	1	1	0	1
0	0	0	1	1	1	1	0	2
1	0	0	1	0	0	1	0	3
1	0	0	1	1	1	0	1	**4**
0	0	1	1	1	0	0	1	5
1	1	1	1	0	0	1	0	6
0	0	1	1	0	0	1	1	7
1	0	0	0	1	1	1	1	**8**
0	1	1	0	0	1	0	0	9

WORD

WORD

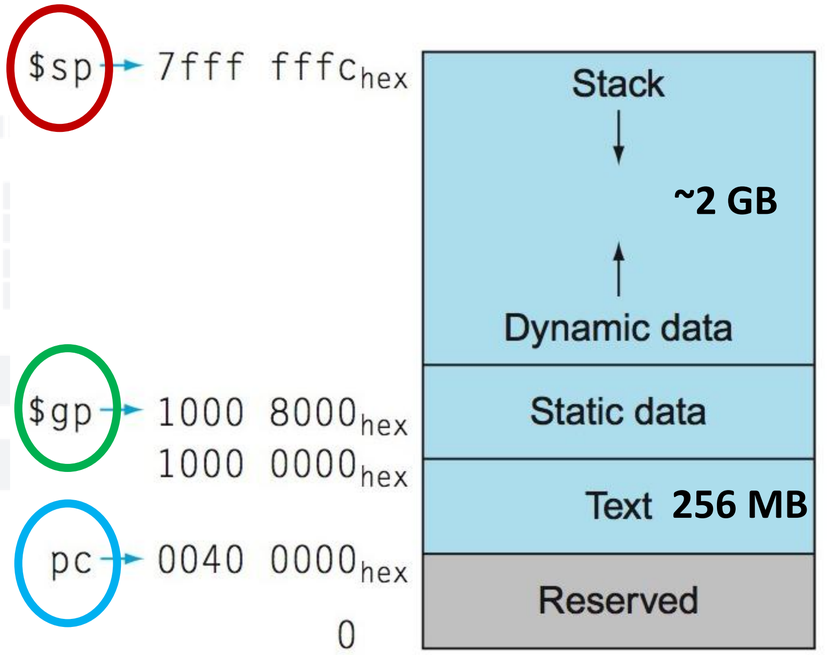
```
Little endian
+----+----+----+----+
|0x67|0x45|0x23|0x01|
+----+----+----+----+

Big endian
+----+----+----+----+
|0x01|0x23|0x45|0x67|
+----+----+----+----+
```

1100 **0000 0000 0000 0000 0100 0011 1011**

Può essere l'indirizzo di un'istruzione?

Dobbiamo verificare se è «allineato», ovvero se porta ad una cella di memoria che ha come indice un multiplo di 4.

Un INDIRIZZO MIPS32 ALLINEATO multiplo di 4 termina sempre con una cifra tra: 0, 4, 8, c



Primo programma Assembly

UNIVERSITÀ DEGLI STUDI DI TRIESTE

AGENDA DI IERI/OGGI

- *Organizzazione della memoria centrale*
- **Istruction Set Architecture (ISA)**
- Catena programmativa

FORMATO J-TYPE

Per codificare operazioni di salto (assoluto) a indirizzi specifici

J-type = jump-type

opcode	Target address

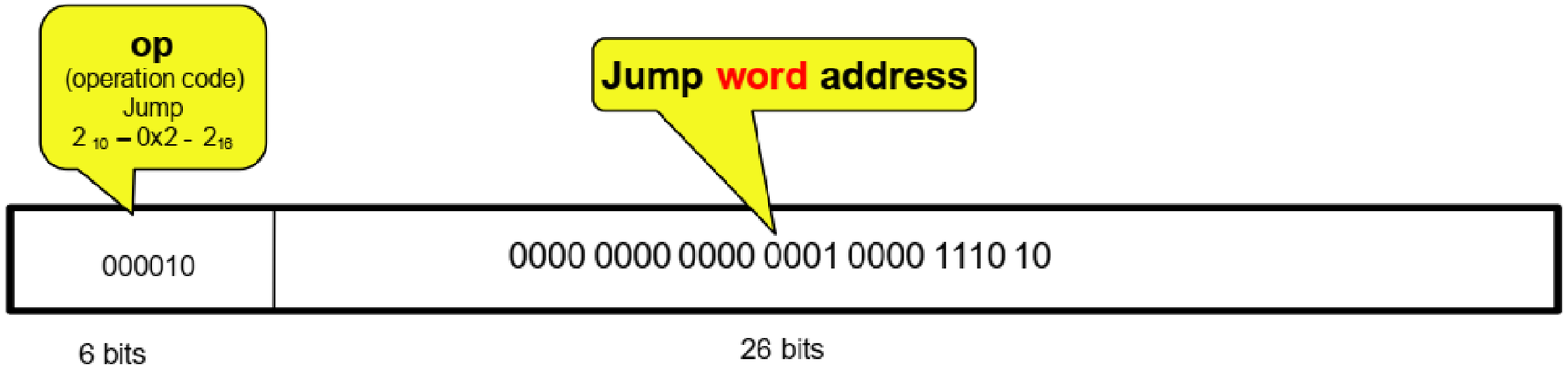
opcode (6 bit) → Indica che è istruzione di salto

Target address (26 bit) → Specifica l'indirizzo di destinazione

Salti assoluti.

FORMATO J-TYPE: ESEMPIO

00001000000000000000010000111010



jump

Jump

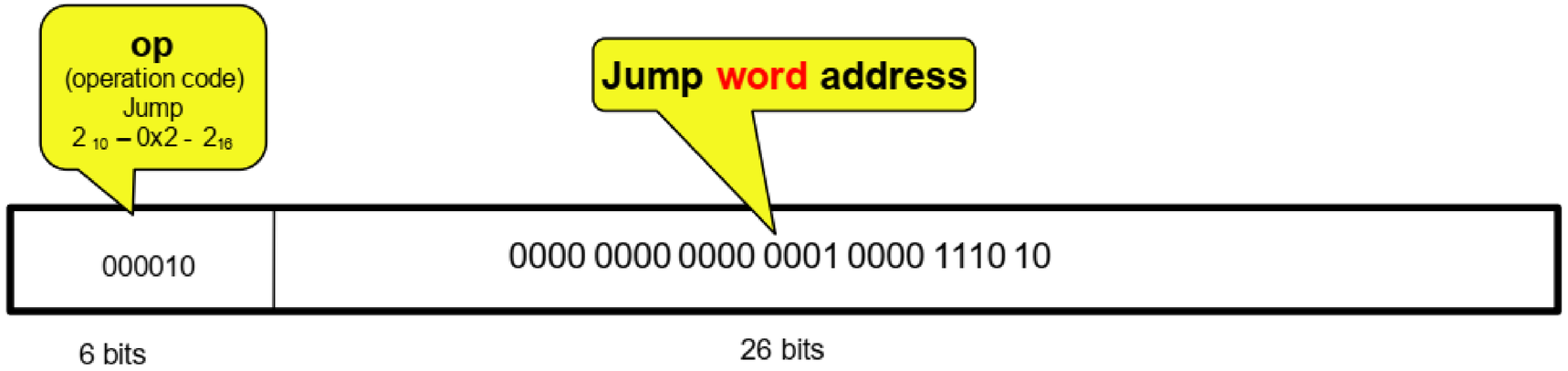
`j target`

2	target
6	26

Unconditionally jump to the instruction at target.

L'**indirizzo target** in un'istruzione J-type non è l'indirizzo completo, ma solo i **26 bit centrali**.

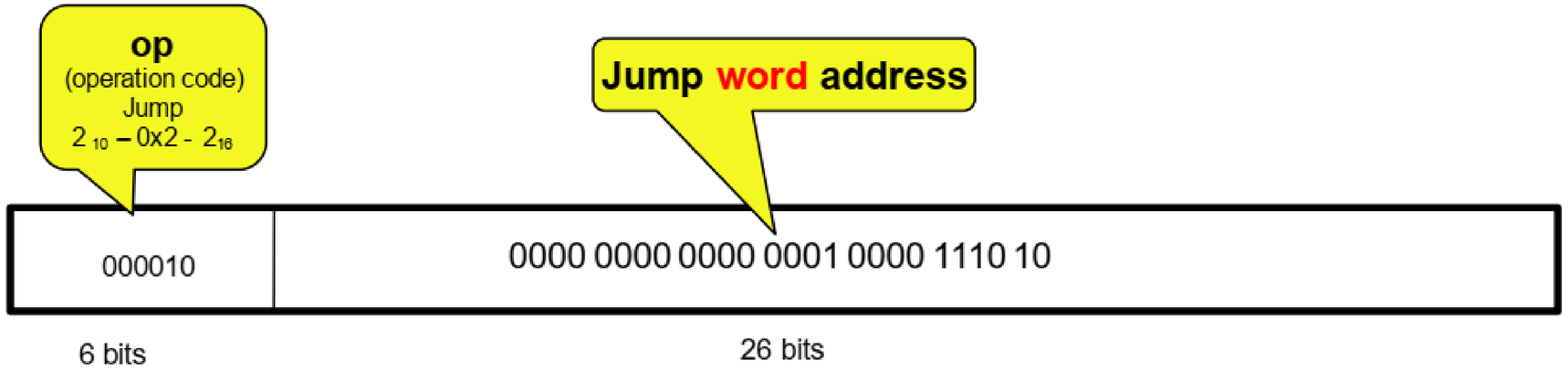
FORMATO J-TYPE: ESEMPIO



L'**indirizzo target** in un'istruzione J-type *non* è l'indirizzo completo, ma solo i **26 bit centrali**.

Per ottenere l'indirizzo effettivo bisogna **shiftare il valore a sinistra di 2 bit** (perché gli indirizzi MIPS sono allineati a 4 byte).

FORMATO J-TYPE: ESEMPIO



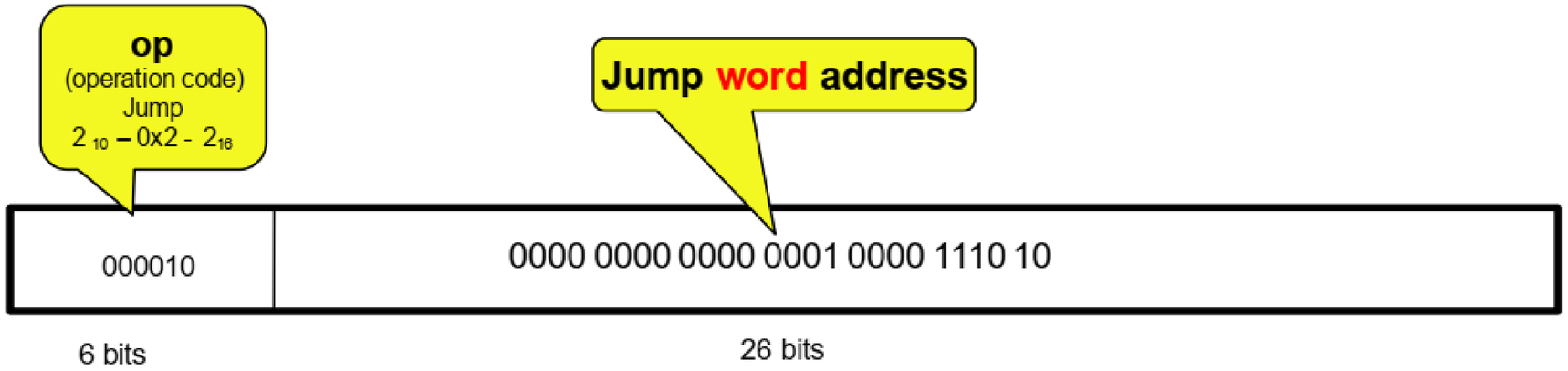
1.Shiftiamo il valore a sinistra di 2 bit (perché gli indirizzi MIPS sono allineati a 4 byte).

Codifica hex del target address: 0x00001E3A

Indirizzo finale di salto: 0x00001E3A << 2 = 0x000078E8

Nota. Ricordate che shiftare a sinistra di 2 bit equivale a **moltiplicare il numero per 4** (poiché 2^2=4).
In effetti: 0x**00001E3A** = **7738**$_{10}$
7738$_{10}$ x 4 = 30952$_{10}$ = 0x**000078E8**

FORMATO J-TYPE: ESEMPIO



2. Aggiungiamo i 4 bit più significativi del PC corrente

Leggo il PC(*) 0000 0000 0100 0000 0000 0000 0011 0100

Compongo l'indirizzo finale 0000 0000 0000 0000 0001 0000 1110 1000

Carico in PC l'indirizzo: **0000** 0000 0000 0000 0001 0000 1110 1000 = $000010E8_{16}$

Invece che eseguire l'istruzione che era prevista (*), verrà eseguita l'istruzione contenuta all'indirizzo $000010E8_{16}$

FORMATO J-TYPE: ESEMPIO

Quindi **con 26 *bit*** si indirizzano **2^{28} *byte* di memoria** programma oppure 2^{26} *word*

Il range di indirizzi a cui posso arrivare in questo modo di indirizzamento (relativo al PC) sono tutti quelli del tipo XXXX 0000 0000 0000 0001 0000 1110 10**00**

26 bit

26 bit → 2^{26} configurazioni → Ciascuna di esse indirizza 1 byte ogni 4, ovvero l'inizio di word consecutive

→ 2^{28} configurazioni, se lascio variare anche gli ultimi due bit(LSB) → 2^{28} indirizzi (se ammetto anche gli indirizzi non multipli di 4 byte)

FORMATO I-TYPE

Per codificare operazioni con costanti o accesso alla memoria.

I-type = immediate*-type
***costante**

opcode	rs	rt	Immediate (offset)

opcode (6 bit) → Indica il tipo di istruzione.

rs (5 bit) → Registro sorgente.

rt (5 bit) → Registro di destinazione.

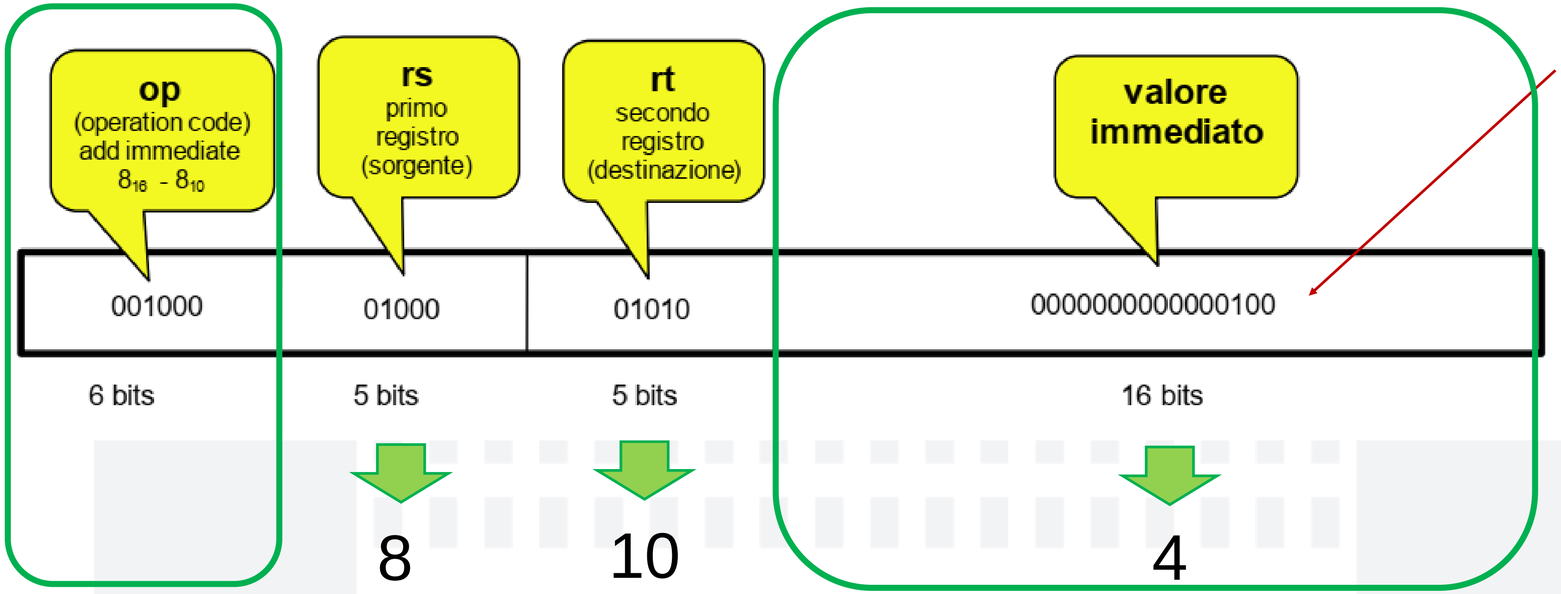
immediate (16 bit) → Valore immediato (costante o offset).

Operazioni aritmetiche con **valori immediati**.
Accesso alla memoria (load/store).
Salti condizionati.

FORMATO I-TYPE: ESEMPIO (ADD IMMEDIATE)

0010000100001010000000000000100

“somma il valore 4 al contenuto del registro 8 e metti il risultato nel registro 10”

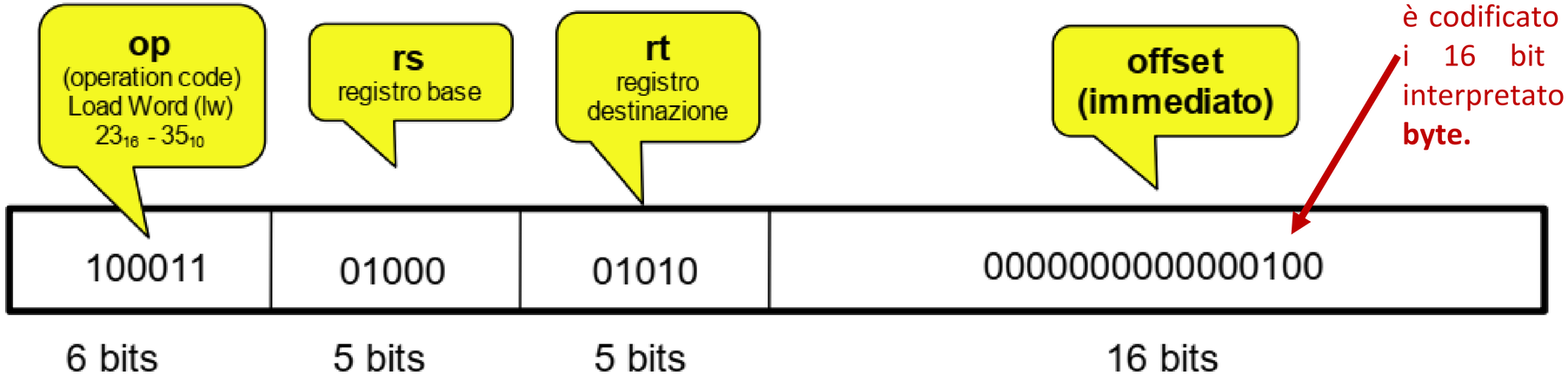


Nota. Qual è il **range di valori immediati** che si può esprimere con 16 bit in complemento a 2?
(-32768 <= valore <= 32767).

addi $10, $8, 4

UNIVERSITÀ DEGLI STUDI DI TRIESTE

FORMATO I-TYPE: LOAD WORD



Immediate: il valore decimale che è codificato in questo campo (con i 16 bit a disposizione) va interpretato come **numero di byte.**

Carica nel registro 10 il contenuto della parola (32 bit) che è **all'indirizzo di memoria** ottenuto come somma del contenuto del registro 8 e dell'offset immediato 4

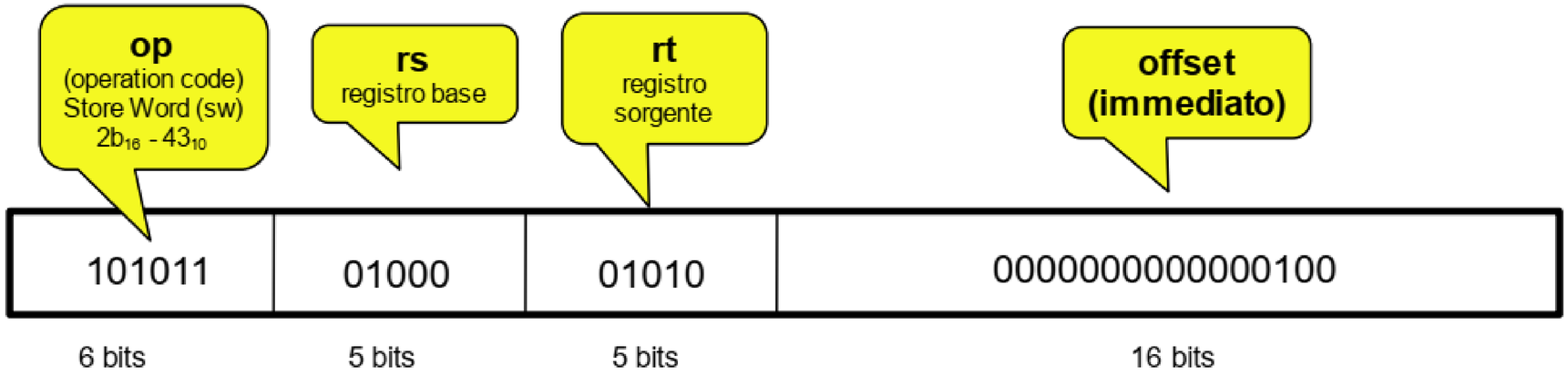
```
lw $10, 4($8)
```

Indirizzo di memoria di PROVENIENZA!

Essendo un usando la **numero di byte** e volendo noi accedere ad una word in memoria (stiamo load *word*), **esso deve sempre essere un multiplo di 4.**

lw $t0, **1**($s1) ❌ ERRORE DI ACCESSO NON ALLINEATO!

FORMATO I-TYPE: STORE WORD



Memorizza il contenuto del registro 10 (32 bit) all'indirizzo di memoria ottenuto come somma del contenuto del registro 8 e dell'offset immediato 4

```
sw $10, 4($8)
```

Indirizzo di memoria di DESTINAZIONE!

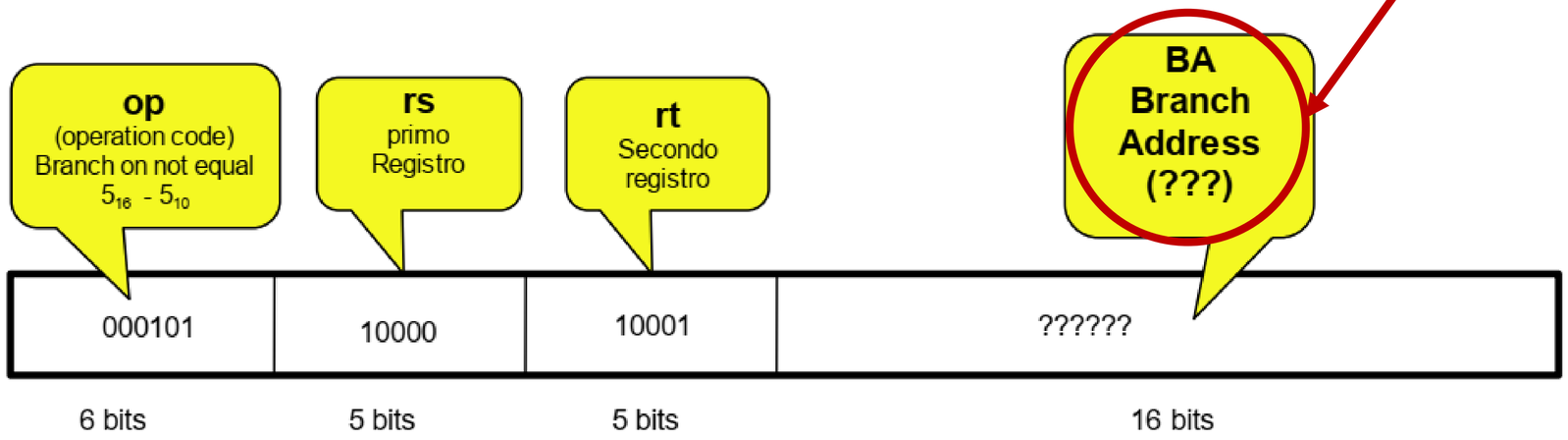
Essendo un usando la **numero di byte** e volendo noi accedere ad una word in memoria (stiamo store *word*), **esso deve sempre essere un multiplo di 4.**

FORMATO I-TYPE: BRANCH

Nota. Espressa in n. di **word**

Perché?

Ricorda: le istruzioni sono memorizzate su 4 Byte = 1 word (si rimanda alla discussione fatta durante la lezione)



BranchAddr rappresenta il numero di word (4Byte) da saltare rispetto all'indirizzo memorizzato nel *Program Counter (PC)*

```
bne $16, $17, Exit
```

Salta a Branch Address se il contenuto del registro 16 (rs) è diverso dal contenuto del registro 17 (rt)

INDIRIZZAMENTO IN MIPS32

L'**indirizzamento** in informatica e nelle architetture dei processori (come MIPS32) si riferisce al **modo in cui un'istruzione specifica l'ubicazione dei dati o dell'indirizzo di salto** all'interno della memoria o dei registri.

In altre parole, l'indirizzamento definisce **come il processore interpreta gli operandi** di un'istruzione per eseguire operazioni su dati memorizzati in registri, memoria o forniti direttamente nell'istruzione stessa.

CATEGORIE DI INDIRIZZAMENTO IN MIPS32 (1/2)

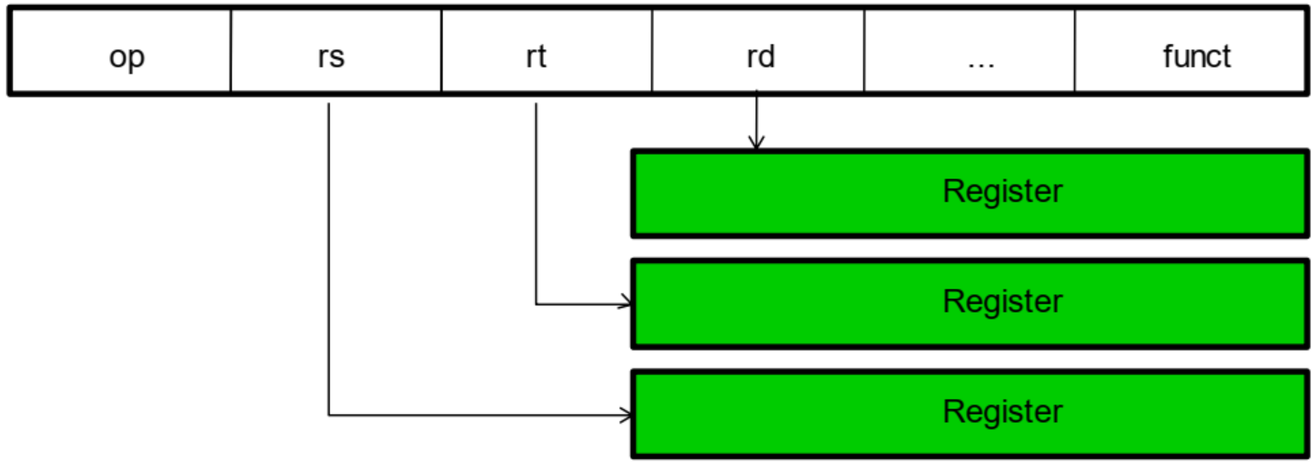
Immediate addressing

addi $t0, $zero, 10

L'operando è un valore costante incluso direttamente nell'istruzione.

Register addressing

add $10, $8, $9



L'operando è contenuto in un registro della CPU.

Jump addressing

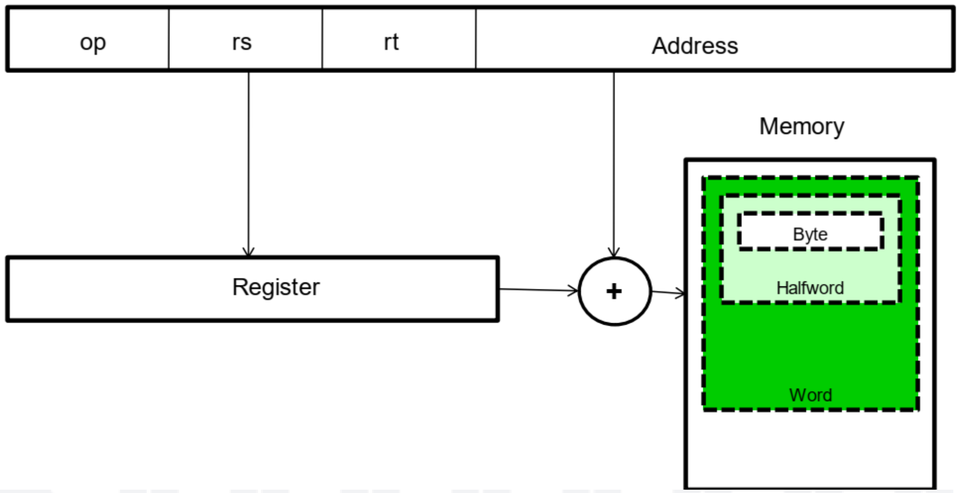
j 0x000010E8

L'operando è un **indirizzo assoluto** (parziale) memorizzato nei 26 bit dell'istruzione.

CATEGORIE DI INDIRIZZAMENTO IN MIPS32 (2/2)

Base addressing

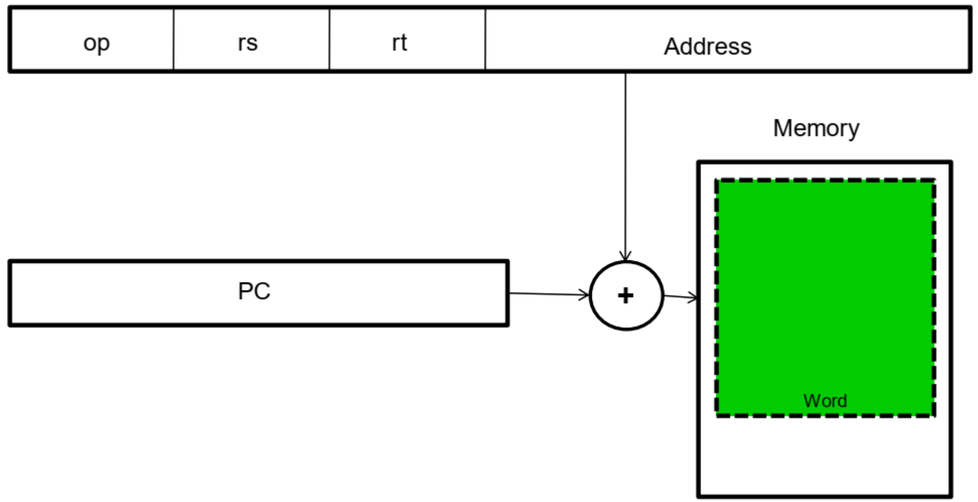
lw $4, 0($29)



L'operando è memorizzato in RAM, e si usa un registro base più un offset.

PC addressing

bne $16, $17, Exit



L'indirizzo di destinazione è calcolato sommando un offset al valore del Program Counter (PC).

ESERCIZIO 1

Determinare a quale istruzione macchina MIPS corrisponde la sequenza binaria

00100010111010110000000001100000

Soluzione nelle slide della prossima lezione

ESERCIZIO 2

A quale istruzione macchina MIPS corrisponde il

codice esadecimale: 0x8fa40000

Soluzione nelle slide della prossima lezione

Materiale per la lezione

- Lo stesso della lezione precedente