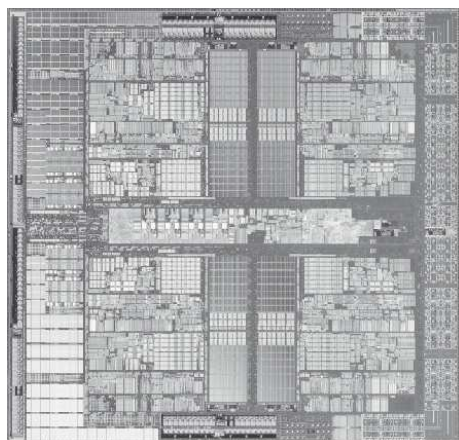




Università degli Studi
di Cassino



Corso di Calcolatori Elettronici

*Istruzioni di confronto
Istruzioni di controllo
Formato delle istruzioni in L.M.*

Anno Accademico 2010/2011

Francesco Tortorella

Istruzioni di confronto

Istruzione

Significato

slt \$t1,\$t2,\$t3	<i>if (\$t2<\$t3) \$t1=1; else \$t1=0</i>	<i>Confronto tra registri (con segno)</i>
slti \$t1,\$t2,100	<i>if (\$t2<100)\$t1=1; else \$t1=0</i>	<i>Cfr. registro-costante (con segno)</i>
sltu \$t1,\$t2,\$t3	<i>if (\$t2<\$t3) t\$1=1; else \$t1=0</i>	<i>Cfr. tra registri (senza segno)</i>
sltiu \$t1,\$t2,100	<i>if (\$t2<100) \$t1=1; else \$t1=0</i>	<i>Cfr. registro-costante (senza segno)</i>

Istruzioni per il controllo di flusso

- Sono istruzioni che permettono di alterare il flusso sequenziale di esecuzione delle istruzioni del programma, trasferendo il controllo ad una istruzione diversa da quella che segue immediatamente nel programma (“salto”).
- Se il salto è realizzato sulla base del verificarsi di una condizione, si parla di *istruzioni di salto condizionato*, altrimenti si definiscono *istruzioni di salto incondizionato*.

Istruzioni per il controllo di flusso

- A seconda di come viene specificata la destinazione del salto nella codifica in linguaggio macchina dell'istruzione, le istruzioni di salto si dividono in due classi:
 - **Istruzioni di *branch***
la destinazione del salto è specificata tramite uno spiazzamento rispetto al valore del PC.
 - **Istruzioni di *jump***
la destinazione del salto è specificata tramite un indirizzo assoluto.

Istruzioni di branch

beq \$t1,\$t2,label	<i>if (\$t1 == \$t2) branch label</i>	<i>Test di uguaglianza</i>
bne \$t1,\$t2,label	<i>if (\$t1 != \$t2) branch label</i>	<i>Test di disuguaglianza</i>
bgez \$t1,label	<i>if (\$t1 >= \$zero) branch label</i>	<i>Verifica se il valore in \$t1 è non negativo</i>
bgtz \$t1,label	<i>if (\$t1 > \$zero) branch label</i>	<i>Verifica se il valore in \$t1 è positivo</i>
blez \$t1,label	<i>if (\$t1 <= \$ zero) branch label</i>	<i>Verifica se il valore in \$t1 è non positivo</i>
bltz \$t1,label	<i>if (\$t1 < \$ zero) branch label</i>	<i>Verifica se il valore in \$t1 è negativo</i>

Istruzioni di jump

<i>Istruzione</i>	<i>Significato</i>	
j label	<i>jump label</i>	<i>Salta all'istruzione all'indirizzo label (indirizzo da 26 bit)</i>
jr \$t0	<i>jump (\$t0)</i>	<i>Salta all'istruzione all'indirizzo contenuto nel registro \$t0 (indirizzo da 32 bit)</i>
<i>Uso particolare di jr</i>		
jr \$ra	<i>jump (\$ra)</i>	<i>Per i ritorni da procedura (indirizzo da 32 bit)</i>

Istruzioni per la chiamata di sottoprogramma

Sono particolari istruzioni per il controllo di flusso. Prima del salto, l'indirizzo dell'istruzione successiva viene salvato nel registro \$ra (\$31).

bgezal \$t1,label	<i>if (\$t1 >= \$zero) { \$ra = PC + 4; branch label }</i>	<i>salto condizionato</i>
bltzal \$t1, label	<i>if (\$t1 < \$zero) { \$ra = PC + 4; branch label }</i>	<i>salto condizionato</i>
jal label	<i>\$ra = PC + 4; jump label</i>	<i>salto incondizionato</i>
jalr \$t1	<i>\$ra = PC + 4; jump (\$t1)</i>	<i>salto incondizionato</i>

Rappresentazione delle istruzioni

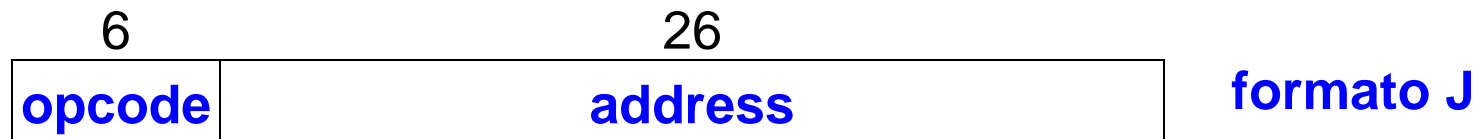
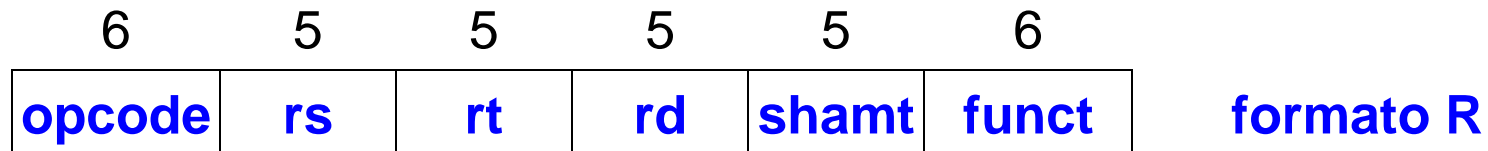
- Le istruzioni sono rappresentate tramite stringhe di bit memorizzate nei registri di memoria (come i dati numerici).
- Conseguenza: interi programmi (dati+istruzioni) possono essere memorizzati sullo stesso organo di memoria semplificando l'architettura degli elaboratori.
- Sia dati che istruzioni sono quindi identificabili tramite indirizzi.

Rappresentazione delle istruzioni

- I dati con cui opera il MIPS sono organizzati in word (blocchi da 32 bit)
 - Ogni registro ospita una word
 - lw e sw accedono in memoria una word alla volta
- Per mantenere semplice (e quindi efficiente) l'architettura del processore, anche le istruzioni sono di **dimensione fissa** e **pari ad una word**
- La word contenente un'istruzione è strutturata come un insieme di “campi” (fields), ognuno dei quali riporta un'informazione relativa all'istruzione.

Formati delle istruzioni

Esistono 3 possibili formati per le istruzioni:



Formati delle istruzioni

- Formato “I”: usato per le istruzioni con immediati, lw e sw (gli offset sono di fatto immediati) e i salti di tipo branch (beq e bne)
- Formato “J”: usato per i salti di tipo jump (j e jal)
- Formato “R”: usato per tutte le altre istruzioni

Formato R: significato dei campi

- **opcode**: specifica (parzialmente) il codice operativo presente nell'istruzione
 - Questo campo è pari a 0 per tutte le istruzioni con formato R
- **funct**: insieme con opcode, specifica esattamente il codice operativo presente nell'istruzione

Formato R: significato dei campi

- **rs** (Source Register): in genere usato per specificare il registro contenente il primo operando
- **rt** (Target Register): in genere usato per specificare il registro contenente il secondo operando
- **rd** (Destination Register): in genere usato per specificare il registro che riceverà il risultato dell'elaborazione
- **Note:**
 - Ogni campo contiene 5 bit (indirizza uno dei 32 registri)
 - Alcune istruzioni non usano determinati campi, es.:
 - mult e div usano solo rs e rt (hanno come destinazione hi e lo)
 - mfhi e mflo usano solo rd (l'operando sorgente è specificato dall'istruzione)

Formato R: significato dei campi

- **shamt** (SHift AMounT): contiene l'ampiezza dello scorrimento eseguito da un'istruzione di shift. Su una word di 32 bit il massimo scorrimento possibile è di 31 posizioni, perciò questo campo è costituito da 5 bit.
- **Note:**
 - Questo campo è messo a 0 in tutte le istruzioni diverse da quelle di shift

Formato R: esempio

- Consideriamo l'istruzione `add $t0,$t1,$t2`
- Consultiamo il manuale MIPS

Addition (with overflow)

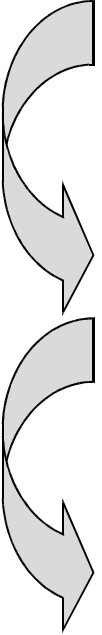
`add rd, rs, rt`

0	rs	rt	rd	0	0x20
6	5	5	5	5	6

- I campi sono quindi così definiti:
 - opcode: 0
 - funct: 32
 - rd: 8 (\$t0)
 - rs: 9 (\$t1)
 - rt: 10 (\$t2)
 - shamt: 0

Formato R: esempio

add \$t0,\$t1,\$t2



6	5	5	5	5	6
0	9	10	8	0	32

opcode rs rt rd shamt funct

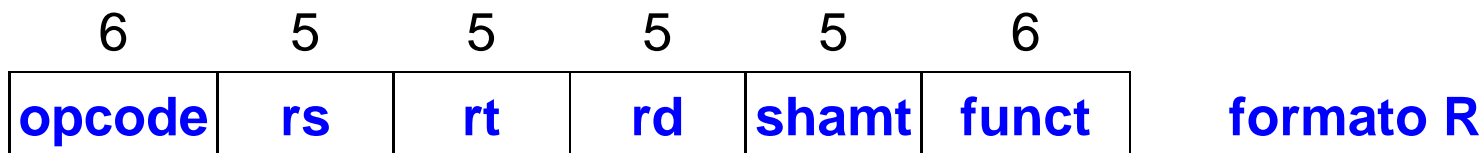
6	5	5	5	5	6
000000	01001	01010	01000	00000	100000

opcode rs rt rd shamt funct

Istruzione in linguaggio macchina

Formato I

- Il formato I è parzialmente consistente con il formato R (sui primi 16 bit), mentre gli altri 16 bit sono usati per ospitare un operando immediato



- Le istruzioni con questo formato usano al più due registri

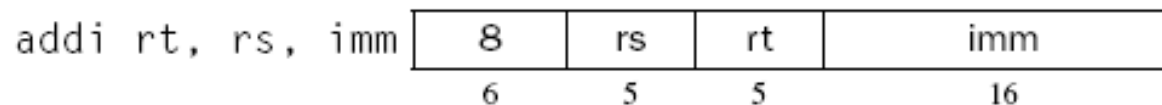
Formato I: significato dei campi

- **opcode**: specifica (completamente) il codice operativo presente nell'istruzione
- **rs** : specificare l'unico (eventuale) registro operando
- **rt**: specifica il registro che riceverà il risultato dell'elaborazione
- **immediate**: operando immediato che può essere
 - Un operando immediato vero e proprio (con sign extension): `addi $s5,$s6,-50`
 - Un offset (con sign extension) per specificare un indirizzo: `sw $t2,-64($20)`

Formato I: esempio

- Consideriamo l'istruzione `addi $s5,$s6,-50`
- Consultiamo il manuale MIPS

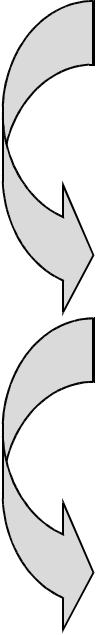
Addition immediate (with overflow)



- I campi sono quindi così definiti:
 - opcode: 8
 - rs: 22 (\$s6)
 - rt: 21 (\$s5)
 - immediate: -50

Formato I: esempio

addi \$s5,\$s6,-50



6	5	5	16
8	22	21	-50

opcode rs rt immediate

6	5	5	16
001000	10110	10101	1111111111001110

opcode rs rt immediate

Istruzione in linguaggio macchina

Formato I: questioni


- Come fare se l'operando immediato di un'istruzione è più ampio di 16 bit ? Es.: `addi $t0, $t0, 0xABCD1234`
- Soluzione software: si impiega l'istruzione **lui** per cui l'istruzione viene tradotta come:

```
lui $at, 0xABCD  
ori $at, $at, 0x1234  
add $t0, $t0, $at
```
- In questo modo, tutte le istruzioni hanno operandi immediati da 16 bit

Formato I: indirizzamento PC-relative

- Il formato I è usato anche dalle istruzioni di branch (beq, bne). Es.:

```
Loop:    beq    $t1,$zero,End
          add    $t0,$t0,$10
          addi   $t1,$t1,-1
          j      Loop
End:      slt    $t5,$t6,$t7
```



- Gli indirizzi delle istruzioni sono a 32 bit. Come rappresentare su 16 bit l'indirizzo cui saltare ?

Formato I: indirizzamento PC-relative

- In effetti, le istruzioni beq, bne vengono solitamente usate per realizzare costrutti di selezione (if, if-else) o cicli (while, for).
- In questi casi, l'ampiezza del salto è limitata: l'istruzione cui saltare è nelle vicinanze dell'istruzione di branch.
- Soluzione: non è necessario specificare l'intero indirizzo cui saltare, ma basta specificare lo spiazzamento rispetto all'indirizzo corrente (presente nel Program Counter) → **indirizzamento PC-relative**

Formato I: indirizzamento PC-relative

- Nel campo immediate viene quindi rappresentato l'offset dell'istruzione rispetto al **contenuto attuale del PC**.
 - L'offset sarà negativo se l'istruzione precede l'istruzione di branch, altrimenti sarà positivo.
- **Achtung**: al momento dell'aggiornamento, **il PC contiene l'indirizzo dell'istruzione successiva a quella di branch**.

Formato I: indirizzamento PC-relative

- **Da notare:** gli indirizzi delle istruzioni sono allineati alla word (multipli di 4), per cui si considera come offset **il numero di word** da aggiungere al PC
Ampiezza del salto: $\pm 2^{15}$ words = ± 128 Kbyte
- Esecuzione del branch:
 - Nel caso non si operi il salto: $PC = PC + 4$
 - In caso di salto: $PC = (PC + 4) + (\text{immediate} * 4)$

Formato I: esempio

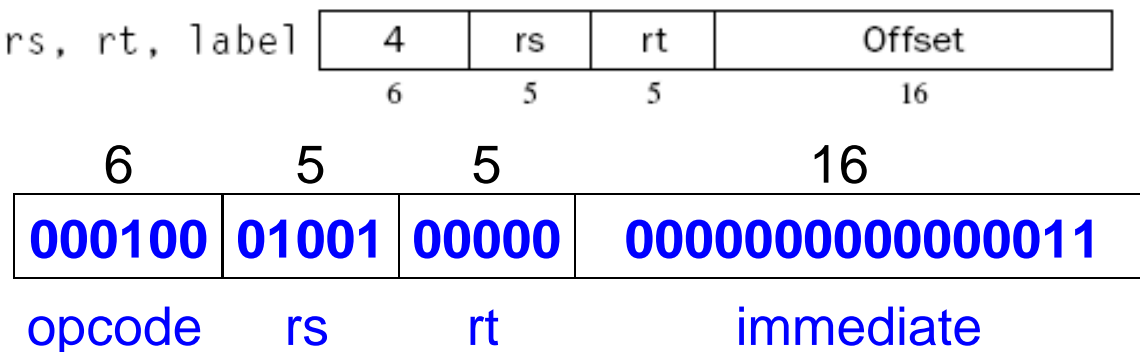
```
Loop:    beq    $t1,$zero,End
         add    $t0,$t0,$10
         addi   $t1,$t1,-1
         j      Loop
End:      slt    $t5,$t6,$t7
```

- Consultiamo il manuale MIPS

- opcode: 8
- rs: 9 (\$t1)
- rt: 0 (\$zero)
- immediate: +3

Branch on equal

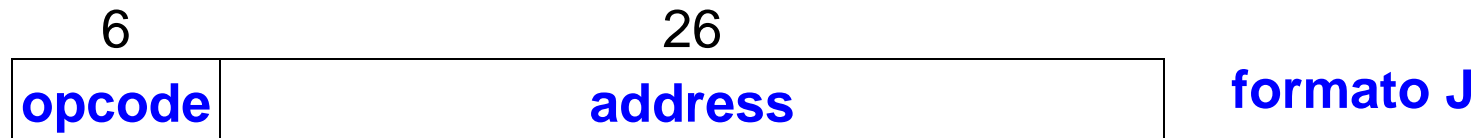
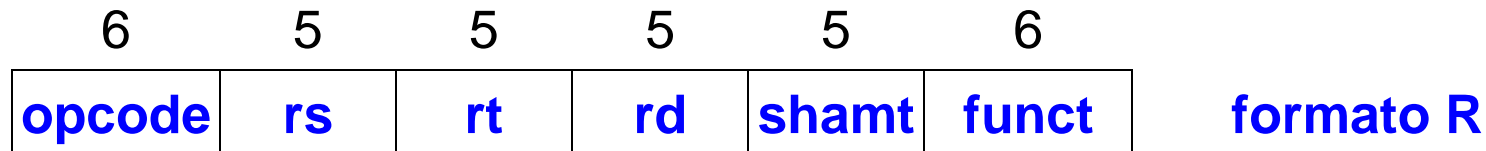
beq rs, rt, label



Formato J

- Le istruzioni di jump (j e jal) non pongono vincoli sulla destinazione del salto.
- In questo caso, non è possibile specificare lo spiazzamento, ma è necessario specificare (in linea di principio, l'intero indirizzo a 32 bit.
- Problema 1: lo spazio a disposizione per un'istruzione è di 32 bit. Come inserire l'indirizzo ?
- Problema 2: è necessario mantenere una consistenza con i formati R e I (primi 6 bit impegnati per opcode).

Formato J



- Restano quindi solo 26 bit per specificare l'indirizzo

Formato J

- Da notare: gli indirizzi delle istruzioni sono allineati alla word (ultimi 2 bit: 00) → quindi bisogna in effetti specificare solo i 30 bit più significativi.
- Soluzione: si assumono i 4 bit più significativi dal PC.
- In questo modo si possono rappresentare indirizzi in una pagina di 2^{26} word o di 256 Mbyte

