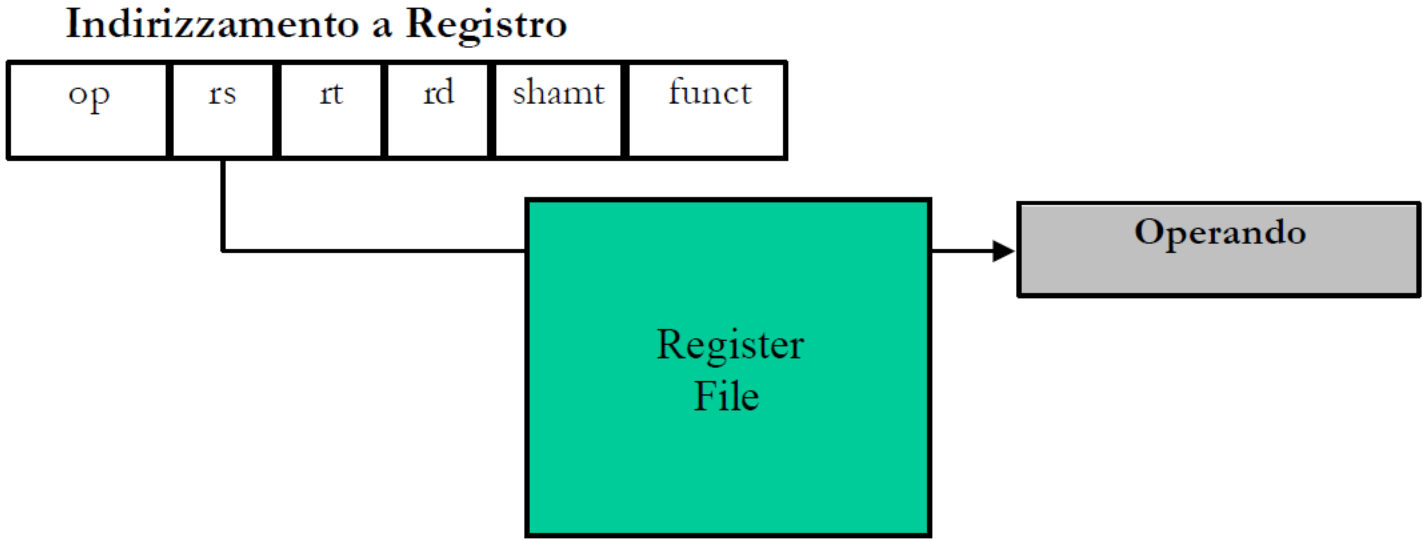
Modi di indirizzamento

- **Le modalità di indirizzamento indicano le diverse modalità attraverso le quali far riferimento ai dati ed alle istruzioni in memoria e nel register file.**
- **MIPS ha solo 5 modalità di indirizzamento:**
  - A registro
  - Immediato
  - Con base
  - Relativo al Program Counter
  - Pseudo-diretto

Indirizzamento a registro

- **L'operando è il contenuto di un registro della CPU: il nome (numero = indirizzo) del registro è specificato nell'istruzione.**



Esempi di indirizzamento a registro

- **Le istruzioni che usano solamente questo tipo di indirizzamento hanno formato di tipo R.**
- **Esempio: istruzione aritmetico-logica:**

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`add $s1, $s2, $s3`	000000	10010	10011	10001	00000	100000

Indirizzamento immediato

- **L'operando è una costante il cui valore è contenuto nell'istruzione.**
- **L'indirizzamento immediato si usa per specificare il valore di un operando sorgente, non ha senso usarlo come destinazione.**

- **Le istruzioni che usano questo tipo di indirizzamento hanno formato I**
  - La costante è memorizzata nel campo a 16-bit

Esempi di indirizzamento immediato

- **Esempio: operazione aritmetico-logica con operando immediato (formato tipo I):**

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`addi $s1, $s1, 4`	001000	10001	10001	0000 0000 0000 0100

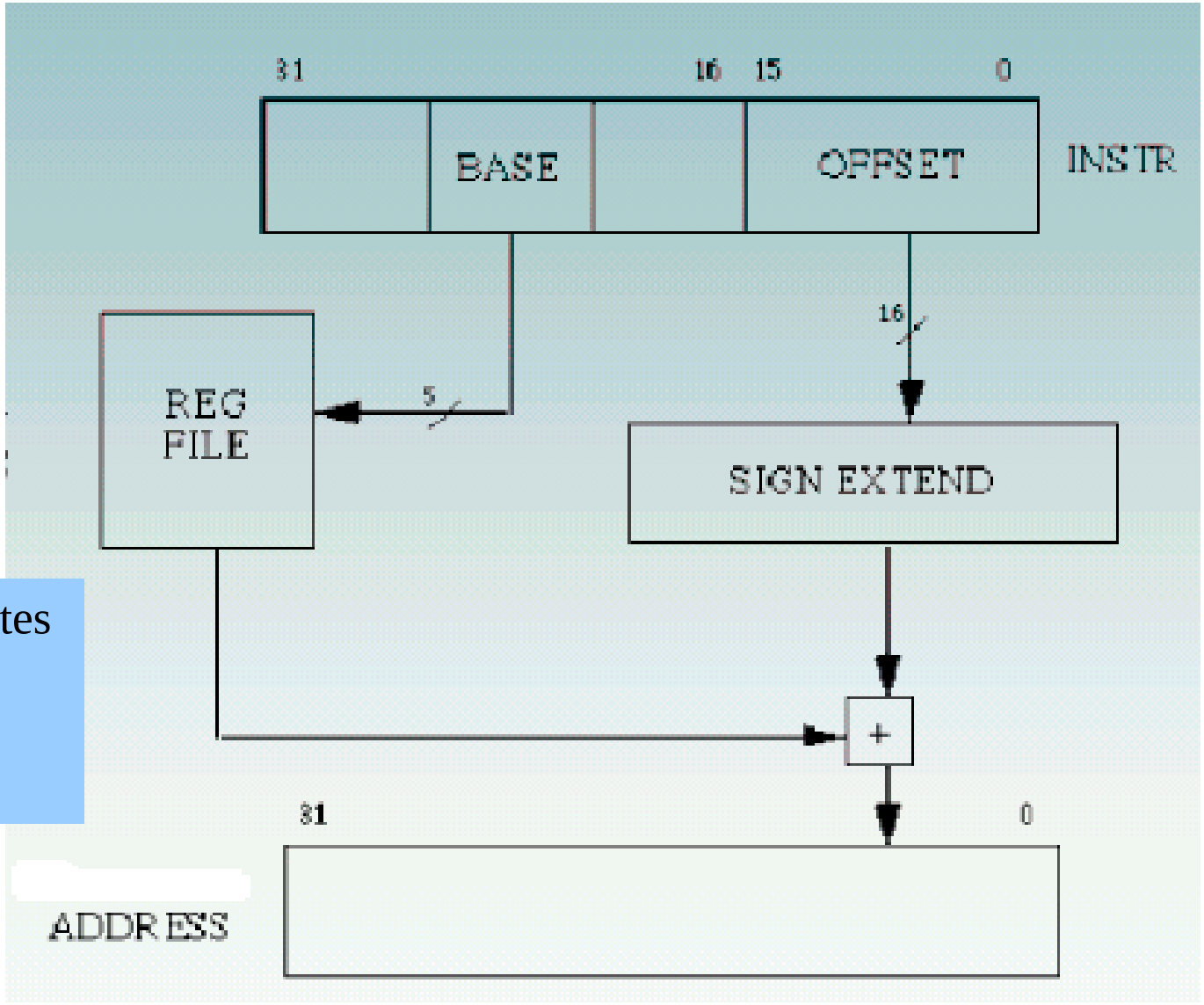
- **Esempio: operazione di confronto con operando immediato (formato tipo I):**

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`slti $t0, $s2, 8`	001010	10010	01000	0000 0000 0000 1000

Indirizzamento con base

- **L’operando è in una locazione di memoria il cui indirizzo si ottiene sommando il contenuto di un registro base ad un valore costante (*offset o spiazzamento)* contenuto nell’istruzione (sign-extended).**
- **Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo I.**

Indirizzamento con base



Intervallo: $\pm 2^{15}$ bytes
(± 32Kb)
(±8K parole)
dalla base

Esempi di indirizzamento con base

- **Esempio: istruzione di load - `lw $t0, 32 ($s3)`**
  - L'operando di trova in memoria all'indirizzo **`32+[$s3]`**
- **Esempio: istruzione di store - `sw $t0, 32 ($s3)`**
  - L'operando viene copiato in memoria all'indirizzo **`32+[$s3]`**

L'indirizzo e' espresso in numero di byte.

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`lw $t0, 32 ($s3)`	100011	10011	01000	0000 0000 0010 0000

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`sw $t0, 32 ($s3)`	101011	10011	01000	0000 0000 0010 0000

Indirizzamento relativo al PC

- **L'istruzione è in una locazione di memoria il cui indirizzo si ottiene sommando il contenuto del *Program Counter* ad un valore costante (*offset o spiazzamento*) contenuto nell'istruzione (sign-extended):**
- **Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo I.**

Indirizzamento relativo al PC

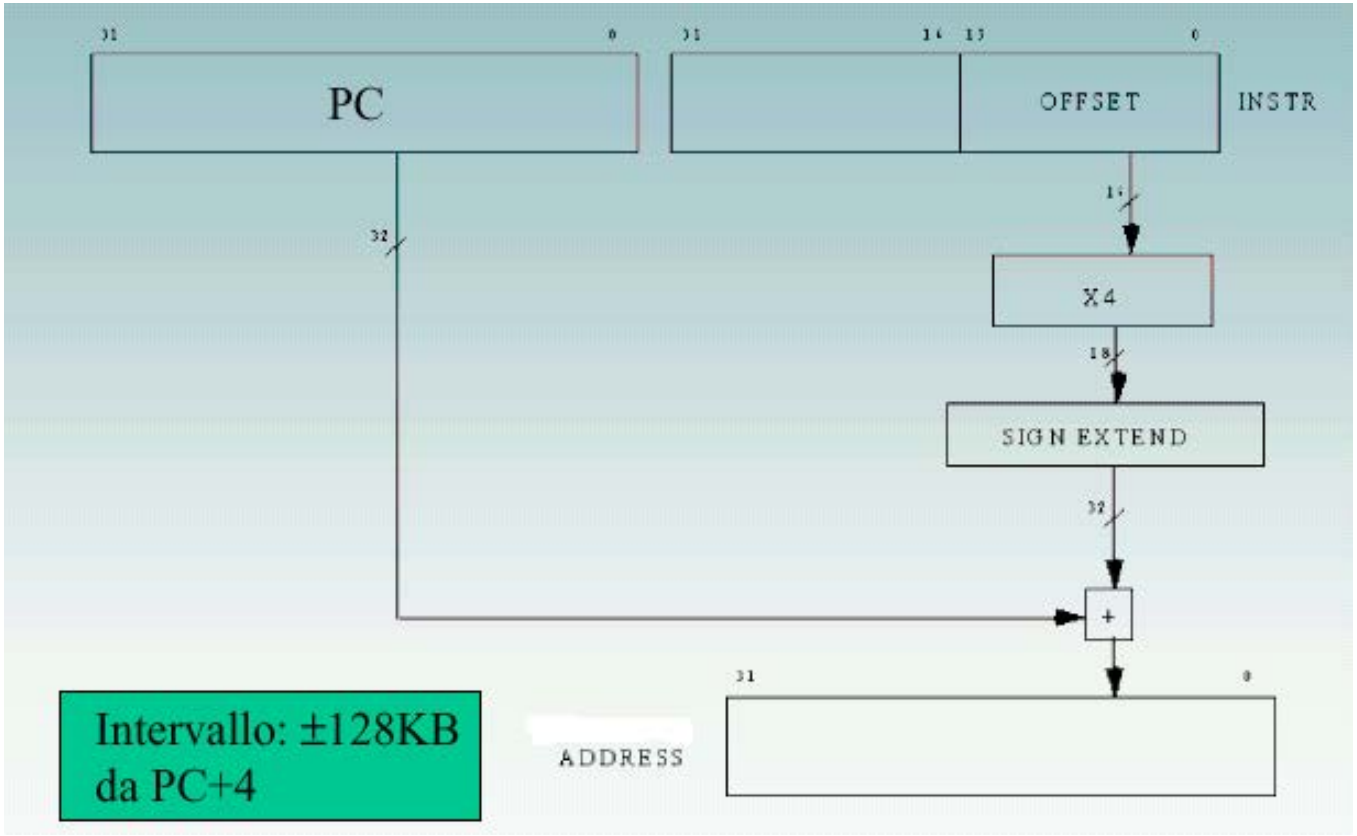
indirizzoMemoria = registro(PC) + *indirizzo di salto*

- si può saltare fino ad una distanza di $\pm 2^{15}$ ($\pm$32K) istruzioni (words, non bytes!) ($\pm$128KB) (principio di località)

Importante:

- **l'offset è relativo all'indirizzo dell'istruzione successiva (PC+4)**
- **l'offset usa un indirizzamento relativo alle parole (per avere l'indirizzo di salto, và moltiplicato per 4, prima di essere sommato a PC+4)**

Indirizzamento relativo al PC



Esempio di indirizzamento relativo al PC

- **Operazione di salto condizionato (formato tipo I).**
  - Si usa l'indirizzamento relativo al PC nei salti condizionati in quanto la destinazione del salto in tali istruzioni è in genere prossima al punto di salto.
  - Avendo a disposizione 16 bit di *Offset* ➔ è possibile saltare in un'area tra -2^{15} e $+2^{15}-1$ parole rispetto all'istruzione successiva.

Es: - istruzione beq a indirizzo 80000
- label a indirizzo 80104

#words da saltare a partire da PC+4

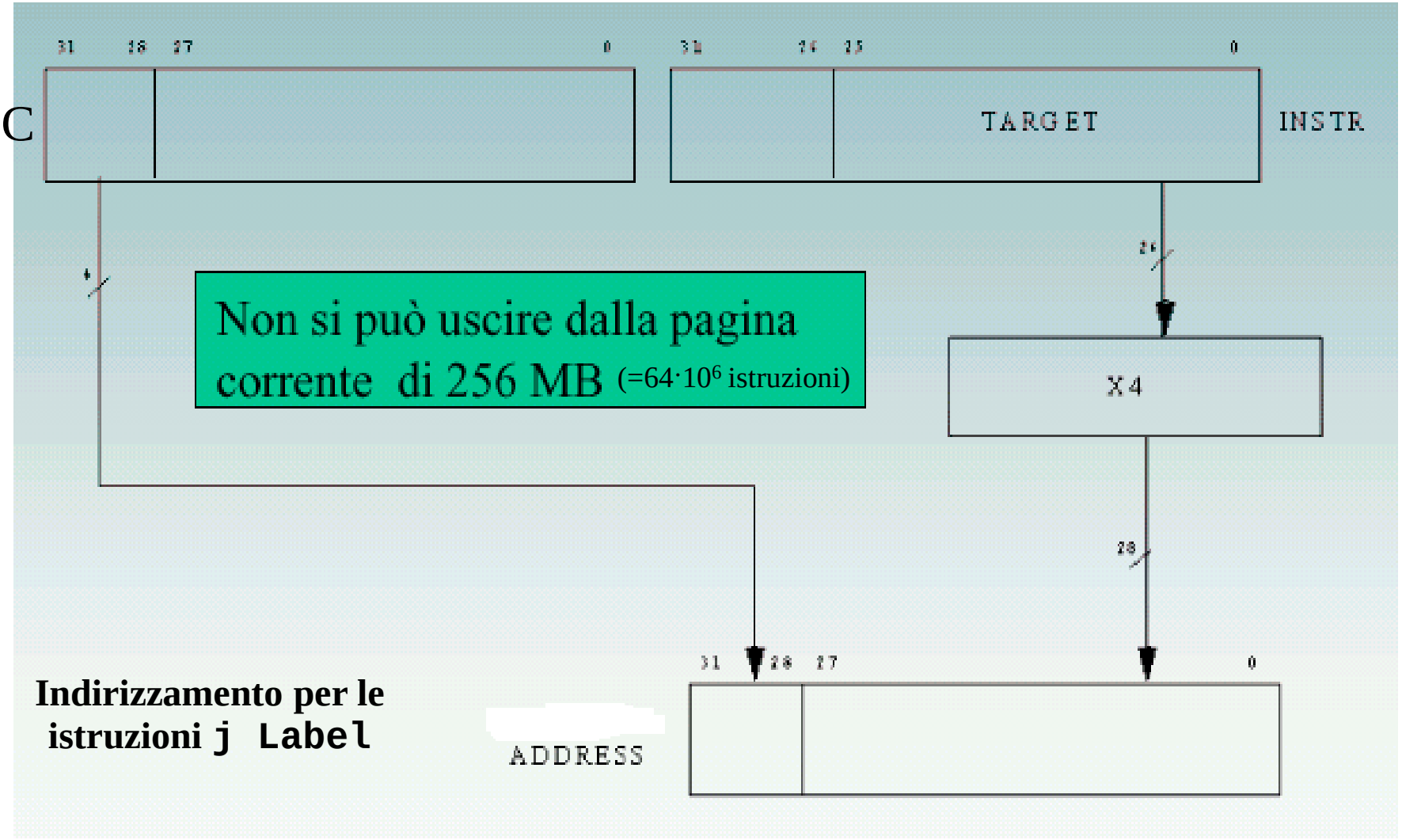
Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`beq $s1, $s2, label`	000100	10001	10010	0000 0000 0001 1001

25

Indirizzamento pseudo-diretto

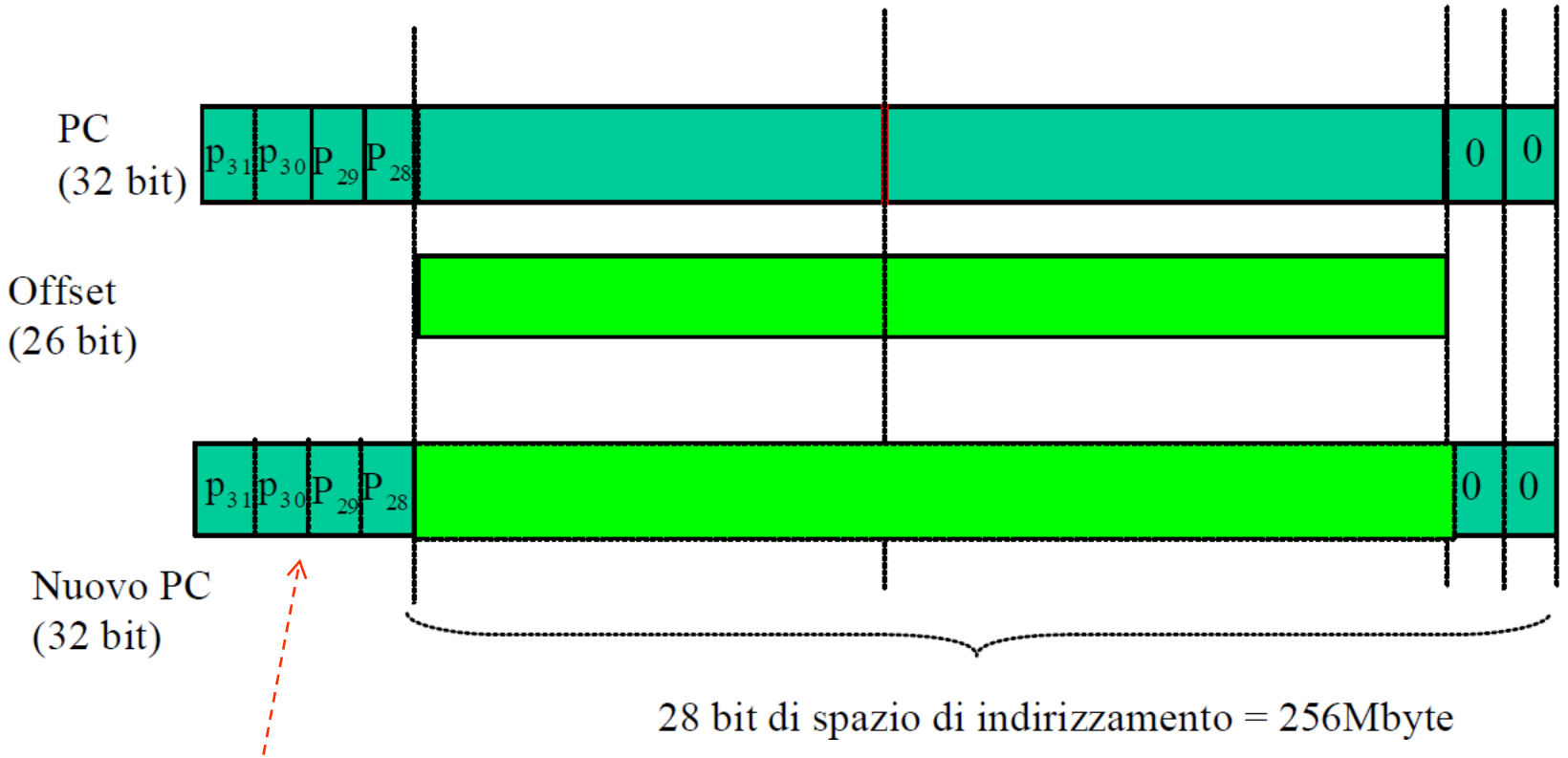
- Una parte dell'indirizzo è presente come valore costante (offset) nell'istruzione ma deve essere completato.
- Le istruzioni che usano questo tipo di indirizzamento hanno formato di tipo J.
- L'offset su 26 bit usa un **indirizzamento relativo alle parole** (come per l'indirizzamento relativo al PC), quindi va moltiplicato per 4 per indirizzare al byte (vedi sotto)
- L'indirizzo di salto si calcola facendo uno **shift a sinistra di 2 bit dei 26-bit di offset contenuti nell'istruzione** (aggiungendo 00 nei bit meno significativi per passare da 26 a 28-bit) e **concatenando i 28-bit con i 4-bit più significativi del Program Counter.**

Indirizzamento pseudo-diretto



Indirizzamento per le istruzioni `j Label`

Indirizzamento pseudo-diretto: utilizzo dell'offset



Esempio di indirizzamento pseudo-diretto

- **Operazione di salto incondizionato (formato J)**

Es: - istruzione j a indirizzo 0x 56767250
- label a indirizzo 0x 50000020

Nome campo	op	indirizzo				
Dimensione	6-bit	26-bit				
j label	000010	00 0000	0000	0000 0000	0000	1000

8 words
= 32 bytes

Pseudoistruzioni

- **Sono istruzioni accettate dall'assemblatore MIPS che non hanno corrispettivo nel linguaggio macchina per uno dei seguenti motivi:**
  - usano un'operazione **non implementata dall'hardware**
  - usano un modo di indirizzamento esteso, **non implementato dall'hardware**
- **L'assemblatore le espande in sequenze di (poche) istruzioni macchina, usando il registro `$1 ($at)`, riservato a questo scopo**

Pseudoistruzioni

- **aritmetiche**
  - abs, neg, negu, mul, mulo, mulou, rem, remu, div *(forma pseudo)*
- **logiche**
  - not
  - rol, ror
- **di trasferimento dati**
  - la, ld, ulh, ulhu, ulw, li
  - sd, ush, usw
  - move
- **di confronto**
  - seq, sge, sgeu, sgt, sgtu, sle, sleu, sne

Pseudoistruzioni

- **di controllo e salto**
  - b, beqz, bge, bgeu, bgt, bgtu, ble, bleu, blt, bltu, bnez
- **dei coprocessori**
  - mfc1.d
- **floating-point**
  - li.s(.d), l.s(.d), s.s(.d)

Pseudoistruzioni: esempi

- `la $a0, label`

```
lui $at, UpperPartLabelAddress
ori $a0, $at, LowerPartLabelAddress
```

- `bge $t0,$t1,address`

```
slt $at, $t0, $t1
beq $at, $zero, address
```

- `move $t0,$t2`

```
addu $t0, $zero, $t2
```

Pseudoistruzioni: esempi

- `abs $t1,$t0`

```
addu $t1,$zero,$t0
bgez $t0,L
sub $t1,$zero,$t0
L: ...
```

- `abs $t0,$t0`

```
bgez $t0,L
sub $t0,$zero,$t0
L: ...
```

Pseudoistruzioni: esempi

- `li $t0,-1`

```
lui $at,-1
ori $t0,$at,-1
```

- `div $t2,$t1,$t0`

```
    bne $t0,$zero,L
    break 0
L:  div $t1,$t0
    mflo $t2
```

Pseudo-Indirizzamento

- **Si ha quando il modo di indirizzamento usato non è direttamente fornito dal processore**

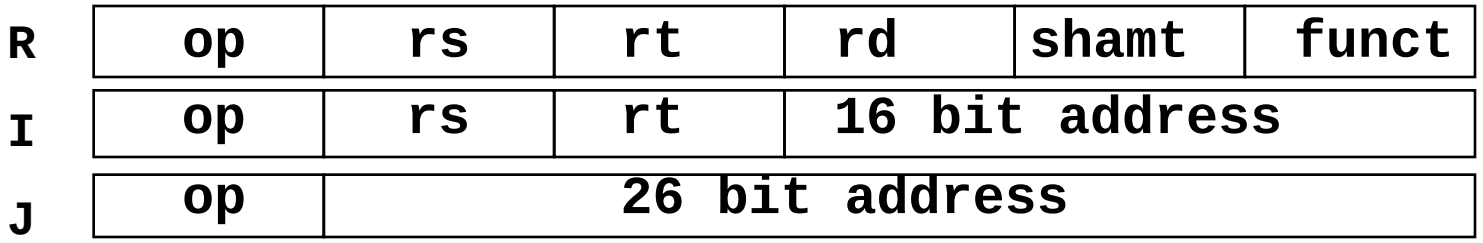
- **Anche in questo caso una singola (pseudo)istruzione viene tradotta in una sequenza di più istruzioni di macchina**

- **Esempio: istruzioni lw/sw. L'hw permette solo `imm(register)`, ma è possibile anche:**
  - `(register)`
  - `imm`
  - `label`
  - `label +/- imm`
  - `label +/- imm(register)`

Es. `sw $t0, L+16($s0)`

Riassumendo

- **Tutte le istruzioni sono lunghe 32 bits (1 word)**
- **Solo tre i formati delle istruzioni:**



MIPS operands

Name	Example	Comments
	`$s0-$s7, $t0-$t9, $zero,`	Fast locations for data. In MIPS, data must be in registers to perform
32 registers	`$a0-$a3, $v0-$v1, $gp,`	arithmetic. MIPS register $zero always equals 0. Register $at is
	`$fp, $sp, $ra, $at`	reserved for the assembler to handle large constants.
	Memory[0],	Accessed only by data transfer instructions. MIPS uses byte addresses, so
2^{30} memory	Memory[4], ...,	sequential words differ by 4. Memory holds data structures, such as arrays,
words	Memory[4294967292]	and spilled registers, such as those saved on procedure calls.

Riassumendo

MIPS assembly language				
Category	Instruction	Example	Meaning	Comments
	add	`add $s1, $s2, $s3`	`$s1 = $s2 + $s3`	Three operands; data in registers
Arithmetic	subtract	`sub $s1, $s2, $s3`	`$s1 = $s2 - $s3`	Three operands; data in registers
	add immediate	`addi $s1, $s2, 100`	`$s1 = $s2 + 100`	Used to add constants
	load word	`lw $s1, 100($s2)`	`$s1 = Memory[$s2 + 10`	Word from memory to register
	store word	`sw $s1, 100($s2)`	Memory[$s2 + 100] = $s1	Word from register to memory
Data transfer	load byte	`lb $s1, 100($s2)`	`$s1 = Memory[$s2 + 10`	Byte from memory to register
	store byte	`sb $s1, 100($s2)`	Memory[$s2 + 100] = $s1	Byte from register to memory
	load upper immediate	`lui $s1, 100`	$s1 = 100 * 2^{16}	Loads constant in upper 16 bits
	branch on equal	`beq $s1, $s2, 25`	if (`$s1 == $s2`) go to PC + 4 + 100	Equal test; PC-relative branch
Conditional branch	branch on not equal	`bne $s1, $s2, 25`	if (`$s1 != $s2`) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	`slt $s1, $s2, $s3`	if (`$s2 < $s3`) `$s1` = 1; else `$s1` = 0	Compare less than; for beq, bne
	set less than immediate	`slti $s1, $s2, 100`	if (`$s2 < 100`) `$s1` = 1; else `$s1` = 0	Compare less than constant
	jump	`j 2500`	go to 10000	Jump to target address
Uncondi-	jump register	`jr $ra`	go to `$ra`	For switch, procedure return
tional jump	jump and link	`jal 2500`	`$ra` = PC + 4; go to 1000(	For procedure call

Riassumendo

- **Tutte le istruzioni modificano tutti i 32 bits del registro destinazione (incluso lui, lb, lh) e tutte leggono i 32 bits della sorgente (add, sub, and, or, …)**
- **Le istruzioni Immediate aritmetiche e logiche sono estese nel seguente modo:**
  - gli operandi *logici* immediati sono "zero extended" a 32 bits
  - gli operandi *aritmetici* immediati sono "sign extended" a 32 bits (incluso addu)
- **I dati caricati da lb e lh sono estesi nel seguente modo:**
  - lbu, lhu "zero extended"
  - lb, lh "sign extended"
- **Overflow viene rilevato dall'hw in:**
  - add, sub, addi
- **Viene ignorato con:**
  - addu, subu, addiu, mult, multu, div, divu, …

Istruzioni Aritmetiche

Istruzione	Esempio	Significato	Commenti
add	**add $1,$2,$3**	**$1 = $2 + $3**	**3 *operandi; eccezione possibile***
add immediate	**addi $1,$2,100**	**$1 = $2 + 100**	***+ cost; eccezione possibile***
add unsigned	**addu $1,$2,$3**	**$1 = $2 + $3**	***3 operandi; nessuna eccezione***
add imm. unsign.	**addiu $1,$2,100**	**$1 = $2 + 100**	***+ costante; nessuna eccezione***
subtract	**sub $1,$2,$3**	**$1 = $2 – $3**	***3 operandi; eccezione possibile***
subtract unsigned	**subu $1,$2,$3**	**$1 = $2 – $3**	***3 operandi; nessuna eccezione***
multiply	**mult $2,$3**	**Hi, Lo = $2 x $3**	***64-bit prodotto con segno***
multiply unsigned	**multu $2,$3**	**Hi, Lo = $2 x $3**	***64-bit prodotto senza segno***
divide	**div $2,$3**	**Lo = $2 ÷ $3, Hi = $2 mod $3**	***Lo = quoziente, Hi = resto***
divide unsigned	**divu $2,$3**	**Lo = $2 ÷ $3, Hi = $2 mod $3**	***Quoziente & resto senza segno***

Istruzioni Logiche

Istruzione	Esempio	Significato	Commento
and	and $1,$2,$3	$1 = $2 & $3	3 reg. operands; Logical AND
or	or $1,$2,$3	$1 = $2 | $3	3 reg. operands; Logical OR
xor	xor $1,$2,$3	$1 = $2 ^ $3	3 reg. operands; Logical XOR
nor	nor $1,$2,$3	$1 = ~($2 |$3)	3 reg. operands; Logical NOR
and immediato	andi $1,$2,10	$1 = $2 & 10	Logical AND reg, constant
or immediato	ori $1,$2,10	$1 = $2 | 10	Logical OR reg, constant
xor immediato	xori $1, $2,10	$1 = ~$2 &~10	Logical XOR reg, constant
shift left logical	sll $1,$2,10	$1 = $2 << 10	Shift left by constant
shift right logical	srl $1,$2,10	$1 = $2 >> 10	Shift right by constant
shift right arithm.	sra $1,$2,10	$1 = $2 >> 10	Shift right (sign extend)
shift left logical	sllv $1,$2,$3	$1 = $2 << $3	Shift left by variable
shift right logical	srlv $1,$2, $3	$1 = $2 >> $3	Shift right by variable
shift right arithm.	srav $1,$2, $3	$1 = $2 >> $3	Shift right arith. by variable

Istruzioni di trasferimento dati

ISTRUZIONE	ESEMPIO	SIGNIFICATO
caricamento di una costante	li $s1, 10	$s1 = 10
caricamento costante di tipo indirizzo (32bit)	la $s1, *ind*	$s1 = *ind*
caricamento del byte memorizz. all'indirizzo *ind*	lb $s1, *ind*	$s1 = (*ind*)
caricamento della word memorizz. all'indirizzo *ind*	lw $s1, *ind*	$s1 = (*ind*)
memorizz. del byte contenuto nel registro $1 all'indirizzo *ind*	sb $s1, *ind*	*ind* = ($s1)
memorizz. della word contenuto nel registro $1 all'indirizzo *ind*	sw $s1, *ind*	*ind* = ($s1)

NOTA: "li" e "la" sono pseudoistruzioni

Istruzioni di confronto

Istruzione	Esempio	Significato	Commenti
set less than	slt $1,$2,$3	if ($2<$3) $1=1; else $1=0	*Confronto tra registri (con segno)*
set less than imm.	slti $1,$2,100	if ($2<100)$1=1; else $1=0	*Confronto registro-costante (con segno)*
set less than uns.	sltu $1,$2,$3	if ($2<$3) $1=1; else $1=0	*Confronto tra registri (senza segno)*
set l. t. imm. uns.	sltiu $1,$2,100	if ($2<100) $1=1; else $1=0	*Confronto registro-costante (senza segno)*

Branch e Branch&Link

Istruzione	Esempio	Significato	Commenti
branch on equal	beq $1,$2,L	if ($1 == $2) go to L	*Test di uguaglianza; PC-rel.*
branch on not eq.	bne $1,$2,L	if ($1!= $2) go to L	*Test di disuguaglinza; PC-rel.*
branch gr. eq. zero	bgez $1, L	if ($1>= $0) go to L	*Test di non negativo; PC-rel.*
branch gr. th. zero	bgtz $1, L	if ($1> $0) go to L	*Test di positivo; PC-rel.*
branch less eq. zero	blez $1, L	if ($1<= $0) go to L	*Test di non positivo; PC-rel.*
branch less th. zero	bltz $1, L	if ($1< $0) go to L	*Test di negativo; PC-rel.*
branch gr. eq. zero and link	bgezal $1, P	if ($1>= $0) { $31 = PC + 4; go to P}	*Test di non negativo; PC-rel.* *Per le chiamate a procedura*
branch less th. zero and link	bltzal $1, P	if ($1< $0) { $31 = PC + 4; go to P}	*Test di negativo; PC-rel.* *Per le chiamate a procedura*

Jump

Istruzione	Es.	Significato	Commenti
jump	j L	go to L	*Salta all'istruzione di etichetta L*
jump register	jr $31	go to $31	*Per i ritorni da procedura*
jump and link	jal P	$31 = PC + 4; go to P	*Per le chiamate di procedura*
jump and link reg.	**jalr $1**	$31 = PC + 4; go to $1	*Per le chiamate di procedura*