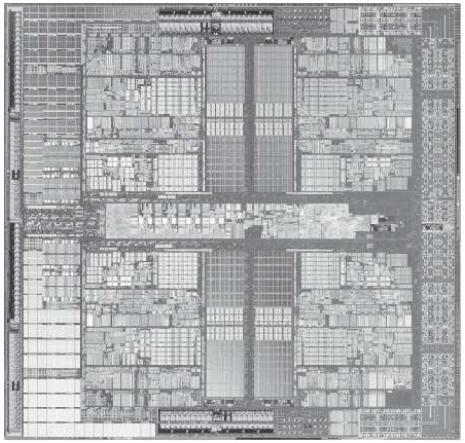
Università degli Studi di Cassino



Corso di Calcolatori Elettronici

Indirizzamento
Pseudo istruzioni

Anno Accademico 2010/2011

Francesco Tortorella

Modi di indirizzamento

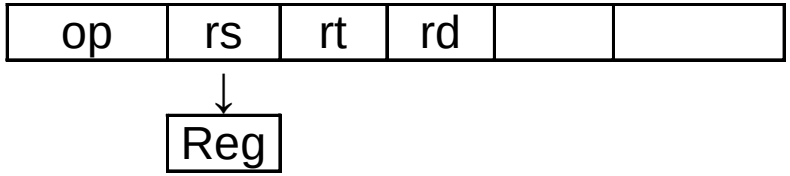
- Indicano come accedere all'operando di interesse dell'istruzione. Il MIPS ha 5 modi di indirizzamento:

- **immediato** (immediate)
  slti $1,$2,100
- **registro** (register)
  slti $1,$2,100
- **base/spiazzamento** (base/displacement)
  lw $t0,4($t1)
- **relativo rispetto al PC** (PC-relative)
  beq $1,$2,label
- **pseudodiretto** (pseudo direct)
  j label

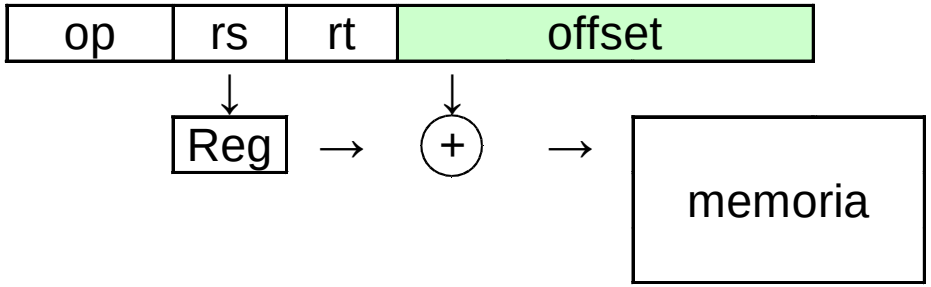
Modi di indirizzamento

immediato

registro



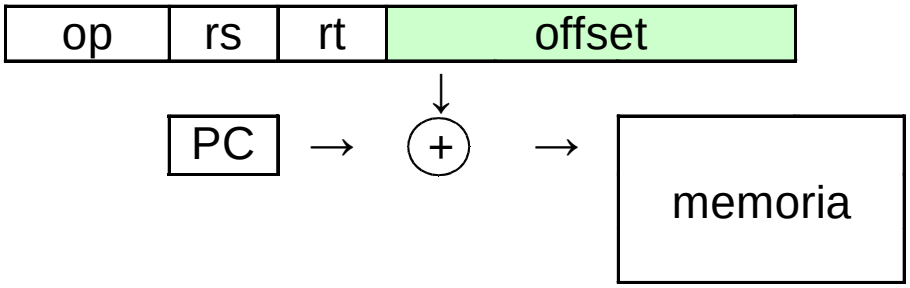
base (+offset)



Modi di indirizzamento

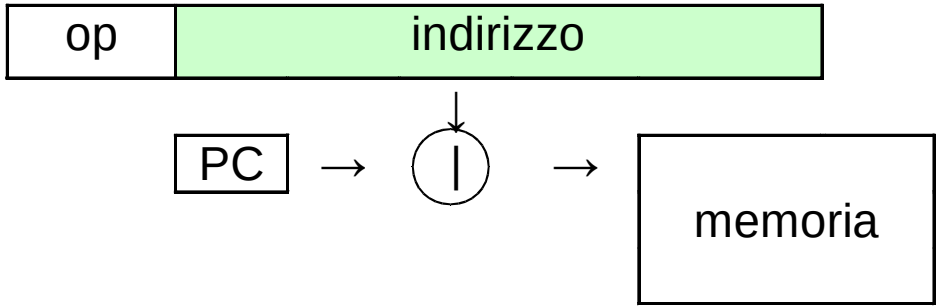
PC-relative (*branch*)

± 128 KB da PC + 4



PC-relative (*jump*)

entro la pagina di 256 MB a partire da PC + 4



Pseudoistruzioni

Sono istruzioni accettate dall'assemblatore MIPS alle quali non corrisponde un effettivo codice operativo in linguaggio macchina.

Prima della traduzione del programma in linguaggio macchina, le pseudo istruzioni vengono espanse dall'assemblatore in sequenze di istruzioni ammesse, usando il registro $1 ($at), riservato a questo scopo.

Una stessa pseudo istruzione può essere espansa in modi diversi, a seconda degli operandi contenuti.

Es.:

li $s0,0x1234 → ori $s0,$0,0x1234

li $s0,-1 → lui $1, -1
ori $s0,$1,-1

li $s0,0x12345678 → lui $1,0x1234
ori $s0,$1,0x5678

Pseudoistruzioni aritmetiche

Pseudoistr.	***Significato***	***Esempio***	
abs *rd,rs*	*rd*= ABS(*rs*)	abs $1,$2	
div *rd,rs,src*	*rd*=*rs* ÷ *src*	div $1,$2,100	*eccezione possibile*
divu *rd,rs,src*	*rd*=*rs* ÷ *src*	divu $1,$2,$3	*nessuna eccezione*
mul *rd,rs,src*	*rd*=*rs* x *src*	mul $1,$2,100	*eccezione possibile*
mulo *rd,rs,src*	*rd*=*rs* x *src*	mulo $1,$2,$3	*eccezione possibile*
mulou *rd,rs,src*	*rd*=*rs* x *src*	mulou $1,$2,$3	*unsigned*
			eccezione possibile
rem *rd,rs,src*	*rd*=*rs* mod *src*	rem $1,$2,100	*signed*
remu *rd,rs,src*	*rd*=*rs* mod *src*	remu $1,$2,100	*unsigned*

Altre pseudoistruzioni

Confronto	***Controllo***	***Load/Store***	***Move***
seq *rd,rs1,rs2*	b *label*	la *rd,address*	move *rd,rs*
sge *rd,rs1,rs2*	beqz *rs,label*	li *rd,costant*	
sgeu *rd,rs1,rs2*	bge *rs1,rs2,label*	ld *rd,address*	
sgt *rd,rs1,rs2*	bgeu *rs1,rs2,label*	ulh *rd,address*	
sgtu *rd,rs1,rs2*	bgt *rs1,rs2,label*	ulhu *rd,address*	
sle *rd,rs1,rs2*	bgtu *rs1,rs2,label*	ulw *rd,address*	
sleu *rd,rs1,rs2*	ble *rs1,rs2,label*	sd *rs,address*	
	bleu *rs1,rs2,label*	ush *rs,address*	
	blt *rs1,rs2,label*	usw *rs,address*	
	bltu *rs1,rs2,label*		
	bnez *rs,label*		

Altri modi di indirizzamento

L'assembler MIPS permette anche altri modi di indirizzamento oltre quelli direttamente implementati in hardware (*pseudo-indirizzamento*).
Vengono realizzati con i modi disponibili in una o più istruzioni.

diretto

lw $t0,vett → lui $1,vett$_{HIGH}$
lw $8,vett$_{LOW}$($1)

registro indiretto

lw $t0,($t1) → lw $8,0($9)

base+registro indiretto

lw $t1, vett+4($t0) → lui $1, vett$_{HIGH}$
addu $1, $1, $8
lw $9, {4+vett$_{LOW}$ }($1)

Direttive

Forniscono all'assemblatore istruzioni relative all'assemblaggio del programma. La loro interpretazione **non genera codice**, ma provoca lo svolgimento di particolari azioni da parte dell'assemblatore.

Principali direttive

```
.align n
.ascii str
.asciiz str
.byte b1,...,bn
.data <addr>
.double d1,…,dn
.extern sym
.float f1,…,fn
.globl sym
.half h1,…,hn
.kdata <addr>
.ktext <addr>
.set noat
.set at
.space n
.text <addr>
.word w1,...,wn
```

.align n

Allinea il dato successivo a blocchi di 2^n byte.

Es.: .align 2 allinea alla word il dato successivo

.align 0 elimina l'allineamento automatico

.ascii str

.asciiz str

Mette in memoria la stringa str (non) terminata dal carettere null

.byte b1, …, bn

.half h1,…,hn

.word w1,…,wn

Memorizza n byte (halfword,word) in parole consecutive della memoria

.space n

Alloca n byte a partire dall'indirizzo corrente.

.data <addr>

.kdata <addr>

Gli elementi successivi sono memorizzati nel segmento dati utente (kernel)

.text <addr>

.ktext <addr>

Gli elementi successivi sono memorizzati nel segmento testo utente (kernel)

Definizione costanti

Costanti Numeriche

12	decimale
0x2F	esadecimale

Costanti Carattere

Delimitate da doppi apici. Generano la sequenza di byte corrispondenti ai codici ASCII dei relativi caratteri

```
mesg: .asciiz “ciao”
mesg: .byte   $43,$49,$41,$4F,0
```

Chiamate di sistema (syscall)

- Sono un insieme di servizi di sistema messi a disposizione dall'ambiente di simulazione (sia SPIM che MARS) invocabili con l'istruzione **`syscall`**. La maggior parte di esse è dedicata all'input/output.
- Uso delle chiamate di sistema:
  1. Si carica il codice della system call in $v0
  2. Si caricano gli argomenti, se ci sono, in $a0, $a1, $a2 o $f12
  3. Si esegue l'istruzione **`syscall`**
  4. Si recuperano i risultati, se ci sono, nei registri specificati

Chiamate di sistema (syscall)

Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

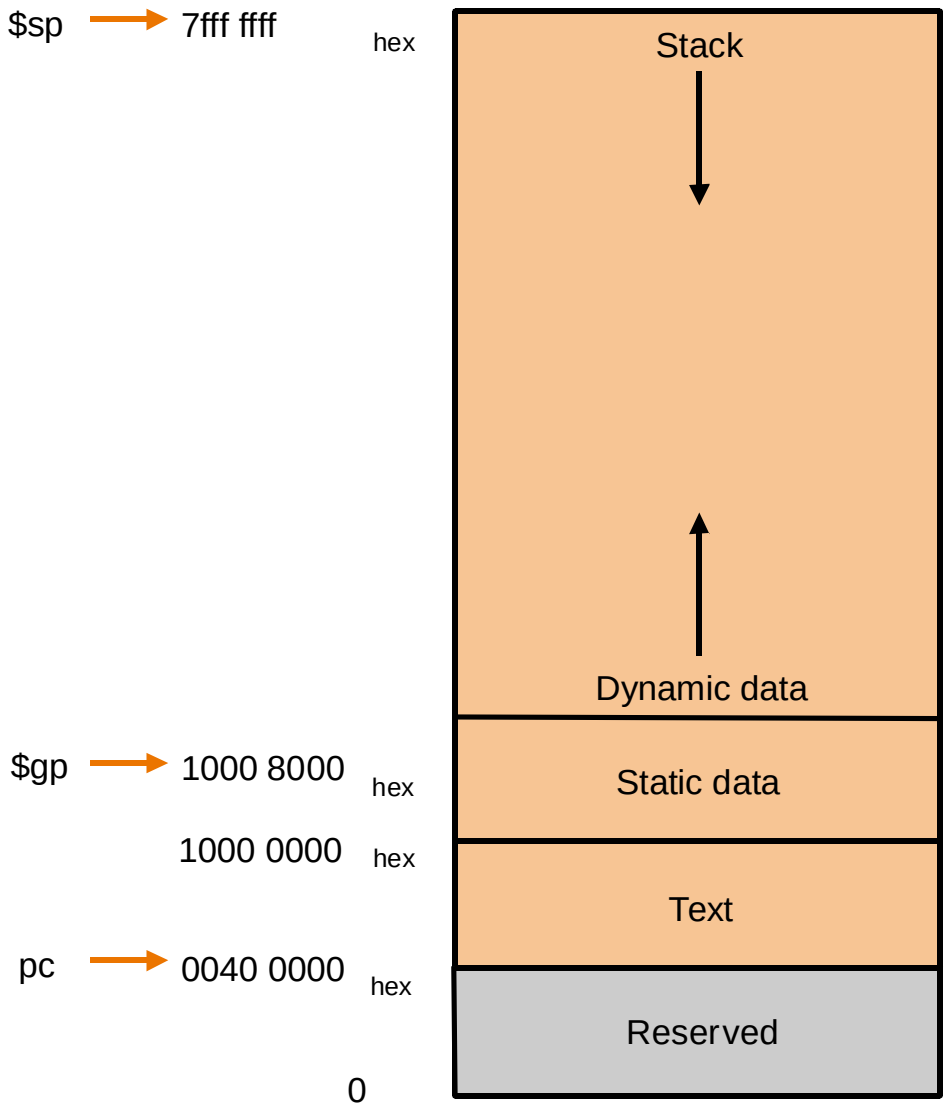
Esempio di chiamate di sistema

```
    .data
str:      .asciiz "the answer is "
    .align 2
W1: .word 5

    .text
    li $v0, 4     # system call code for print_str
    la $a0, str   # address of string to print
    syscall       # print the string

    li $v0, 1     # system call code for print_int
    lw $a0, W1    # integer to print
    syscall       # print it
```

Allocazione di memoria del MIPS



```
        .data
W1:     .word 10
W2:     .word 20,30
buf:    .space 12

        .text
        la $t0,buf
        lw $t1,W1
        sw $t1,($t0)
        lw $t1,W2
        sw $t1,4($t0)
        la $t2,W2
        lw $t1,4($t2)
        sw $t1,8($t0)

        li $v0,10
        syscall
```