

ISTRUZIONI E PSEUDO-ISTRUZIONI MIPS

ARITMETICA

<i>add</i>	\$1, \$1, \$3	$\$1 := \$2 + \$3$	addizione
<i>addu</i>	\$1, \$1, \$3	$\$1 := \$2 + \$3$	addizione naturale
<i>addi</i>	\$1, \$2, cost	$\$1 := \$2 + \text{cost}$	addizione di costante
<i>addiu</i>	\$1, \$2, cost	$\$1 := \$2 + \text{cost}$	addizione naturale di costante
<i>sub</i>	\$1, \$2, \$3	$\$1 := \$2 - \$3$	sottrazione
<i>subu</i>	\$1, \$2, \$3	$\$1 := \$2 - \$3$	sottrazione naturale
<i>mult</i>	\$1, \$2	$\text{hi} \text{lo} := \$1 \times \2	moltiplicazione (risultato a 64 bit)

ARITMETICA – pseudo-istruzioni

<i>subi</i>	\$1, \$2, cost	$\$1 := \$2 - \text{cost}$	sottrazione di costante
<i>subiu</i>	\$1, \$2, cost	$\$1 := \$2 - \text{cost}$	sottrazione naturale di costante
<i>neg</i>	\$1, \$2	$\$1 := -\2	negazione aritmetica (compl. a 2)

CONFRONTO

<i>slt</i>	\$1, \$2, \$3	if $\$2 < \3 then $\$1 := 1$ else $\$1 := 0$	poni a 1 se strettamente minore
<i>sltu</i>	\$1, \$2, \$3	if $\$2 < \3 then $\$1 := 1$ else $\$1 := 0$	poni a 1 se strettamente minore naturale
<i>slti</i>	\$1, \$2, cost	if $\$2 < \text{cost}$ then $\$1 := 1$ else $\$1 := 0$	poni a 1 se strettamente minore di costante
<i>sltiu</i>	\$1, \$2, cost	if $\$2 < \text{cost}$ then $\$1 := 1$ else $\$1 := 0$	poni a 1 se strettamente minore di costante naturale

LOGICA

<i>or</i>	\$1, \$2, \$\$3	$\$1 := \$2 \text{ or } \$3$	somma logica bit a bit
<i>and</i>	\$1, \$2, \$\$3	$\$1 := \$2 \text{ and } \$3$	prodotto logico bit a bit
<i>ori</i>	\$1, \$2, cost	$\$1 := \2 or cost	somma logica bit a bit con costante
<i>andi</i>	\$1, \$2, cost	$\$1 := \2 and cost	prodotto logico bit a bit con costante
<i>nor</i>	\$1, \$2, \$\$3	$\$1 := \$2 \text{ nor } \$3$	somma logica negata bit a bit
<i>sll</i>	\$1, \$2, cost	$\$1 := \$2 \ll \text{cost}$	scorrimento a sinistra (left) del numero di bit specificato da cost
<i>srl</i>	\$1, \$2, cost	$\$1 := \$2 \gg \text{cost}$	scorrimento a destra (right) del numero di bit specificato da cost

LOGICA – pseudo-istruzione

<i>not</i>	\$1, \$2	$s1 := \text{not } s2$	negazione logica (not bit a bit)
------------	----------	------------------------	----------------------------------

SALTO INCONDIZIONATO: ASSOLUTO, INDIRETTO E CON COLLEGAMENTO

<i>j</i>	indir	$\text{pc} := \text{indir (28 bit)}$	salto incondizionato assoluto
<i>jr</i>	\$1	$\text{pc} := \$1 \text{ (32 bit)}$	salto incondizionato indiretto da registro
<i>jal</i>	indir	$\text{pc} := \text{indir (28 bit)}$ e collega il registro \$ra	salto incondizionato assoluto con collegamento

SALTO CONDIZIONATO

<i>beq</i>	\$1, \$2, spi	if $\$1 = \2 salta relativo a PC	salto condizionato di uguaglianza
<i>bne</i>	\$1, \$2, spi	if $\$1 \neq \2 salta relativo a PC	salto condizionato di disuguaglianza

SALTO CONDIZIONATO – pseudo-istruzioni

<i>blt</i>	\$1, \$2, spi	if $\$1 < \2 salta relativo a PC	salta se strettamente minore
<i>bgt</i>	\$1, \$2, spi	if $\$1 > \2 salta relativo a PC	salta se strettamente maggiore
<i>ble</i>	\$1, \$2, spi	if $\$1 \leq \2 salta relativo a PC	salta se minore o uguale
<i>bge</i>	\$1, \$2, spi	if $\$1 \geq \2 salta relativo a PC	salta se maggiore o uguale

TRASFERIMENTO TRA PROCESSORE E MEMORIA

<i>lw</i>	\$1, spi (\$2)	\$1 := mem (\$2 + spi)	carica parola (a 32 bit)
<i>sw</i>	\$1, spi (\$2)	mem (\$2 + spi) := \$1	memorizza parola (a 32 bit)
<i>lh, lhu</i>	\$1, spi (\$2)	\$1 := mem (\$2 + spi)	carica mezza parola (a 16 bit)
<i>sh</i>	\$1, spi (\$2)	mem (\$2 + spi) := \$1	memorizza mezza parola (a 16 bit)
<i>lb, lbu</i>	\$1, spi (\$2)	\$1 := mem (\$2 + spi)	carica byte (a 8 bit)
<i>sb</i>	\$1, spi (\$2)	mem (\$2 + spi) := \$1	memorizza byte (a 8 bit)

TRASFERIMENTO TRA PROCESSORE E MEMORIA – pseudo-istruzioni (vedi nota 1 sotto)

<i>lw</i>	\$1, etichetta	\$1 := mem (\$gp + spi di etichetta)	carica parola (a 32 bit)
<i>sw</i>	\$1, etichetta	mem (\$gp + spi di etichetta) := \$1	memorizza parola (a 32 bit)

TRASFERIMENTO TRA REGISTRI (non referenziabili)

<i>mflo</i>	\$1	\$1 := lo	copia registro lo
<i>mfhi</i>	\$1	\$1 := hi	copia registro hi

TRASFERIMENTO TRA REGISTRI – pseudo-istruzione

<i>move</i>	\$1, \$2	\$1 := \$2	copia registro
-------------	----------	------------	----------------

CARICAMENTO DI COSTANTE IN REGISTRO

<i>lui</i>	\$1, cost	\$1 (16 bit più signif.) := cost	carica cost (in 16 bit più signif. di \$1) (16 bit meno signif. di \$1 posti a 0)
------------	-----------	----------------------------------	--

CARICAMENTO DI COSTANTE / INDIRIZZO IN REGISTRO – pseudo-istruzioni (vedi nota 2 sotto)

<i>li</i>	\$1, cost	\$1 := cost (32 bit)	carica costante a 32 bit
<i>la</i>	\$1, indir	\$1 := indir (32 bit)	carica indirizzo a 32 bit

REGISTRI MIPS**REGISTRI REFERENZIABILI**

0	<i>0</i>	costante 0 (denotabile anche come \$zero)
1	<i>at</i>	uso riservato all'assembler-linker (per espandere pseudo-istruzioni e macro)
2 - 3	<i>v0 - v1</i>	valore restituito da funzione (v0 per dati di tipo scalare, si aggiunge v1 per numeri reali di tipo double)
4 - 7	<i>a0 - a3</i>	argomenti in ingresso a funzione (max quattro)
8 - 15	<i>t0 - t7</i>	registri per valori temporanei (p.es. calcolo delle espressioni)
16 - 23	<i>s0 - s7</i>	registri usabili (se possibile) per variabili locali scalari di sottoprogramma
24 - 25	<i>t8 - t9</i>	registri per valori temporanei (in aggiunta a t0 - t7), come i precedenti tx
26 - 27	<i>k0 - k1</i>	registri riservati per il nucleo (kernel) del Sistema Operativo
28	<i>gp</i>	global pointer (puntatore all'area dati globale)
29	<i>sp</i>	stack pointer (puntatore alla pila)
30	<i>fp</i>	frame pointer (puntatore all'area di attivazione di sottoprogramma)
31	<i>ra</i>	return address (indirizzo di rientro da chiamata a sottoprogramma)

REGISTRI NON REFERENZIABILI

	<i>pc</i>	program counter (contatore di programma)
	<i>hi</i>	registro per risultato di moltiplicazione e divisione (32 bit più significativi)
	<i>lo</i>	registro per risultato di moltiplicazione e divisione (32 bit meno significativi)

Nota 1: anche le pseudo-istruzioni *lh*, *lhu*, *sh*, *lb*, *lbu* e *sb* con etichetta sottintendono il registro \$gp.

Nota 2: entrambe le pseudo-istruzioni *li* e *la* caricano un valore a 32 bit in un registro, ma *li* va usata per caricare una costante e *la* per caricare un indirizzo.