CODICI OPERATIVI MIPS

Linguaggio assembler MIPS

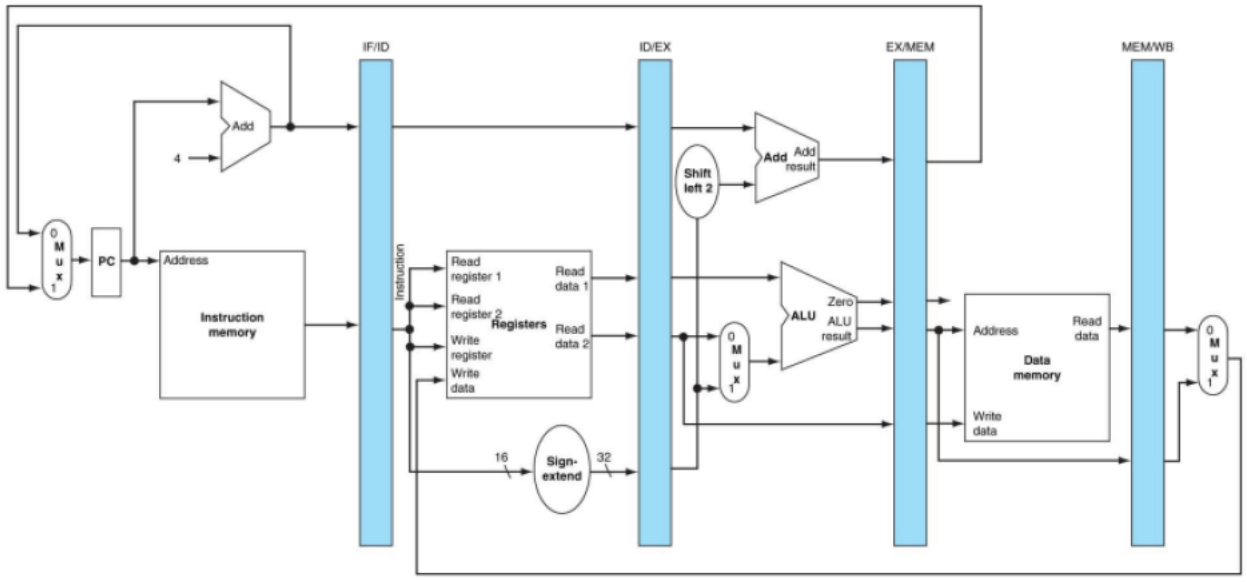
Tipo di istruzioni	Istruzioni	Esempio	Significato	Commenti
Aritmetiche	Somma	`add $s1,$s2,$s3`	`$s1 = $s2 + $s3`	Operandi in tre registri
	Sottrazione	`sub $s1,$s2,$s3`	`$s1 = $s2 – $s3`	Operandi in tre registri
	Somma immediata	`addi $s1,$s2,20`	`$s1 = $s2 + 20`	Utilizzata per sommare delle costanti
Trasferimento dati	Lettura parola	`lw $s1,20($s2)`	`$s1`=Memoria [`$s2`+20]	Trasferimento di una parola da memoria a registro
	Memorizzazione parola	`sw $s1,20($s2)`	Memoria[`$s2`+20]= `$s1`	Trasferimento di una parola da registro a memoria
	Lettura mezza parola	`lh $s1,20($s2)`	`$s1`=Memoria[`$s2`+20]	Trasferimento di una mezza parola da memoria a registro
	Lettura mezza parola, senza segno	`lhu $s1,20($s2)`	`$s1`=Memoria[`$s2`+20]	Trasferimento di una mezza parola da memoria a registro
	Memorizzazione mezza parola	`sh $s1,20($s2)`	Memoria[`$s2`+20]= `$s1`	Trasferimento di una mezza parola da registro a memoria
	Lettura byte	`lb $s1,20($s2)`	`$s1`=Memoria[`$s2`+20]	Trasferimento di un byte da memoria a registro
	Lettura byte senza segno	`lbu $s1,20($s2)`	`$s1`=Memoria[`$s2`+20]	Trasferimento di un byte da memoria a registro
	Memorizzazione byte	`sb $s1,20($s2)`	Memoria[`$s2`+20]= `$s1`	Trasferimento di un byte da registro a memoria
	Lettura di una parola e blocco	`ll $s1,20($s2)`	`$s1`=Memoria[`$s2`+20]	Caricamento di una parola come prima fase di un'operazione atomica
	Memorizzazione condizionata di una parola	`sc $s1,20($s2)`	Memoria[`$s2`+20]= `$s1`; `$s1`=0 oppure 1	Memorizzazione di una parola come seconda fase di un'operazione atomica
	Caricamento costante nella mezza parola superiore	`lui $s1,20`	`$s1` = 20 * 2^{16}	Caricamento di una costante nei 16 bit più significativi

Tipo di istruzioni	Istruzioni	Esempio	Significato	Commenti
Logiche	And	`and $s1,$s2,$s3`	`$s1 = $s2 & $s3`	Operandi in tre registri; AND bit a bit
	Or	`or $s1,$s2,$s3`	`$s1 = $s2 | $s3`	Operandi in tre registri; OR bit a bit
	Nor	`nor $s1,$s2,$s3`	`$s1 = ~ ($s2 | $s3)`	Operandi in tre registri; NOR bit a bit
	And immediato	`andi $s1,$s2,20`	`$s1 = $s2 & 20`	AND bit a bit tra un operando in registro e una costante
	Or immediato	`ori $s1,$s2,20`	`$s1 = $s2 | 20`	OR bit a bit tra un operando in registro e una costante
	Scorrimento logico a sinistra	`sll $s1,$s2,10`	`$s1 = $s2 << 10`	Spostamento a sinistra del numero di bit specificato dalla costante
	Scorrimento logico a destra	`srl $s1,$s2,10`	`$s1 = $s2 >> 10`	Spostamento a destra del numero di bit specificato dalla costante
Salti condizionati	Salta se uguale	`beq $s1,$s2,25`	Se (`$s1==$s2`) vai a PC+4+100	Test di uguaglianza; salto relativo al PC
	Salta se non è uguale	`bne $s1,$s2,25`	Se (`$s1!=$s2`) vai a PC+4+100	Test di disuguaglianza; salto relativo al PC
	Poni uguale a 1 se minore	`slt $s1,$s2,$s3`	Se (`$s2 < $s3`) `$s1 = 1`; altrimenti `$s1 = 0`	Comparazione di minoranza; utilizzata con bne e beq
	Poni uguale a uno se minore, numeri senza segno	`sltu $s1,$s2,$s3`	Se (`$s2 < $s3`) `$s1 = 1`; altrimenti `$s1 = 0`	Comparazione di minoranza su numeri senza segno
	Poni uguale a uno se minore, immediato	`slti $s1,$s2,20`	Se (`$s2 < 20`) `$s1 = 1`; altrimenti `$s1 = 0`	Comparazione di minoranza con una costante
	Poni uguale a uno se minore, immediato e senza segno	`sltiu $s1,$s2,20`	Se (`$s2 < 20`) `$s1 = 1`; altrimenti `$s1 = 0`	Comparazione di minoranza con una costante, con numeri senza segno
Salti incondizionati	Salto incondizionato	`j 2500`	Vai a 10 000	Salto all'indirizzo della costante
	Salto indiretto	`jr $ra`	Vai all'indirizzo contenuto in `$ra`	Salto all'indirizzo contenuto nel registro, utilizzato per il ritorno da procedura e per i costrutti switch
	Salta e collega	`jal 2500`	`$ra` = PC+4; vai a 10 000	Chiamata a procedura

MODELLO DI PROGRAMMAZIONE MIPS

Nome	Numero	Utilizzo	Preservato durante le chiamate
$zero	0	costante zero	*Riservato MIPS*
$at	1	riservato per l'assemblatore	*Riservato Compiler*
$v0-$v1	2-3	valori di ritorno di una procedura	No
$a0-$a3	4-7	argomenti di una procedura	No
$t0-$t7	8-15	registri temporanei (non salvati)	No
$s0-$s7	16-23	registri salvati	Si
$t8-$t9	24-25	registri temporanei (non salvati)	No
$k0-$k1	26-27	gestione delle eccezioni	*Riservato OS*
$gp	28	puntatore alla global area (dati)	Si
$sp	29	stack pointer	Si
$s8	30	registro salvato (fp)	Si
$ra	31	indirizzo di ritorno	No

PIPELINE DEL MIPS

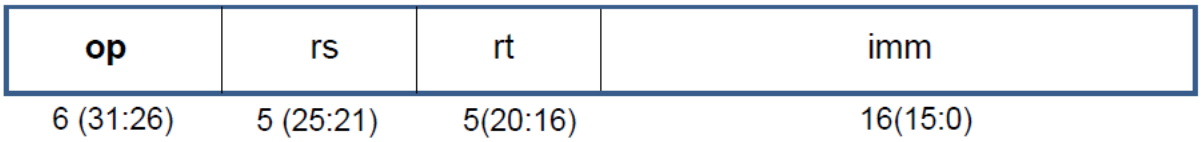


FORMATO ISTRUZIONI MIPS

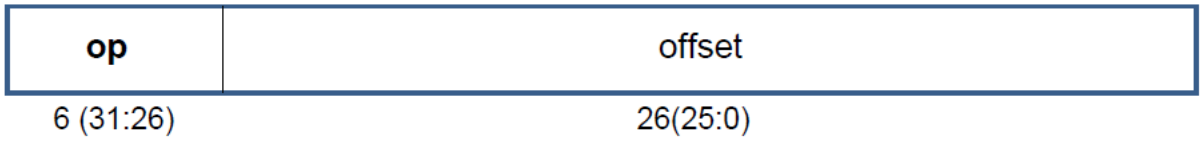
Istruzioni di tipo R

op	rs	rt	rd	*shamt*	*funct*
6 (31:26)	5 (25:21)	5(20:16)	5(15:11)	5(10:6)	6(5:0)

Istruzioni di tipo I



Istruzioni di tipo J



Syscall MIPS

Service	Code in $v0	Arguments	Results
print_int	1	$a0 = integer to be printed	
print_float	2	$f12 = float to be printed	
print_double	3	$f12 = double to be printed	
print_string	4	$a0 = address of string in memory	
read_int	5		integer returned in $v0
read_float	6		float returned in $v0
read_double	7		double returned in $v0
read_string	8	$a0 = memory address of string input buffer $a1 = length of string buffer (n)	
sbrk	9	$a0 = amount	address in $v0
exit	10		

Codici operativi 68000

Trasferimento dati [11 codici]
{EXG, LEA, LINK, MOVE, MOVEA, MOVEM, MOVEP, MOVEQ, PEA, SWAP, UNLK}
Es. MOVE <ea>,<ea>

Arithmetica di interi [18 codici]
{ADD, ADDA, ADDQ, ADDI, ADDX, CLR, DIVS, DIVU, EXT, MULS, MULU, NEG, NEGX, SUB, SUBA, SUBI, SUBQ, SUBX}
Es. ADD <ea>,Dn
ADD Dn,<ea>

Test/Compare [6 codici]
{TAS, TST, CMP, CMPA, CMPM, CMPI}
Es. CMP <ea>,Dn
CMPA <ea>,An

Aritmetica di decimali in BCD [3 codici]
{ABCD, SBCD, NBCD}

Logiche [7 codici]
{AND, ANDI, OR, ORI, EOR, EORI, NOT}
Es. AND <ea>,Dn
AND Dn,<ea>
ANDI #<dato>,<ea>

Shift/Rotate [8 codici]
{ASL, ASR, LSL, LSR, ROL, ROR, ROXL, ROXR}
Es. LSL Dx,Dy
LSL #<dato>,Dy
LSl <ea>

Controllo programma [9 codici]
{Bcc, DBcc, Scc, BRA, BSR, JMP, JSR, RTR, RTS}
Es. Bcc LABEL

Operazioni su bit [4 codici]
{BTST, BSET, BCLR, BCHG}
Es. BTST Dn,<ea>
BTST #<dato>,<ea>

Operazioni di controllo sistema [17 codici]
{RESET, RTE, STOP, ORI to SR, MOVE USP, ANDI to SR, EORI to SR, MOVE EA to SR, TRAP, TRAPV, CHK, ANDI to CCR, EORI to CCR, MOVE EA to CCR, ORI to CCR, MOVE SR to EA, MOVE EA to USP}

MODELLO DI PROGRAMMAZIONE 68000

Denominazione	Parallelismo
PC	*32 bit*
D0-D7	*32 bit*
A0-A7	*32 bit*
SP=A7	*32 bit*
SR	*16 bit*
CCR	*8 bit*