Instruction Set Collection

OpType	Name	Type	Syntax	Binary	Remark
Arithmetic	Add	R	`add $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100000`	
	Add unsigned	R	`addu $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100001`	
	Subtract	R	`sub $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100010`	
	Subtract unsigned	R	`subu $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100011`	
	Add immediate	I	`addi $t,$s,C`	`001000 sssss ttttt CCCCC CCCCC CCCCCC`	
	Add immediate unsigned	I	`addiu $t,$s,C`	`001001 sssss ttttt CCCCC CCCCC CCCCCC`	
	Multiply	R	`mult $s,$t`	`000000 sssss ttttt ddddd 00000 011000`	`(HI,LO) = (64-bit) $s * $t`
	Multiply unsigned	R	`multu $s,$t`	`000000 sssss ttttt ddddd 00000 011001`	`(HI,LO) = (64-bit) $s * $t`
	Divide	R	`div $s,$t`	`000000 sssss ttttt ddddd 00000 011010`	`LO = $s / $t, HI = $s % $t`
	Divide unsigned	R	`divu $s,$t`	`000000 sssss ttttt ddddd 00000 011011`	`LO = $s / $t, HI = $s % $t`
Logical	And	R	`and $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100100`	
	And immediate	I	`andi $t,$s,C`	`001100 sssss ttttt CCCCC CCCCC CCCCCC`	
	Or	R	`or $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100101`	
	Or immediate	I	`ori $t,$s,C`	`001101 sssss ttttt CCCCC CCCCC CCCCCC`	
	Exclusive or	R	`xor $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100110`	
	Nor	R	`nor $d,$s,$t`	`000000 sssss ttttt ddddd 00000 100111`	
	Set on less than	R	`slt $d,$s,$t`	`000000 sssss ttttt ddddd 00000 101010`	
	Set on less than unsigned	R	`sltu $d,$s,$t`	`000000 sssss ttttt ddddd 00000 101011`	
	Set on less than immediate	I	`slti $t,$s,C`	`001010 sssss ttttt CCCCC CCCCC CCCCCC`	
Bitwise Shift	Shift left logical immediate	R	`sll $d,$t,shamt`	`000000 sssss ttttt ddddd 00000 000000`	`$d = $t << shamt`
	Shift right logical immediate	R	`srl $d,$t,shamt`	`000000 sssss ttttt ddddd 00000 000010`	`$d = {16'b0, $t >> shamt}`
	Shift right arithmetic immediate	R	`sra $d,$t,shamt`	`000000 sssss ttttt ddddd 00000 000011`	`$d = {{16{t[31]}}, $t >> shamt}`
	Shift left logical	R	`sllv $d,$t,$s`	`000000 sssss ttttt ddddd 00000 000100`	`$d = $t << $s`
	Shift right logical	R	`srlv $d,$t,$s`	`000000 sssss ttttt ddddd 00000 000110`	`$d = {16'b0, $t >> $s}`
	Shift right arithmetic	R	`srav $d,$t,$s`	`000000 sssss ttttt ddddd 00000 000111`	`$d = {{16{t[31]}}, $t >> $s}`
Data Transfer	Load word	I	`lw $t,C($s)`	`100011 sssss ttttt CCCCC CCCCC CCCCCC`	
	Load halfword	I	`lh $t,C($s)`	`100001 sssss ttttt CCCCC CCCCC CCCCCC`	
	Load halfword unsigned	I	`lhu $t,C($s)`	`100101 sssss ttttt CCCCC CCCCC CCCCCC`	
	Load byte	I	`lb $t,C($s)`	`100000 sssss ttttt CCCCC CCCCC CCCCCC`	
	Load byte unsigned	I	`lbu $t,C($s)`	`100100 sssss ttttt CCCCC CCCCC CCCCCC`	
	Store word	I	`sw $t,C($s)`	`101011 sssss ttttt CCCCC CCCCC CCCCCC`	
	Store half	I	`sh $t,C($s)`	`101001 sssss ttttt CCCCC CCCCC CCCCCC`	
	Store byte	I	`sb $t,C($s)`	`101000 sssss ttttt CCCCC CCCCC CCCCCC`	
	Load upper immediate	I	`lui $t,C`	`001111 00000 ttttt CCCCC CCCCC CCCCCC`	
	Move from high	R	`mfhi $d`	`000000 00000 00000 ddddd 00000 010000`	
	Move from low	R	`mflo $d`	`000000 00000 00000 ddddd 00000 010010`	
Conditional branch	Branch on equal	I	`beq $s,$t,C`	`000100 sssss ttttt CCCCC CCCCC CCCCCC`	
	Branch on not equal	I	`bne $s,$t,C`	`000101 sssss ttttt CCCCC CCCCC CCCCCC`	
Unconditional jump	Jump	J	`j C`	`000010 CCCCC CCCCC CCCCC CCCCC CCCCCC`	`addr = {PC_plus_4[31:28], C << 2}`
	Jump register	R	`jr $s`	`000000 sssss 00000 00000 00000 001000`	
	Jump and link	J	`jal C`	`000011 CCCCC CCCCC CCCCC CCCCC CCCCCC`	`addr = {PC_plus_4[31:28], C << 2}`