

MIPS reference card

add	rd, rs, rt	Add	rd = rs + rt	R 0 / 20	registers											
sub	rd, rs, rt	Subtract	rd = rs - rt	R 0 / 22	\$0 \$zero											
addi	rt, rs, imm	Add Imm.	rt = rs + imm±	I 8	\$1 \$at											
addu	rd, rs, rt	Add Unsigned	rd = rs + rt	R 0 / 21	\$2-\$3 \$v0-\$v1											
subu	rd, rs, rt	Subtract Unsigned	rd = rs - rt	R 0 / 23	\$4-\$7 \$a0-\$a3											
addiu	rt, rs, imm	Add Imm. Unsigned	rt = rs + imm±	I 9	\$8-\$15 \$t0-\$t7											
mult	rs, rt	Multiply	{hi, lo} = rs * rt	R 0 / 18	\$16-\$23 \$s0-\$s7											
div	rs, rt	Divide	lo = rs / rt; hi = rs % rt	R 0 / 1a	\$24-\$25 \$t8-\$t9											
multu	rs, rt	Multiply Unsigned	{hi, lo} = rs * rt	R 0 / 19	\$26-\$27 \$k0-\$k1											
divu	rs, rt	Divide Unsigned	lo = rs / rt; hi = rs % rt	R 0 / 1b	\$28 \$gp											
mfhi	rd	Move From Hi	rd = hi	R 0 / 10	\$29 \$sp											
mflo	rd	Move From Lo	rd = lo	R 0 / 12	\$30 \$fp											
and	rd, rs, rt	And	rd = rs & rt	R 0 / 24	\$31 \$ra											
or	rd, rs, rt	Or	rd = rs rt	R 0 / 25	hi —											
nor	rd, rs, rt	Nor	rd = ~(rs rt)	R 0 / 27	lo —											
xor	rd, rs, rt	eXclusive Or	rd = rs ^ rt	R 0 / 26	PC —											
andi	rt, rs, imm	And Imm.	rt = rs & imm0	I c	c0 \$13 c0_cause											
ori	rt, rs, imm	Or Imm.	rt = rs imm0	I d	c0 \$14 c0_epc											
xori	rt, rs, imm	eXclusive Or Imm.	rt = rs ^ imm0	I e												
sll	rd, rt, sh	Shift Left Logical	rd = rt << sh	R 0 / 0	syscall codes											
srl	rd, rt, sh	Shift Right Logical	rd = rt >>> sh	R 0 / 2	for MARS/SPIM											
sra	rd, rt, sh	Shift Right Arithmetic	rd = rt >> sh	R 0 / 3	1 print integer											
sllv	rd, rt, rs	Shift Left Logical Variable	rd = rt << rs	R 0 / 4	2 print float											
srlv	rd, rt, rs	Shift Right Logical Variable	rd = rt >>> rs	R 0 / 6	3 print double											
srav	rd, rt, rs	Shift Right Arithmetic Variable	rd = rt >> rs	R 0 / 7	4 print string											
slt	rd, rs, rt	Set if Less Than	rd = rs < rt ? 1 : 0	R 0 / 2a	5 read integer											
sltu	rd, rs, rt	Set if Less Than Unsigned	rd = rs < rt ? 1 : 0	R 0 / 2b	6 read float											
slti	rt, rs, imm	Set if Less Than Imm.	rt = rs < imm±? 1 : 0	I a	7 read double											
sltiu	rt, rs, imm	Set if Less Than Imm. Unsigned	rt = rs < imm±? 1 : 0	I b	8 read string											
j	addr	Jump	PC = PC&0xF0000000 (addr0<< 2)	J 2	9 sbrk/alloc. mem.											
jal	addr	Jump And Link	\$ra = PC + 8; PC = PC&0xF0000000 (addr0<< 2)	J 3	10 exit											
jr	rs	Jump Register	PC = rs	R 0 / 8	11 print character											
jalr	rs	Jump And Link Register	\$ra = PC + 8; PC = rs	R 0 / 9	12 read character											
beq	rt, rs, imm	Branch if Equal	if (rs == rt) PC += 4 + (imm±<< 2)	I 4	13 open file											
bne	rt, rs, imm	Branch if Not Equal	if (rs != rt) PC += 4 + (imm±<< 2)	I 5	14 read file											
syscall		System Call	c0_cause = 8 << 2; c0_epc = PC; PC = 0x80000080	R 0 / c	15 write to file											
lui	rt, imm	Load Upper Imm.	rt = imm << 16	I f	16 close file											
lb	rt, imm(rs)	Load Byte	rt = SignExt(M1[rs + imm±])	I 20												
lbu	rt, imm(rs)	Load Byte Unsigned	rt = M1[rs + imm±] & 0xFF	I 24	exception causes											
lh	rt, imm(rs)	Load Half	rt = SignExt(M2[rs + imm±])	I 21	0 interrupt											
lhu	rt, imm(rs)	Load Half Unsigned	rt = M2[rs + imm±] & 0xFFFF	I 25	1 TLB protection											
lw	rt, imm(rs)	Load Word	rt = M4[rs + imm±]	I 23	2 TLB miss L/F											
sb	rt, imm(rs)	Store Byte	M1[rs + imm±] = rt	I 28	3 TLB miss S											
sh	rt, imm(rs)	Store Half	M2[rs + imm±] = rt	I 29	4 bad address L/F											
sw	rt, imm(rs)	Store Word	M4[rs + imm±] = rt	I 2b	5 bad address S											
ll	rt, imm(rs)	Load Linked	rt = M4[rs + imm±]	I 30	6 bus error F											
sc	rt, imm(rs)	Store Conditional	M4[rs + imm±] = rt; rt = atomic ? 1 : 0	I 38	7 bus error L/S											
pseudo-instructions					8 syscall											
bge	rx, ry, imm	Branch if Greater or Equal	R <table><tr><td>6 bits</td><td>5 bits</td><td>5 bits</td><td>5 bits</td><td>5 bits</td><td>6 bits</td></tr><tr><td>op</td><td>rs</td><td>rt</td><td>rd</td><td>sh</td><td>func</td></tr></table>	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	op	rs	rt	rd	sh	func	9 break
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits											
op	rs	rt	rd	sh	func											
bgt	rx, ry, imm	Branch if Greater Than				a reserved instr.										
ble	rx, ry, imm	Branch if Less or Equal	I <table><tr><td>6 bits</td><td>5 bits</td><td>5 bits</td><td>16 bits</td></tr><tr><td>op</td><td>rs</td><td>rt</td><td>imm</td></tr></table>	6 bits	5 bits	5 bits	16 bits	op	rs	rt	imm			b coproc. unusable		
6 bits	5 bits	5 bits	16 bits													
op	rs	rt	imm													
blt	rx, ry, imm	Branch if Less Than				c arith. overflow										
la	rx, label	Load Address	J <table><tr><td>6 bits</td><td>26 bits</td></tr><tr><td>op</td><td>addr</td></tr></table>	6 bits	26 bits	op	addr			F: fetch instr.						
6 bits	26 bits															
op	addr															
li	rx, imm	Load Immediate				L: load data										
move	rx, ry	Move register				S: store data										
nop		No Operation														