Instruction	Opcode (ALU Op)	Type	Usage	Operation
add	00000 (00000)	R	add $rd, $rs, $rt	$rd = $rs + $rt
sub	00000 (00001)	R	sub$rd, $rs, $rt	$rd = $rs - $rt
and	00000 (00010)	R	and $rd, $rs, $rt	$rd = $rs AND $rt
or	00000 (00011)	R	or $rd, $rs, $rt	$rd = $rs OR $rt
sll	00000 (00100)	R	sll $rd, $rs, shamt	$rd = $rs << shamt
sra	00000 (00101)	R	sra $rd, $rs, shamt	$rd = $rs * 2^shamt
mul	00000 (00110)	R	mul $rd, $rs, $rt	$rd = $rs * $rt (16b X 16b)
div	00000 (00111)	R	div $rd, $rs, $rt	$rd = $rs / $rt (32b div. by 16b)
CUSTOM R (in ALU)	00000 (01000) ... 00000 (11111)	R	CUSTOM_R# $rd, $rs, $rt	$rd = CUSTOM#($rs, $rt)t
j	00001	JI	j N	PC = N
bne	00010	I	bne $rd, $rs, N	if($rd != $rs) PC = PC+1+N
jal	00011	JI	jal N	$r31 = PC+1; PC=N
jr	00100	JII	jr $rd	PC = $rd
addi	00101	I	addi $rd, $rs, N	$rd = $rs + N
blt	00110	I	blt $rd, $rs, N	if($rd < $rs) PC=PC+1+N
sw	00111	I	sw $rd, N($rs)	MEM[$rs + N] = $rd
lw	01000	I	lw $rd, N($rs)	$rd = MEM[$rs + N]
custr1	01001	R	CUSTOM $rd, $rs, $rt	Custom
custr2	01010	R		
custr3	01011	R		
custr4	01100	R		
custr5	01101	R		
custr6	01110	R		
custr7	01111	R		
custr8	10000	R		
custi1	10001	I	CUSTOM $rd, $rs, N	Custom
custi2	10010	I		
custi3	10011	I		
custi4	10100	I		
custj1	10101	JI	CUSTOM N	Custom
custj2	10110	JI		

Instruction Type	Instruction Format						
R	Opcode [31:27]	RD [26:22]	RS [21:17]	RT [16:12]	shiftamt [11:7]	ALUop [6:2]	Zeros [1:0]
I	Opcode [31:27]	RD [26:22]	RS [21:17]	Immediate [16:0]			
J (JII)	Opcode [31:27]	Target [26:0]					
	Opcode [31:27]	RD [26:22]					

I-type immediate field [16:0] is signed 2's complement and sign-extended to the full 32-bit word size.

J-type target field [26:0] is extended to the full 32-bit PC size using the upper bits from the current PC+1.

Register fields that are undefined are filled with zeroes by the assembler.

Register $r0 always equals zero. Registers $r1 through $r30 are general purpose. Register $r31 stores the link address of a jump-and-link instruction.

Instructions that change control flow (beq, blt, j, jal, jr) do not have a delay slot.

Memory is word-addressed. The instruction and data memory address spaces are separate. Static data begins at data memory address zero. Stack data begins at the end of the data memory and grows downwards. There is no preset boundary between the end of static data and the start of the upwardsgrowing heap; this is a property of the assembly program.

After a reset, all register values are zero and program execution begins from instruction memory address zero. The memory's contents are not reset.