Instruction Format	Opcode	Mnemonic	Semantics	Examples
Arithmetic. The twelve data bits in an instruction contain three four-bit register numbers, which are referred to here as *ra*, *rb*, and *rc*.	0000	add	Reg[*ra*] = Reg[*rb*] + Reg[*rc*].	add $1 $2 $3
	0001	sub	Reg[*ra*] = Reg[*rb*] - Reg[*rc*]	sub $1 $1 $8
	0010	mul	Reg[*ra*] = Reg[*rb*] * Reg[*rc*]	mul $2 $7 $7
	0011	div	Reg[*ra*] = Reg[*rb*] / Reg[*rc*], but if Reg[*rc*] is zero, the computer halts because of a division by zero error.	div $1 $15 $13
	0100	sll	Reg[*ra*] = Reg[*rb*] << (Reg[*rc*] & 15); shift left logical (with zero fill)	sll $1 $2 $3
	0101	srl	Reg[*ra*] = Reg[*rb*] >>> (Reg[*rc*] & 15); shift right logical (with zero fill)	srl $3 $2 $1
	0110	nor	Reg[*ra*] = ~(Reg[*rb*] | Reg[*rc*]); bitwise logical NOR operation.	nor $1 $2 $3
	0111	slt	Reg[*ra*] = (Reg[*rb*] < Reg[*rc*]) ? 1 : 0; set if less than	slt $1 $2 $3
Immediate. The first four data bits, *ra*, represent a register number. The last eight data bits, shown here as *limm*, represent a signed 8-bit number. "limm" stands for "long immediate," and an "immediate" is a field in an instruction that represents a constant rather than a register number.	1000	li	Reg[*ra*] = sext(*limm*); load immediate	li $1 0 li $3 0xA7
	1001	lui	Reg[*ra*] = *limm* << 8; load upper immediate	lui $1 42 lui $8 -3
	1010	beqz	if (Reg[*ra*] == 0) PC = PC + sext(*limm*); branch if equal to zero	beqz $1 5 beqz $0 -19
	1011	bnez	if (Reg[*ra*] != 0) PC = PC + sext(*limm*); branch if not equal to zero	bnez $14 87
Memory. Data bits are two 4-bit register numbers, *ra* and *rb*, and and a signed 4-bit number, *simm*. "simm" stands for "short immediate."	1100	lw	Load value from memory location Reg[*rb*]+sext(*simm*) into Reg[*ra*]	lw $1 $2 lw $1 $2 3
	1101	sw	Store value from Reg[*ra*] into memory location Reg[*rb*]+sext(*simm*)	sw $1 $2 sw $1 $2 -5
Jump. Two 4-bit fields, *ra* and *rb*; last four data bits are ignored.	1110	jalr	Jump-and-link-register: Save current PC in Reg[*ra*] and set PC to Reg[*rb*].	jalr $11 $1 jalr $0 $2
Syscall. All data bits are ignored.	1111	syscall	Call system subroutine number Reg[1].	syscall