Name	Format	Functionality	Specialized signal	
		R-type		
add	add $r1, $r2, $r3	add two register numbers ($r2 and $r3) and store the result in one register ($r1)	Write memory	
and	and $r1, $r2, $r3	bit-wise logical AND of two register numbers ($r2 and $r3), and store the result in a register ($r1)	Write memory	
or	or $r1, $r2, $r3	bit-wise logical OR of two register numbers ($r2 and $r3), and store the result in a register ($r1)	Write memory	
slt	slt $r1, $r2, $r3	if the register number ($r2) is less than the register number ($r3), then $r1=1; or $r1=0.	Less than	
lsl	lsl $r1, $r2, $r3	logical-shift a register number ($r2) to left for several digits ($r3), and store the result in a register ($r1)		
lsr	lsr $r1, $r2, $r3	logical-shift a register number ($r2) to right for several digits ($r3), and store the result in a register ($r1). The shift-in high bits are all 0.		
neg	neg $r1, $r2	negate a register number ($r2) and store the result in the register ($r1)		
		I-type		
addi	addi $r1, $r2, x	add a register number ($r2) and an immediate number (x), and store the result in one register ($r1)	immediate	

andi	andi $r1, $r2, x	bit-wise logical AND of a register number ($r2) and an immediate number x, and store the result in a register ($r1)	immediate	
ori	ori $r1, $r2, x	bit-wise logical OR of a register number ($r2) and an immediate number x, and store the result in a register ($r1)	immediate	
ble	ble $r1, $r2, x	if the first register number ($r1) is smaller than or equal to the second register number ($r2), then jump to address **(PC + x)**.	Less than or equal to	
li	li $r1, x	load immediate number to one register ($r1)		
		J-type		
load	load $r1, $r2	load a 16-bit number from data memory to a register ($r1). The data memory address is the value in a register ($r2)		
store	store $r1, $r2	load a 16-bit number from a register ($r1) to data memory. The data memory address is the value in a register ($r2)		
move	move $r1, $r2	copy the value of one register ($r2) to another register ($r1)		
jump	jump x	Jump to address (PC + x).	immediate	
call	call x	Jump to address (PC + x), where x is an immediate number, and at the same time save the address of the next instruction to a special register $ra ($ra is not one of the 8 general registers $r1 -- r8).	immediate	

rtn	rtn	Jump to address stored in $ra.		
reboot	reboot	Directly jump to address 0		
halt	halt	all registers (including PC, the 8 general purpose registers and all other registers) in the processor are disabled (so the processor halts).		