add t1,t2,t3	Addition: set t1 to (t2 plus t3)
addi t1,t2,-100	Addition immediate: set t1 to (t2 plus signed 12-bit immediate)
and t1,t2,t3	Bitwise AND : Set t1 to bitwise AND of t2 and t3
andi t1,t2,-100	Bitwise AND immediate : Set t1 to bitwise AND of t2 and sign-extended 12-bit immediate
auipc t1,100000	Add upper immediate to pc: set t1 to (pc plus an upper 20-bit immediate)
beq t1,t2,label	Branch if equal : Branch to statement at label's address if t1 and t2 are equal
bge t1,t2,label	Branch if greater than or equal: Branch to statement at label's address if t1 is greater than or equal to t2
bgeu t1,t2,label	Branch if greater than or equal to (unsigned): Branch to statement at label's address if t1 is greater than or equal to t2 (with an unsigned interpretation)
blt t1,t2,label	Branch if less than: Branch to statement at label's address if t1 is less than t2
bltu t1,t2,label	Branch if less than (unsigned): Branch to statement at label's address if t1 is less than t2 (with an unsigned interpretation)
bne t1,t2,label	Branch if not equal : Branch to statement at label's address if t1 and t2 are not equal
csrrc t0, fcsr, t1	Atomic Read/Clear CSR: read from the CSR into t0 and clear bits of the CSR according to t1
csrrci t0, fcsr, 10	Atomic Read/Clear CSR Immediate: read from the CSR into t0 and clear bits of the CSR according to a constant
csrrs t0, fcsr, t1	Atomic Read/Set CSR: read from the CSR into t0 and logical or t1 into the CSR
csrrsi t0, fcsr, 10	Atomic Read/Set CSR Immediate: read from the CSR into t0 and logical or a constant into the CSR
csrrw t0, fcsr, t1	Atomic Read/Write CSR: read from the CSR into t0 and write t1 into the CSR
csrrwi t0, fcsr, 10	Atomic Read/Write CSR Immediate: read from the CSR into t0 and write a constant into the CSR
jal t1, target	Jump and link : Set t1 to Program Counter (return address) then jump to statement at target address
jalr t1, t2, -100	Jump and link register: Set t1 to Program Counter (return address) then jump to statement at t2 + immediate
lb t1, -100(t2)	Set t1 to sign-extended 8-bit value from effective memory byte address
lbu t1, -100(t2)	Set t1 to zero-extended 8-bit value from effective memory byte address
lh t1, -100(t2)	Set t1 to sign-extended 16-bit value from effective memory halfword address
lhu t1, -100(t2)	Set t1 to zero-extended 16-bit value from effective memory halfword address
lui t1,100000	Load upper immediate: set t1 to 20-bit followed by 12 0s
lw t1, -100(t2)	Set t1 to contents of effective memory word address
mul t1,t2,t3	Multiplication: set t1 to the lower 32 bits of t2*t3
sb t1, -100(t2)	Store byte : Store the low-order 8 bits of t1 into the effective memory byte address
sh t1, -100(t2)	Store halfword : Store the low-order 16 bits of t1 into the effective memory halfword address
sll t1,t2,t3	Shift left logical: Set t1 to result of shifting t2 left by number of bits specified by value in low-order 5 bits of t3
slli t1,t2,10	Shift left logical : Set t1 to result of shifting t2 left by number of bits specified by immediate
slt t1,t2,t3	Set less than : If t2 is less than t3, then set t1 to 1 else set t1 to 0
slti t1,t2,-100	Set less than immediate : If t2 is less than sign-extended 12-bit immediate, then set t1 to 1 else set t1 to 0
sltiu t1,t2,-100	Set less than immediate unsigned : If t2 is less than sign-extended 16-bit immediate using unsigned comparison, then set t1 to 1 else set t1 to 0
sltu t1,t2,t3	Set less than : If t2 is less than t3 using unsigned comparision, then set t1 to 1 else set t1 to 0
sra t1,t2,t3	Shift right arithmetic: Set t1 to result of sign-extended shifting t2 right by number of bits specified by value in low-order 5 bits of t3
srai t1,t2,10	Shift right arithmetic : Set t1 to result of sign-extended shifting t2 right by number of bits specified by immediate
srl t1,t2,t3	Shift right logical: Set t1 to result of shifting t2 right by number of bits specified by value in low-order 5 bits of t3
srli t1,t2,10	Shift right logical : Set t1 to result of shifting t2 right by number of bits specified by immediate
sub t1,t2,t3	Subtraction: set t1 to (t2 minus t3)
addi t1,t2,%lo(label)	Load Lower Address : Set t1 to t2 + lower 12-bit label's address
b label	Branch : Branch to statement at label unconditionally
beqz t1,label	Branch if EQual Zero : Branch to statement at label if t1 == 0
bgez t1,label	Branch if Greater than or Equal to Zero : Branch to statement at label if t1 >= 0
bgt t1,t2,label	Branch if Greater Than : Branch to statement at label if t1 > t2
bgtu t1,t2,label	Branch if Greater Than Unsigned: Branch to statement at label if t1 > t2 (unsigned compare)
bgtz t1,label	Branch if Greater Than Zero: Branch to statement at label if t1 > 0
ble t1,t2,label	Branch if Less or Equal : Branch to statement at label if t1 <= t2
bleu t1,t2,label	Branch if Less or Equal Unsigned : Branch to statement at label if t1 <= t2 (unsigned compare)
blez t1,label	Branch if Less than or Equal to Zero : Branch to statement at label if t1 <= 0
bltz t1,label	Branch if Less Than Zero : Branch to statement at label if t1 < 0

```
li t1,10000000          Load Immediate : Set t1 to 32-bit immediate
lui t1,%hi(label)       Load Upper Address : Set t1 to upper 20-bit label's address
lw t1,%lo(label)(t2)    Load from Address
lw t1,(t2)              Load Word : Set t1 to contents of effective memory word address
lw t1,-100              Load Word : Set t1 to contents of effective memory word address
lw t1,10000000          Load Word : Set t1 to contents of effective memory word address
lw t1,label            Load Word : Set t1 to contents of memory word at label's address
```

```
sb t1,(t2)              Store Byte : Store the low-order 8 bits of t1 into the effective memory byte address
sb t1,-100              Store Byte : Store the low-order 8 bits of $t1 into the effective memory byte address
sb t1,10000000,t2       Store Byte : Store the low-order 8 bits of $t1 into the effective memory byte address
sb t1,label,t2          Store Byte : Store the low-order 8 bits of $t1 into the effective memory byte address
```