nop

Opcode: 0
Asm: nop
This instruction does nothing. “Stalls” the processor for one clock cycle.

rets

Opcode: 1
Asm: rets
Swaps the register set and program counter used from the system (aka kernel) set, to the user set, or vice versa. Execution resumes at the instruction after rets in the program in each mode. The return address and PC are restored to what they were before (these elements are stored in a register).

move

Opcode: 6
Asm: move rd, rs/imm8
Move the value in a register or immediate 8 into a different register

loadc

Opcode: 7
Asm: loadc rd, rs/imm8
rd ← pmem[pc + imm8]
rd ← pmem[rs]
Loads a constant from the program memory

jump

Opcode: 8
Asm: jump rd/imm11
pc ← rd
pc ← pc + imm11
Sets the program counter to move locations within the program instructions

call

Opcode: 10
Asm: call rd/imm12
r0 ← pc, pc ← + imm11
r0 ← pc, pc ← rd

Load

Opcode: 12
Asm: load rd, [rs, imm4]
rd ← mem[rs + imm4*2]
Loads a 2 byte word into rd. Memory is aligned, so that's why we multiply imm4 by 2

Store

Opcode: 13
Asm: store [rs, imm4], rd
mem[rs + imm4 * 2] ← rd
Stores a 2 byte word into memory. Memory is aligned so that's why we multiply imm4 by 2

Loadb

Opcode: 14
Asm: loadb rd, [rs, imm4]
rd ← mem[rs + imm4]
Load a byte from memory

Storeb

Opcode: 15
Asm: store [rs, im4], rd
memb[rs + imm4] ← rd
Store a byte into memory

Add

Opcode: 16
Asm: add rd, rs/imm6
rd ← rd + rsval

Sub

Opcode: 17
Asm: sub rd, rs/imm6
rd ← rd + rsval

XOR

Opcode: 20
Asm: xor rd, rs/imm6
rd ← rd xor rsval

AND

Opcode: 21
Asm: and rd, rs/imm6
rd ← rd and rsval

OR

Opcode: 22
Asm: or rd, rs/imm6
rd ← rd or rsval

SHL

Opcode: 23
Asm: shl rd, rs/imm6
rd ← rd << val
Logical shift left

SHR

Opcode: 24
Asm: shr rd, rs/imm6
Rd ← rd >> rsval
Logical shift right

ASR

Opcode: 25
Asm: asr rd, rs/imm6
rd ← rd >> rsval
Arithmetic shift right (signed, two's complement)

if.eq

Opcode: 26
Asm: if.eq rd, rs/imm6
S ← !(rd == rsval)
Checks if the values are equal. S is a "skip flag." If the condition is evaluated to false, the next instruction will be skipped

if.ne

Opcode: 27
Asm: if.ne rd, rs/imm6
S ← !(rd != rsval)
Checks if the values are not equal. S is a "skip flag." If the condition is evaluated to false, the next instruction will be skipped

if.lt

Opcode: 28
Asm: if.lt rd, rs/imm6
S ← !(rd < rsval)
Checks if rd is less than rsval. S is a “skip flag.” If the condition is evaluated to false, the next instruction will be skipped

if.ge

Opcode: 29
Asm: if.ge rd, rs/imm6
S ← !(rd >= rsval)
Checks if rd is greater than or equal to rsval. S is a “skip flag.” If the condition is evaluated to false, the next instruction will be skipped

In skipping instructions, exactly one regular instruction will be skipped if the condition is false. Each skipped instruction uses one clock cycle (processor treats this as a nop instruction)