Instruction Set Reference

Mostly an extension of Dr. Bruce Jacob's RiSC-16 ISA: https://user.eng.umd.edu/ blj/risc/RiSC-isa.pdf
- 3 Bit opcodes, 8 registers, 4 flags (Zero, Sign, Carry, Overflow)
- reads from r0 always return 0, writes to r0 are ignored
- Memory is word addressable, each address contains contains a 16 bit word

ALU Instructions

Opcode: `000`
Format: `000aaabbbxxxxccc`
`rA` is the destination, `rB` and `rC` are sources.

Binary Pattern	Instruction	Description
000aaabbb0000ccc	`nand rA, rB, rC`	Bitwise NAND
000aaabbb0001ccc	`add rA, rB, rC`	Addition
000aaabbb0010ccc	`addc rA, rB, rC`	Add with carry
000aaabbb0011ccc	`or rA, rB, rC`	Bitwise OR
000aaabbb0100ccc	`subc rA, rB, rC`	Subtract with carry
000aaabbb0101ccc	`and rA, rB, rC`	Bitwise AND
000aaabbb0110ccc	`sub rA, rB, rC`	Subtraction
000aaabbb0111ccc	`xor rA, rB, rC`	Bitwise XOR
000aaabbb1000ccc	`not rA, rC`	Bitwise NOT
000aaabbb1001ccc	`shl rA, rC`	Logical shift left
000aaabbb1010ccc	`shr rA, rC`	Logical shift right
000aaabbb1011ccc	`rotl rA, rC`	Rotate left
000aaabbb1100ccc	`rotr rA, rC`	Rotate right
000aaabbb1101ccc	`sshr rA, rC`	Arithmetic shift right
000aaabbb1110ccc	`shrc rA, rC`	Shift right with carry
000aaabbb1111ccc	`shlc rA, rC`	Shift left with carry

Immediate Instructions

addi – Add Immediate

Opcode: `001`
Format: `001aaabbbiiiiiii`
`i` is a 7-bit immediate, sign-extended.
`rA = rB + i`

lui – Load Upper Immediate

Opcode: `011`
Format: `011aaaiiiiiiiiii`
`i` is a 10-bit immediate left-shifted by 6.
`rA = (i << 6)`

Memory Instructions

sw – Store Word

Opcode: `100`
Format: `100aaabbbiiiiiii`

Store `rA` at address `rB + i`

lw – Load Word

Opcode: `101`
Format: `101aaabbbiiiiiii`
Load into `rA` from address `rB + i`

Branch Instructions

Opcode: `110`
Format: `110ccccccllllll`
`i` is a 7-bit immediate, sign-extended.
Branch target = `pc + 1 + i`

Binary Pattern	Mnemonic(s)	Condition
`1100000000iiiiiii`	`bz, beq`	Branch if zero / equal
`1100000001iiiiiii`	`bp`	Branch if positive
`1100000010iiiiiii`	`bn`	Branch if negative
`1100000011iiiiiii`	`bc`	Branch if carry
`1100000100iiiiiii`	`bo`	Branch if overflow
`1100000101iiiiiii`	`bnz, bne`	Branch if nonzero / not equal
`1100000110iiiiiii`	`jmp`	Unconditional jump
`1100000111iiiiiii`	`bnc`	Branch if not carry
`1100001000iiiiiii`	`bg`	Branch if greater (signed)
`1100001001iiiiiii`	`bge`	Branch if $\geq$ (signed)
`1100001010iiiiiii`	`bl`	Branch if less (signed)
`1100001011iiiiiii`	`ble`	Branch if $\leq$ (signed)
`1100001100iiiiiii`	`ba`	Branch if above (unsigned)
`1100001101iiiiiii`	`bae`	Branch if above or equal
`1100001110iiiiiii`	`bb`	Branch if below (unsigned)
`1100001111iiiiiii`	`bbe`	Branch if below or equal
`1100010000iiiiiii`	`bno`	Branch if not overflow

Other Instructions

jalr – Jump and Link Register

Opcode: `111`, bottom 7 bits must be 0
Format: `111aaabbb0000000`
`rA = pc + 1`, jump to address in `rB`

halt – Halt Execution

`jalr` with any **nonzero** immediate in the bottom 7 bits
Format: `111xxxyyy1110000`

Unused

Opcode `010` is currently unused.