Load/Store Instructions

- Address computation: RD := M(sext(imm)+RS1) means:

$$RD \leftarrow M[adr]$$

where

$$adr = \mathrm{mod}(\langle R_{\langle RS1 \rangle} \rangle + \langle sext_{32}(imm) \rangle, 2^{32}).$$

- Load/Store-instructions:

Load/Store	Semantics
`lw RD RS1 imm`	RD := M(sext(imm)+RS1)
`sw RD RS1 imm`	M(sext(imm)+RS1) := RD

Add immediate

- RD := RS1 + sext(imm) means:

$$[R_{\langle RD \rangle}] \leftarrow \mathrm{mod}([R_{\langle RS1 \rangle}] + [sext_{32}(imm)], 2^{32}).$$

- Immediate–instructions:

Instruction	Semantics
`addi RD RS1 imm`	RD := RS1 + sext(imm)

Shift Compute Instructions

Instruction	Semantics
`slli RD RS1`	RD := RS1 $<<$ 1
`srli RD RS1`	RD := RS1 $>>$ 1
`add RD RS1 RS2`	RD := RS1 $+$ RS2
`sub RD RS1 RS2`	RD := RS1 $-$ RS2
`and RD RS1 RS2`	RD := RS1 $\wedge$ RS2
`or RD RS1 RS2`	RD := RS1 $\vee$ RS2
`xor RD RS1 RS2`	RD := RS1 $\oplus$ RS2

Test Instructions

Instruction	Semantics
`s`*rel*`i RD RS1 imm`	RD := 1, if condition is satisfied,
	RD := 0 otherwise
if *rel* =`lt`	test if RS1 $<$ sext(imm)
if *rel* =`eq`	test if RS1 $=$ sext(imm)
if *rel* =`gt`	test if RS1 $>$ sext(imm)
if *rel* =`le`	test if RS1 $\leq$ sext(imm)
if *rel* =`ge`	test if RS1 $\geq$ sext(imm)
if *rel* =`ne`	test if RS1 $\neq$ sext(imm)

Jump & Misc. Instructions

- Jump–instructions:

Instruction	Semantics
`beqz RS1 imm`	PC = PC + 1 + sext(imm), if RS1 = 0
	PC = PC + 1, if RS1 $\neq$ 0
`bnez RS1 imm`	PC = PC + 1, if RS1 = 0
	PC = PC + 1 + sext(imm), if RS1 $\neq$ 0
`jr RS1`	PC = RS1
`jalr RS1`	R31 = PC+1; PC = RS1

- Miscellaneous–instructions:

Instruction	Semantics
`special-nop`	causes transition to Init/Fetch states
`halt`	causes transition to HALT state