INST	MNEMONIC	ACTION	FORM	FUN 7	FUN 3	OPCODE
LOAD (5)	LB RD, RS1, IMM	[RD] <- [[RS1 + IMM]] (Byte-word)	I TYPE	-	000	0000011
	LH RD, RS1, IMM	[RD] <- [[RS1 + IMM]] (Half-word)			001	
	LW RD, RS1, IMM	[RD] <- [[RS1 + IMM]] (word)			010	
	LBU RD, RS1, IMM	[RD] <- [[RS1 + IMM]] (Byte-word Upper)			100	
	LHU RD, RS1, IMM	[RD] <- [[RS1 + IMM]] (Half-word Upper)			101	
STORE (3)	SB RS1, RS2, IMM	[[RS1+IMM]] <- RS2 (Byte-word)	S TYPE	-	000	0100011
	SH RS1, RS2, IMM	[[RS1+IMM]] <- RS2 (Half-word)			001	
	SW RS1, RS2, IMM	[[RS2+IMM]] <- RS1 (word)			010	
SHIFTS (6)	SLL RD, RS1, RS2	[RD] <- ([RS1] << [RS2])	R TYPE	0000000	001	0110011
	SRL RD, RS1, RS2	[RD] <- ([RS1] >> [RS2])		0000000	101	
	SRA RD, RS1, RS2	[RD] <- ([RS1] >>> [RS2])		0100000	101	
	SLLI RD, RS1, SHAMT	[RD] <- ([RS1] << IMM [4:0]) (lower 5 bits)	I TYPE	0000000	001	0010011
	SRLI RD, RS1, SHAMT	[RD] <- ([RS1] >> IMM [4:0]) (lower 5 bits)		0000000	101	
	SRAI RD, RS1, SHAMT	[RD] <- ([RS1] >>> IMM [4:0]) (lower 5 bits)		0100000	101	
ARITH (8)	ADD RD, RS1, RS2	[RD] <- [RS1] + [RS2]	R TYPE	0000000	000	0110011
	ADDI RD, RS1, IMM	[RD] <- [RS1] + IMM	I TYPE	-	000	0010011
	SUB RD, RS1, RS2	[RD] <- [RS1] - [RS2]	R TYPE	0100000	000	0110011
	LUI RD, IMM	[RD [31:12]] <- IMM; [RD [11:0]] <- 0's	U TYPE	-	-	0110111
	MUL RD, RS1, RS2	[RD] <- [RS1]*[RS2](Stores lower bits, U*U)	R TYPE	0000001	000	0110011
	MULH RD, RS1, RS2	[RD] <- [RS1]*[RS2](Stores upper bits, S*S)			001	
	MULHSU RD, RS1, RS2	[RD] <- [RS1]*[RS2](Stores upper bits, S*U)			010	
	MULHU RD, RS1, RS2	[RD] <- [RS1]*[RS2](Stores upper bits, U*U)			011	
LOGIC (6)	AND RD, RS1, RS2	[RD] <- [RS1] & [RS2]	R TYPE	0000000	111	0110011
	OR RD, RS1, RS2	[RD] <- [RS1] | [RS2]			110	
	XOR RD, RS1, RS2	[RD] <- [RS1] ^ [RS2]			100	
	ANDI RD, RS1, IMM	[RD] <- [RS1] & IMM	I TYPE	-	111	0010011
	ORI RD, RS1, IMM	[RD] <- [RS1] | IMM			110	
	XORI RD, RS1, IMM	[RD] <- [RS1] ^ IMM			100	
CMP (4)	SLT RD, RS1, RS2	[RD] <- 1'b1 if [RS1] < [RS2]	R TYPE	0000000	010	0110011
	SLTU RD, RS1, RS2	[RD] <- 1'b1 if [RS1]< [RS2] (unsigned RS2)			011	
	SLTI RD, RS1, IMM	[RD] <- 1'b1 if [RS1] < IMM	I TYPE	-	010	0010011
	SLTIU RD, RS1, IMM	[RD] <- 1'b1 if [RS1]< IMM (unsigned IMM)			011	
BRNCH (6)	BEQ RS1, RS2, IMM	[PC] <- [PC] + IMM if [RS1] === [RS2]	SB TYPE (offset) +/- 4KB (MULT of 2)	-	000	1100011
	BNE RS1, RS2, IMM	[PC] <- [PC] + IMM if !([RS1] === [RS2])			001	
	BLT RS1, RS2, IMM	[PC] <- [PC] + IMM if [RS1] < [RS2]			100	
	BGE RS1, RS2, IMM	[PC] <- [PC] + IMM if [RS1] >= [RS2]			101	
	BLTU RS1, RS2, IMM	[PC] <- [PC] + IMM if [RS1]< RS2](Unsigned)			110	
	BGEU RS1, RS2, IMM	[PC]<-[PC]+IMM if [RS1]>= RS2] (Unsigned)			111	

JUMP (2)	JAL RD, IMM	[RD] <- [PC] + 4, [PC] <- [PC] + IMM	UJ	-	-	1101111
	JALR RD, RS1, IMM	[RD] <- [PC] + 4, [PC] <- [PC] + [RS1] + IMM	I			1100111
IO (2)	IN	[RD] <- Input Port (32 bit data)	IO	IO		0000101
	OUT	Output Port (32 bit data) <- ALU, DMEM	IO	IO		0000101
PC	AUIPC	[RD] <- [PC] + IMM [31:12]	U TYPE			0010111
FIXED POINT	ADDF RD, RS1, RS2	[RD] <- [RS1][15:0] + [RS2][15:0]	R TYPE	0000010	000	0110011
	MULF RD, RS1, RS2	[RD] <- [RS1][15:0] * [RS2][15:0]	R TYPE	0000010	001	0110011

**Highlighted instructions not supported yet*

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6–7	t1–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller

`nop`	`addi x0, x0, 0`	No operation
`li rd, immediate`	*Myriad sequences*	Load immediate
`mv rd, rs`	`addi rd, rs, 0`	Copy register
`not rd, rs`	`xori rd, rs, -1`	One's complement
`neg rd, rs`	`sub rd, x0, rs`	Two's complement
`negw rd, rs`	`subw rd, x0, rs`	Two's complement word
`sext.w rd, rs`	`addiw rd, rs, 0`	Sign extend word
`seqz rd, rs`	`sltiu rd, rs, 1`	Set if = zero
`snez rd, rs`	`sltu rd, x0, rs`	Set if ≠ zero
`sltz rd, rs`	`slt rd, rs, x0`	Set if < zero
`sgtz rd, rs`	`slt rd, x0, rs`	Set if > zero

`beqz rs, offset`	`beq rs, x0, offset`	Branch if = zero
`bnez rs, offset`	`bne rs, x0, offset`	Branch if ≠ zero
`blez rs, offset`	`bge x0, rs, offset`	Branch if ≤ zero
`bgez rs, offset`	`bge rs, x0, offset`	Branch if ≥ zero
`bltz rs, offset`	`blt rs, x0, offset`	Branch if < zero
`bgtz rs, offset`	`blt x0, rs, offset`	Branch if > zero
`bgt rs, rt, offset`	`blt rt, rs, offset`	Branch if >
`ble rs, rt, offset`	`bge rt, rs, offset`	Branch if ≤
`bgtu rs, rt, offset`	`bltu rt, rs, offset`	Branch if >, unsigned
`bleu rs, rt, offset`	`bgeu rt, rs, offset`	Branch if ≤, unsigned
`j offset`	`jal x0, offset`	Jump
`jal offset`	`jal x1, offset`	Jump and link
`jr rs`	`jalr x0, rs, 0`	Jump register
`jalr rs`	`jalr x1, rs, 0`	Jump and link register
`ret`	`jalr x0, x1, 0`	Return from subroutine