Instruction Formats

31 16	15 13	12 10	9 7	6 4	3 0	
	rs1	fun3	rs2	rd	opcode	R-Type
	rs1	fun3	imm[2:0]	rd	opcode	I-Type
	rs1	fun3	imm[5:0]		opcode	B-Type
	rs1	imm[5:0]		rd	opcode	L-Type
	rs1	imm[5:3]	rs2	imm[2:0]	opcode	S-Type
	imm[8:0]			rd	opcode	J-Type
imm[15:0]	fun3	rs1	rs2	rd	opcode	W-Type

Instructions

Instr	Name	Ty	Op	Fun3	Description
add	Add	R	0000	000	rd, CRY = rs1 + rs2
sub	Subtract	R	0000	001	rd, CRY = rs1 - rs2
sl	Shift Left	R	0000	010	rd, CRY = rs1 << rs2
sr	Shift Right	R	0000	011	rd, CRY = rs1 >> rs2 *arithmetic shift
mul	Multiply	R	0000	100	rd, CRY = rs1 * rs2 *signed
or	Or	R	0000	101	rd = rs1 | rs2
xor	Exclusive Or	R	0000	110	rd = rs1 ^ rs2
and	And	R	0000	111	rd = rs1 & rs2
addc	Add Carry	R	0001	000	rd, CRY = rs1 + rs2 + CRY
subc	Subtract Carry	R	0001	001	rd, CRY = rs1 - rs2 - CRY
slc	Shift Left Carry	R	0001	010	rd, CRY = rs1 << rs2 *carry-filled
src	Shift Right Carry	R	0001	011	rd, CRY = rs1 >> rs2 *carry-filled
addi	Add Immediate	I	0010	000	rd, CRY = rs1 + imm
subi	Subtract Immediate	I	0010	001	rd, CRY = rs1 - imm
sli	Shift Left Immediate	I	0010	010	rd, CRY = rs1 << imm
sri	Shift Right Immediate	I	0010	011	rd, CRY = rs1 >> imm
muli	Multiply Immediate	I	0010	100	rd, CRY = rs1 * imm *signed
ori	Or Immediate	I	0010	101	rd = rs1 | imm

Instr	Name	Ty	Op	Fun3	Description
xori	Exclusive Or Immediate	I	0010	110	rd = rs1 ^ imm
andi	And Immediate	I	0010	111	rd = rs1 & imm
addsi	Add Accumulate	B	0011	000	rs1, CRY = rs1 + imm *imm is unsigned
subsi	Subtract Accumulate	B	0011	001	rs1, CRY = rs1 - imm *imm is unsigned
slsi	Shift Left Accumulate	B	0011	010	rs1, CRY = rs1 << imm *imm is unsigned
srsi	Shift Right Accumulate	B	0011	011	rs1, CRY = rs1 >> imm *imm is unsigned
mulsi	Multiply Accumulate	B	0011	100	rs1, CRY = rs1 * imm *signed
orsi	Or Accumulate	B	0011	101	rs1 = rs1 | imm
xorsi	Exclusive Or Accumulate	B	0011	110	rs1 = rs1 ^ imm
andsi	And Accumulate	B	0011	111	rs1 = rs1 & imm
addiw	Add Immediate Word	W	0100	000	rd, CRY = rs1 + imm
subiw	Subtract Immediate Word	W	0100	001	rd, CRY = rs1 - imm
sliw	Shift Left Immediate Word	W	0100	010	rd, CRY = rs1 << rs2
sriw	Shift Right Immediate Word	W	0100	011	rd, CRY = rs1 >> rs2 *arithmetic shift
muliw	Multiply Immediate Word	W	1.	100	rd, CRY = rs1 * imm
oriw	Or Immediate Word	W	0100	101	rd = rs1 | imm
xoriw	Exclusive Or Immediate Word	W	0100	110	rd = rs1 ^ imm
andiw	And Immediate Word	W	0100	111	rd = rs1 & imm
li	Load Immediate	J	0101		rd = imm
liw	Load Immediate Word	W	0110		rd = imm
lw	Load Word	L	0111		rd = M[rs1 + imm]
sw	Store Word	S	1000		M[rs1 + imm] = rs2
beqz	Branch Equal Zero	B	1001	000	if rs1 == 0 then PC += imm
bnez	Branch Not Equal Zero	B	1001	001	if rs1 ≠ 0 then PC += imm
bgtz	Branch Greater Than Zero	B	1001	010	if rs1 > 0 then PC += imm
blez	Branch Less or Equal Zero	B	1001	011	if rs1 ⩽ 0 then PC += imm
bltz	Branch Less Than Zero	B	1001	110	if rs1 < 0 then PC += imm
bgez	Branch Greater Or Equal Zero	B	1001	111	if rs1 ⩾ 0 then PC += imm

Instr	Name	Ty	Op	Fun3	Description
j	Jump	J	1010		`PC += imm`
jr	Jump Register	I	1011		`PC = rs1 + imm`
jal	Jump and Link	J	1100		`rd = PC + 1; PC += imm`
jwal	Jump Word and Link	W	1101		`rd = PC + 2; PC = imm`
iact	Interrupt Activation	I	1110	000	`IE[imm[2:0]] = 0`
iact	Interrupt Activation	I	1110	001	`IE[imm[2:0]] = 1`
iloc	Interrupt Location	I	1110	010	`IL[imm[2:0]] = rs1`
itrg	Interrupt Trigger	I	1110	011	`trigger interrupt imm[2:0]`
iret	Return From Interrupt	I	1110	100	`PC = IR; IA=0`
lcry	Load Carry	I	1111	000	`rd = CRY`
stmr	Store Timer	I	1110	010	`TMR[imm[1:0]] = rs1`
ltmr	Load Timer	I	1110	011	`rd = TMR[imm[1:0]]`
break	Breakpoint	I	1111	111	`Pause execution`

User Registers

Register	Asm. Name	Description	Saver
x0	ra	Return Address	Caller
x1	sp	Stack Pointer	
x2-x3	t0-t1	Temporary	Caller
x4-x5	s0-s1	Saved	Callee
x6-x7	a0-a1	Argument/Return	Caller

Special Registers

Register	Name	Description
PC	Program counter	Location of program execution
CRY	Carry	Holds the carry result from arithmetic
M	Memory	Anything in the address space
IE	Interrupt enable	Enables or disables interrupts
IL	Interrupt location	Interrupt handler addresses
IA	Interrupt active	Is an interrupt being handled, alternative user registers are used if true

Register	Name	Description
`IR`	Interrupt return	Return address for interrupt
`TMR`	Timer	Decrements every microsecond

Address space

Range	Name	Description
`0000-1FFF`	Program ROM	8K ROM that can only by executed
`0000-1FFF`	Data ROM	8K ROM that can only by read
`2000-3FFF`	Memory	8K RAM for user code