Instruction Formats

23 22	21 17	16 12	11 8	7 3	2 0	
fun2	rs2	rs1	fun4	rd	opcode	R-type
imm[6:0]		rs1	fun4	rd	opcode	I-type
imm[6:5]	rs2	rs1	fun4	imm[4:0]	opcode	B-type
imm[6:0|15:7]				rd	opcode	J-type

*imm is sign extended

Instructions

Instr	Name	F	Op	Fun2	Fun4	Description
add	Add	R	000	00	0000	`rd, CRY = rs1 + rs2`
addc	Add Carry	R	000	10	0000	`rd, CRY = rs1 + rs2 + CRY`
sub	Subtract	R	000	01	0000	`rd, CRY = rs1 - rs2`
subc	Subtract Carry	R	000	11	0000	`rd, CRY = rs1 - rs2 - CRY`
mul	Multiply	R	000	00	0001	`rd, CRY = rs1 * rs2`
mulc	Multiply Carry	R	000	10	0001	`rd, CRY = rs1 * rs2 + CRY`
sll	Shift Left Logical	R	000	00	0010	`rd, CRY = rs1 << rs2`
slc	Shift Left Carry	R	000	10	0010	`rd, CRY = rs1 << rs2   *carry-filled`
srl	Shift Right Logical	R	000	00	0011	`rd, CRY = rs1 >>> rs2`
sra	Shift Right Arithmetic	R	000	01	0011	`rd, CRY = rs1 >> rs2`
src	Shift Right Carry	R	000	10	0011	`rd, CRY = rs1 >> rs2   *carry-filled`
or	Or	R	000	00	0100	`rd = rs1 | rs2`
xor	Exclusive Or	R	000	00	0101	`rd = rs1 ^ rs2`
and	And	R	000	00	0110	`rd = rs1 & rs2`
st.eq	Set Equal	R	000	00	1000	`rd = rs1 == rs2`
st.ne	Set Not Equal	R	000	00	1001	`rd = rs1 ≠ rs2`
st.gt	Set Greater Than	R	000	00	1010	`rd = rs1 > rs2`
st.le	Set Less or Equal	R	000	00	1011	`rd = rs1 ⩽ rs2`
st.gtu	Set Greater Than U.	R	000	00	1100	`rd = rs1 > rs2   *unsigned`
st.leu	Set Less or Equal U..	R	000	00	1101	`rd = rs1 ⩽ rs2   *unsigned`

Instr	Name	F	Op	Fun2	Fun4	Description
addi	Add Immediate	I	001		0000	`rd, CRY = rs1 + imm`
muli	Multiply Immediate	I	001		0001	`rd, CRY = rs1 + imm`
slli	Shift Left Logical Imm.	I	001		0010	`rd, CRY = rs1 << imm`
srli	Shift Right Logical Imm.	I	001		0011	`rd, CRY = rs1 >> imm`
srai	Shift Right Arith. Imm.	I	001	`imm[5]=1`	0011	`rd, CRY = rs1 >>> imm`
ori	Or Immediate	I	001		0100	`rd = rs1 | imm`
xori	Exclusive Or Immediate	I	001		0101	`rd = rs1 ^ imm`
andi	And Immediate	I	001		0110	`rd = rs1 & imm`
st.eqi	Set Equal Immediate	I	001		1000	`rd = rs1 == imm`
st.nei	Set Not Equal Imm.	I	001		1001	`rd = rs1 != imm`
st.gti	Set Greater Than Imm.	I	001		1010	`rd = rs1 > imm`
st.lei	Set Less or Equal Imm.	I	001		1011	`rd = rs1 <= imm`
st.gtui	Set Greater Than U. Imm.	I	001		1100	`rd = rs1 > imm *unsigned`
st.leui	Set Less or Equal U. Imm.	I	001		1101	`rd = rs1 <= imm *unsigned`
jalr	Jump and Link Register	I	001		1110	`rd = PC + 3; PC = rs1 + imm`
br.eq	Branch Equal	B	010		1000	`if rs1 == rs2 then PC += imm`
br.ne	Branch Not Equal	B	010		1001	`if rs1 != rs2 then PC += imm`
br.gt	Branch Greater Than	B	010		1010	`if rs1 > rs2 then PC += imm`
br.le	Branch Less or Equal	B	010		1011	`if rs1 <= rs2 then PC += imm`
br.gtu	Branch Greater Than U.	B	010		1100	`if rs1 > rs2 then PC += imm   *unsigned`
br.leu	Branch Less or Equal U.	B	010		1101	`if rs1 <= rs2 then PC += imm   *unsigned`
lb	Load Byte	I	011		0000	`rd = M[rs1 + imm][7:0]`
lbu	Load Byte U.	I	011		0010	`rd = M[rs1 + imm][7:0] *zero-extended`
lw	Load Word	I	011		0001	`rd = M[rs1 + imm]`
sb	Store Byte	B	100		0000	`M[rs1 + imm] = rs2[7:0]`
sw	Store Word	B	100		0001	`M[rs1 + imm] = rs2`
jal	Jump and Link	J	101			`rd = PC + 3; PC = imm`
li	Load Immediate	J	110			`rd = imm`

Instr	Name	F	Op	Fun2	Fun4	Description
intr.di	Disable Interrupt	I	111	imm[3]=0	0000	IE[imm[2:0]] = 0
intr.en	Enable Interrupt	I	111	imm[3]=1	0000	IE[imm[2:0]] = 1
intr.sl	Set Interrupt Location	I	111		0001	IL[imm[2:0]] = rs1

intr.t	Trigger Interrupt	I	111		0010	if ID < 3 && IE[imm[2:0]] then
						  rd=1; IS[ID]=PC; PC=IL[imm[2:0]]; ID++
						else rd=0

intr.r	Return From Interrupt	I	111		0011	PC = IS[ID]; ID--
tmr.wl	Write timer low	I	111	imm[2]=0	0100	TMR[imm[1:0]][15:0] = rs1
tmr.wh	Write timer high	I	111	imm[2]=1	0100	TMR[imm[1:0]][31:16] = rs1
tmr.rl	Read timer low	I	111	imm[2]=0	0101	rd = TMR[imm[1:0]][15:0]
tmr.rh	Read timer high	I	111	imm[2]=1	0101	rd = TMR[imm[1:0]][31:16]
sbrom	Select Bootloader ROM	I	111		0110	BS = imm[0]
st.lti	Set Less Than Imm.	I	111		1010	rd = rs1 < imm
st.gei	Set Greater or Equal Imm.	I	111		1011	rd = rs1 ⩾ imm
st.ltui	Set Less Than U. Imm.	I	111		1100	rd = rs1 < imm *unsigned
st.geui	Set Greater or Equal U. Imm.	I	111		1101	rd = rs1 ⩾ imm *unsigned
break	Breakpoint	I	111		1111	Pause execution

User Registers

Register	Asm. Name	Description	Saver
x0	zero	Zero constant	
x1	ra	Return Address	Caller
x2	sp	Stack Pointer	
x3-x10	t0-t7	Temporary	Caller
x11-x23	s0-s12	Saved	Callee
x24-x31	a0-a7	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
IS	Interrupt stack	Stack of return address for interrupts
ID	Interrupt depth	Number of interrupts being handled
TMR	Timer	Decrements every 500 nanoseconds
BS	Bootloader Select	Selects the bootloader, non-volatile

Address space

Range	Name	Description
`0000-0FFF`	Bootloader	4K ROM with write-protected startup codeSS
`0000-0FFF`	User bootloader	4K EEPROM with alternative user startup code
`1000-1FFF`	Storage	4K EEPROM to save user state
`2000-3FFF`	Memory	8K RAM for user code