General Name	ABI Name	Function
x0	zero	Hardwired 0
x1-x10	a0-a9	General Purpose/Function Arguments/Return Values
x11	sp	Stack Pointer
x12	fp	Frame Pointer
x13	tp	Thread Pointer
x14	ra	Return Address
x15-x24	s0-s9	Saved Registers
x25-x31	t0-t6	Temporary Registers

I	31-20	19-16	15-11	10-6	5-0		
Immediate	imm [11:0]	func1	rs1	rd	opc		
R	31-25	24-21	20-16	15-11	10-6	5-0	
Register	func2	func1	rd	rs2	rs1	opc	
SR	31-25	24-21	20-16	15-11	10-6	5-0	
Store R+R	func2	func1	rs3	rs2	rs1	opc	
S	31-20	19-16	15-11	10-6	5-0		
Store R+I	imm [11:0]	func1	rs1	rs2	opc		
SI	31-16	15-11	10-6	5-0			
Store I	imm [15:0]	func1	rs1	opc			
B	31-20	19-16	15-11	10-6	5-0		
Branch	imm [11:0]	func1	rs1	rs2	opc		
U	31-12	11	10-6	5-0			
Upper	imm[19:0]	func1	rd	opc			

MX RISC Base/I

inst	name	FMT	Opcode	func1	func2	Description	Note	Imm Note
add	Add	R	0x00	0x0	Reserved	rd=rs1+rs2		
sub	Sub	R	0x00	0x1	Reserved	rd=rs1-rs2		
xor	Xor	R	0x00	0x2	Reserved	rd=rs1^rs2		
or	Or	R	0x00	0x3	Reserved	rd=rs1|rs2		
and	And	R	0x00	0x4	Reserved	rd=rs1&rs2		
sll	Shift Left Logical	R	0x00	0x5	Reserved	rd=rs1<<rs2		
srl	Shift Right Logical	R	0x00	0x6	Reserved	rd=rs1>>rs2	zero extends	
sra	Shift Right Arithmetic	R	0x00	0x7	Reserved	rd=rs1>>>rs2	msb extends	
slt	Set Less Than	R	0x00	0x8	Reserved	rd=(rs1<rs2)?1:0	signed	
sltu	Set Less Than Unsigned	R	0x00	0x9	Reserved	rd=(rs1<rs2)?1:0	unsigned	
seq	Set Equal	R	0x00	0xA	Reserved	rd=(rs1=rs2)?1:0	signed	
add	Add Immediate	I	0x01	0x0	N/A	rd=rs1+imm		msb extends
sub	Sub Immediate	I	0x01	0x1	N/A	rd=rs1-imm		msb extends
xor	Xor Immediate	I	0x01	0x2	N/A	rd=rs1^imm		msb extends
or	Or Immediate	I	0x01	0x3	N/A	rd=rs1|imm		msb extends
and	And Immediate	I	0x01	0x4	N/A	rd=rs1&imm		msb extends
sll	Shift Left Logical Immediate	I	0x01	0x5	N/A	rd=rs1<<imm		zero extends
srl	Shift Right Logical Immediate	I	0x01	0x6	N/A	rd=rs1>>imm	zero extends	zero extends
sra	Shift Right Arithmetic Immediate	I	0x01	0x7	N/A	rd=rs1>>>imm	msb extends	zero extends
slt	Set Less Than Immediate	I	0x01	0x8	N/A	rd=(rs1<imm)?1:0	signed	msb extends
sltu	Set Less Than Unsigned Immediate	I	0x01	0x9	N/A	rd=(rs1<imm)?1:0	unsigned	zero extends
seq	Set Equal Immediate	I	0x01	0xA	N/A	rd=(rs1=imm)?1:0	signed	msb extends
sb	Store Byte	S	0x02	0x0	N/A	M[rs1+imm][7:0]=rs2		msb extends
sb	Store Byte	SR	0x02	0x1	Reserved	M[rs1+rs2][7:0]=rs3		
sb	Store Byte	SR	0x02	0x2	Reserved	M[rs1][7:0]=rs2+rs3		
sb	Store Byte	S	0x02	0x3	N/A	M[rs1][7:0]=rs2+imm		msb extends
sb	Store Byte	SI	0x02	0x4	N/A	M[rs1][i:0]=imm		
sh	Store Halfword	S	0x02	0x5	N/A	M[rs1+imm][15:0]=rs2		msb extends
sh	Store Halfword	SR	0x02	0x6	Reserved	M[rs1+rs2][15:0]=rs3		
sh	Store Halfword	SR	0x02	0x7	Reserved	M[rs1][15:0]=rs2+rs3		
sh	Store Halfword	S	0x02	0x8	N/A	M[rs1][15:0]=rs2+imm		msb extends
sh	Store Halfword	SI	0x02	0x9	N/A	M[rs1][15:0]=imm		
sw	Store Word	S	0x02	0xA	N/A	M[rs1+imm][31:0]=rs2		msb extends
sw	Store Word	SR	0x02	0xB	Reserved	M[rs1+rs2][31:0]=rs3		
sw	Store Word	SR	0x02	0xC	Reserved	M[rs1][31:0]=rs2+rs3		
sw	Store Word	S	0x02	0xD	N/A	M[rs1][31:0]=rs2+imm		msb extends
lb	Load Byte	I	0x03	0x1	N/A	rd=M[rs1+imm][7:0]	msb extends	msb extends
lb	Load Byte	R	0x03	0x2	0b0000000	rd=M[rs1+rs2][7:0]	msb extends	
lb	Load Byte	R	0x03	0x3	0b0000000	rd=(M[rs1]+rs2)	msb extends	
lb	Load Byte	I	0x03	0x4	N/A	rd=(M[rs1]+imm)	msb extends	msb extends
lbu	Load Byte Unsigned	I	0x03	0x5	N/A	rd=M[rs1+imm][7:0]	zero extends	msb extends
lbu	Load Byte Unsigned	R	0x03	0x2	0b0000100	rd=M[rs1+rs2][7:0]	zero extends	
lbu	Load Byte Unsigned	R	0x03	0x3	0b0000100	rd=(M[rs1]+rs2)	zero extends	
lbu	Load Byte Unsigned	I	0x03	0x6	N/A	rd=(M[rs1]+imm)	zero extends	msb extends
lh	Load Halfword	I	0x03	0x7	N/A	rd=M[rs1+imm][15:0]	msb extends	msb extends
lh	Load Halfword	R	0x03	0x2	0b0000001	rd=M[rs1+rs2][15:0]	msb extends	
lh	Load Halfword	R	0x03	0x3	0b0000001	rd=(M[rs1]+rs2)	msb extends	
lh	Load Halfword	I	0x03	0x8	N/A	rd=(M[rs1]+imm)	msb extends	msb extends
lhu	Load Halfword Unsigned	I	0x03	0x9	N/A	rd=M[rs1+imm][15:0]	zero extends	msb extends
lhu	Load Halfword Unsigned	R	0x03	0x2	0b0000101	rd=M[rs1+rs2][15:0]	zero extends	
lhu	Load Halfword Unsigned	R	0x03	0x3	0b0000101	rd=(M[rs1]+rs2)	zero extends	
lhu	Load Halfword Unsigned	I	0x03	0xA	N/A	rd=(M[rs1]+imm)	zero extends	msb extends
lw	Load Word	I	0x03	0xB	N/A	rd=M[rs1+imm][31:0]		msb extends
lw	Load Word	R	0x03	0x2	0b0000010	rd=M[rs1+rs2][31:0]		
lw	Load Word	R	0x03	0x3	0b0000010	rd=(M[rs1]+rs2)		
lw	Load Word	I	0x03	0xC	N/A	rd=(M[rs1]+imm)		msb extends
beq	Branch ==	B	0x04	0x0	N/A	if(rs1==rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
bne	Branch !=	B	0x04	0x1	N/A	if(rs1!=rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
blt	Branch <	B	0x04	0x2	N/A	if(rs1<rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
bgte	Branch >=	B	0x04	0x5	N/A	if(rs1>=rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
bltu	Branch < Unsigned	B	0x04	0x6	N/A	if(rs1<rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
bgteu	Branch >= Unsigned	B	0x04	0x9	N/A	if(rs1>=rs2) PC+=imm		msb extends/imm is an offset of 4 bytes
jal	Jump and Link	U	0x5	Reserved	N/A	rd=PC+4; PC+=imm		msb extends/imm is an offset of 4 bytes
jalr	Jump and Link Reg	I	0x6	Reserved	N/A	rd=PC+4; PC=rs1+imm		msb extends/imm is an offset of 4 bytes
lui	Load Upper Imm	U	0x7	0x0	N/A	rd=imm<<12	zero extends	
lli	Load Lower Imm	U	0x7	0x1	N/A	rd=imm	msb extends	msb extends
auipc	Add Upper Imm to PC	U	0x8	Reserved	N/A	rd=PC+(imm<<12)		