Assembly Language Lecture Series: RISC-V Instruction Format

Roger Luis Uy
College of Computer Studies
De La Salle University
Manila, Philippines

RISC-V instruction format

- 4 core instruction formats: (R-type/I-Type/S-type/U-type)
- 2 branch instruction formats: (SB-type/UJ-type)
- Fixed instruction format: 32-bit

Observations:
- opcode: 7 bits
- rs1/rs2/rd: 5 bits
- rs1/rs2/rd are in the same locations
- imm are sign-extended and sign-bit in bit 31
- imm are always on the leftmost bits

	31-25	24-20	19-15	14-12	11-7	6-0
Register (R)	funct7	rs2	rs1	funct3	rd	opcode

	31-20	19-15	14-12	11-7	6-0
Immediate (I)	imm[11:0]	rs1	Funct3	rd	opcode

	31-25	24-20	19-15	14-12	11-7	6-0
Store (S)	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode

	31-25	24-20	19-15	14-12	11-7	6-0
Branch (SB)	Imm[12][10:5]	rs2	rs1	funct3	imm[4:1][11]	opcode

RISC-V instruction Format

	31-12	11-7	6-0
Upper immediate (U)	imm[31:12]	rd	opcode

	31-12	11-7	6-0
Jump (UJ)	Imm[20][10:1][11][19:12]	rd	opcode

R-type

opcode = 0110011

31-25	24-20	19-15	14-12	11-7	6-0
funct7	rs2	rs1	funct3	rd	Opcode
7	5	5	3	5	7

Example

	31-25	24-20	19-15	14-12	11-7	6-0
add x10, x6, x7	0000000	00111	00110	000	01010	0110011
sub x11, x6, x7	0100000	00111	00110	000	01011	0110011

I-type

opcodes: various (load: 0000011; arith/logic: 0010011)

31-20	19-15	14-12	11-7	6-0
imm[11:0]	rs1	Funct3	rd	Opcode
12	5	3	5	7

Example

	31-20	19-15	14-12	11-7	6-0
	imm[11:0]	rs1	Funct3	rd	opcode
lw x6, 0(x8)	0000 0000 0000	01000	010	00110	0000011
addi x7, x0, 0x9	0000 0000 1001	00000	000	00111	0010011

S-type

opcodes: 0100011

31-25	24-20	19-15	14-12	11-7	6-0
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
7	5	5	3	5	7

Example

	31-25	24-20	19-15	14-12	11-7	6-0
sw x10, 0(x9)	0000000	01010	01001	010	00000	0100011

U-type

opcodes: lui (0110111); auipc (0010111)

31-12	11-7	6-0
imm[31:12]	rd	opcode
20	5	7

Example

	31-12	11-7	6-0
lui x10, 0x1234	0000 0001 0010 0011 0100	01010	0110111
Auipc x11, 0xf1234	1111 0001 0010 0011 0100	01011	0010111

SB-type

opcodes: 1100111 (branch)

12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0	1	0	0	0

31-25	24-20	19-15	14-12	11-7	6-0
Imm[12][10:5]	rs2	rs1	funct3	imm[4:1][11]	opcode

Example

Offset = (target address – branch address)/2

Address (in dec)	Instruction	Opcode
00000008	BLT x5, x6, L1	0000000 00110 00101 100 10000 1100011
00000012	xor x7, x5, x6	
00000016	and x8, x5, x6	
00000020	JAL x0, FIN	
00000024 L1:	sll x7, x5, x6	
00000028	srl x8, x5, x6	
00000032 FIN:	addi x0, x0, 0	

L1 is address 24; address of BLT is 8; offset is 24-8 = 16; but encoded as multiple of 2 bytes (i.e., assume the stored offset will be multiplied by 2); so the encoded offset is 16/2 = 8 (i.e., encoded as 12-bit)

UJ-type

opcodes: 1101111 (JAL)

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

31-12	11-7	6-0
Imm[20][10:1][11][19:12]	rd	opcode

Example

Offset = (target address – branch address)/2

Address (in dec)	Instruction	Opcode
00000008	BLT x5, x6, L1	
00000012	xor x7, x5, x6	
00000016	and x8, x5, x6	
00000020	JAL x0, FIN	00000001100000000000 00000 1101111
00000024 L1:	sll x7, x5, x6	
00000028	srl x8, x5, x6	
00000032 FIN:	addi x0, x0, 0	

FIN is address 32; address of JAL is 20; offset is 32-20 = 12; but encoded as multiple of 2 bytes (i.e., assume the stored offset will be multiplied by 2); so the encoded offset is 12/2 = 6 (i.e., encoded as 20-bit)

Try

Address (in dec)	Instruction	Opcode
00000000	addi x5, x0, 4	
00000004	addi x6, x0, 2	
00000008	BLT x5, x6, L1	
00000012	xor x7, x5, x6	
00000016	and x8, x5, x6	
00000020	JAL x0, FIN	
00000024 L1:	sll x7, x5, x6	
00000028	srl x8, x5, x6	
00000032 FIN:	addi x0, x0, 0	