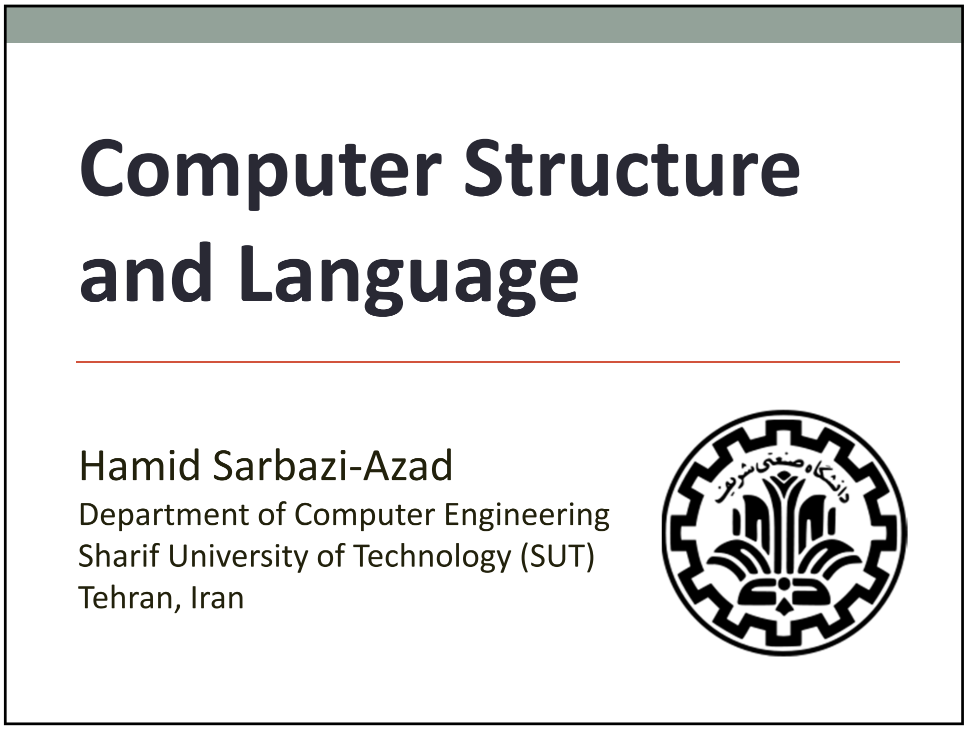
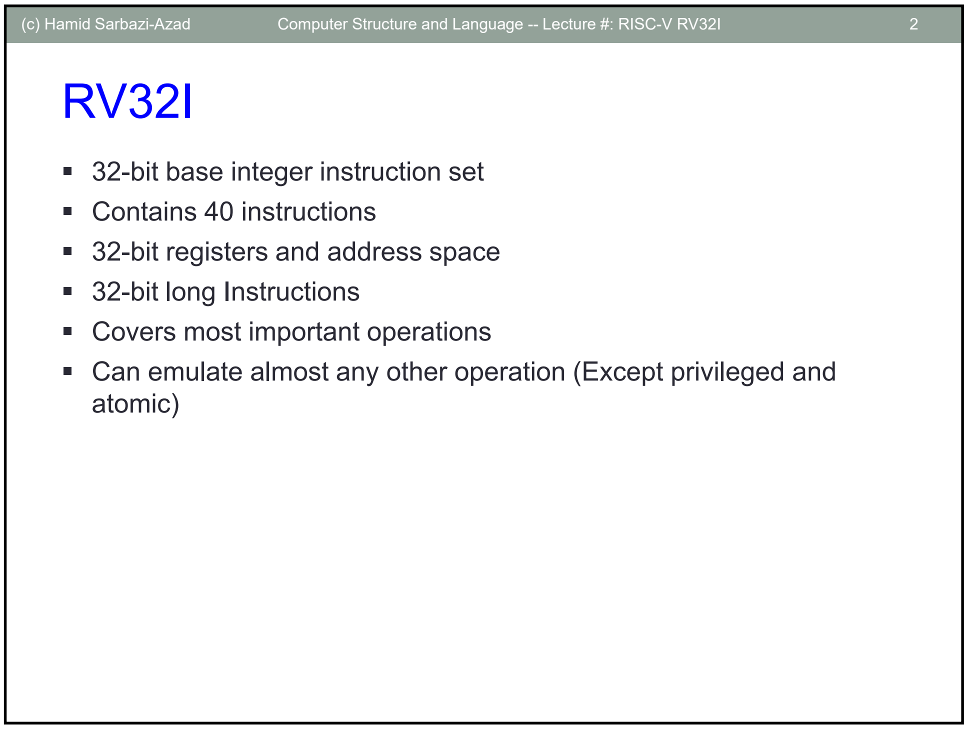


Computer Structure and Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

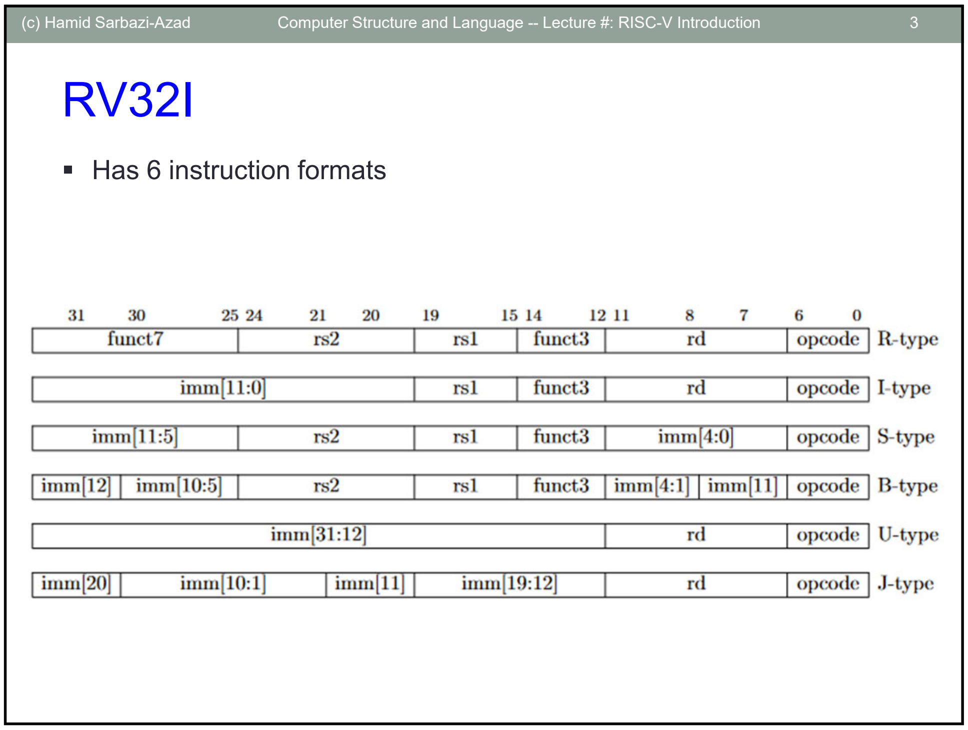
1



RV32I

- 32-bit base integer instruction set
- Contains 40 instructions
- 32-bit registers and address space
- 32-bit long Instructions
- Covers most important operations
- Can emulate almost any other operation (Except privileged and atomic)

2

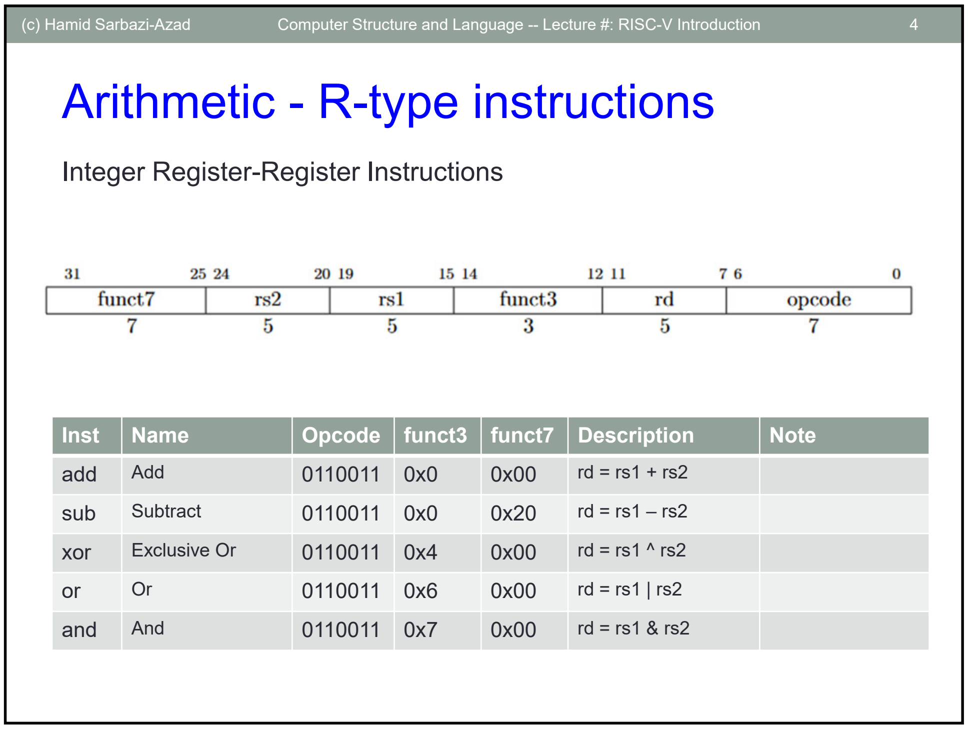


RV32I

- Has 6 instruction formats

31	30 ... 25	24 ... 21	20	19 ... 15	14 ... 12	11 ... 8	7	6 ... 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	J-type

3



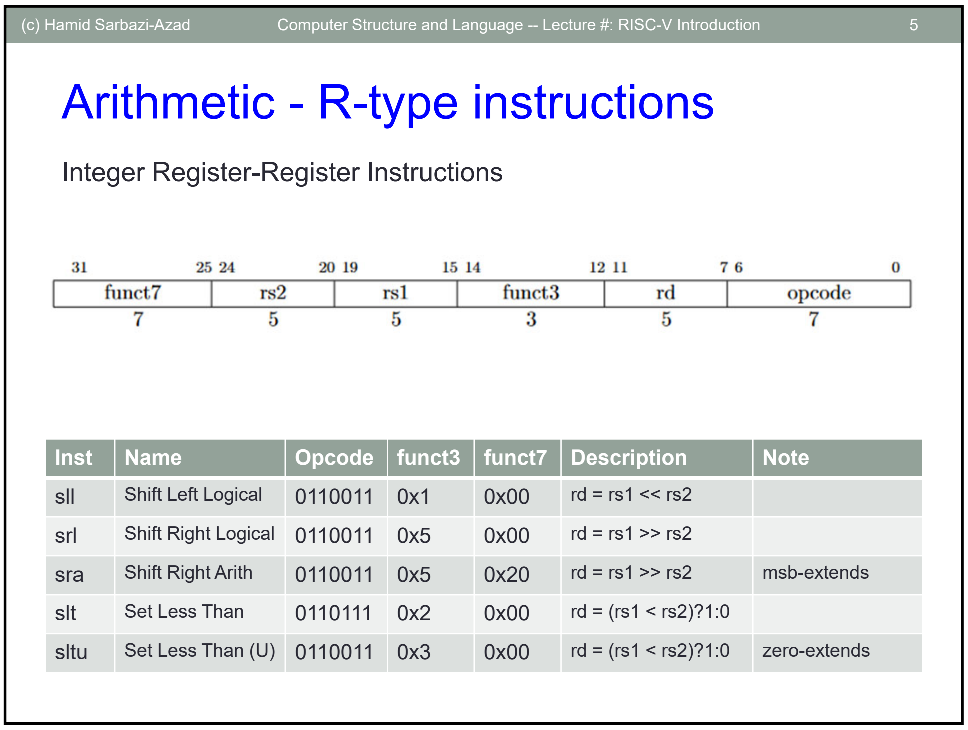
Arithmetic - R-type instructions

Integer Register-Register Instructions

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

Inst	Name	Opcode	funct3	funct7	Description	Note
add	Add	0110011	0x0	0x00	rd = rs1 + rs2	
sub	Subtract	0110011	0x0	0x20	rd = rs1 – rs2	
xor	Exclusive Or	0110011	0x4	0x00	rd = rs1 ^ rs2	
or	Or	0110011	0x6	0x00	rd = rs1 | rs2	
and	And	0110011	0x7	0x00	rd = rs1 & rs2	

4



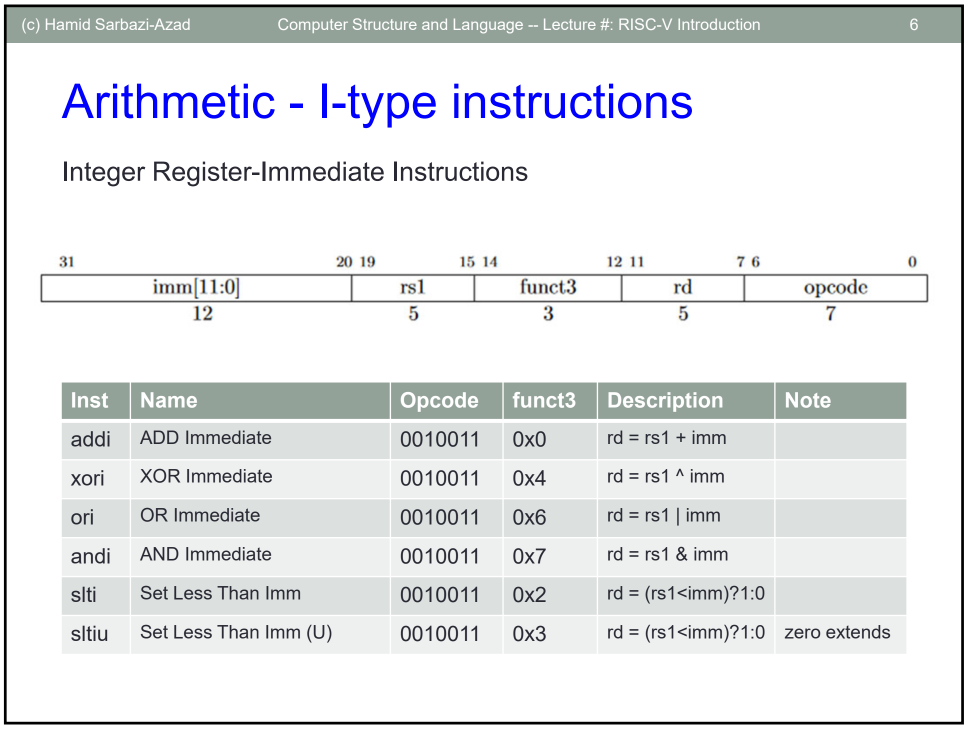
Arithmetic - R-type instructions

Integer Register-Register Instructions

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

Inst	Name	Opcode	funct3	funct7	Description	Note
sll	Shift Left Logical	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith	0110011	0x5	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	0110111	0x2	0x00	rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	0110011	0x3	0x00	rd = (rs1 < rs2)?1:0	zero-extends

5



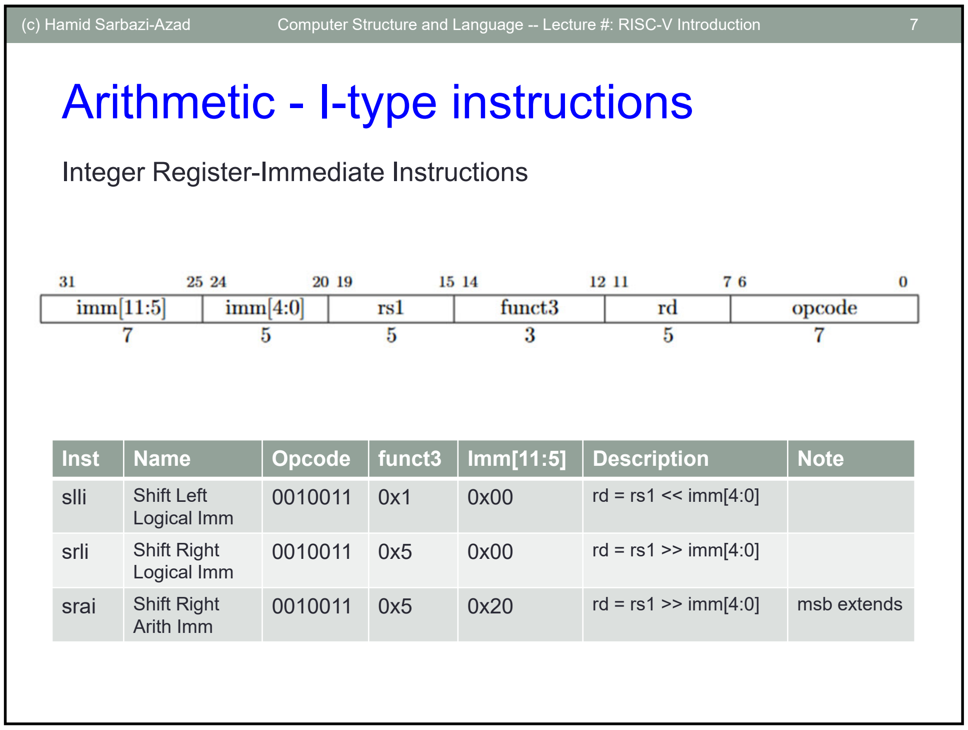
Arithmetic - I-type instructions

Integer Register-Immediate Instructions

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7

Inst	Name	Opcode	funct3	Description	Note
addi	ADD Immediate	0010011	0x0	rd = rs1 + imm	
xori	XOR Immediate	0010011	0x4	rd = rs1 ^ imm	
ori	OR Immediate	0010011	0x6	rd = rs1 | imm	
andi	AND Immediate	0010011	0x7	rd = rs1 & imm	
slti	Set Less Than Imm	0010011	0x2	rd = (rs1<imm)?1:0	
sltiu	Set Less Than Imm (U)	0010011	0x3	rd = (rs1<imm)?1:0	zero extends

6



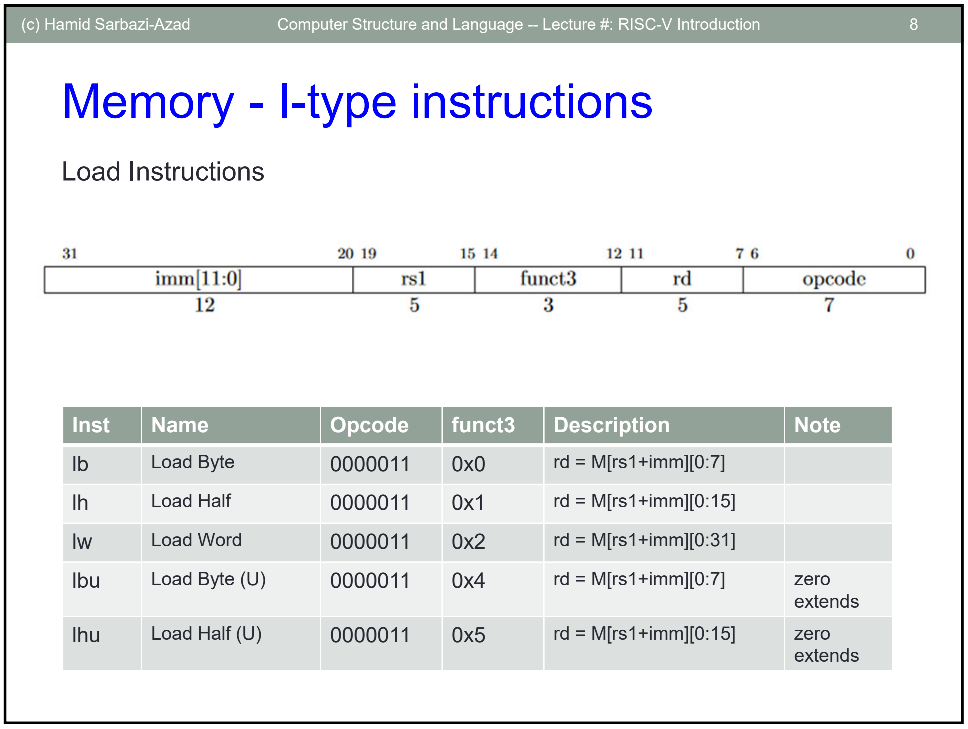
Arithmetic - I-type instructions

Integer Register-Immediate Instructions

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

Inst	Name	Opcode	funct3	Imm[11:5]	Description	Note
slli	Shift Left Logical Imm	0010011	0x1	0x00	rd = rs1 << imm[4:0]	
srli	Shift Right Logical Imm	0010011	0x5	0x00	rd = rs1 >> imm[4:0]	
srai	Shift Right Arith Imm	0010011	0x5	0x20	rd = rs1 >> imm[4:0]	msb extends

7



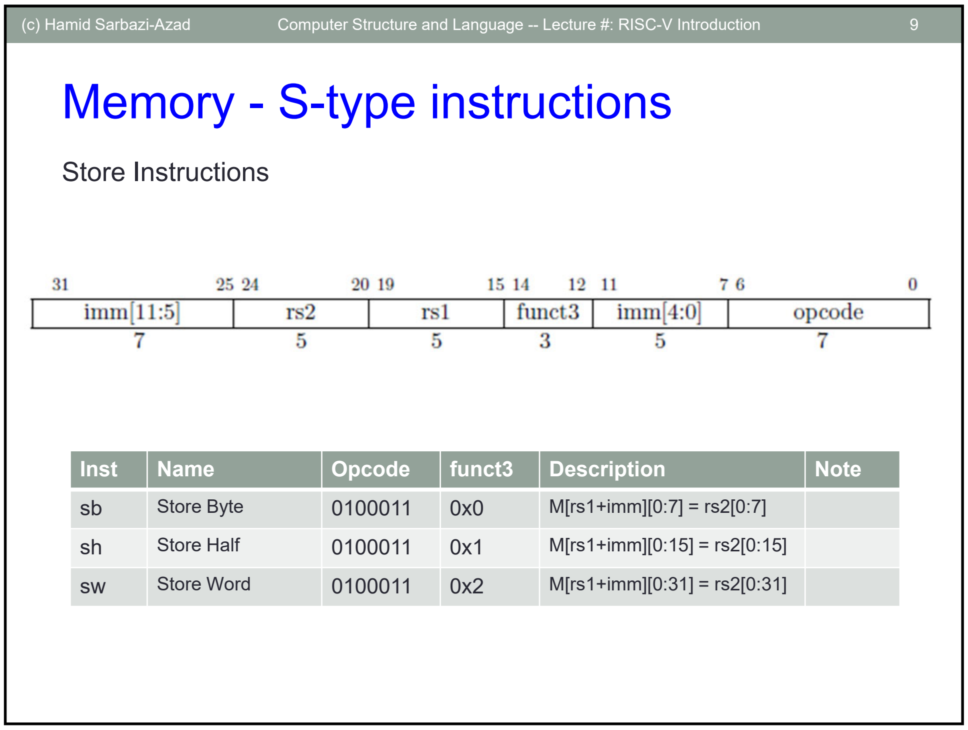
Memory - I-type instructions

Load Instructions

31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7

Inst	Name	Opcode	funct3	Description	Note
lb	Load Byte	0000011	0x0	rd = M[rs1+imm][0:7]	
lh	Load Half	0000011	0x1	rd = M[rs1+imm][0:15]	
lw	Load Word	0000011	0x2	rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	0000011	0x4	rd = M[rs1+imm][0:7]	zero extends
lhu	Load Half (U)	0000011	0x5	rd = M[rs1+imm][0:15]	zero extends

8



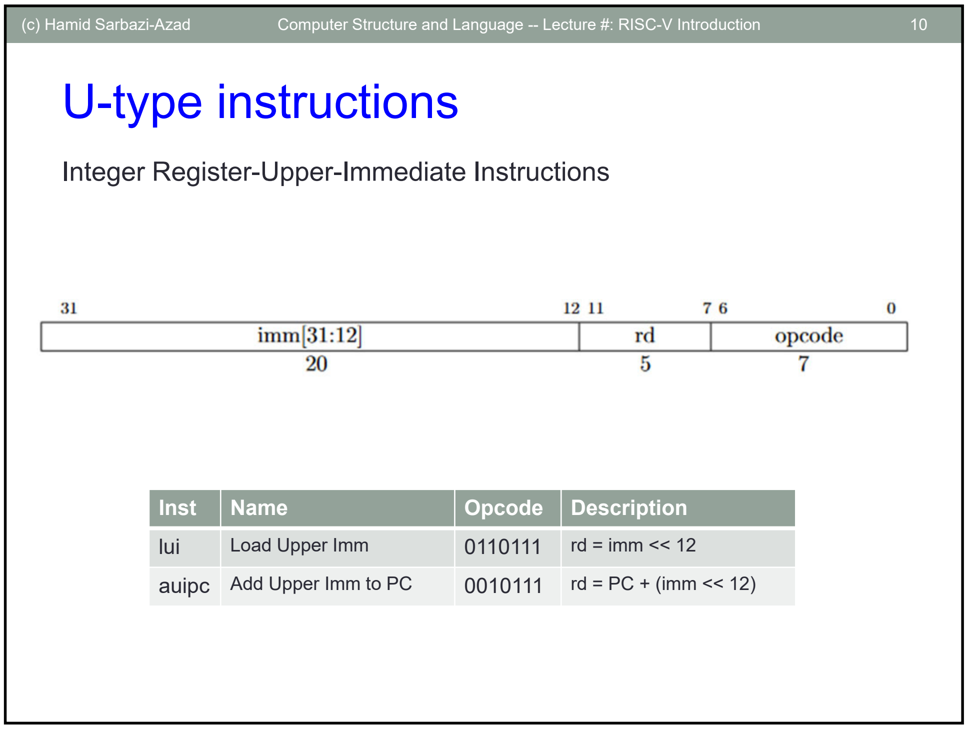
Memory - S-type instructions

Store Instructions

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

Inst	Name	Opcode	funct3	Description	Note
sb	Store Byte	0100011	0x0	M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	0100011	0x1	M[rs1+imm][0:15] = rs2[0:15]	
sw	Store Word	0100011	0x2	M[rs1+imm][0:31] = rs2[0:31]	

9



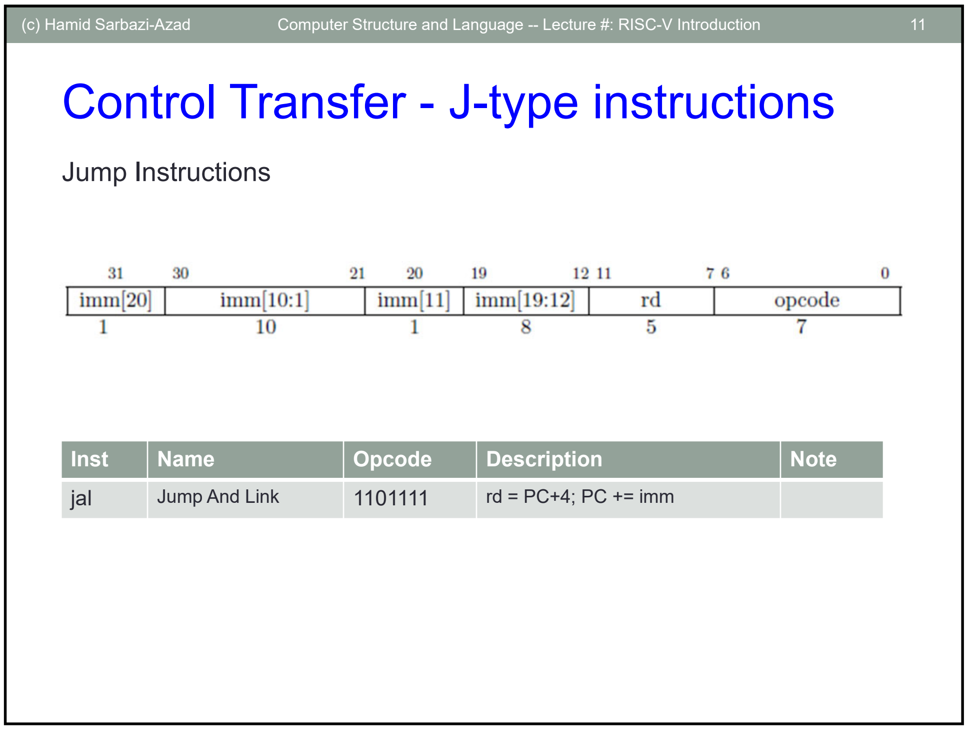
U-type instructions

Integer Register-Upper-Immediate Instructions

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

Inst	Name	Opcode	Description
lui	Load Upper Imm	0110111	rd = imm << 12
auipc	Add Upper Imm to PC	0010111	rd = PC + (imm << 12)

10



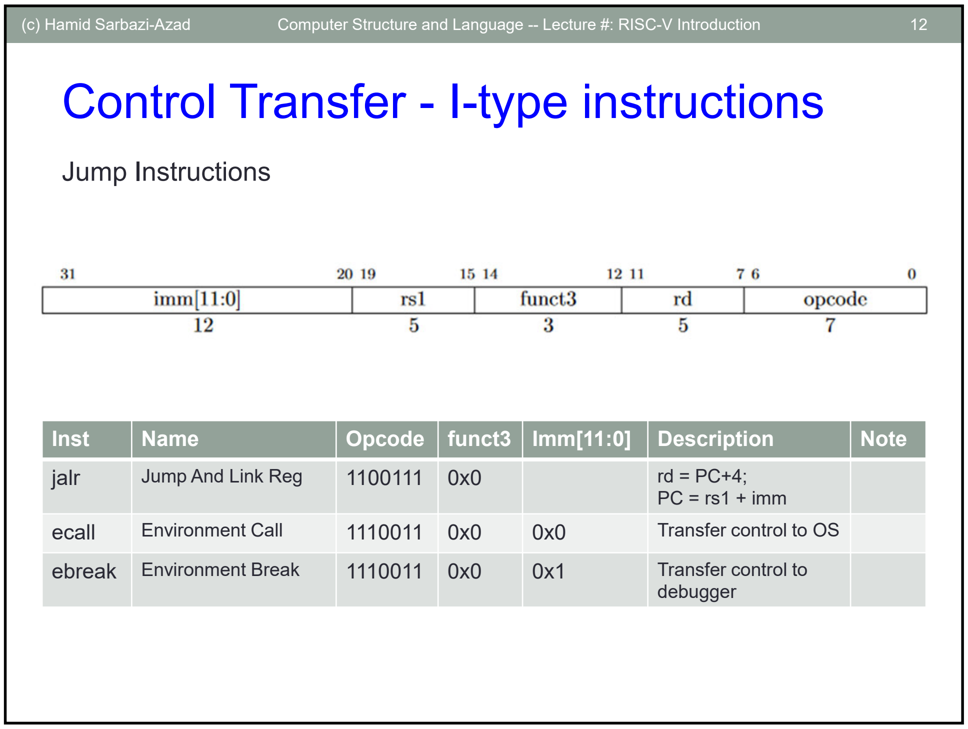
Control Transfer - J-type instructions

Jump Instructions

31	30 — 21	20	19 — 12	11 — 7	6 — 0
imm[20]	imm[10:1]	imm[11]	imm[19:12]	rd	opcode
1	10	1	8	5	7

Inst	Name	Opcode	Description	Note
jal	Jump And Link	1101111	rd = PC+4; PC += imm	

11



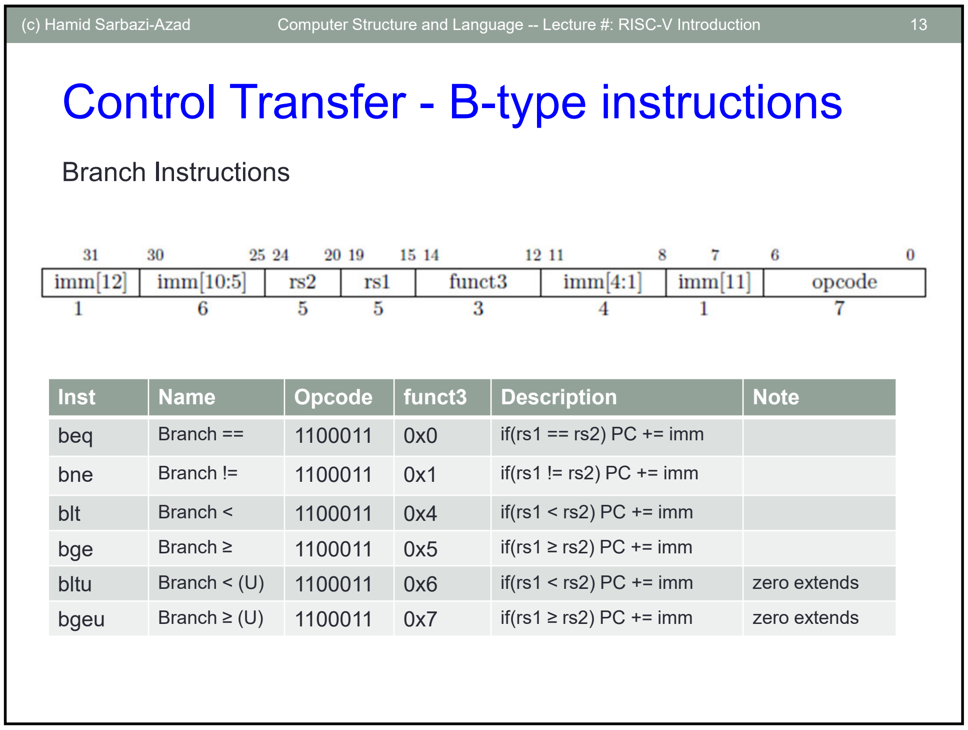
Control Transfer - I-type instructions

Jump Instructions

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7

Inst	Name	Opcode	funct3	Imm[11:0]	Description	Note
jalr	Jump And Link Reg	1100111	0x0		rd = PC+4; PC = rs1 + imm	
ecall	Environment Call	1110011	0x0	0x0	Transfer control to OS	
ebreak	Environment Break	1110011	0x0	0x1	Transfer control to debugger	

12



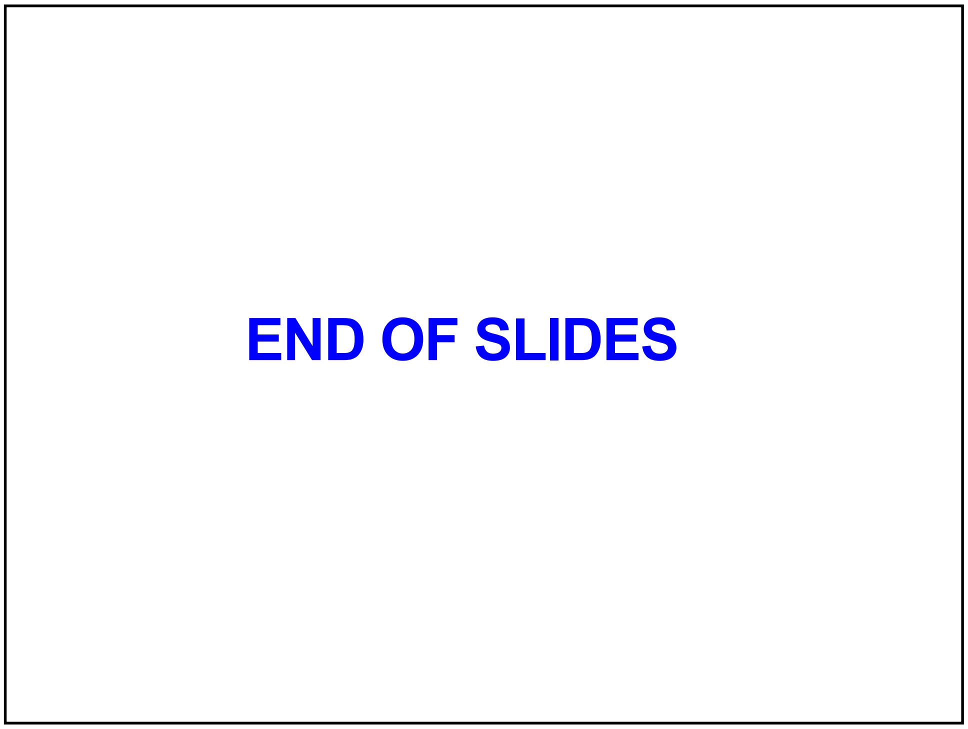
Control Transfer - B-type instructions

Branch Instructions

31	30 25	24 20	19 15	14 12	11 8	7	6 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7

Inst	Name	Opcode	funct3	Description	Note
beq	Branch ==	1100011	0x0	if(rs1 == rs2) PC += imm	
bne	Branch !=	1100011	0x1	if(rs1 != rs2) PC += imm	
blt	Branch <	1100011	0x4	if(rs1 < rs2) PC += imm	
bge	Branch ≥	1100011	0x5	if(rs1 ≥ rs2) PC += imm	
bltu	Branch < (U)	1100011	0x6	if(rs1 < rs2) PC += imm	zero extends
bgeu	Branch ≥ (U)	1100011	0x7	if(rs1 ≥ rs2) PC += imm	zero extends

13



END OF SLIDES

14