Encoding of instructions in RISC-V (U/J)

Sparsh Mittal

Courtesy for some slides: Dr Garcia and Dr Nikolic

U-format instruction encoding

jal	Jump And Link	J	1101111			rd = PC+4; PC += imm
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm
lui	Load Upper Imm	U	0110111			rd = imm << 12
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)

How to handle long jump

Idea: Convert branch instruction to jump instruction

Original code:

```
beq x10,x0, far
# next instruction
```

Challenge: with branch, immediate can be only 12-bits

Rewritten code:

```
bne x10,x0,next
j far
next: #next instruction
```

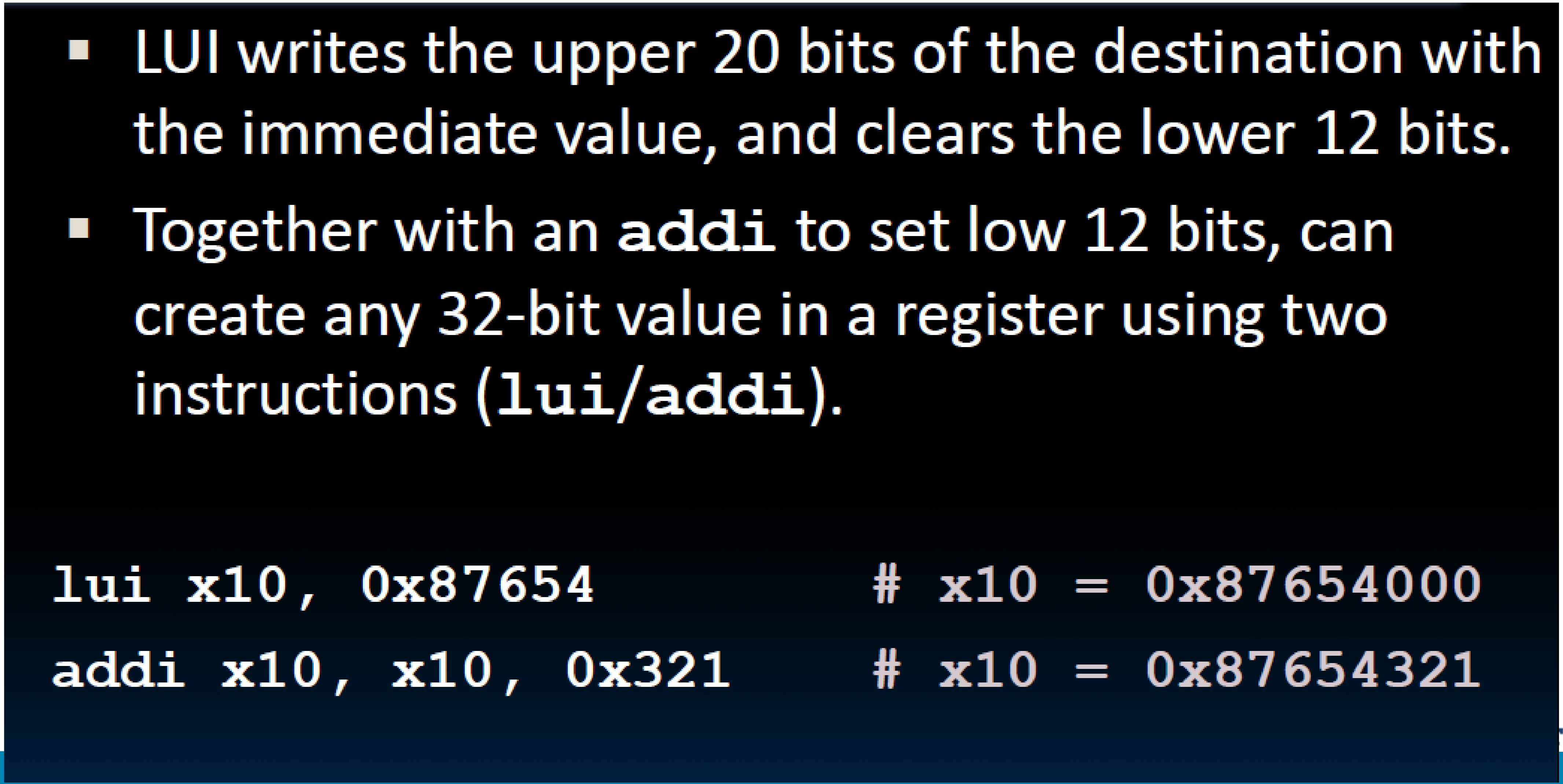
With jump, immediate can be 20-bits

U-format for upper immediate instructions

31 12	11 7	6 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

- Has 20-bit immediate in upper 20 bits of 32-bit instruction word
- One destination register, rd
- Used for two instructions
  - `lui` – Load Upper Immediate
  - `auipc` – Add Upper Immediate to PC

LUI instruction



- LUI writes the upper 20 bits of the destination with the immediate value, and clears the lower 12 bits.
- Together with an **`addi`** to set low 12 bits, can create any 32-bit value in a register using two instructions (**`lui/addi`**).

```
lui x10, 0x87654          # x10 = 0x87654000
addi x10, x10, 0x321      # x10 = 0x87654321
```

AUIPC

- Adds upper immediate value to PC and places result in destination register
- Used for PC-relative addressing

```
Label: AUIPC x10, 0 # Puts address of
                    # Label in x10
```

J-format instruction encoding

jal	Jump And Link	J	1101111			rd = PC+4; PC += imm
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm
lui	Load Upper Imm	U	0110111			rd = imm << 12
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)

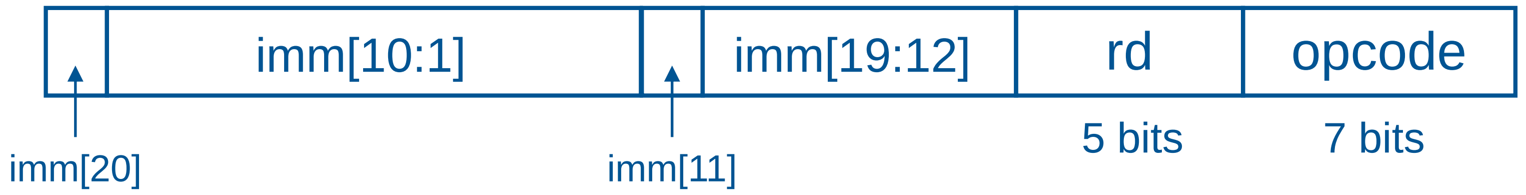
J-format for jump instructions

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
offset[20:1]				dest	JAL

- `jal` saves PC+4 in register `rd` (the return address)
  - Assembler "`j`" jump is pseudo-instruction, uses JAL but sets `rd=x0` to discard return address
- Set PC = PC + offset (PC-relative jump)
- Target somewhere within $\pm 2^{19}$ locations, 2 bytes apart
  - $\pm 2^{18}$ 32-bit instructions
- Immediate encoding optimized similarly to branch instruction to reduce hardware cost

Jump Addressing

- Jump and link (`jal`) target uses 20-bit immediate for larger range
- J format:



- For long jumps, eg, to 32-bit absolute address
  - lui: load address[31:12] to temp register
  - jalr: add address[11:0] and jump to target

jal	Jump And Link	J	1101111			rd = PC+4; PC += imm
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm
lui	Load Upper Imm	U	0110111			rd = imm << 12
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)

Example instruction	Instruction name	Meaning
jal x1,offset	Jump and link	Regs[x1]←PC+4; PC←PC + (offset<<1)
jalr x1,x2,offset	Jump and link register	Regs[x1]←PC+4; PC←Regs[x2]+offset

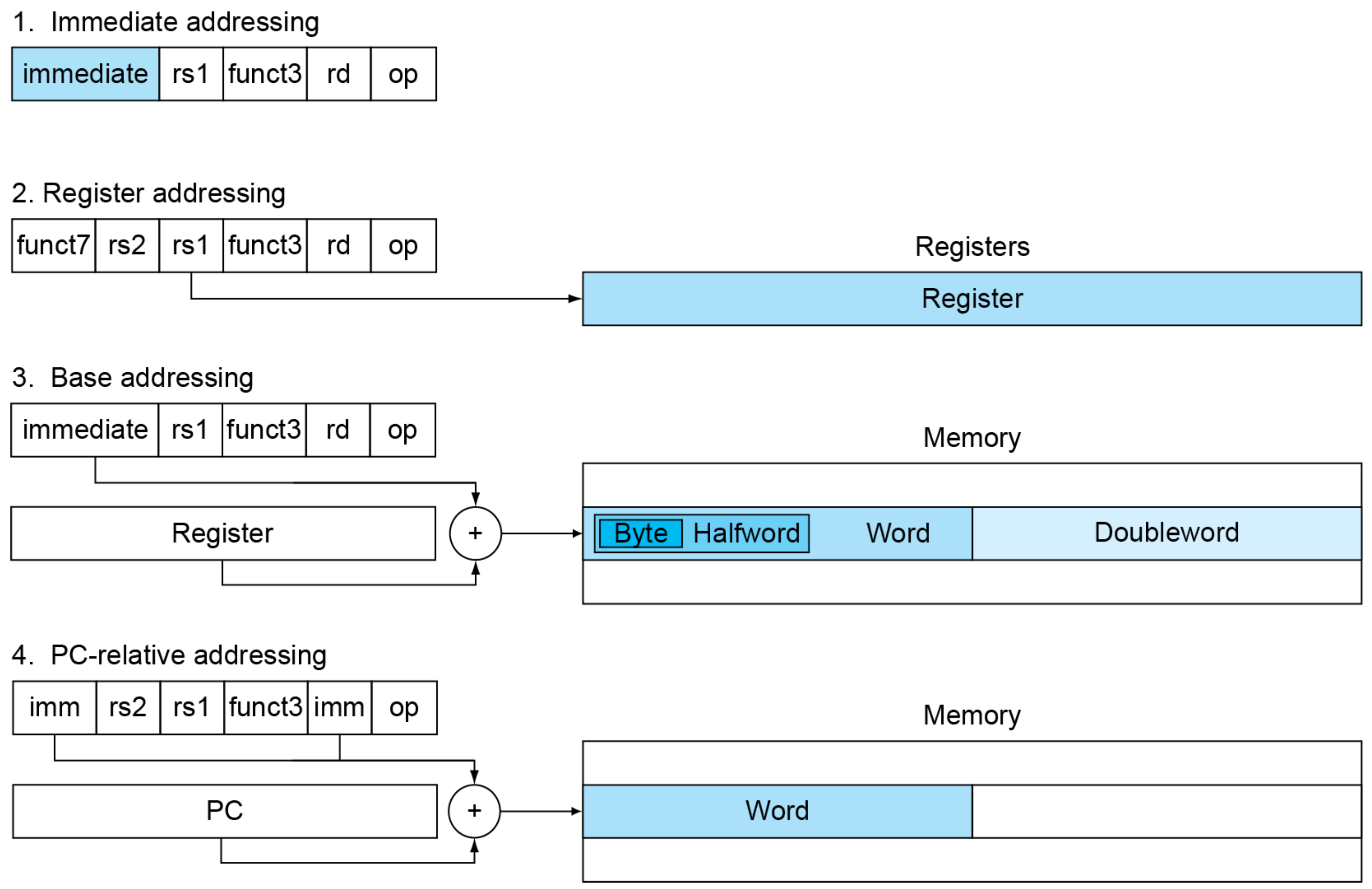
31 20	19 15	14 12	11 7	6 0
imm[11:0]	rs1	func3	rd	opcode
12	5	3	5	7
offset[11:0]	base	0	dest	JALR

- `jalr rd, rs, immediate`
  - Writes PC+4 to rd (return address)
  - Sets PC = `rs + immediate`
  - Uses same immediates as arithmetic and loads
    - *no* multiplication by 2 bytes
    - In contrast to branches and `jal`

Uses of JALR

```
# ret and jr psuedo-instructions
ret = jr ra = jalr x0, ra, 0
# Call function at any 32-bit absolute
address
lui x1, <hi20bits>
jalr ra, x1, <lo12bits>
# Jump PC-relative with 32-bit offset
auipc x1, <hi20bits>
jalr x0, x1, <lo12bits>
```

RISC-V Addressing Summary



RISC-V Encoding Summary

Name	Field						Comments
(Field Size)	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

- https://web.eecs.utk.edu/~smarz1/courses/ece356/notes/assembly/