

SimpleRISC Encoding

**Slide courtesy: Smruti
Ranjan Sarangi**

Slides adapted by: Dr Sparsh Mittal

Encoding Instructions

- * Encode the SimpleRisc ISA using 32 bits.
- * We have 21 instructions. Let us allot each instruction an unique code (opcode)

Instruction	Code	Instruction	Code	Instruction	Code
add	00000	not	01000	beq	10000
sub	00001	mov	01001	bgt	10001
mul	00010	lsl	01010	b	10010
div	00011	lsr	01011	call	10011
mod	00100	asr	01100	ret	10100
cmp	00101	nop	01101		
and	00110	ld	01110		
or	00111	st	01111		

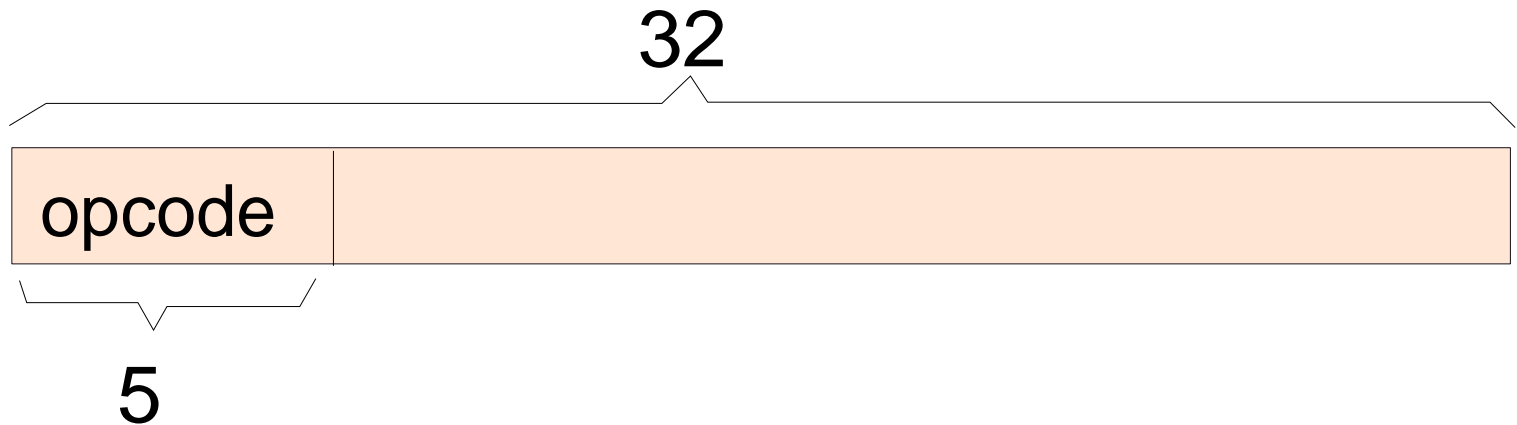
Basic Instruction Format



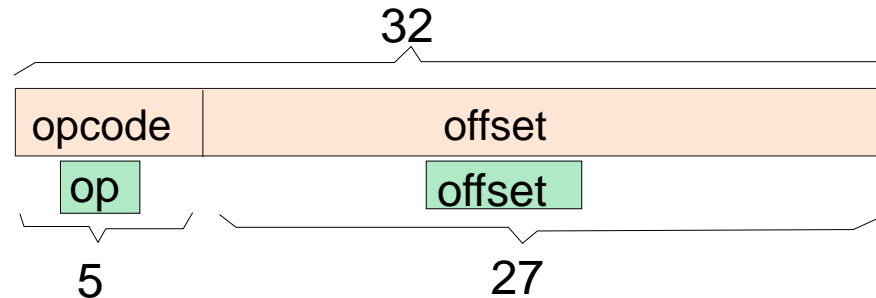
Inst.	Code	Format	Inst.	Code	Format
add	00000	add rd,rs1, (rs2/imm)	lsl	01010	lsl rd, rs1, (rs2/imm)
sub	00001	sub rd,rs1, (rs2/imm)	lsr	01011	lsr rd, rs1, (rs2/imm)
mul	00010	mul rd,rs1, (rs2/imm)	asr	01100	asr rd, rs1, (rs2/imm)
div	00011	div rd,rs1, (rs2/imm)	nop	01101	nop
mod	00100	mod rd,rs1, (rs2/imm)	ld	01110	ld rd,imm[rs1]
cmp	00101	cmp rs1, (rs2/imm)	st	01111	st rd, imm[rs1]
and	00110	and rd, rs1, (rs2/imm)	beq	10000	beq offset
or	00111	or rd,rs1, (rs2/imm)	bgt	10001	bgt offset
not	01000	not rd, (rs2/imm)	b	10010	b offset
mov	01001	mov rd, (rs2/imm)	call	10011	call offset
			ret	10100	ret

0-Address Instructions

* **nop** and **ret** instructions



1-Address Instructions

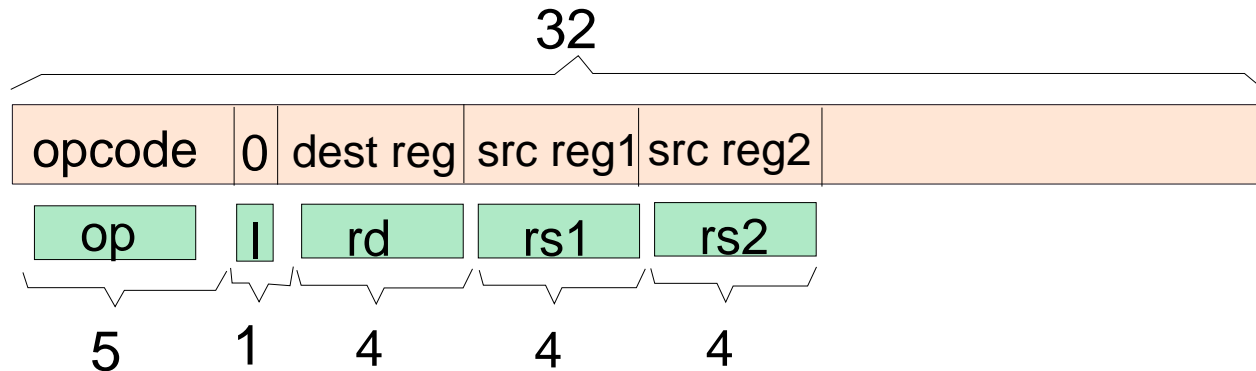


- * Instructions – **call, b, beq, bgt**
- * Use the **branch** format
- * Fields :
 - * 5 bit **opcode**
 - * 27 bit **offset** (PC relative addressing)
 - * Since the offset **points to a 4 byte word address**
 - * The **actual** address computed is : $PC + offset * 4$

3-Address Instructions

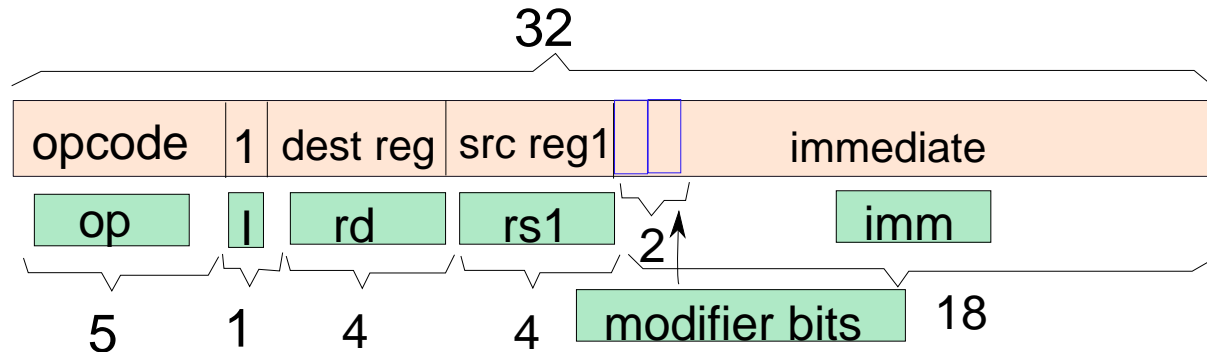
- * Instructions – add, sub, mul, div, mod, and, or, lsl, lsr, asr
- * Generic 3 address instruction
 - * <opcode> rd, rs1, <rs2/imm>
- * Let us use the I bit to specify if the second operand is an immediate or a register.
 - * I = 0 → second operand is a register
 - * I = 1 → second operand is an immediate
- * Since we have 16 registers, we need 4 bits to specify a register

Register Format



- * **opcode** → type of the instruction
- * **I** bit → 0 (second operand is a register)
- * **dest reg** → rd
- * **source register 1** → rs1
- * **source register 2** → rs2

Immediate Format



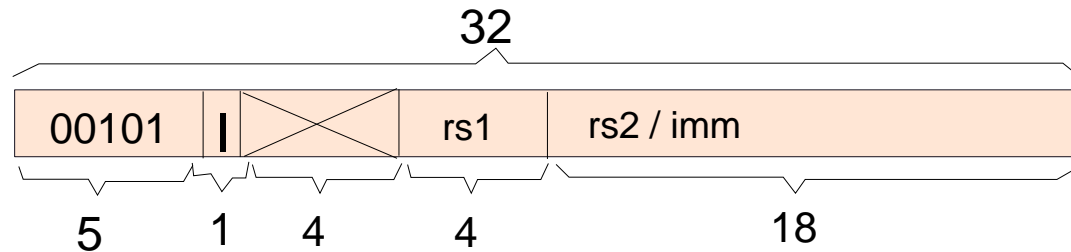
- * **opcode** → type of the instruction
- * **I** bit → 1 (second operand is an immediate)
- * **dest reg** → rd
- * **source register 1** → rs1
- * **Immediate** → imm
- * **modifier** bits → 00 (**default**), 01 (**u**), 10 (**h**)

2 Address Instructions

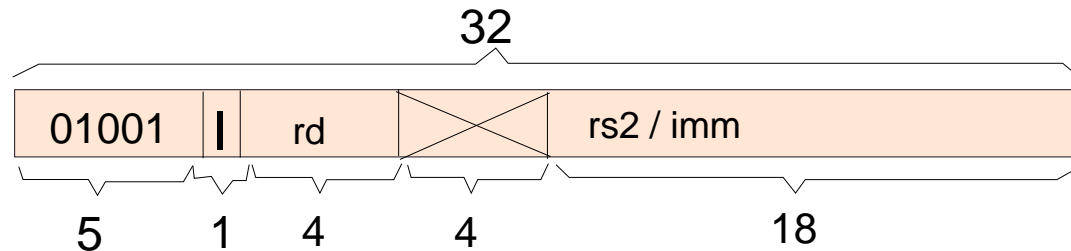
- * `cmp`, `not`, and `mov`
- * Use the 3 address : immediate or register formats
- * Do not use one of the fields

cmp, not, and mov

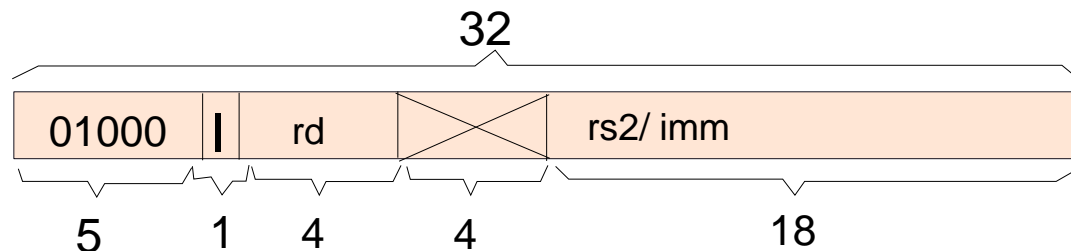
cmp



mov

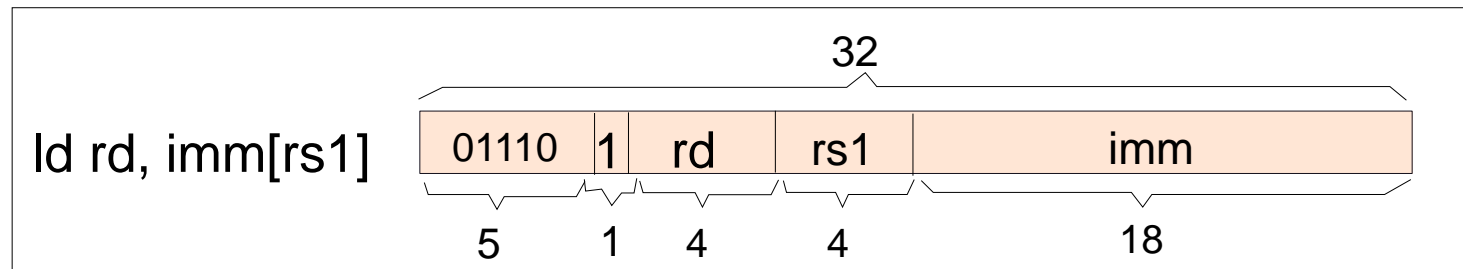


not



Load and Store Instructions

- * `ld rd, imm[rs1]`
- * `rs1` → base register
- * Use the **immediate** format.

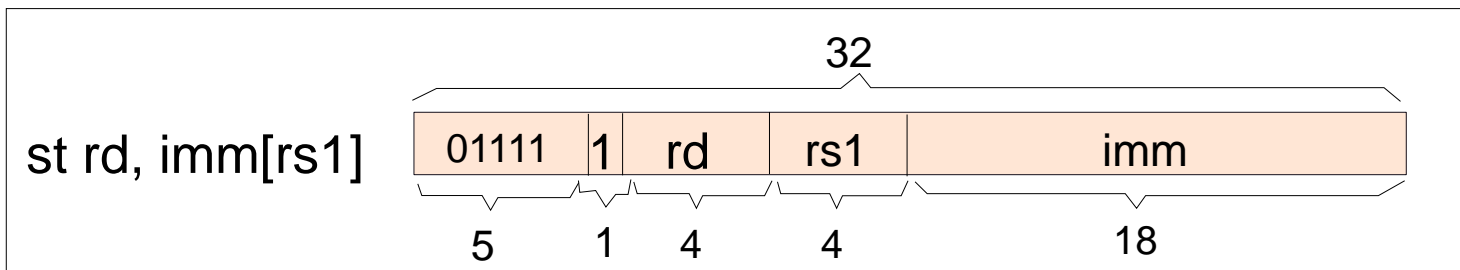


Store Instruction

- * **Strange case** of the store inst.
- * `st reg1, imm[reg2]`
- * has two register **sources**, no register **destination**, 1 **immediate**
- * Should we define a new format for store instructions ?

Store Instruction

- * Let us **make an exception** and use the **immediate format**.
- * We use the **rd field** to save one of the **source registers**
- * **st rd, imm[rs1]**



Summary of Instruction Formats

Format	Definition					
<i>branch</i>	<i>op</i> (28-32)	<i>offset</i> (1-27)				
<i>register</i>	<i>op</i> (28-32)	<i>I</i> (27)	<i>rd</i> (23-26)	<i>rs 1</i> (19-22)	<i>rs 2</i> (15-18)	
<i>immediate</i>	<i>op</i> (28-32)	<i>I</i> (27)	<i>rd</i> (23-26)	<i>rs 1</i> (19-22)	<i>imm</i> (1-18)	
<i>op</i> → opcode, <i>offset</i> → branch offset, <i>I</i> → immediate bit, <i>rd</i> → destination register						
<i>rs1</i> → source register 1, <i>rs2</i> → source register 2, <i>imm</i> → immediate operand						

- * **branch format** → nop, ret, call, b, beq, bgt
- * **register format** → ALU instructions
- * **immediate format** → ALU, ld/st instructions