RISC V Architecture

Mahesh Awati
Department of Electronics and Communication Engg.

RISC V ARCHITECTURE

UNIT 2 – Instructions: The Language of Computer

Mahesh Awati
Department of Electronics and Communication Engineering

Instructions – Language of Computer

Representing Instructions in the Computer

how do we represent instructions?

We know that Computer only understands 1s and 0s.

- A assembler string like “add x10,x11,x0” is meaningless to hardware as hardware can understand only machine code.
  - RISC-V seeks simplicity: since data is in words, make instructions of same 32 bit words. As **RISC-V seeks simplicity, so define six basic types of instruction formats:**
    - **R-format** for register-register arithmetic operations
    - **I-format** for register-immediate arithmetic operations and loads
    - **S-format** for stores
    - **B-format** for branches (minor variant of S-format)
    - **U-format** for 20-bit upper immediate instructions
    - **J-format** for jumps (minor variant of U-format)
- **All these Instruction formats use 32 bit Instruction word and 32 bit word is divided into “fields”**
- Each field tells processor something about instruction.

Machine Code/Language : Binary representation used for communication within a computer system.

Instruction format: A form of representation of an instruction **composed of fields of binary numbers.**

Instructions – Language of Computer

Representing Instructions in the Computer

RISC-V – It's Instruction Format

32-bit RISC-V Instruction Formats

Instruction Formats	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register/register	funct7							rs2					rs1					funct3			rd					opcode						
Immediate	imm[11:0]												rs1					funct3			rd					opcode						
Upper Immediate	imm[31:12]																				rd					opcode						
Store	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
Branch	[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				[11]	opcode						
Jump	[20]	imm[10:1]										[11]	imm[19:12]								rd					opcode						

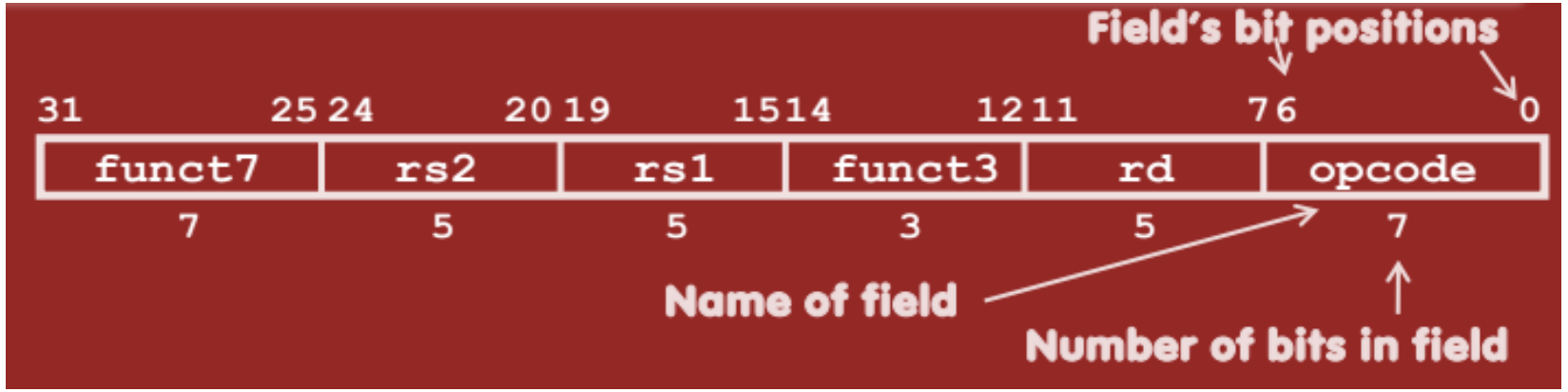
- **opcode (7 bit):** partially specifies which of the 6 types of *instruction formats*
- **funct7 + funct3 (10 bit):** combined with **opcode**, these two fields describe what operation to perform
- **rs1 (5 bit):** specifies register containing first operand
- **rs2 (5 bit):** specifies second register operand
- **rd (5 bit):**: Destination register specifies register which will receive result of computation

Instructions – Language of Computer

Representing Instructions in the Computer

R-Format Instruction Layout:

Syntax: mnemonics rd,rs1,rs2



Opcode: Basic operation of the instruction like I,r,s,u and so on,

Note: This field is equal to **0110011_2 for all R-Format** register-register arithmetic instructions

funct7 + funct3: combined with opcode, these two fields describe what operation to perform

rs1 (Source Register #1): specifies register containing first operand

rs2 (Source Register #2): specifies register containing Second operand

rd(Destination Register):specifies register which will receive result of computation

Each register field holds a 5-bit unsigned integer (0-31) corresponding to a register number (x0-x31)

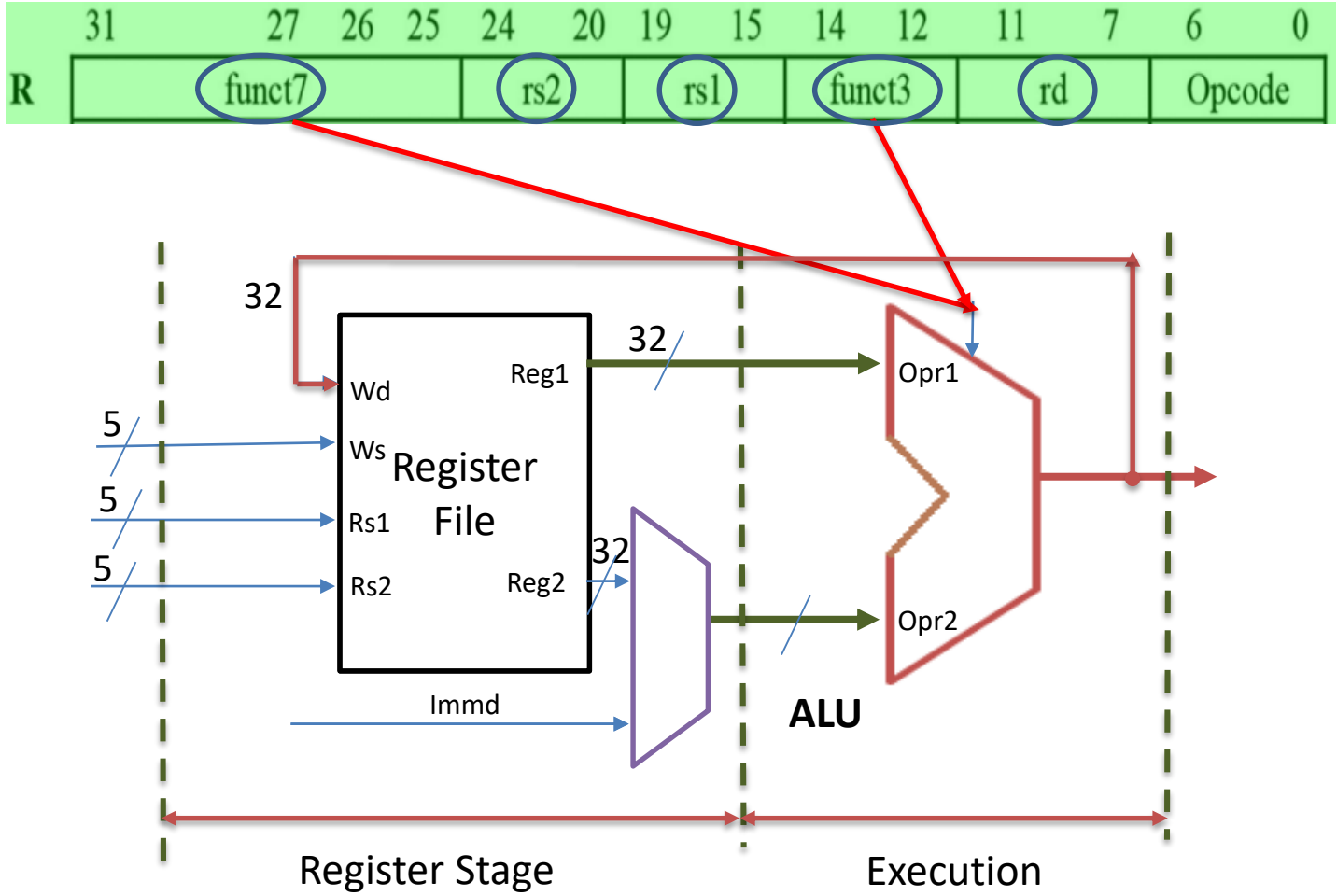
Question: why aren't opcode, funct7 and funct3 a single 17-bit field?

Register	b4 b3 b2 b1 b0
r0	00000
r1	00001
r2	00010
r30	11110
r31	11111

Instructions – Language of Computer

R-type Instruction Data Path

Anything that stores data or operates on data within a processor is called data path.



- **funct7+ funct3 (10):** combined with opcode, these two fields describe what operation to perform
- How many R-format instructions can we encode?
- with opcode fixed at 0b0110011, just funct varies: (2^7) x (2^3)= (2^{10}) = 1024.
- **rs1 (5):** 1st operand (“source register 1”)
- **rs2 (5):** 2nd operand (second source register)
- **rd (5):** “destination register” — receives the result of computation

Instructions – Language of Computer

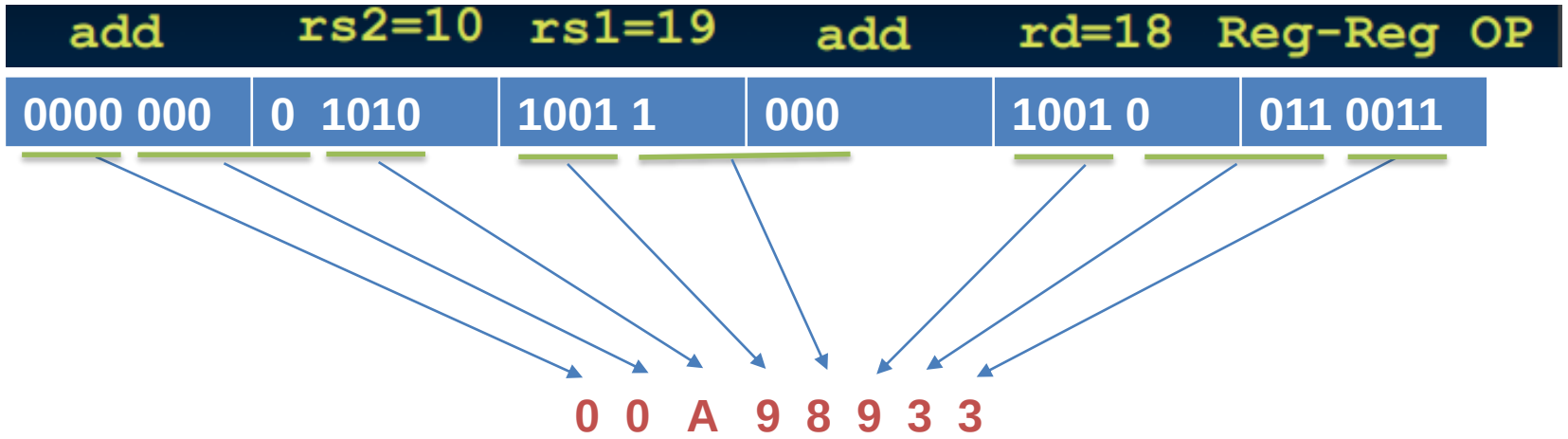
Representing Instructions in the Computer

How R-Format Instruction are Encoded ?

Syntax: mnemonics rd,rs1,rs2

add x18,x19,x10

Instruction	Format	funct7	rs2	rs1	funct3	rd	opcode
add (add)	R	0000000	reg	reg	000	reg	0110011
sub (sub)	R	0100000	reg	reg	000	reg	0110011



Hexadecimal Representation: 0x00A9 8933

Register	b4b3b2b1b0
r0	00000
r1	00001
r2	00010
	.....
r10	01010
	.....
r18	10010
r19	10011
	.....
r31	11111

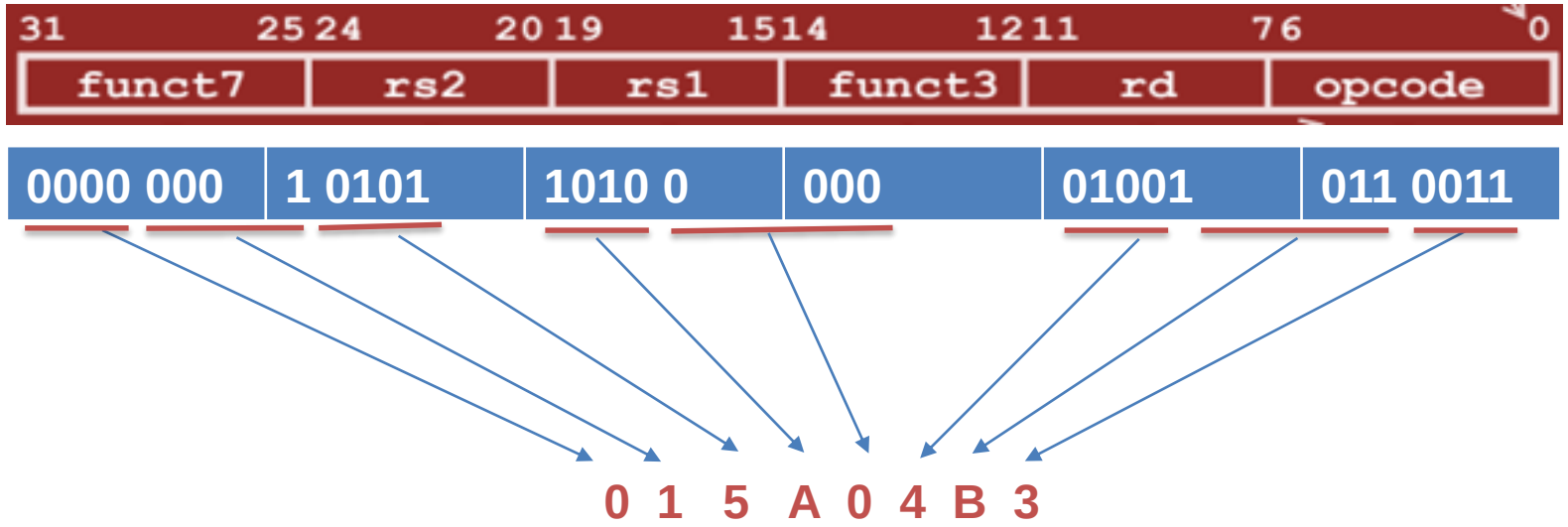
Instructions – Language of Computer

Representing Instructions in the Computer

How R-Format Instruction are Encoded ?

Syntax: mnemonics rd,rs1,rs2

add x9, x20, x21



Hexadecimal Representation: 0x015A 04B3

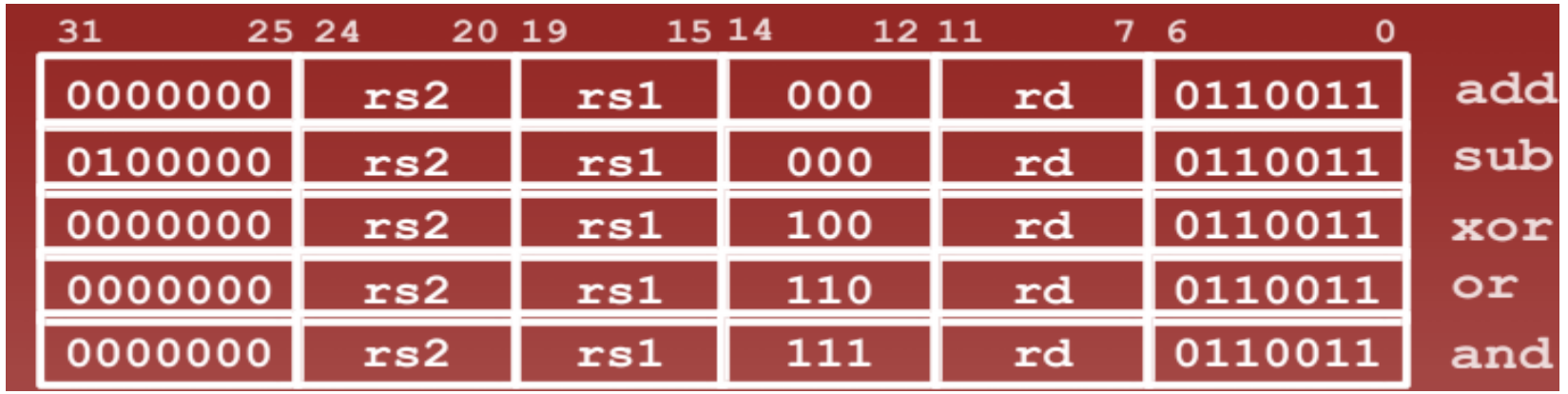
Register	b4b3b2b1b0
r0	00000
r1	00001
r2	00010
	.....
r10	01010
	.....
r18	10010
r19	10011
	.....
r31	11111

Instructions – Language of Computer

Representing Instructions in the Computer

How R-Format Instruction are Encoded ?

Syntax: mnemonics rd,rs1,rs2



31 25	24 20	19 15	14 12	11 7	6 0	
0000000	rs2	rs1	000	rd	0110011	add
0100000	rs2	rs1	000	rd	0110011	sub
0000000	rs2	rs1	100	rd	0110011	xor
0000000	rs2	rs1	110	rd	0110011	or
0000000	rs2	rs1	111	rd	0110011	and

What is correct encoding of add x4, x3, x2 ?

1) 0x4021 8233
2) 0x0021 82b3
3) 0x4021 82b3
4) 0x0021 8233
5) 0x0021 8234

0000 000	0 0010	0001 1	000	0010 0	011 0011

Hexadecimal Representation: 0x0021 8233

Register	b4b3b2b1b0
r0	00000
r1	00001
r2	00010
	.....
r10	01010
	.....
r18	10010
r19	10011
	.....
r31	11111

THANK YOU

Mahesh Awati
Department of Electronics and Communication

mahesha@pes.edu

+91 9741172822