Instruction Encoding & Single-Cycle CPU (Slides T4-T5)

VE370SU22 TA Runxi Wang

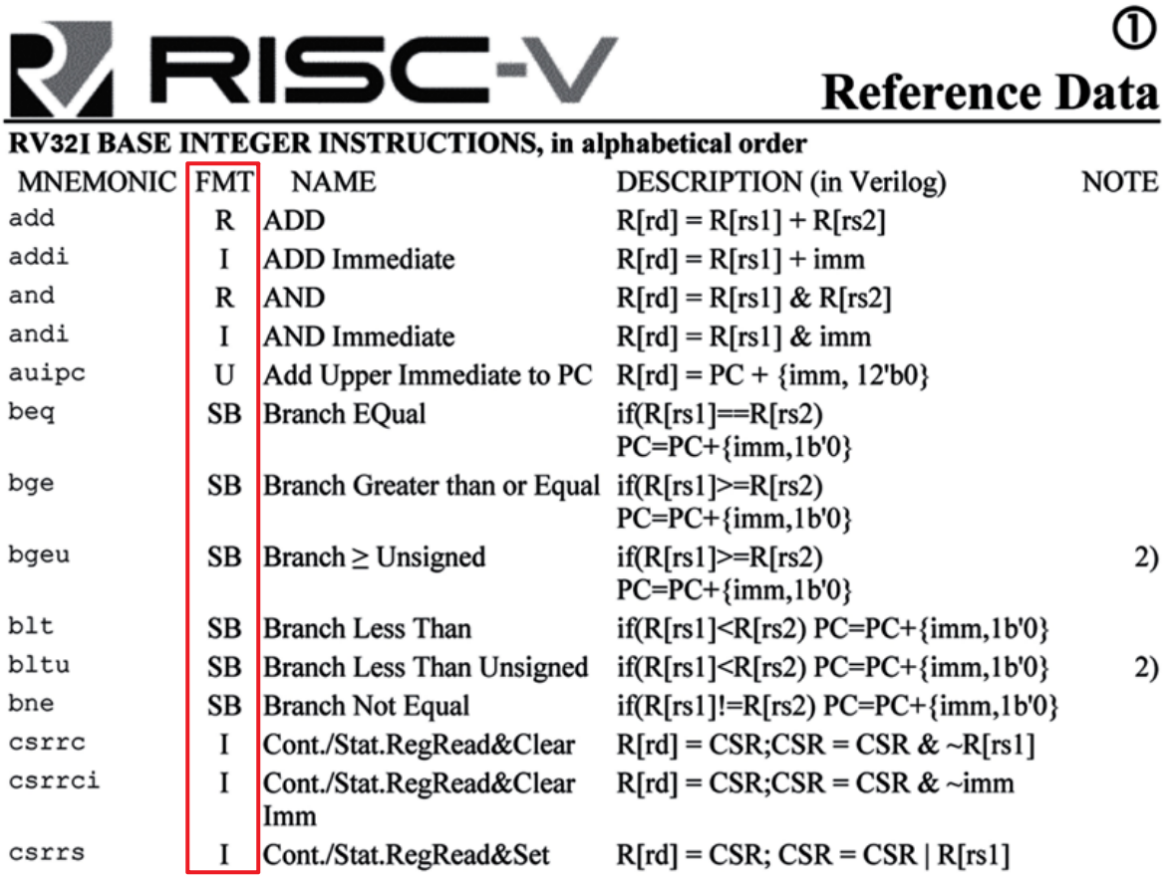
Instructions in 32-bit RISC-V

- All 32 bits long
- Machine code
- including all information of a line of instruction (e.g. add x5, x6, x7)

From assembly to Machine Code

TIPS

- Make use of "RISC-V Reference Data"
- Check instruction type



RISC-V ①

Reference Data

RV32I BASE INTEGER INSTRUCTIONS, in alphabetical order

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)	NOTE
add	R	ADD	R[rd] = R[rs1] + R[rs2]	
addi	I	ADD Immediate	R[rd] = R[rs1] + imm	
and	R	AND	R[rd] = R[rs1] & R[rs2]	
andi	I	AND Immediate	R[rd] = R[rs1] & imm	
auipc	U	Add Upper Immediate to PC	R[rd] = PC + {imm, 12'b0}	
beq	SB	Branch EQual	if(R[rs1]==R[rs2) PC=PC+{imm,1b'0}	
bge	SB	Branch Greater than or Equal	if(R[rs1]>=R[rs2) PC=PC+{imm,1b'0}	
bgeu	SB	Branch ≥ Unsigned	if(R[rs1]>=R[rs2) PC=PC+{imm,1b'0}	2)
blt	SB	Branch Less Than	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	
bltu	SB	Branch Less Than Unsigned	if(R[rs1]<R[rs2) PC=PC+{imm,1b'0}	2)
bne	SB	Branch Not Equal	if(R[rs1]!=R[rs2) PC=PC+{imm,1b'0}	
csrrc	I	Cont./Stat.RegRead&Clear	R[rd] = CSR;CSR = CSR & ~R[rs1]	
csrrci	I	Cont./Stat.RegRead&Clear Imm	R[rd] = CSR;CSR = CSR & ~imm	
csrrs	I	Cont./Stat.RegRead&Set	R[rd] = CSR; CSR = CSR | R[rs1]	

- Check the formats of the corresponding type you found

CORE INSTRUCTION FORMATS

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	Opcode
I	imm[11:0]			rs1	funct3	rd	Opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

- Check opcode, funct3 (if available), and funct7 (if available)

OPCODES IN NUMERICAL ORDER BY OPCODE

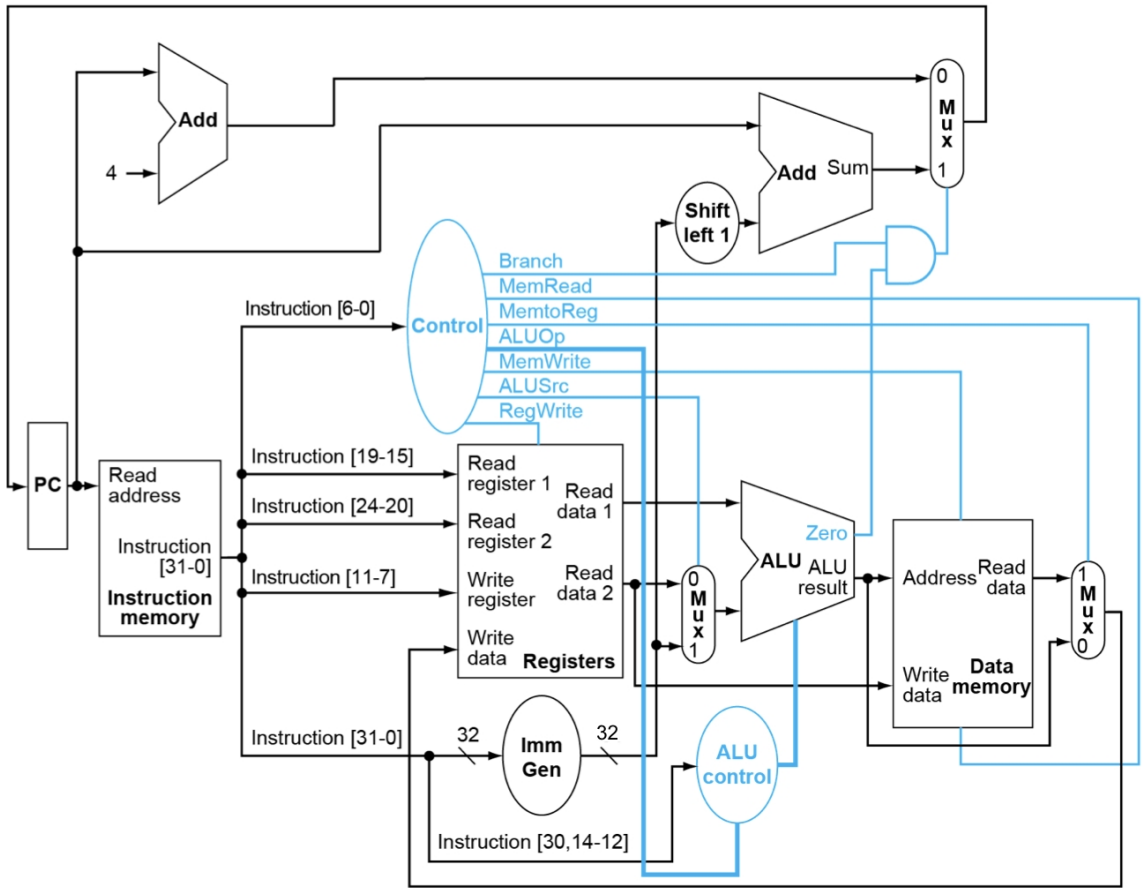
MNEMONIC	FMT	OPCODE	FUNCT3	FUNCT7 OR IMM	HEXADECIMAL
lb	I	0000011	000		03/0
lh	I	0000011	001		03/1
lw	I	0000011	010		03/2
lbu	I	0000011	100		03/4
lhu	I	0000011	101		03/5
fence	I	0001111	000		0F/0
fence.i	I	0001111	001		0F/1
addi	I	0010011	000		13/0
slli	I	0010011	001	0000000	13/1/00
slti	I	0010011	010		13/2
sltiu	I	0010011	011		13/3
xori	I	0010011	100		13/4
srli	I	0010011	101	0000000	13/5/00
srai	I	0010011	101	0100000	13/5/20
ori	I	0010011	110		13/6
andi	I	0010011	111		13/7
auipc	U	0010111			17

- Transfer register number to binary and place it to corresponding slots of the instruction format
    - Don't mistake the order of rs1, rs2, and rd
    - e.g. add rd, rs1, rs2
    - e.g. lw rd, imm(rs1)
    - e.g. sw rs2, imm(rs1)
- For immediate, pay attention to different treatments for different types
    - I-type: Just turn the immediate into binary and put it into the corresponding slots
    - B-type: imm = (Branch target label - Current PC) / 2, and notice the special placement of bits of the immediate in B-type format
    - J-type: imm = (Jump target label - Current PC) / 2, and notice the special placement of bits of the immediate in J-type format
    - S-type: pay attention the special placement of bits of the immediate in S-type format
    - U-type: pay attention the special placement of bits of the immediate in U-type format

Why we need to swirl the immediate bits?

To save hardware on the critical path!

Single-Cycle CPU

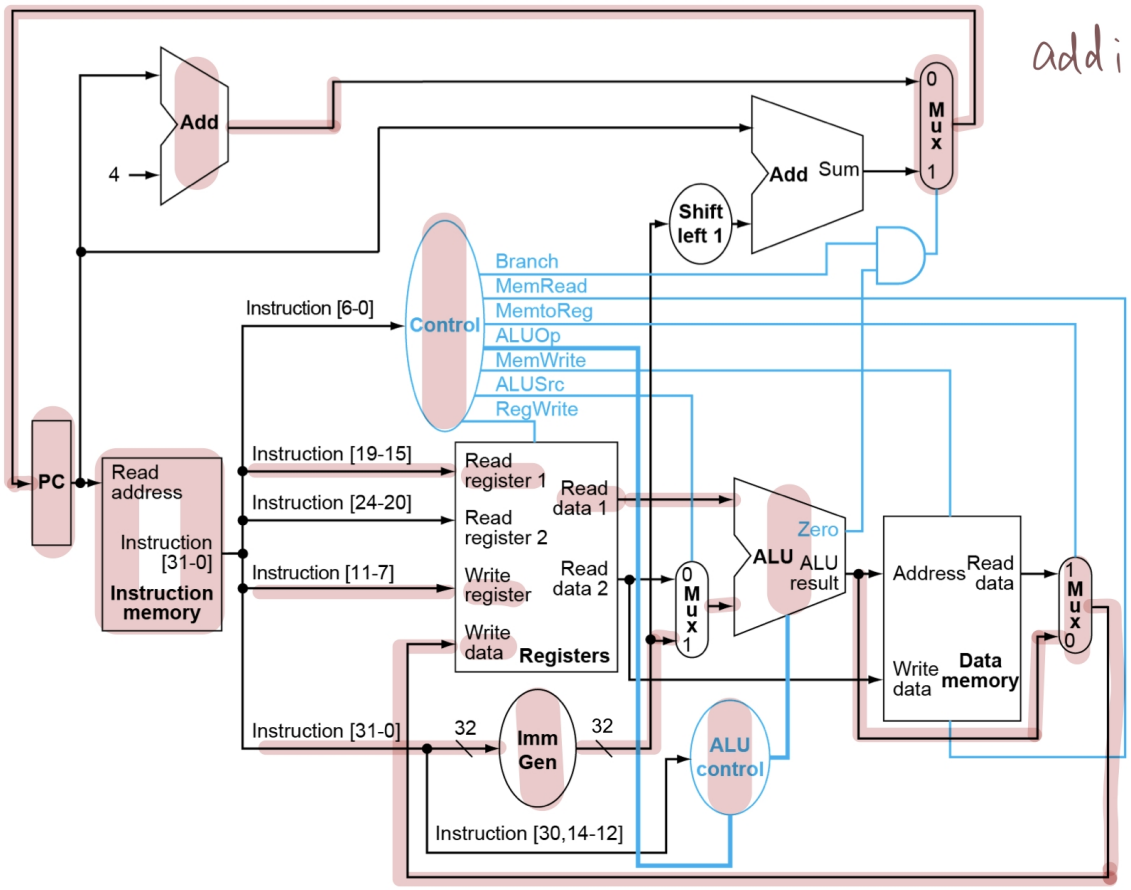


*You have to know how to code every block in the above diagram using Verilog so that you can finish project 3 :)

Block(Resources) Usages

For every (type) of instructions, you have to know exactly which block is used to execute it.

Example. arithmetic immediate operations (use `addi` as an example)



Control Signal Generation

For every (type) of instructions, after you have known exactly which block is used, it is time to assign them corresponding control signals!

Examples.

	Branch	MemRead	MemtoReg	ALUOp	MemWrite	ALUSrc	RegWrite
sub	0	0	0	10	0	0	1
lw	0	1	1	00	0	1	1
beq	1	0	0	01	0	0	0

Final Tips for Project 3

- Read manual and specifications carefully
- Debug a module/block once you finish it
- Be careful about the modules that need clock signal
- Start early :)

Reference

[1] VE370 SU22 Slides T4

[2] VE370 SU22 Slides T5