ECE3700JFA23 RC3

TA: Xu Weiqing

HW1 Review

RISC-V Instructions

1. encoded as 32-bit words
2. machine code (binary)
3.

rs1 / rs2	source register	5 bits	add x9, **x20**, **x21**
rd	destination register	5 bits	add **x9**, x20, x21
Imm	Immediate	? bits	addi x9, x20, **21**
opcode	operation code	7 bits	lb vs. addi
funct3	function code	3 bits	sb vs. sw
funct7	function code	7 bits	add vs. sub

4. 6 formats
   a. **R-type rs1, rs2 → rd**
   b. **I-type other imm**
   c. **S-type store**
   d. **U-type lui, auipc**
   e. (S)**B-type branch**
   f. (U)**J-type jal**
5. Can you remember some examples for each format?

   How to check the format for a specific instruction?

   How to check opcode & funct3 & funct7 for a specific instruction?

R-type

funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7

Note the order of rs1 and rs2

I-type

Immediate[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7

Note that the immediate is sign extended. (Only _____ sees it as unsigned.)

The range is from 0x800~0x7ff, -2048~2047

S-type

Imm[11:5]	rs2	rs1	funct3	Imm[4:0]	opcode
7	5	5	3	5	7

B-type

Imm[11, 9:4]	rs2	rs1	funct3	Imm[3:0, 10]	opcode
7	5	5	3	5	7

Target PC = Current PC + Imm[11:0] * 2 (Bytes)

Range: -2^**12** ~ + 2^**12 (Bytes)**

U-type

Imm[19:0]	rd	opcode
20	5	7

J-type

Imm[19, 9:0, 10, 18:11]	rd	opcode
20	5	7

Target PC = Current PC + Imm[19:0] * 2 (Bytes)

Range: -2^**20** ~ + 2^**20 (Bytes)**

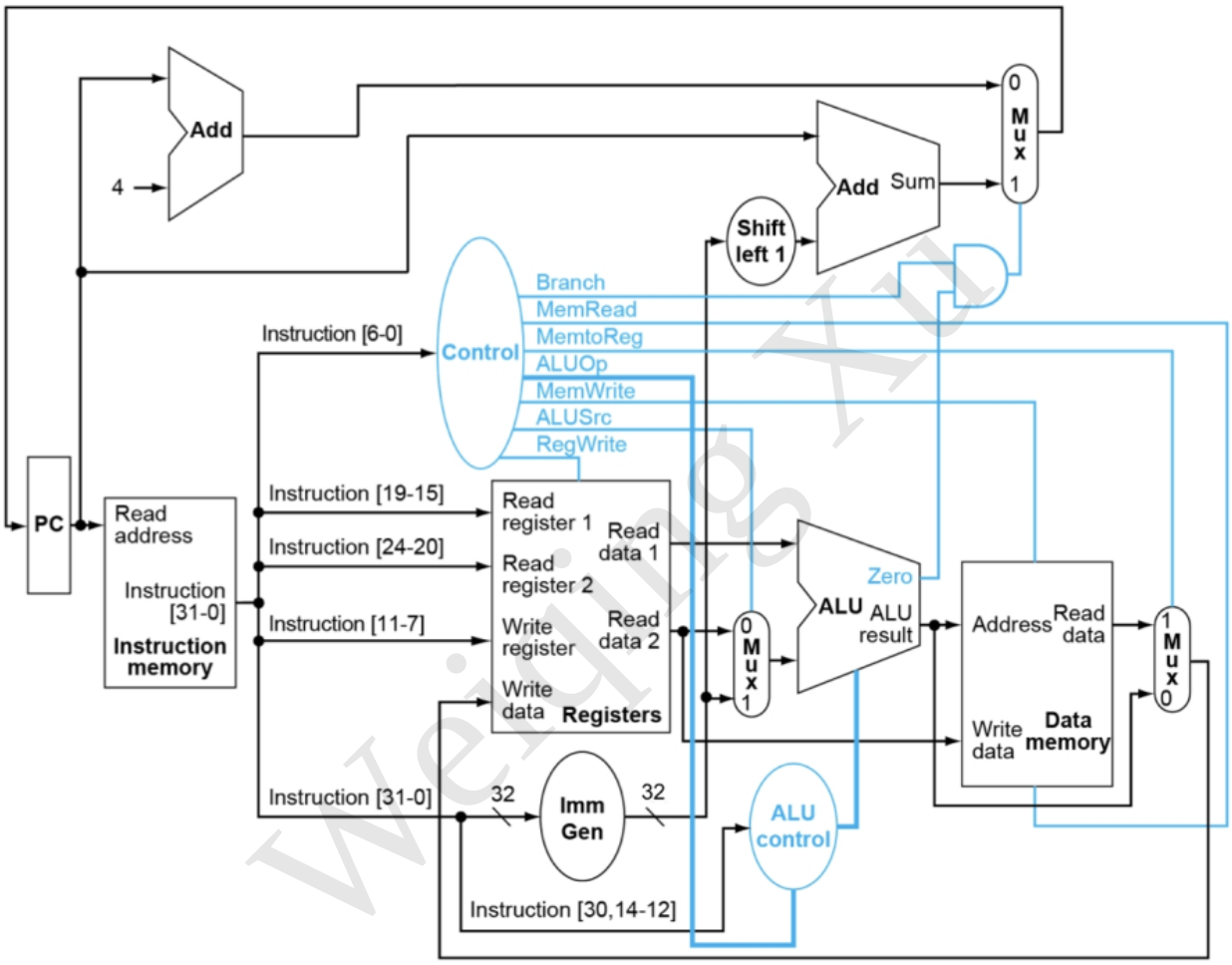
Summary

	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

1. rs1, rs2, rd, opcode, funct3, funt7 are all settled:
    a. rs1: 19~15
    b. rs2: 24~20
    c. rd: 11~7
    d. opcode: 6~0
    e. funct3: 14~12
    f. funct7: 31~25
2. Immediate: fill in blank
    a. I-type & U-type: No swirled imm
    b. Can you find some similarity? e.g. imm[10:5]

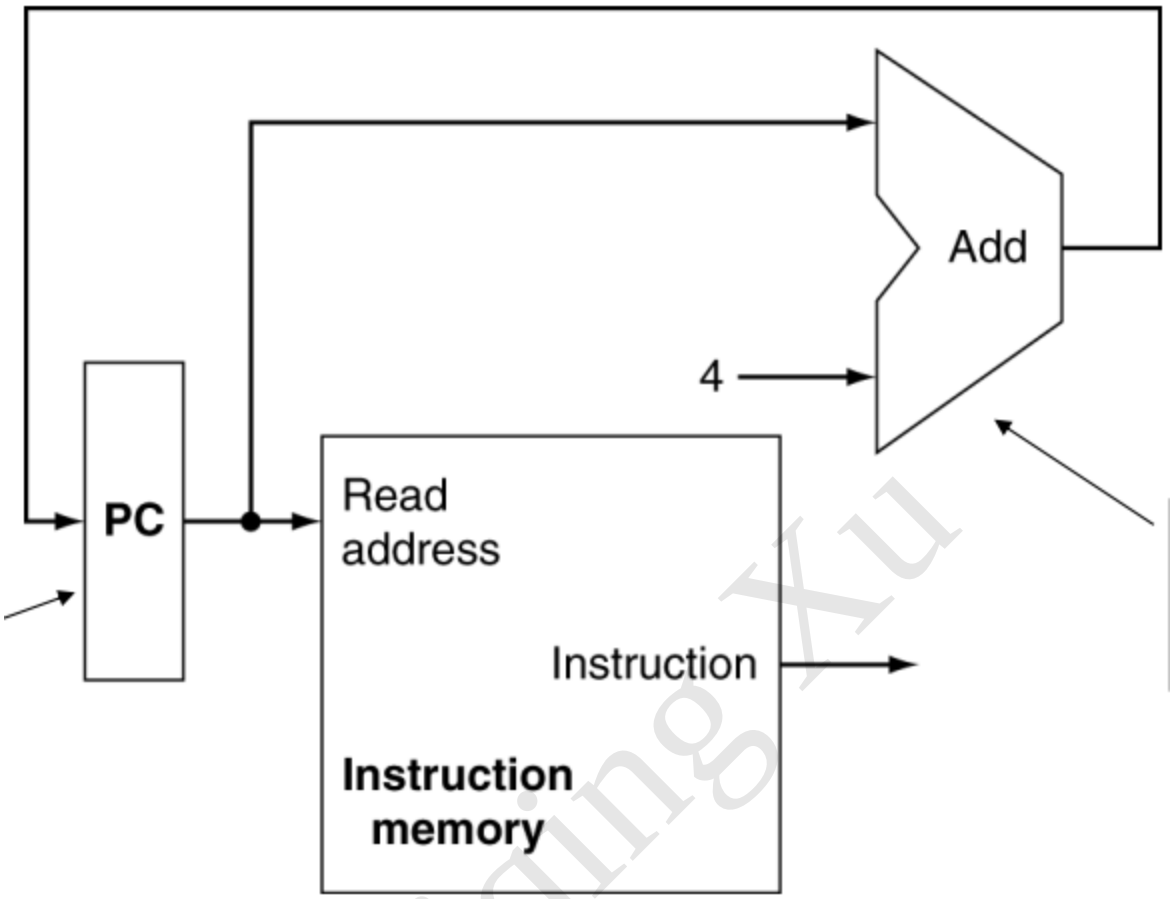
How to read from machine code?

Single-Cycle CPU

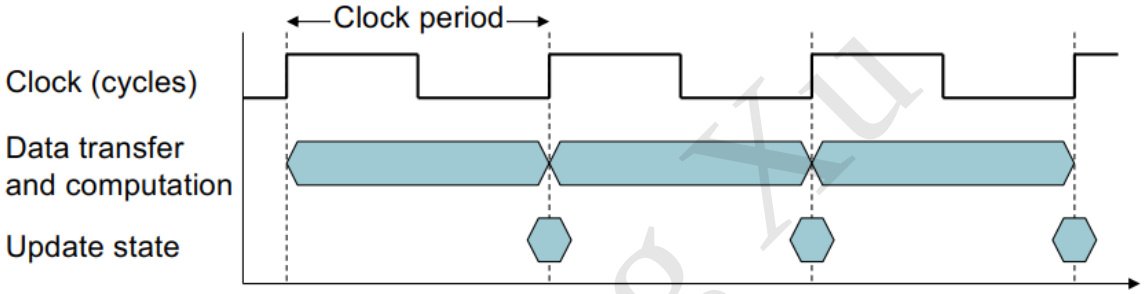


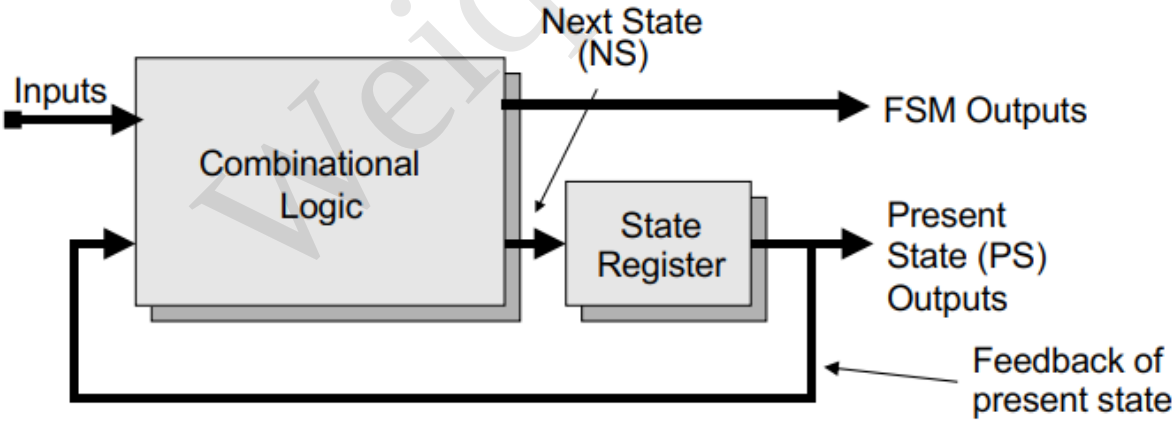
You need to code every thing to finish the coming labs. :(

Start from basic…



1. Remeber some 270 memory…
    a. ALU and mux
    b. clock
    c. combinational logic & sequential logic
    d. conbinational logic & state register in FSM design
    e. controller and datapath in RTL design





2. PC is program memory
3. PC <= PC + 4
4. RF

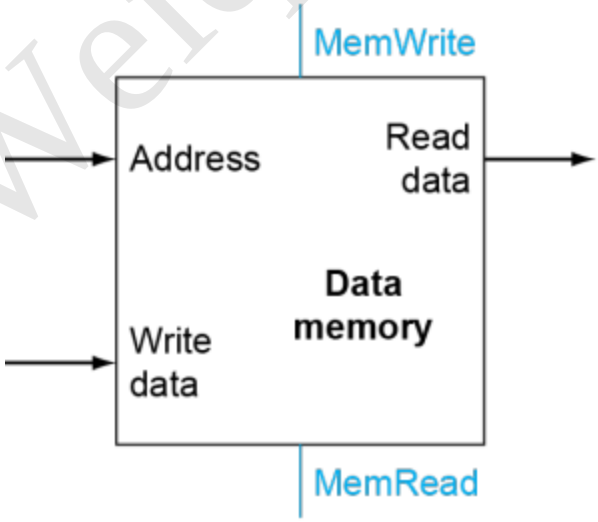
read	write

RegWrite=0	RegWrite=1
Always (provided register numbers)	Time sensitive

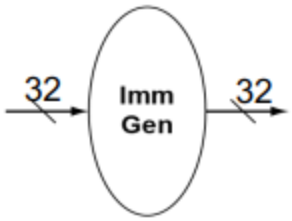
For example, design: write first, read next in one clock cycle

5. Memory

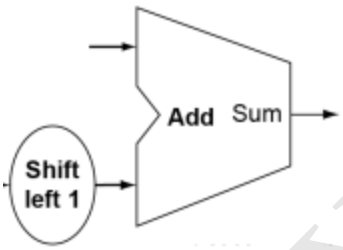
   Read VS. Write. Cannot be 1 at the same time.



6. Immediate Generator



7. Branch



8. Control Unit
    a. Only use **OPCODE**
    b. Can decide:
        i. Branch: whether PC jumps (together with ALU result (only ALUzero))
        ii. MemRead & MemWrite: whether to read & write data in memory
        iii. MemtoReg: select the data to be written to RF, between ALU result and read from memory
        iv. ALUsrc: select the data to be calculated in ALU, between immediate data and read from RF
        v. RegWrite: whether to read & write data in RF
        vi. ALUop

    Can you write examples for each of them? (may help you with your labs)
9. ALU control
    a. Use: Inst[30, 14:12], ALUop
    b. Can decide ALU operation, like add / subtract / and / or …

Performance

1. Save Hardware

   RISC-V Architecture: more complicated instruction format, but simpler hardware, thus better performance

2. Execution (response) time: How long it takes to do a task

   a. Elapsed time: total

   b. CPU time: exclude I/O time

   c. CPU time = CPU clock cycles * Clock cycle time = CPU clock cycles / Clock rate

   d. Clock cycles = Instruction Count (IC) * Cycles per instruction (CPI)

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

3. Related factors

   a. Algorithm: Determines number of operations executed, **IC (CPI)**

   b. Programming language, compiler, architecture: Determine number of machine instructions (lines ofsource code) executed per operation **IC, CPI**

   c. Processor and memory system: Determine how fast instructions are executed **IC, CPI**

   d. I/O system (including OS): Determines how fast I/O operations are executed **IC, CPI, Tc**

   e. Critical data path (combinational logic): **Tc**

4. Single Cycle Processor VS. Pipelined Processor: **CPI**

Exercise & Genral Tips for Lab3

1. Find a design principle in the lecture.
2. Use pipes for machine code translation, read & write
3. Mark which blocks and databus are used to execute a specific instruction, for example, `addi` ?
4. Determine the result of Control Unit for any specific instruction, for example, `sub` , then try to transfer it into your design of its logic.
5. Doing lab3 step by step.
6. Doing homework and review T2-T5 for the midterm.

Now you (will) have done all the develop procedure from and/or logic gates to C codes. Congratulations! But how about performance…

Appendix A

R	funct7	rs2	rs1	funct3	rd	opcode
	7	5	5	3	5	7
I	Imm[11:0]		rs1	funct3	rd	opcode
	12		5	3	5	7
S	Imm[11:5]	rs2	rs1	funct3	Imm[4:0]	opcode
	7	5	5	3	5	7
B	Imm[11, 9:4]	rs2	rs1	funct3	Imm[3:0, 10]	opcode
	7	5	5	3	5	7
U	Imm[19:0]				rd	opcode
	20				5	7
J	Imm[19, 9:0, 10, 18:11]				rd	opcode
	20				5	7

Appendix B

R type: `add` `and` `or` `sll` `slt` `sltu` `sra` `srl` `sub` `subw` `xor`

I-type: `addi` `andi` `jalr` `lb` `lbu` `lw` `lh` `lhu` `ori` `slli` `slti` `sltiu` `srai` `srli` `xori`

S-type: `sb` `sw` `sh`

B-type: `beq` `bge` `blt` `bne` `bgeu` `bltu`

U-type: `lui`

J-type: `jal`

References

ECE3700JFA23 Slides T4 T5