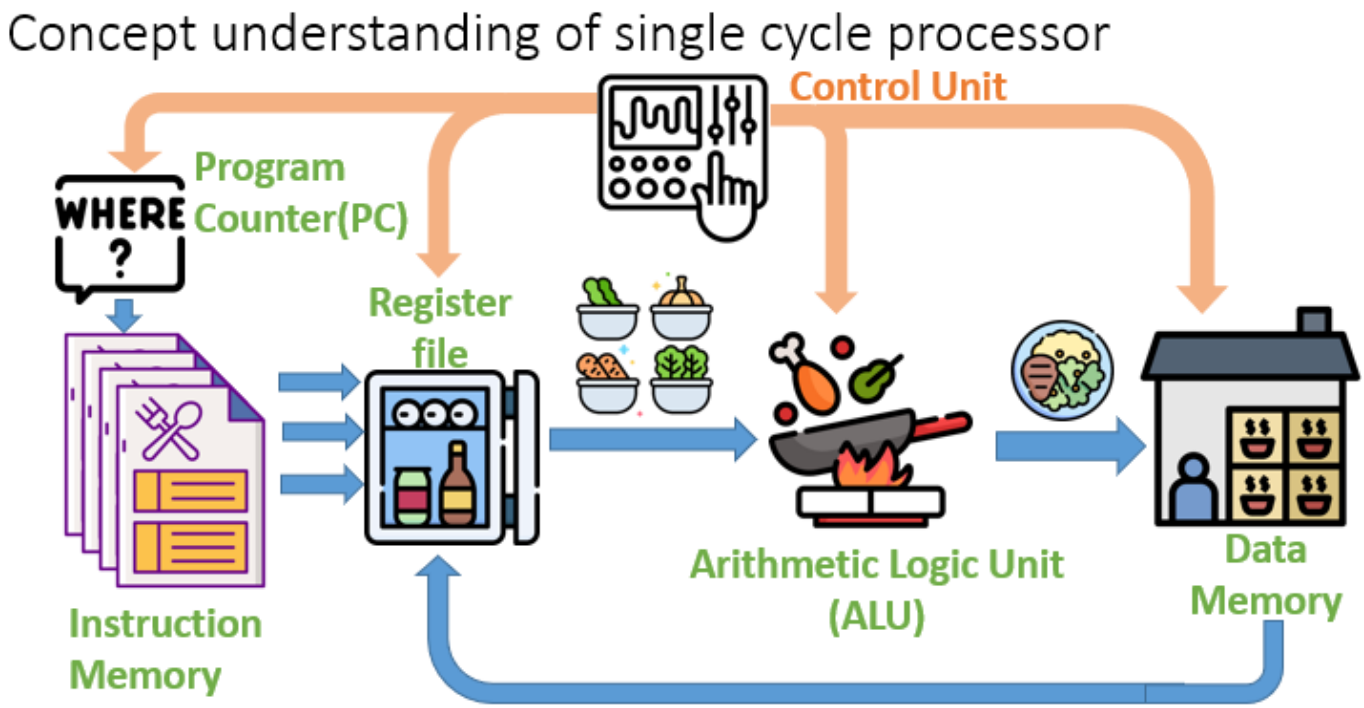
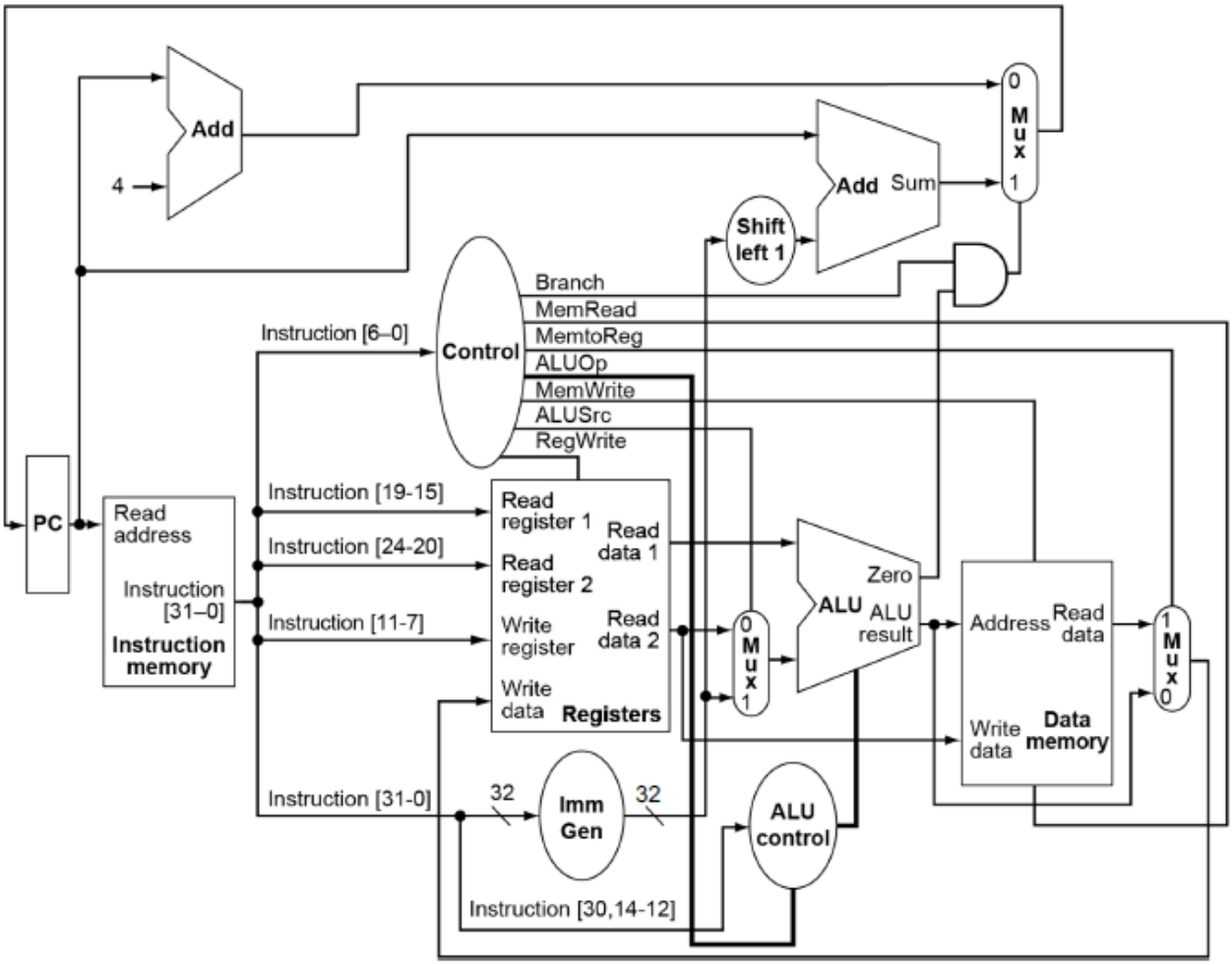
Single Cycle Processor & Pipelined Processor (T5-T6)

VE370FA22 TA Jiajun Gu

Single Cycle Processor

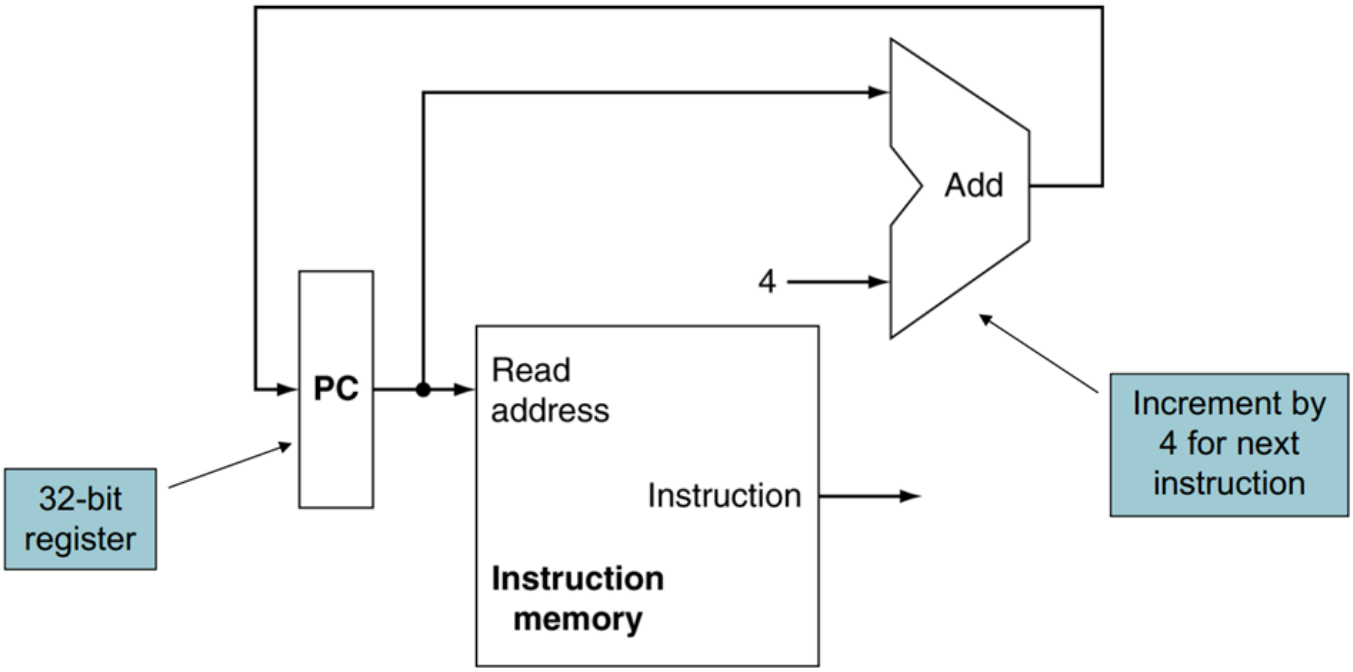


(cited from VE370FA2021 TA Pengcheng Xu)



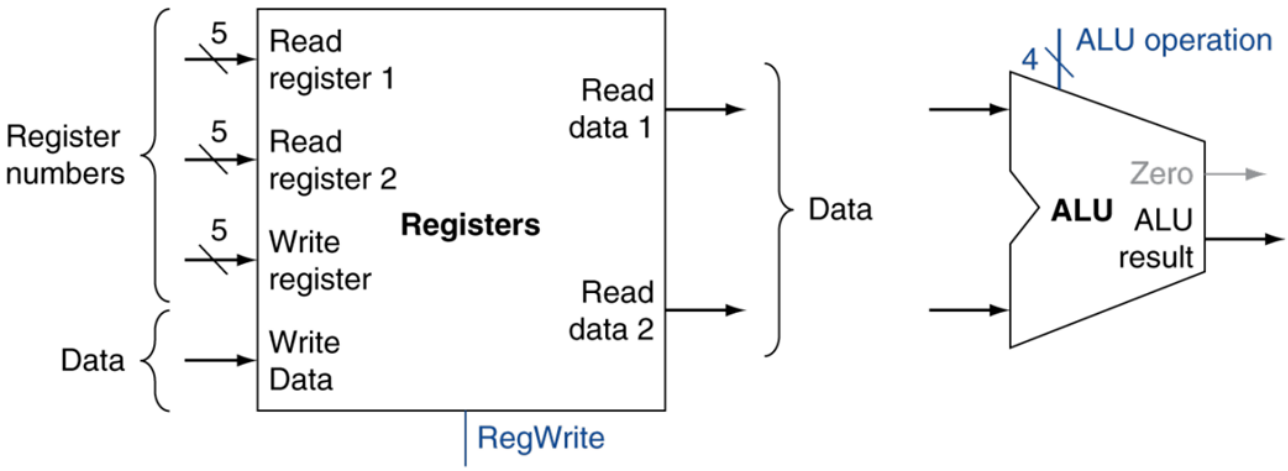
- Datapath:
  - *elements that process, store, or route data in the CPU*: registers, ALU, mux's, memories, ...
- Only **one instruction** can be executed in a clock cycle

Instruction Fetch



R-Format Instructions

- Read two register operands
- Perform arithmetic/logic operation
- Write register result

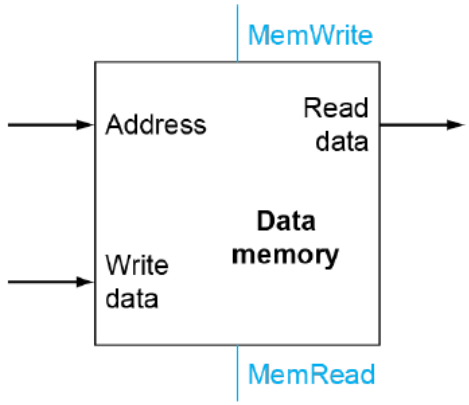


Register Files:

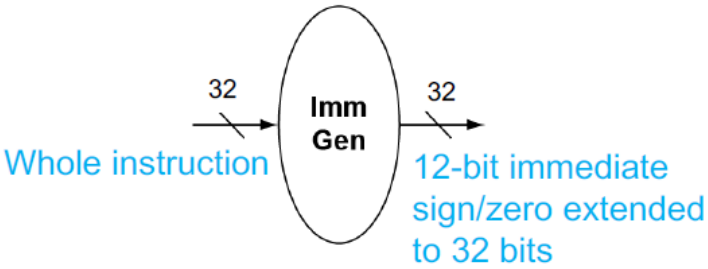
- Data is read out **whenever** register numbers are provided (`assign`)
- Writes:
  - Edge-triggered
  - All the write inputs (Write Data, Write Register, RegWrite) must be valid before the clock edge – setup time

Load/Store Instructions

- Read register operands
- Calculate address using 12-bit immediate
- Load: Read memory and update register
- Store: Write register value to memory



a. Data memory unit

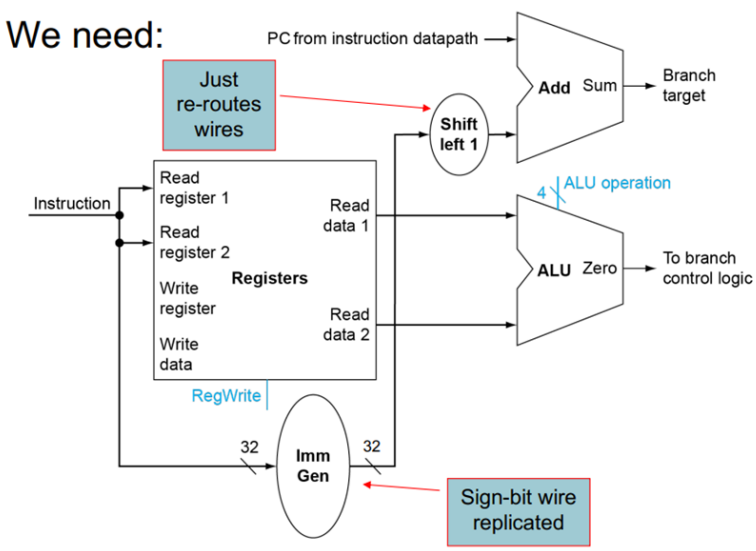


b. Immediate generation unit

Contents are stored into Data Memory `sw x7, 0(x0)`

Beq Instructions

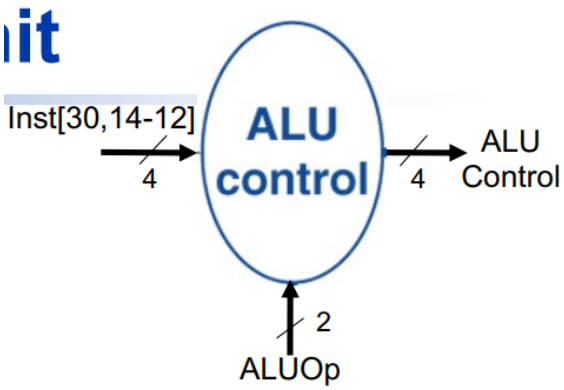
- Read register operands
- Compare operands
  - rs1==rs2 ? ⇒ rs1-rs2==0
  - Use ALU, subtract then check the Zero output of the ALU
- Calculate target address: **Target PC = Current PC + immediate * 2**
  1. Sign-extended the immediate
  2. Shift left 1 place (multiply by 2)
  3. Add to PC value



ALU Control (Unit)

- Load/Store: add
- Branch: substract
- R-type: depends on other control signals (i[30] + funct3)

ALU control Signal	Function
0000	AND
0001	OR
0010	add
0110	subtract



Input:

- ALUOp: 2-bit derived from opcode
- Instruction[30, 14-12] (i[30] + funct3)

Output: 4-bit ALU control signal

Instr.	Operation	ALU function	Opcode field	ALUOp (input)	i[30] (input)	funct3 (input)	ALU control (output)
lw	load register	add	XXXXXXX	00	X	010	0010
sw	store register	add	XXXXXXX	00	X	010	0010
beq	branch on equal	subtract	XXXXXXX	01	X	000	0110
R-type	add	add	100000	10	0	000	0010
	subtract	subtract	100010		1	000	0110
	AND	AND	100100		0	111	0000
	OR	OR	100101		0	110	0001

Additional bit (instruction[30]) is needed for R-type

Type	Field					
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
R-type	funct7	rs2	rs1	funct3	rd	opcode

Control Signals

Make sure you are clear about everything in this table!

Inst.	ALU Src	Reg Dst	ALU Op	Mem Write	Mem Read	Branch	Memto Reg	Reg Write
add	0		10	0	0	0	0	1
addi	1		?	0	0	0	0	1
lw	1		00	0	1	0	1	1
sw	1		00	1	0	0	X	0
beq	0		01	0	0	1	X	0

Exercise

1. In RISC-V implementation, why do we need two separate control units (main and ALU control units) instead of one? What is the benefit of doing this?

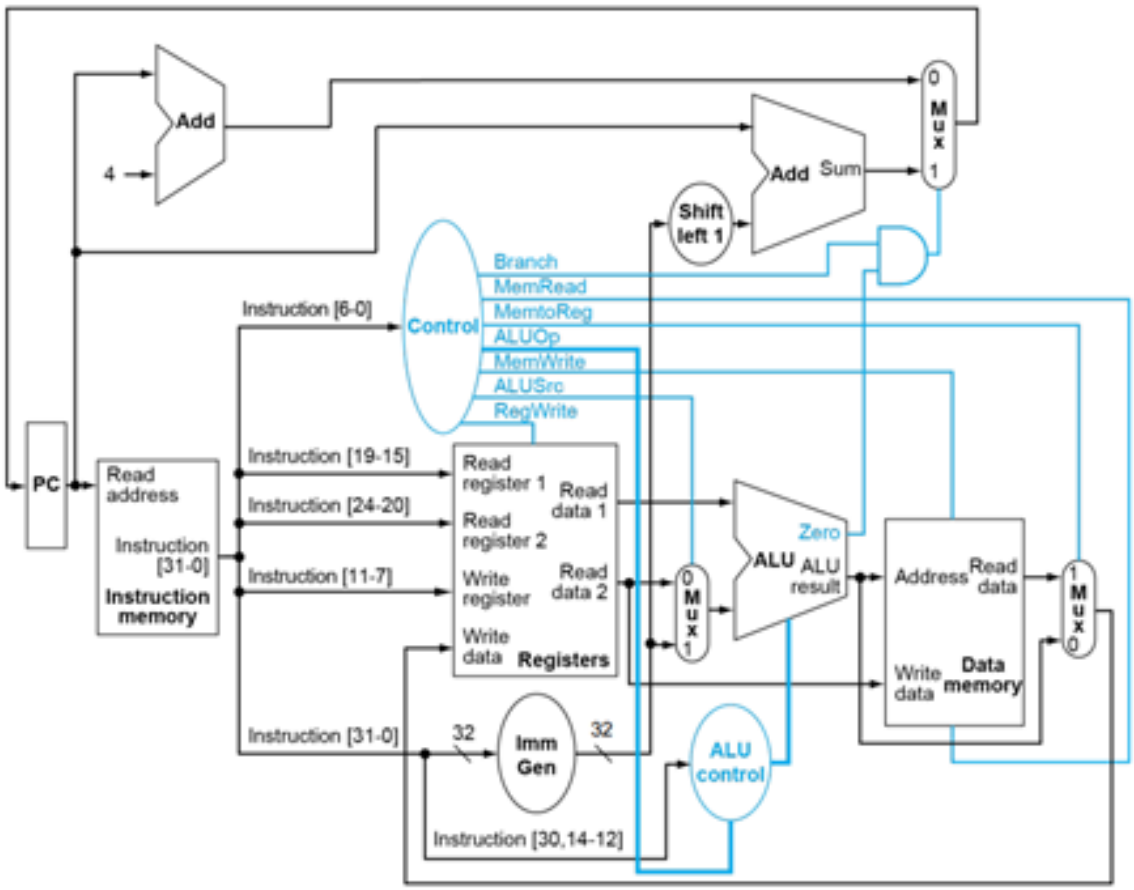
   Answer: This style of using multiple levels of decoding—that is, the main control unit generates the ALUOp bits, which then are used as input to the ALU control that generates the actual signals to control the ALU unit—is a common implementation technique. Using multiple levels of control can ***reduce the size of the main control unit***. Using several smaller control units may also ***potentially reduce the latency of the control unit***. Such optimizations are important, since the latency of the control unit is often a critical factor in determining the clock cycle time.

2. Given the following RISC-V assembly: `slti Rd, Rs, Imm`. •What are the values of the following control signals for this instruction? (A table for ALUOp control bits is listed below for reference)

   Note: `slti` *refers to "Set if less than, immediate", more can be found in appended RISC-V reference card.*

ALUOp control bits and the different opcodes

opcode	ALUOp
ld	00
sd	00
beq	01
R-type	10



Answer:

ALUsrc = 1, ALUop = 10

MemWrite = 0, MemRead = 0, MemtoReg = 0

Branch = 0

RegWrite = 1

Performance Evaluation

Execution time:

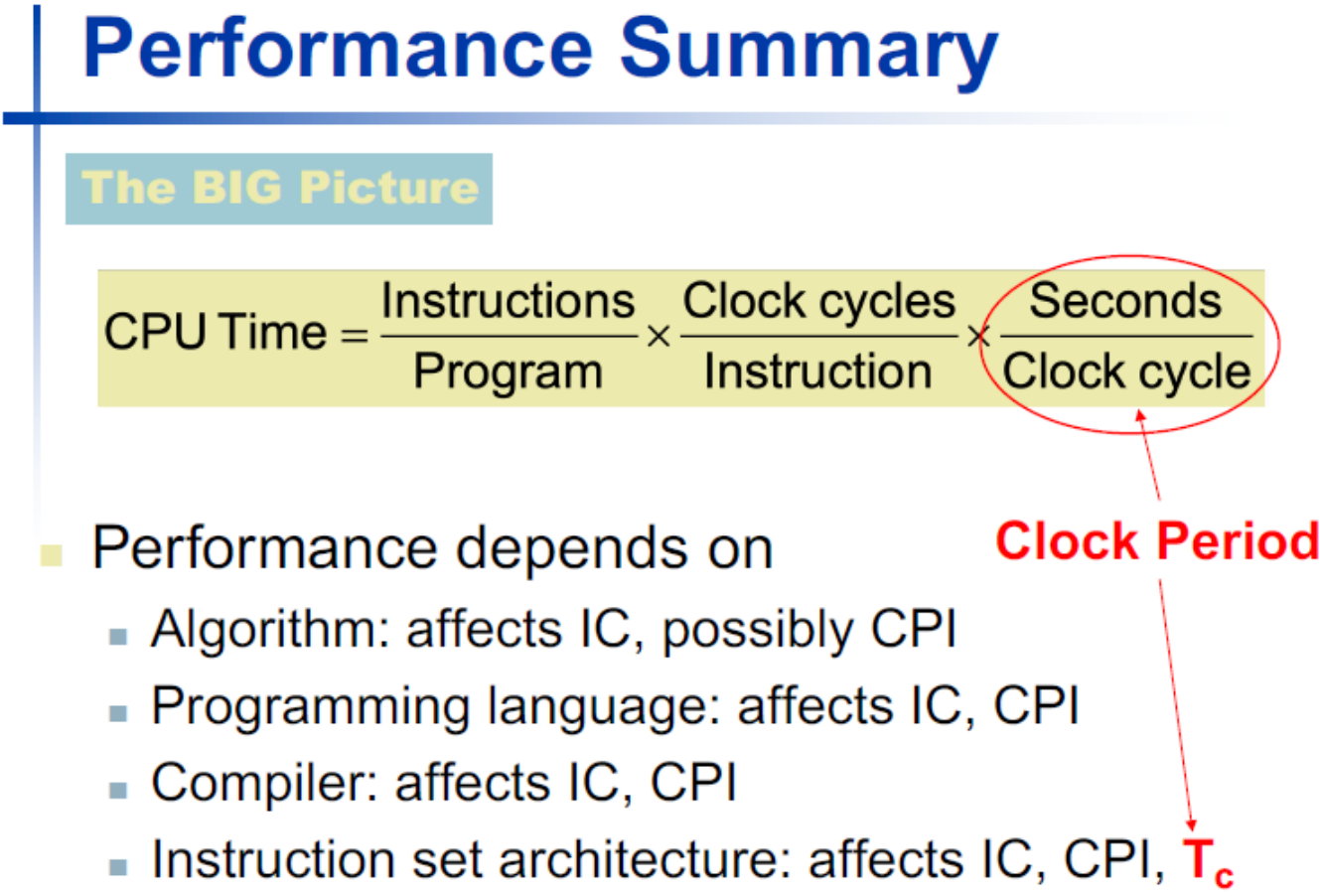
- Elapsed time - for **System Performance**
- CPU time - for **CPU Performance**

$$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
$$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

How to improve performance?

- Number of clock cycles↓
- Clock rate ↑
- Tradeoff

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$
$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$
$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$



Exercise

Assume a typical program has the following instruction type breakdown:

35% loads, 10% stores, 50% adds, 3% multiplies, 2% divides

Assume the current-generation RISC-V processor has the following instruction latencies:

loads: 4 cycles, stores: 4 cycles, adds: 2 cycles, multiplies: 16 cycles, divides: 50 cycles

If for the next-generation design you could pick one type of instruction to make twice as fast (half the latency), which instruction type would you pick? Why

Answer:

Calculate the CPI of each type of instruction:

loads: 1.4, stores: 0.4, adds: 1.0, multiplies: 0.48, divides: 1.0

Attention! Hw3 Ex5

5. (30 points) Problems in this exercise assume that the logic blocks used to implement a processor's datapath have the following latencies:

I-Mem / D-Mem	Register File	Mux	ALU	Adder	Single gate	Register Read	Register Setup	Sign extend	Control
250 ps	150 ps	25 ps	200 ps	150 ps	5 ps	30 ps	20 ps	30 ps	100 ps

In above table, "Register Read" is the time needed after the rising clock edge for the new register value to appear on the output. This value applies to the PC only. "Register Setup" is the amount of time a register's data input must be stable before the rising edge of the clock. This value applies to both the PC and Register File.

Attention! Hw3 Ex6

6. (10 points) Modify the single-cycle processor datapath to support the `jal` instruction:

Answer:

jal rd, Target involves two operations:

1). R[rd] <= PC+4.
To load PC + 4 into register file, a new MUX is needed to select between PC + 4 and MemtoReg MUX output for the Write Data input to the register file. The new MUX should be controlled by a new control signal generated by the Control unit for jal.

2). PC <= PC + imm20 << 2
To load the Target into PC, a new MUX is needed to select between PCSur MUX output and the calculated Target. The new MUX should be controlled by a new control signal generated by the Control unit for jal.