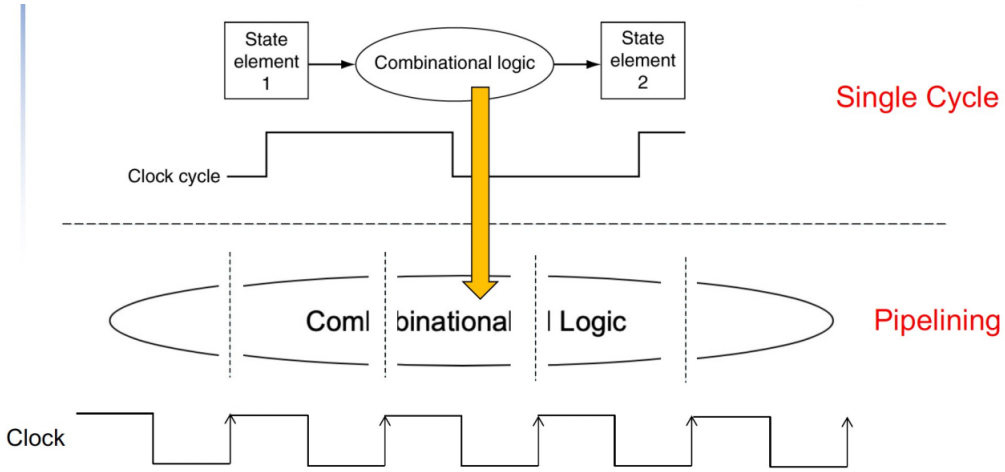
Pipeline Processors (T6)

VE370 FA22 Yijun Lu

Pipeline Processor

From Single-Cycle to Pipeline

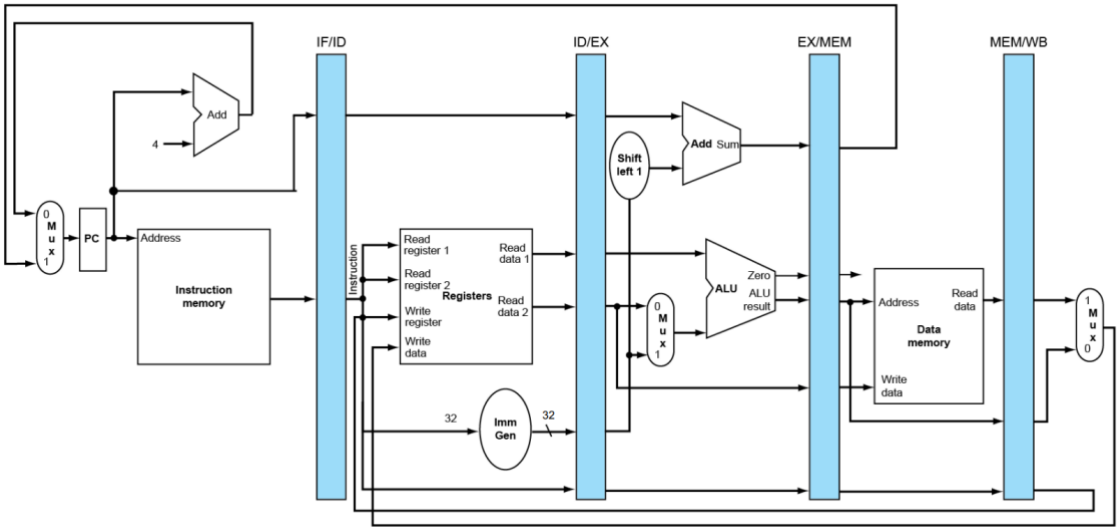
- Shorter clock cycle time. Speed up the process a lot.
- Introduce hazards: data hazards (T7) and control hazards (T8)



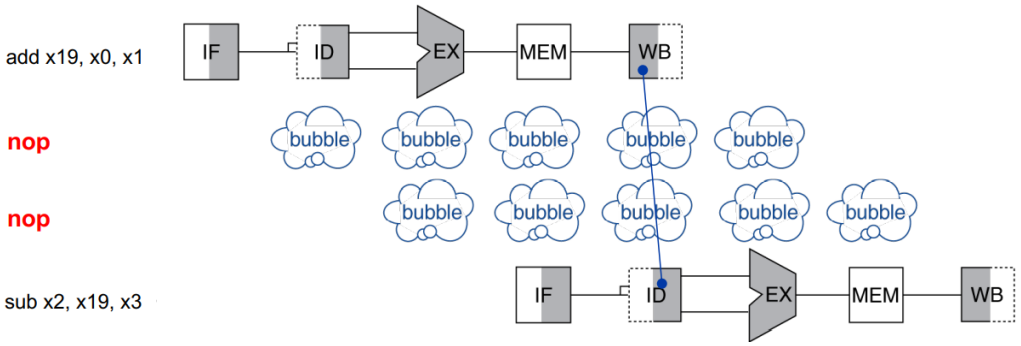
5 Stages:

- IF: Instruction fetch from memory
- ID: Instruction decode & register read
- EX: Execute operation or calculate address
- MEM: Access memory operand
- WB: Write result back to register

Datapath:

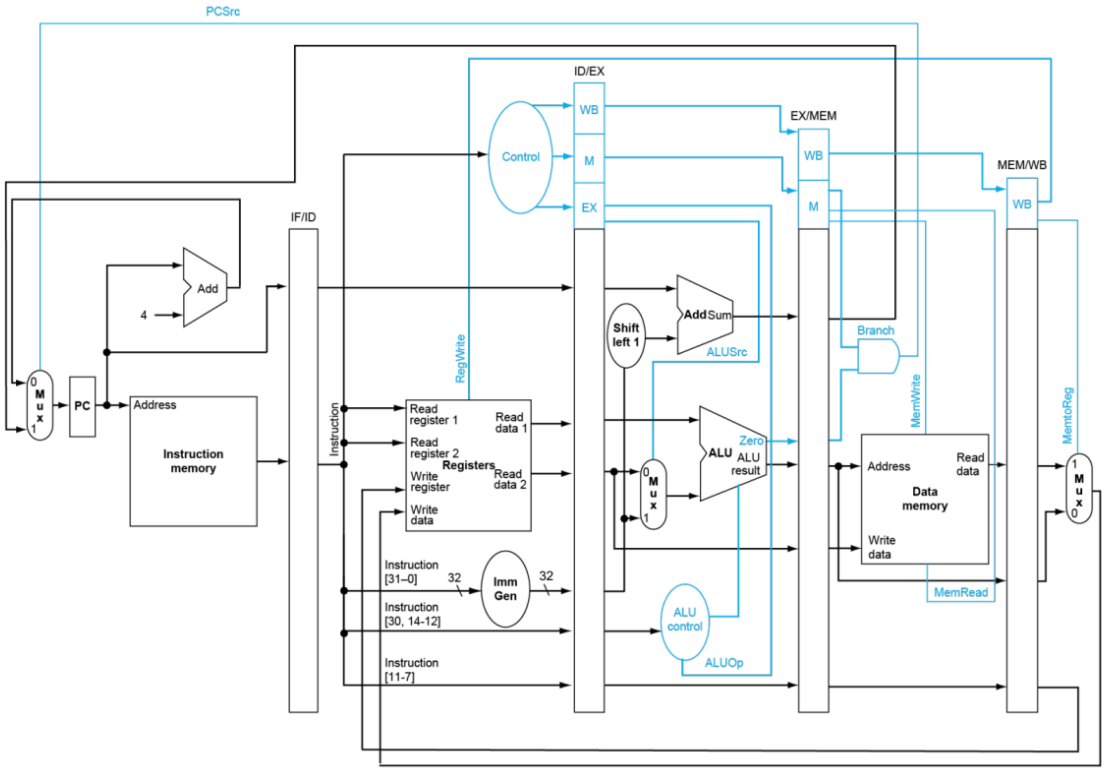


- Add registers between stages: hold information produced in previous cycle
- Write reg in the first half of clock, then read reg in the second half



- n+4 clock cycles for n instruction if no hazards

Control Signals



- Each module should have their control signals as assigned in single-cycle processors.
- Consumed in appropriate stages.
- Control signals will be useless after it finishes its job in corresponding stage.

Example

Suppose we have the following instructions to be processed in the pipeline structure. (No hazard)

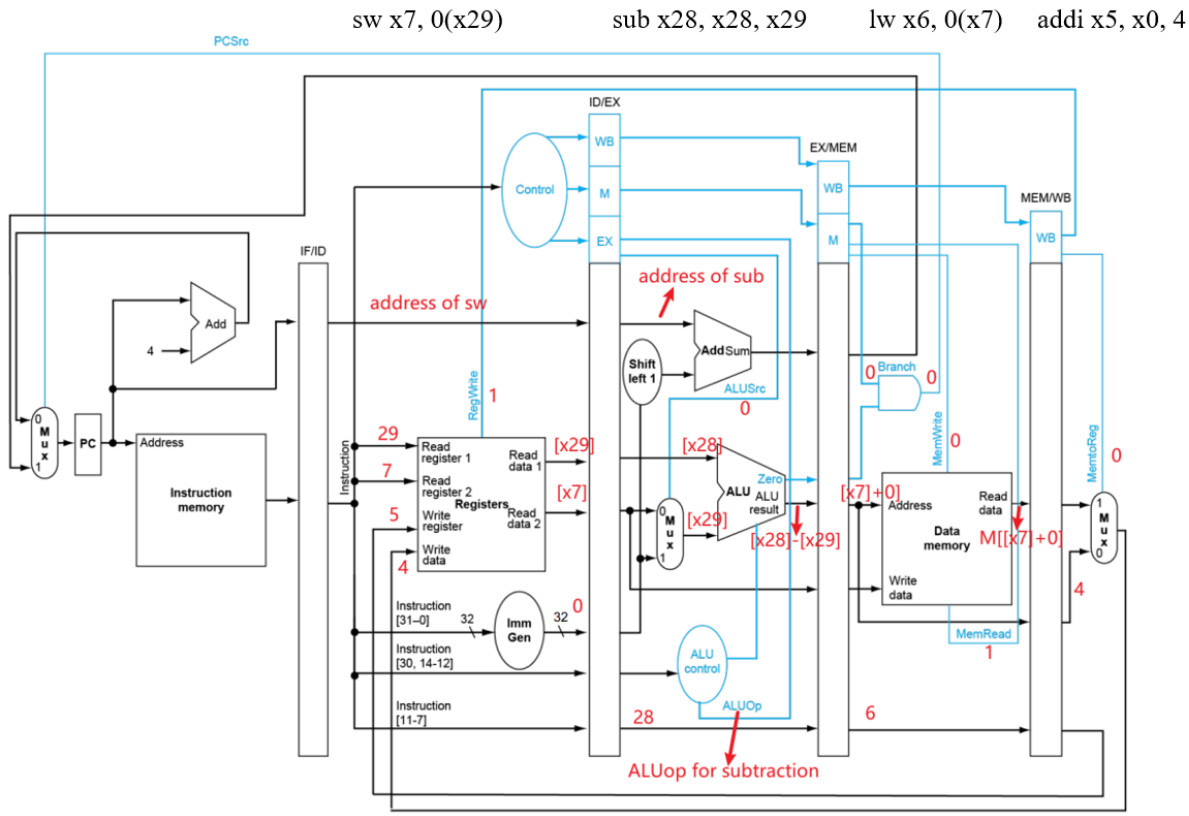
```
addi x5, x0, 4
lw x6, 0(x7)
sub x28, x28, x29
sw x7, 0(x29)
```

Multi-cycle pipeline diagram

	C1	C2	C3	C4	C5	C6	C7	C8	C9
addi	IF	ID	EX	MEM	WB				
lw		IF	ID	EX	MEM	WB			
sub			IF	ID	EX	MEM	WB		
sw				IF	ID	EX	MEM	WB	

Execution details

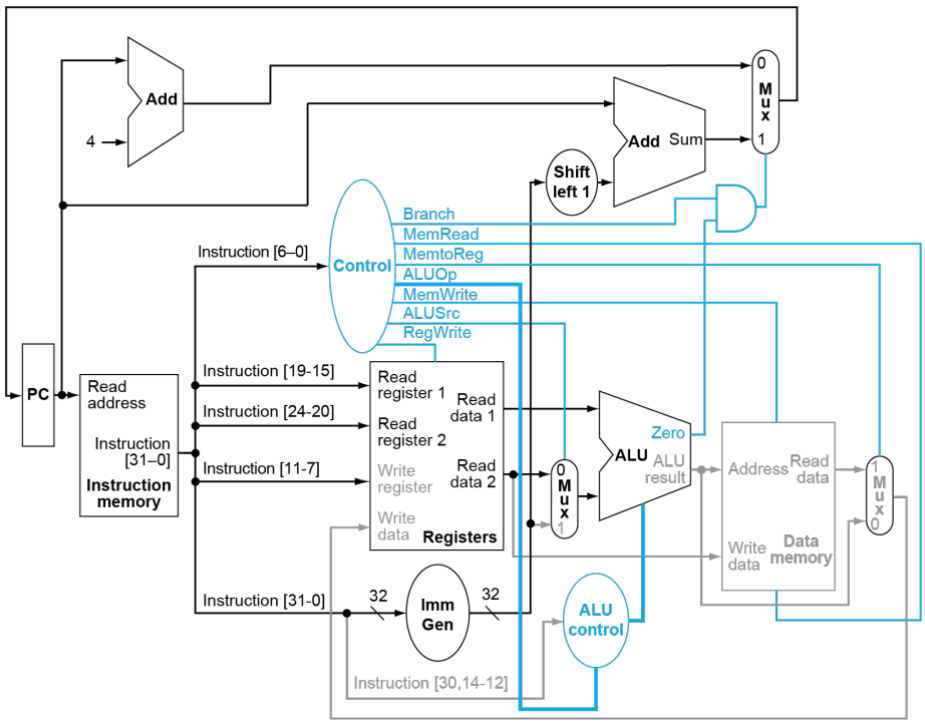
C5:



Homework and Exercise

HW3 ex6

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



Exercise

(10 points) Modify the single-cycle processor datapath to add a proposed new assembly instruction:

```
ss rs1, rs2, immediate  #Store Sum
```

Operation: Mem[Reg[rs1]]=Reg[rs2]+immediate