CSE 2600
Intro. To Digital Logic & Computer Design

Bill Siever
&
Michael Hall

This week

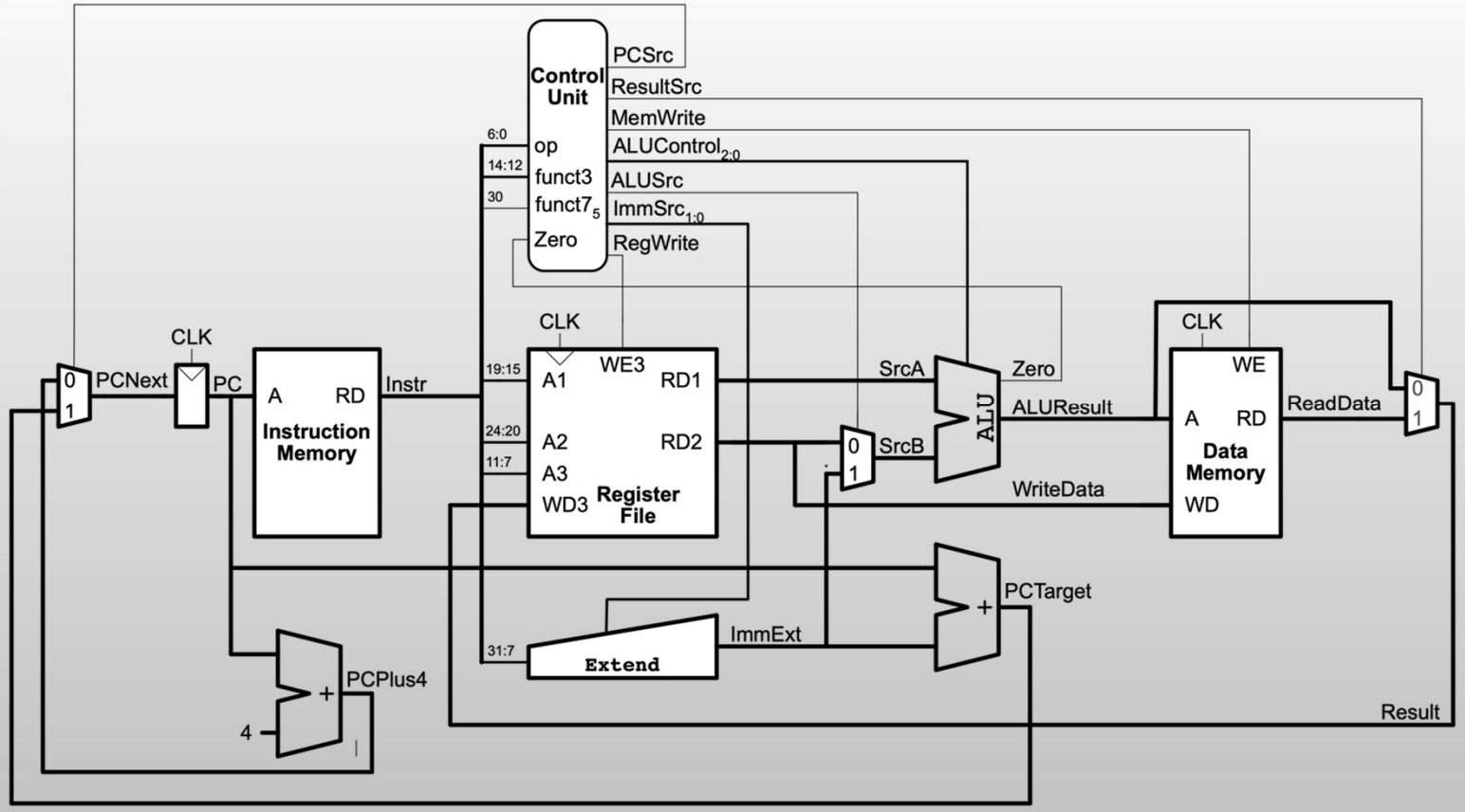
- Homework 8 – Optional, extra credit
  - Hard deadline of submission to Gradescope on Wed, Dec. 3 by 11:59 pm.
  - Must demo to either Prof. Hall or Siever by the 12th
  - Additional office hours will be posted for the 8-12th

Studio Review

- Control Signals…
- jal encoding / impact
- riscvsingle.sv

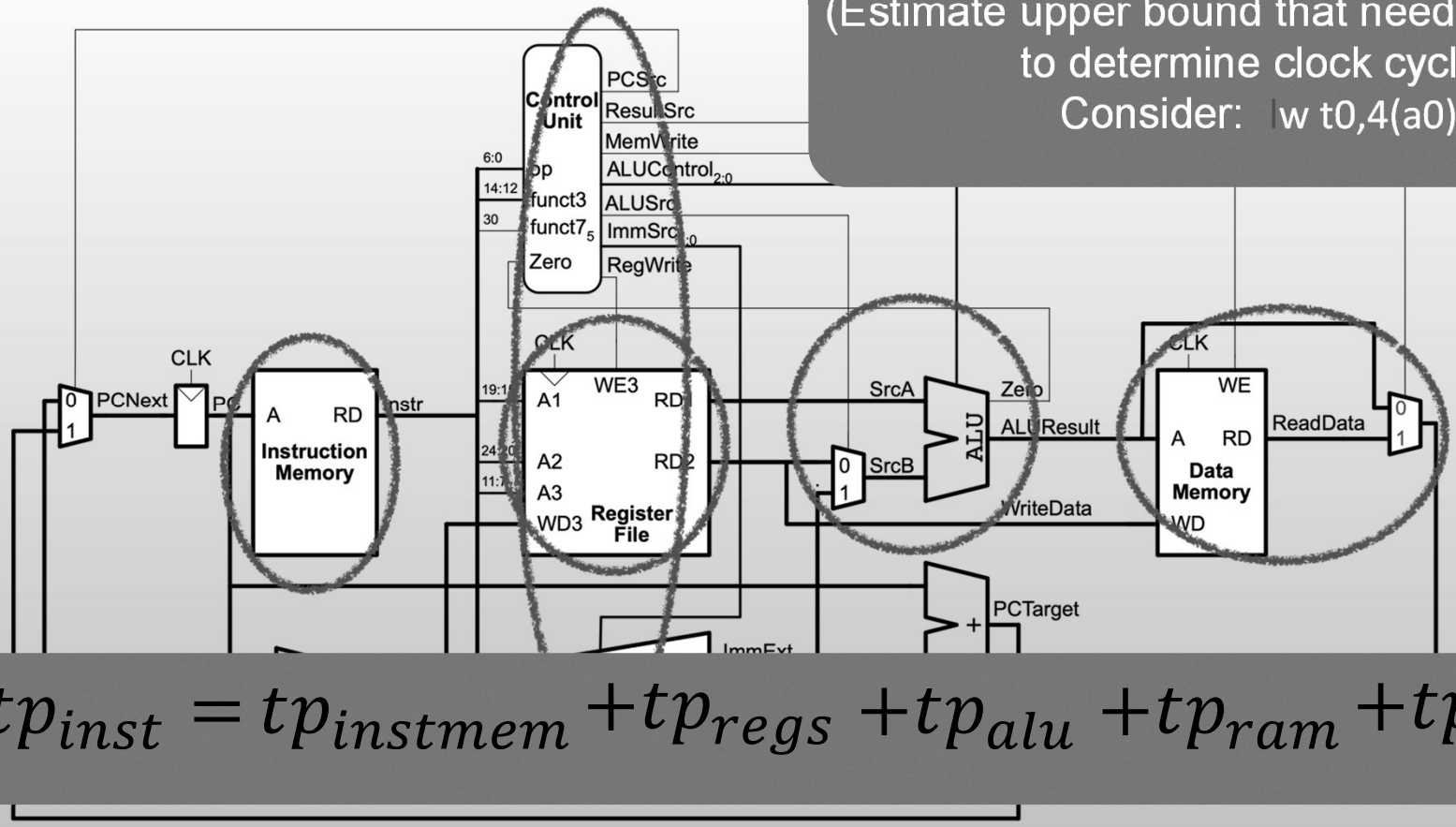
Chapter 7

Simple (Single-Cycle) RISC-V Computer



Simple, Single-Cycle

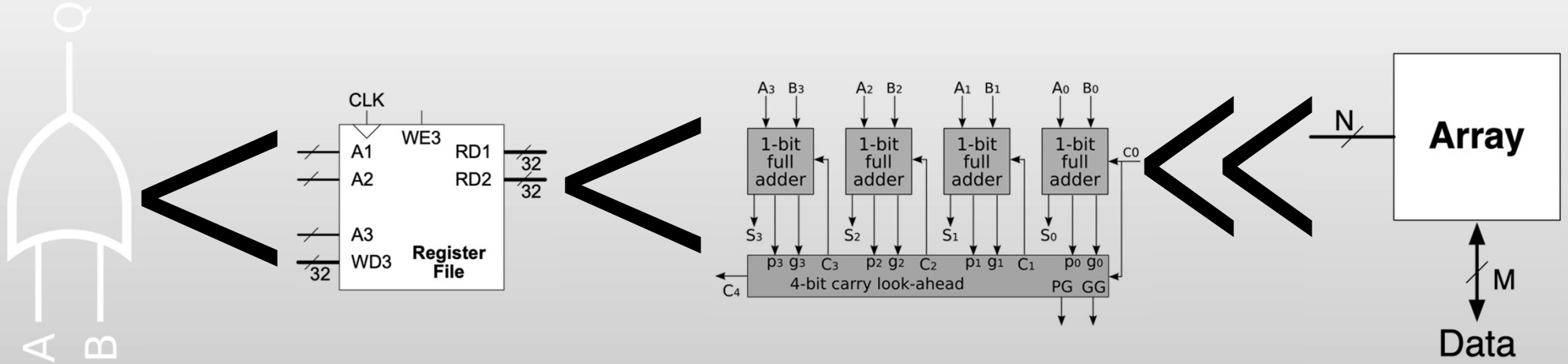
Identify items that are part of the "propagation delay" of an instruction. (Estimate upper bound that needs to be used to determine clock cycle)
Consider: lw t0,4(a0)



$$tp_{inst} = tp_{instmem} + tp_{regs} + tp_{alu} + tp_{ram} + tp_{regs}$$

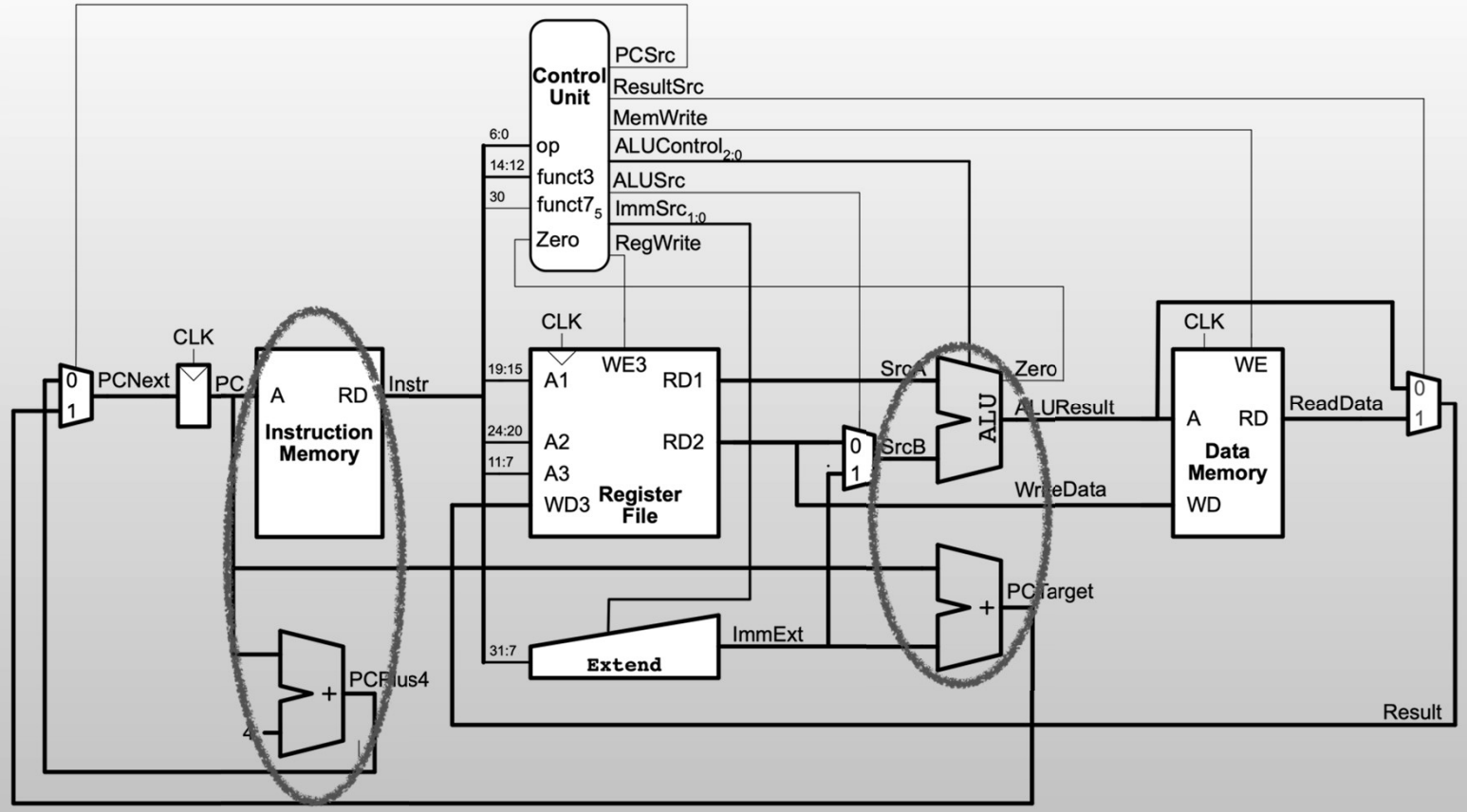
Consider Performance (prop delay) of Parts

Consider: add Consider: or Consider: sw Consider: lw



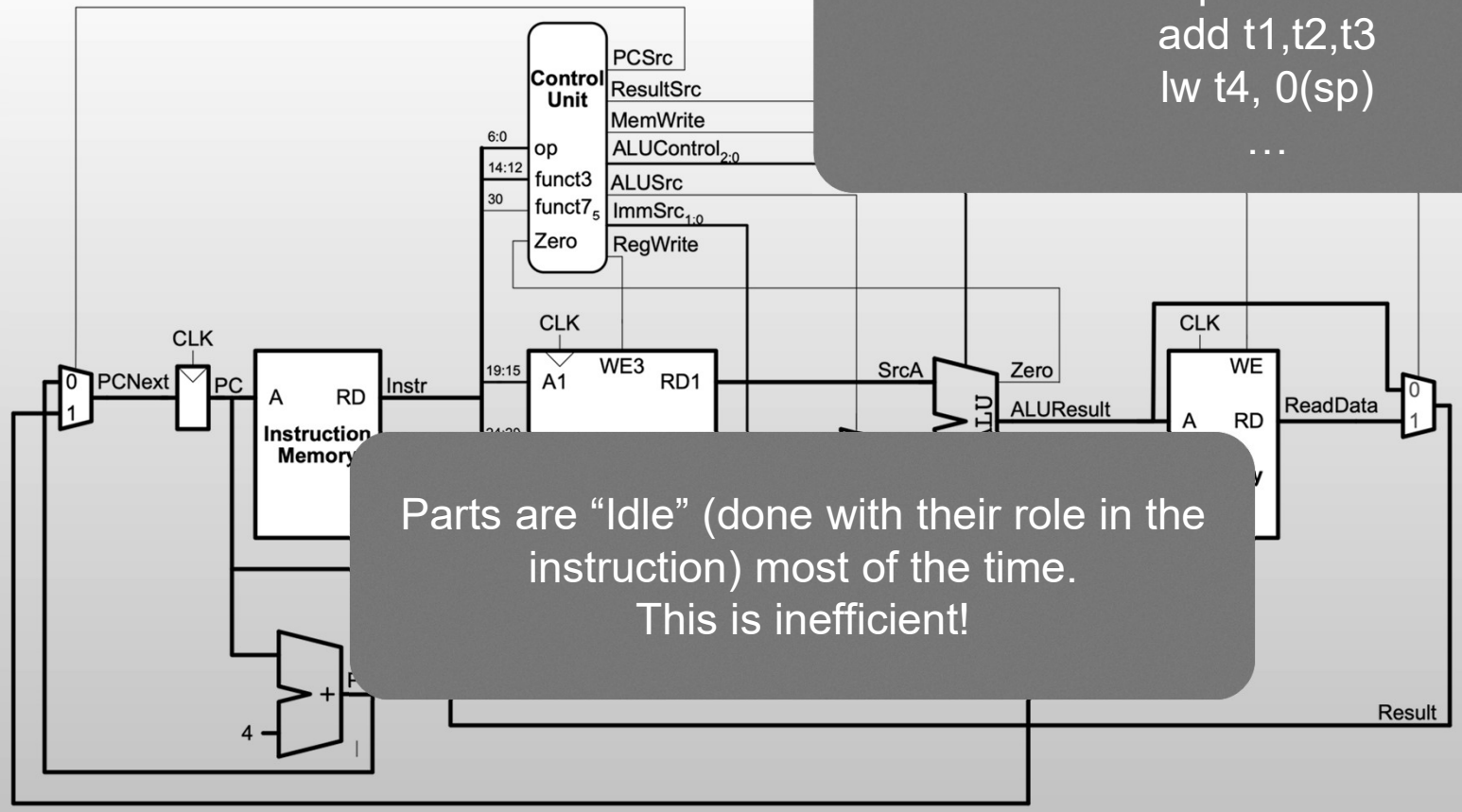
Simple, Single-Cycle RISC-V Computer

Consider times when adders are “working”



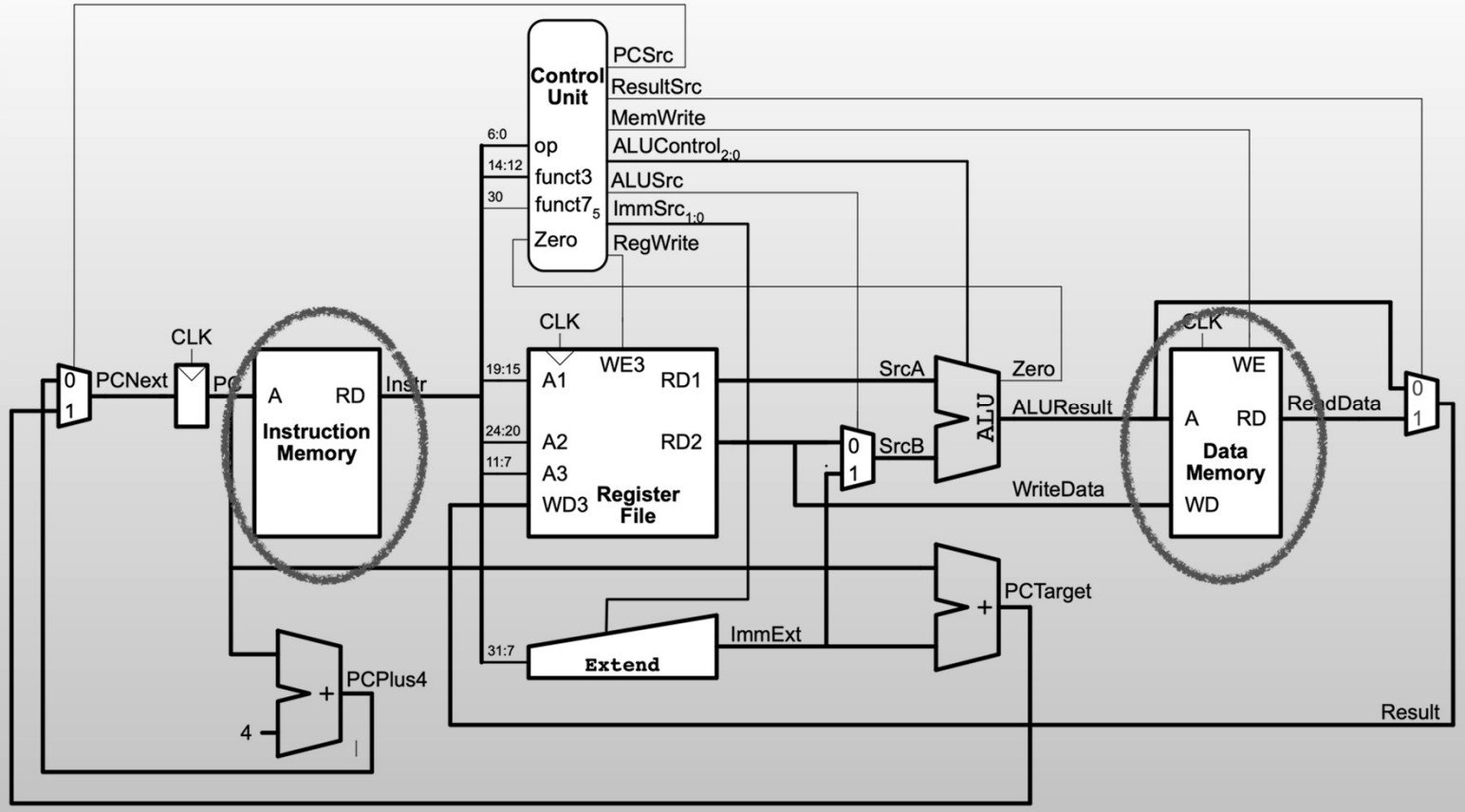
Simple, Single-Cycle RISC-V Computer

Consider a sequence of instructions:
add t1,t2,t3
lw t4, 0(sp)
…



Parts are “Idle” (done with their role in the instruction) most of the time.
This is inefficient!

Simple, Single-Cycle RISC-V Computer

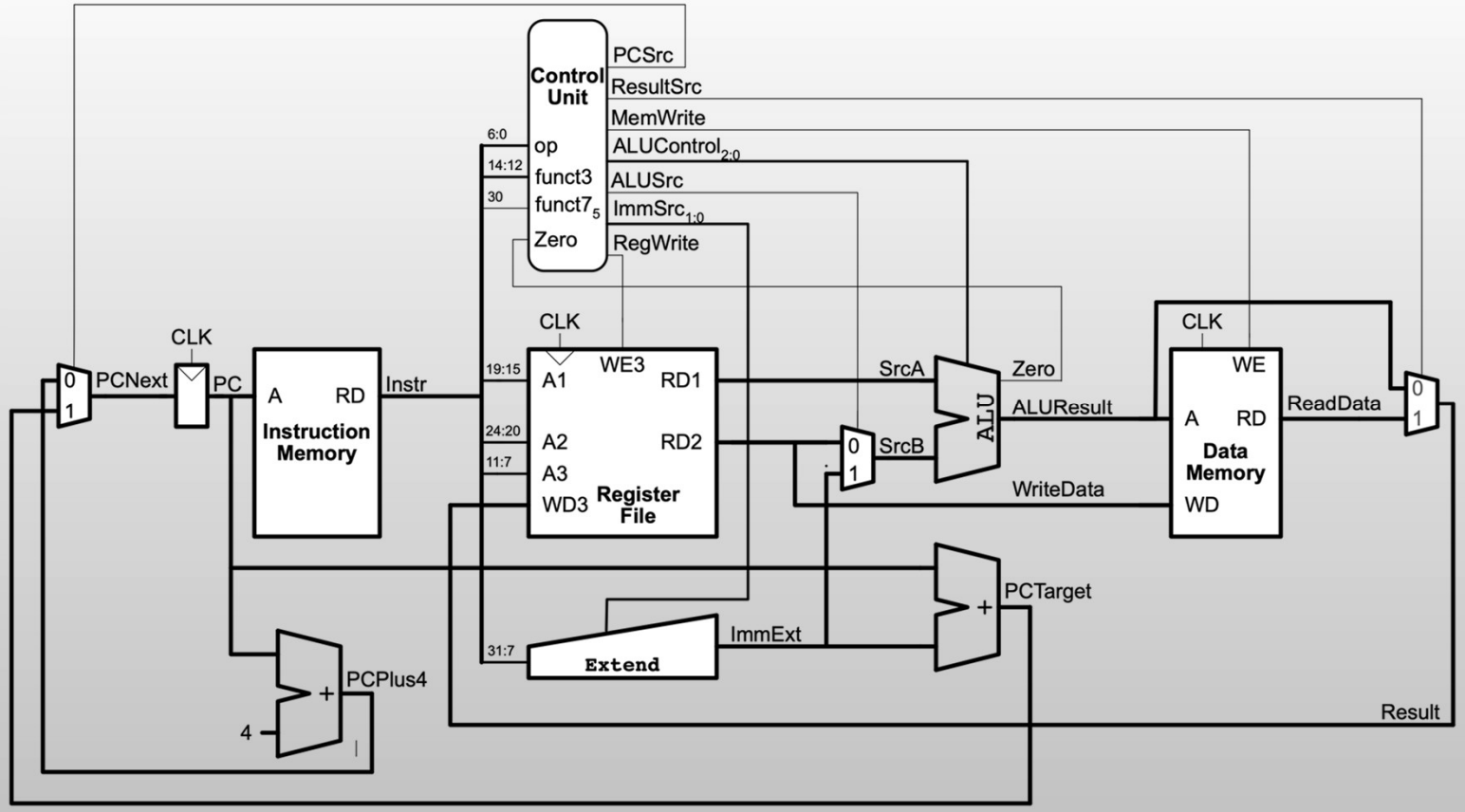


Architectures

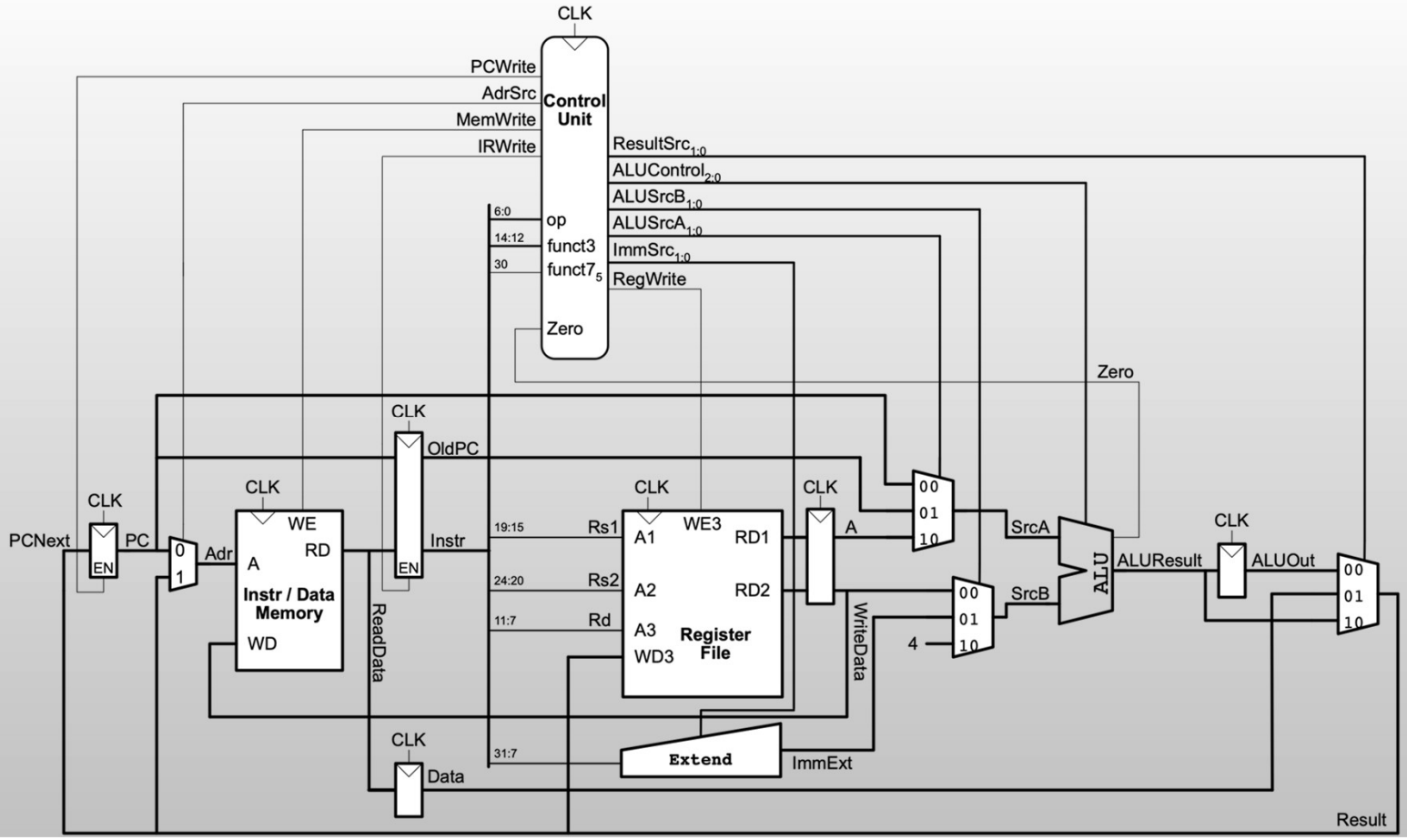
- Harvard Architecture
  - Fixed program?
  - Not so uncommon: Car, appliances, small electronics
    - Questions: Are single-cycle things used? Yes.

- von Neumann
  - General purpose: Magic of being able to change programs *easily*
    - *Programs (operating sys) can change program: "Computer"* , tablets, phones, …

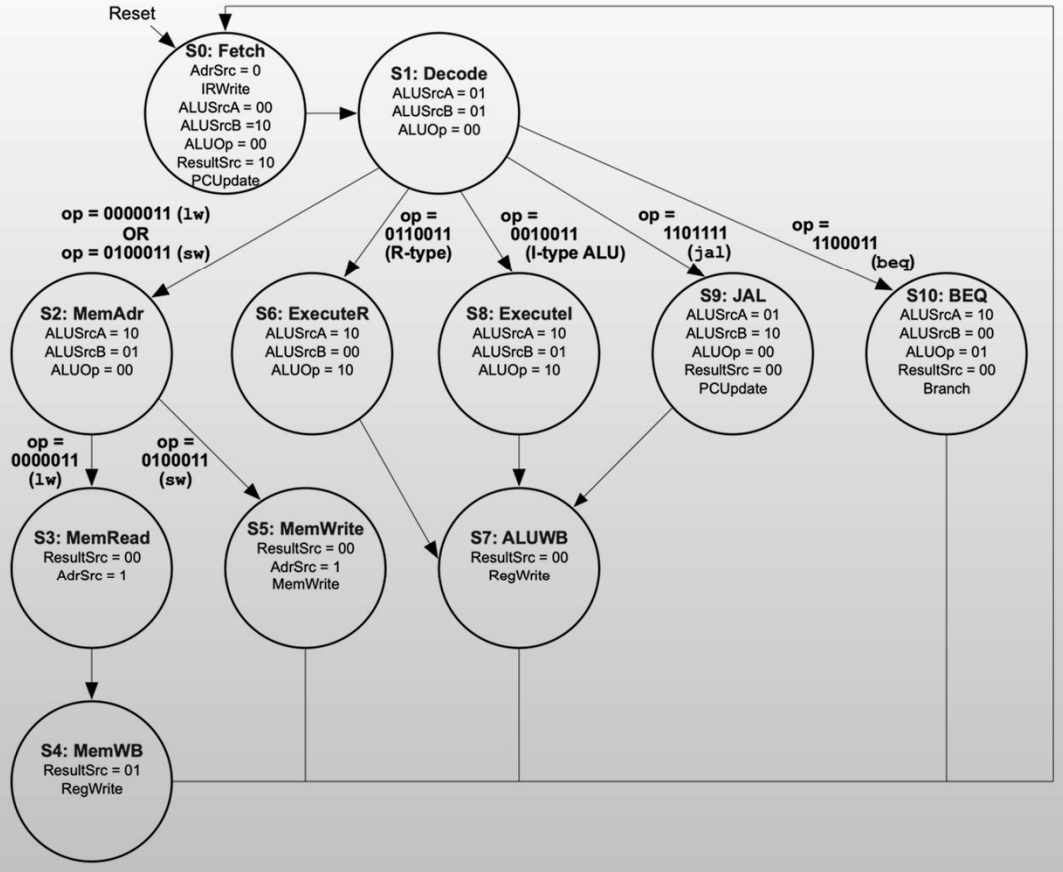
Simple, Single-Cycle RISC-V Computer



Multi-Cycle RISC-V Computer



Process


Reset
S0: Fetch
AdrSrc = 0
IRWrite
ALUSrcA = 00
ALUSrcB =10
ALUOp = 00
ResultSrc = 10
PCUpdate
S1: Decode
ALUSrcA = 01
ALUSrcB = 01
ALUOp = 00
op = 0000011 (lw)
OR
op = 0100011 (sw)
op = 0110011 (R-type)
op = 0010011 (I-type ALU)
op = 1101111 (jal)
op = 1100011 (beq)
S2: MemAdr
ALUSrcA = 10
ALUSrcB = 01
ALUOp = 00
S6: ExecuteR
ALUSrcA = 10
ALUSrcB = 00
ALUOp = 10
S8: ExecuteI
ALUSrcA = 10
ALUSrcB = 01
ALUOp = 10
S9: JAL
ALUSrcA = 01
ALUSrcB = 10
ALUOp = 00
ResultSrc = 00
PCUpdate
S10: BEQ
ALUSrcA = 10
ALUSrcB = 00
ALUOp = 01
ResultSrc = 00
Branch
op = 0000011 (lw)
op = 0100011 (sw)
S3: MemRead
ResultSrc = 00
AdrSrc = 1
S5: MemWrite
ResultSrc = 00
AdrSrc = 1
MemWrite
S7: ALUWB
ResultSrc = 00
RegWrite
S4: MemWB
ResultSrc = 01
RegWrite

Pros/Cons of Multi-Cycle

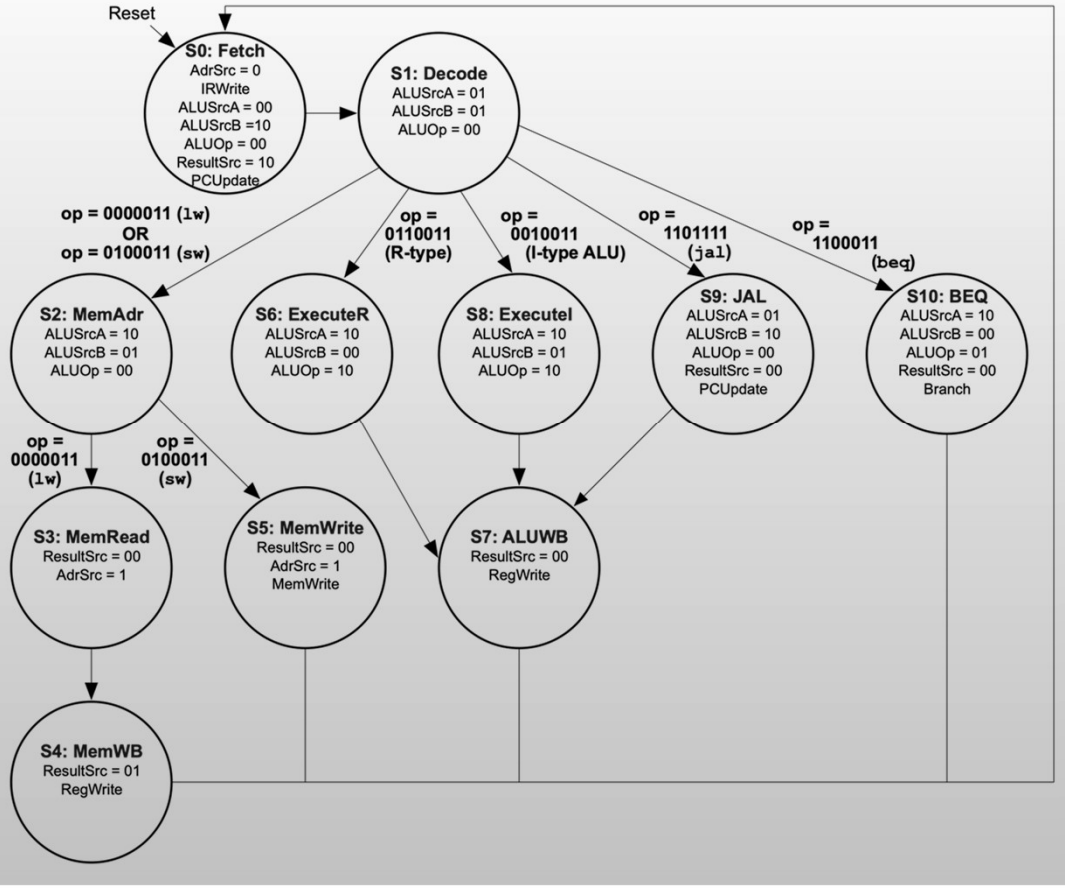
- Instructions take only required time: Not constrained by the slowest instruction!
- A little more complex

Questions: What do you think homework…

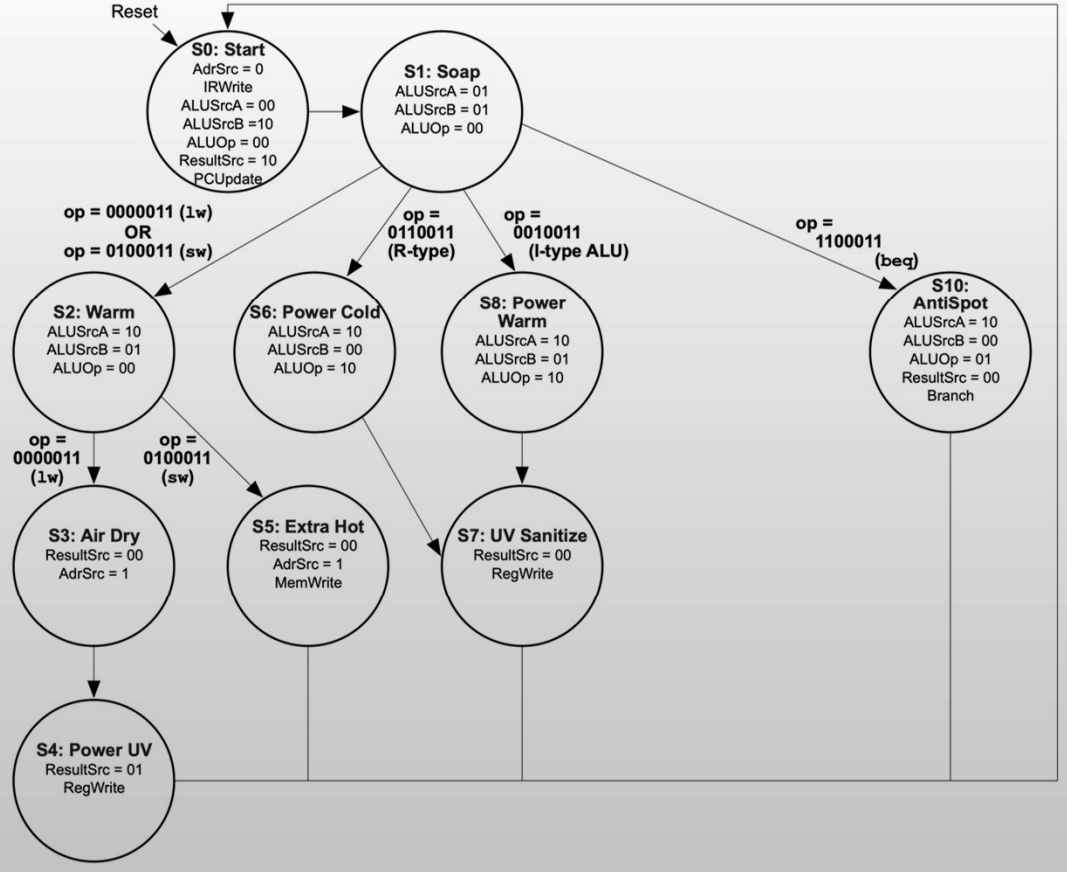
- …hope we don’t have to make the confusing state machine…

Good news!

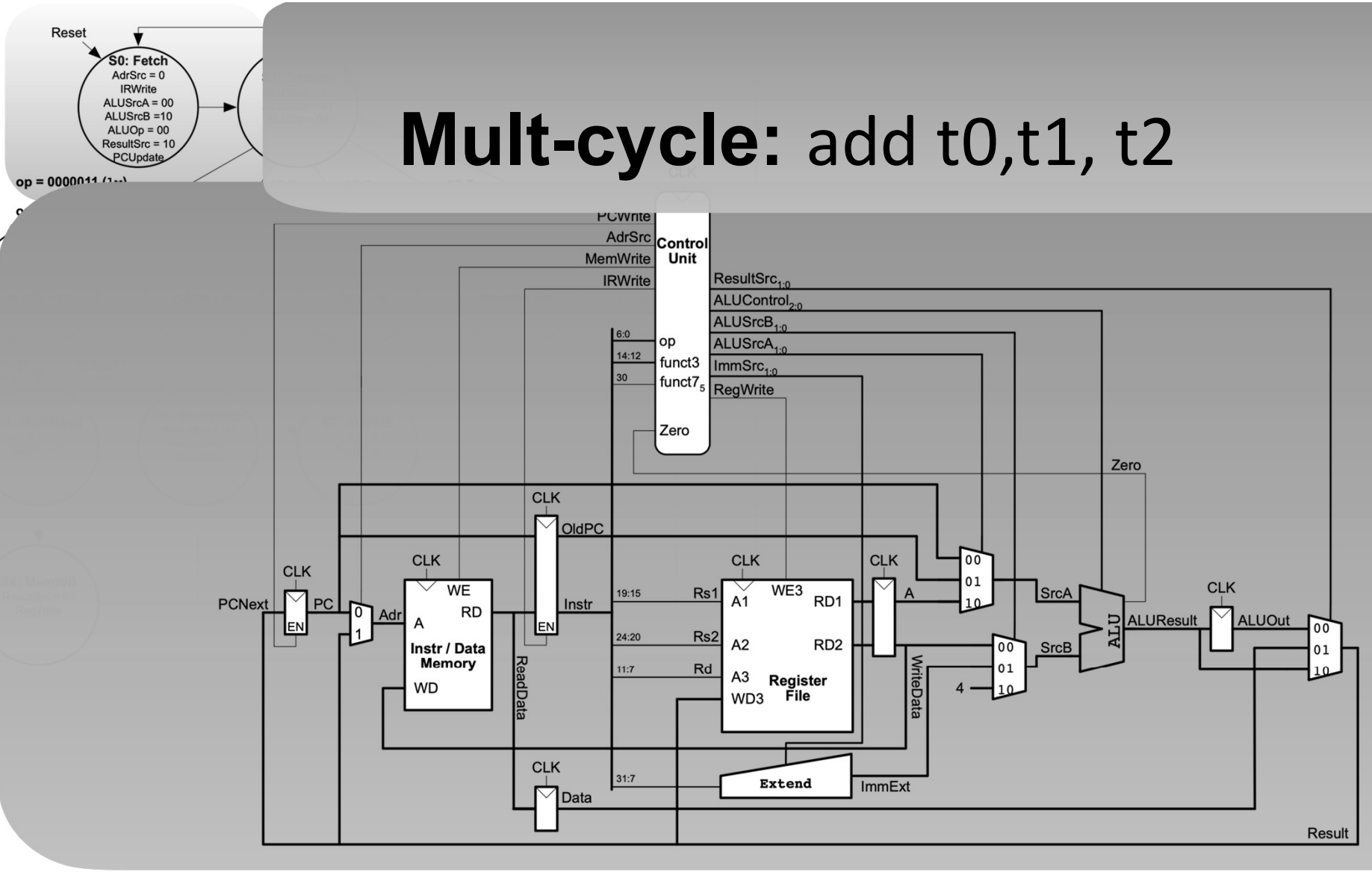
Process


Reset
S0: Fetch
AdrSrc = 0
IRWrite
ALUSrcA = 00
ALUSrcB =10
ALUOp = 00
ResultSrc = 10
PCUpdate
S1: Decode
ALUSrcA = 01
ALUSrcB = 01
ALUOp = 00
op = 0000011 (lw)
OR
op = 0100011 (sw)
op = 0110011 (R-type)
op = 0010011 (I-type ALU)
op = 1101111 (jal)
op = 1100011 (beq)
S2: MemAdr
ALUSrcA = 10
ALUSrcB = 01
ALUOp = 00
S6: ExecuteR
ALUSrcA = 10
ALUSrcB = 00
ALUOp = 10
S8: ExecuteI
ALUSrcA = 10
ALUSrcB = 01
ALUOp = 10
S9: JAL
ALUSrcA = 01
ALUSrcB = 10
ALUOp = 00
ResultSrc = 00
PCUpdate
S10: BEQ
ALUSrcA = 10
ALUSrcB = 00
ALUOp = 01
ResultSrc = 00
Branch
op = 0000011 (lw)
op = 0100011 (sw)
S3: MemRead
ResultSrc = 00
AdrSrc = 1
S5: MemWrite
ResultSrc = 00
AdrSrc = 1
MemWrite
S7: ALUWB
ResultSrc = 00
RegWrite
S4: MemWB
ResultSrc = 01
RegWrite

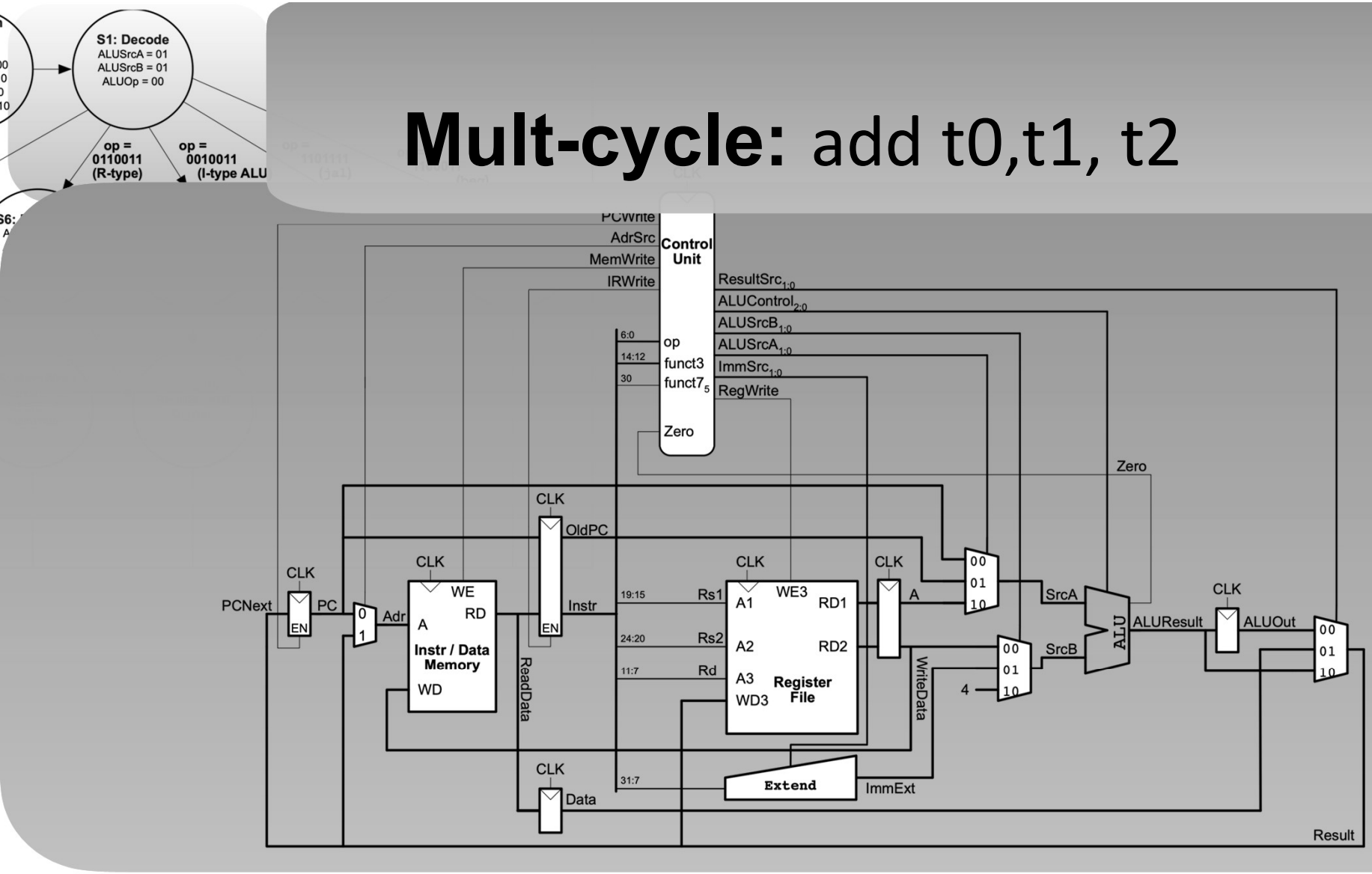
Already have - it's a Washer.
Hw 3B & 4B

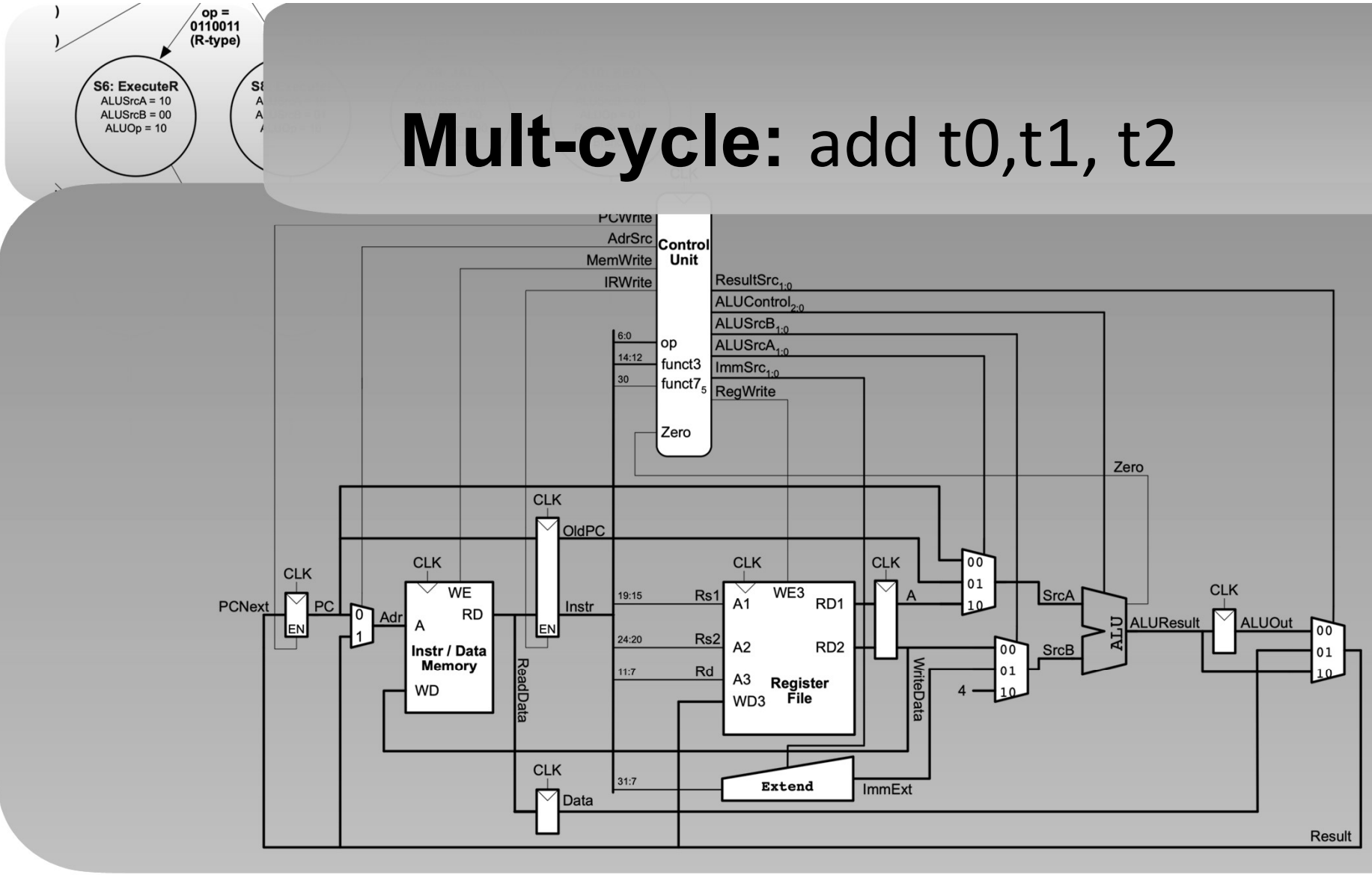


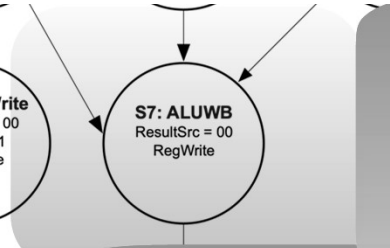
Mult-cycle: add t0,t1, t2



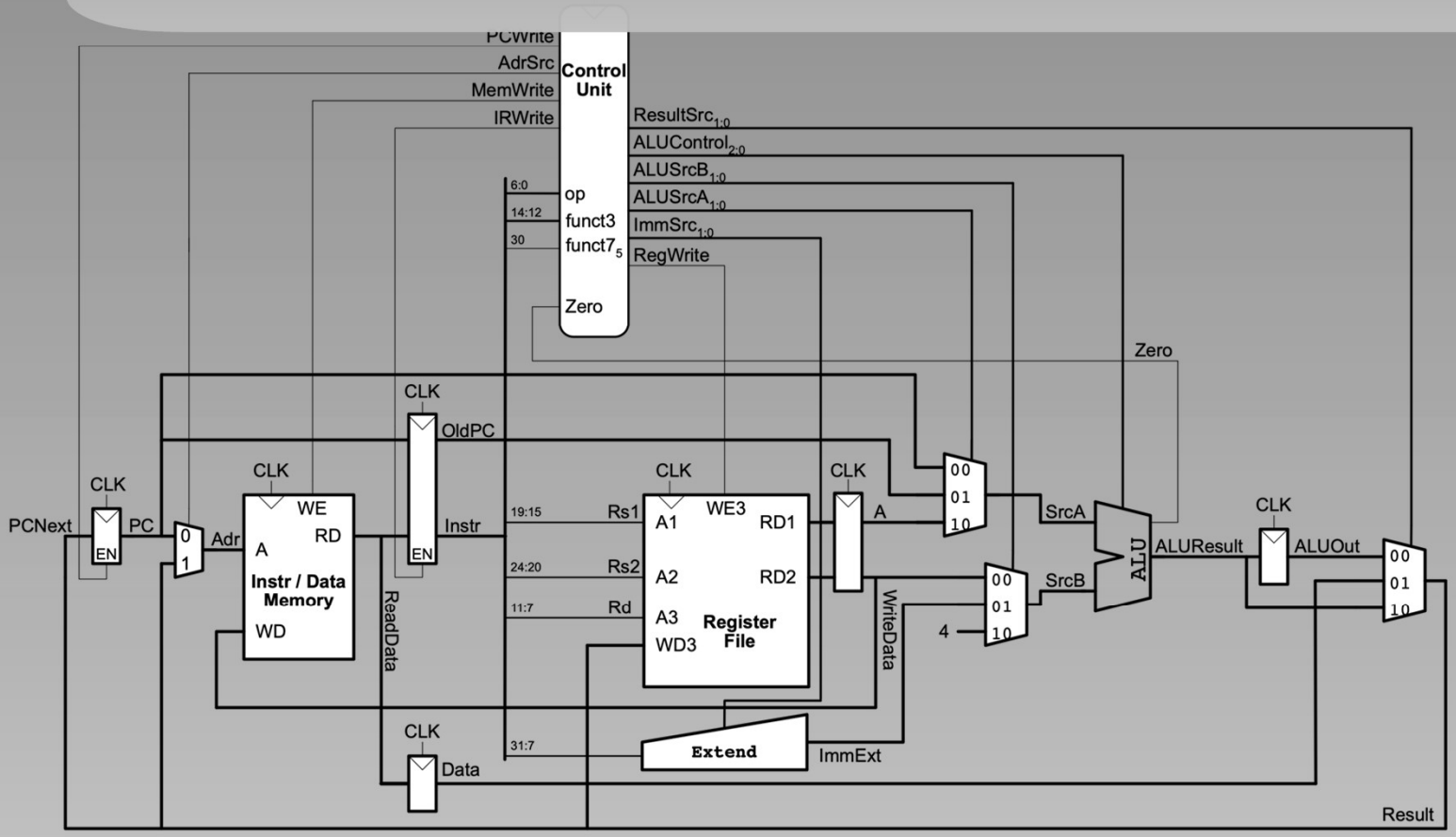
Mult-cycle: add t0,t1, t2




op =
0110011
(R-type)
S6: ExecuteR
ALUSrcA = 10
ALUSrcB = 00
ALUOp = 10
Mult-cycle: add t0,t1, t2
PCWrite
AdrSrc
MemWrite
IRWrite
Control Unit
ResultSrc1:0
ALUControl2:0
ALUSrcB1:0
ALUSrcA1:0
ImmSrc1:0
RegWrite
Zero
6:0
14:12
30
op
funct3
funct75
CLK
OldPC
PCNext
PC
EN
0
1
Adr
WE
RD
A
Instr / Data Memory
WD
ReadData
Instr
19:15
24:20
11:7
Rs1
Rs2
Rd
WE3
A1
A2
A3
RD1
RD2
Register File
WD3
WriteData
A
00
01
10
SrcA
SrcB
4
ALU
ALUResult
ALUOut
31:7
Extend
ImmExt
Data
Result

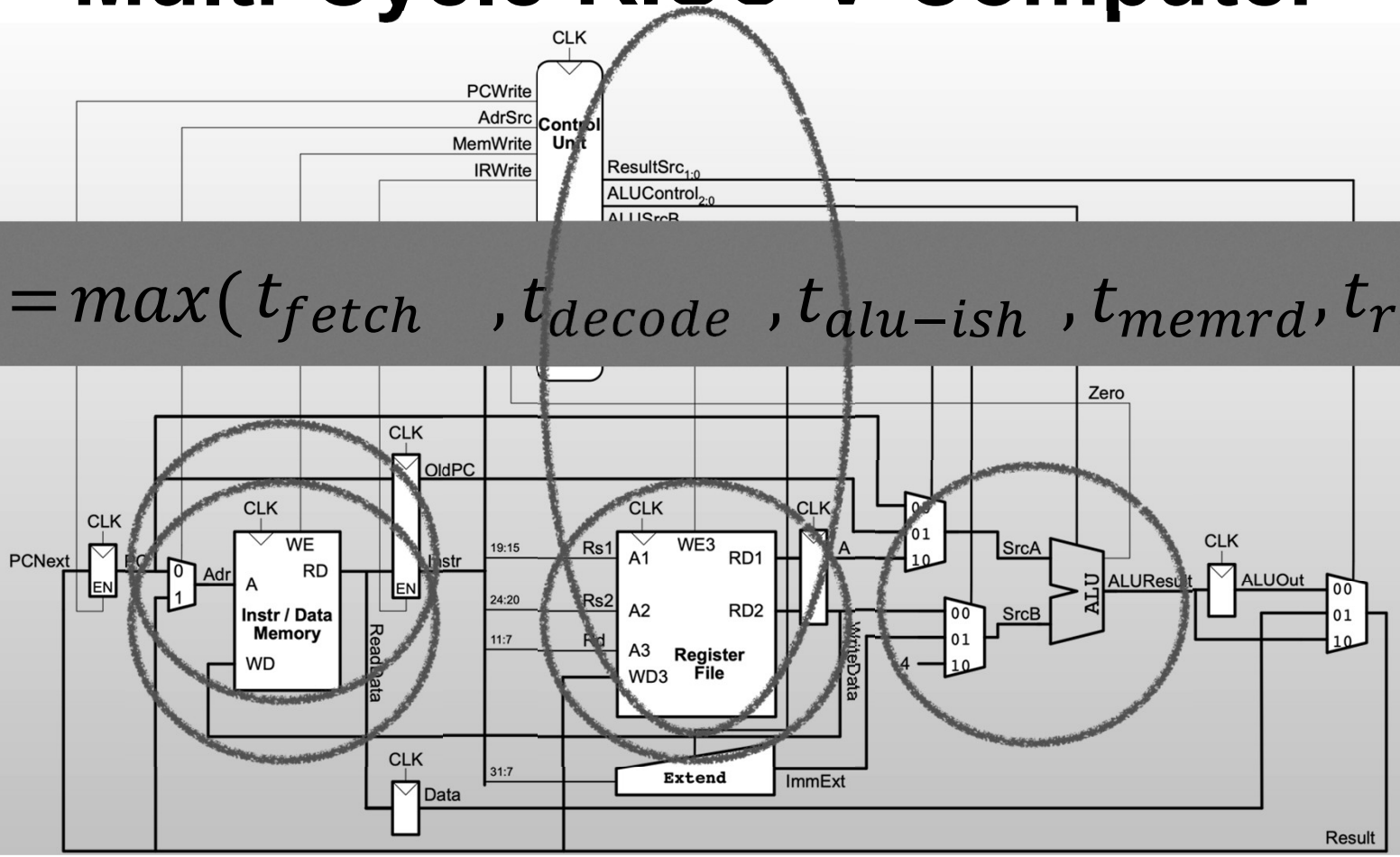


Mult-cycle: add t0,t1, t2



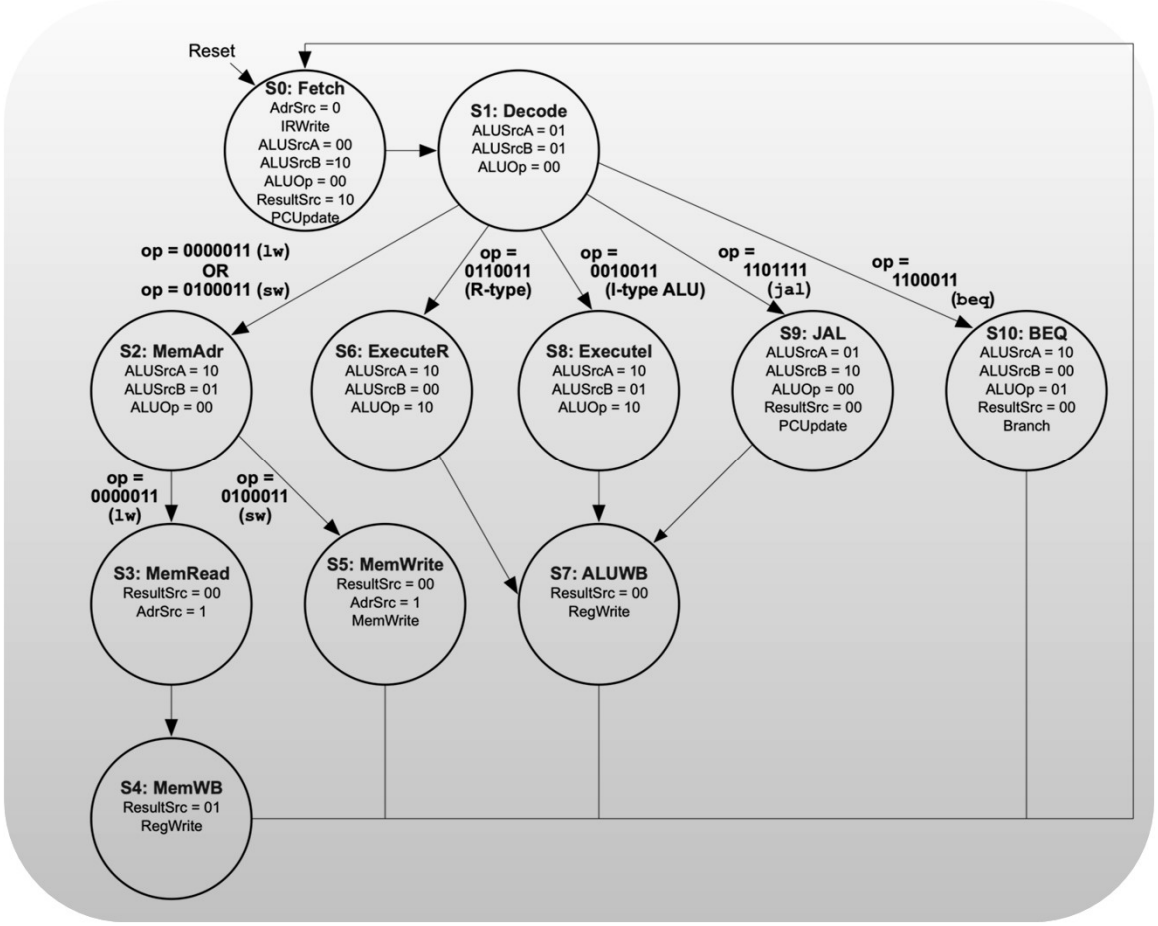
Multi-Cycle RISC-V Computer

$$t_{clock} = max(t_{fetch}, t_{decode}, t_{alu-ish}, t_{memrd}, t_{regwr})$$



Instruction Times

- 3-5 clock cycles
  - Some instructions *may* be faster



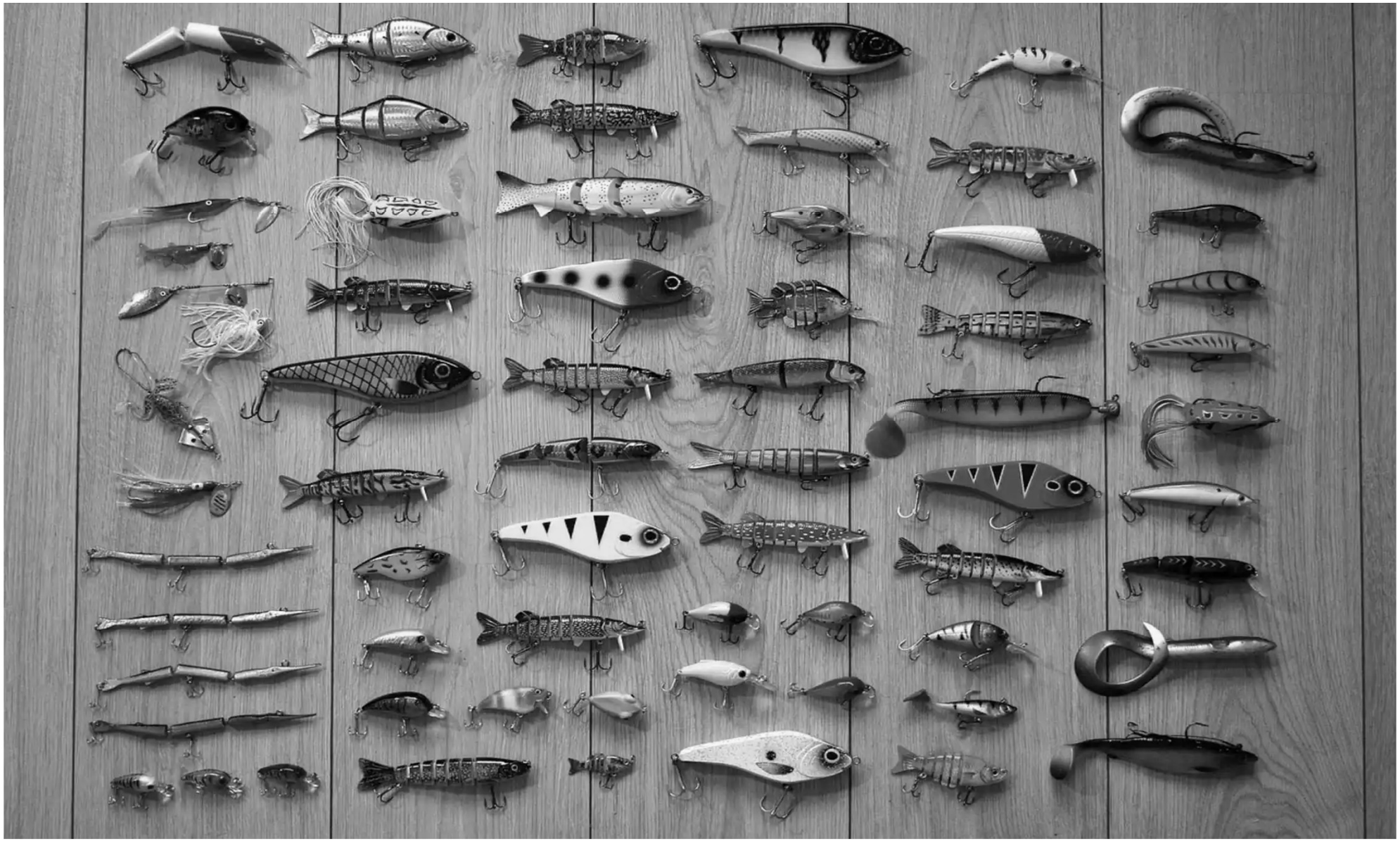
Next Improvement: Pipelines

Laundry

- Laundry machines
  - *Washer* takes 30 minutes
  - Dryer takes 1 hour (ugh)
- How long does it take to do 1 load of laundry all the way through?
- What about 2 loads?
- What's the approx. average for 50 loads of laundry?
- What if I'm doing 12 loads of laundry and put something in on load 4 that I *really* need?

A Pipeline / Factory

- My career: Develop Medical Equipment
  - Along with….



Customer Order

- Body style / size
- “Flair” (style and color)
- Body shape

Process

1. Parts prep: read order, put order in bin, put part for order in bin

2. Slide bin down line to “assembler”.

3. Slide down to cleaner

4. Slide down to packer

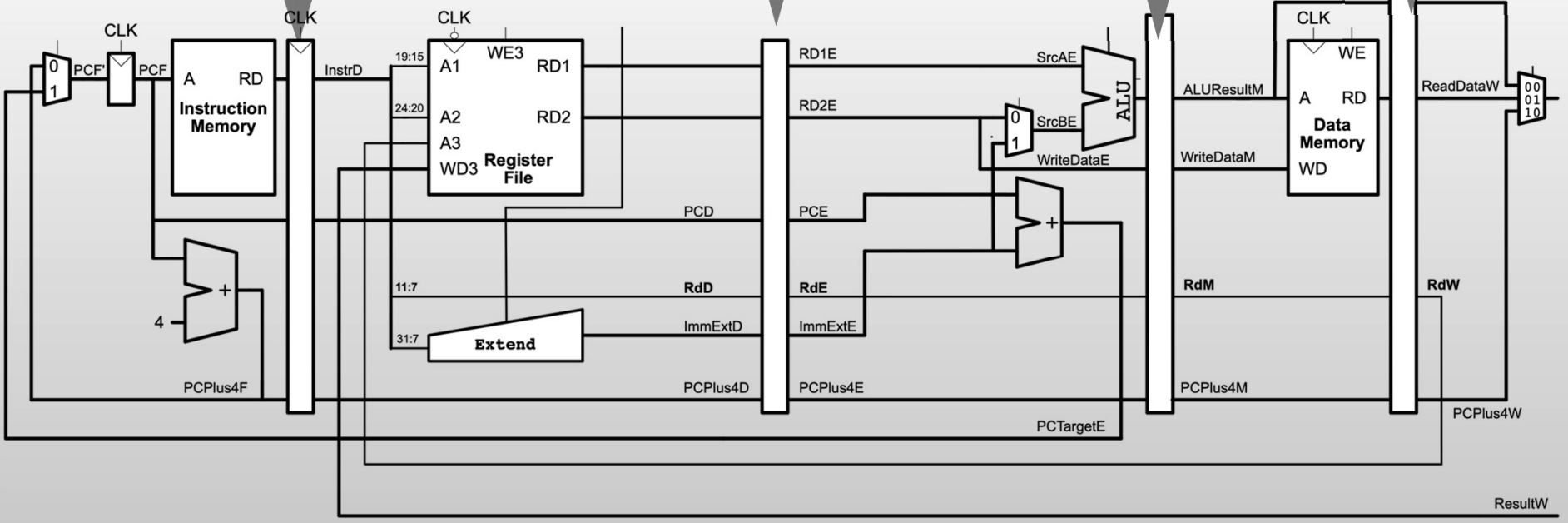
5.Move to shipping

Registers:
Between stages;
Like the parts bin (hold parts for inst), but parts move, not bins

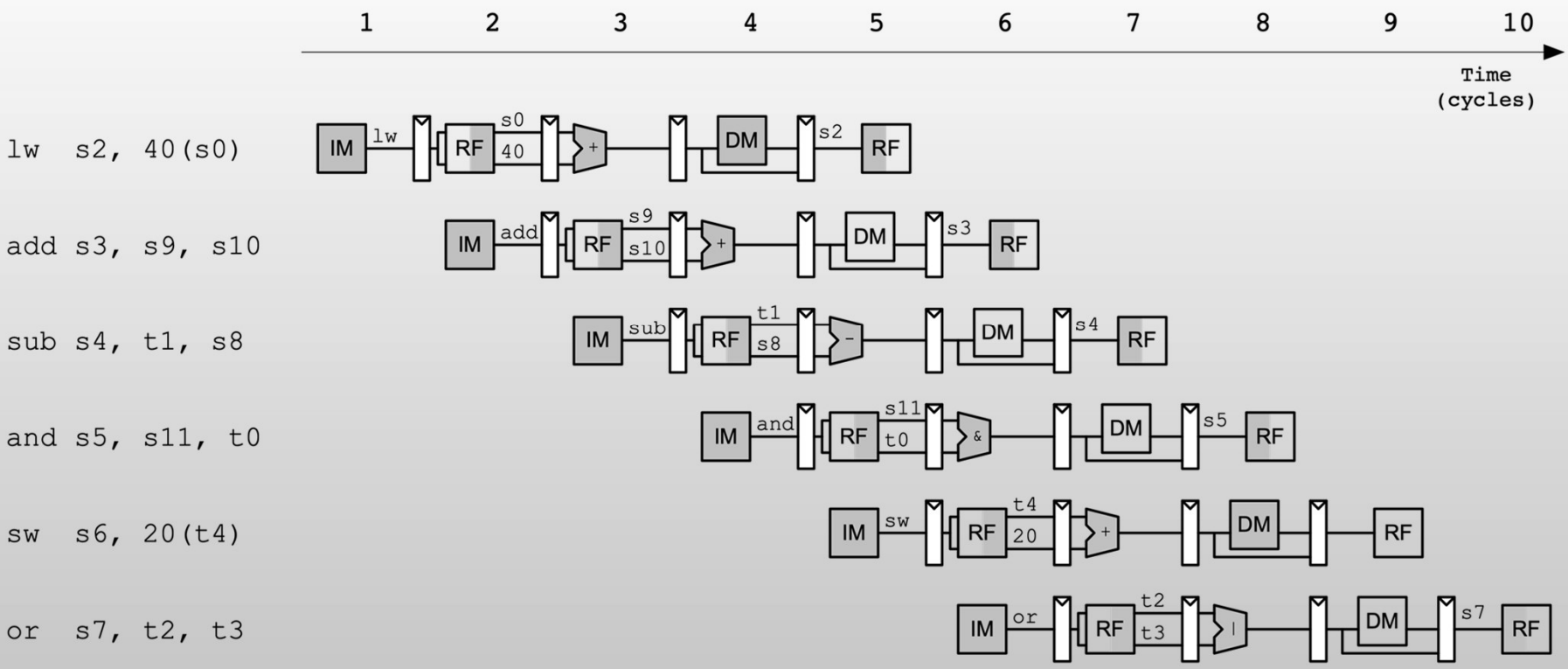
Each Stage

Each Stage

Each Stage



Pipeline CPU



Instruction Time

- Clock = Still controlled by slowest part
  - Average instructions per clock = 1 cycle though!
  - Significant improvement over prior…

Questions

- Why don’t commercial microprocessors use 100 pipeline stages if increasing stages allows for higher frequency?
- In the HDL version of the pipeline, how are the stages and pipeline registers usually organized?
- When designing hardware in HDL, are there cases where the code compiles with no error messages but the circuit still doesn’t work as expected? If so, what systematic methods do engineers use to debug and fix these kinds of “silent” bugs?
- How different do real CPU diagrams look from the RISC-V ones we have been studying in class?
- Which method of accounting for hazards is typically used in real pipelined processors, or is it problem dependent?

Next Time

- Studio!