CSE 260M / ESE 260
Intro. To Digital Logic & Computer Design

Bill Siever
&
Jim Feher

This week

- Thursday: Break!
- Hw #7 — Posted Tonight! (Due Friday after break)
- Tues (Dec. 3): Exam 2. IN ***Hillman 70***

Lecture Review

- Last week:
  - Compiler
    ```
    li a0,4096
    addi   a0,a0,954
    ```
  - Load Immediate (li) is a pseudo-op.
    - My summary was sloppy / incorrect
    - https://pages.hmc.edu/harris/ddca/ddcarv/DDCArv_AppB_Harris.pdf
  - Translates to two instructions: lui and addi. Exactly as shown

lui: Load Upper Immediate

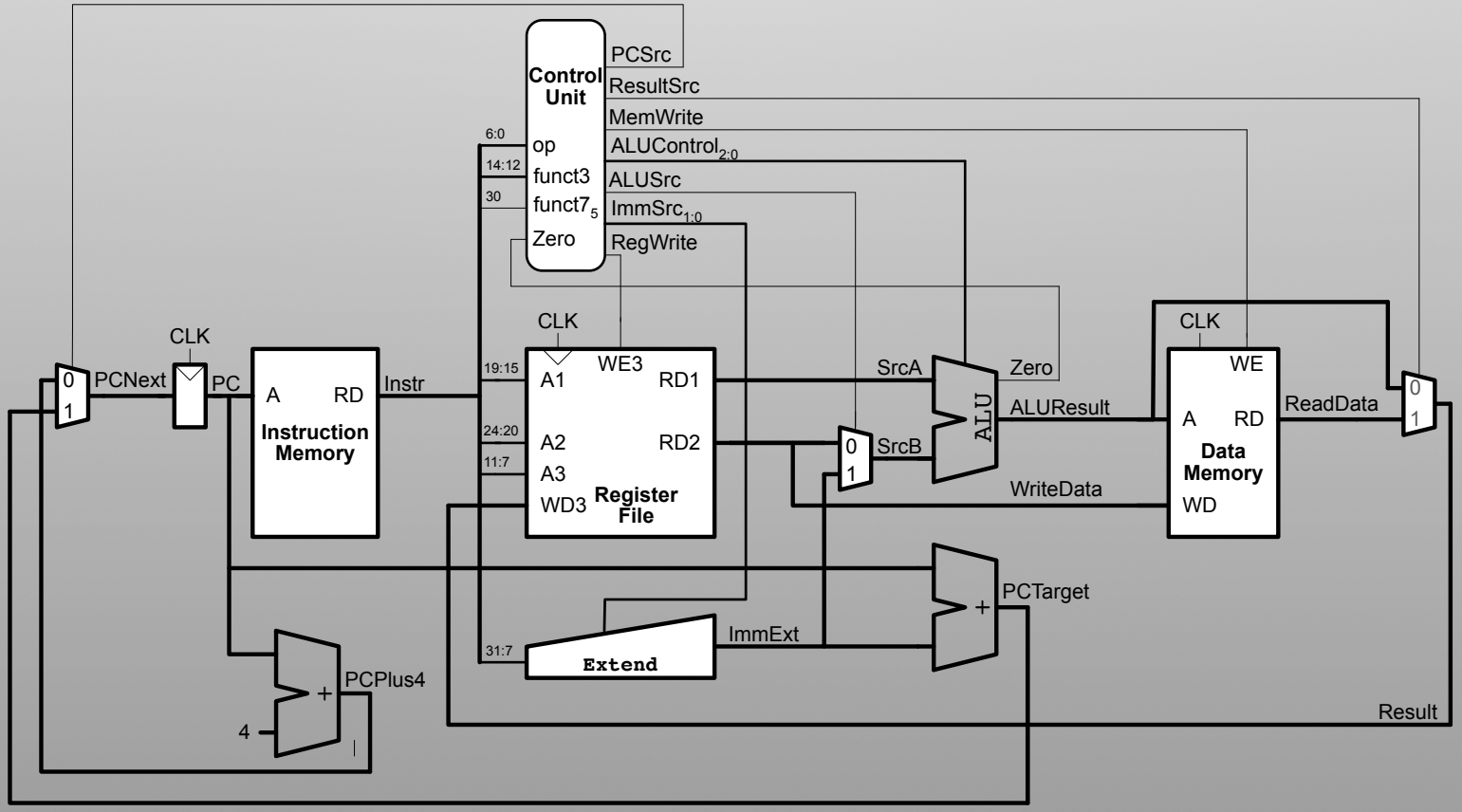
- Lui is really useful for working with larger values

Studio Review

- Display setup / features
- Quick test
- 7-segment display

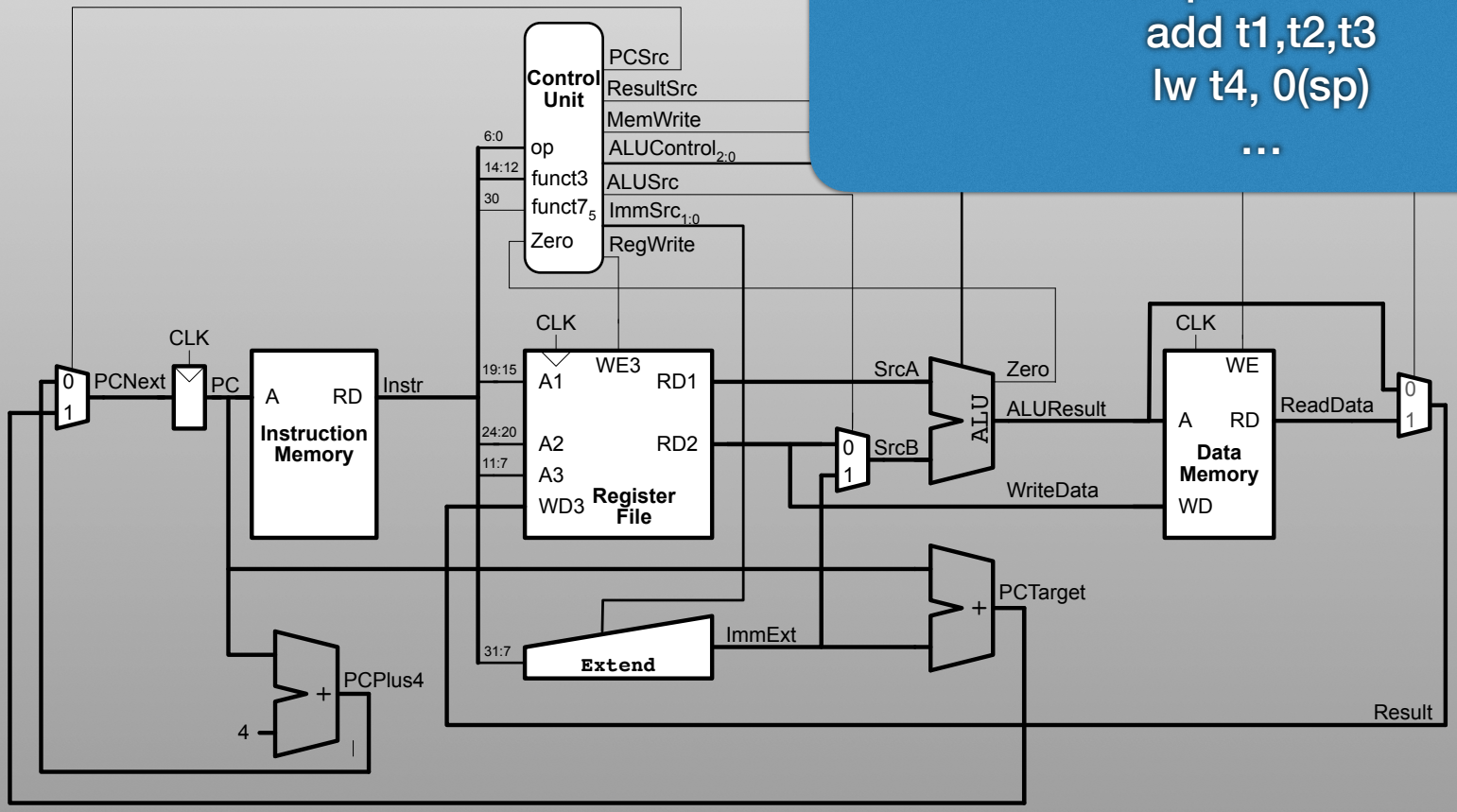
Chapter 7

Simple, Single-Cycle RISC-V Computer

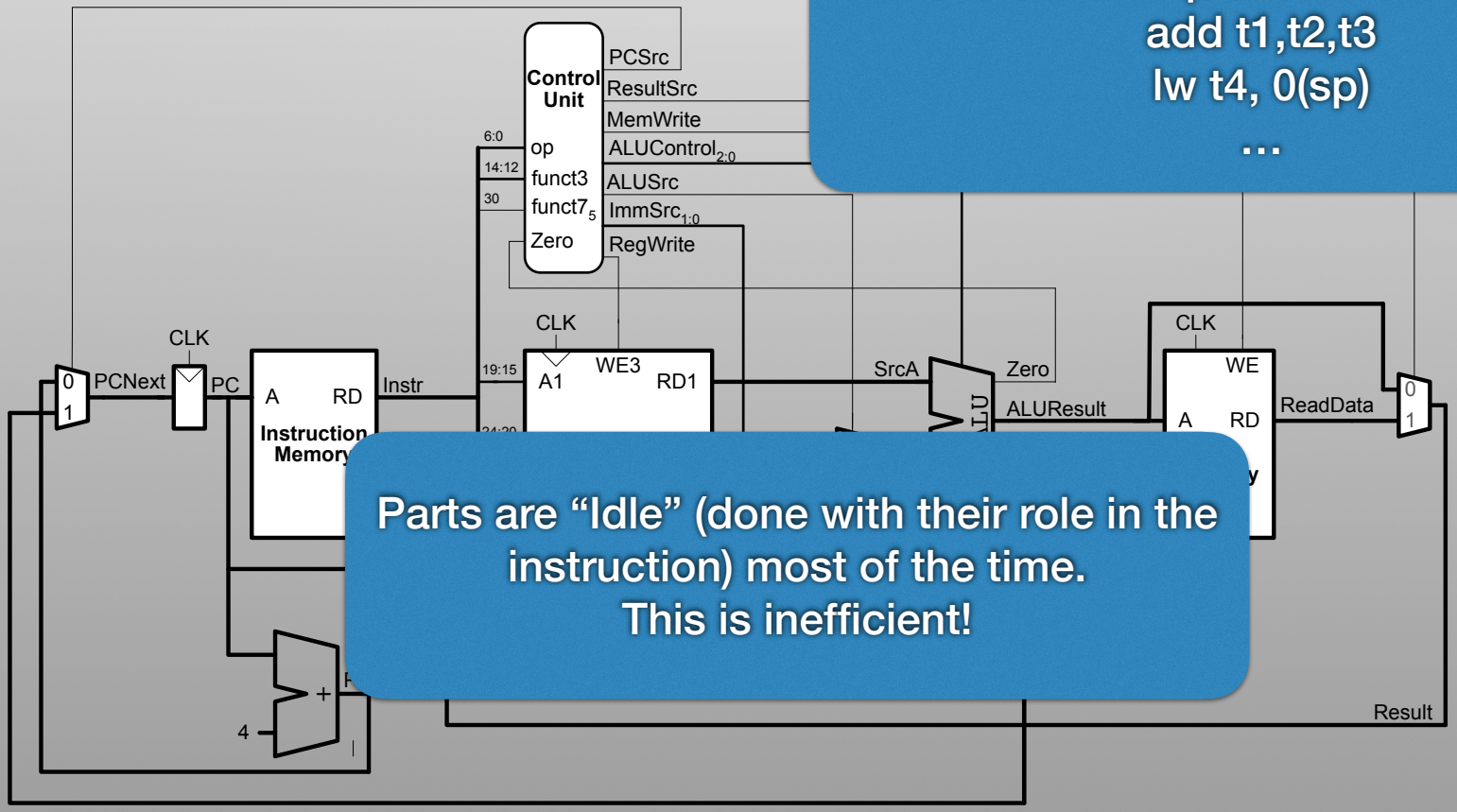


Simple, Single-Cycle RISC-V Computer

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



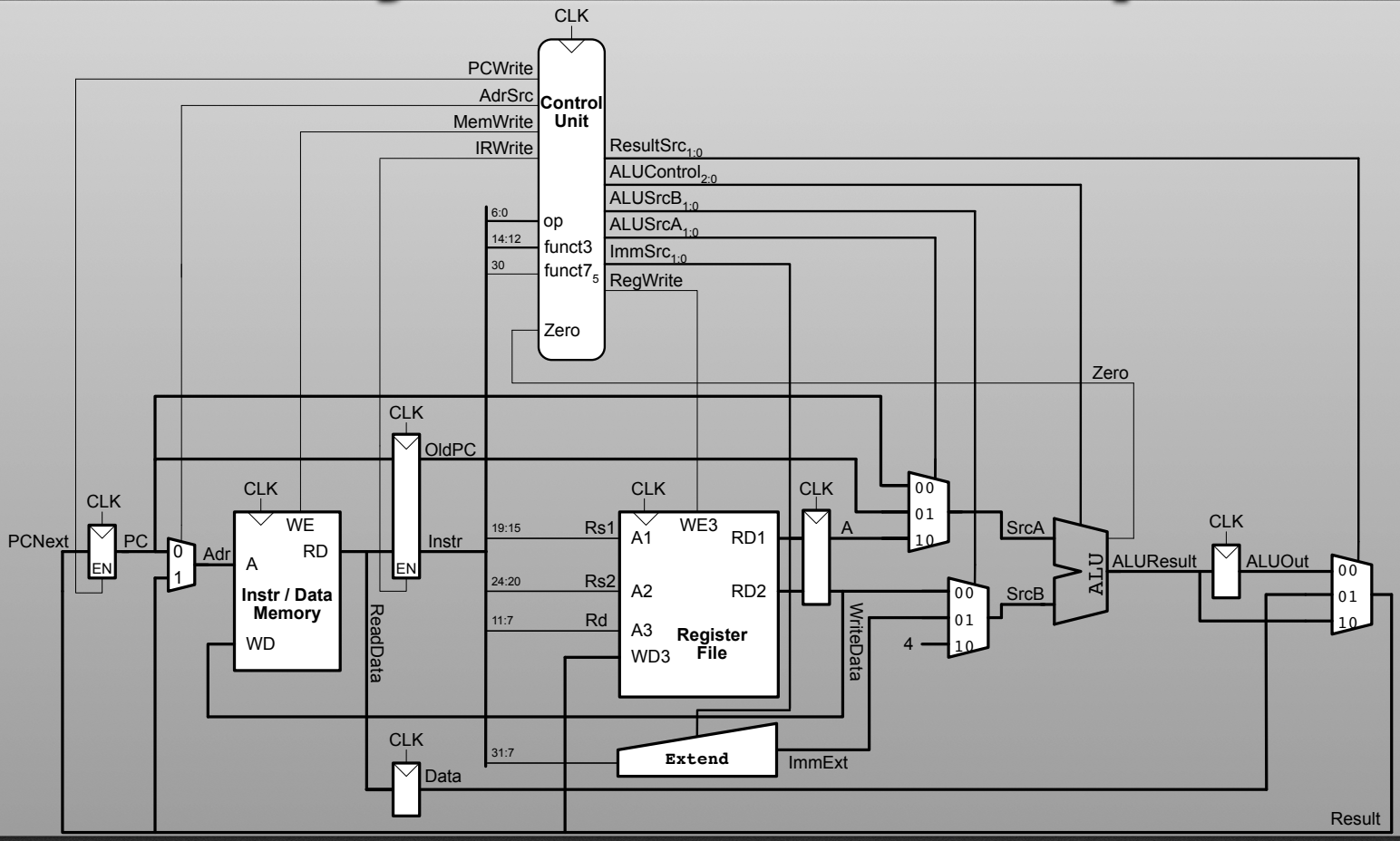
Simple, Single-Cycle RISC-V Computer



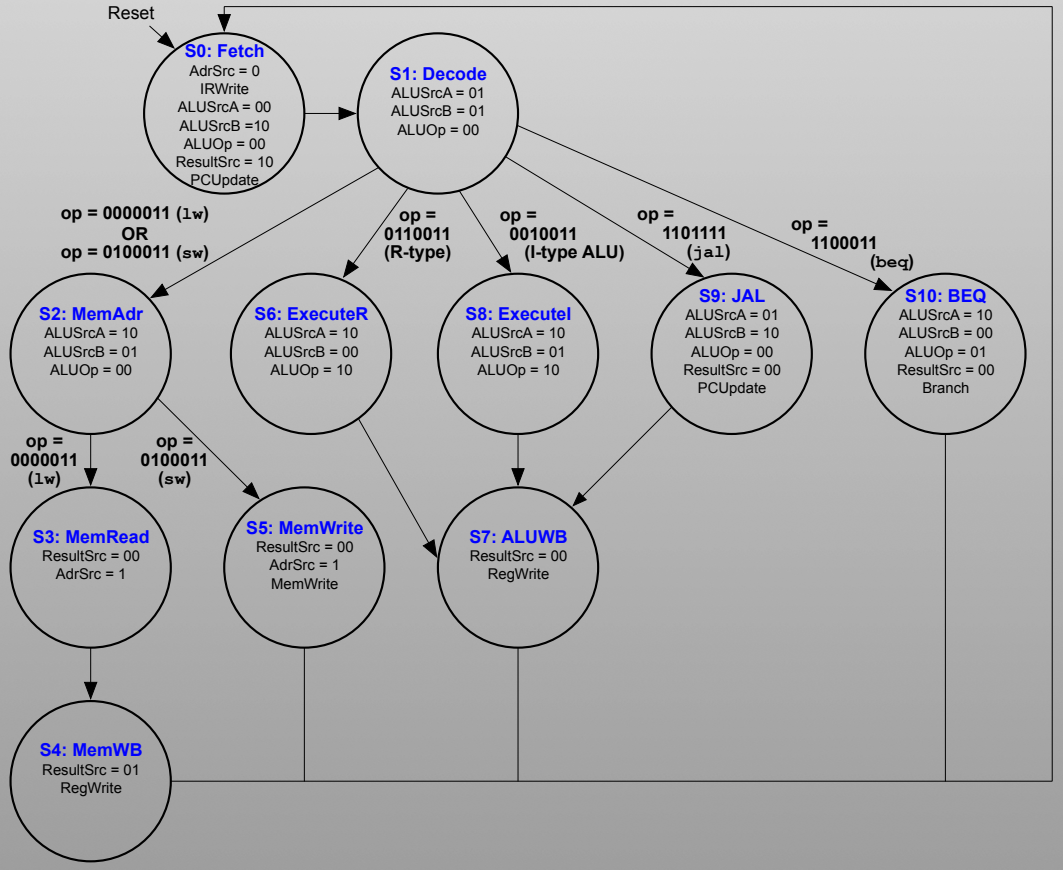
Consider a sequence of instructions:
add t1,t2,t3
lw t4, 0(sp)
…

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

Multi-Cycle RISC-V Computer



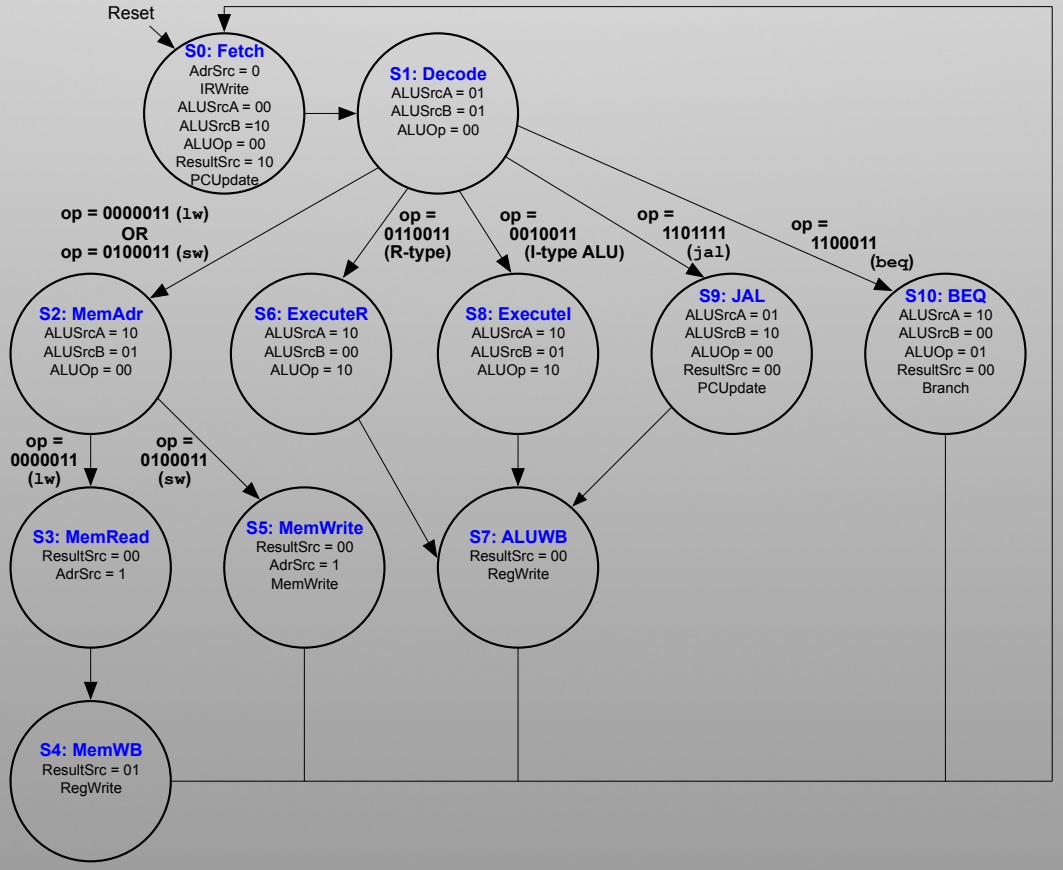
Process


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

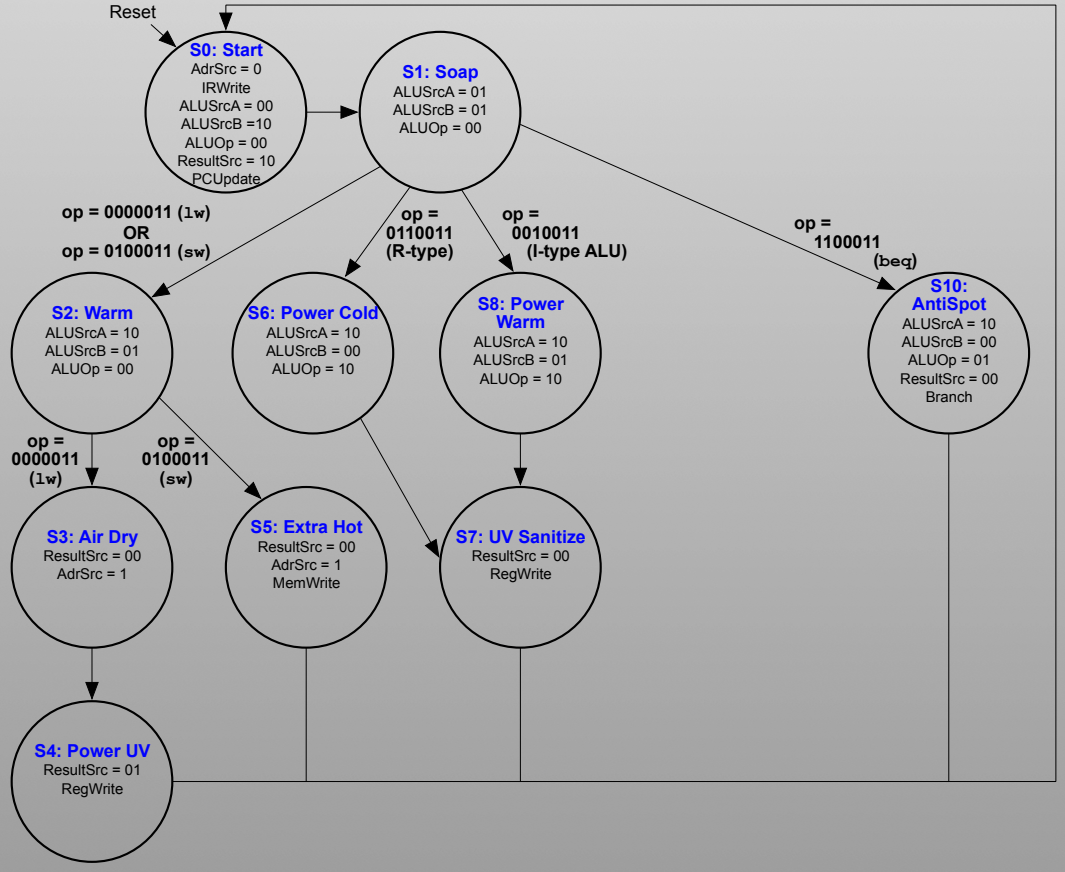
Pros/Cons of Multi-Cycle

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

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

Process: Hw 4 / Washer



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?

A Pipeline / Factory

- My career: Develop Medical Equipment
  - Along….

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

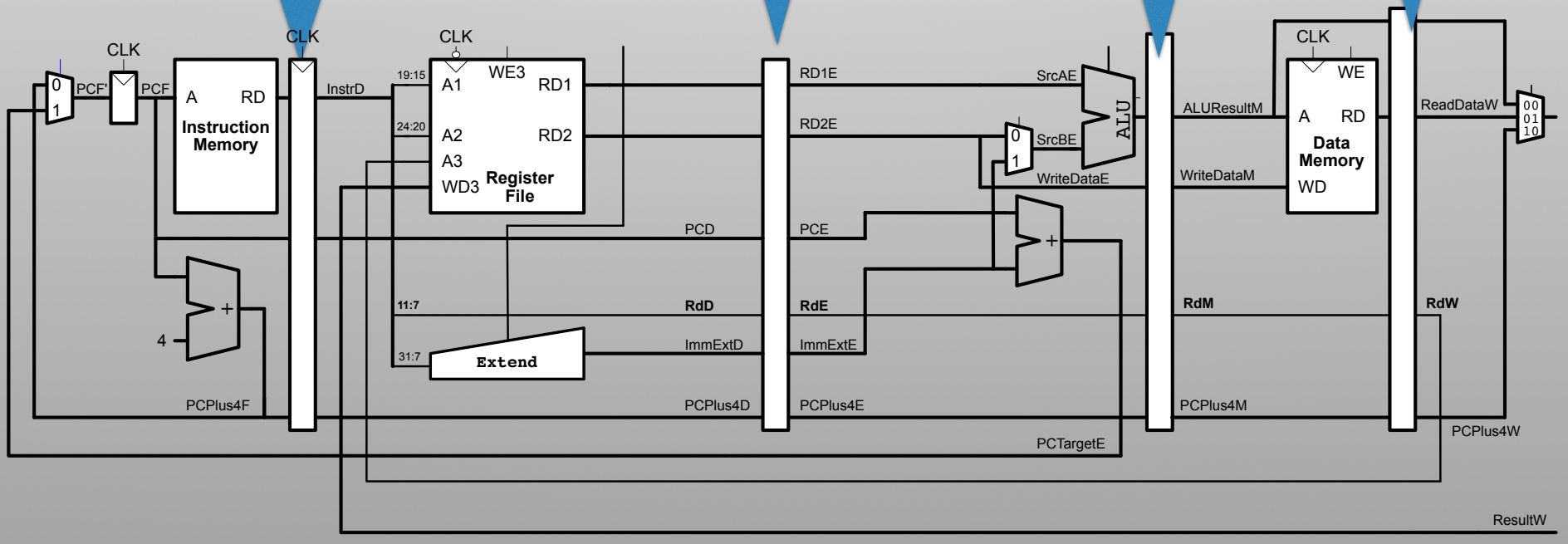
5.Move to shipping

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

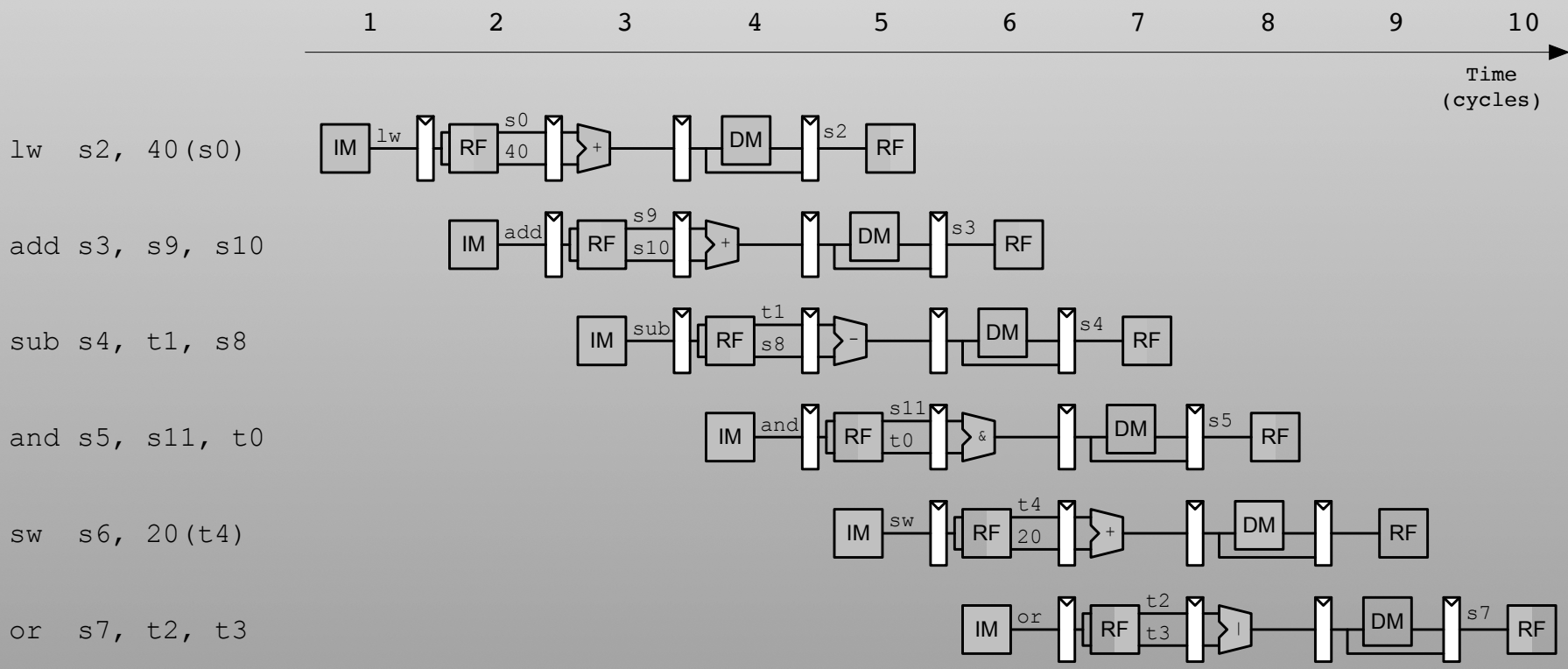
Each Stage

Each Stage

Each Stage



Pipeline CPU



Pipelining: A sequence of operations
(and overlapping a single “instruction” (load of laundry))
Execution overlaps in time

Next Time

- Studio
  - Bring full kits!