# CS61C Summer 2018 Discussion 6 – Single Cycle Datapath

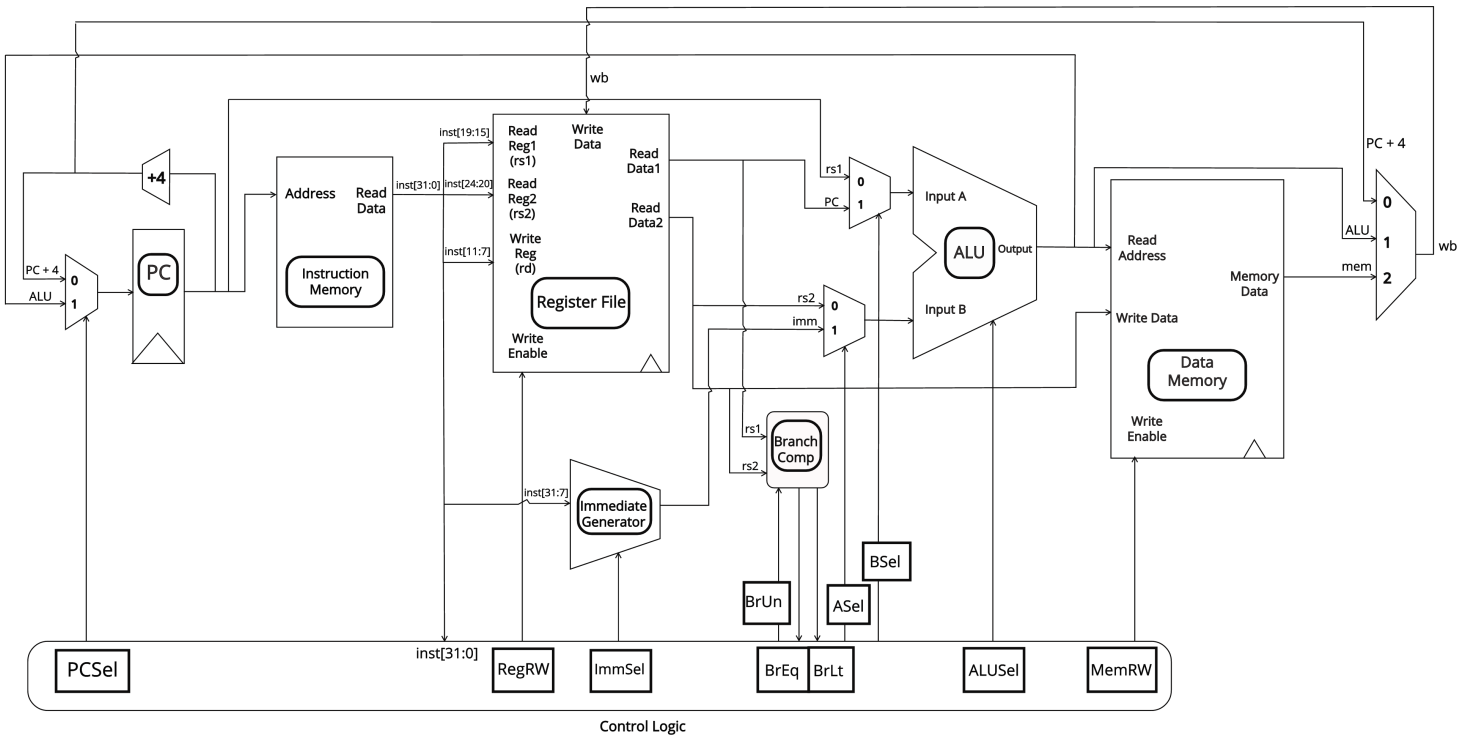
## Single Cycle CPU Design

1. Fill in each blank box with **round** edges on the datapath below with the name of the datapath component.

2. Name each datapath stage and explain what happens in that stage.

| Stage | Functionality |
|---|---|
| IF: Instruction Fetch | Send an address to the instruction memory, and read IMEM at that address |
| ID: Instruction Decode | Generate control signals from the instruction bits, generate the immediate, and read registers from the RegFile |
| EX: Execute | Perform ALU operations, and do branch comparison |
| MEM: Memory | Read from or write to the data memory |
| WB: Writeback | Write back the PC + 4, the result of the ALU operation, or data from memory to the RegFile |

3. Fill in each blank box with with **square** edges on the datapath below with the name of the control signal.




## Single Cycle CPU Control Logic

1. Fill out the following table with the control signals for each instruction based on the datapath on the previous page. Wherever possible, use * to indicate that what this signal is does not matter.

| | BrEq | BrLT | PCSel | ImmSel | BrUn | ASel | BSel | ALUSel | MemRW | RegWEn | WBSel |
|---|---|---|---|---|---|---|---|---|---|---|---|
| add | * | * | 0 | * | * | 0 (Reg) | 0 (Reg) | add | 0 | 1 | 1 (ALU) |
| ori | * | * | 0 | I | * | 0 (Reg) | 1 (Imm) | or | 0 | 1 | 1 (ALU) |
| lw | * | * | 0 | I | * | 0 (Reg) | 1 (Imm) | add | 0 | 1 | 2 (MEM) |
| sw | * | * | 0 | S | * | 0 (Reg) | 1 (Imm) | add | 1 | 0 | * |
| beq | 1/0 | 1/0 | 1/0 | SB | * | 1 (PC) | 1 (Imm) | add | 0 | 0 | * |
| jal | * | * | 1 | UJ | * | 1 (PC) | 1 (Imm) | add | 0 | 1 | 0 (PC + 4) |
| bltu | 1/0 | 1/0 | 1/0 | SB | 1 | 1 (PC) | 1 (Imm) | add | 0 | 0 | * |

# Single Cycle CPU Performance Analysis

## Clocking Methodology Overview

- A **"state element"** is an element connected to the clock (denoted by a triangle at the bottom). The **input signal** to each state element must stabilize before each **rising edge**.
- The **critical path** is the longest delay path between state elements in the circuit. If we place registers in the critical path, we can shorten the period by **reducing the amount of logic between registers**.

For this exercise, assume the delay for each stage in the datapath is as follows:

| Stage | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| Delay | 200 ps | 100 ps | 200 ps | 200 ps | 100 ps |

1. Mark the stages of the datapath that the following instructions use and calculate the total time needed to execute the instruction.

| | IF | ID | EX | MEM | WB | Total Time |
|---|---|---|---|---|---|---|
| add | X | X | X | | X | 600 ps |
| ori | X | X | X | | X | 600 ps |
| lw | X | X | X | X | X | 800 ps |
| sw | X | X | X | X | | 700 ps |
| beq | X | X | X | | | 500 ps |
| jal | X | X | X | | X | 600 ps |
| bltu | X | X | X | | | 500 ps |

2. Which instruction(s) exercise the critical path?
   Load word (lw), which uses all 5 stages.

3. What is the fastest you could clock this single cycle datapath?
   $\frac{1}{800}$ picoseconds = $\frac{1}{800*10^{-12}}$ seconds = 1,2500,000 $s^{-1}$ = 1.25 GHz

4. Why is the single cycle datapath inefficient?
   At any given time, most of the parts of the single cycle datapath are sitting unused. Also, even though not every instruction exercises the critical path, the datapath can only be clocked as fast as the slowest instruction.

5. How can you improve its performance? What is the purpose of pipelining?
   Performance can be improved with pipelining, or putting registers between stages so that the amount of conditional logic between registers is reduced, allowing for a faster clock time. (Learn more in Discussion 7!)