1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

1.1   The single cycle datapath makes use of all hardware units for each instruction.

1.2   It is possible to execute the stages of the single cycle datapath in parallel to speed up execution of a single instruction.

1.3   The auipc instruction and jump instructions (jal, jalr, and any pseudoinstruction) are the only instructions that set PC = PC + offset.

1.4   Storing instructions and loading instructions are the only instructions that actively require going to and from DMEM.

1.5   It is possible to use both the output of the immediate generator and the value in register rs2.

1.6   Combinational logic is only used in the instruction decode stage.

2   Single-Cycle CPU

2.1   For this worksheet, we will be working with the single-cycle CPU datapath on the last page.

(a) Explain what happens in each datapath stage, and which hardware units in the datapath are used.

   IF Instruction Fetch

   ID Instruction Decode

   EX Execute

   MEM Memory

   WB Writeback

(b) On the datapath, fill in each **round** box with the name of the datapath component, and each **square** box with the name of the control signal.

(c) List all possible signals that each control signal may take on for the single cycle datapath. Then mark which ones are actively used. If there are any non-used signals, write a short explanation for why it exists but is not used.

Signal Name	Values	Signal Name	Values
PCSel		RegWEn	
ImmSel		BrEq	
BrLt		ALUSel	
MemRW		WBSel	

2.2   Fill out the following table with the control signals for each instruction based on the datapath on the previous page. If the value of the signal does not affect the execution of an instruction, i.e. the correct execution will still occur, you may use the ∗ (don't care) symbol or write **all** the possible values (e.g. 0/1 for standard datapath signals, or 0/1/2/3 for the WBSel).

Original phrasing: Wherever possible, use ∗ to indicate that what this signal is does not matter (as in, letting the value be whatever it wants won't affect the execution of the instruction). If the value of the signal does matter for correct execution, but can vary, list all of the values (for example, for a signal that matters with possible values of 0 and 1, write 0/1).

	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add											
ori											
lw											
sw											
beq											
jal											
bltu											

2.3 **Clocking Methodology**

- 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. The circuit cannot be clocked faster than this, since anything faster would mean that the correct value is not guaranteed to reach the state element in the alloted time. If we place registers in the critical path, we can shorten the period by **reducing the amount of logic between registers**.

For this exercise, the delay for each circuit element is given as follows:

Clk-to-Q	RegFile Read	RegFile Setup	Mux
5ns	35ns	20ns	15ns

ALU	Branch Comp	Imm Gen	MEM Read	MEM Write
100ns	50ns	45ns	300ns	200ns

(a) Mark the stages of the datapath that the following instructions use

	IF	ID	EX	MEM	WB
add					
ori					
lw					
sw					
beq					
jal					

(b) Assume the RegFile setup and PC setup times are equivalent. Ignoring the length of a clock cycle, how long does it take to execute the instruction:

1. jal

2. lw

3. sw

(c) Which instruction(s) exercise the critical path?

(d) What is the fastest you could clock this single cycle datapath?

(e) Why is the single cycle datapath inefficient?

(f) How can you improve its performance? What is the purpose of pipelining?