The purpose of the Final Exam is to provide an experience for designing a single-cycle CPU with a specific instruction set, and then creating a pipelined version of that same CPU. The Datapath and Controllers are shown and redrawn in both cases, to account for the change due to the pipelining paradigm.

**Single Cycle Implementation**

The diagram below shows the Datapath and Controller interconnections for the CPU to be used in this exam, with the provided instruction set. All Datapath components are shown in **red**, and all signals to and from the Controller are shown in **blue**:



With this Datapath and Controller schematic, the provided instruction set will be able to be implemented as intended for the single-cycle CPU. All instructions happen in exactly one clock cycle.

**Controller (Verilog Code)**

It is important to note that this Controller is being implemented as a combinational circuit. Since this is a single-cycle CPU, state transitions will not be handled by the Controller, but by the outside memories external *clk*. This implies that the Controller is simply made up of a procedural *always* statement, which is invoked when the *opcode* changes from the current instruction received.

The code for the Controller of this CPU is shown below, and is also provided in the ZIP file for this Final Exam: *Controller.v*

// Carlos Lazo

// ECE505

// Final Exam

// Controller of 16-bit CPU

`timescale 1ns/100ps

module Controller (

input [2:0] opcode,

input zero,

output reg add, enPC, ld\_r0, rd\_mem, pc\_src, r0\_src, swap\_reg, sub, wr\_mem);

always @(opcode) begin

// Initialize all outputs to 0 to prevent latches:

add = 0; enPC = 0; ld\_r0 = 0; rd\_mem = 0; pc\_src = 0;

r0\_src = 0; swap\_reg = 0; sub = 0; wr\_mem = 0;

case (opcode)

3'b000: begin // LDR

enPC = 1; rd\_mem = 1; r0\_src = 1;

end

3'b001: begin // STR

enPC = 1; wr\_mem = 1;

end

3'b010: begin // ADD

enPC = 1; add = 1; ld\_r0 = 1;

end

3'b011: begin // SUB

enPC = 1; sub = 1; ld\_r0 = 1;

end

3'b100: begin // JMP

pc\_src = 1; enPC = 1;

end

3'b101: begin // JEZ

if (zero)

pc\_src = 1;

enPC = 1;

end

3'b110: begin // SWP

enPC = 1; swap\_reg = 1;

end

3'b111: begin // HLT

enPC = 0;

end

default: enPC = 0;

endcase

end

endmodule

Note that since only the Controller code is shown, all signals asserted here are assumed to have the desired effect on all Datapath components, as far as proper CPU operation is concerned.

**Pipeline Implementation**

The diagram below shows the Datapath and Controller connections for the CPU to be used in this exam, now with a pipelining structure. Pipeline registers have been added into the Datapath, as they are necessary in order for all instructions to carry through correctly. All Datapath components are shown in **red**, and all signals to and from the Controller are shown in **blue**:



In the pipelined version of this CPU, there are a total of **5** cycles for a full-instruction to resolve in through the entire DataPath. There are a total of **4**pipeline registers utilized in this design, with descriptions of functionality and sizes listed below. Note that the size listed below encapsulates both the data portion of the pipeline register, along with the control portion.

1. **IF/ID Register**

**Size** = 16 bits for opcode + memory address / constant

* Receives instruction from the Instruction memory at a given *addr*

1. **ID/EX Register**

**Size** = 13 + 13 + 13 + 13 + 8 = 60 bits

- 13 bits for both R0 and R1

- 13 bits to pass along memory address / constant

- 13 bits to pass along current *PC* count

- 8 bits for all Controller signals propagated through pipeline

* Decodes *opcode*, propagates current *PC* count, instr, register values, and control signals

1. **EX/MEM Register**

**Size** = 13 + 13 + 13+ 6 = 45 bits

- 13 bits for ALU result

- 13 bits to pass along memory address / constant

- 13 bits to pass along current *PC* count

- 6 bits for propagate remaining Control signals

* Stores ALU output, propagates current *PC* count, instr, and control signals

1. **MEM/WB Register**

**Size** = 13 + 13 + 3 = 29 bits

- 13 bits for information read from Data Memory

- 13 bits to pass along memory address / constant

- 3 bits for propagate remaining Control signals

* Passes along information from Data Memory, instr, and control signals

These registers are each unique and must exist for the pipelining to function correctly. Writing into the registers happens with each rising/falling edge of the clock, depending on the CPU implementation.

The following table below depicts the operation of all the different control signals at each step of the pipeline. It showcases the different control signals that are set in order for the CPU to function correctly and for all registers to be updated correctly:



All signals in **blue** above refer to controller signals. R0/R1/IR/PC are the different data registers within the Datapath, and the [#1]/[#2] notation depicts the where information is being stored into pipeline registers, depicted in **red**. **NOP** in the WB (Write Back) stage implies that the final clock cycle was not needed when encountering that particular instruction.

This concludes the analysis for the Final Exam.