Single-cycle CPU: Implementation of Loop Instruction

Take the Single Cycle datapath posted on the web, and modify it so that it can also execute the instruction

\texttt{loop r1, r2, offset}

This is a branch instruction that increments register \texttt{r1}, and compares it to a loop bound of \texttt{r2}. If these two values are not equal then the PC is set to \texttt{PC + offset}. The instruction uses the Immediate MIPS instruction format. This instruction has the same effect as sequentially executing the following two instructions on the MIPS architecture:

\texttt{addi r1, r1, 1}
\texttt{bne r1, r2, offset}

For this question do NOT modify the instruction memory, data memory, or the register file. You can only modify/add control lines, MUXs, ALUs, and data path lines. To answer this question (1) give the above sequence of instructions using the rs, rt, rd, and immediate fields from the immediate format, (2) draw the parts of the datapath that have changed, and (3) give the state for the control for this instruction in the modified datapath. For the state show the values for all the control lines and any MUX in the original Single Cycle Datapath along with any new control lines added for this problem.
**Multi-cycle CPU: Implementation of MemIndAdd Instruction**

Take the multi-cycle datapath figure posted on the web, and modify it so that it can also execute the instruction

**MemIndAdd r1, offset(r2)**

This instruction calculates the address M[offset + r2], which is a pointer back into memory. It then loads the value stored at address M[offset + r2], and adds it to r1, storing the result back into r1. The instruction uses the Immediate MIPS instruction format. This instruction has the same effect as sequentially executing the following code:

\[
\text{tmp} = \text{Memory}[\text{offset} + \text{r2}] \\
\text{tmp} = \text{Memory}[\text{tmp}] \\
r1 = r1 + \text{tmp}
\]

For this question, do not add any new ALUs, do not modify the instruction and data memory, and do not modify the register file. In addition, do not add any registers to the register file or temporary registers to the data path. You can only modify/add data paths, control lines, and MUXs. To answer this question give

(1) How would you encode the instruction in I-format?

(2) Modify the multiple cycle processor to execute the new instruction, and

(3) Provide the finite state machine (FSM) for the control for this datapath. For the finite state machine show the values for the control lines and all MUXs in the figure shown on the last page. For the ALU control line, just give the ALU operation for each cycle. Show the important control line and MUX values for every cycle (starting with the fetch cycle). In addition, show how many cycles it takes to execute the MemIndAdd instruction with your FSM.