Rushi Patel
Lab3 Description

Multicycle CPU

Using the notes provided for a multicycle CPU, the datapath is very close with a few exceptions due to the lack in Jump instructions and the branch on equal function.

Major Blocks:

Control

The control block contains all the logic needed to progress through each state of the moore state machine. There are a total of 9 states:

1) fetch - used to get the next instruction and increment program counter by one
2) decode - decide which next state based on instruction and retrieve values from register file
3) alu_mem_location - for Store and Load instructions hold the memory location in alu register
4) mem_read - for a load read the memory and store data in data register
5) reg_mem_write_back - write data from data register into register file
6) mem_store - read register file and store into memory based on location in alu register
7) execute_alu_rtype - for any alu operations rtype or itype perform alu operation and store result in alu register. (Name may be misleading)
8) write_back_rtype - write result into destination register. (Name may be misleading)
9) alu_branch - for branch operations, perform check and write new PC if needed.

ALU

The ALU is very similar to the previous lab, however some of the functions needed to be updated.
List of operations:

1) NOOP - Output 0
2) MOV - Output A
3) “MOV B” - Output B (This is useful for LI, LWI, and SWI instructions)
4) ADD - A + B
5) SUB - A - B (Used for both Sub and BEQ to check for equal status)
6) OR - A | B
7) AND - A & B

The addition of a zero bit was needed for branch instructions. If the output is equal to zero, then set the zero bit to a one.

Register File

The register file was a direct copy from the previous lab.

Memory File
The memory file contains 65536 entries each 32bits long. Since there were no constraints on the memory functionality, I made an asynchronous read and clock triggered write. For testing purposes I had to implement the TA method of filling the memory using 3 input wires. However for my personal testing I used an “initial” block which took the contents of a memory text file and filled the memory block using “$readmemb”.

2/4 input mux
The 2 input mux was copied from the previous lab, however the 4 input mux was created. Select is a 2 bit input which chooses 1 of 4 possible inputs A_0, B_1, C_2, D_3.

32 bit Flip Flop w Reset and Enable
The 32bit FFwReset from the previous lab was adapted to have an enable line which indicates whether to latch the input on the next clock positive edge.