Project 6 Report

What did you learn from the project?
We learned a lot from this project. We can list it as follow:

a. Design various function module in Verilog. We practice our Verilog design skill.

b. Learned different architecture of CPU and how these architectures work.

c. Learned how assembly language and machine connected with each other.

d. Learned how to down load Verilog code to FPGA board and make it work.

e. Since we did the whole project as a team, we practiced our communication ability and build up our team work spirit.

What would you do differently next time?
If we have another chance to do it, we will take synchronous read into consideration. We can improve the design. For example we can reduce the execution time of branch instruction to two cycles. Or we can try pipeline architecture next time. Besides, we have some redundant instructions in our instruction set, for example, branch equal and branch not equal, shift left logic and shift right logic. So, we can implement some other useful instruction.

What is your advice to someone who is going to work on a similar project
I think there is a big problem with this project. The whole multi-cycle datapath is based on asynchronous read and we implement the datapath under the assumption that the main memory has an asynchronous read port. However, the memory we got from the library has synchronous read function. This would cause big trouble. In this multi-cycle datapath, the PC signal updates in the first cycle for branch and jump instructions, but the transition of PC signal comes a little later than positive edge of clock. If we have an asynchronous read function memory, the datapath will work properly. But if the memory has a synchronous read function, the value of PC should be updated and stable before the positive clock transition. So we have do redesign our datapath, this take so much time that we have very little time to do the last project.

What we do to avoid this problem is simply add extra clock cycle to jr, jal, beq, bne, lw and sw to get new PC value stable before we go into the next instruction so that we will get the right result at the output of synchronous read memory. Another thing we can do to avoid this problem is we can add another input to the mux whose output is connected to the input address of memory and the extra input of the mux comes from output of ALU. When the datapath executes the last cycle of instruction jr, jal, beq, bne, lw or sw, the control unit activate the ALU output-to-memory path so that we don’t have to wait for another cycle to get new memory address value into PC. Thus the problem is sloved.