EECE 3324
Summer 2019
Assignment 7

Delayed branching for *MIPS* will be added to complete the 5-stage CPU pipeline as shown below from our textbook, *Patterson and Hennessey*. Figure 1 shows an overview design of how the **CPU should look like**. Figure 1 is used to give you a visual/an overall idea of how the design should look like without all the little details. **Your job is to fill in all the little details**. Your design needs to run at a 20 MHz clock rate (a full clock cycle is 50 nsec. and a half cycle is 25 nsec.). This time, you will need the equality detection logic and the extra adder to compute the branch address. Forwarding logic and hazard detection logic will NOT be needed for this assignment.

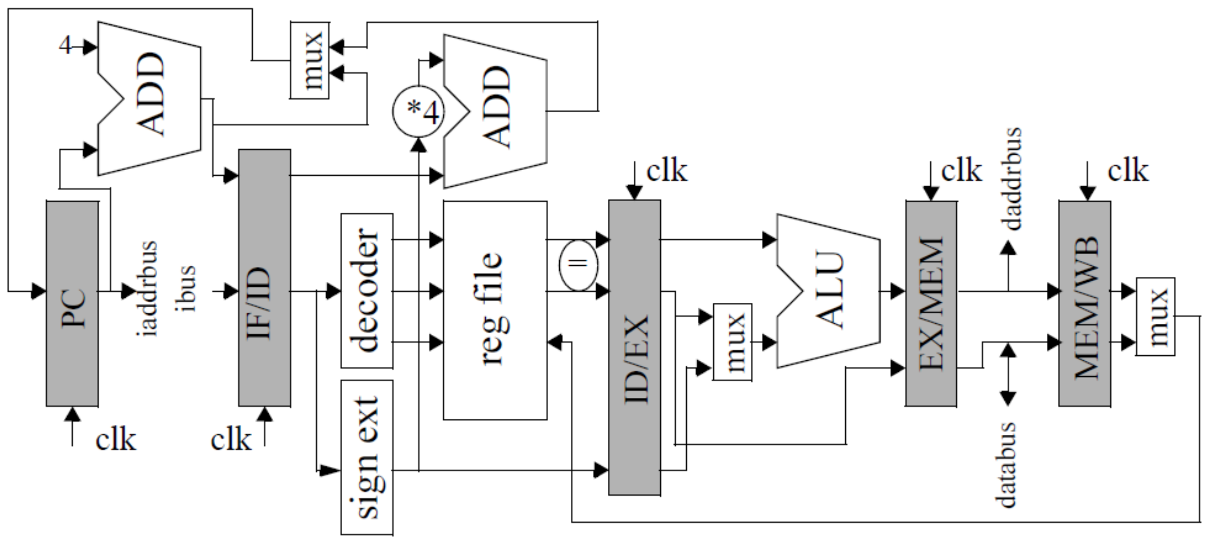


Figure 1. Overview of 5-stage pipeline

Design decision: Now, you need to make a judgement call on how to proceed. You can use your cpu4.v from Assignment 6 as a starting point or you can start from scratch. You are the designer, not the course instructor.

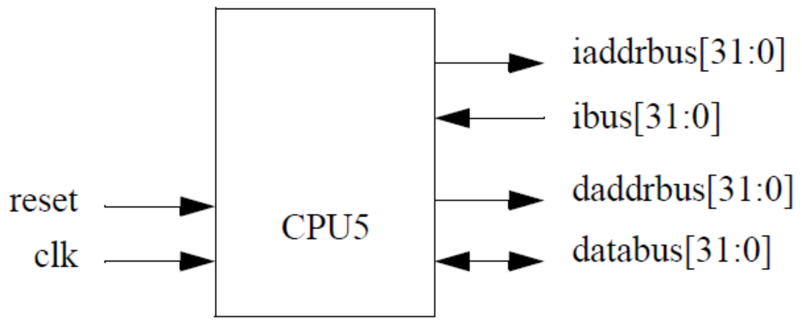
You will need to add/modify the decode logic so that the following instructions are implemented. These op codes include branch and set instructions but are otherwise the same as assignment 6.

Code	Function	Name	Format
000011		ADDI	I
000010		SUBI	I
000001		XORI	I
001111		ANDI	I
001100		ORI	I
011110		LW	I
011111		SW	I
110000		BEQ	I
110001		BNE	I
000000	000011	ADD	R
000000	000010	SUB	R
000000	000001	XOR	R
000000	000111	AND	R
000000	000100	OR	R
000000	110110	SLT	R
000000	110111	SLE	R

Table 1. Subset of instruction codes to be implemented

The instruction formats follow MIPS except that only the I and R formats are implemented, and the logic operations are slightly different. All other op-codes should be treated as NOP's.

As before, the instruction cache and data cache will not be modeled. Instead the instruction and data busses will be made available as external terminals and the test bench will simulate the behavior of the memory. Your CPU design should have the terminals shown below.



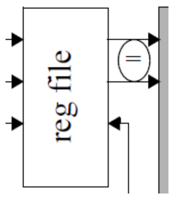
Note that the databus must be bi-directional.

The reset input should clear the PC (set the PC register to all 0's). It is acceptable for all the other registers to remain undefined until something is clocked into them.

Design Hints/Strategy:

1. The instruction address must be produced by your CPU. This requires your CPU to put the next instruction address onto the iaddrbus at the beginning of each clock cycle. To keep the hardware simple, a one cycle delayed branch will be used (no branch stalls required). That is, the instruction after the branch is always fetched. The compiler is assumed to insert another useful instruction after the branch or a NOP otherwise. This has been used in real MIPs processors.

2. The WB stage must be disabled for the branch instructions (BEQ and BNE). The easiest way to do this is to change the destination register to R0. (Quiz time: What happens to register file if you do not do this?)
   What happens to the CPU when the instruction is:
   BEQ and branch is not taken
   BEQ and branch is taken
   BNE and branch is not taken
   BNE and branch is taken



3. The comparisons for the set instructions can be done with an ALU subtract operation. Then you can use the status outputs from the ALU. If Z is the output of the zero detector and C is the carry out of the ALU, then set (to 1) bit 0 (the least significant bit) in the destination register when the following conditions are true.

instruction	**set bit 31**
SLT	not C and not Z
SGT	C and not Z
SLE	not C or Z
SGE	C or Z

   Otherwise, put all zeroes in the destination register. Note, these comparisons are for unsigned numbers. Different logic would be needed to compare signed numbers.

4. The equality detector on the register file outputs controls the PC input MUX. The delay of the register file, the equality detector and the MUX must be less than half a clock period to insure that the PC gets the correct value.

Draw All the interconnections using the information you have gathered from Figure 1 and Design Strategy.

Do not assume anything. DO NOT start writing Verilog code until you fully understand the concepts behind this assignment. Doing so may not be a wise thing to do.

A Verilog testbench file to provide inputs to your design will be provided by your course instructor. A similar file may be used to grade your design. Your highest level Verilog file needs to be named **cpu5.v** to be compatible with the testbench file.

Expand the testbench file so that it covers at least 20 more test scenarios.

Rules of Engagement:

1. You are only allowed to complete this assignment individually.
2. You are NOT allowed to work/collaborate/show your work to other people.
3. Your work must be your own. Should you be found guilty of using someone else's work (either full or partly), an "F" will be assigned as your overall course grade for the semester.
4. You must submit all your files and report to BB.
5. Read, understand and follow NEU Honor Code.
6. You are only allowed to turn in your work after you have demonstrated your work to your course instructor or grader. Failure to demonstrate your work to your course instructor or grader will result in a zero as the final grade of this assignment.
7. All work must be done using Xilinx Vivado software. Failure to do this will result in a zero as the final grade of this assignment.
8. **The use of * (star symbol) in Verilog is strictly prohibited**. Using a * (star symbol) will result in a zero as the final grade of this assignment.
9. Your course instructor will only help you AFTER you have completed a table that lists all the ports you use for this assignment along with their corresponding values for every clock cycle and drew all the interconnections derived from Figure 1. You need to look at the test bench file and propagate the signals accordingly.

Lab Submission

You must show a live **working demo** to your course instructor to receive a grade.

Upload all the Verilog files used and the improved test bench file to Blackboard. Make sure that your Verilog files (including the test bench file) are heavily commented as it is part of the grading. Your work will also be judged by the complexity of your modified test bench file. The more complex and rigorous your test bench file in testing different scenarios, the more points you earn.

Do not zip all your files together.

An automatic zero will be assigned for anyone who fails to show a live working demo to your course instructor.