EECE 3324
Summer 2022
Assignment 6

Add the **data memory interface** to the design of Assignment 5 to make it a 4-stage pipelined 32-bit CPU capable of doing load and stores as shown below. 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 small details**. Your overall design needs to run at a 20 MHz clock rate (a full clock cycle is 50 nsec. and a half cycle is 25 nsec.).

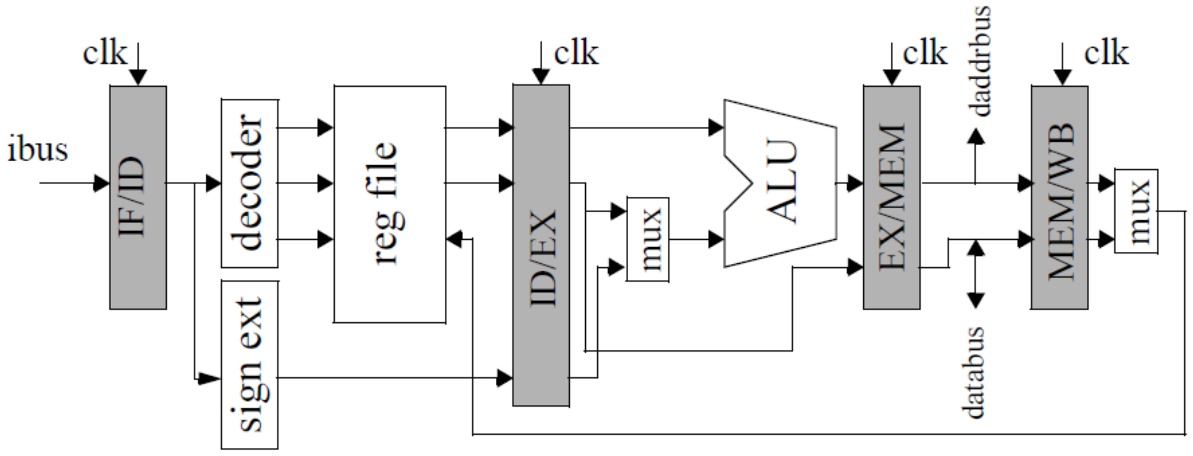
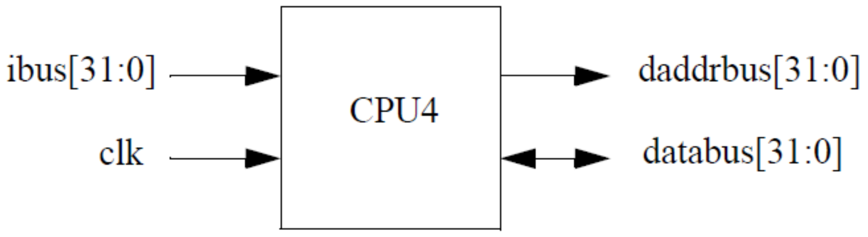


Figure 1. Overview of 4-stage pipeline

To facilitate testing of your design, the data cache will not be modeled. Instead the 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 (not the same as the dbus) must be bi-directional.

Design decision: Now, you need to make a judgement call on how to proceed. You can use your cpu3.v from Assignment 5 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 are the same op codes as Assignment 5 except for the load and store instructions.

Code	**Func**	**Name**	**Format**
000011		ADDI	I
000010		SUBI	I
000001		XORI	I
001111		ANDI	I
001100		ORI	I
011110		LW	I
011111		SW	I
000000	000011	ADD	R
000000	000010	SUB	R
000000	000001	XOR	R
000000	000111	AND	R
000000	000100	OR	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 NOPs.

Design Hints/Strategy:

1. The ID, EX and WB stages are the same as in Assignment 5, except that the b-operand from the register file must be passed through the EX stage into the EX/MEM pipeline register to drive the databus for **stores (SW)**. (Quiz time: what is the content of b-operand for stores?)

2. The WB stage must be disabled for stores. The easiest way to do this is to change the destination register to R0. (Quiz time: why do you want to do disable WB stage for stores?)

3. The controller (in ID) must be redesigned to pass control signals for MEM through the ID/EX and EX/MEM pipeline registers and pass control signals for WB through the ID/EX, EX/MEM and MEM/WB pipeline registers. Control signals for EX should be the same.

4. **Use tri-state buffers to control the bi-directional databus** so that a store instruction drives the databus with the value in the EX/MEM pipeline register. A load instruction should leave the databus floating so that the data memory can drive the databus and the value gets clocked into the MEM/WB pipeline register. The databus must be left floating when it is not used by load or store instructions.

5. To avoid timing problems in the behavioral simulation, do not connect pipeline register outputs directly to pipeline register inputs. For example, in the MEM stage the following assignment statement can be used to buffer the EX/MEM pipeline register outputs before using them as inputs to the MEM/WB pipeline register.

   ```
   assign mem_wb_in = ex_mem_out;
   ```

   This statement causes the synthesizer to insert extra buffers to slow the signals which prevents hold time violations.

6. Review the concept of Dselect from Assignment 5 and its relationship to the number stages of pipeline.

Before you start designing the hardware as shown in Figure 2 using Verilog, you need to:

1. **Draw the interconnections** of pipeline block diagram for this assignment in great details. You can find an example of it in Assignment 5 - Figure 2.

   You must use a software such as Microsoft Word or Power Point to draw the interconnections. Using your own handwriting will not earn you any points.

2. **Fill out the table** on the next page and find the value of the wires and output ports at each clock cycle. The purpose of filling out the table is to check your basic understanding of the hardware. If you believe the output is undefined, then use Xs to denote it, for example 32'hxxxxxxxx.

   You must use a software such as Microsoft Word or Microsoft Excel to fill out the table. Using your own handwriting will not earn you any points.

Submit your drawing for part 1 and the table for part 2 as part of your final submission to Canvas.

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.

Clock Status	ibus	Asel	Bsel	abus	bbus	Imm	S	Cin	daddrbus	databus	Dsel	mux2 select	dbus
clk = 0	32'b0000110000010100 1111111111111111												
↑	(same as above)												
clk = 1	(same as above)												
↓	(same as above)												
clk = 0	32'b0000110000010101 0000000000000001												
↑	(same as above)												
clk = 1	(same as above)												
↓	(same as above)												
clk = 0	32'b0000110000010110 0000000000000010												
↑	(same as above)												
clk = 1	(same as above)												
↓	(same as above)												
clk = 0	32'b0111101010011000 0000000000000000												
↑	(same as above)												
clk = 1	(same as above)												
↓	(same as above)												
clk = 0	32'b0111101010111001 0000000000000000												
↑	(same as above)												
clk = 1	(same as above)												
↓	(same as above)												
clk = 0	32’b0111111011010100 0001000000000000												

A Verilog testbench file to provide inputs to your design will be provided by your course instructor. A similar file will be used to grade your design. Your highest-level Verilog file needs to be named **cpu4.v** to be compatible with the testbench file.

Create your own the testbench file that fully tests your design with at least 32 test scenarios. You must provide comments on your testbench file so that others can follow your work. You will post your testbench to Assignment 6 Discussion on Canvas. Your classmates will review your testbench and provide feedback about it. Your classmates will also grade/rate your testbench file.

Please submit all your Verilog files including your new testbench file to Assignment 6 Link on Canvas. For your initial post to Assignment 6 Discussion, you only need to post your new testbench file.

Rules of Engagement:

1. You are only allowed to complete this assignment individually/as a group as assigned by the course instructor.
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. Read, understand, and follow NU Honor Code.
5. After you have completed your work, you must submit all your files to Canvas. Once you have submitted all your code to Canvas, you then need to be live interviewed by your TA/course instructor to verify all your work for this assignment using Xilinx Vivado. Failure to demonstrate your work to your grader or course instructor will result in a zero as the final grade of this assignment.
6. 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.
7. **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.

Lab Submission:

You must show a live **<u>working demo</u>** to your course instructor or grader to receive a grade. The demo must be completed individually/as a group as instructed by the course instructor. Questions will be asked during the demo. Points will be deducted from your Lab grade should you fail to answer any questions. Comments such as "I cannot remember why I did that" will not help you. Come prepared.

Upload all the Verilog files used and the improved test bench file to Canvas. <u>Make sure that your Verilog files (including the test bench file) are heavily commented as it is part of the grading</u>. 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.

<u>Do not zip all your files together</u>.

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