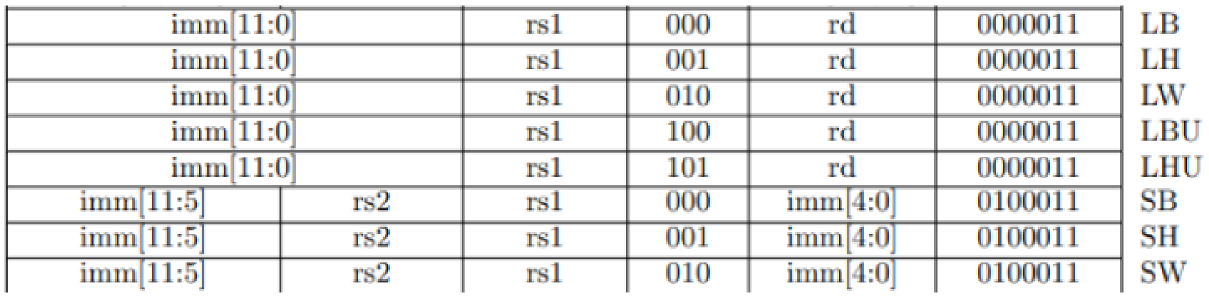
Lab 6

Now we will implement the Load-Store instructions



imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

CPU will have an additional output port to support byte and halfword LD/ST

Name	Direction	Width	Details
byte_en	Output	3:0	CPU sends which bytes to store. Each bit enables store for 1 corresponding byte in a 4 byte word

1. Add a "byte_en" port that is output of CPU (please add it to your cpu ports) and is connected to the ram via the tb.sv.

2. The byte_en must be driven in your cpu for byte and half word stores so if byte_en = 4'b1010, the ram will only store bytes 3 and 1 of the 4 byte word. Similarly, if byte_en = 4'b0110, the ram will store only bytes 2 and 1. To store all bytes, byte_en = 4'b1111.

3. At your end, you need to add the output port to your cpu module, add the logic to drive byte_en for stores that are byte or half word sized, and use the new tb.sv and ram.sv. That should do it.

4. You need to load the instruction memory - rom.sv with “ldst.dat”.

5. For Load and Stores, we need to use a data memory - ram.sv. For this lab, we do Stores before Loads to any and every location, so data memory initialization happens to not matter here, though we will need to do it in Lab 7.

You may load the data memory with some “dummy.dat” or nothing at all for Lab 6 (you do not need to change anything, just add the provided files to your project as usual).

Deliverables -

Demo	- 50 marks
Code submission	- 50 marks