Dear Architects (CPU Architects ☺),

Hope you are doing well; this is your first assignment. Please pay attention and consider to this issues:

• Implement all modules in **data flow** modeling.

• Implement all modules in one Xilinx project.

• For each module you have to create **Verilog test fixture** (test bench), If you didn't create test therefore **you will lose half of the grade**.

• In some questions it is needed first to provide a schematic and label the wire and then implement a circuit in Verilog language. So put your schematic besides your code.

• After finishing your work, it's better to **clean up** your project in Xilinx to reduce the file size.

• Put all of your work in final folder and change the folder name to this format **FnameLname_Assignment2.rar/zip** and after that rar or zip folder and upload it on the site.

• There is a section in your assignment which labeled as **Bounce**, you can do it to use extra marks.

• If you faced with any problems or question, please feel free to ask your question in the site ASAP.

• Please answer the evaluation section seriously. We need and would like to see your comments.

Hint: To find the schematic it's enough to study your logic book (Mano), Also you can find the electronic version in week 0 of site.

Good Luck

1. A) Implement **1-bit half adder**. Test it as (in1=1, in2=1), (in1=1, in1=0), (in1=0, in2=1), (in1=0, in2=0)

B) Implement **2-bit by 2-bit binary multiplier** using 1-bit half adder implemented in part A. Attach the schematic. Test it as (A=0, B=3), (A=2, B=1), (A=2, B=2)

2. A) Implement **3 × 8 decoder**. Test it as (xyz=110), (xyz=111), (xyz=000)

B) Implement a **full Adder** with a decoder using 3 × 8 decoder implemented in part A. Test it as (in1=1, in2=0, C=0), (in1=1, in2=1, C=1), (in1=1, in2=0, C=1)

3. Implement **1-bit Majority**. Majority circuit has 4 inputs and 1 output. The output is 1 when the most of inputs (3 or more) are 1.

4. A) You've studied chapter 2 of Computer Architecture course and you are now expert in **MIPS instructions**. Write MIPS codes for adding values of 10 subsequent blocks of memory and store the sum on the 11th block. Hint: you should load values from memory to registers and, then store the sum from register to the memory.

B) After finishing part A, write machine codes (**binary codes**) of each instruction. This part is very important because you will use these codes in your assignments after midterm. This codes are inputs of the CPU which you will implement later. *CPU just understand binary codes*.