CO II Project II Description 2019-2020

Please read very carefully.
In this project you are required to implement a functional Mips Processor in Verilog. It has two parts : The Basic Part, and The Bonus Part, and The Bahaleel El Se3eed Competition Part.

- Number of students : up to 8 , preferably 5-8.

The Basic Part [10 Marks]: you are required to implement the single cycle mips processor in verilog. You already have all the information you need to implement that from 2nd Electrical CO course and this shouldn't be a problem. This part is worth 10 marks of your yearwork marks.

Your processor must be able to take as input a text file with binary instructions [Which are assembly instructions converted to their binary form, either R format, I format, or J format - There might be online converters for this conversion from assembly to binary or you can use the MARS simulator for mips assembly]. The processor will then read those instructions into its instruction memory and start performing them all just as a true processor will do, updating memory and register file contents. You should support the following instructions in your implementation (Search for all the information you need regarding the instruction formats and opcodes in the reference or online) :
[add - sw - lw - sll - and - or - beq - J - JAL - JR - addi - ori - slt].

The important requirement is that the outputs must be completely correct, regardless of your design choices. That's what you would expect from a real processor. I Can't Give You Marks Unless Your Processor Does Work As Expected, Even If You Show Me One Thousand Lines Of Code. Also You must use standard opcodes, AluOps and so on.

For a review on the single cycle processor and how those assembly instructions work and translate into binary instructions, see the following playlist :
https://www.youtube.com/playlist?list=PLQkyODvJ8ywv5YjAQ2uC550ZP1ZCIP4yn

You must create a good report explaining your design, and having at least five different and complex test-cases (both in assembly and the equivalent C code). Every test case will have the code, the expected output, and proof that your design does produce this expected output. Your test-cases must contain all the required instructions. you will lose marks If you don't have this in your report

This will still be needed in the bonus part and the Bahaleel El Se3eed Part.

The Report should also include the following :

- **A brief description of your implementation and design choices.**
- **The Datapath you used including any necessary extensions you added to support all the required instructions.**
- **Any Assumptions you added that are not standard**
- **A section showing the contribution of each student in the project (Who did what exactly and the percentage of the overall effort every one made)**

The Bonus Part [upto 7 Bonus Marks] :

In addition to that, you need to do implement the following :

1- Your code MUST BE synthesizable. you will show me that on xilinx software, and show me te synthesized circuit. The synthesizer should output NO WARNINGS AT ALL. [2 Marks]

2- You must implement an assembler for your processor. I will write code inside the assembler (in assembly, not in binary). The assembler gui will translate the assembly instructions into binary and Automatically call (modelsim / xilinx or iverilog), run the code, and then receive the final state back into the gui. Final state means : The contents of memory, regFile, and PC after the program execution. [2 Marks]

3- You must implement some automated testing for your design. Here is the idea : Create let's say 10 test cases (10 assembly programs) in 10 separate files. Also create 10 files for the correct output (register and memory contents) after the execution of those programs. Now, Using

any scripting language [You can use python or perl for example] , Loop over all those cases, open the file, use your own assembler to transform the assembly into binary code, save the binary code into a file, and order (modelsim/xilinx/ or iverilog) to execute this code, and then retrieve the state (regFile and memory contents) and compare this state with the "correct" state you have. If the contents match, then this specific test case passed, else this test case failed. Create corner and complex test cases. This whole process must be fully integrated and automatic, you just run one command and all this should happen on its own. [3 Marks]
The great advantage of this method of testing is that whenever you make any change to your design, you can just re-run the script and find out whether your modification affected the correctness of your design or not. You can also have a huge number of test cases, and add to them incrementally.

-

Note that you need to implement all that on your own. Assume any missing details, and do not ask me regarding any implementation details. You should be familiar enough with verilog to completely depend on yourselves.

Also in this part, the report will need to also include :
1- An extensive overview of your testing strategy, and screenshots of your automated testing running correctly as expected.

The Bahaleel El se3eed Competition [up to 25 extra Marks] : Those Bahaleel will participate in a competition and also get bonus marks. Two teams can win. The first prize would be 3000 L.E for the best winning team. The second team will get $10 for each bahlool of the team, regardless of the total number of bahaleel.

To Enter the competition, you must First implement all the requirements in the bonus part. Now this competition part will be highly competitive. Again, all the requirements in the bonus part must be first implemented, especially the automated testing requirement.
If ALL TEAMS Did only the bonus part without this Bahaleel Part, The prize will be given to the best team of them. However, If one team decides to implement

this Bahaleel part, this team will be considered best, and will be given the prize. There are three suggestions for the bahaleel part :

- **The first suggestion [for 10 Marks] will require a lot of self-study. In this part, you will need to implement a functional 5-stage pipelined mips processor. You will need to study a lot on your own, and you will find the necessary information in this playlist (at least the first 17 videos). Note that anyway you will need to study the pipelined mips processor in your manhag, however this will be near the end of the semester. I am giving a lot of bonus marks here as you will both study and implement on your own.**
  https://www.youtube.com/playlist?list=PLQkyODvJ8ywsbQJjDEOxY8oHkC-ea_y5m
  You will implement all the instructions required in the basic part in this pipelined processor. You also must implement Memory to ALU forwarding and ALU to ALU forwarding, and handle any other possible hazards by any method of your choice. Again, The important requirement is that the outputs must be completely correct, regardless of your design choices. That's what you would expect from a real processor.

  Note also that if you implement the pipelined version and it worked correctly, there is no need to show me the single cycle version. However, I do recommend that you implement the single cycle version first as it will be much harder to build the pipelined version directly.

  For this part, your report must also include :

    1- A detailed description of your datapath, your design choices, and any additional hardware you had to add to support all the required instructions. Be as clear and detailed as possible as this will affect your evaluation.

- **The second suggestion [For 5 Marks] : In addition to the previous implementations, You will create an educational software for the pipelined mips processor. In this software, we will have a GUI having all the components of the processor. We can write assembly code in your program, and execute the code cycle by cycle. In each cycle, the datapath will change colors and values, indicating which instruction is**

executing in which stage at this specific cycle, and also the values stored in the regFile , memory, the pipeline registers, and all the wires. You also must support an "animation" feature where we see the instructions propagating in the datapath cycle by cycle. You are free to make this a completely separate software, or make this software communicate with iVerilog/modelsim/xilinx, as long as the whole simulation is done using the software GUI.

- **The third suggestion [For 10 Marks] would be to implement the processor on an actual FPGA and devise a way to show me it does work correctly for different programs I give you. If you do that and if you also get chosen as the first winning team, you will get an extra 1000 L.E for the FPGA implementation.**

You are free to add any more features you want. Everything you add will be considered in the evaluation of the best teams. For example :

If you implement correctly working dynamic branch predictions and all the other competitors implemented stall on branch , you will be considered better than them.
- If your code is better, more organized, better commented, you will be considered better.
- If your test cases are better, more representative, and cover difficult or corner cases, more in number, or more clearly documented, you will be considered better.
- If your document / report is better, clearer, more organized, or more professional, you will be considered better.

In addition for this part , you need to provide a professional user-guide for your simulation software showing step-by-step execution of sample programs on the simulator, and showing all the implemented features in the clearest way possible.
Again, please add ANY additional features in a clear way in your report.

FINALLY, For all of you, you have to write your groups and names in this google sheet :

https://docs.google.com/spreadsheets/d/1nwMxN2xj

8MSiq3TibV5jLLsiOOGz69Vl3U8Pp1c5cNA/edit#gid=0

The Deadline for this project is 15/11/2019. I will receive the project face to face, and will ask every specific group member what he/she contributed to the project.
Good Luck !