# Computer Organization Project (Weight: 30%)

# Project Deadline: April 30, 2023, Sunday, 23:59

# Adding an Instruction to Single Cycle 16 Bit RISC Processor

Max number of students in a group: 6

1. Download the 16-bit RISC processor Verilog implementation source codes from UZEM.
2. Open it in VS Code using the folder containing the project. (Just open the folder, you do not need to create a new project) Investigate each design file and the test bench. There is an .mhtml file describing the processor. Read it and understand how the processor has been designed and implemented.
3. Compile the project using Icarus Verilog and view the waves using GTKWave. Use the following commands in a command prompt window:

iverilog -o RISC\_16\_bit.vvp Testbench.v

vvp RISC\_16\_bit.vvp

gtkWave RISC\_16\_bit.vcd  
  
Using the waveforms examine how instructions are implemented.

1. (30 points) Provided processor design implements the register resets using initial blocks and for loops. Initial blocks cannot be synthesized. Remove all the initial blocks and change the registers in the processor to resettable and synthesizable ones. Add a reset input to the processor and in testbench apply reset signal for 2 clock cycles before the processor runs the instructions. This reset should write all 0’s to register file registers, instruction memory, program counter and any other register that should be reset using this signal.
2. (70 points) Add a new instruction to the instruction set:

An example new instruction:

Set on Greater Than:

**SGT ws, rs1, rs2** ws:=1 if rs1 > rs2; ws:=0 if rs1 <= rs2

**DO NOT add SGT instruction**. Do your own research and decide which new instruction should be added first to make the processor more efficient. In your report explain why you have chosen this instruction based on your research. Your instruction MUST be different from other group instructions. DO NOT COPY from other groups.

Use operation code 0x0014 for the new instruction.

1. Do all necessary changes in the data path and control units. (You may need to change width of the alu\_control signal from 3 bits to 4\_bits, etc.)
2. Add an example of the new instruction to “test.prog”
3. Update the “’Parameter.v” file to reflect the new program memory size.
4. Your new instruction should change the data memory after simulation. Show how data memory reflects the correct output after simulation. (test\_data\_result.txt)
5. Run the simulation and show that your new instruction functions properly using the wave window.
6. Write a report explaining each step clearly using wave windows and code segments taken from your implementation.

Submission Material:

One single ZIP file containing

1. Project Report (Write all the group member names and student numbers on the cover page)

2. Source Code Folder

NOTE: DO NOT GIVE YOUR WORK TO OTHERS. COPY REPORTS WILL GET ZERO POINT. LATE REPORTS WILL NOT BE ACCEPTED. ALL GROUP MEMBERS WILL SUBMIT THE SAME REPORT AND SOURCE CODES. DO YOUR SUBMISSIONS TO UZEM ONLY.