# Laboratory 06: CPU Sim – Computer Architecture Simulation – Part VI

Recall your learnings from previous CPUSim exercises. In this experiment, you will use features of CPUSim to

1) be familiar with instructions that accepts more than one operand
2) design an architecture that hides registers from the user
3) design computer architectures and organizations to fulfil a specific task
4) debug assembly code to identify issues with architecture, organization and code
5) enhance knowledge on designing the logical framework of an IC
6) practice and enhance knowledge on embedded systems

In this lab you are supposed to identify the required Hardware, Micro Instructions to realize a given Instruction Set in order to achieve a specific task. You will also learn how to manage instructions with multiple operands.

To get started:

1) Create a **work folder** in desktop with your student number in the format ***PX_2020_###***.
2) Download the *CPUSim_Lab06_pkg.zip* file from the LMS.
3) Unzip the downloaded file and **move** the *CPUSim_Lab06_pkg* **folder** to the **work folder**.
4) Double click on *CPUSim.bat* file, which is inside the *CPUSim_Lab06_pkg* **folder** to get started.
5) Create a new folder called *Answers* inside the **work folder** (***PX_2020_###*** on the Desktop**)**. All your machine files and programme files should be saved in the *Answers* folder.

Follow the guidelines carefully and complete the exercises.

You should save all your work in the ***Answers*** folder.

You might find following information useful when you do the exercise.

- CPUSim RAM (Main Memory) is byte addressable, i.e., one word in memory is 8 bits.
- In a good design of a computer Architecture, there should be a Micro Instruction to increment the programme counter (The register which keeps track of the next instruction to be fetched) with a desired step.
- When you define a Memory access Micro Instruction in CPUSim, the amount of data read or written to the Memory is equal to the relevant Register width. The registers that load data/instruction to/from Memory should have widths of integer multiple of 8.
- You should define a *halt-bit*, to facilitate proper exit from the programme.
- You may have multiple registers with different widths to handle different types of data.

## Exercise

Your task in this laboratory is to

1) **define a computer architecture in CPUSim** that has ONLY the instructions (instruction set) given in Table 1,
2) **define the computer organization** that allows user find the factorial of a number less than or equal to 12,
3) **write a programme** that would allow user to enter an integer number and output the factorial of the user input value and exits the programme.

Instruction Set

| Instruction name | Description |
|---|---|
| MULM3 | Accepts three operands as three memory locations, multiply the two memory location values relevant to the 2nd and 3rd operands and stores the answer in memory location specified by the 1st operand.<br>*MULM3 X Y Z → M(X) = M(Y) * M(Z)* |
| JMPLEZM2 | Accepts two operands as a memory addresses. If the value in the memory location specified by the 2nd operand is less than or equal to zero, jump to the location specified by the 1st operand.<br>*JUMPLEZM2 X Y* |
| JUMP | Jump to the location specified by the operand. *Unconditional jump.* |
| INCM | Accepts an operand as a memory address and increase the value at memory location specified by the operand by 1.<br>*INCM X → M(X) = M(X) + 1* |
| SUBM3 | Accepts three operands as three memory addresses, subtracts the value at memory location specified by the 2nd operand by the value at memory location specified by the 3rd operand and places the answer in the memory location specified by the 1st operand.<br>*SUBM3 X Y Z → M(X) = M(Y) – M(Z)* |
| READM | Accepts one operand as a memory address. Prompts user to enter an integer value and stores the value entered by the user at memory location specified by the operand.<br>*READM X → M(X) = <user input>* |
| WRITEM | Accepts one operand as a memory address. Displays the integer value at memory location specified by the operand on screen. |
| STOP | Indicates the programme to stop execution. |

Table 1

Save the programme and machine inside the *Answers* folder; file names should be in the format ***PX_2020_###_excercise.a*** and ***PX_2020_###_exercise_machine***, respectively.

Note that, in the instructions given in Table 1, LOAD and STOR instructions are not included. Also, this architecture does not let the user directly access register values in the processor. In other words, registers are hidden from the user.

**After you complete all the Exercises, submit your work using LMS.**
Explain the pros of cons of this design, in Online text section in LMS.