You are going to build a RISC-V CPU as the final project for this course. Your CPU is going to execute the code which was mentioned during the course in RISC-V assembly sessions. Your CPU will support instructions which are used in `RISCV_Code.S`, which means you will have to consider Multiply and Fixed-Point extensions in your design (RV32IMF).

Modules of this CPU are included in `RISCV_CPU.zip` and its ports are also defined, based on the block diagram of this specific CPU microarchitecture. You have to complete these modules according to block diagrams, instructions and input/output signals which are predefined in files.

Your main CPU core is not going to be designed as a top module in your circuits. `Core.v` file which is included in `RISCV_CPU.zip` is actually a testbench which all of your modules and signals are included inside it.

In the first phase of this project, you are going to design Registers, Integer and Fixed-Point Register Files, Fetch circuitry and Program Counter based on different stages of this 4-stage CPU.

1) Design and implement a register module using D-flip flops. You'll need this module in order to create some of your main registers in CPU such as PC (Program Counter) and IR (Instruction Register). Your register module needs to have the following input signals:

`CLK / Enable / Register_Input`

And the following output:

`Register_Output`

2) Design and implement the integer and fixed point register files of your CPU. Your register files must support 32 registers each having 32 bits. Your register file, also needs to have the following input signals:

`Reset / WriteEnable / WriteIndex / WriteData / ReadEnable_1 / ReadIndex_1 / ReadEnable_2 / ReadIndex_2`

And the following outputs:

`ReadData_1 / ReadData_2`

[Hint: use `genvar` and `generate` plus `for` loops for replicated hardware]

*** Important Note:**

Registers indexed zero are considered hard-wired zeroes in RISC-V. Your register files should avoid writing to this index.

3) Instruction and data memory:

An instruction memory is provided in `Instruction_Memory.v`. This module contains the machine code of the path distance calculation program written in RISC-V assembly language (from previous sessions of the course). Include this module in your core file.

Module input: `ImemReadEnable / ImemReadAddress`

Module outputs: `ImemReadData`

4) Program Counter:

- **a.** Include `Register.v` in your module and create PC register.
- **b.** Consider following signals as inputs:
  `CLK / Reset / Stage / Branch_Enable / immediate (32 bits)`
  And the following outputs:
  `PC (32 bits)`
- **c.** Design the PC counter circuit. (Note that PC needs to incremented by 4 each time)
- **d.** Assign your PC register input as `32'b0 – 4` when `Reset` signal is enabled.
- **e.** Create a wire named `branch_target` and assign `PC + immediate` to it.
- **f.** Create a wire named `next_PC` and assign

  `branch_enable ? branch_target : PC + 4` to it. (Adding PC according to branch enable signal)

- **g.** Create a wire named `enable` and assign value to it according to `stage == 3` and `Reset` signal.

5) Fetch Unit:

- **a.** Consider following signals as inputs:
  `Stage / PC (32 bits)`
  And the following outputs:
  `Instruction (32 bits)`
- **b.** Create a wire named `enable` inside your module and assign it to `stage == 3`.

c. Include `Instruction_Memory.v` in your module and create a memory inside your module (`Imem`).

d. Instruction Memory has 2 input signals:
`ImemReadEnable` / `ImemReadAddress (32 bits)`

And one output signal:
`ImemReadData (32 bits)`
Create Instruction memory in your fetch module as the following command:

```
Instruction_Memory Imem (enable, PC / 4, instruction);
```

[Hint: take a look at `RISCV_CPU_Datapath.pdf`. Block diagrams will help you with designing the Program Counter, Fetch Unit and in conclusion, stage 1 of your 4-stage CPU]

Notes:

Send all your assignment related files (*Top module* and *Testbench* [.v] and [.vvp] and [.vcd] files) along with a detailed report in [.pdf] format all in one zip file to the email address of the class.

For this assignment you can work in groups of two, each participant must submit the assignment individually with their respective name and student number.

If you have any questions regarding this assignment, feel free to contact us.

Please submit your final project in the following format:

Name_StudentNumber_Project (BillGates_12345678_Project)

Good Luck!