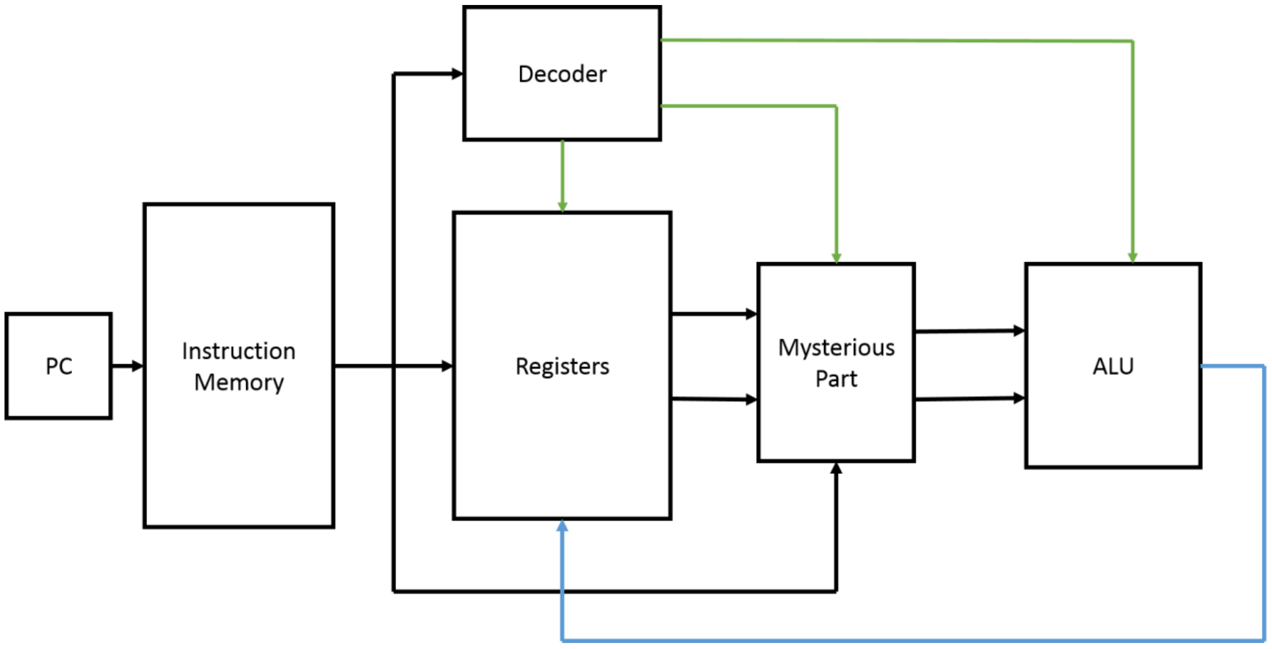
Final Project(20+5(bounus))

In final project, you have to design a simple CPU. It can read machine codes from instruction register, get values from registers, compute result in ALU, and store the result back in registers. The picture below is the schematic:



The picture above is only for clues. The wiring is not complete. You need to design by yourself.

1. PC

Program Counter. This module’s output controls which instruction should output. PC add one by itself per clock. (PC=PC+1)
For example, if PC=0, the IR will output first instruction. If PC=1, the IR will output the second instruction ….
Since there are 25 instructions in this design, PC only need to increment until 24 and then it should stop at 24. **We will give you this module**.

2. Instruction register (IR)

It contains machine codes inside. The value it outputs decided by PC. **We will give you this module**. Every instruction is 20 bits.

3. Registers

There are totally 4 registers in this design. This module has two outputs. The up output is decided by the first register address, the down output is decided by the second register address. (per register is 9 bits one bit is used for overflow flag, which is the top bit)

4. Decoder

The decoder is the control unit. It receives op-code from instruction and decode the op-code to control everything in your design. It is very important part. You need to design by yourself.

5. ALU (Arithmetic Logic Unit)

An arithmetic logic unit is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers. In this project, there are only 10 operations you need to perform, and the length of output should be 8 bits. The most important thing is that you must use gate-level design to implement these operations, which means you cannot use + - in your design, or you will get zero point.

Specification:
In every instruction, you will receive 20 bits. The upper 4 bits is called op-code, the middle 8 bits is register address, and the last 8 bits could be register address or immediate value.
For example, 0010 is add, 0011 is addi. If you get 0010 00000001 00000011 from IR.
0010 -> add   00000001->R1   00000011->R3
So the final result is R1=R1+R3
If you get 0011 00000001 00000111 from IR.
0011->addi   00000001->R1   00000111->7
So the final result is R1=R1+7
NOTE: the middle 8 bits could never be immediate value.

Op-code	assembly	meaning
0000	null	do nothing
0010	add	add the values from two registers
0011	addi	add an immediate value to the register
0100	sub	subtract the back register from the first register
0101	subi	subtract an immediate value from the register
1110	and	and the values from two registers
1100	or	or the values from two registers
1010	xor	xor the values from two registers
1000	not	not the value from the first register(1’s complement)

This project will only test by testbench.
Notice : we will give you top module and part of register module, please use the included modules and don’t modify them, or if the output format is different to the testbench, you will not get points!!!!

Grading Policy

10 points for ALU. (if you don’t use gate-level design, you will get 0 point at this part)
5 points for Decoder.
5 points for Registers.
5 points for overflow and zero flag