CS F342 COMPUTER ARCHITECTURE

ASSIGNMENT 1

Implement a 5-stage pipelined processor in Verilog. This processor supports basic instructions load (LW), store (SW), jump (J), register left shift (LSR), and register right shift (RSR). The processor should implement forwarding to resolve data hazards. The processor has Reset, CLK as inputs and no outputs. The processor has instruction fetch unit, decode, reg read (with 32 32-bit registers), execution, memory, and writeback units. The processor also contains four pipelined registers IF/ID, ID/EX, EX/MEM, and MEM/WB. When reset is activated, the PC, IF/ID, ID/EX, EX/MEM, and MEM/WB registers are initialized to 0. The instruction memory and register file get loaded by predefined values.

The pipelined registers contain unknown values when the instruction unit starts fetching the first instruction. When the second instruction is being fetched in the IF unit, the IF/ID register will hold the instruction code for the first instruction. When the third instruction is being fetched by the IF unit, the IF/ID register contains the instruction code of the second instruction, the ID/RR register contains information related to the first instruction, and so on. (Assume a 32-bit PC. Also, Assume Address and Data size as 32-bit).

The instruction and its 32-bit instruction format are shown below:
LW destinationReg, offset [sourceReg] (Sign extends data specified instruction field (15:0) to 32-bits, add it with register specified by register number in rs field and store the data corresponding to the memory location defined by the calculated address in the rt register. Opcode for LW is 101000).

op	rs	rt	offset
6 bits (31-26)	5-bits (25-21)	5-bits (20-16)	16-bits (15-0)

SW SourceReg, offset [destinationReg] (Signextends data specified in instruction field (15:0) to 32 bits, add it with register specified by register number in rs field. Opcode for SW is 100011).

op	rs	rt	offset
6 bits (31-26)	5-bits (25-21)	5-bits (20-16)	16-bits (15-0)

LSR destinationReg, sourceReg 1, sourceReg 2 (Perform left shift operation on the register specified by register number in RS field by the value stored in the register specified by register number in RT in and save the result in the register specified by register number in RD field. OPCODE for LSR is 110010, SHAMT is 00000, and FUNCT is 000000).
[NOTE: You must solve this part, without using the SHAMT section of the instruction]

op	rs	rt	rd	shamt	funct
6 bits (31-26)	5-bits (25-21)	5-bits (20-16)	5-bits (15-11)	5-bits (10-6)	6-bits (5-0)

RSR destinationReg, sourceReg 1, sourceReg 2 (Perform right shift operation on the register specified by register number in RS field by the value stored in the register specified by register number in RT in and save the result in the register specified by register number in RD field. Opcode for RSR is 111011, SHAMT is 00000, and FUNCT is 000000.
[NOTE: You must solve this part, without using the SHAMT section of the instruction]

op	rs	rt	rd	shamt	funct
6 bits (31-26)	5-bits (25-21)	5-bits (20-16)	5-bits (15-11)	5-bits (10-6)	6-bits (5-0)

J target (Jumps to an address generated by appending 00 to the right and 4 higher order bits of the program counter to the left, of the 26 bit address field in the instruction. Opcode for j is 000010).

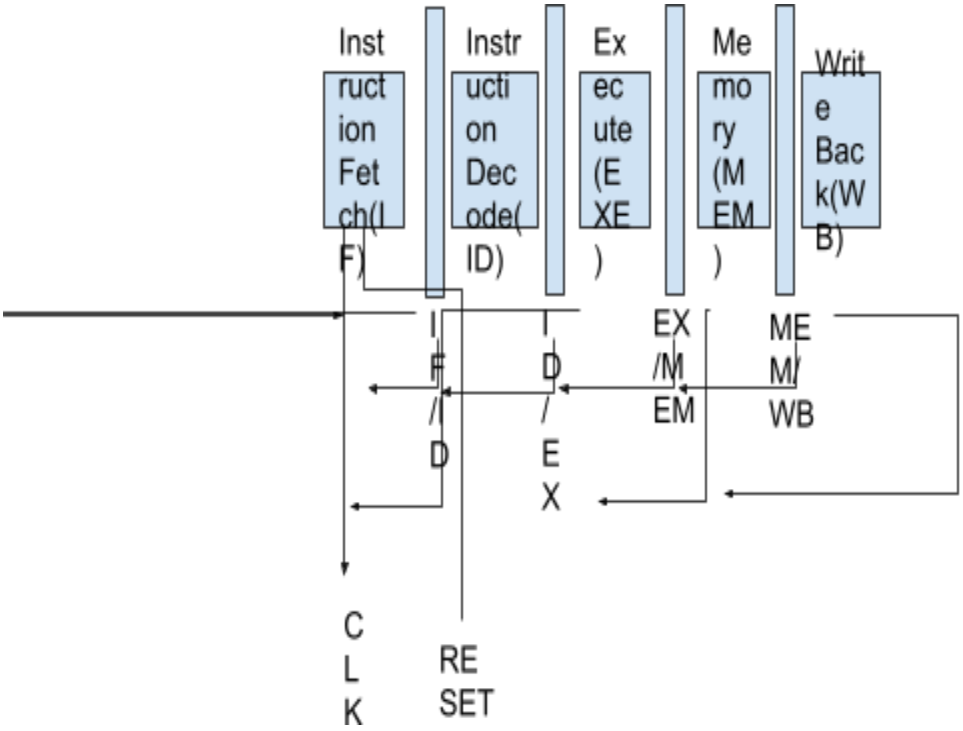
op	address
6 bits (31-26)	26-bits (25-0)

Assume the register file contains 32 registers (R0-R31) each register can hold 32-bit data. On reset, PC and all register file registers should get initialized to 0. Ensure r0 is always zero. Each location in DMEM has 8-bit data. So, to store a 32-bit value, you need 4 locations in the DMEM, stored in big-endian format. Also ensure that on reset, the instruction memory gets initialized with the following instructions, starting at address 0:

```
        LW R1, R11, #12
        RSR R3, R8, R2
        LSR R2, R1, R3
        J L1
        RSR R6, R6, R6
L1:     RSR R4, R3, R5
```

The above code should run correctly on the processor implementation. Ensure that you handle the data hazards present if any.

A partial block-level representation of the 5-stage pipelined processor is shown below. **Please note that for register file implementation, write should be on the positive edge, and read should be on the negative edge of the clock.** Write operation depends on the control signal.



As part of the assignment, three files should be submitted in a zipped folder.

1. PDF version of this Document with all the Questions below answered with the file name **IDNO_NAME.pdf.**
2. Design Verilog Files for all the Sub-modules (instruction fetch, Register file, forwarding unit).
3. Design a Verilog file for the main processor.

   The name of the zipped folder should be in the format IDNO_NAME.zip