Experiment: Design and Synthesize a 32-bit Processor using Verilog

Design a 32-bit RISC-like processor in Verilog with the following specifications:

Specifications

- Sixteen 32-bit general-purpose registers R0…R15, organized as a register bank with two read ports and one write port.
- R0 is a special read-only register that is assumed to contain the fixed value of 0.
- Memory is byte addressable with a 32-bit memory address. All operations are on 32-bit data, and all loads and stores occur from memory addresses that are multiples of 8.
- A 32-bit program counter (PC).
- A 32-bit stack pointer (SP).
- Addressing modes to be supported:
  a) Register addressing
  b) Immediate addressing
  c) Base addressing for accessing memory (with any of the registers used as base register)
  d) PC relative addressing for branch
  e) Indirect addressing

Instruction Set

a) **Arithmetic and logic instructions:** ADD, SUB, AND, OR, XOR, NOR, NOT, SL, SRL, SRA, INC, DEC, SLT, SGT, LUI, HAM. Immediate addressing versions have the suffix "I" (e.g., ADDI, SUBI).

   Example instructions:

```
ADDI R3,#25    // R3 <= R3 + 25
ADDI R5,#-1    // R5 <= R5 - 1
ADD R1,R2,R3   // R1 <= R2 + R3
SLA R5,R7      // R5 <= R5 << R7[0]
SLAI R5,#1     // R5 <= R5 << 1
```

b) **Load and store instructions:** LD, ST (all 32-bit). Register indexed addressing is used.

Example instructions:

```
LD R2,10(R6)   // R2 <= Mem[R6+10]
ST R2,-2(R11)  // Mem[R11-2] <= R2
```

c) **Branch instructions:** BR, BMI, BPL, BZ.

Example instructions:

```
BR #10         // PC <= PC + 10
BMI R5,#-10    // PC <= PC - 10 if (R5 < 0)
BPL R5,#30     // PC <= PC + 30 if (R5 > 0)
BZ R8,#-75     // PC <= PC - 75 if (R8 = 0)
```

d) **Register to register transfer:** MOVE, CMOV.

Example instructions:

```
MOVE R10,R5    // R10 = R5
MOVE R2,R0     // R2 = R0
MOVE R7,SP     // R7 = SP
CMOV R1,R2,R3  // if R1 = (R2 < R3) ? R2 : R3
```

e) **Program control:** HALT, NOP.

- HALT waits for an interrupt on input pin “INT”.
- NOP performs no operation.

Steps of Implementation

1. Customize the ALU in Verilog to cover all functions required to implement the instruction set. Write a test bench for simulation.
   Deadline: 10th September 2025

2. Design the instruction encoding for the ISA. Clearly mention assumptions and provide justifications. Provide the overall schematic diagram of the data path and detailed control unit design (hardwired control unit).
   Deadline: 10th September 2025

3. Implement the register bank and integrate it with the ALU module. Write a top-level module for testing by implementing operations like:
   $R_x = R_y \, op \, R_z$
   Download the design on FPGA and test functionality.
   Deadline: TBD

4. Implement memory as a one-dimensional register array. Complete the processor design by:

   a) Implementing the data path using structural design methodology.

b) Preparing a table depicting RTL micro-operations for instructions like ADD, ADDI, LD, ST, BMI, MOVE, CALL, RET, with corresponding control signals.

c) Implementing the control path using behavioural FSM design.

d) Downloading design on FPGA and testing functionality.

Deadline: TBD

5. Write programs for any three of the following and test them on the ISA (show run on FPGA):

   a) Divide two integers using non-restoring division algorithm.

   b) Multiply two integers using Booth's algorithm.

   c) Sort 10 integers using insertion sort.

   d) Recursive evaluation of factorial.

   Deadline: TBD