Hardware Modeling Course Final Project
QIAU - Summer 2013

Design an 8-bit CPU according to the following description:

1. The CPU includes 8 internal registers named as R0 to R8.
2. CPU operation is based on load-store mechanism. It means that the operands should be transferred from memory to general purpose registers before any processing.
3. CPU supports the following addressing modes:
   - Immediate (`LDI` instruction)
   - Direct (`LD` and `ST` instructions)
   - Register (all arithmetic and `MOV` instructions)
   - Absolute (`JZ` and `JNZ` instructions)
4. It is able to run the below instructions:

```
LDI   Rx, immediate_value   //Rx ← immediate_value
LD    Rx, direct_add        //Rx ← Mem[direct_add]
ST    Rx, direct_add        //Mem[direct_add] ← Rx
MOV   Rx, Ry                //Rx ← Ry
CLR   Rx                    //Rx ← 0
ADD   Rx, Ry                //Rx ← Rx+Ry
SUB   Rx, Ry                //Rx ← Rx-Ry
INC   Rx                    //Rx ← Rx+1
DEC   Rx                    //Rx ← Rx-1
JZ    absolute_add          //if R0=0 then PC ← absolute_add
JNZ   absolute_add          //if R0≠0 then PC ← absolute_add
HLT                         //halt the program execution
```

5. Each instruction is a 16-bit word while the operand length is 8 bits.
6. All data and instructions are stored in a single memory module. The memory is word addressable and the 8-bit data is stored in the lower byte of the word.
7. The CPU code should be synthesizable.

Write the CPU code, and then connect it to a memory module inside a testbench to start execution of the instructions inside memory. Place the following code in the memory and run it on your CPU.

```
CLR   R2
LDI   R1, 5
LDI   R0, 10      //initialize the R0 as the loop counter
ADD   R2, R1      //add R1 to R2 one more time
DEC   R0          //decrement counter
JNZ   3           //if counter≠0, jump to mem[3] (ADD R2, R1)
ST    R2, 17      //store the final result in low byte of mem[17]
HLT
```