COE 405
Design & Modeling of Digital Systems

Course Project
Simple CPU Design
Due on: Saturday Jan. 11

Your are to design and model a simple Central Processing Unit (CPU) using VHDL. The CPU is an *8-bit machine*, has *4 general-purpose registers* (R0-R3), an *8-bit data bus* and an *8-bit address bus*. The CPU has **14 instructions** using a *3-bit opcode* as shown below:

Opcode	Instruction	Operand1	Operand2	Effect on Flags	Size (bytes)
000	**NOP**	-	-	----	1
001	**LOAD**	Reg	Reg, Mem, Immediate	----	1 2 2
010	**STORE**	Reg	Mem	----	2
011	**ADD**	Reg	Reg, Immediate	OCZS	1 2
100	**SUB**	Reg	Reg, Immediate	OCZS	1 2
101	**SHL**	Reg	Immediate	OCZS	1
110	**SRA**	Reg	Immediate	OCZS	1
111	**JMP**	address	-	----	2
111	**JE**	address	-	----	2
111	**JNE**	address	-	----	2
111	**JG**	address	-	----	2
111	**JGE**	address	-	----	2
111	**JL**	address	-	----	2
111	**JLE**	address	-	----	2
111	**JC**	address	-	----	2

The first instruction, **NOP**, is a no operation instruction. The **LOAD** instruction has two operands. It loads the value of the second operand into the first operand. The first operand can only be a register, while the second operand can be a register, memory, or immediate value. The address of the memory operand is 8 bits. The immediate value is an 8-bit value. In the case where the second operand is a register, the instruction size is 1 byte. However, it is 2 bytes in the other two cases. The **STORE** instruction stores the value of a register into memory. Its size is 2 bytes. The **ADD** and **SUB** instructions have two operands. The ADD (SUB) instruction adds (subtracts) the second operand to (from) the first operand, and stores the result in the first operand. The first operand (destination) can only be a register while the second can be either a register or immediate value. The **SHL** (shift left) and **SAR** (shift arithmetic right) shift the first operand by a number of bits specified as an immediate value. The first operand can only be a register, and the immediate value can be between 0 and 7 (3 bits). The size of these instructions is 1 byte. The jump instructions are either unconditional or conditional. They share the same opcode and they get distinguished by using 3 more bits to indicate the jump type. The **JE** instruction jumps if the Zero flag is 1 while the **JNE** instruction jumps if the Zero flag is 0. The **JC** instruction jumps if the Carry flag is 1. The jump address is stored as part of the instruction as an absolute address. The CPU has four flags namely, the **Sign flag (S),** the **Zero flag (Z),** the **Carry flag (C)**, and the **Overflow flag (O).** The effect of the 14 instructions on these flags is indicated in the table above. ADD, SUB, SHL, and SHR instructions affect the flags as in the 8086 Assembly language. All other instructions have no effect on the flags.

(i) Design the instruction format for the 14 instructions for this CPU.

(ii) Write a behavioral model for the CPU and write a test bench to test your model. The test bench should contain Memory, which contains an assembly program for the CPU to execute.

(iii) Partition your CPU into a data path unit and control unit. Write a data flow model for the designed CPU. The data path should be described structurally while the components in the data path i.e. ALU, registers can be described in a behavioral or dataflow style. The control unit is to be modeled as a finite state machine, which can be modeled in a behavioral or dataflow style.

(iv) In the modeling of the data path use a **single-bus design**.

(v) Use **std_logic and std_logic_vector** for all signals in the design.

(vi) Include all functions, procedures, and user-defined types and constants in a package to be used by the CPU.

(vii) For each entity that you model, test it and include simulation output indicating that it is working properly. The data-path should be tested before it is connected to the control unit. This way you can detect problems in the design and modeling early.

(viii) Assume that the CPU-Memory interface contains the following control signals: Read, Write, and MFC (Memory Function Complete). Both the CPU and Memory are synchronized by the same clock. When a Read or Write request is initiated by the CPU, it remains 1 until the memory indicates that it finished the requested operation by setting the MFC signal to 1. Assume that the MFC signal will remain 1 for 1 clock cycle. Also, assume that the MFC signal changes based on the falling edge of the clock while the CPU signals change on the rising edge of the clock.

*(ix) If you manage to synthesize your CPU, map it to FPGA and demonstrate its proper operation, you will get a **5% bonus** from your total mark in the course.*

Clearly state your assumptions and have your design well documented. Write a professional report indicating all design stages, modeling and testing of each component and the final design. Include both a hard copy and a soft copy of your report and all VHDL files. The grading policy for the project is shown below:

Grading Criteria	**Mark**
Instruction Formats Design	5
CPU Behavioral Description & Test Bench	25
Basic Components Modeling & Test	10
Data-path Design, Modeling & Test	15
Control Unit Design, Modeling & Test	20
Whole CPU Design, Modeling & Test	20
Report Organization	5
Total	100