Implement with FPGA technology using Synopsis VHDL a processor. Implement the necessary logic and data path to execute the instruction set provided. The processor should also have the following hardware resources:

- 16 internal (work) registers of 16 bits each named D0 – D15.
- A loop hardware machine which has a count register LC of 4 bits capacity. The loop hardware machine should execute in parallel with the instructions contained within the loop block.

Note: An instruction word for this processor is 16 bits wide with the most significant bit B15 and the least significant bit B0.

**Instruction Set Description:**

*NOTE: The values for PC, "bbbb", "nnnn" and "mmmm" should wrap around. Also neglect all arithmetic overflows.*

1. **LD Dn,Mem**
   This is a load instruction which loads one memory word from memory address Mem into register Dn.

   Instruction word description:

   
   \[
   \begin{array}{cccc}
   B15 & x & x & x & n & n & n & mmmm \\
   \end{array}
   \]

   
   - B15: memory address: 0000 to 1111
   - B0: register number: 0000 to 1111
   - x: don’t care value
2. **ST** Dn,Mem  
   This is a store instruction which stores the content of register Dn into the memory location Mem

   Instruction word description:

   ![](image)

   memory address : 0000 to 1111  
   register number : 0000 to 1111  
   don’t care value

3. **LDI** B,Dn  
   This is an immediate load which loads the value of b into the register Dn. This instruction is a 2 consecutive words instruction.

   Instruction word description:

   ![](image)

   register number : 0000 to 1111  
   the value of B represented as 2’s complement

4. **MOVE** r Da,Dn  
   This is a register to register move instruction which moves the content of register Da into register Dn. The content of register Da must not be altered by the transfer.

   Instruction word description:

   ![](image)

   register number : 0000 to 1111  
   don’t care value
5. ADD Da,Db,Dn
This is an add instruction in 2’s complement arithmetic which adds the
content of the following registers:

\[ [Dn] \leftarrow [Da] + [Db] \]

Note: The content of the registers Da and Db must not be altered
by this operation. The previous content of the register Dn is destroyed.

Instruction word description:

<table>
<thead>
<tr>
<th>B15</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0</td>
<td>a a a a b b b b n n n n</td>
</tr>
</tbody>
</table>

- Destination register: 0000 to 1111
- Operand 2 register: 0000 to 1111
- Operand 1 register: 0000 to 1111

6. IMAC Da,Db,Dn
This is an integer multiply accumulate instruction performed in
2’s complement arithmetic. The following operation is performed
on the content of the registers:

\[ [Dn] \leftarrow [Dn] + [Da.L] \times [Db.L] \]

Note: Da.L and Db.L represent the least significant 8 bits of the register.
Again the content of Da and Db should not be altered by this operation.

Instruction word description:

<table>
<thead>
<tr>
<th>B15</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1</td>
<td>a a a a b b b b n n n n</td>
</tr>
</tbody>
</table>

- Destination register: 0000 to 1111
- Operand 2 register: 0000 to 1111
- Operand 1 register: 0000 to 1111
7. LOOP  n, c, LC

This is a hardware loop instruction which must execute in parallel with the instruction block following the LOOP instruction word.

The instruction block is formed of the n instructions following LOOP.

The operations performed by this hardware loop are:
1. \( [LC] \leftarrow c \); the loop counter register is loaded with the count value c.
2. the loop loop body which is n instructions is repeatedly executed c times.

**EXAMPLE:**

<table>
<thead>
<tr>
<th>PC0</th>
<th>LOOP 3,2,LC;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC0+1</td>
<td>instr.1;</td>
</tr>
<tr>
<td>PC0+2</td>
<td>instr.2;</td>
</tr>
<tr>
<td>PC0+3</td>
<td>instr.3;</td>
</tr>
<tr>
<td>PC0+4</td>
<td>instr.4;</td>
</tr>
</tbody>
</table>

In this example the instruction block is formed of: instr.1, instr.2 and instr.3.

When the CPU encounters this loop instruction the register LC is loaded with the value "0010".

The instruction block is repeatedly executed 2 times in the following order:
instr.1, instr.2, instr.3, instr.1, instr.2, instr.3, instr.4, ...

**Note 1:** PC0 represents the value of the Program Counter PC at the LOOP instr.

The instructions in the instruction block can not be LOOP or HLT.

The values of c and n must be: \( c >= 2 \) and \( n >= 1 \).

**Note 2:** The CPU should only fetch and decode the LOOP instruction and pass the execution to the hardware loop logic. After LOOP decode the CPU should continue with the next instruction in line. The hardware loop logic should perform the above operations such that by the time the CPU’s program counter PC reaches to the value PC0 + n + 1 it is ready to tell if the loop continues with the next instruction or jumps back to the instruction after the LOOP.
Instruction word description:

```
0110 | nnnn | cccc | xxxx
```

- **opcode**: don’t care value
- **loop count value**: 0010 to 1111
- **number of instructions contained by the loop block**: 0001 to 1110

---

8. **HLT**

This is a halt instruction.

When this instruction is encountered the processor goes into an inactive state where PC is not advanced. All signals are set to inactive values. The content of the code memory, data memory and registers must be preserved. The data memory must be accessible from the PC interface.

Instruction word description:

```
0111 | xxxx | xxxx | xxxx
```

- **opcode**: don’t care value
- **don’t care value**: 0111
- **don’t care value**: xxxx
- **don’t care value**: xxxx
ALL OPERATIONS MUST BE PERFORMED IN 2’s COMPLEMENT ARITHMETIC.
The instruction code memory and data memory are already implemented in CHIP: R1.
DO NOT CHANGE CHIP R1 !!!
DO NOT IMPLEMENT ANY OF YOUR HARDWARE IN CHIP R1!!!
IMPLEMENT ALL YOUR HARDWARE IN THE OTHER 3 CHIPS!!!

ALL THE INFORMATION (simulation, synthesis) necessary to implement your hardware is found in the following directories:
~ta427/TA_labEXAMPLEEs
~ta427/labWINTER01

THE EXAMPLES PRESENTED IN THE APPENDIX OF THIS MANUAL ARE FOUND ON THE SUNEE MACHINES UNDER THE FOLLOWING DIRECTORY:
~ta427/TA_labEXAMPLEEs

In "~ta427/TA_labEXAMPLEEs/lab2example" directory there is a sample project with a different instruction set. The data path and the hardware partitioning can be used as an example. The VHDL example implementation does not have many comments. This example should be used only as a guide. Read all the README files and prepare a REPORT file for your submission similar with the REPORT file from this lab example.

UNDER "~ta427/TA_labEXAMPLEEs/lab1example" there is a sample 16 bit project based on a VHDL implementation of AM2901 ALU chip. This example provides specific information about implementing an ALU, a data path, a CPU and some registry using VHDL. There is complete information about this sample lab in the Appendix.

The purpose of this lab is to obtain experience with FPGA implementation and VHDL synthesis. Also you will gain experience with FPGA partitioning, mapping and routing.
The development of the project should pass through 2 major stages:
1. SIMULATION stage;
2. SYNTHESIS stage.
UNDER ~ta427/TA_labEXAMPLEEs/lab2example directory refer to the SIMULATION and SYNTHESIS directories for specific details.
The design should be implemented based on the COST/SIMPLEx principle introduced in the course. The COST/SIMPLEx measure for this lab is the RATIO factor:

$$\text{RATIO} = \frac{\text{MIPS}}{\# \text{CLBs}}$$

where:

- \#CLBs – is the total number of Complex Logic Blocks used by the design
- MIPS – is the average Million Instructions Per Second executed by the design

**MIPS Calculation**

$$\text{MIPS} = \frac{10^9}{(\text{average CPI}) \times (\text{Clock time in ns}) \times 10^6}$$

where:

- average CPI – represents the average number of cycles per instruction
- Clock time – is equal to the "Max pin delay" reported for the design by the synthesizer.

### Calculate first an average CPI_0 based on the following instruction frequencies:

- LD .................. 10%
- ST .................. 20%
- MOVEr ............... 5%
- STI .................. 20%
- ADD .................. 10%
- IMAC .................. 30%
- LOOP .................. 5% for LOOP instruction in the CPI_0 calculation

use only the number of cycles that are performed by the CPU.

### Calculate the average CPI using the following formula:

$$\text{average CPI} = \left( \frac{100 \times \text{CPI}_0 - 30 \times \text{number of cycles per hardware loop}}{100} \right)$$

The RATIO is used in the marking scheme. Based on the RATIO values the labs are divided into 4 groups G1, G2, G3 and G4. Group G4 contains the labs with the lowest RATIO values and group G1 the labs with the highest RATIOs.

### MARKING SCHEME:

- 14 marks .... full simulation (otherwise 2 marks/each simulated instruction )
- 8 marks .... full hardware implementation (otherwise 1 mark/impl.instruction)

**Group RANKING:**

- G1 ... 8 marks ;
- G2 ... 4 marks ;
- G3 .... 6 marks;
- G1 .... 2 marks;