Description:

Consider the multi-cycle MIPS processor described in Figure 5.33 (page 383) of the text *Computer Organization and Design: Hardware/Software Interface*. Let us make a few changes to the processor described in the book. Let us design a 16-bit MIPS processor, which we call as the MIPS-L processor.

Let us have only 8 registers in the register file for the MIPS-L processor, numbered 0 to 7. Table 1 summarizes the register conventions for the MIPS-L processor. The MIPS-L instruction format is shown in Table 2.

### Table 1. MIPS-L register convention

<table>
<thead>
<tr>
<th>Name</th>
<th>Register Number</th>
<th>Usage</th>
<th>Preserved on call?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$zero</td>
<td>0</td>
<td>the constant value 0</td>
<td>n. a.</td>
</tr>
<tr>
<td>$v0-$v1</td>
<td>1-2</td>
<td>values for results and expression evaluation</td>
<td>No</td>
</tr>
<tr>
<td>$t0-t2</td>
<td>3-5</td>
<td>temporaries</td>
<td>No</td>
</tr>
<tr>
<td>$sp</td>
<td>6</td>
<td>stack pointer</td>
<td>Yes</td>
</tr>
<tr>
<td>$ra</td>
<td>7</td>
<td>return address</td>
<td>No</td>
</tr>
</tbody>
</table>

### Table 2. MIPS-L instruction formats

<table>
<thead>
<tr>
<th>Name</th>
<th>Fields</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field size</td>
<td>3 bits</td>
<td>3 bits</td>
</tr>
<tr>
<td>R-format</td>
<td>op</td>
<td>rs</td>
</tr>
<tr>
<td>I-format</td>
<td>op</td>
<td>rs</td>
</tr>
<tr>
<td>J-format</td>
<td>op</td>
<td>target address</td>
</tr>
</tbody>
</table>
Design such a multi-cycle MIPS processor that can implement the following instructions:

- R-type instructions add, sub, and, or, and slt
- lw and sw
- beq
- addi
- j
- jr
- jal

The MIPS-L machine language for the above instructions is shown in Figure 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Format</th>
<th>Example</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>R</td>
<td>0 2 3 1 0</td>
<td>add $1,$2,$3</td>
</tr>
<tr>
<td>sub</td>
<td>R</td>
<td>0 2 3 1 1</td>
<td>sub $1,$2,$3</td>
</tr>
<tr>
<td>and</td>
<td>R</td>
<td>0 2 3 1 2</td>
<td>and $1,$2,$3</td>
</tr>
<tr>
<td>or</td>
<td>R</td>
<td>0 2 3 1 3</td>
<td>or $1,$2,$3</td>
</tr>
<tr>
<td>slt</td>
<td>R</td>
<td>0 2 3 1 4</td>
<td>slt $1,$2,$3</td>
</tr>
<tr>
<td>jr</td>
<td>R</td>
<td>0 7 0 0 8</td>
<td>jr $7</td>
</tr>
<tr>
<td>lw</td>
<td>I</td>
<td>4 2 1 100</td>
<td>lw $1,100($2)</td>
</tr>
<tr>
<td>sw</td>
<td>I</td>
<td>5 2 1 100</td>
<td>sw $1,100($2)</td>
</tr>
<tr>
<td>beq</td>
<td>I</td>
<td>6 1 2 100</td>
<td>beq $1,$2,100</td>
</tr>
<tr>
<td>addi</td>
<td>I</td>
<td>7 2 1 100</td>
<td>addi $1,$2,100</td>
</tr>
<tr>
<td>j</td>
<td>J</td>
<td>2 5000</td>
<td>j 10000</td>
</tr>
<tr>
<td>jal</td>
<td>J</td>
<td>3 5000</td>
<td>jal 10000</td>
</tr>
</tbody>
</table>

**Figure 1. MIPS-L machine language**

Implement the MIPS-L design using Verilog. Note that we have already discussed many of the MIPS components in the class lectures, and Homework 9 is to design the single-cycle MIPS processor. However, the controller will be different from the single-cycle MIPS processor. jr and jal are also not covered in Homework 9.

Synthesize your design using the tools in the Xilinx WebPack. Note that Xilinx Spartan XC200 FPGA is big enough to download a synthesized multi-cycle MIPS-L processor. The assignment consists of three parts as follows.

**Assignment:** (50 pts)

**Demonstration 1 [30 pts].** Build a verilog implementation of the MIPS-L that can run the R-type (add, sub, and, or and slt), lw, sw, beq, addi, and j instructions. Your MIPS-L module will have a form similar to the following module definition:

```verilog
module MultiMips(pcout, aluresult, reset, clock, instr); // Note that this definition may be incomplete
output [15:0] pcout; // The program counter value
```
output [15:0] aluresult; // Output of the ALU. Used for testing.
input clock;        // Clock signal
input reset;  // Reset signal. PC = 0 when reset is invoked.
input [15:0] instr; // Input from memory.

Figure 2. Example of a MIPS-L module definition

Your MIPS should include all components except the memory. Notice that the MIPS has a reset input. If reset = 0 then the MIPS will run as normal. If reset = 1, then the MIPS has its PC value set to 0. Your test bench should instantiate the MultiMIPS module and the memory. It will also generate the clock signal and the reset. It will display the outputs "pcout", "aluresult", and display the contents of certain memory cells.

Create a project labeled "FProjDemo1.PRJ". Verify that your MIPS-L module works by simulating it. Save the project because you will turn it in later. Demonstrate to the TA that your design works by executing the following instructions on your MIPS-L processor and printing out the values of the registers.

lw $3,0($0)     # i = 0;
Loop:  slti $1,$3,100    # $1 = 1 if i < 100
       beq $1,$0,Skip   # if $1 == 0 (i.e., i >= 100) then
       # skip for-loop
       add $4,$4,$3     # j = j + i;
       addi $3,$3,1      # i++;
       beq $0,$0,Loop   # go back to beginning of for-loop
Skip:

Figure 3. Sample Program-1

Demonstration 2 [Add 10 pts for a total of 40 pts].

Add modules to the MIPS that you built in Demonstration 1 so that you can also run the instructions jr and jal. Create a project labeled "FProjDemo2.PRJ". Save the project because you will turn it in later. Demonstrate to the TA that your MIPS module works by executing the following statements on your design and printing out the results.

main()
{
    k = non_neg(x);
}

non_neg(int i)
{
    if (i<0)
        return 0;
    else
        return i;
}

ret1:  move $3,$4     # $4=x
       # $3=i
       jal non_neg
       move $5,$2     # $5=k
       non_neg:
       slt $1,$3,$0   # $1=1 if
       # i<0
       beq $1,$0,Else
       move $2,$0     # return 0
       jr $7
Else:   move $2,$3     # return i
       jr $7

Figure 4. Sample Program-2
Demonstration 3 [Add 10 pts for a total of 50 pts].

Synthesize your MIPS design from Demonstration 2. Configure an FPGA with your MIPS design. Demonstrate this to the TA in one of the labs. Determine the frequency at which your MIPS design would work. (Hint: Use the timing reports).

Due Dates: December 2, 2002 (Tuesday). Turn in your lab report along with a printout of your code. Also turn in a copy of your programs by email to the TA. Demonstrations are due in class. Note that you can provide demos during the lab sessions even before the due date.

Lab Report: Each group must turn in a single lab report. Failure to turn in a report is an automatic deduction of 50 pts. The objective of your report is to explain what you did to complete this project. You should explain the problems you faced as well as your successes. Organize your lab report in the same format as the sample report given to you in the beginning of the semester.