**Lab 11: Single-Cycle Processor**

***Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)***

**Objective**

In this lab, you will extend the single-cycle RISC-V processor from the textbook and lecture to support two additional instructions: lui and xor.

**1. Multicycle RISC-V Processor**

Figure 1 shows the complete single-cycle processor from the textbook. Figure 2 shows the Control Unit, and Figure 3 shows the ALU. Tables 1 and 2 are the Main Decoder and ALU Decoder truth tables. Table 3 lists the ImmSrc encoding. Figure 4 shows the test program for the RISC-V single-cycle processor in the textbook.

**What to Turn In**

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.
2. A marked up version of Figure 1 (single-cycle processor) showing the needed modifications (lui and xor)
3. A marked up version of Figures 2 and 3 (if modified) showing the needed modifications for the additional instructions (lui and xor)
4. Amended Main Decoder, ALU Decoder, and ImmSrc truth tables to support lui and xor
5. Amended SystemVerilog code to add support for lui and xor
6. Modified test program that uses lui and xor
7. Simulation waveforms (in the order listed above: clk, reset, PC, Instr, SrcA, SrcB, ALUResult, DataAdr, WriteData, and MemWrite – all displayed in hexadecimal for ease of reading). Does your system pass your testbench? Circle or highlight the waves showing that the correct value is written to the correct address, and make sure it is legible.

Please indicate any bugs you found in this lab manual, or any suggestions you would have to improve the lab.



**Figure 1: RISC-V single-cycle processor**



**Figure 2: RISC-V single-cycle processor control unit**



**Figure 3: ALU**

Table 1. Main Decoder Truth Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Opcode** | **RegWrite** | **ImmSrc** | **ALUSrcA** | **ALUSrcB** | **MemWrite** | **ResultSrc** | **Branch** | **ALUOp** | **Jump** |
| lw | 0000011 | 1 | 00 | 0 | 1 | 0 | 01 | 0 | 00 | 0 |
| sw | 0100011 | 0 | 01 | 0 | 1 | 1 | xx | 0 | 00 | 0 |
| R-type | 0110011 | 1 | xx | 0 | 0 | 0 | 00 | 0 | 10 | 0 |
| beq | 1100011 | 0 | 10 | 0 | 0 | 0 | xx | 1 | 01 | 0 |
| I-type ALU | 0010011 | 1 | 00 | 0 | 0 | 0 | 00 | 0 | 10 | 0 |
| jal | 1101111 | 1 | 11 | x | x | 0 | 10 | 0 | xx | 1 |

**Table 2. ALU Decoder Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ALUOp*1:0** | ***funct3*2:0** | **{*op*5, *funct7*5}** | ***ALUControl*2:0** | **Operation** |
| 00 | x | x | 000 | Add |
| 01 | x | x | 001 | Subtract |
| 10 | 000 | 00, 01, 10 | 000 | Add |
| 000 | 11 | 001 | Subtract |
| 010 | x | 101 | SLT |
| 110 | x | 011 | OR |
| 111 | x | 010 | AND |

**Table 3. *ImmSrc* encoding**

|  |  |  |  |
| --- | --- | --- | --- |
| ***ImmSrc*** | ***ImmExt*** | **Type** | **Description** |
| 00 | {{20{*Instr*[31]}}, *Instr*[31:20]} | I | 12-bit signed immediate |
| 01 | {{20{*Instr*[31]}}, *Instr*[31:25], *Instr*[11:7]} | S | 12-bit signed immediate |
| 10 | {{20{*Instr*[31]}}, *Instr*[7], *Instr*[30:25], *Instr*[11:8], 1’b0} | B | 13-bit signed immediate |
| 11 | {{12{*Instr*[31]}}, *Instr*[19:12], *Instr*[20], *Instr*[30:21], 1’b0} | J | 21-bit signed immediate |

# RISC-V Assembly Description Address Machine

main: addi x2, x0, 5 # x2 = 5 0 00500113

addi x3, x0, 12 # x3 = 12 4 00C00193

addi x7, x3, -9 # x7 = (12 - 9) = 3 8 FF718393

or x4, x7, x2 # x4 = (3 OR 5) = 7 C 0023E233

and x5, x3, x4 # x5 = (12 AND 7) = 4 10 0041F2B3

add x5, x5, x4 # x5 = (4 + 7) = 11 14 004282B3

beq x5, x7, end # shouldn't be taken 18 02728863

slt x4, x3, x4 # x4 = (12 < 7) = 0 1C 0041A233

beq x4, x0, around # should be taken 20 00020463

addi x5, x0, 0 # shouldn't happen 24 00000293

around: slt x4, x7, x2 # x4 = (3 < 5) = 1 28 0023A233

add x7, x4, x5 # x7 = (1 + 11) = 12 2C 005203B3

sub x7, x7, x2 # x7 = (12 - 5) = 7 30 402383B3

sw x7, 84(x3) # [96] = 7 34 0471AA23

lw x2, 96(x0) # x2 = [96] = 7 38 06002103

add x9, x2, x5 # x9 = (7 + 11) = 18 3C 005104B3

jal x3, end # jump to end, x3 = 0x44 40 008001EF

addi x2, x0, 1 # shouldn't happen 44 00100113

end: add x2, x2, x9 # x2 = (7 + 18) = 25 48 00910133

sw x2, 0x20(x3) # mem[100] = 25 4C 0221A023

done: beq x2, x2, done # infinite loop 50 00210063

**Figure 4. RISC-V test program**