Alex Oswald

EECE 2323

Due: 6 Nov 2020

Pre-Lab Assignment 6

**§1 Instruction Decoder**

**§1.1 Control Signal**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Opcode | RegDst | RegWrite | ALUSrc1 | ALUSrc2 | ALUOp[2:0] | MemWrite | MemtoReg |
| lw | 0000 | 0 | 1 | 0 | 1 | 000 | 0 | 1 |
| sw | 0001 | 0 | 0 | 0 | 0 | 000 | 1 | 0 |
| add | 0010 | 1 | 1 | 0 | 0 | 000 | 0 | 0 |
| addi | 0011 | 0 | 1 | 0 | 1 | 000 | 0 | 0 |
| inv | 0100 | 1 | 1 | 0 | 0 | 001 | 0 | 0 |
| and | 0101 | 1 | 1 | 0 | 0 | 010 | 0 | 0 |
| andi | 0110 | 0 | 1 | 0 | 1 | 010 | 0 | 0 |
| or | 0111 | 1 | 1 | 0 | 0 | 011 | 0 | 0 |
| ori | 1000 | 0 | 1 | 0 | 1 | 011 | 0 | 0 |
| sra | 1001 | 0 | 1 | 0 | 0 | 100 | 0 | 0 |
| sll | 1010 | 0 | 1 | 0 | 0 | 101 | 0 | 0 |
| beq | 1011 | 0 | 0 | 0 | 0 | 110 | 0 | 0 |
| bne | 1100 | 0 | 0 | 0 | 0 | 111 | 0 | 0 |
| clr | 1101 | 1 | 1 | 1 | 0 | 010 | 0 | 0 |
| -- | 1110 | -- | -- | -- | -- | -- | -- | -- |
| -- | 1111 | -- | -- | -- | -- | -- | -- | -- |

**§1.2 Instruction Decoder**

Please see attached *decoder.v* document for decoder module.

The testbench was assembled in *decoder\_tb.v* with

The following output was derived from said supplied testbench.

A screen shot of a computer

Description automatically generated

**§2 Translate the steps to Instructions!**

• instruction 1:

1- Change the regfile read address2 to 1 <added >

2- Set ALUSrc2 to 0

3- Set the ALUOp to 1 (for ’inv’ operation)

4- Set MemToReg to 0

5- Change the regfile write address2 to 1 <added >

6- Set RegWrite to 1

7- Reset RegWrite back to 0

inv $1 $1

• instruction 2:

1- Change the regfile read address2 to 1 <added >

2- Change the regfile write address to 1

3- Change the instr i to 8’h03 <added >

4- Change the ALUOp to 0x5 (for ’sll’ operation)

5- Set RegWrite to 1

6- reset RegWrite back to 0

sll $1, $1, 0x03

• instruction 3:

1- Set the ALUSrc2 to 1

2- Set the value of instr i to 0xFF

3- Set the ALUOp to 0 (for ’add’ operation)

4- Set the regfile read address1 to 3

5- Change the regfile read address2 to 1

6- Set MemWrite to 1

7- Reset MemWrite back to 0

sw $1, 0xFF($3)

• instruction 4:

1- Set the value of instr i to 0xFF

2- Set the ALUSrc2 to 1

3- Change the regfile read address1 to 3

4- Set the ALUOp to 0 for ’add’

5- Set MemToReg to 1

6- Set regfile write address to 2

7- Set RegWrite to 1

8- Set RegWrite to 0

addi $2, $3, 0xFF

• instruction 5:

1- Change the regfile read address1 to 2

2- Change the ALUOp to 3’b011

3- Change the instr i to 8’hF0

4- Change the ALUSrc2 to 1

5- Change the regfile write address to 2

6- Change the RegWrite to 1

7- Change the RegWrite to 0

ori $2, $2, 0xF0

**§3 Generate the Machine Code**

Using the field coding from the Instruction Set document, the set of assembly instructions from §2 is converted to 16-bit Machine Code and listed in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Instruction | Machine Code | .. in Hex |
| R-type | inv $1 $1 | 0100\_00\_01\_01\_000000 | 0x4140 |
| I-type | sll $1, $1, 0x03 | 1010\_01\_01\_00000011 | 0xa502 |
| I-type | sw $1, 0xFF($3) | 0001\_11\_01\_11111111 | 0x1dff |
| I-type | addi $2, $3, 0xFF | 0011\_11\_10\_11111111 | 0x3eff |
| I-type | ori $2, $2, 0xF0 | 1000\_10\_10\_11110000 | 0x8af0 |