**HOMEWORK 4 WRITTEN ASSIGNMENT**

## 1. Overview:

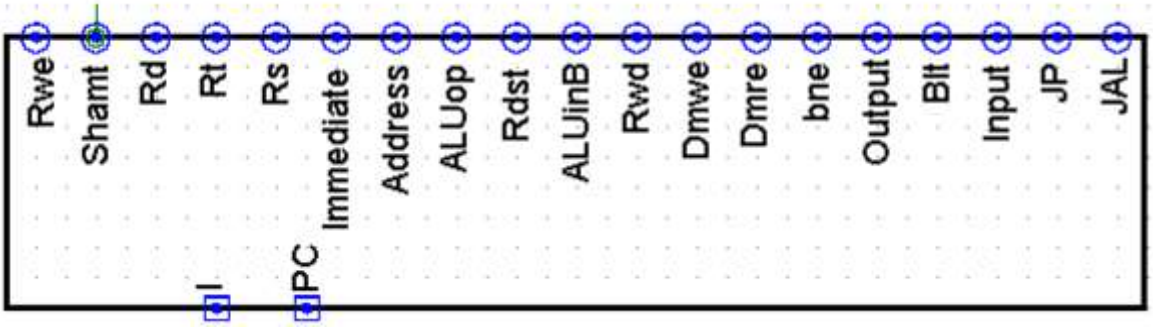
All the Components outlined in this pdf were used in the assembly of the CPU I have submitted on Gradescope. Furthermore, my device was able to pass all the available tests on the autograder. Please refer to the following section to see how each component was made and what function it performs vis-à-vis the rest of the system.

## 2. Subcircuits:

The CPU provided as part of this submission has the following sub-circuits –

a. Instruction_map: Converts the 16-bit instruction into smaller components to execute the R, I and J type instructions and further decodes it to create suitable control signals.

i. Abstraction –



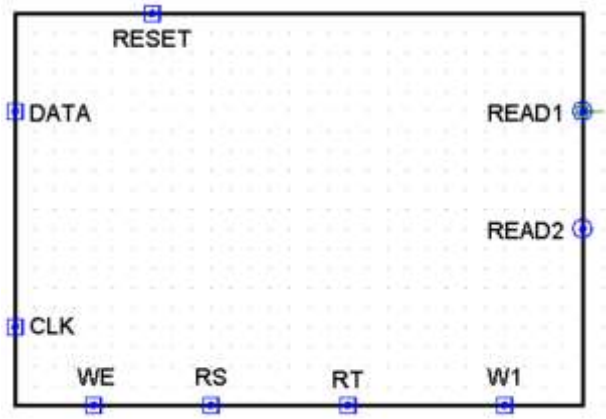

ii. Signal description –

| Signal | Description |
|---|---|
| Rwe | Enables writing to the Register File |
| Shamt | Specifies the shift amount for the ALU |
| Rd | Destination register |
| Rt | Register Val1 |
| Rs | Register Val2 |
| Immediate | Gives immediate value of the I-type instruction |
| Address | Gives the final address value for all J-type instructions, with the PC value appended |
| ALUop | Specifies the operation performed by the ALU<br>000 - ADD<br>001 - SUB<br>010 - XOR<br>011 – NOT A<br>100 – LEFT SHIFT A<br>101 – RIGHT SHIFT A |
| Rdst | Replaces Rd with Rt in the register file if it is 1 |

| | |
|---|---|
| ALUinB | Reads sign extended value of Immediate in the ALU if it is 1 |
| Rwd | Reads from the data memory if it is 1 |
| Dmwe | Enables writing to data memory if it is 1 |
| Dmre | Enables reading from data memory if it is 1 |
| Bne | Executes the Bne instruction if it is 1 |
| Output | Executes the Output instruction if it is 1 |
| Blt | Executes the Blt instruction if it is 1 |
| Input | Executes the Input instruction if it is 1 |
| JP | Executes the various jump instructions as follows –<br>00 – No jump<br>01 – J instruction<br>10 – JR instruction<br>11 – JAL instruction |
| JAL | Executes the JAL instruction if it is 1 |

b. RegisterFile: A 16-bit register file which has 2 read ports and 1 write port.

i. Abstraction –

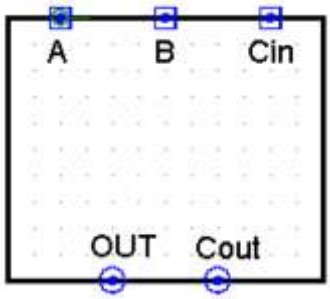



ii. Signal Description –

| Signal | Description |
|---|---|
| RESET | Resets all the registers in the File |
| DATA | Provides 16-bit input to the file |
| CLK | Provides clock |
| WE | Enables writing to the File |
| RS | Describes register to write to |
| RT | Describes register to write to |
| W1 | First Write port |
| READ1 | First Read port |
| READ2 | Second Read port |

c. 1-bit Adder: Adds 1-bit inputs if the Cin is 0. Subtracts 1-bit inputs if the Cin is 1.

i. Abstraction –

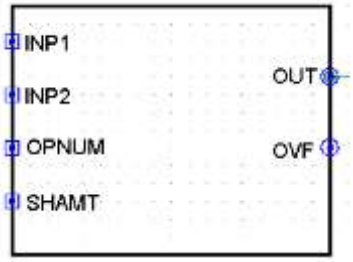



ii. Signal Description –

| Signal | Description |
|---|---|
| A | First Input (Chronologically) |
| B | Second Input (Chronologically) |
| Cin | Control Unit<br>0 – performs A + B<br>1 – performs A – B |
| Cout | Indicates overflow if equal to 1 |
| OUT | Gives the result of the operation |

d. ALU: Performs the following 16-bit operations – ADD, SUB, XOR, NOT, Shift Left and Shift Right.

i. Abstraction –




ii. Signal Description –

| Signal | Description |
|---|---|
| INP1 | The first input of the ALU |
| INP2 | The second input of the ALU |
| OPNUM | Specifies operation performed by the ALU (codes provided above) |
| SHAMT | Specifies the shift amount |
| OUT | Gives output |
| OVF | Signals overflow if equal to 1 |

e. LeftShift: Shifts the given input left (logically) by upto a maximum of seven places.

i. Abstraction –


ii. Signal Description –

| Signal | Description |
|---|---|
| Data | Provides 16-bit input |
| SHAMT | Provides the shift amount |
| OUT | Provides the output |

f. RightShift: Shifts the given input right (logically) by upto a maximum of seven places.

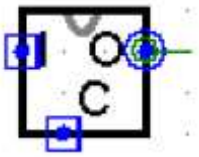
i. Abstraction –


ii. Signal Description –

| Signal | Description |
|---|---|
| Data | Provides 16-bit input |
| SHAMT | Provides the shift amount |
| OUT | Provides the output |

g. ZeroChecker: Checks if the input is 0 if the Control signal is 0. Checks if the input is negative or positive if the control signal is 1.

i. Abstraction –

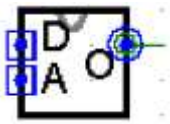



ii. Signal Description –

| Signal | Description |
|---|---|
| I | Provides 16-bit input |
| C | Checks if input is<br>Zero if C = 0<br>Negative/Positive if C = 1 |
| O | Based on value of C, returns<br>1 if input is zero<br>Or 1 I input is negative |

h. ASCII_inp: If the data is available in the keyboard, takes the 7-bit buffer and extends it to a 16-bit output.

i. Abstraction –

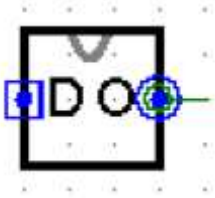



ii. Signal Description –

| Signal | Description |
|---|---|
| D | Provides 7-bit input from the keyboard |
| A | Indicates if data is available or not<br>1 – Data is available<br>0 – Data is unavailable |
| O | Returns the 16-bit output |

i. ASCII_out: Reads the first 6 bits of the input data

i. Abstraction –

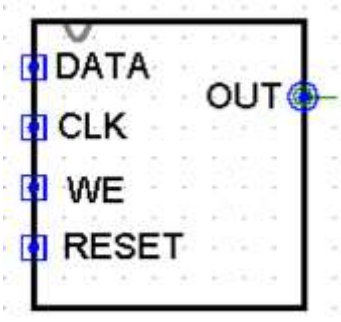



ii. Signal Description –

| Signal | Description |
|---|---|
| D | Provides 16-bit input |
| O | Returns the 7-bit data to be written to TTY |

j. 16-bit Register: Standard 16-bit register implemented using D flip-flops

i. Abstraction –

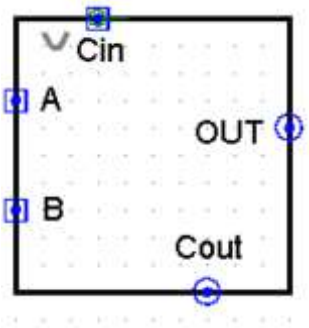



ii. Signal Description –

| Signal | Description |
|---|---|
| RESET | Resets all the registers in the register |

| | |
|---|---|
| DATA | Provides 16-bit input to the register |
| CLK | Provides clock |
| WE | Enables writing to the register |
| OUT | Provides output (stored data in the register) |

k. 16-bit Adder: Standard 16-bit adder, made using 1-bit adder as sub-circuit
   i. Abstraction –




   ii. Signal Description – Similar to 1-bit adder, but operates at a data width of 16 bits.