North South University

Department of Electrical & Computer Engineering

Lab Report

Experiment No:	**04**
Experiment Title:	**Design of a Register File**
Course Code:	**CSE332L**
Course Name:	**Computer Organization & Architecture Lab**
Name & ID:	**Ishrat Jahan,1921909042**
Date of Experiment:	**18/11/2021**
Date of Submission:	**23/11/2021**

Objective:

1. Understanding what a Register file is and how it works.

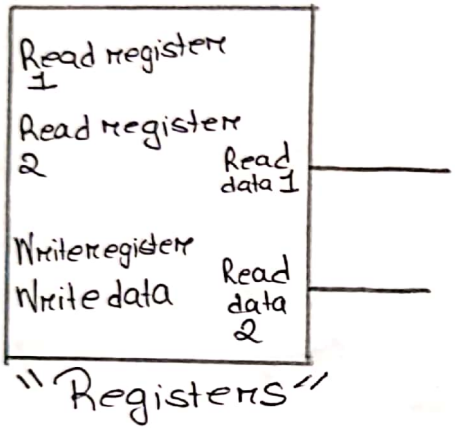
2. Learning about the different components of a Register File.

3. Knowing about the different types of Register File

4. Constructing a Register File for the provided ISA or bit width.

Theory:

Register File: Many registers together make up a Register File. It is not related to disk file. It has high speed. and can partially be called cache. It is usually located inside CPU along with ALU, CU and instruction fetch [Instruction Memory].

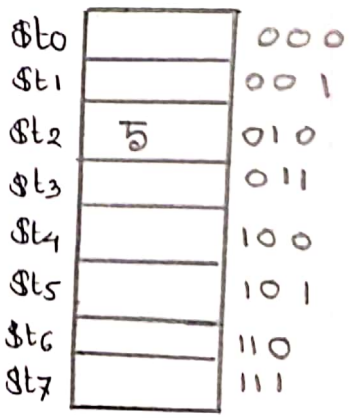


There are different kinds of register such as IR (instruction register), PC (Program register). Instruction register usually instructs the sequence of the way a instruction will be carried out. Besides there are registers to store operands of instruction such as add, sub etc.

The address in a register file is represented as $t0, $t1, $t2, ...

Components of Register: A register has the following components ——
i) Read register 1
ii) Read register 2
iii) Write register.
iv) Write data.
v) Read data 1
vi) Read data 2.

Read register 1: It reads the address of the place whose data we want to read. For example if we want the value -5, it will read the address 010.



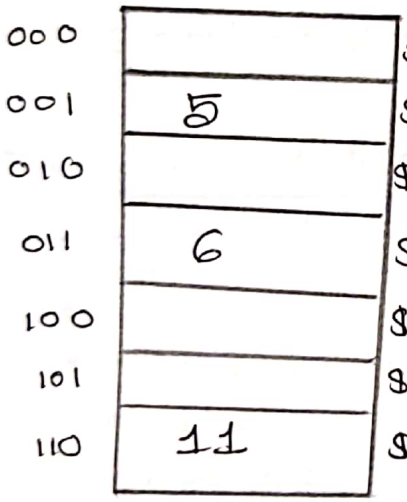
Read register 2: It also reads the address of the place whose data we want to read.

Read data 1 / Read data 2: They read the data value stored in the addresses read by read register 1 and 2. For example – Read data 1 will have 5.

Write data: It stores the data output after a certain operation is carried on the data's read.

Write register: It fetches the address, where we want to store the data which the write data received.

For example:- If read data 1 reads 5 and read data 2 read 6 from two addresses and we carry out an addition between them, we get 11 as the output.



Then we have to pass an address, to the write register where we want to store the 11. The output data 11 is initially stored in write data and won't be written in the register unless an address is passed.

R-type Register:

OP	RS	Rt	Rd.

i) OP = It stores instructions like Add, Sub, XOR, NOT etc. If the register is 8 bit, there will be 8 types of operation there.

ii) RS = Read Register 1

iii) rt = Read Register 2

iv) rd = Write Register.

The rs, rt and rd must be of same bits.

ISA: It stands for instruction set architecture. It has the machine code of the instructions that will be carried out.

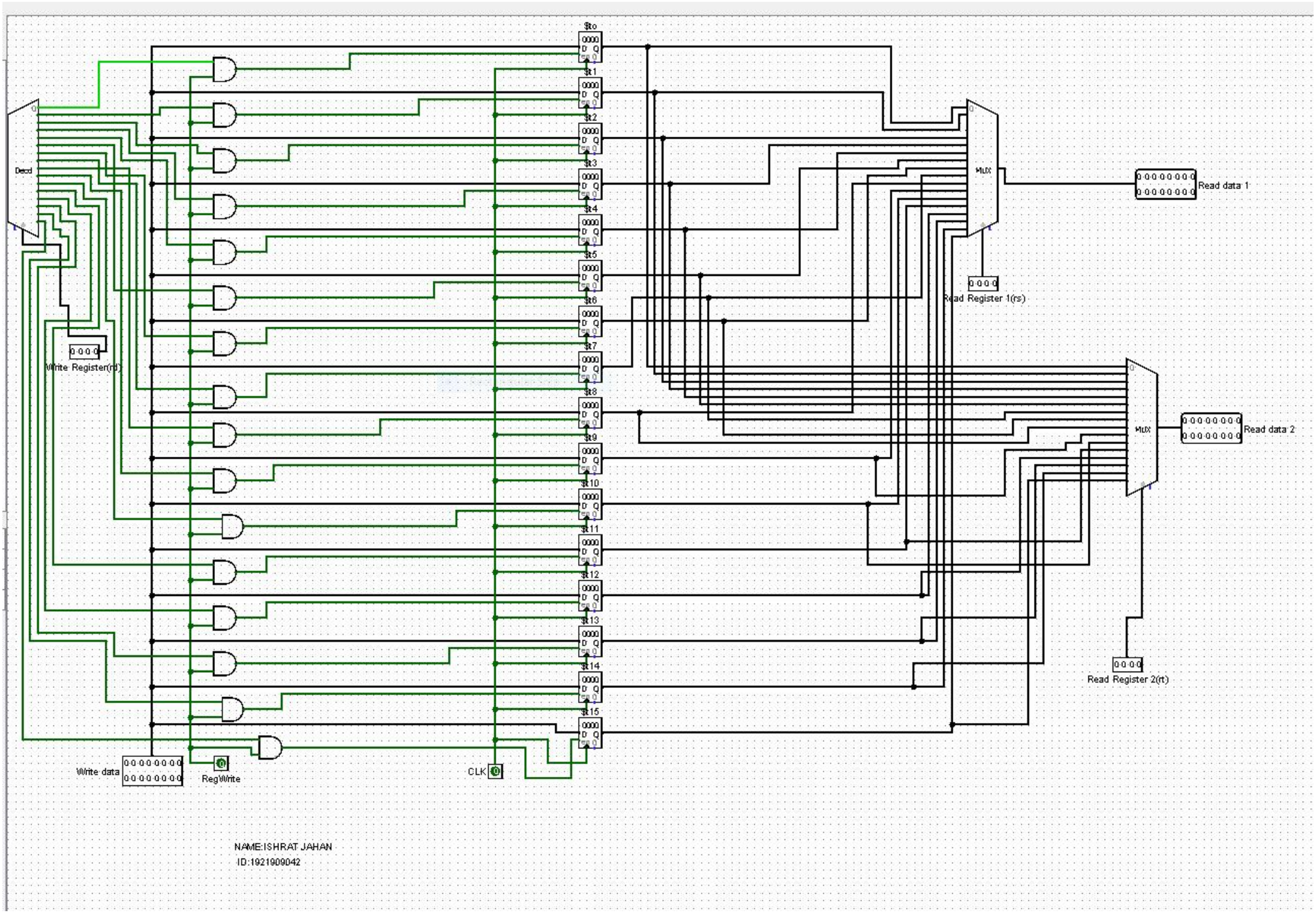
For example: If op is 4bit, rs = rt = rd = 4 bit, then it is called 16 bit ISA.

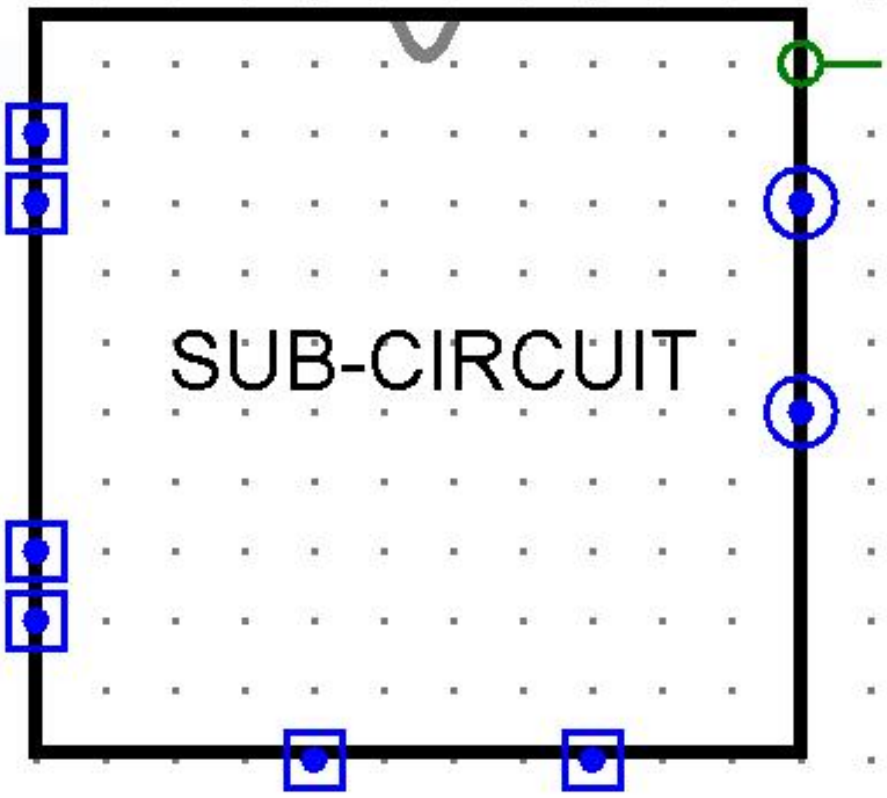
We can also workout op or rs, rt, rd from ISA. For example, if ISA = 15 bit and op = 3bit,

rs = rt = rd = (15 − 3)/3 = 4 bit.

Read data 1 / Read data 2 bits: If we have a 8 bit register, then $[2^3 = 8 - 1 = 0 - 7]$. It will be able to read data's between 0-7. As per our previous example of 8 bit register, addition of 5+6 results in 11 which doesn't fall between 0-7. Therefore we need a high data bit. It's usually kept as 2^{16} ~~[0 − 65536]~~ [0 − 65555], so we can keep many data. The other option is to set their data bit the same as the ALU.

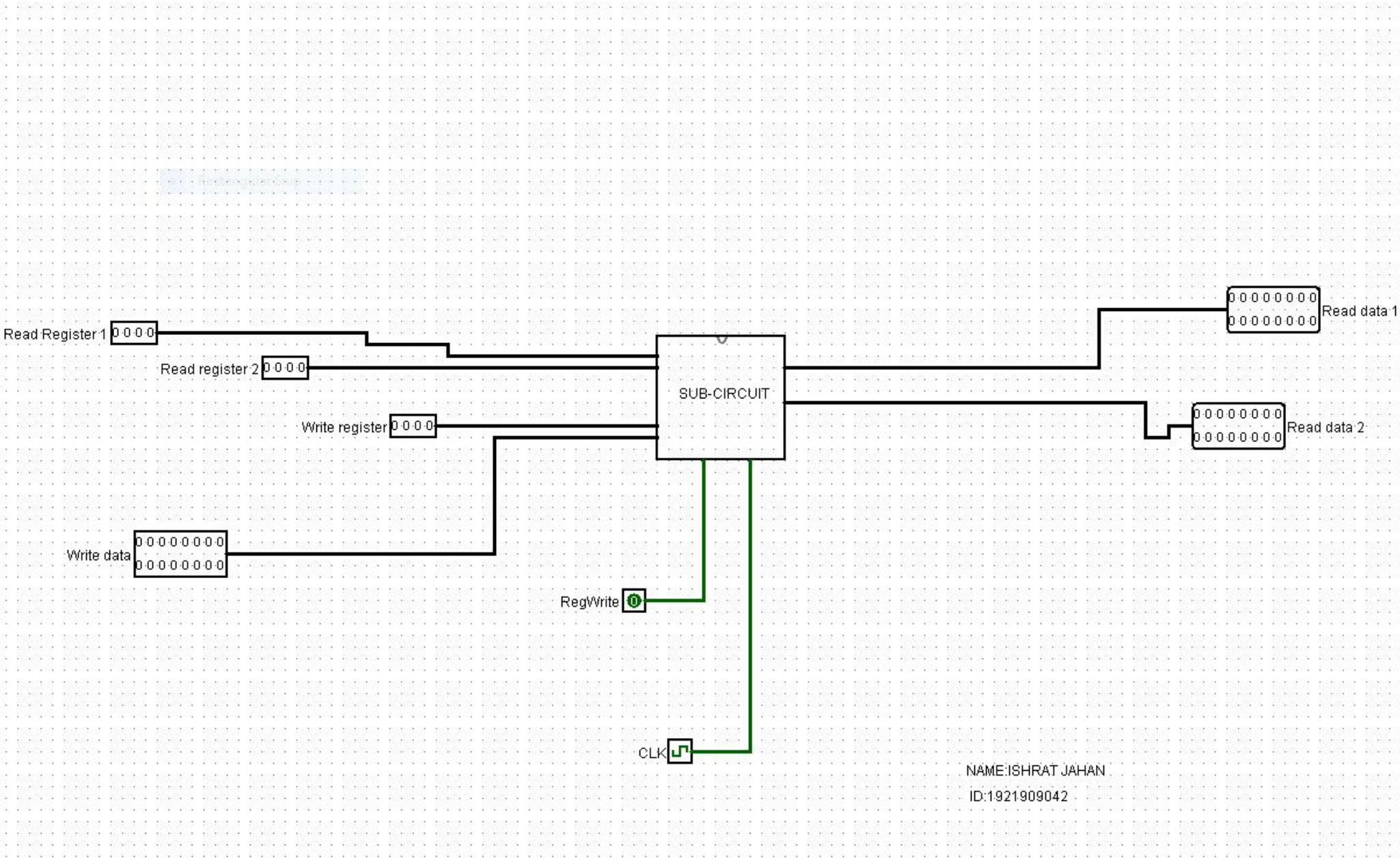
RegWrite: It controls if we will write the data on the Register File or not. If it's 1, the data is written and if it's 0, the data is not written.


$t0
$t1
$t2
$t3
$t4
$t5
$t6
$t7
$t8
$t9
$t10
$t11
$t12
$t13
$t14
$t15
Decd
MUX
MUX
Read data 1
Read data 2
Read Register 1(rs)
Read Register 2(rt)
Write Register(rd)
Write data
RegWrite
CLK
NAME:ISHRAT JAHAN
ID:1921909042



Name:ISHRAT JAHAN

ID:1921909042


Read Register 1
0 0 0 0
Read register 2
0 0 0 0
Write register
0 0 0 0
Write data
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
RegWrite
CLK
SUB-CIRCUIT
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Read data 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
Read data 2
NAME:ISHRAT JAHAN
ID:1921909042

Discussion

The experiment of Lab-04 was based on "Design of a Register File". In this lab we learned how a Register File works and how to design one in Logisim.

At first, we studied where and what a Register File is. Register file is a collection of many registers and it is inside CPU along with ALU, CU and instruction memory. We know what an ALU does and how CU controls which actions will be performed in what order. Instruction memory on the otherhand have all instructions inside it which are fetched and then are carried out. Secondly, we learned that a Register file is not related to a disk file, it has high speed and can be partially called a cache memory. Then we proceeded to know about some special purpose registers as IR and PC and some general purpose registers which stores operands of instructions such as Add, Sub, Division etc. We also studied what operands are.

Then we proceeded on the details of how a Register File works. Firstly, we learnt the addresses in a Register file is written as $t0, $t1, $t2.... Secondly, we learnt about the basic structures in it. It has Read Register 1, Read Register 2, Read data 1, Read data 2, Write data and Write Register. The function of Read Register 1 and Read Register 2 is to receive the address of the place in the register file whose data we want to read. On the otherhand, Read data 1 and Read data 2 reads the data on the address's read by Read Register 1 and Read Register 2. We are receiving two data or

have only two inputs because as we normally take two inputs while carrying out a logic or arithmatic operation. Also an ALU can take two inputs and if we want more inputs we need to use a storage device. When the ALU performs the desired operation on the Read data 1 and Read data 2, the output is written on Write Data. only if any address is provided in the write Register. That is, Write Data stores the output and will write it on the Register file only if an address is provided for it to write. For example:- If Read Data 1 read 2 and Read Data 2 read 4, and the ALU add them, the output is 6. This output 6 is stored in the Write data and will be written on register file if Write Register has an address to read.

As we were suppossed to study R-type Register File, we only focused on it's format although we were introduced to J-type and J-type. We also learnt what an ISA is. It stands for instruction set architecture. For example: 16 bit ISA means the machine code for instruction is 16 bit. As for the format of R-type register, it has OP which stores instructions like Add, sub, XOR, rs (read register 1), rt (read register 2) and rd (Write ~~data~~ register). Also during design there will be an input pin labelled RegWrite which controls if the data on Write Data will be written on the register or not.

To be able to understand the bits of each section of the R-type register, we were provided few examples. For example, if it's said the ISA is 15 bit, OP is 3bit rs, rt, rd

will be 4 bit each. The bit of three of them will always be same. As for displaying the output we can keep the data bits of ALU and Read Data 1 and Read Data 2 same or use 2^{16} bit's as output as it will be able to display between 0 — 65555. Any data between these range calculated by the ALU will get displayed. The theoretical part of the lab ended after this.

To make it easier for us, our lab instructor showed us how to create a Register File with Register Address bit = 2, Number of Register's = 4 and Data bit of the register = 8. We carefully followed her instructions and designed this register file as practice. After that we were asked to complete our lab task which was to design a 16 bit wide Register File.

As per the instructions, a 16 bit wide register file will require 16 registers. each of data bit 16. As the Number of register is 16, Address bit will be 4 bits. The first step was to create the register file as sub-circuit. Therefore, in Logisim we used the add-circuit to create a project named Register File which will have the register. For that, I took 16 registers and placed them in vertical order with their data bit's changed to 16. The I took a input pin, labelled it as "CLK" and connected it to the clock portion of all registers. As it will be used as ~~a~~ sub-circuit, I used an input pin for clock. I also labelled all the registers as $t0, $t1 upto $t15. For the data connection, I took an input pin of 16 bits, labelled it as Write Data and connected it to all the register's D input.

For the next step I used a decoder with select bits 4, as it's 16 bit register. We needed a decoder as we needed to select one of a number of connected ~~devid~~ devices to activate in response to the input. One of the select bit in the decoder was connected with an input pin of 4 bit and labelled as Write Register. We needed to control whether the read data will be written or not, so I took an input pin, labelled it as RegWrite and connected it to an AND gate with the other input being the input from the decoder. I used 16 AND gates for this and the outputs of the AND gates were connected to the registers' enable input.

The next step was to use MUX for display of the output. For that, I took two MUX from the plexers section changed it's data bits to 4. As a result, both of them had 16 input each. I connected the output(0) from the register's into both of the MUX. Moreover, I took two input pin, changed it's data bits to 4, labelled it as Read Register 1 and Read Register 2 and connected it to MUX's select bit. Then I took two output pin, changed it's data bits to 16, labelled them as Read data 1 and Read data 2 and connected them to the MUX. The Circuit of the Register File was complete.

Next, I proceeded to make the sub-circuit. For that, I clicked on the subcircuit image and adjusted the subcircuit accordingly. The left side had Read Register 1, Read Register 2, Write Register and Write data.

The lower portion had the RegWrite and the clock while the right side had Read data 1 and Read data 2. Then I clicked on the main in Logisim, dragged the subcircuit of the Register File by click and drag on the file 'Register File'. Then I took input pin of 4 data bits and connected to Read Register 1, Read Register 2 and Write Register. Secondly, I took another input pin of 16 data bit, and labelled it as Write data and connected to Write data port. I took a clock from wiring section and connected it to where it's suppossed to be. After that I took another input pin of data bits 1, labelled it as RegWrite (Controller) and connected it to its desired area. This for both the Read data 1 and Read data 2, I took two output pins and connected then on the right side of the sub-circuit by changing their data bits to 16. The circuit was completed. After completion, I checked the outputs. The circuit worked correctly and the experiment was successful.

The only problem I faced in this experiment, is the circuit of 16 bit wide register file was too big and congested. So, there were chances, I kept forgetting to add some connections. Moreover, there were also very less space as there were many components in the circuit, so I had to be very careful while connecting the inputs. Overall, it ~~was~~ is easy to construct if the register file is designed cautiously with enough time on hand.