# Computer Architecture
# Lab #1
# Register File

Yusuf Maged Elnady – 20012286 – EED2025
Ziad Mohammed Elbouriny – 20010643 – EED2025

## Design:

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_file is
    port (
        clk, RegWrite: in std_logic;
        write_data: in std_logic_vector (31 downto 0);
        rreg_index1, rreg_index2, wreg_index: in std_logic_vector (4 downto 0);

        read_data1, read_data2: out std_logic_vector (31 downto 0)
    );
end reg_file;

architecture behavior of reg_file is
    type reg_vector is array(0 to 31) of std_logic_vector(31 downto 0);
    signal stored_d: reg_vector := (others => (others => '0'));
begin
    process (clk)
    begin
        if clk = '1' then
            if RegWrite = '1' then
                stored_d(to_integer(unsigned(wreg_index))) <= write_data;
            end if;
        else
            read_data1 <= stored_d(to_integer(unsigned(rreg_index1)));
            read_data2 <= stored_d(to_integer(unsigned(rreg_index2)));
        end if;
    end process;


end architecture behavior;
```

## Test Bench:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity lab1_tb is
5  end lab1_tb;
6
7  architecture test of lab1_tb is
8      component reg_file
9          port(
10             clk, RegWrite: in std_logic;
11
12             write_data: in std_logic_vector (31 downto 0);
13             rreg_index1, rreg_index2, wreg_index: in std_logic_vector (4 downto 0);
14
15
16             read_data1, read_data2: out std_logic_vector (31 downto 0)
17         );
18     end component;
19
20     signal clk, RegWrite: std_logic;
21     signal write_data, read_data1, read_data2: std_logic_vector (31 downto 0);
22     signal rreg_index1, rreg_index2, wreg_index: std_logic_vector (4 downto 0);
23 begin
24     smth: reg_file port map(clk, RegWrite, write_data, rreg_index1,
25                             rreg_index2, wreg_index, read_data1, read_data2);
26
27     stimulus: process is
28     begin
29         clk <= '1';
30         RegWrite <= '0';
31         rreg_index1 <= "00000"; rreg_index2 <= "00001";
32         wreg_index <= "00000";
33         write_data <= x"ABCDABCD";
34
35         wait for 10 ns;
36
37         clk <= '0';
38
39         wait for 10 ns;
40
41         clk <= '1';
42         RegWrite <= '1';
43
44         wait for 10 ns;
45
46         clk <= '0';
47
48         wait for 10 ns;
49
50         clk <= '1';
51         RegWrite <= '1';
52         rreg_index1 <= "00110"; wreg_index <= "00110";
53         write_data <= x"ABBAABBA";
54
55         wait for 10 ns;
56
57         clk <= '0';
58
59         wait for 10 ns;
60         wait;
61     end process stimulus;
62
63 end architecture test;
```

## Test Cases:

### 1st Clock Cycle:

"RegWrite" is disabled, while the chosen "read registers" are the first and second registers, and the first register is chosen as the "write register", which would store the value "ABCDABCD" in the nearest first half cycle when "RegWrite" is enabled.

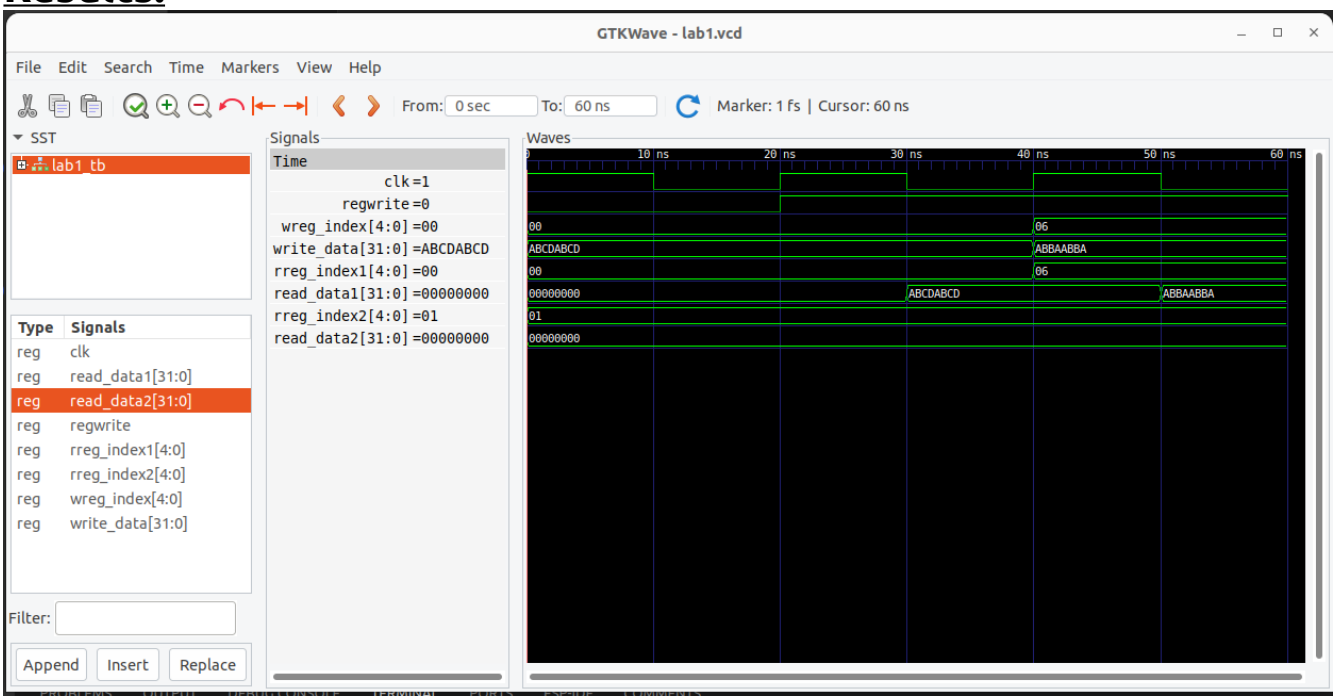
### 2nd Clock Cycle:

"RegWrite" is enabled.

### 3rd Clock Cycle:

The register file attempts to read and write to the same register (00110) in the same instant.

## Results:




In the first clock cycle, no data is written to the register file since "RegWrite" is disabled, and since the register file is initially empty, zeros are read.

In the second clock cycle, The data is written in the first half cycle, and then read (from the "read_data1" register output) in the second half cycle.

In the third clock cycle, the register file attempts to read and write data at the same time, and so the data is written in the first half cycle and then is immediately read in the second one.

## Code Link: https://github.com/yusufElnady/CA-Labs/tree/main/Lab1