

The background features several thin, light-colored lines that form abstract, angular shapes. These lines are scattered across the dark blue background, with some extending from the edges towards the central text box. The lines vary in length and orientation, creating a sense of dynamic movement and geometric complexity.

# RegisterFile Decode

# TABLE OF CONTENTS

**1. INTRODUCTION TO REG FILE**  
Getting acquainted with MIPS different instruction types

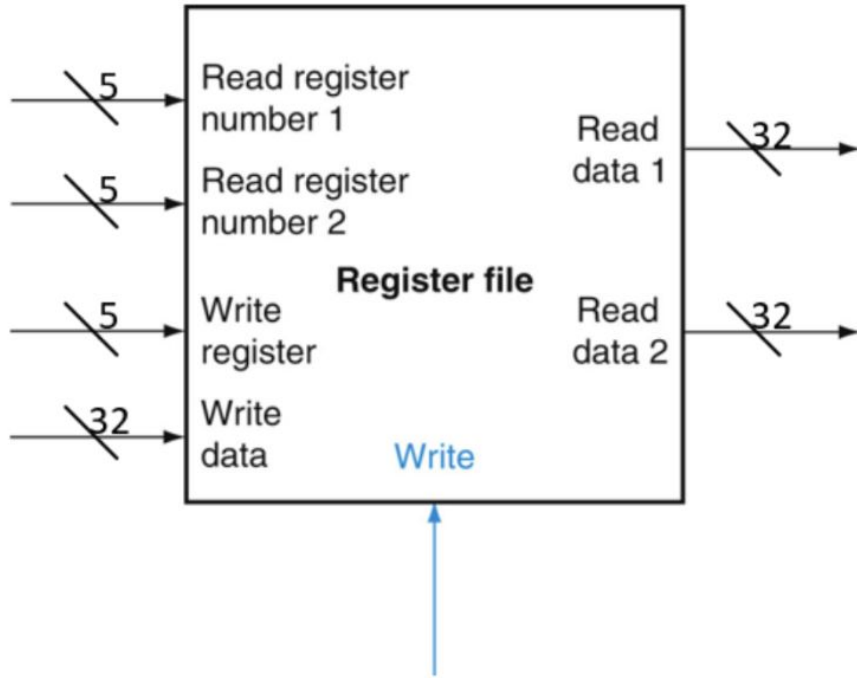
**2. IMPLEMENTATION OF REG FILE**  
Provide an introduction on how register file works

**3. MIPS INSTRUCTION TYPES**  
Find out about different components required for implementing register file

**4. FINALIZE DECODE STAGE**  
Finalize the second stage of pipeline

**5. INPUTS & OUTPUTS**  
Describe inputs and outputs of required modules

# 1. INTRODUCTION TO REG FILE



## HINTS:

1. We can read two registers simultaneously but only write one at a time.
2. As 5 bits are used for indicating a register number, there can be a total of 32 registers available.

## 2. IMPLEMENTATION OF REG FILE

### INSTRUCTIONS:

1. Initialize 32 registers each 32 bits wide in the Initial block of the register file module. Make sure to initialize **Zero Register** to zero.
2. **Zero Register** should always have a value of zero, so make sure you don't write to it. So, if zero register is being written into, prevent this and display a message indicating that this operation is forbidden.
3. Since reading doesn't need any special control, it could be done outside of the always block, but writing should be done inside an always block sensitive to the posedge, and only when the **RegWrite** signal is 1.

### 3. MIPS INSTRUCTION TYPES

#### R-TYPE

Add, Sub, Or, And, ... are examples of this type.

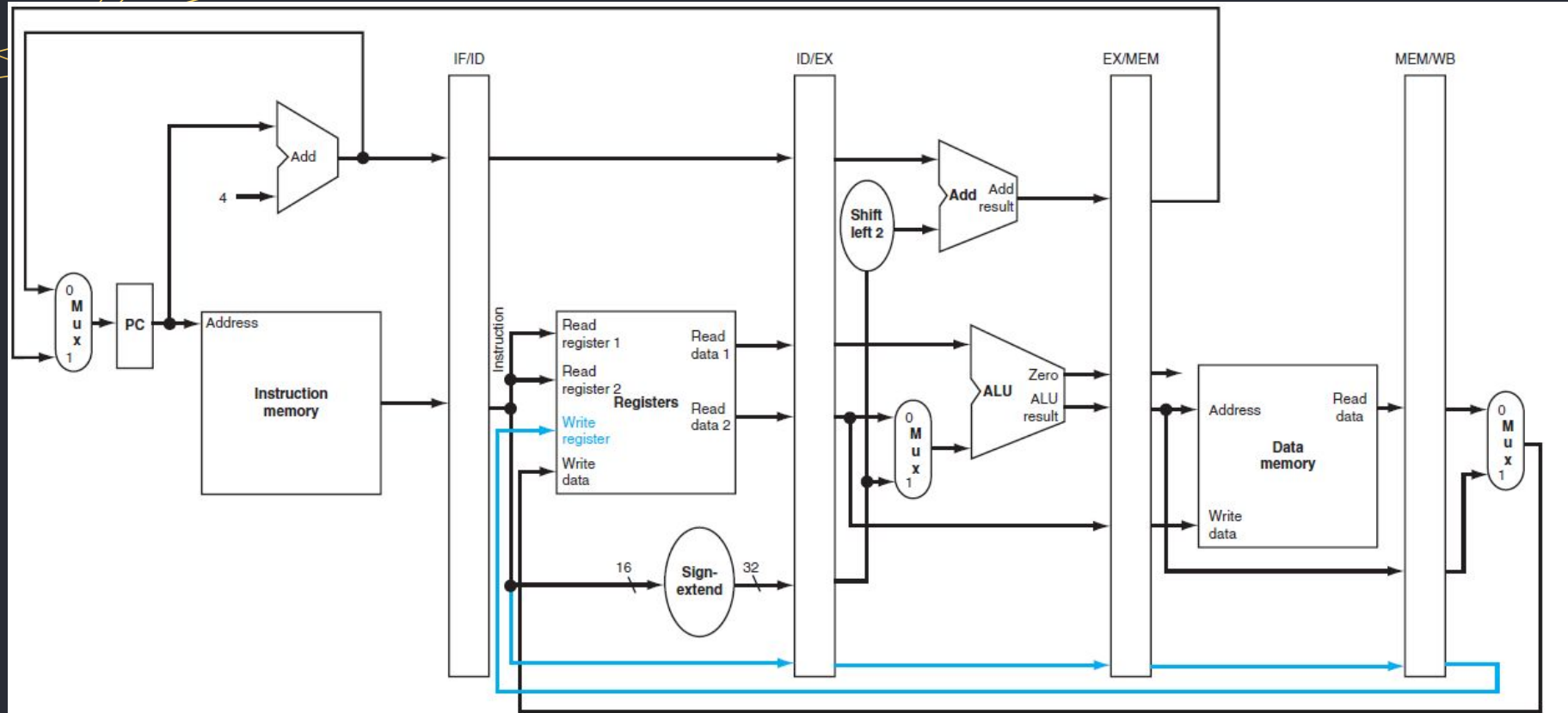
0	rs	rt	rd	shamt	funct
31:26	25:21	20:16	15:11	10:6	5:0

#### I-TYPE

Addi, Ori, Andi, LW, SW, ...

opcode	rs	rt	address
31:26	25:21	20:16	15:0

## 4. FINALIZE DECODE STAGE



## 5. INPUTS & OUTPUTS

Finally, it's time to instantiate the modules and connect them together. Modules which need to be instantiated and connected are as follows:

### **RegisterFile:**

- Inputs: clk (1 bit), RegWrite (1 bit), read\_reg1 (5 bits), read\_reg2 (5 bits), write\_reg (5 bits), write\_data (32 bits)
- Outputs: read\_data1 (32 bits), read\_data2 (32 bits)

### **Decode:**

- Inputs: clk (1 bit), instruction (32 bits)
- Outputs: opcode (6 bits), read\_data1 (32 bits), read\_data2 (32 bits), sign\_extended\_immediate(32 bits), rt (5 bits), rd (5 bits)

**SignExtend** (sign extends a 16-bit signal to a 32-bit signal):

- Inputs: clk value (16 bits)
- Outputs: sign\_extended\_value (32 bits)