MIPS Processor

The goal of this project was to create a multi-cycle processor using Verilog hardware description language. The processor comprises two main components: the **Control Unit** and the **Datapath**. The Control Unit is implemented as a finite state machine that determines the flow of operations within the processor. The Datapath, on the other hand, encompasses the following stages:

1. Instruction Recall Stage (IF)
2. Command Decoding Stage (Decode)
3. Instruction Execution Unit (ALU)
4. Memory Access Level (Mem)

Each of these stages plays a crucial role in the functioning of the processor, and the entire system is designed to ensure maximum efficiency and speed.

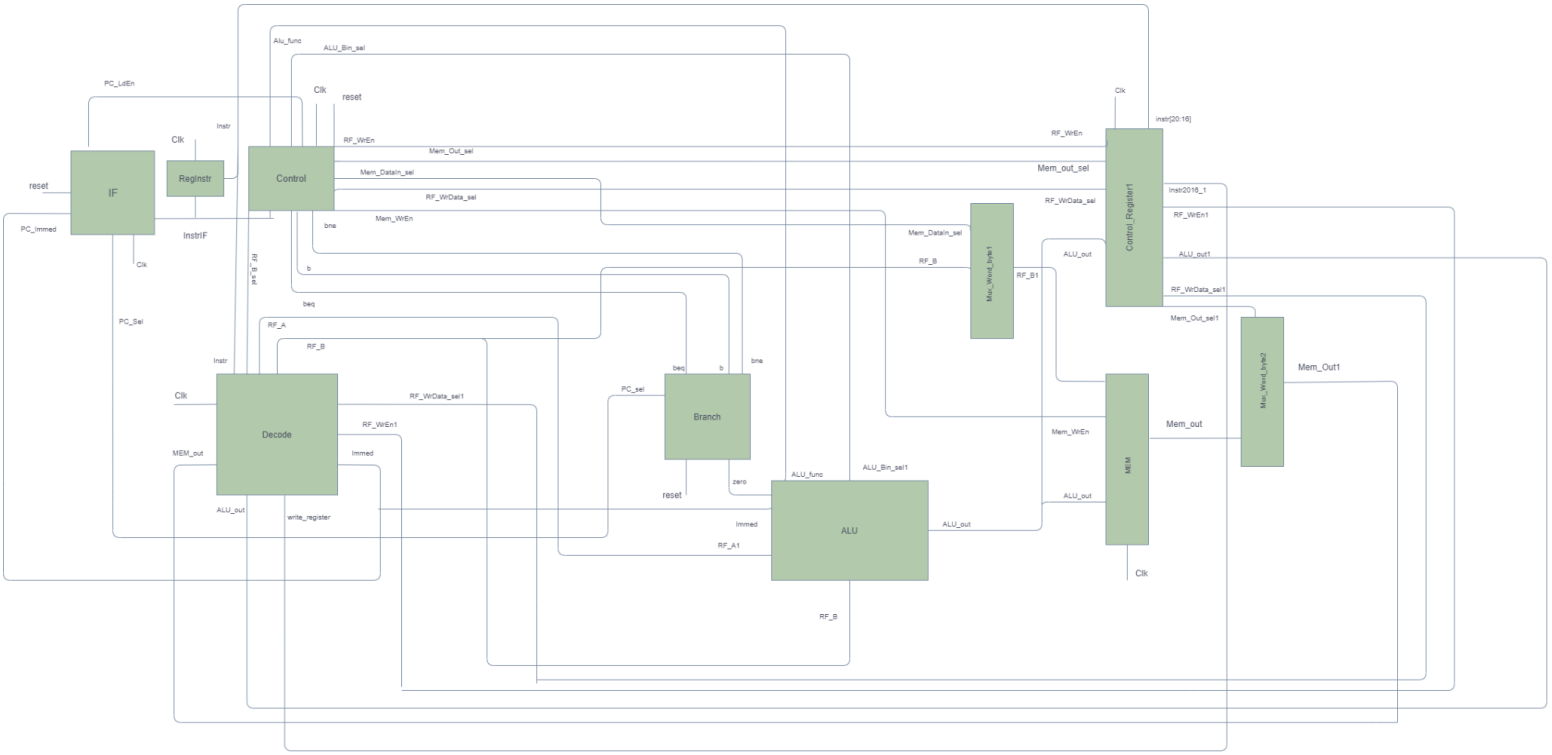
In the **Instruction Recall Stage** (IF), the processor retrieves the instruction from memory and prepares it for further processing. This stage is responsible for fetching the instruction and making it available to the next stage of the pipeline.

The **Command Decoding Stage** (Decode) takes the instruction retrieved in the previous stage and decodes it to determine what operation needs to be performed. This stage is crucial as it generates the necessary signals for the Instruction Execution Unit (ALU) to perform the operation.

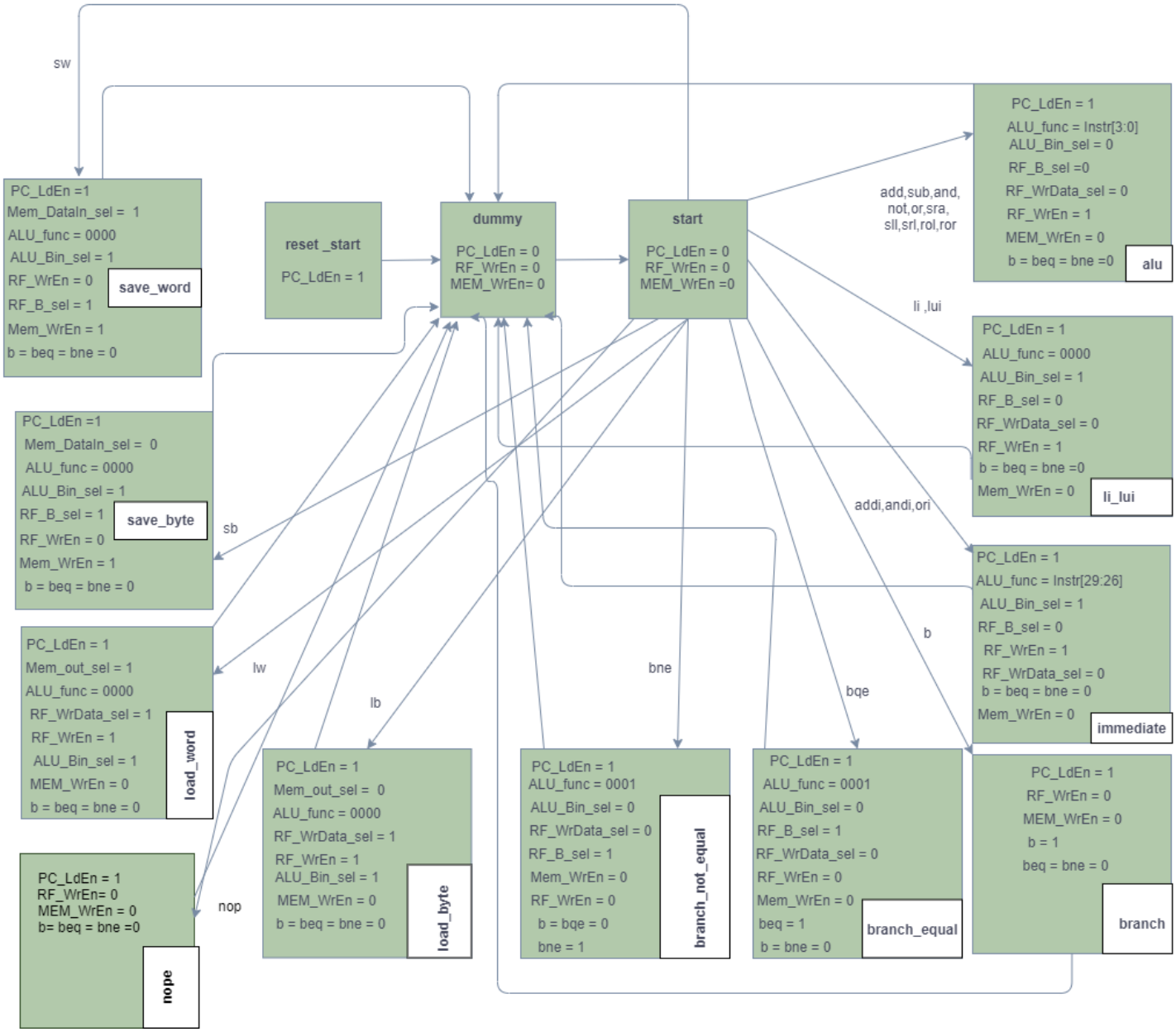
The **Instruction Execution Unit** (ALU) performs the actual operation specified by the instruction. This stage is responsible for executing the arithmetic and logical operations specified by the instruction.

Finally, the **Memory Access Leve**l (Mem) accesses the memory to either read or write data. This stage is responsible for reading data from memory and writing the results of the ALU operations back to memory.

The Control Unit and the Datapath of the MIPS processor are presented in the following diagram.



Also the Finite State Machine that is used in the Control Unit is presented in the following diagram.



The processor requires three clock cycles to execute each instruction. At the start, two additional cycles are required for resetting and to allow the first instruction to reach the control unit.

In the **first cycle**, after the reset, the instruction is stored in the **regInstr** register. The control unit then provides the necessary control signals to the processor, depending on the instruction. These signals are sent to the **decode stage**, the **ALU**, the **branch module**, the **RAM**, and the **Control_register1** register. The ALU performs the required operation and the branch module selects the PC increment value based on whether or not a branch instruction is present. The Control_register1 register holds

the data for the next cycle, and the **IFSTAGE** is instructed to increment the PC (PC_LdEn = 1).

In the **second cycle**, data is fetched from the Control_Register1 register and from the RAM if required. The PC value is not incremented again (PC_LdEn = 0).

In the **third cycle**, the data from the Control_register1 register or the RAM is written to the register in the decode stage that requires it. At the same time, the new instruction has arrived at the control input and the **regInstr** register.

The control unit of the datapath, depending on the specific command being executed, determines the values of several control signals, which are:

- ALU_func: Specifies the operation to be performed by the ALU.
- ALU_Bin_sel: Determines the second input of the ALU.
- RF_WrEn: Controls whether the output from the ALU or RAM will be written to the register in the Decode stage.
- Mem_out_sel: Decides whether the output from RAM will be the full word stored at the location indicated by the ALU's result or just the last byte of the word.
- Mem_DataIn_sel: Controls whether the entire content of the RF_B register will be written to RAM or just the last byte.
- RF_WrData_sel: Specifies the source (RAM or ALU) of the data to be written to a register in the Decode stage.
- Mem_WrEn: Controls whether data will be written to RAM.
- bne: Indicates if the command is "bne".
- b: Indicates if the command is "b".
- beq: Indicates if the command is "beq".
- RF_B_sel: Defines what the second register will be during the reading process in the Decode stage.
- PC_LdEn: Controls whether the PC should be incremented or not.

The processor units have been enhanced with the following features:

- Branch Module: If the command to be executed by the processor is a branch type, it accepts the zero output from the ALU as input and decides if the next instruction will be at the PC + 4 or PC + 4 + Immed position.
- Mux_word_byte1: It selects whether to write the full contents of the RF_B register to RAM or just the last byte.

- Mux_word_byte2: It chooses whether the output from RAM is the complete word located at the position indicated by the result of the ALU or just the last byte of the word.
- Control_Register1: It stores some signals from the control module and from other units that will be required in the next cycle where the data is recorded.

During the development of the processor, some challenges were encountered. The commands lb, lw, sb, sw require either the last byte or the entire word from RAM as input or output respectively. To resolve this issue, the Mux_word_byte1 and Mux_word_byte2 multiplexes were introduced. Also, since RAM takes a cycle to produce data, the Control_Register1 register was added. This register holds signals from the control unit and other units that will be necessary in the next cycle. An additional state (dummy) was added, as it takes an extra cycle for the correct command to reach the FSM and for the data to be written to the DECODE registers. Furthermore, because the FSM also takes a cycle to produce values, the regInstr register was introduced to hold the command and provide it after one cycle. Finally, the write input register was added to DECODE, which indicates the destination register for the data. This is necessary because both the data and its corresponding register address come from the Control_register1 register.