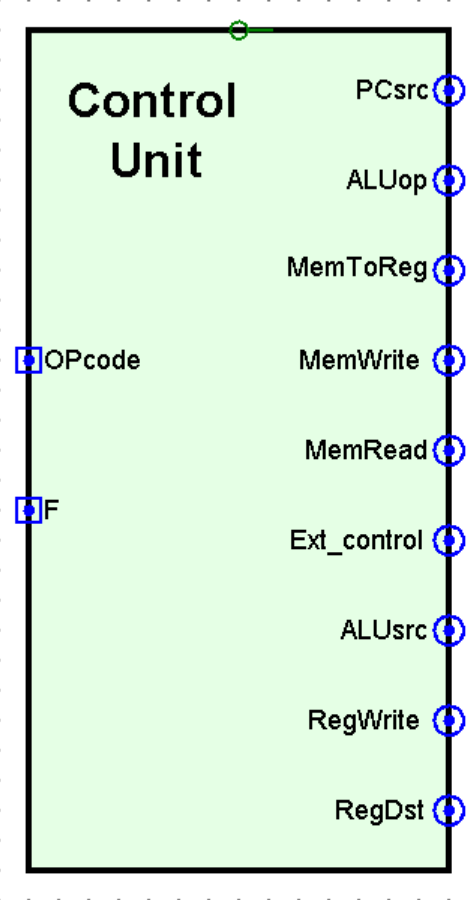
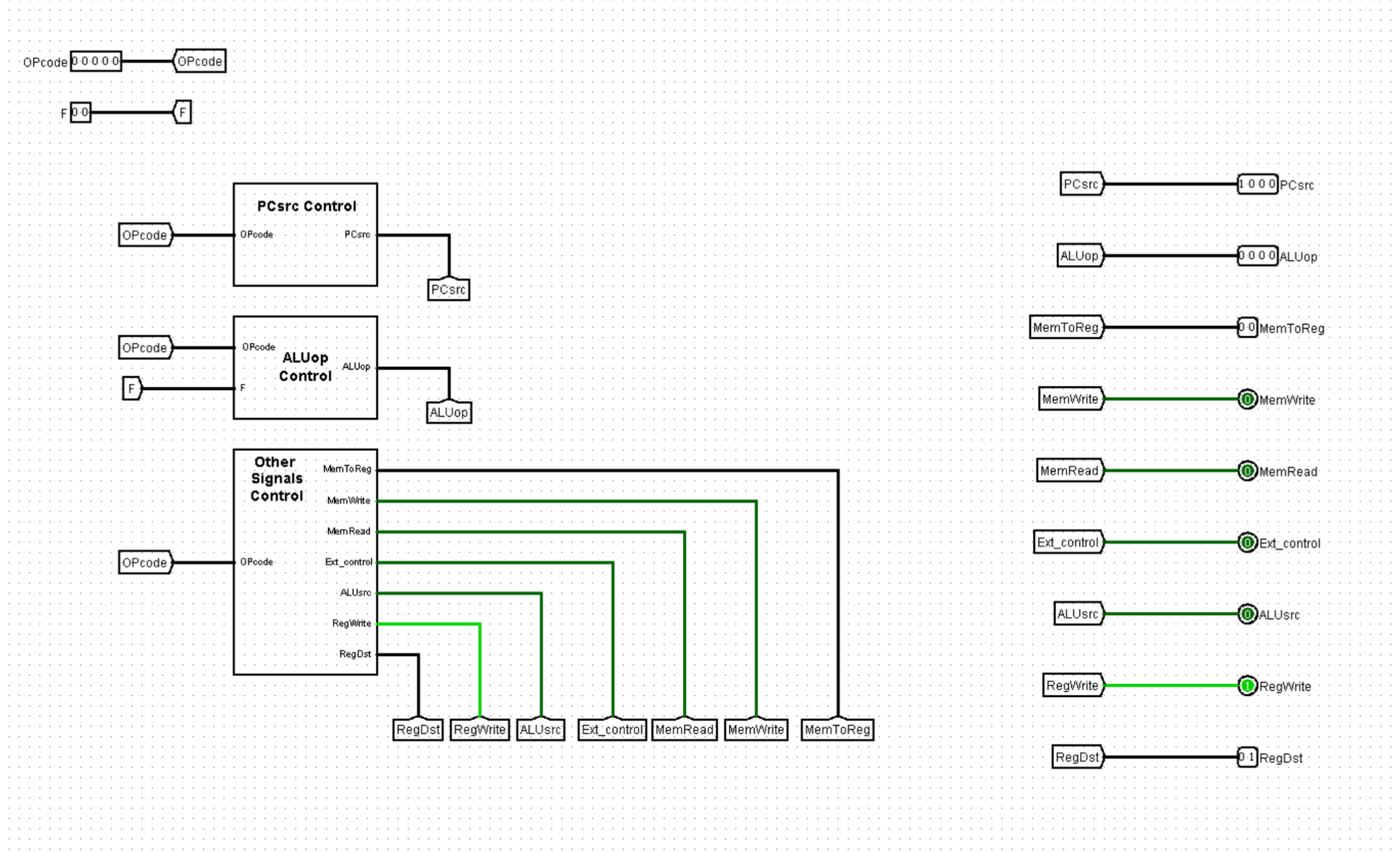
**2.7. Control Unit**

Utilising the 5-bit Opcode and 2-bit Function, the control unit orchestrates the execution of various operations based on the decoded instruction.

* ***The internal design***

**7.1. Input signals**

***7.1.1. The Opcode*** *(5-bit)****:***

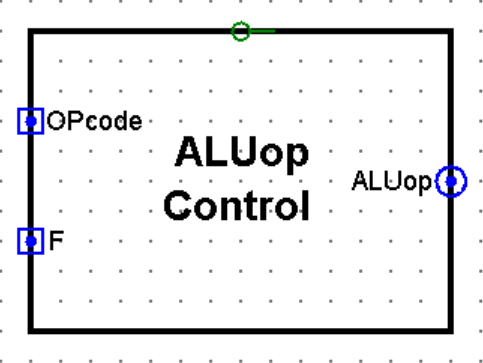
The opcode tells the processor what to do (e.g., add, and, load, store, branch, jump…etc)

***7.1.2. The Function*** *(2-bit)****:***

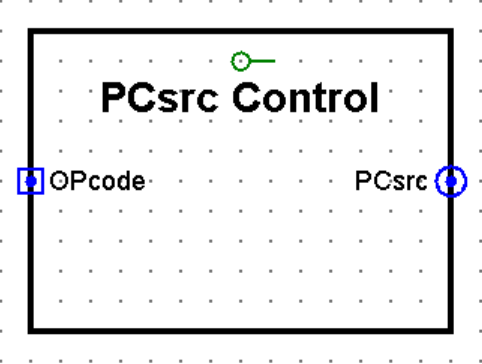
The R-type instruction format utilises the 2-bit function field to precisely control ALU operations.

**7.2. Output signals**

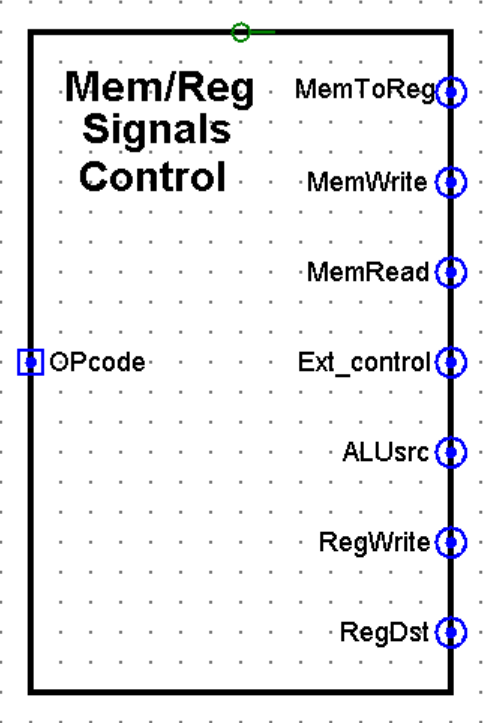
The output signals originate from three distinct subcircuits, each responsible for regulating various facets of the processor's functionality.

***7.2.1 ALU signal generator:***

Utilising the Opcode and Function, It generates the **ALUop signal, a** **4-bit** **code** that dictates the ALU operation to be performed. The lower 2 bits determine the operation type (arithmetic, logic, or shift), while the upper 2 bits specify the particular operation (add, sub,etc., and, or, etc., sll, srl, etc.).

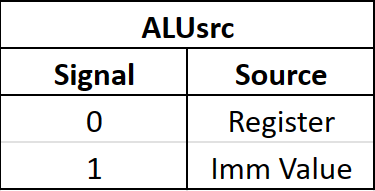
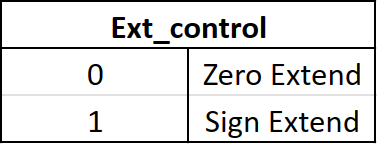
***7.2.2 PC signal generator:***

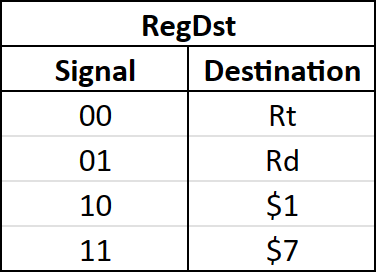
Itutilises the Opcode to generate a **4-bit PCsrc signal.** This signal determines the next Program Counter (PC) value, selecting between incrementing by 1, jumping to a specified address, or branching based on certain conditions.

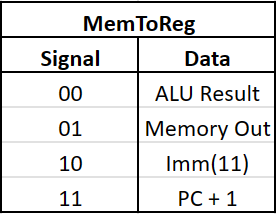


***7.2.3 Mem/Reg signals generator:***

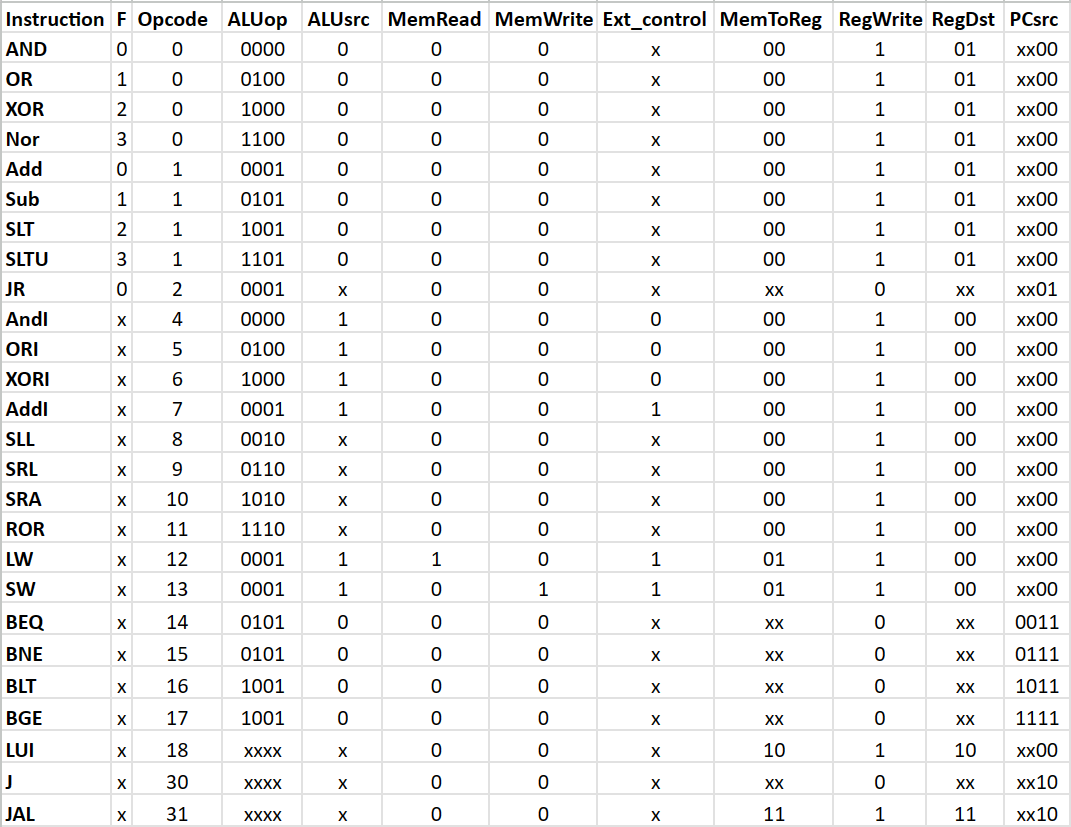
It employs the Opcode to derive values for the remaining control signals necessary for proper operation. These signals encompass various aspects of the CPU operation, such as memory access, register manipulation, and data transfer.

* **ALUsrc (1-bit):** This signal dictates the source of the second operand for the ALU. A logic value of 1 instructs the ALU to utilise the immediate value from the instruction, while a 0 specifies a register value (typically rt in the instruction).
* **Ext\_control (1-bit):** This signal controls the extension mode for immediate values. A logic value of 1 keeps the sign of the immediate, while a logic value of 0 sets the empty bits to zero.
* **RegWrite (1-bit):** This control signal governs write operations to the register file. A logic value of 1 enables writing, allowing data to be stored in the designated register. Conversely, a logic value of 0 disables writing, preventing any changes to the register's content.



* **RegDst (2-bit):** This control signal selects the destination register for write operations based on the instruction type. It has different values depending on the instruction type: 0 for I-type instructions, 1 for R-type instructions, and 2 or 3 for jump instructions.
* **MemRead (1-bit):** This signal tells the processor to grab data from memory. The control unit activates this signal whenever an instruction needs something from memory.
* **MemWrite (1-bit):** This signal, on the other hand, tells the processor to write data to memory. The control unit activates it whenever an instruction needs to store something in memory.
* **MemToReg (2-bit):** This control signal selects the data source for writing to the designated register.

***7.3. Signals values for each instruction***



***7.4. Signals expressions***

| ***ALUop\_0*** | ~OP3 ~OP2 OP0 + ~OP3 ~OP2 OP1 + ~OP3 OP1 OP0 + OP3 OP2 + OP4 |
| --- | --- |
| ***ALUop\_1*** | OP3 ~OP2 |
| ***ALUop\_2*** | ~OP4 ~OP3 ~OP2 ~OP1 F0 + ~OP3 OP2 ~OP1 OP0 + OP3 ~OP2 OP0 + OP3 OP2 OP1 |
| ***ALUop\_3*** | ~OP3 ~OP2 ~OP1 F1 + ~OP2 OP1 OP0 + ~OP3 OP2 OP1 ~OP0 + OP3 ~OP2 OP1 + OP4 |

| ***PCsrc\_0*** | ~OP4 ~OP3 ~OP2 OP1 + ~OP4 OP3 OP2 OP1 + OP4 ~OP1 |
| --- | --- |
| ***PCsrc\_1*** | OP3 OP2 OP1 + OP4 ~OP1 |
| ***PCsrc\_2*** | OP0 |
| ***PCsrc\_3*** | ~OP1 |

| ***RegDst\_0*** | ~OP3 ~OP2 ~OP1 + OP4 OP3 |
| --- | --- |
| ***RegDst\_1*** | OP4 |

| ***MemToReg\_0*** | OP3 OP2 |
| --- | --- |
| ***MemToReg\_1*** | OP4 |

| ***ALUsrc*** | ~OP3 OP2 + OP3 ~OP1 + OP4 OP1 |
| --- | --- |
| ***Ext\_control*** | OP1 OP0 + OP3 |
| ***RegWrite*** | ~OP4 ~OP3 ~OP1 + ~OP4 ~OP1 ~OP0 + ~OP4 ~OP3 OP2 + ~OP4 OP3 ~OP2 + OP4 ~OP3 ~OP2 OP1 ~OP0 + OP4 OP3 OP2 OP1 OP0 |
| ***MemWrite*** | ~OP4 OP3 OP2 ~OP1 OP0 |
| ***MemRead*** | ~OP4 OP3 OP2 ~OP1 ~OP0 |