UNIT IV PROCESSOR

Instruction Execution

The execution of an instruction in a processor can be split up into a number of stages. How many stages there are, and the purpose of each stage is different for each processor design. Examples includes 2 stages (Instruction Fetch / Instruction Execute) and 3 stages (Instruction Fetch, Instruction Decode, Instruction Execute).

The MIPS processor has 5 stages:

The Instruction Fetch stage fetches the next instruction from memory using the address **IF** in the PC (Program Counter) register and stores this instruction in the IR (Instruction Register)

The Instruction Decode stage decodes the instruction in the IR, calculates the next PC,**ID** and reads any operands required from the register file.

The Execute stage "executes" the instruction. In fact, all ALU operations are done in **EX** this stage. (The ALU is the Arithmetic and Logic Unit and performs operations such as addition, subtraction, shifts left and right, etc.)

The Memory Access stage performs any memory access required by the current **MA** instruction, So, for loads, it would load an operand from memory. For stores, it would store an operand into memory. For all other instructions, it would do nothing.

For instructions that have a result (a destination register), the Write Back writes this **WB** result back to the register file. Note that this includes nearly all instructions, exceptVnops (a nop, no-op or no-operation instruction simply does nothing) and s (stores).

BASIC MIPS IMPLEMENTATION

MIPS implementation includes a subset of the core MIPS instruction set:

- The memory-reference instructions *load word* (lw) and *store word* (sw)
- The arithmetic-logical instructions add, sub, AND, OR, and slt
- The instructions *branch equal* (beq) and *jump* (j), which we add last

This subset does not include all the integer instructions (for example, shift,multiply, and divide are missing), nor does it include any floating-point instructions.

An Overview of the Implementation

The core MIPS instructions includes

- The integer arithmetic-logical instructions,
- The memory-reference instructions, and
- The branch instructions.

For every instruction, the first two steps are identical:

1. Send the *program counter* (PC) to the memory that contains the code andfetch the instruction from that memory.
2. Read one or two registers, using fields of the instruction to select the registersto read. For the load word instruction, we need to read only one register, butmost other instructions require reading two registers.

- For example, all instruction classes, except jump, use the arithmetic-logical unit(ALU) after reading the registers.
- The memory-reference instructions use the ALU for an address calculation, the arithmetic-logical instructions for the operationexecution, and branches for comparison.
- A memory-reference instruction will need to access the memory either to read data for a load or write data for a store.
- An arithmetic-logical or load instruction must write the data from the ALU or memory back into a register.
- Lastly, for a branch instruction, we may need to change the next instruction address based on the comparison; otherwise, the PC should be incremented by 4 to get the address of the next instruction.

Figure 3.1 shows the high-level view of a MIPS implementation, focusing onthe variousfunctional units and their interconnection.

- A logic element need to be added that chooses from among the multiple sources and steers one of those sources to its destination.
- This selection is commonly done with a device called a ***multiplexor***, although this devicemight better be called a ***data selector***.
- The multiplexor selects from among several inputs based on the setting of its control lines. The control lines are set based primarily on information taken from the instruction being executed.

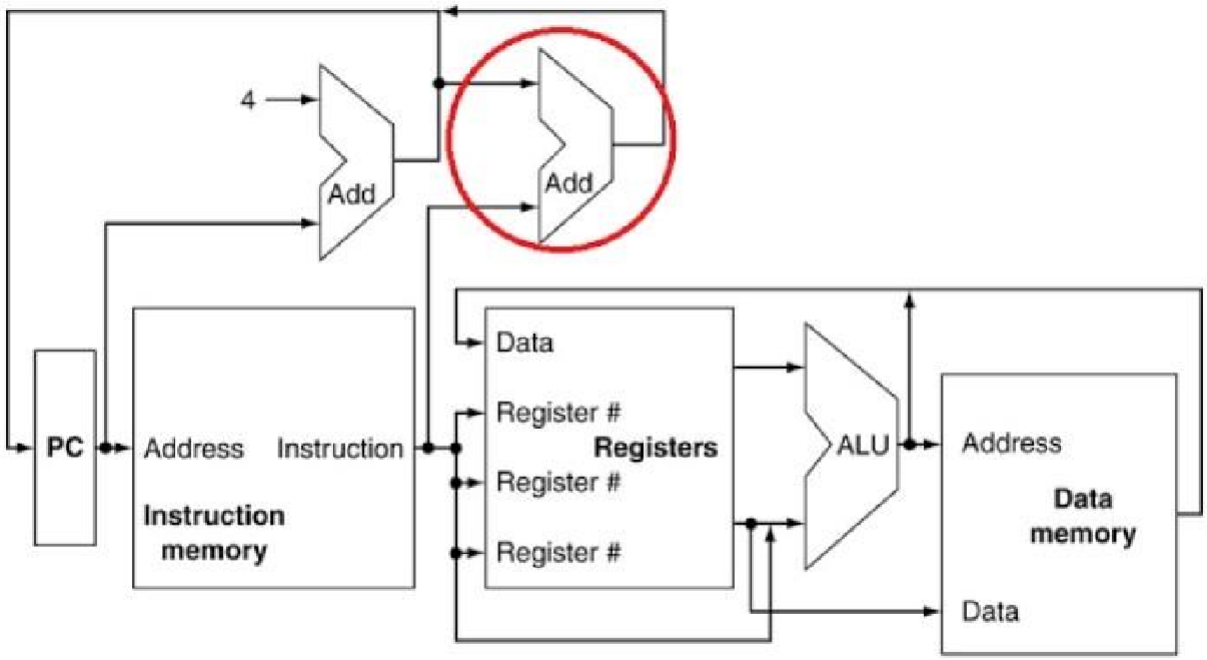

Fig 3.1: An abstract view of the implementation of the MIPS subset showing the major functional units and the major connections between them.

Figure 3.2 shows the datapath of Figure 3.1 with the three required multiplexors added, as well as

control lines for the major functional units. A *control unit*, which has the instruction as an input, is used to determine how to set the control lines for the functional units and two of the multiplexors.

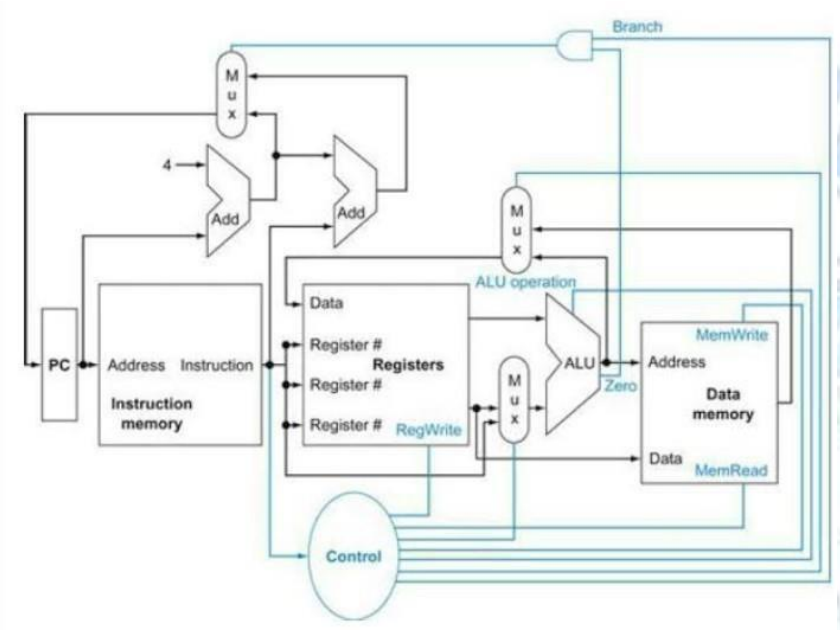

Fig 3.2: The basic implementation of the MIPS subset, including the necessary multiplexors and control lines.

- The top multiplexor controls the value of PC(PC+4 or the branch destination address.
- The middle multiplexor is used to steer the output of the ALU for writing in to the register file.
- The bottom most multiplexor is used to determine whether the second ALU input is from the registers