R TYPE

In this format (R-type), the Program Counter (PC) provides an address that is sent via the address bus to the instruction memory. The corresponding instruction is fetched, and simultaneously, the PC is incremented by 4 to point to the next instruction.

From the fetched instruction, the opcode is decoded to identify it as an R-type instruction. The register file is then accessed to read the values of the two source registers (rs and rt) specified in the instruction. These operand values are forwarded to the Arithmetic Logic Unit (ALU), along with the ALU control signal derived from the funct field of the instruction, to perform the required operation (e.g., add, subtract, AND, OR).

Since R-type instructions do not access data memory, the MemRead and MemWrite control signals are both asserted as 0. The ALU result is then passed through the data path directly to the write-back stage, where it is written into the destination register (rd) in the register file, as enabled by the RegWrite control signal.

I type

In an I-type load instruction (such as lw), the instruction is first fetched and decoded. The base address is obtained from the source register (rs) specified in the instruction, which is read from the register file. The 16-bit signed immediate (offset) from the instruction is sign-extended to 32 bits.

The ALU then adds the sign-extended offset to the base address (from rs) to compute the effective memory address. This address is sent to the data memory unit, and the **MemRead** control signal is asserted to enable a read operation.

The data retrieved from memory is then placed on the data bus and forwarded to the write-back stage, where it is written into the destination register (rt) in the register file, as controlled by the **RegWrite** signal.

**F-Type (Floating-Point) Instructions**

In the F-type format, used for floating-point operations, the Program Counter (PC) supplies an address that is sent over the address bus to the instruction memory. The corresponding instruction is fetched, and the PC is incremented by 4 to point to the next sequential instruction.

The opcode field of the fetched instruction is decoded by the control unit, identifying it as an F-type operation. Unlike R-type instructions that use the general-purpose register file, the F-type instruction accesses the floating-point register file to read the values from the two source registers (Fs1and Fs2) specified in the instruction.

These floating-point operand values are then routed to the Floating-Point Unit (FPU). The specific operation to be performed (e.g., addition, subtraction, multiplication) is determined by the funct field of the instruction, which generates the FPUOp control signal.

Since F-type instructions are register-to-register operations and do not access the main data memory, the MemRead and MemWrite control signals are deasserted (0). The result computed by the FPU is passed directly to the write-back stage. Finally, the result is written into the destination floating-point register (Fd) in the floating-point register file, a process enabled by asserting the FPRegWrite control signal.

### **J-Type (Jump/Branch) Instructions**

For J-type instructions, which handle control flow changes like jumps and branches, the Program Counter (PC) provides the address for fetching the instruction from memory. Simultaneously, the PC is incremented by 4 (PC + 4).

The instruction is then decoded. For a conditional branch (e.g., BR\_EQ), the control unit uses the condition reg field to select a specific general-purpose register from the register file. The value of this register is compared against the condition code in the Status Register to determine if the branch should be taken.

Concurrently, the offset field from the instruction is sign-extended to 32 bits and shifted left by 2 (to convert the word offset into a byte offset). This adjusted offset is then added to the PC +4 value to calculate the branch target address.

A key control signal for J-type instructions is Branch, which is asserted when the instruction is a branch. If the branch condition is true (e.g., the register is zero for BR\_EQ), the multiplexer controlling the next PC value selects the calculated branch target address instead of the default PC +4. This new address is loaded into the PC at the end of the cycle, redirecting the program flow. For unconditional jumps (BR\_JMP), the target address is calculated and loaded into the PC without a condition check. Since J-type instructions only alter the control flow, they do not write back to any register, and thus the RegWrite signal remains deasserted (0).