Datapath and control including the jump instruction.

|  |  |  |
| --- | --- | --- |
| Field | 00010 | adress |
| Bit postions | 31:26 | 0:25 |

The jump instruction, looks somewhat like a branch Instruction. but computes the target PC differently and is not conditional.

Like branch, the low-order 2 bits of a jump address are always 00two.

The next lower 26 bits of this 32‑bit address come from the 26‑bit immediate field in the instruction.

The upper 4 bits of the address that should replace the PC come from the PC of the jump instruction plus 4. Thus, we can implement a jump by storing into the PC the concatenation of:

The upper 4 bits of the current PC 4 (these are bits 31:28 of the sequentially following instruction address).

The 26‑bit immediate field of the jump instruction.

The bits 00two.

An additional multiplexor is used to select the source for the new PC value, which is either the incremented PC (PC 4), the branch target PC, or the jump target PC. One additional control signal is needed for the additional multiplexor.

This control signal, called *Jump*, is asserted only when the instruction is a jump—that is, when the opcode is 2.

The destination address for a jump instruction is formed by concatenating the upper 4 bits of the current PC 4 to the 26‑bit address field in the jump instruction and adding 00 as the 2 low-order bits.

The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC 4 as the high-order bits, thus yielding a

32-bit address.