### 4.4 A Simple Implementation Scheme

#### Implementing Jumps

Figure 4.17 shows the implementation of many of the instructions we looked at in Chapter 2. One class of instructions missing is that of the jump instruction. Let's extend the datapath and control of Figure 4.17 to include the jump instruction. We'll describe how to set any new control lines.

The jump instruction, shown in Figure 4.23, looks somewhat like a branch instruction but computes the target PC differently and is not conditional. Like a branch, the low-order 2 bits of a jump address are always 00\textsubscript{two}. The next 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 00\textsubscript{two}

Figure 4.24 shows the addition of the control for jump added to Figure 4.17. An additional multiplexer 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 multiplexer. This control signal, called Jump, is asserted only when the instruction is a jump—that is, when the opcode is 2.

![Figure 4.24](image)

**Figure 4.24** The simple control and datapath are extended to handle the jump instruction. An additional multiplexer (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 multiplexer 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.

The clock cycle is determined by the longest possible path in the processor. This path is almost certainly a load instruction, which uses five functional units in series: the instruction memory, the register file, the ALU, the data memory, and the register file. Although the CPI is 1 (see Chapter 1), the overall performance of a single-cycle implementation is likely to be poor, since the clock cycle is too long.

---

**4.23 Instruction format for the jump instruction (opcode = 2).** The destination address of the jump instruction is formed by concatenating the upper 4 bits of the current PC + 4 to the 26-bit address of the jump instruction and adding 00 as the 2 low-order bits.