2. 用转发来避免数据冒险

PIPE -的设计是在译码阶段从寄存器文件中读入源操作数，但是对这些源寄存器的写有可能要在写回阶段才能进行。与其暂停直到写完成，不如简单地将要写的值传到流水线寄存器 E 作为源操作数。图 4-49 用 prog2 周期 6 的流水线图的扩展描述来说明了这一策略。译码阶段逻辑发现，寄存器%rax 是操作数 valB 的源寄存器，而在写端口 E 上还有一个对%rax 的未进行的写。它只要简单地将提供到端口 E 的数据字(信号 W_valE)作为操作数 valB 的值，就能避免暂停。这种将结果值直接从一个流水线阶段传到较早阶段的技术称为*数据转发*(data forwarding，或简称*转发*，有时称为*旁路*(bypassing))。它使得 prog2 的指令能通过流水线而不需要任何暂停。数据转发需要在基本的硬件结构中增加一些额外的数据连接和控制逻辑。

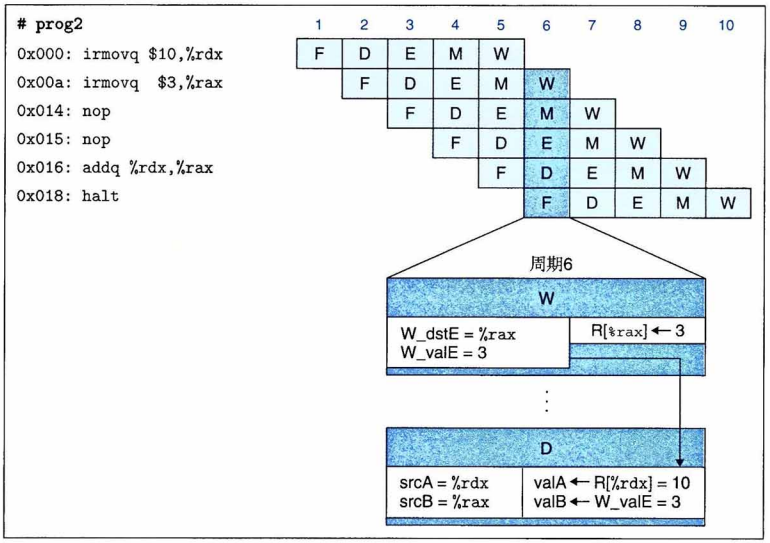

图 4-49　prog2 使用转发的流水线化的执行。在周期 6 中，译码阶段逻辑发现有在写回阶段中对寄存器%rax 未进行的写。它用这个值，而不是从寄存器文件中读出的值，作为源操作数 valB

如图 4-50 所示，当访存阶段中有对寄存器未进行的写时，也可以使用数据转发，以避免程序 prog3 中的暂停。在周期 5 中，译码阶段逻辑发现，在写回阶段中端口 E 上有对寄存器%rdx 未进行的写，以及在访存阶段中有会在端口 E 上对寄存器%rax 未进行的写。它不会暂停直到这些写真正发生，而是用写回阶段中的值(信号 W_valE)作为操作数 valA，用访存阶段中的值(信号 M_valE)作为操作数 valB。

为了充分利用数据转发技术，我们还可以将新计算出来的值从执行阶段传到译码阶段，以避免程序 prog4 所需要的暂停，如图 4-51 所示。在周期 4 中，译码阶段逻辑发现在访存阶段中有对寄存器%rdx 未进行的写，而且执行阶段中 ALU 正在计算的值稍后也会写入寄存器%rax。它可以将访存阶段中的值(信号 M_valE)作为操作数 valA，也可以将 ALU 的输出(信号 e_valE)作为操作数 valB。注意，使用 ALU 的输出不会造成任何时序问题。译码阶段只要在时钟周期结束之前产生信号 valA 和 valB，这样在时钟上升开始下一个周期时，流水线寄存器 E 就能装载来自译码阶段的值了。而在此之前 ALU 的输出已经是合法的了。