如图 4-54 所示，我们可以将暂停和转发结合起来，避免加载/使用数据冒险。这个需要修改控制逻辑，但是可以使用现有的旁路路径。当 mrmovq 指令通过执行阶段时，流水线控制逻辑发现译码阶段中的指令(addq)需要从内存中读出的结果。它会将译码阶段中的指令暂停一个周期，导致执行阶段中插入一个气泡。如周期 8 的扩展说明所示，从内存中读出的值可以从访存阶段转发到译码阶段中的 addq 指令。寄存器%rbx 的值也可以从访存阶段转发到译码阶段。就像流水线图，从周期 7 中标号为“D”的方框到周期 8 中标号为“E”的方框的箭头表明的那样，插入的气泡代替了正常情况下本来应该继续通过流水线的 addq 指令。

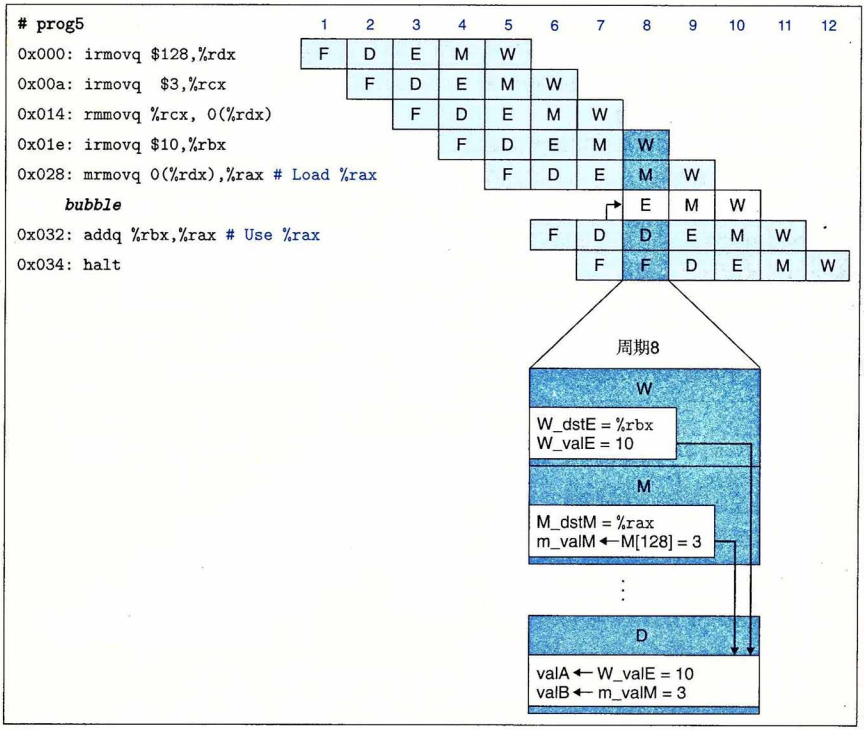

图 4-54　用暂停来处理加载/使用冒险。通过将 addq 指令在译码阶段暂停一个周期，就可以将 valB 的值从访存阶段中的 mrmovq 指令转发到译码阶段中的 addq 指令

这种用暂停来处理加载/使用冒险的方法称为*加载互锁*(load interlock)。加载互锁和转发技术结合起来足以处理所有可能类型的数据冒险。因为只有加载互锁会降低流水线的吞吐量，我们几乎可以实现每个时钟周期发射一条新指令的吞吐量目标。

4. 避免控制冒险

当处理器无法根据处于取指阶段的当前指令来确定下一条指令的地址时，就会出现控制冒险。如同在 4.5.4 节讨论过的，在我们的流水线化处理器中，控制冒险只会发生在 ret 指令和跳转指令。而且，后一种情况只有在条件跳转方向预测错误时才会造成麻烦。在本小节中，我们概括介绍如何来处理这些冒险。作为对流水线控制更一般性讨论的一部