```
3      irmovq  $3,%rcx  # I3
4      irmovq  $4,%rdx  # I4
5      halt             # I5
```

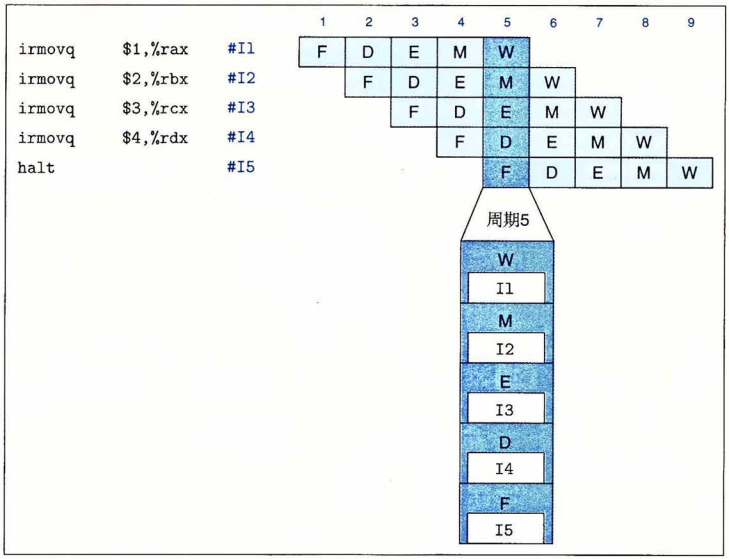

图 4-42　指令流通过流水线的示例

图中右边给出了这个指令序列的流水线图。同 4.4 节中简单流水线化的计算单元的流水线图一样，这个图描述了每条指令通过流水线各个阶段的行进过程，时间从左往右增大。上面一条数字表明各个阶段发生的时钟周期。例如，在周期 1 取出指令 I1，然后它开始通过流水线各个阶段，到周期 5 结束后，其结果写入寄存器文件。在周期 2 取出指令 I2，到周期 6 结束后，其结果写回，以此类推。在最下面，我们给出了当周期为 5 时的流水线的扩展图。此时，每个流水线阶段中各有一条指令。

从图 4-42 中还可以判断我们画处理器的习惯是合理的，这样，指令是自底向上的流动的。周期 5 时的扩展图表明的流水线阶段，取指阶段在底部，写回阶段在最上面，同流水线硬件图(图 4-41)表明的一样。如果看看流水线各个阶段中指令的顺序，就会发现它们出现的顺序与在程序中列出的顺序一样。因为正常的程序是从上到下列出的，我们保留这种顺序，让流水线从下到上进行。在使用本书附带的模拟器时，这个习惯会特别有用。

4.5.3　对信号进行重新排列和标号

顺序实现 SEQ 和 SEQ+在一个时刻只处理一条指令，因此诸如 valC、srcA 和 valE 这样的信号值有唯一的值。在流水线化的设计中，与各个指令相关联的这些值有多个版本，会随着指令一起流过系统。例如，在 PIPE－的详细结构中，有 4 个标号为“Stat”的白色方框，保存着 4 条不同指令的状态码(参见图 4-41)。我们需要很小心以确保使用的是正确版本的信号，否则会有很严重的错误，例如将一条指令计算出的结果存放到了另一条指令指定的目的寄存器。我们采用的命名机制，通过在信号名前面加上大写的流水线寄存