

存阶段才能发现非法数据地址。

 **练习题 4.30** 写出信号 `f_stat` 的 HCL 代码，提供取出的指令的临时状态。

## 2. 译码和写回阶段

图 4-58 是 PIPE 的译码和写回逻辑的详细说明。标号为“dstE”、“dstM”、“srcA”和“srcB”的块非常类似于它们在 SEQ 的实现中的相应部件。我们观察到，提供给写端口的寄存器 ID 来自于写回阶段(信号 `W_dstE` 和 `W_dstM`)，而不是来自于译码阶段。这是因为我们希望进行写的目的寄存器是由写回阶段中的指令指定的。

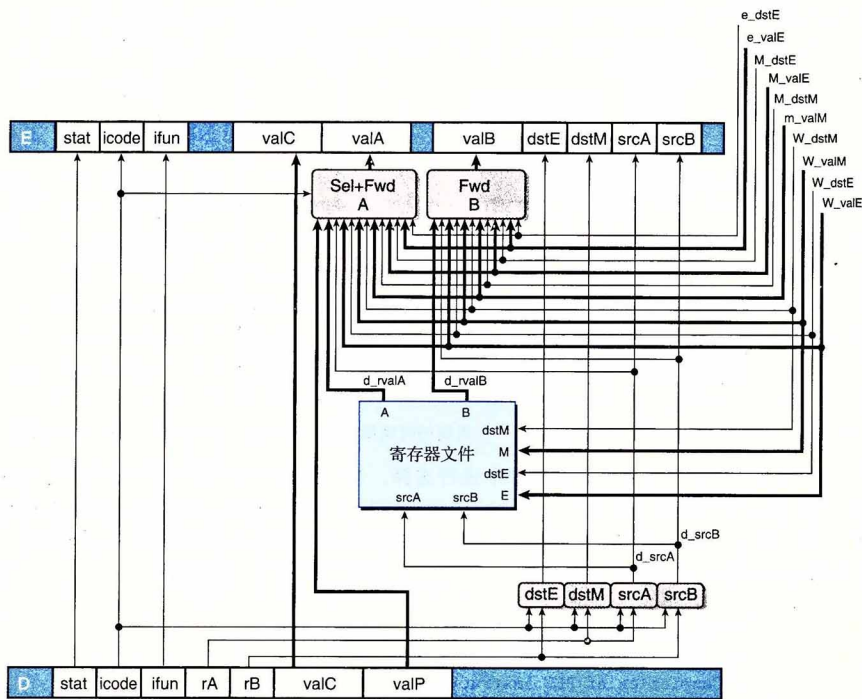



图 4-58 PIPE 的译码和写回阶段逻辑。没有指令既需要 `valP` 又需要来自寄存器端口 A 中读出的值，因此对后面的阶段来说，这两者可以合并为信号 `valA`。标号为“Sel+Fwd A”的块执行该任务，并实现源操作数 `valA` 的转发逻辑。标号为“Fwd B”的块实现源操作数 `valB` 的转发逻辑。寄存器写的位置是由来自写回阶段的 `dstE` 和 `dstM` 信号指定的，而不是来自于译码阶段，因为它要写的是当前正在写回阶段中的指令的结果

 **练习题 4.31** 译码阶段中标号为“dstE”的块根据来自流水线寄存器 D 中取出的指令的各个字段，产生寄存器文件 E 端口的寄存器 ID。在 PIPE 的 HCL 描述中，得到的信号命名为 `d_dstE`。根据 SEQ 信号 `dstE` 的 HCL 描述，写出这个信号的 HCL 代码。(参考 4.3.4 节中的译码阶段。)目前还不关心实现条件传送的逻辑。

这个阶段的复杂性主要是跟转发逻辑相关。就像前面提到的那样，标号为“Sel+Fwd A”的块扮演两个角色。它为后面的阶段将 `valP` 信号合并到 `valA` 信号，这样可以减少流水线寄存器中状态的数量。它还实现了源操作数 `valA` 的转发逻辑。