器名字作为前缀，存储在流水线寄存器中的信号可以唯一地被标识。例如，4 个状态码可以被命名为 D_stat、E_stat、M_stat 和 W_stat。我们还需要引用某些在一个阶段内刚刚计算出来的信号。它们的命名是在信号名前面加上小写的阶段名的第一个字母作为前缀。以状态码为例，可以看到在取指和访存阶段中标号为“Stat”的控制逻辑块。因而，这些块的输出被命名为 f_stat 和 m_stat。我们还可以看到整个处理器的实际状态 Stat 是根据流水线寄存器 W 中的状态值，由写回阶段中的块计算出来的。

旁注　信号 M _ stat 和 m _ stat 的差别

在命名系统中，大写的前缀“D”、“E”、“M”和“W”指的是流水线寄存器，所以 M _ stat 指的是流水线寄存器 M 的状态码字段。小写的前缀“f”、“d”、“e”、“m”和“w”指的是流水线阶段，所以 m _ stat 指的是在访存阶段中由控制逻辑块产生出的状态信号。

理解这个命名规则对理解我们的流水线化处理器的操作是至关重要的。

SEQ+和 PIPE−的译码阶段都产生信号 dstE 和 dstM，它们指明值 valE 和 valM 的目的寄存器。在 SEQ+中，我们可以将这些信号直接连到寄存器文件写端口的地址输入。在 PIPE−中，会在流水线中一直携带这些信号穿过执行和访存阶段，直到写回阶段才送到寄存器文件(如各个阶段的详细描述所示)。我们这样做是为了确保写端口的地址和数据输入是来自同一条指令。否则，会将处于写回阶段的指令的值写入，而寄存器 ID 却来自于处于译码阶段的指令。作为一条通用原则，我们要保存处于一个流水线阶段中的指令的所有信息。

PIPE−中有一个块在相同表示形式的 SEQ+中是没有的，那就是译码阶段中标号为“Select A”的块。我们可以看出，这个块会从来自流水线寄存器 D 的 valP 或从寄存器文件 A 端口中读出的值中选择一个，作为流水线寄存器 E 的值 valA。包括这个块是为了减少要携带给流水线寄存器 E 和 M 的状态数量。在所有的指令中，只有 call 在访存阶段需要 valP 的值。只有跳转指令在执行阶段(当不需要进行跳转时)需要 valP 的值。而这些指令又都不需要从寄存器文件中读出的值。因此我们合并这两个信号，将它们作为信号 valA 携带穿过流水线，从而可以减少流水线寄存器的状态数量。这样做就消除了 SEQ(图 4-23)和 SEQ+(图 4-40)中标号为“Data”的块，这个块完成的是类似的功能。在硬件设计中，像这样仔细确认信号是如何使用的，然后通过合并信号来减少寄存器状态和线路的数量，是很常见的。

如图 4-41 所示，我们的流水线寄存器包括一个状态码 stat 字段，开始时是在取指阶段计算出来的，在访存阶段有可能会被修改。在讲完正常指令执行的实现之后，我们会在 4.5.6 节中讨论如何实现异常事件的处理。到目前为止我们可以说，最系统的方法就是让与每条指令关联的状态码与指令一起通过流水线，就像图中表明的那样。

4.5.4　预测下一个 PC

在 PIPE−设计中，我们采取了一些措施来正确处理控制相关。流水线化设计的目的就是每个时钟周期都发射一条新指令，也就是说每个时钟周期都有一条新指令进入执行阶段并最终完成。要是达到这个目的也就意味着吞吐量是每个时钟周期一条指令。要做到这一点，我们必须在取出当前指令之后，马上确定下一条指令的位置。不幸的是，如果取出的指令是条件分支指令，要到几个周期后，也就是指令通过执行阶段之后，我们才能知道是否要选择分支。类似地，如果取出的指令是 ret，要到指令通过访存阶段，才能确定返回地址。

除了条件转移指令和 ret 以外，根据取指阶段中计算出的信息，我们能够确定下一条