旁注　列举数据冒险的类型

当一条指令更新后面指令会读到的那些程序状态时，就有可能出现冒险。对于Y86-64来说，程序状态包括程序寄存器、程序计数器、内存、条件码寄存器和状态寄存器。让我们来看看在提出的设计中每类状态出现冒险的可能性。

程序寄存器：我们已经认识这种冒险了。出现这种冒险是因为寄存器文件的读写是在不同的阶段进行的，导致不同指令之间可能出现不希望的相互作用。

程序计数器：更新和读取程序计数器之间的冲突导致了控制冒险。当我们的取指阶段逻辑在取下一条指令之前，正确预测了程序计数器的新值时，就不会产生冒险。预测错误的分支和ret指令需要特殊的处理，会在4.5.5节中讨论。

内存：对数据内存的读和写都发生在访存阶段。在一条读内存的指令到达这个阶段之前，前面所有要写内存的指令都已经完成这个阶段了。另外，在访存阶段中写数据的指令和在取指阶段中读指令之间也有冲突，因为指令和数据内存访问的是同一个地址空间。只有包含自我修改代码的程序才会发生这种情况，在这样的程序中，指令写内存的一部分，过后会从中取出指令。有些系统有复杂的机制来检测和避免这种冒险，而有些系统只是简单地强制要求程序不应该使用自我修改代码。为了简便，假设程序不能修改自身，因此我们不需要采取特殊的措施，根据在程序执行过程中对数据内存的修改来修改指令内存。

条件码寄存器：在执行阶段中，整数操作会写这些寄存器。条件传送指令会在执行阶段以及条件转移会在访存阶段读这些寄存器。在条件传送或转移到达执行阶段之前，前面所有的整数操作都已经完成这个阶段了。所以不会发生冒险。

状态寄存器：指令流经流水线的时候，会影响程序状态。我们采用流水线中的每条指令都与一个状态码相关联的机制，使得当异常发生时，处理器能够有条理地停止，就像在4.5.6节中会讲到的那样。

这些分析表明我们只需要处理寄存器数据冒险、控制冒险，以及确保能够正确处理异常。当设计一个复杂系统时，这样的分类分析是很重要的。这样做可以确认出系统实现中可能的困难，还可以指导生成用于检查系统正确性的测试程序。

1. 用暂停来避免数据冒险

暂停(stalling)是避免冒险的一种常用技术，暂停时，处理器会停止流水线中一条或多条指令，直到冒险条件不再满足。让一条指令停顿在译码阶段，直到产生它的源操作数的指令通过了写回阶段，这样我们的处理器就能避免数据冒险。这种机制的细节会在4.5.8节中讨论。它对流水线控制逻辑做了一些简单的加强。图4-47(prog2)和图4-48(prog4)中画出了暂停的效果。(在这里的讨论中我们省略了prog3，因为它的运行类似于其他两个例子。)当指令addq处于译码阶段时，流水线控制逻辑发现执行、访存或写回阶段中至少有一条指令会更新寄存器%rdx或%rax。处理器不会让addq指令带着不正确的结果通过这个阶段，而是会暂停指令，将它阻塞在译码阶段，时间为一个周期(对prog2来说)或者三个周期(对prog4来说)。对所有这三个程序来说，addq指令最终都会在周期7中得到两个源操作数的正确值，然后继续沿着流水线进行下去。

将addq指令阻塞在译码阶段时，我们还必须将紧跟其后的halt指令阻塞在取指阶段。通过将程序计数器保持不变就能做到这一点，这样一来，会不断地对halt指令进行取指，直到暂停结束。