思考题

1.乘法和除法操作通常比加法和减法要复杂得多，且需要更多的时钟周期。将乘除法部件从 ALU 中分离出来，能够使 ALU 专注于简单的加法、减法、位移等运算，从而提高整体的运算速度。

2.真实的流水线 CPU 中，乘法器通常是一个独立的部件，并且是流水线化的。乘法器根据操作数的位数，通过多个时钟周期完成乘法运算。 在一些现代 CPU 中，乘法可能通过使用并行乘法器、流水线化的 Booth 算法或乘法查找表（LUT）来加速。 流水线乘法器通常需要处理延迟，因此需要与其他流水线阶段（如 ALU）协同工作，确保乘法结果能正确与其他操作（如加法、存储）结合。

3.Busy 信号：通常在 CPU 中，busy 信号用来表示某个操作是否还在进行，特别是在多周期的操作（如乘法、除法或内存访问）中。当 busy 信号为高电平时，CPU 知道操作尚未完成，需要暂停其他操作并等待。

周期阻塞：在实现中，我们可以在执行周期中检查 busy 信号，如果该信号为高电平，则暂停当前指令的执行，直到 busy 信号变为低电平。这样可以避免在硬件资源未准备好时强行执行下一个指令，从而避免数据冲突或不正确的结果。

处理方法：

在 busy 信号有效时，指令流会被暂停，直到该信号解除，保证后续操作能在正确的时序下进行。 这种方法确保了多周期操作（如乘法、除法）不会与其他操作冲突，避免了数据冲突或竞争条件。

4.使用字节使能信号能够明确地控制每个字节的写入，避免不必要的写操作。这使得写指令的行为更加清晰，能精确指定哪些字节被写入存储器。

5.在按字节读时，虽然每次读取的是一个字节，但内存总线通常是按更宽的宽度（如 32 位或 64 位）来读取数据的。通过字节使能信号，可以选择性地读取其中的某一个字节。 在按字节写时，写操作也通常是按字节处理，通过字节使能信号控制哪些字节被写入。

6.在实现时，我们可能会采用模块化设计，把每个功能单元（如 ALU、乘法器、除法器、寄存器堆等）抽象为独立的模块，每个模块负责独立的操作。这使得复杂性得到管理，便于调试和维护。 数据冲突）可以通过流水线前递或暂停来解决。通过在流水线中引入控制信号，检测并处理数据冒险，保证指令流的正确性。

7.常见的冲突包括数据冒险（如寄存器读取前的写入）、控制冒险（如分支指令导致的流水线冲突）等。 解决方法： 使用前递来避免数据冒险，确保数据在需要时可以直接从执行阶段传递到下游指令。 使用暂停来处理无法通过前递解决的冲突，强制暂停流水线，直到所需数据准备好。