【理解ILP记分牌算法】

[关于流水线的三种冒险 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/447682231)

[ILP——指令级并行2：记分牌（Scoreboard）技术 (exyb.cn)](https://www.exyb.cn/news/show-3992303.html?action=onClick)

[计算机系统结构：指令的动态调度-记分牌算法\_kikato2022的博客-CSDN博客](https://blog.csdn.net/weixin_40064300/article/details/124443945)

【回答问题思路】

对于super-scalar out-of-order processor：

A deeper pipeline means that a mispredicted branch causes a longer pipeline flush, leading to a greater number of wasted cycles for every mispredict.（**speculative execution**）unnecessary execution of instructions, wasting power and computational resources

Super-scalar processors can dispatch and execute multiple instructions per cycle. A branch misprediction in such an environment could waste more resources as multiple pipelines or functional units could be left idle.（instruction throughput）

reorder buffer (***ROB***) to keep track of the out-of-order instructions. Mispredicted branches can fill up the ROB with instructions that will later have to be discarded, thus preventing other legitimate instructions from entering the pipeline.

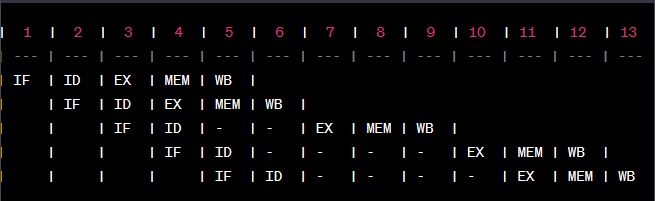
The instruction window or **reservation stations** hold the instructions waiting for their operands to be ready. Mispredicted branches can fill up this window with instructions that will not retire, limiting the processor's ability to find other parallelism.

A branch misprediction might cause unnecessary cache fills, polluting the cache with data and instructions that won't be used.

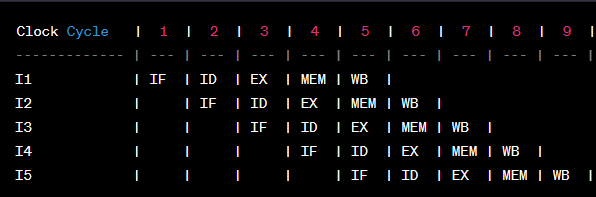
As we scale the size of out-of-order engines (i.e., wider issue width, larger instruction windows, etc.)...

【五阶段流水线分析】

Case 1: without bypassing



Case 2: with bypassing



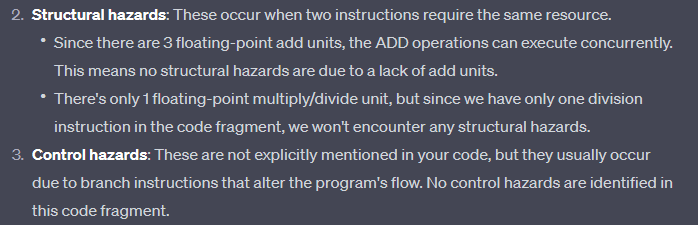
Obviously, stalls have been effectively prevented by applying the bypassing or forwarding techniques.

【概念区分】

Data **dependence**: true data dependence, anti-dependence, output dependence

Data **hazards** including: RAW, WAR, WAW

Other types of hazards:



**Score-boarding** is an out-of-order execution technique that monitors and resolves hazards, allowing the CPU to execute multiple instructions concurrently, even if they are not adjacent in the program order.



**Tomasulo’s** algorithm, through the use of reservation stations, register renaming, and the common data bus, effectively handles the data hazards present in the code fragment.

Tomasulo's algorithm uses register renaming to resolve WAR and WAW hazards. As for RAW hazard:

