计算机系统结构

第二讲 指令的动态调度

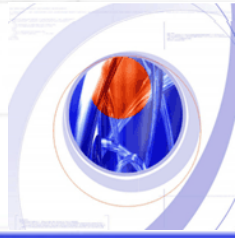
谢长生

武汉光电国家研究中心

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

1.指令调度的分类

- 静态调度
  - 依靠编译器对代码进行静态调度，以减少相关和冲突。
  - 它不是在程序执行的过程中、而是在编译期间进行代码调度和优化。
  - 通过把相关的指令拉开距离来减少可能产生的停顿。
- 动态调度
  - 在程序的执行过程中，依靠专门硬件对代码进行调度，减少数据相关导致的停顿。
  - 优点：
    - 能够处理一些在编译时情况不明的相关（比如涉及到存储器访问的相关），并简化了编译器；
    - 能够使本来是面向某一流水线优化编译的代码在其它的流水线（动态调度）上也能高效地执行。
    - 以硬件复杂性的显著增加为代价



2.经典（顺序）流水线的局限性

- 指令是按序流出和按序执行的
- 考虑下面一段代码：

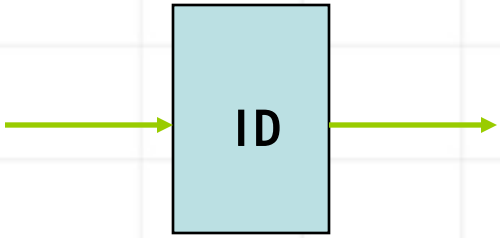
```
DIV.D F4，F0，F2
ADD.D F10，F4，F6
SUB.D F12，F6，F14
```

ADD.D指令与DIV.D指令关于F4相关，导致流水线停顿。

SUB.D指令与流水线中的任何指令都没有关系，但也因此受阻。

3. SUB指令需要等待的原因

在前面的基本流水线中：



检测结构冲突

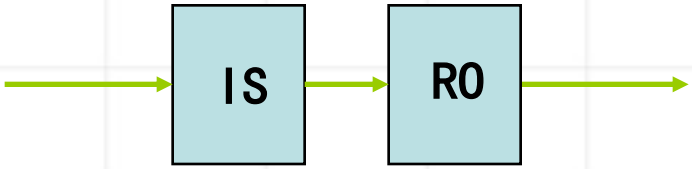
检测数据冲突

一旦一条指令受阻，其后的指令都将停顿。

解决办法：允许乱序执行

4. 乱序执行

为了支持乱序执行，我们将5段流水线的译码阶段再分为两个阶段：



流出　Issue，IS
　　in-order issue
读操作数　Read Operands，RO
　　out of order execution

5-1. WAR与WAW冲突

在前述5段流水线中，是不会发生WAR冲突和WAW冲突的。但乱序执行就使得它们可能发生了。

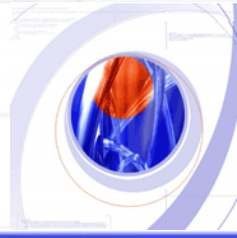
- 例如，考虑下面的代码

```
DIV.D   F10, F0, F2
ADD.D   F10, F4, F6
SUB.D   F6, F8, F14
```

存在输出相关：DIV.D 与 ADD.D

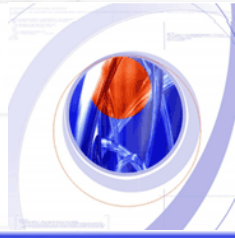
存在反相关：ADD.D 与 SUB.D

可以通过使用寄存器重命名来消除。



5-2. 多条指令同时处于执行或访存中

- 动态调度的流水线支持多条指令同时处于执行当中。
- 要求
  - 具有多个功能部件
  - 或者功能部件流水化
  - 或者兼而有之。
- 我们假设具有多个功能部件。

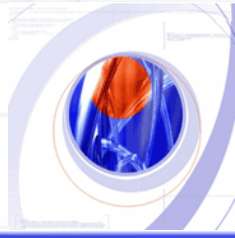


5-3. 复杂的异常处理

- 精确异常
  - 如果发生异常时，处理机的现场跟严格按程序顺序执行时指令i的现场相同。
- 不精确异常
  - 当执行指令i导致发生异常时，处理机的现场（状态）与严格按程序顺序执行时指令i的现场不同。
  - 发生不精确异常的原因：因为当发生异常（设为指令i）时
    - 流水线可能已经执行完按程序顺序是位于指令i之后的指令；
    - 流水线可能还没完成按程序顺序是指令i之前的指令。
  - 不精确异常使得在异常处理后难以接着继续执行程序。

5-4. 复杂的异常处理2

- 动态调度的处理机要保持正确的异常行为
  - 对于一条会产生异常的指令来说，只有当处理机确切地知道该指令将被执行时，才允许它产生异常。
- 即使保持了正确的异常行为，动态调度处理机仍可能发生不精确异常。



谢谢大家