体系结构知识

1.寄存器重命名

1.1 假冒险

```
; RAW 是真相关
ADD R3, R1, R2
AND R5, R3, R4
; WAW 重命名第二、三个R3寄存器为R10 或者说 重命名冒险中的目的寄存器
ADD R3, R1, R2
NOP
AND R3, R6, R7
ADD R9, R3, R8
; WAR 重命名第二个R3寄存器为R10 或者说 重命名冒险中的目的寄存器
ADD R1, R3, R2
ADD R3, R5, R4
```

- **寄存器重命名的目的是消除指令间写后写、读后写这两种假数据冒险**，在现代处理器中，“寄存器重命名”往往因为其实现的复杂性而被拿出来单独用一个周期（甚至两个周期）来完成。
- **在顺序发射、乱序执行的处理器里，就有可能发生上述两种假冒险——后序指令可能先写回结果，从而导致前序指令读到错误数据，或是前序指令更晚写回，从而导致最终结果出错。**
- 如果解决方案是**让发生冒险的指令顺序执行**，会浪费处理器的性能。但WAR和WAW的两条指令间没有真正的数据相关，没有依赖就不影响指令的并行执行。
- 出现这种冒险的本质原因是**处理器的逻辑寄存器数量不足。**

- 在重命名的过程中，**处理器首先根据指令的源寄存器编号查询“重命名映射表”。**
  - 重命名映射表记录了逻辑寄存器和物理寄存器的对应关系（物理寄存器是一些逻辑寄存器之外的、指令集规定之外的寄存器，大的寄存堆）
- 通过映射结果，指令可以找到源寄存器对应的物理寄存器，从而得到需要的数据。
- **接下来，指令从空闲寄存器列表中找到空闲的物理寄存器，用来和指令的目的寄存器建立映射关系。**
- 经过这两步操作，指令之间的假相关就被消除了。

1.2 三种方法实现重命名

- 使用重排序缓冲（Reorder Buffer，ROB）来实现寄存器重命名；
- 将逻辑寄存器（Architecture Register File，ARF）扩展来实现寄存器重命名；
- 使用统一的物理寄存器（Physical Register File，PRF）来实现寄存器重命名

CPU信息

- L1/L2/L3的cacheline都是64B

> 三老婆

- L1_I-Cache是32KB、L1_D-Cache是48KB
  - 属于Core，latency大概3~4个cycle
- L2_Cache是1.25M
  - 属于Core，latency大概10~20个cycle
- L3_Cache是12M
  - 多个Core共享，latency大概40~45个cycle
- 内存的latency大概是120~240个cycle

CPU并行方法

- SIMD：向量寄存器
- 多线程
- 循环展开
- 消息传递多机并行