Superscalar Organization

Instruction-Level Parallelism

Iron Law: time of program = # of instructions $\times$ CPI $\times$ clock cycle time

一般的标量处理器每个周期最多 fetch 并处理 1 条指令，CPI $\geqslant 1$

Superscalar processor：利用 ILP，每个周期 fetch 并处理执行多条指令，通常 CPI < 1，一般用 IPC 来做指标，IPC 是 CPI 的倒数

> ILP 用于测量指令间的依赖程度，一般可以认为 ILP 为理想机器上的指令数与周期数的比，假定
>
> - 资源无限
> - 最优 fetch
> - 所有指令延迟相同
>
> ILP 不是 IPC，ILP 是理想化的情况，是 IPC 的上界

一般现实中的程序 ILP 大约在 0.9-1.9

Amdahl's Law：设

- f 为程序能并行的部分
- N 为处理器数

则加速比为

$$Speedup = \frac{1}{1 - f + \frac{f}{N}}$$

加速比的上限由程序串行的部分决定，**串行部分是瓶颈**

Superscalar

Instruction Fetch

每个周期 fetch 多条指令，fetch 速率决定了 IPC，IPC 不可能高于 fetch 的速率。考虑到 branch 预测错误的情况，一般 fetch 要比执行的速率高一点

需要考虑两方面的问题

- 对齐：instruction cache 的 cache line 的跨行读取

- branch prediction

Dependency Resolving

数据依赖包括

- RAW
- WAR, WAW

保证超标量处理器乱序执行不出错

控制依赖

- 静态调度，如 loop unrolling
- 预测执行
- BR 和 speculation

> loop unrolling 即将循环体重复，从而减少循环次数
>
> - 优点
>   - 减少循环变量的计算次数
>   - 增大基本块，利于优化
> - 缺点
>   - 如果循环次数不是展开常数的倍数，需要额外代码处理
>   - 增大代码体积
>   - 可能增加寄存器的需求

Function unit

除了控制依赖和数据依赖，还有结构冒险需要处理，超标量处理器需要提供多条流水线以支持多条指令的同时运行

一般不会把多条流水线设计成相同的样子

- 利用率低，每条流水线都需要同样的 ALU，浮点单元，内存读取等
- 指令延时增加：不同的功能需要的 stage 数不同

大多数情况超标量处理器有多条不同的流水线，这样有可能产生结构冒险，即多条指令需要同一 function unit

Completion

指令以乱序执行完成，需要按照原本的顺序 commit，这样才能让程序员看起来是按序执行的

speculation：当有几十上百条指令运行中，如果 BP 预测错误，如何冲刷受影响的指令

精确中断：中断发生时如何维护按序

Overview

总的来说超标量需要解决的问题可以总结为

- fetch
    - fetch 多条指令
    - branch prediction
    - 对齐问题
- decode
    - 解码
    - 确定指令依赖
- execution
    - 指令的分发
    - 依赖的处理
    - bypass/forwarding
    - 多路访存
- completion
    - 乱序完成
    - speculative instruction
    - 精确中断