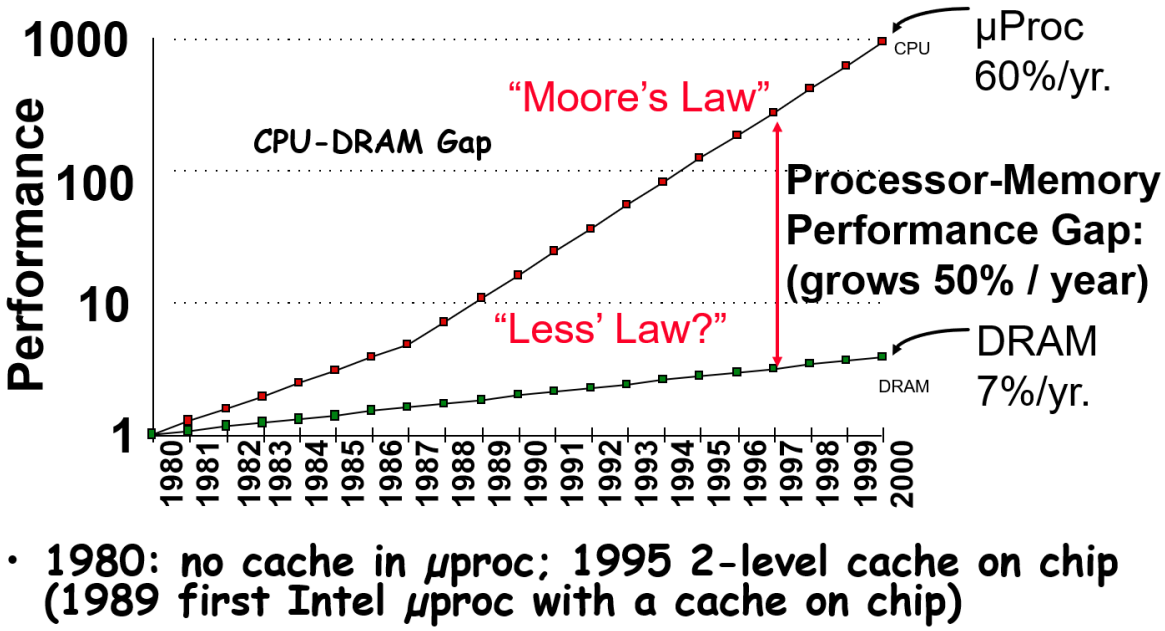
Cache I

Cache 存在的必要性

DRAM（动态随机存取记忆体）的性能（与性能提升的速度）远小于微处理器。

内存与 CPU 间巨大的性能差距使得 cache 节约时间的作用非常显著。



（其中的 "Less' Law" 是对 Moore's Law 的仿词）

什么是 cache

小而快的存储结构，用于加快对内存的平均访问时间。

在很多结构中都应用了跟 cache 一样的 locality 思想，如：

- 寄存器可以认为是变量的 cache
- 一级 cache 是二级 cache 的 cache
- 二级 cache 是内存的 cache
- 内存是外存的 cache（联想 BPT 的缓存）

（Cache 是一个法语词，原意为 mask 或 hiding place。）

公式

$$\text{CPUTime} = IC \times (\frac{\text{ALUOps}}{\text{Inst}} \cdot CPI_{\text{ALUOps}} + \frac{\text{MemAccess}}{\text{Inst}} \times \text{AMAT}) \times \text{CycleTime}$$

$$\text{AMAT}_{\$} = T_{\text{hit}} + \eta_{\text{miss rate}} \cdot T_{\text{penalty}}$$

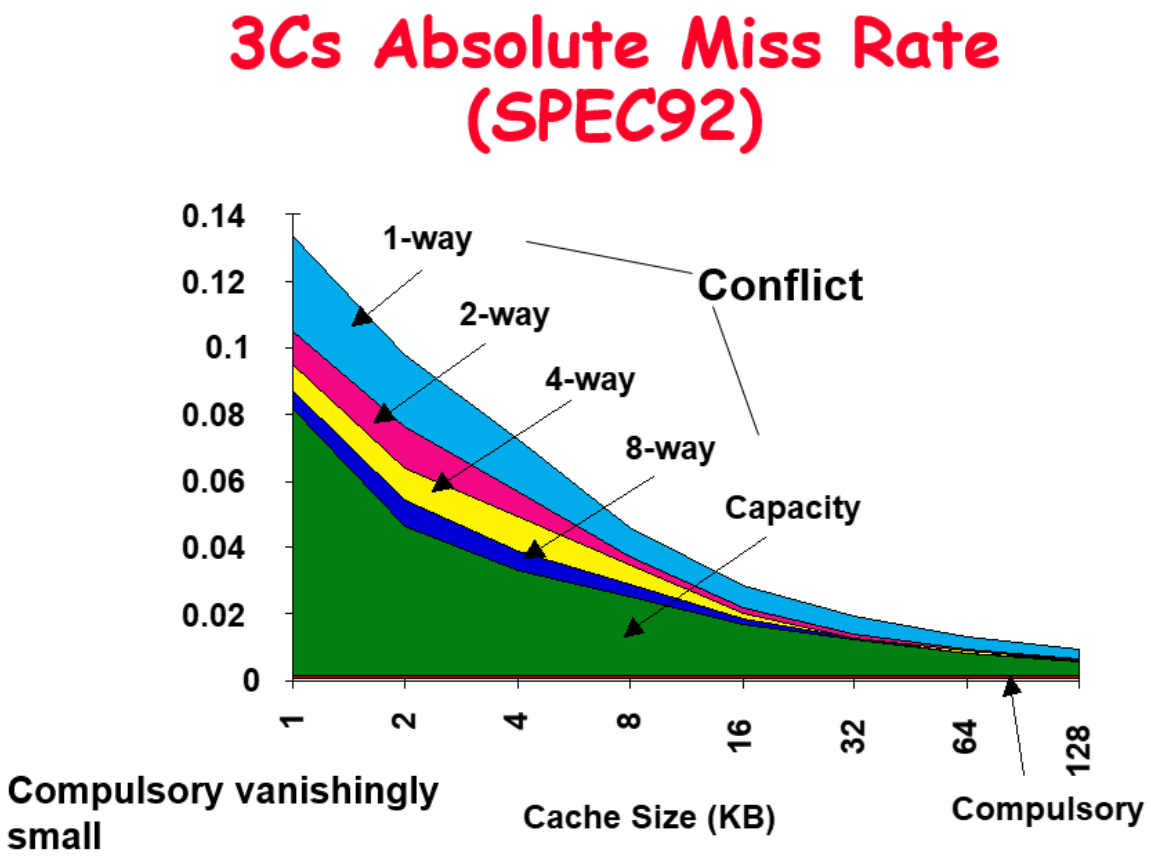
(Average Memory Access Time)

哈佛结构：指令放在指令 cache 中，data 放在 data cache 中。

减少 miss rate

对 miss 原因分类：

1. Compulsory 强制 miss：一块的第一次访问肯定不会在 cache 里面。就算 cache 大小无限也会 miss。也叫 cold start miss 或 first reference miss。
2. Capacity 容量不够：容量不足导致 miss（有的块无法进入 cache，因为容量不足被后面的块顶掉）。在全组关联 cache 中很容易理解（没有 index）。
3. Conflict 冲突：由于多个块都映射到同一个区域，导致块被顶掉。在直接映射法中很容易理解：只有 tag 不同，但 index 相同的几个块会轮番顶掉在 cache 中的块。
4. Coherence 相干性：现在多核 CPU 中每个核都有自己的 cache，如果一个核改变了内存中一处的值，它无法改动别的核的 cache。为保证数据正确性，一般采用强制将其他核的 cache invalidate（valid 位置为 0）的方式处理，从而导致 cache miss。



对这张图的观察：

- Cache size 增大时，conflict 和 capacity miss 都大幅下降。
- 提高组关联的 way 数会显著减小 conflict miss 数，但 cache size 大时这种效果在减小。
- Compulsory miss 比例非常小。

要求 cache size 小时，就多加 way。

`miss rate of 1-way associative cache size X = miss rate 2-way associative cache size X/2`