

## 9. 解

性能分析对比方法：

(1) 占用率：程序运行占用处理器时间与对应处理器运行总时间的比值，

若运行同一程序， $a$ 处理器占用率比**b**处理器低，则  $a$ 处理器对这一程序进行处理的效率更高

(2) 程序运行指令数：

MIPS (Million Instructions Per Second)：

$$\text{MIPS} = \frac{\text{指令数}}{(\text{执行时间} \times 10^6)} = \frac{\text{指令数}}{(\text{指令数} \cdot \text{CPI} / \text{freq} \cdot 10^6)}$$
$$= \text{freq} / (\text{CPI} \cdot 10^6)$$

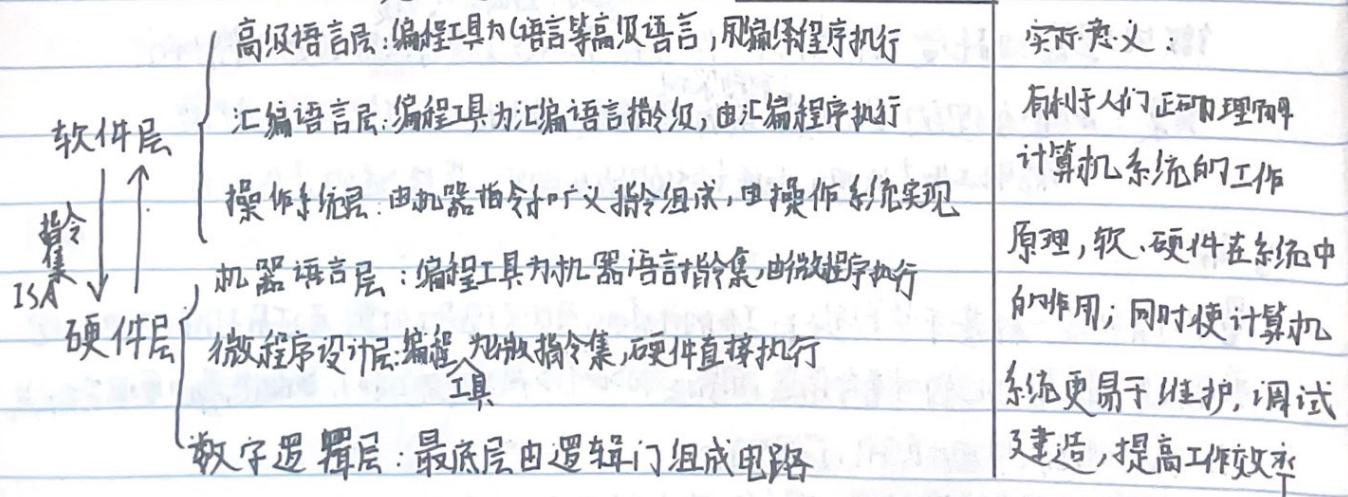
CPI：执行一条指令所需周期数

测试程序 BenchMark：

SPEC	Dhrystone: 单位每秒计算多少次 Dhrystone, 指在 VAX 11/780 上运行 1757 Dhrystones 定义为 1 DMIPS
CoreMark: 相比 Dhrystone, 算法更复杂	实际价值
EEMBC	

10. 解：计算机的设计总体分为了两个层次 硬件设计和软件设计

而指令集则是沟通软件与硬件间的桥梁



## 附加题2：

文章提出了3个核心观点，第一：软件的进步可以激励架构创新；其次：提高硬件/软件接口为架构创新提供了机会；第三：市场最终会解决选择何种架构的争论；

作为沟通硬件与软件的桥梁，ISA在过去的几十年间取得了巨大的发展。从CISC的广泛使用到RISC的诞生与逐步发展，我们总是在寻找一种最优化的指令架构提高计算机性能。但自苹果凭借iPhone开启Post-PC时代，从2010年起，ISA的代表架构x86的出货量就每年下降近10%，与此同时，带有RISC处理器的芯片的市场份额却日趋庞大。可以说CISC赢得了PC时代的后期，RISC则正在赢得Post-PC时代，而这也正是市场所决定的。值得注意的是，RISC与CISC并非水火不相容，受到流水线简单指令启发，我们可以将复杂的x86转换为内部类似RISC的微指令，将执行跨并行指令的思想整合到x86中。

但当我们执行某一特定领域的问题时，使用通用处理器很可能会导致部分资源调用的不足与浪费，为了提高某效率，我们首先可以提高高级语言的性能。如执行矩阵乘法时，通过对语言的优化，可以大幅提升效率，比原版本快62000倍以上；其次，针对该领域或定制专门的架构，如DSL（domain-specific language）就是硬件-软件接口的一种交互语言，支持对应 ISA架构，使开发者从指令集中解放，提高了处理器性能，且能更好地保障其安全性。

随着Dennard缩放定律和Moore定律的终结，以及标准微处理器性能增长的减缓，我们将面临当前巨大的挑战，但同时如果我们能够抓住，加深对计算机架构的研究，大胆预测，快速改进，迎来计算机架构的又一个黄金时代。