全美经典
学习指导系列

计算机体系结构
习题与解答

Computer Architecture

最佳的复习资料，实用的辅助教材

与国外高校计算机水平保持同步

为考研和出国深造奠定坚实基础

（美） Nicholas Carter 著
肖明 王永红 等译

本系列丛书
全球销售超过
3000万册！

机械工业出版社
China Machine Press

全美经典
学习指导系列

计算机体系结构习题与解答

Computer Architecture

（美） Nicholas Carter 著
肖明 王永红 等译

机械工业出版社
China Machine Press

本书介绍了与计算机体系结构相关的各种主题。第1章到第5章介绍了计算机体系结构课程中的许多基本概念，包括：数据表示与算术运算、计算机组织、编程模型、处理器设计。第6章和第7章分别讨论了流水线和指令级并行性方法，它们都是影响现代处理器的重要因素。第8章到第10章介绍了各种存储器系统的设计，包括存储器层次结构、高速缓冲存储器和虚拟存储器。第11章讨论了输入/输出系统。第12章介绍了多处理器系统。

通过阅读本书，读者能够迅速了解与计算机体系结构相关的各种知识，并将其应用到其他课程的学习和编程实践中。本书内容全面，每章均提供了大量的实例和习题，是学习计算机体系结构课程的一本极佳教辅材料。

本书版权登记号：图字：01-2003-4671

图书在版编目（CIP）数据

计算机体系结构习题与解答/（美）卡特（Carter, N.）著；肖明等译. －北京：机械工业出版社，2004.9

（全美经典学习指导系列）

书名原文：Computer Architecture

ISBN 7-111-14912-2

Ⅰ. 计… Ⅱ. ①卡…②肖… Ⅲ 计算机体系结构－解题 Ⅳ TP303-44

中国版本图书馆 CIP 数据核字（2004）第 074478 号

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）

责任编辑：刘立卿

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2004 年 9 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 17 印张

印数：0 001－5 000 册

定价：25.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

本书是一本讲述计算机体系结构相关知识的教辅书。

全书共分 12 章，主要包括：计算机体系结构概述、数据表示与计算机算术运算、计算机组织、编程模型、处理器设计、流水线、指令级并行性、存储器系统、高速缓冲存储器、虚拟存储器、输入/输出系统、多处理器系统等内容。每章正文部分均精选了相关的主要概念和关键技术，每章最后还提供了许多较实用的习题及习题解答，有利于读者深入理解计算机体系结构相关知识，便捷地检验学习效果。另外，本书还可以与计算机体系结构方面的任何教材配合使用，并可根据个人实际情况合理安排学习进度，有助于读者迅速牢记各章知识要点，掌握计算机体系结构的基本原理以及求解计算机体系结构问题的思路与方法，提高分析与解决相关问题的能力。

本书内容翔实，适合于想要了解计算机体系结构的各类读者阅读。对于与计算机体系结构课程相关的师生来说，本书是一本很好的教辅材料；对于从事各类实际工作的相关技术人员来说，本书是极佳的参考读物；对于准备参加计算机专业考试的各类考生来说，本书是优秀的复习资料。

本书主要由北京师范大学信息技术与信息管理系的肖明副教授翻译，王永红、续鸿飞、陈颖等参加了本书第 3、4、5、7、11 等章的翻译。全书由肖明统一审校，所有插图由王永红负责绘制和修改。

由于本书涉及的内容较多，而译者水平有限，所以书中难免会有疏漏或者错误之处，敬请广大读者不吝指教。如有任何批评意见和建议，请与译者联系：ming_xiao02@sohu.com。

肖 明
北京师范大学管理学院
2004 年 6 月 28 日

前　言

计算机体系结构中最引人注目的一个方面就是其变化频率之快。在该领域，差不多每一天都会产生新的变革，这也为每个人提供了细致研究的良机。但是，这种快速变化对于讲授计算机体系结构和组织课程的人员来说是一大挑战。与其他许多领域不同的是，计算机体系结构这门课程内容的组织，每学期都必须有所改变，以吸收该领域中的最新进展；与此同时，又不致因大量的材料造成学生的学习负担。同样，编写适用于该领域的教材也存在着一定困难，这主要是由于作者必须在前沿材料与历史观点之间寻找某种平衡。

本书经历了主题选择的过程，目的是适应那些以前曾对本领域有过广泛了解的读者们。第1～5章涉及到计算机组织领域中的许多基本概念，包括怎样测度计算机性能、计算机如何表示数值数据和程序、计算机中的不同编程模型，以及有关处理器设计的基础知识。第6、7章涵盖了流水线和指令级并行性这两种方法，它们对于现代处理器的性能来说至关重要。第8～10章涵盖了存储器系统设计方面的内容，包括存储器层次结构、高速缓冲存储器以及虚拟存储器。第11章描述了输入/输出系统。第12章介绍了多处理器系统，即将多个处理器组合在一起，以改进系统性能。

希望读者在学习计算机体系结构课程时能够觉得本书很有用。我尝试尽可能清楚地解释好每一个主题，而又避免陷入细节之中。将计算机体系结构和组织领域中的所有内容都压缩到像本书这种篇幅的教材中，是一项挑战。因此，我期待来自读者的各种评论，包括对材料选择、课程练习等的评论，也可以是与本书相关的其他评论。

总之，我想感谢使本书得以出版的所有人员，包括我的父母、朋友、伊利诺伊（Illinois）大学的同事们，以及曾经教育过我的所有老师们。此外，我还想特别感谢 McGraw－Hill 出版公司的工作人员，是他们鼓励我完成了本书的写作，并且允许我拖延了本书的写作日程安排。

Nicholas P. Carter

目　录

第1章 概 述

1.1 本书目的

本书适合用作在校大学生的计算机体系结构课程的高级辅助教材，或者研究生计算机体系结构课程的入门教材。其主要读者对象是与计算机体系结构课程相关的广大师生，这些人会对书中附加的解释、实践问题和样例感兴趣。利用这些解释、实践问题和样例，可以帮助他们理解书中所介绍的相关材料，或者有助于他们备课。

1.2 假定背景

本书假定读者具有类似以下背景：电子工程或者计算机科学程序设计专业的大学二、三年级的学生，但还没有学习过有关计算机组织或者计算机体系结构方面的课程。还假定读者基本熟悉各种计算机运算和术语，并且能在一定程度上熟练使用高级程序设计语言进行编程。

1.3 涵盖材料

本书所涵盖的主题要比一学期的计算机体系结构课程中所涉及的主题稍微宽泛一些，以增强其适用性。读者或许能够发现书中的附加材料对于复习学过的内容很有用，或者可将它们看作是介绍更高级主题的入门读物。本书首先讨论了数据表示和计算机算术运算。接着，分章介绍了计算机组织和编程模型。从第5章开始的三章（即第5～7章）讨论了处理器设计问题，其中包括流水线和指令级并行性。接下来的三章都是与存储器系统相关的内容，包括虚拟存储器和高速缓冲存储器。最后两章讨论了输入/输出系统，并且介绍了多处理器系统。

1.4 本章目标

本章目标是为读者准备与后续章节相关的材料，这主要是通过讨论影响计算机性能的基本技术和计算机性能的测度方法来实现的。

读完本章内容和完成课后练习以后，读者应该能够做到：

1）了解并且能够讨论有关晶体管密度、电路性能以及计算机系统整体性能等方面改进速度的历史情况。

2）了解评价计算机性能的常用方法。

3）能够计算出计算机系统某一组成部分中所发生的变化对该系统整体性能的影响情况。

1.5 技术趋势

自20世纪80年代初期以来，计算机性能一直受集成电路性能改进方面的影响，集成电路被用来实现微处理器、存储器芯片以及其他计算机部件。随着时间的推移，集成电路在以下方

面获得了改进：密度（在固定面积的硅芯片上能够放置多少晶体管和电线）、速度（基本逻辑门和存储设备的运算速度）和面积（能够制造的最大集成电路的物理尺寸）。

在过去的二十年中，推动计算机性能获得极大增长的原因是，芯片的速度和密度改进呈几何速度而不是线性速度增长。这就意味着，性能的年增长速度与其前一年度的性能比率相对不变，而不是一个绝对不变的常量。硅芯片上所能够制造的晶体管数的年均增长率约为50%，而晶体管的速度增长情况是：基本逻辑门（比如与门、或门等）延时的年均下降比例是13%。计算机性能呈几何速度而不是线性速度增长，这一现象被称作是摩尔定律。

【例1-1】自20世纪70年代后期以来，动态随机存取存储器（DRAM）芯片中能够存储的数据量每隔3年增长4倍，年均增长率是60%。

从20世纪70年代后期到20世纪80年代后期，微处理器性能主要受制造技术的改进所驱动，其性能改进的年均增长率是35%。此后，尽管半导体制造技术的增长率相对固定不变，但微处理器性能改进的年均增长率实际上更高了，超过50%。性能改进方面的提高要归功于计算机体系结构和组织方面的改进。这样，就使得计算机体系结构能够充分利用集成电路密度增长所带来的好处，为微处理器和存储器系统增加新特性，从而能够比提高潜在晶体管的速度获得更高的性能改进。

1.6 性能测度

本章讨论了计算机性能随着时间的推移而不断获得改进，但没有给出有关性能的正式定义。这在一定程度上是因为“性能”一词在计算机系统中是一个含义模糊的术语。一般来说，性能描述的是在某个给定系统上执行一个或者多个程序的速度有多快。该系统执行程序所用的时间越短，其性能就越高。

计算机性能的最佳测度指标是用户想要执行的程序的执行时间。但是，在决定想要购买哪台计算机或者做出决策之前，对在某个特定计算机系统上运行的所有程序进行测试通常是不现实的。相反，计算机体系结构设计人员已经提出了用于描述计算机性能的许多测度指标，本章将会讨论其中的部分测度指标。计算机体系结构设计人员还为测度单个计算机子系统的性能设计了许多测度指标，这些测度指标将会在涵盖相关子系统的章节中予以讨论。

需要记住的是，除了性能测度指标以外，还有许多因素也会影响设计或者购买决策。易编程性就是值得考虑的一个重要因素，因为开发特定程序所需要的时间和费用也许要比该程序开发出来以后在执行时间上的差异更加重要。另一重要因素是兼容性。大多数程序是以二进制映象形式进行销售的，它们只能运行在特定系列的处理器上。当用户想要的程序不能在某个给定系统上运行时，无论该系统执行其他程序的速度有多快，都是毫无意义的。

1.6.1 MIPS

计算机性能的早期测度指标是指在某个给定计算机上执行指令的速度，其计算方法是将程序中所执行的指令数除以运行该程序所需要的时间，通常表示为每秒百万条指令数（millions of instructions per second，简称MIPS）。MIPS已经不再用作一种性能测度指标，这主要是因为它不能够解释以下事实，即不同系统在执行给定程序时需要用到不同的指令数。由于计算机的

MIPS 测度指标不能说明执行给定任务时需要多少条指令，所以它要比用于比较不同计算机系统性能的其他测度指标的用处更小。

1.6.2 CPI/IPC

用来描述计算机性能的另一测度指标是执行每条指令所需要的时钟周期数，通常称*每指令周期数*（cycles per instruction，简称 CPI）。计算运行在某个给定系统上的某个程序的 CPI 的方法是，将执行该程序所需要的时钟周期数除以该程序运行过程中所执行的指令数。对于每个周期能够执行多条指令的那些系统来说，通常使用 IPC（instructions executed per cycle，*每周期执行的指令数*）测度指标来代替 CPI。IPC 的计算方法是，将某个程序运行过程中所执行的指令数除以执行该程序所需要的时钟周期数，即 CPI 的倒数。这两个测度指标给出的信息相同，在确定选择使用哪个测度指标时，通常是基于判断其中哪个指标的值大于 1 来进行的。在使用 IPC 和 CPI 来比较不同计算机系统的性能时，重要的是要记住：IPC 值越高，表明 IPC 值高的程序执行指令时所需要的周期数要比 IPC 值低的程序少；CPI 值越高，表明 CPI 值高的程序执行指令时所需要的周期数要比 CPI 值低的程序多。因此，IPC 值较大即表明该系统的性能较好，CPI 值较大则表明系统的性能较差。

【例 1-2】某个给定程序中包括由 100 条指令组成的循环，并且需要执行 42 次循环。在某个给定系统上执行该程序所需要的时间是 16 000 个周期，问：在该系统上运行该程序时，CPI 和 IPC 的值分别是多少？

【解答】由于该循环由 100 条指令组成，并且需要执行 42 次循环，所以执行的指令总数是 $100 \times 42 = 4200$。另外，由于执行该程序需要 16 000 个周期，故 CPI 为：16 000/4200 = 3.81。在计算 IPC 值时，只需要将 4200 条指令除以 16 000 个周期，即可得到 IPC 的值为 0.26。

一般来说，在测度系统性能时，IPC 和 CPI 这两个测度指标比 MIPS 的用处更小，因为它们都没有包含有关系统时钟频率的任何信息，或是该系统在执行某个任务时需要多少条指令。当知道某个给定程序在系统中运行时的 MIPS 值时，将该 MIPS 值与该程序运行过程中所执行的指令数相乘，即可确定需要多长时间才能够执行完该程序。当知道某个给定程序在系统中运行时的 CPI 值时，将该 CPI 值与该程序运行过程中所执行的指令数相乘，即可得到执行完该程序所需要的周期总数，但必须了解每秒钟的时钟周期数（即该系统的时钟频率），才能将其转换成执行该程序所需要的时间。

因此，CPI 和 IPC 很少用来比较实际计算机系统的性能。但在计算机体系结构研究中，它们都是十分常见的测度指标，因为计算机体系结构研究大多在仿真环境下进行，并且利用能够模拟某种特定体系结构的程序来估计在该体系结构中执行某个给定程序所需要的周期数。这些模拟程序通常不能够预测它们所模拟系统的周期时间。因此，CPI 或者 IPC 通常就成为在估计系统性能时所能够获得的最佳测度指标。

1.6.3 基准测试套件

正如前文所讨论的那样，MIPS 和 CPI、IPC 在用作测度计算机性能的指标时都存在明显的局限性。*基准测试套件*是测度计算机性能的第三种指标，用来解决 MIPS、CPI 或者 IPC 所存在

的上述局限性。

基准测试套件由一组程序构成，该组程序被认为是在系统上运行的典型程序。某个系统在基准测试套件中的得分是根据该系统执行该套件中全部程序所需要的时间而定的。由于存在许多不同的基准测试套件，所以会生成该系统在不同应用类型中的性能估计值。

最著名的基准测试套件之一是 SPEC 套件，它由标准性能评价公司（Standard Performance Evaluation Corporation）开发。到本书出版时，SPEC 套件的当前最新版本是 SPEC CPU2000 基准测试套件，这是 SPEC 基准测试套件自 1989 年首次发布以来的第三个主要修订版本。

基准测试套件要比 MIPS、CPI、IPC 等测度指标具有更多的优点。首先，其性能结果是基于总执行时间得出的，而不是基于指令执行的比率。其次，基准测试套件是通过对系统在多个程序中的性能测度值求平均数来生成其平均速度估计值。这样，与该系统在任一程序中获得的 MIPS 分值相比，该系统在基准测试套件中获得的总分值更能够揭示出其总体性能情况。此外，许多基准测试不仅要求制造商能够发布其系统在基准测试套件中的总得分情况，而且要求发布该系统在基准测试套件中各个程序的测试结果。这样，当知道系统将会被用于某个特定应用时，就能够对各个基准测试结果直接进行比较。

1.6.4 几何平均数与算术平均数

在利用基准测试套件中所包含的程序进行测试并对测试结果求平均数时，许多基准测试套件采用的是几何平均数，而不是算术平均数，这是因为某个极端数值对一组数值的几何平均数影响较小，而对该组数值的算术平均数影响较大。在采用几何平均数时，很难通过使系统在基准测试套件中的某个程序上取得较好的性能测试值，来获得较高的基准测试套件总分值。这样，就使得该系统的总分值能够更好地揭示出该系统在大多数程序上的性能情况。

计算 n 个数值的几何平均数的方法是，先将 n 个数值相乘，再求出该乘积的 n 次方根即可。计算一组数值的算术平均数的方法是，先将所有数值相加，再除以这些数值的个数即可。

【例 1-3】 已知一组数值：4、2、4、82，问：其算术平均数和几何平均数分别是多少？

【解答】 该组数值的算术平均数是：

$$\frac{4+2+4+82}{4}=23$$

该组数值的几何平均数是：

$$\sqrt[4]{4\times2\times4\times82}=7.16$$

注意：在该样例中，该组数值中的某个极端数值（即 82）对算术平均数的影响要大于对几何平均数的影响。

1.7 加速比

计算机体系结构中经常使用“加速比”这个术语来描述在对体系结构进行不同改进时该体系结构的性能变化情况。加速比只不过是产生变化之前和之后的执行时间比率，即：

$$\text{加速比}=\frac{\text{变化前的执行时间}}{\text{变化后的执行时间}} \tag{1-1}$$

例如，当某个程序在体系结构旧版本中的执行时间是 25 秒，而在新版本中的执行时间是 15 秒时，则总加速比是 25 秒/15 秒 = 1.67。

1.8 Amdahl 定律

在设计高性能计算机系统时所采用的最重要规则是，*加快经常性事件的速度*。从本质上看，这就意味着：某个给定性能改进措施对总体性能的影响既依赖于使用该改进措施时对整个系统性能的提高程度，也依赖于使用这种改进措施的频率。从本质上看，这条规则可以表示为 Amdahl 定律，即：

$$\text{新执行时间} = \text{旧执行时间} \times \left(\text{Frac}_{\text{unused}} + \frac{\text{Frac}_{\text{used}}}{\text{Speedup}_{\text{used}}} \right) \tag{1-2}$$

在该公式中，$\text{Frac}_{\text{unused}}$是指不使用改进措施的那部分程序的时间部分（而不是执行的指令数），$\text{Frac}_{\text{used}}$是要使用改进措施的那部分程序的时间部分，$\text{Speedup}_{\text{used}}$是指使用改进措施时所产生的加速比（当一直使用该改进措施时，$\text{Speedup}_{\text{used}}$是指总加速比）。注意：在计算 $\text{Frac}_{\text{unused}}$和 $\text{Frac}_{\text{used}}$时，使用的是修改之前的执行时间。在计算这两个数值时，若使用修改之后的执行时间，则会得到错误的结果。

利用加速比定义，可以重写 Amdahl 定律，即：

$$\text{加速比} = \frac{\text{旧执行时间}}{\text{新执行时间}} = \frac{1}{\text{Frac}_{\text{unused}} + \dfrac{\text{Frac}_{\text{used}}}{\text{Speedup}_{\text{used}}}} \tag{1-3}$$

【例 1-4】假定某个给定体系结构中没有用于支持乘法运算的硬件，所以在执行乘法运算时，必须通过重复执行加法运算来实现（这种情况出现在一些早期的微处理器中）。在软件中执行一次乘法运算需要 200 个周期，而在硬件中执行一次乘法运算需要 4 个周期，问：当某个程序将 10%的时间用于执行乘法运算时，从硬件支持乘法运算中所获得的总加速比是多少？当程序将 40%的时间用于执行乘法运算时，从硬件支持乘法运算中所获得的总加速比又是多少？

【解答】在上述两种情况下，使用乘法运算硬件所获得的加速比都是：200/4 = 50（即为不使用乘法运算硬件与使用乘法运算硬件所需要的时间比率）。当程序将 10%的时间用于执行乘法运算时，$\text{Frac}_{\text{unused}} = 0.9$，$\text{Frac}_{\text{used}} = 0.1$。将它们代入到 Amdahl 定律（公式 1-3）中，即可得到：总加速比 = 1/[0.9 + (0.1/50)] = 1.11 。在增加乘法运算硬件之前，当程序将 40%的时间用于执行乘法运算时，$\text{Frac}_{\text{unused}} = 0.6$，$\text{Frac}_{\text{used}} = 0.4$，故总加速比 = 1/[0.6 + (0.4/50)] = 1.64 。

该样例说明了要使用改进措施的时间部分对整体性能的影响情况。当 $\text{Speedup}_{\text{used}}$趋于无穷大时，总加速比就趋于 $1/\text{Frac}_{\text{unused}}$，因为该改进措施对于不使用改进措施的程序部分的执行时间没有任何影响。

1.9 本章小结

本章目的是为本书其他章节提供上下文背景知识。本章解释了驱动计算机性能的一些技术力量，提供了用于讨论和评价系统性能的一个框架，该框架被应用于全书之中。

读完本章之后，要求读者应该了解以下重要概念：

1）计算机技术受半导体制造技术方面的改进所驱动，并且这些改进呈几何速度增长，而不是呈算术速度增长。

2）对计算机性能进行测度有多种方式，对总体性能最有效的测度指标都是基于系统在多个应用程序之上的性能来进行测度的。

3）重要的是要了解如何生成某个给定性能测度指标，以了解它在预测某个给定应用程序的系统性能时所起的作用。

4）体系结构变化对整个性能的影响不仅依赖于使用改进措施时它对整个系统性能的改进程度有多大，而且还依赖于使用该改进措施的频率。因此，改进措施对整个系统性能的影响受限于不使用改进措施的那部分程序的时间，而与使用该改进措施时的加速比无关。

习题与解答

技术趋势（Ⅰ）

1.1 假如计算机技术改进速度之快正如本章所说的那样，请考虑以下情况：当汽车技术的改进速度也同样快时，会是什么情况？假设 1977 年每辆汽车的平均最高速度是每小时 100 英里，汽油平均经济效率是每加仑汽油可行驶 15 英里。若平均最高速度和经济效率在 1977～1987 年期间的年均增长率都是 35%，在 1987～2000 年期间的年均增长率均为 50%。对照计算机性能情况，到 1987 年时每辆汽车的平均最高速度和经济效率分别是多少？到 2000 年时每辆汽车的平均最高速度和经济效率又分别是多少？

解答：

1）在 1987 年：

由于 1977～1987 年之间的时间跨度是 10 年，所以平均最高速度和经济效率的增长比例都是：$(1.35)^{10} = 20.1$。因此，到 1987 年时每辆汽车的平均最高速度是每小时 2010 英里，经济效率是每加仑汽油可行驶 301.5 英里。

2）在 2000 年：

时间又过去了 13 年，并且平均最高速度和经济效率在 1987～2000 年期间的年均增长率均为 50%。因此，与 1987 年的数值相比，平均最高速度和经济效率的增长比例都是：$(1.5)^{13} = 194.6$。因此，到 2000 年时每辆汽车的平均最高速度是每小时 391 146 英里，经济效率是每加仑汽油可行驶 58 672 英里。如果是这样，每辆汽车的平均最高速度就太快了，从地球到月球之间的距离也只需要不到 40 分钟的行驶时间，并且驾车环球行驶一圈所耗用的汽油不到 10 加仑。

技术趋势（Ⅱ）

1.2 自 1987 年以来，计算机性能的年均增长率大约是 50%，其中制造技术改进方面的年均增长率约为 35%，体系结构改进方面的年均增长率大约是 15%。

1）若将 1988 年 1 月 1 日普通计算机的性能定义为 1，问：到 2001 年 1 月 1 日时，普通计

算机的期望性能是多少？

2）假定自 1987 年以来，计算机体系结构方面没有任何改进，所以制造技术方面的改进就成为性能改进方面的惟一来源，问：到 2001 年 1 月 1 日时，普通计算机的期望性能是多少？

3）现在假定自 1987 年以来，制造技术方面没有任何改进，所以计算机体系结构方面的改进就成为性能改进方面的惟一来源，问：到 2001 年 1 月 1 日时，最快计算机的期望性能又是多少？

解答：

1）由于计算机性能改进的年均增长率是 50%，并且从 1998 年 1 月 1 日到 2001 年 1 月 1 日之间的时间跨度是 13 年，所以到 2001 年 1 月 1 日时，普通计算机的期望性能是：$1 \times (1.5)^{13} = 194.6$。

2）在此情况下，由于计算机性能改进的年均增长率为制造技术改进方面的增长率，即 35%，所以到 2001 年 1 月 1 日时，普通计算机的期望性能是 49.5。

3）在此情况下，由于计算机性能改进的年均增长率为计算机体系结构改进方面的增长率，即 15%，所以到 2001 年 1 月 1 日时，普通计算机的期望性能是 6.2。

加速比（Ⅰ）

1.3　在某个计算机的 1998 年版本中，执行某个程序所用的时间是 200 秒。在该计算机的 2000 年版本中，执行同一程序所用的时间是 150 秒。问：在这两年期间，制造商所获得的加速比是多少？

解答：

$$\text{加速比} = \frac{\text{变化前的执行时间}}{\text{变化后的执行时间}}$$

因此，本题中的加速比为 200 秒/150 秒 = 1.33。显然，该制造商已经远远落后于整个业界范围内的性能增长比率。

加速比（Ⅱ）

1.4　要使加速比达到 3，并且某个程序的原有执行时间是 78 秒，问：必须将该程序的执行时间削减到多少秒时才算合适？

解答：

在本题中，由于已知加速比和变化前执行时间的具体值，所以只需要将这两个值代入到加速比计算公式中，即可求出变化后的执行时间。由此可知：只有将该程序的执行时间削减到 26 秒，才能使加速比达到 3。

性能测度（Ⅰ）

1.5　回答以下问题：

1）为什么基准测试程序/套件被用来测度计算机性能？

2）为什么计算机体系结构设计人员使用多个基准测试程序/套件，而不是使用某个“最佳的”基准测试程序/套件？

解答：

1）计算机系统经常被用来运行各种程序，其中一些程序在购买或者建造该系统时很可能并不存在。因此，通常无法对该系统在运行于计算机上的一组程序中的性能进行测度。相反，可以使用基准测试程序/套件来测度计算机系统在一个或者多个应用程序中的性能，并且这些应用程序被看作是在该计算机上运行的一组程序的代表。

2）计算机体系结构设计人员之所以使用多个基准测试程序/套件，是因为计算机被用于多种应用程序，其性能依赖于计算机系统中差别极大的多方面情况。例如，数据库和事务处理应用程序的性能主要依赖于计算机系统中输入/输出子系统的性能。相反，科学计算应用程序主要依赖于计算机系统中处理器和存储器子系统的性能。与应用程序相类似的是，基准测试套件在对计算机系统各个组成子系统的依赖程度方面相差很大，重要的是要使用对子系统和预定应用程序依赖程度相同的基准测试套件。若使用对处理器敏感的某个基准测试套件来评价计算机性能，当该基准测试套件被设计用于事务处理时，将不能够很好地估计出在这些系统上处理事务的速率。

性能测度（Ⅱ）

1.6 在运行某个特定程序时，计算机 A 的速度是 100 MIPS，计算机 B 的速度是 75 MIPS。但是，计算机 A 执行该程序的时间是 60 秒，而计算机 B 执行该程序的时间是 45 秒。问：为什么会出现这种情况？

解答：

MIPS 测度的是处理器执行指令的速率，但在执行某个给定计算时，不同处理器体系结构需要的指令数不一样。当计算机 A 执行完该程序时必须执行的指令数明显多于计算机 B 时，计算机 A 运行该程序所花费的时间很可能会多于计算机 B 运行该程序所花费的时间，除非计算机 A 每秒钟执行更多的指令。

性能测度（Ⅲ）

1.7 计算机 C 在某个基准测试套件中的得分是 42 分（该分值越高越好），而计算机 D 在相同基准测试套件中的得分是 35 分。当运行某个程序时，计算机 C 所花费的时间多于计算机 D。问：为什么会出现这种情况？

解答：

最合理的解释是：尽管该程序高度依赖于计算机系统中某方面的特性，但该特性在基准测

试套件中并不突出强调。例如，当程序中可能会执行大量浮点计算，但基准测试套件强调的是整数性能时，就会出现上述情况，反之亦然。

CPI

1.8　在某个给定系统上运行某个程序所花费的时间是 1 000 000 个周期。如果该系统获得的 CPI 值为 40，问：运行该程序时执行了多少条指令？

解答：

CPI 的计算公式是：CPI = 运行程序所用的周期数/执行的指令数。因此，运行该程序时执行的指令数 = 运行程序所用的周期数/CPI = 1 000 000/40 = 25 000，即运行该程序时执行了 25 000条指令。

IPC

1.9　在运行某个程序时，执行了 35 000 条指令，所用的时间是 17 000 个周期，问：IPC 是多少？

解答：

IPC 的计算公式是：IPC = 执行的指令数/运行程序所用的周期数。因此，运行该程序时，IPC = 35 000/17 000 = 2.06。

几何平均数与算术平均数

1.10　假定在 SPEC2000 基准测试套件的整数部分中各个程序的基准测试得分情况如下表所示，请计算各组分值的算术平均数和几何平均数。注意：这些分值并不代表对某个计算机进行实际测度的集合情况，之所以选择这些分值，是为了举例说明采用不同方法计算平均值对基准测试得分的影响。

基准测试程序	使用改进措施前的得分	使用改进措施后的得分
1.64.gzip	10	12
175.vpr	14	16
176.gcc	23	28
181.mcf	36	40
186.crafty	9	12
197.parser	12	120
252.eon	25	28
253.perlbmk	18	21
254.gap	30	28
255.vortex	17	21
256.bzip2	7	10
300.twolf	38	42

解答：

由于该基准测试套件中共有12个基准测试程序，所以求算术平均数的计算方法是：先将每组数值中的所有数值累加起来，再除以12。求几何平均数的计算方法是：先将每组数值中的所有数值相乘，再对乘积求12次方根。利用这两种方法，即可得到下列数值：

使用改进措施之前：算术平均数 = 19.92，几何平均数 = 17.39。

使用改进措施之后：算术平均数 = 31.5，几何平均数 = 24.42。

从上述计算结果中可以看到：同几何平均数相比，算术平均数对数值集合中变化较大的某个数值更敏感一些。在对计算机体系结构使用改进措施以后，大多数基准测试程序的得分变化相对较小，但197.parser程序的得分提高到原来的10倍。这就使得算术平均数提高了近60%，而几何平均数仅提高了40%。由于几何平均数可以降低对单个数值的敏感程度，所以基准测试专家更愿意采用几何平均数来计算多个基准测试程序测试结果的平均得分，因为即使基准测试结果得分集合中的某个分值极高或者极低，对采用几何平均数计算得到的总得分的影响也较小。

Amdahl定律（Ⅰ）

1.11 假定在运行某个给定程序时，某个计算机将90%的时间用于处理某种特定计算类型。其制造商进行了修改以后，使该计算类型的执行速度提高了10倍。问：

1）当该程序原来的执行时间是100秒时，该程序修改以后的执行时间是多少？

2）新旧系统之间的加速比是多少？

3）新系统会将多大比例的执行时间用于执行改进后的计算类型？

解答：

1）解答本小题可以直接应用Amdahl定律，其计算公式是：

$$\text{新执行时间} = \text{旧执行时间} \times \left(\text{Frac}_{\text{unused}} + \frac{\text{Frac}_{\text{used}}}{\text{Speedup}_{\text{used}}} \right)$$

从题目中的已知条件得到：旧执行时间 = 100秒，$\text{Frac}_{\text{used}} = 0.9$，$\text{Frac}_{\text{unused}} = 0.1$，$\text{Speedup}_{\text{used}} = 10$。将它们代入到上述公式后，即可得到新执行时间是19秒。

2）利用加速比计算公式，即可计算出加速比为5.3。另外一种方法是，将第1小题中的结果值（即19秒。——译者注）代入到Amdahl定律用于计算加速比的计算公式中，也可得到相同结果。

3）应用Amdahl定律并不能够直接回答本小题问题。由于原系统将90%的时间用于执行改进后的计算类型，所以它将执行该程序的100秒时间中的90%的时间（即90秒。——译者注）用于执行该计算类型，即改进后的新系统执行该计算类型的时间是：90/10 = 9（秒）。同新执行时间19秒相比，9秒所占的比例是47%。因此，新系统将47%的执行时间用于执行改进后的该计算类型。

另外一种方法是，先计算出原系统用于执行不会改进的计算类型的时间（即10秒）。由于