Welcome

数据科学与大数据技术专业

计 算 机 系 统 基 础

上海体育学院经济管理学院

Wu Ying

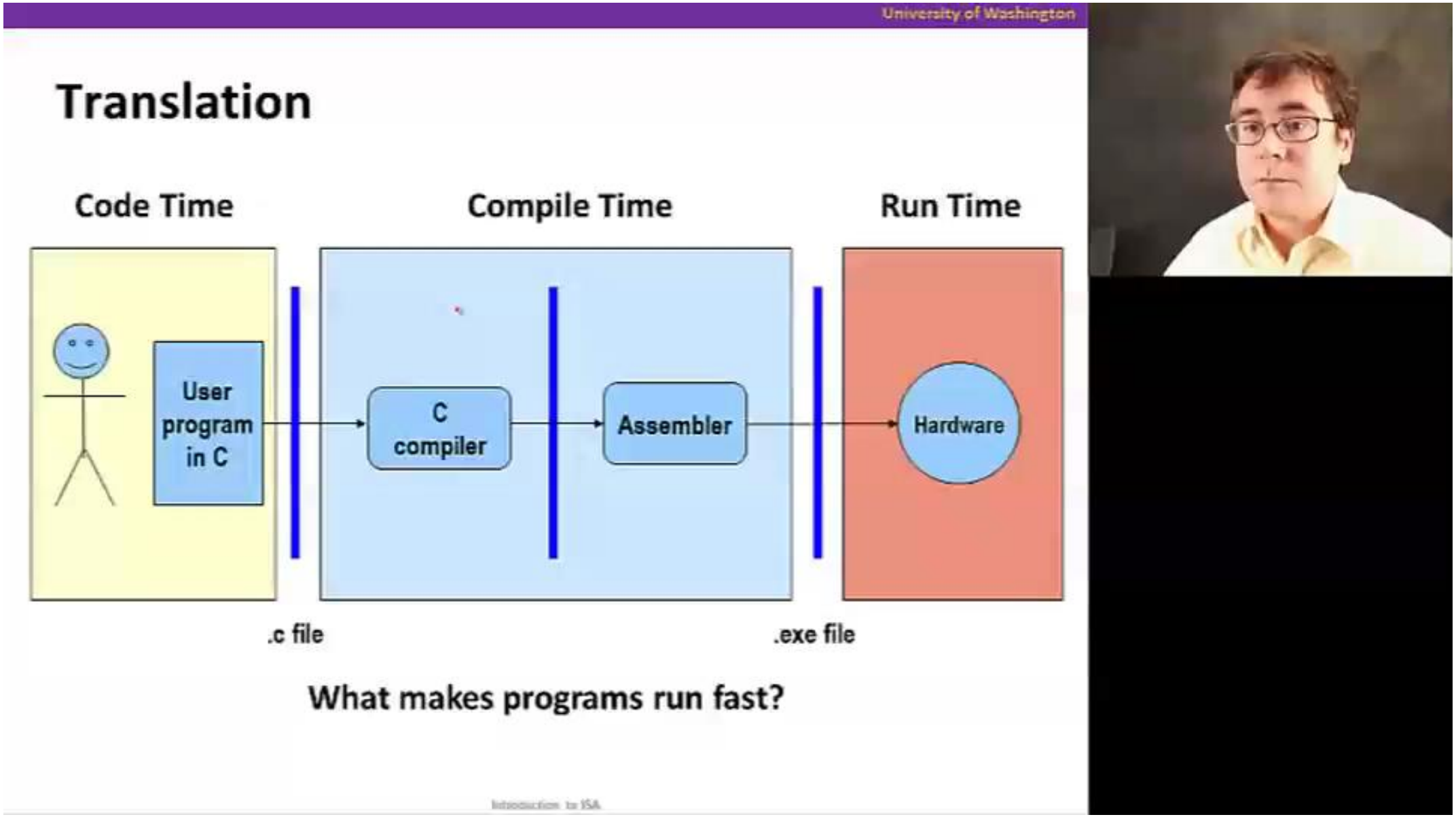
计算机系统
（硬件系统+软件系统）

01

计算机系统的性能评价

- 程序执行时间
- MIPS
- Amdahl定律

在一个给定的处理器上，如何才能让程序跑得更快?



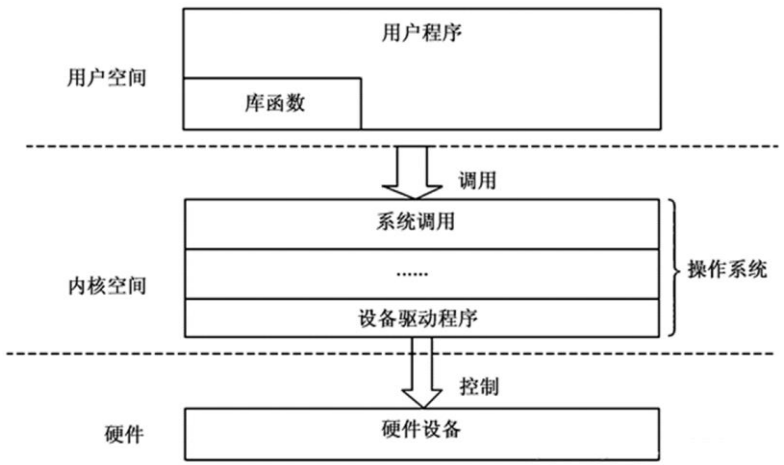
The lifetime of a program 1’ 45” 华盛顿大学 The Hardware/Software Interface
http://courses.cs.washington.edu/courses/cse351

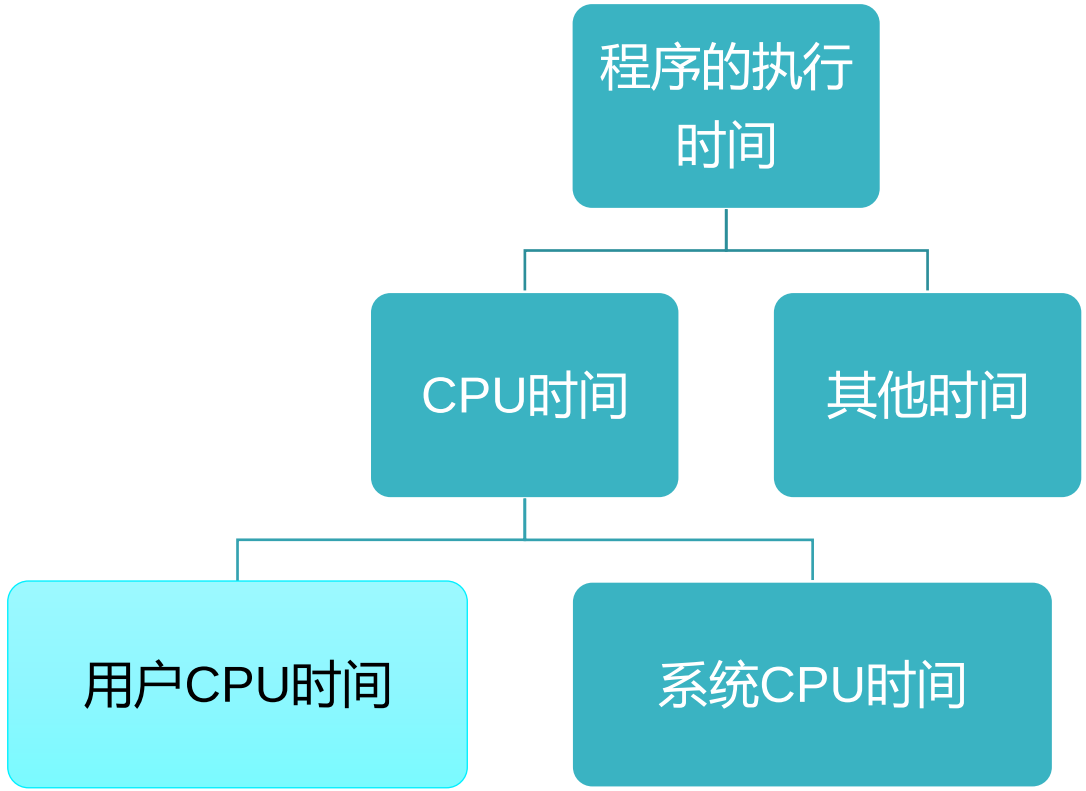
Translation Impacts Performance

- **The time required to execute a program depends on:**
  - *The program* (as written in C, for instance)
  - *The compiler*: what set of assembler instructions it translates the C program into
  - *The instruction set architecture* (ISA): what set of instructions it makes available to the compiler
  - *The hardware implementation*: how much time it takes to execute an instruction

在一个给定的处理器上，如何才能让程序跑得更快?

【1.4.2　P20 】



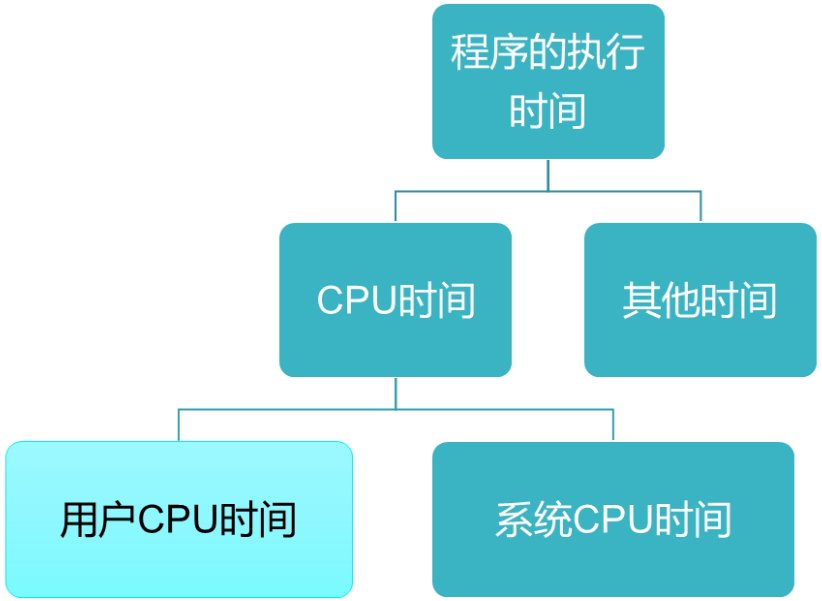


```
Welcome to Alibaba Cloud Elastic Compute Service !

Updates Information Summary: available
    3 Security notice(s)
        3 Moderate Security notice(s)
Run "dnf upgrade-minimal --security" to apply all updates.
Last login: Mon Nov  1 13:35:34 2021 from 118.31.243.210
[root@iZuf6an56zjdutq92u57jkZ ~]# time seq 1000000 | wc -l
1000000

real    0m0.020s
user    0m0.012s
sys     0m0.006s
[root@iZuf6an56zjdutq92u57jkZ ~]# time seq 1000000 | wc -l
1000000

real    0m0.019s
user    0m0.015s
sys     0m0.002s
[root@iZuf6an56zjdutq92u57jkZ ~]#
```

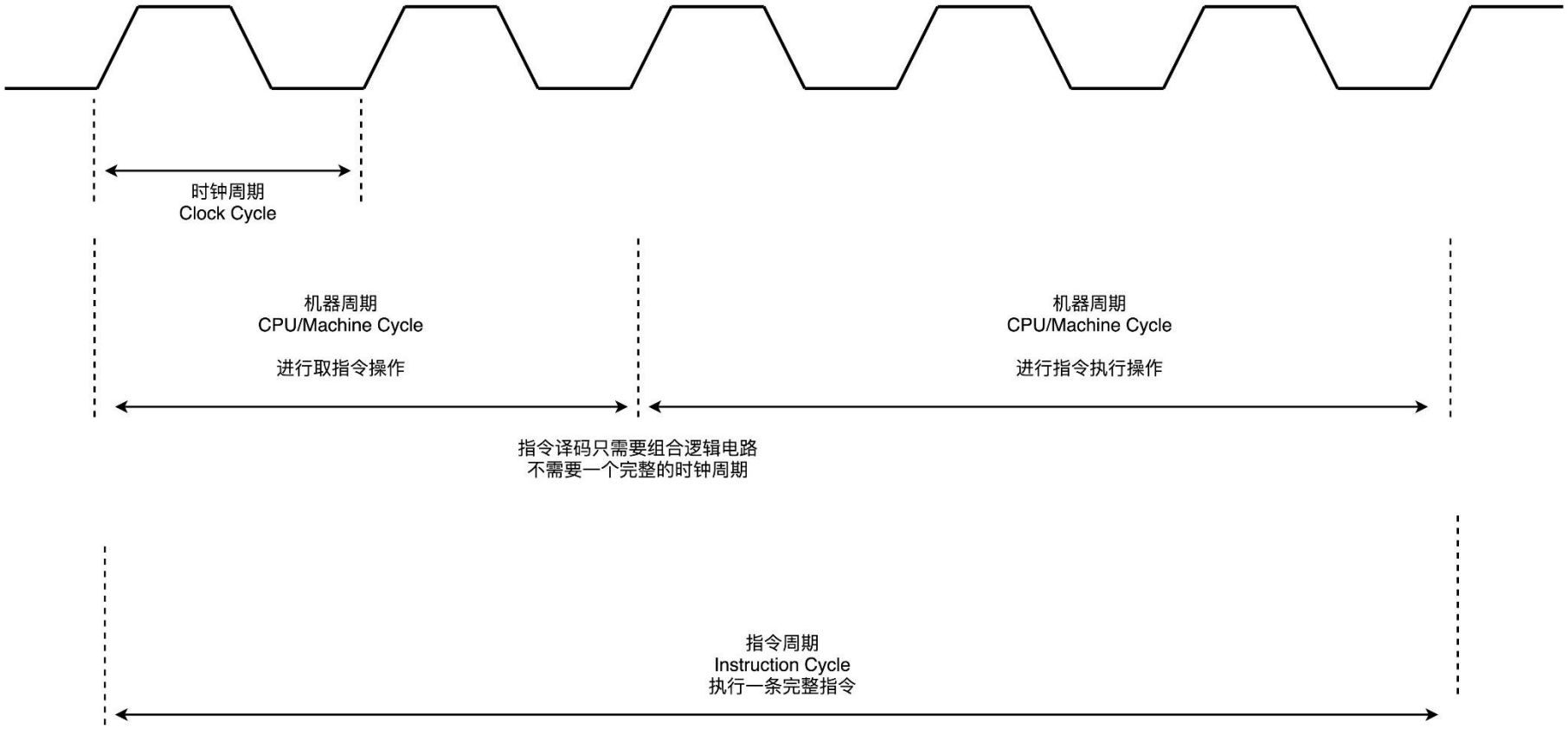


wc -l -(line) 显示行数

程序的 CPU 执行时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

每条指令的平均时钟周期数（Cycles Per Instruction，简称 CPI）

- 一条指令的执行时间包含一个或多个时钟周期



程序的 CPU 执行时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

CPI，Cycle Per Instruction ——每指令周期数，**执行一条指令所需的时钟周期数**

IPC， Instruction Per Cycle——每周期指令数，**每个时钟周期所完成的指令数**

CPI = 1 / IPC

假定某程序P由一个100条指令构成的循环组成

该循环共执行50次

在某系统S中执行程序P花了20000个时钟周期

则系统S在执行程序P时的CPI是多少?

程序P中100条指令，循环执行50次，所以在20000个时钟周期中共执行了5000条指令，所以，系统S在执行程序P时的 CPI = 20000 / 5000 = 4

程序的 CPU 执行时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　×　时钟周期时间 Clock Cycle Time

【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　×　时钟周期时间 Clock Cycle Time

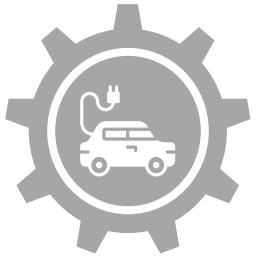
①减少程序总指令数

算法优化

好的算法设计，可能带来更少的指令执行数

用更底层的语言优化

eg. Linux 内核代码使用 C 语言

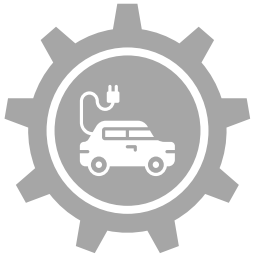


更高效的编译器或者解释器

新的编译器或者解释器，新版本编译器最重要的工作是在新的处理器上，用最新的高效指令，对同样的源代码，可能生成更少的机器码

新的处理器指令

新的处理器指令，对处理某类特殊目的的运算更有帮助



【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　× 时钟周期时间 Clock Cycle Time

一个时钟周期只能执行一条指令

标量处理器 (Scalar Processor)

一个 时钟周期可以执行多条指令

超标量处理器 (Superscalar Processor)，利用 CPU 流水线提高指令并行度

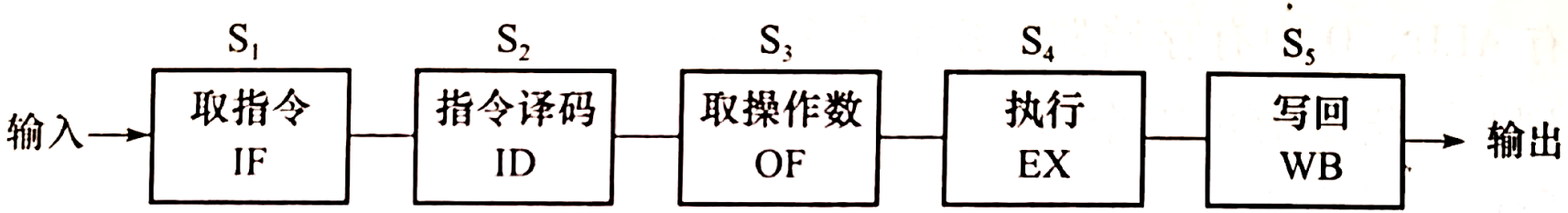
流水线的并行度和效率，又取决于：取指令速度，访存速度，指令乱序执行 (Out-Of-Order Execution)，分支预测执行 (Branch Prediction Execution)和投机执行 (Speculative Execution）的能力等多种因素。一旦流水线并行执行的能力降低，那么程序的性能就会受到影响。

多核处理器系统，程序通过并行编程来提高指令的并行度，CPU 架构转为 Multi-Core 和 Many-Core

②减少每条指令执行所需平均时钟周期数

③减少时钟周期时间，即提高 CPU 的主频

指令流水线【5.3.1 P239】



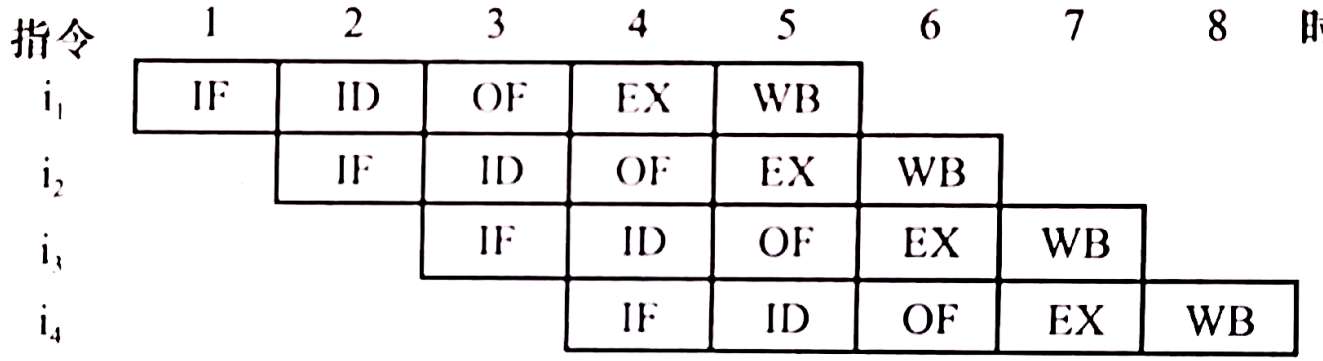


图 5.13 一个 5 段指令流水线

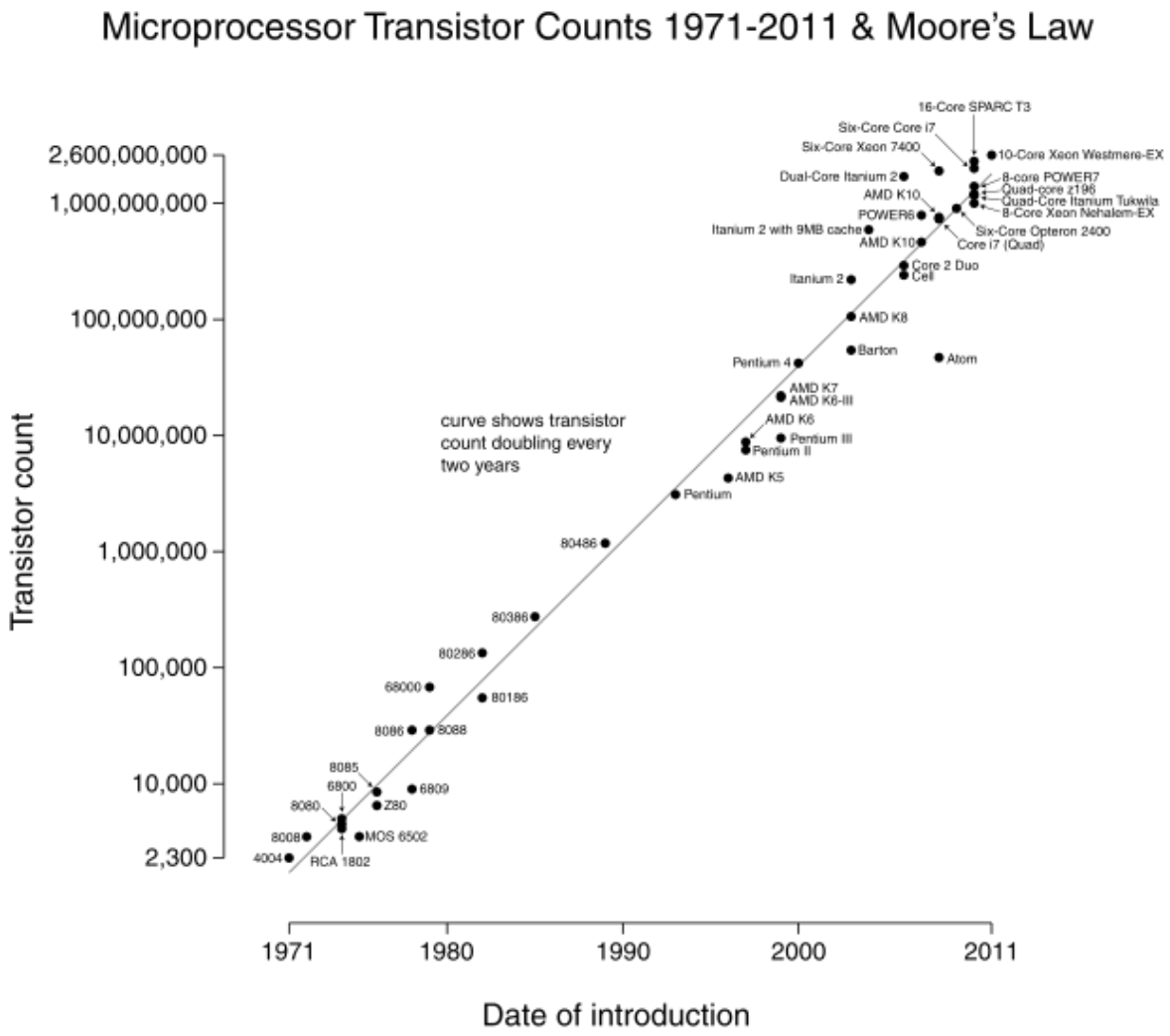
后一条指令的第 i 步，与前一条指令的第 i+1 步同时进行

并行前，4条指令需要20个时钟周期，并行后，8个时钟周期

流水线方式并不能缩短一条指令的执行时间，在划分比较均匀的情况下，可以大大提高整个程序执行的指令吞吐率。降低每条指令执行所需平均时钟周期数

【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　× 时钟周期时间 Clock Cycle Time



②减少时钟周期时间

即提高 CPU 的主频

摩尔定律

集成电路上可以容纳的晶体管数目，大约每经过18个月便会增加一倍。

换言之，处理器的性能每隔两年翻一倍

专家贡献

摩尔定律

语音 编辑 讨论 12 上传视频

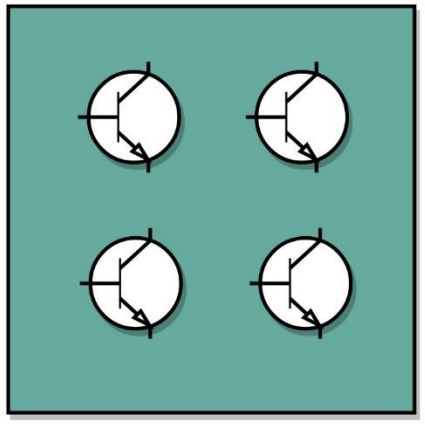
收藏 3349

本词条由"科普中国"科学百科词条编写与应用工作项目 审核 。

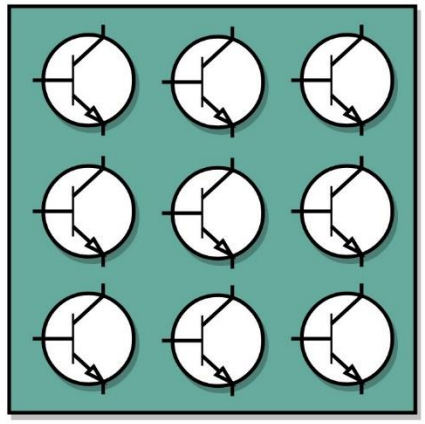
摩尔定律是英特尔创始人之一戈登·摩尔的经验之谈，其核心内容为：集成电路上可以容纳的晶体管数目在大约每经过18个月便会增加一倍。换言之，处理器的性能每隔两年翻一倍。[1]

摩尔定律是内行人摩尔的经验之谈，汉译名为"定律"，但并非自然科学定律，它一定程度揭示了信息技术进步的速度。[2]

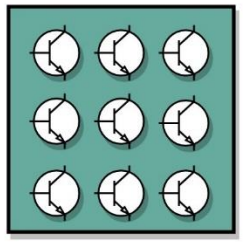
中文名	摩尔定律	提出者	戈登·摩尔（Gordon Moore）
外文名	Moore's Law	提出时间	1965年4月
别　名	摩尔法则	适用领域	半导体



芯片内部的晶体管



更多数量的晶体管会带来能耗和散热的挑战



制程的提升解决了能耗和散热的问题，还带来了更小的芯片

程序的 CPU 执行时间 = 程序指令所需要的总的时钟周期数 Clock Cycles × Clock Cycle Time

【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　×　时钟周期时间 Clock Cycle Time

【1.4.2　P20 】

程序的CPU时间　= 程序的总指令数 × CPI（指令执行所需平均周期数）　×　时钟周期时间 Clock Cycle Time

一个 CPU 时钟周期只能执行一条指令

标量处理器 (Scalar Processor)

一个 CPU 时钟周期可以执行多条指令

超标量处理器 (Superscalar Processor)，利用 CPU 流水线提高指令并行度

流水线的并行度和效率，又取决于：取指令速度，访存速度，指令乱序执行 (Out-Of-Order Execution)，分支预测执行 (Branch Prediction Execution)和投机执行 (Speculative Execution）的能力等多种因素。一旦流水线并行执行的能力降低，那么程序的性能就会受到影响。

多核处理器系统，程序通过并行编程来提高指令的并行度，CPU 架构转为 Multi-Core 和 Many-Core

③减少每条指令执行所需平均时钟周期数

②减少每 CPU 时钟周期时间，即提高 CPU 的主频

提高 CPU 主频的同时，保障一个 CPU 时钟周期可以执行更多的指令。

不断提高制造工艺，降低 CPU 的芯片面积和功耗

1.4.1 计算机系统的性能

- 吞吐率

  单位时间内完成的工作量（所传输的信息量）

  eg. 指令吞吐率，单位时间内完成的指令条数（ IPS）

- 响应时间

  作业从提交开始到作业自行完成所用的时间（执行时间+等待时间）

$$[W_0, W_1, W_2, W_3, W_4, \ldots\ldots, W_{15}]$$

dot product

$$[X_0, X_1, X_2, X_3, X_4, \ldots\ldots, X_{15}]$$

$$= [W_0*X_0 + W_1*X_1 + W_2*X_2 + \ldots + W_{15}*X_{15}]$$

$$= \begin{aligned} &[W_0*X_0 + W_1*X_1 + W_2*X_2 + W_3*X_3] = T_1 \quad \text{CPU 0} \\ &+ \\ &[W_4*X_4 + W_5*X_5 + W_6*X_6 + W_7*X_7] = T_2 \quad \text{CPU 1} \\ &+ \\ &[W_8*X_8 + W_9*X_9 + W_{10}*X_{10} + W_{11}*X_{11}] = T_3 \quad \text{CPU 2} \\ &+ \\ &[W_{12}*X_{12} + W_{13}*X_{13} + W_{14}*X_{14} + W_{15}*X_{15}] = T_4 \quad \text{CPU 3} \end{aligned}$$

$$= T_1 + T_2 + T_3 + T_4 = \text{Result} \quad \text{CPU 0}$$

第一，需要进行的计算，本身可以分解成几个可以并行的任务，不会影响最后的结果。

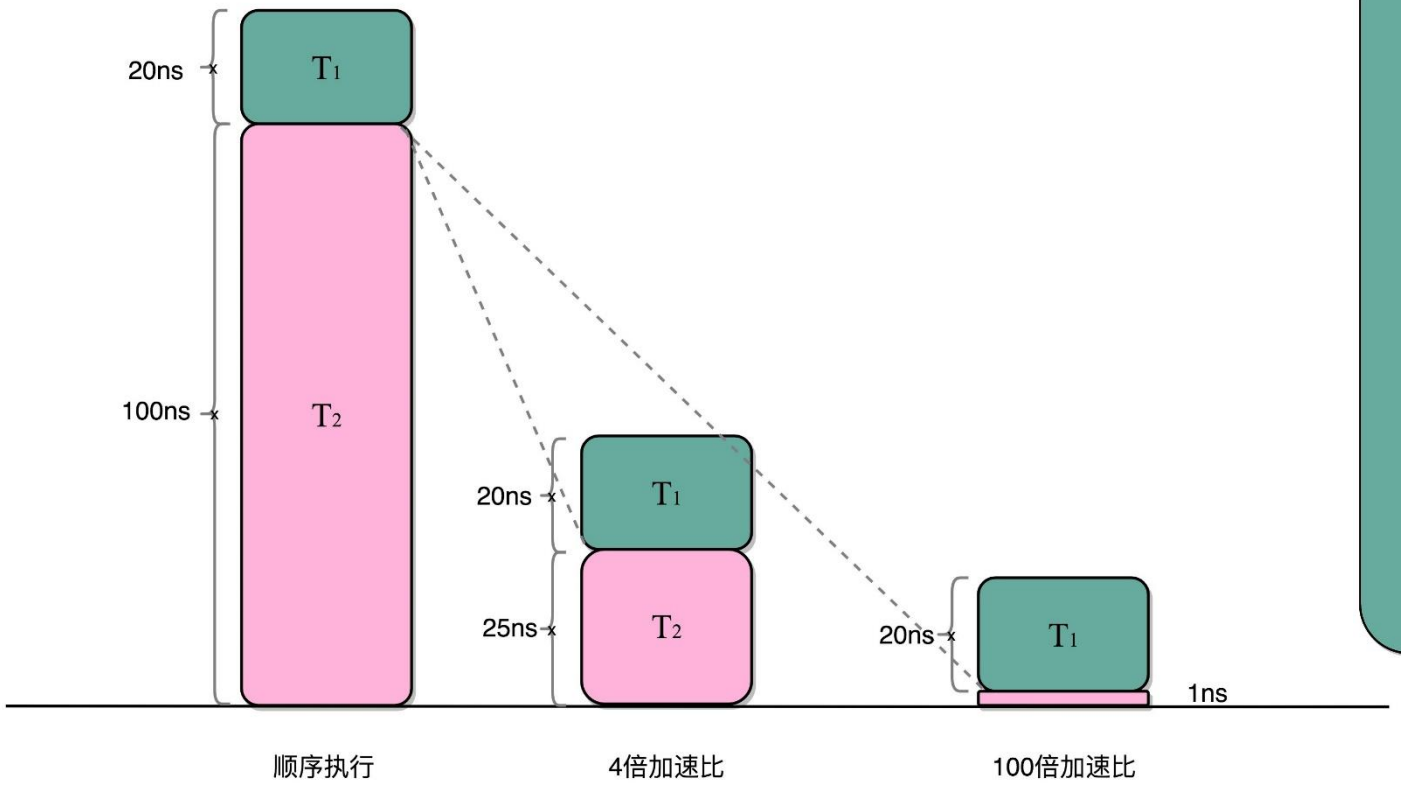
第二，需要能够分解好问题，并确保分解的结果能够汇总到一起。

第三，在“汇总”这个阶段，是没有办法并行进行的，还是得顺序执行，一步一步来。

1.4.5 Amdahl Law

- 对系统中某个硬件部分或者软件中的**某部分进行更新带来的系统性能改进程度**，取决于该硬件部件或软件部分被**使用的频率或其占用总执行时间的比例。**【阿姆达尔定律，系统设计的定量原则之一，1967】

改进后的执行时间 = 该部分当前执行时间 ÷ 改进倍数 + 未改进部分执行时间



T1 = 不受影响的执行时间　　T2 = 受优化影响的执行时间

$$[W_0, W_1, W_2, W_3, W_4, ..., W_{15}]$$

dot product

$$[X_0, X_1, X_2, X_3, X_4, ..., X_{15}]$$

$$= [W_0*X_0 + W_1*X_1 + W_2*X_2 + ... + W_{15}*X_{15}]$$

$$= \begin{aligned} &[W_0*X_0 + W_1*X_1 + W_2*X_2 + W_3*X_3] = T_1 \text{ CPU 0} \\ &+ \\ &[W_4*X_4 + W_5*X_5 + W_6*X_6 + W_7*X_7] = T_2 \text{ CPU 1} \\ &+ \\ &[W_8*X_8 + W_9*X_9 + W_{10}*X_{10} + W_{11}*X_{11}] = T_3 \text{ CPU 2} \\ &+ \\ &[W_{12}*X_{12} + W_{13}*X_{13} + W_{14}*X_{14} + W_{15}*X_{15}] = T_4 \text{ CPU 3} \end{aligned}$$

$$= T_1 + T_2 + T_3 + T_4 = \text{Result CPU 0}$$

例如，上面的各个向量的一小段的点积，需要 100ns，加法需要 20ns，总共需要 120ns

这里通过并行 4 个 CPU 有了 4 倍的加速度。那么最终优化后，就有了 100/4+20=45ns。

100 个 CPU，整个时间？ 100/100+20=21ns

1.4.2 计算机性能的测试

程序的CPU执行时间 = 该程序所包含的指令条数 × 程序综合CPI × 时钟周期

Execution Time (T) = Instruction Count × CPI（Cycle Per Instruction ） × Time Per Cycle

【例1.1P20】

假设某个频繁使用的程序P在机器M1上运行需要10秒， M1的时钟频率为2GHz

设计人员想开发一台与M1具有相同ISA的新机器M2， **采用新技术可使M2的时钟频率增加，但同时也会使CPI增加**

假定程序P在M2上的时钟周期数是在M1上的1.5倍，则M2的时钟频率至少达到多少才能使程序P在M2上的运行时间缩短为6秒？

1.4.2 计算机性能的测试

程序的CPU执行时间 = 该程序所包含的指令条数 × 程序综合CPI × 时钟周期

Execution Time (T) = Instruction Count × CPI（Cycle Per Instruction ） × Time Per Cycle

【例1.1P20】

假设某个频繁使用的程序P在机器M1上运行需要10秒， M1的时钟频率为2GHz

设计人员想开发一台与M1具有相同ISA的新机器M2， **采用新技术可使M2的时钟频率增加，但同时也会使CPI增加**

假定程序P在M2上的时钟周期数是在M1上的1.5倍，则M2的时钟频率至少达到多少才能使程序P在M2上的运行时间缩短为6秒？

【解】

程序P在机器M1上耗用的时钟周期数为：

运行时间×时钟频率 = 10s * 2GHz = 20G

因此，程序P在M2上的时钟周期数为：1.5 * 20 G = 30 G

要使程序P在M2上的运行时间缩短到 6s,则 M2 的时钟频率至少应为：

程序总的时钟周期数 ÷ CPU时间 = 30G / 6s = 5G Hz

由此可见，M2的时钟频率是M1的2.5倍，但M2的速度却只是M1的1.67倍。

1.4.3 用指令执行速度进行性能评估

- MIPS (Million Instructions Per Second)　平均**每秒**执行**百万条**指令

- 等效指令速度，指令平均执行时间：

  每种指令所占的比例 * 该种指令执行所需要的时钟周期数

1.4.3 用指令执行速度进行性能评估

【例1.3 P22】

假设某程序P编译后生成的目标代码由A/B/C/D四类指令组成

它们在程序中所占的比例分别为：43%、21%、12%、24%

已知它们的CPI分别为1/2/2/2，重新对程序P进行编译优化，生成的新目标代码中A类指令条数减少了50%，其他类指令的条数没有变。

(1)编译优化前后程序的CPI各是多少？

(2)假定程序在一台主频为50MHz的计算机上运行，优化前后的MIPS各是多少？

1.4.3 用指令执行速度进行性能评估

【例1.3 P22】

假设某程序P编译后生成的目标代码由A/B/C/D四类指令组成

它们在程序中所占的比例分别为：43%、21%、12%、24%

已知它们的CPI分别为1/2/2/2，重新对程序P进行编译优化，生成的新目标代码中A类指令条数减少了50%，其他类指令的条数没有变。

(1)编译优化前后程序的CPI各是多少？

(2)假定程序在一台主频为50MHz的计算机上运行，优化前后的MIPS各是多少？

解：
优化后A类指令的条数减少了50%，因而各类指令所占比例分别计算如下：
A类指令：21.5 /（21.5+21+12+24） = 27%
B类指令：21 /（21.5+21+12+24）= 27%
C 类指令：12 /（21.5+21+12+24）= 15%
D 类指令：24/（21.5+21+12+24）= 31%

(1)
优化前程序的CPI为： 43% 1 + 21% *2 + 12%*2 + 24%*2 = 1.57
优化后程序的CPI为： 27% 1 + 27% *2 + 15%*2 + 31%*2 = 1.73
(2)
优化前的MIPS 为： 50MHz / 1.57 = 31.8MIPs
优化后的MIPS 为：50MHz / 1.73 =28.9MIPS

从MIPS数来看，优化后的程序执行速度反而变慢了。
这显然是错误的，优化后只减少了A类指令条数而其他指令数没有变，所以程序执行时间一定减少了。
从这个例子可以看出，用MIPS数来衡量性能是不可靠的

用MIPS进行性能评估，有时不准确不客观

优化后，从MIPS数来看，执行速度反而变慢了

但只减少了A类指令条数，其他指令条数未变，程序执行时间一定是减少了

MIPS—执行定点指令的速度，MFLOPS每秒完成浮点操作次数

- MFLOPS——Million FLOating-point operations Per Second　Mflop/s
  基于完成的浮点操作次数衡量

MFLOPS	**M**flop/s	10^6	mega 兆，百万
GFLOPS	**G**flop/s	10^9	giga 吉[咖]，千兆,十亿
TFLOPS	**T**flop/s	10^{12}	tera 太[拉]，万亿
PFLOPS	**P**flop/s	10^{15}	peta 拍
EFLOPS	**E**flop/s	10^{18}	exa 艾
Z			zetta 泽
Y			jotta 尧

Site:	National Super Computer Center in Guangzhou
Manufacturer:	NUDT
Cores:	3,120,000
Linpack Performance (Rmax)	33,862.7 TFlop/s
Theoretical Peak (Rpeak)	54,902.4 TFlop/s
Power:	17,808.00 kW
Memory:	1,024,000 GB
Interconnect:	TH Express-2
Operating System:	Kylin Linux
Compiler:	icc
Math Library:	Intel MKL-11.0.0
MPI:	MPICH2 with a customized GLEX channel

课后练习 P27

1. 概念解释（名词解释）

   前10行 + MIPS

2. 简答

   (1) - (7)

3.

5.

8.

9.

课后练习6.【略】

- 若有两个基准测试程序P1和P2在机器M1和M2上运行，假定M1和M2的价格分别是5000元和8000元，表中给出了P1和P2在机器M1和M2上运行所用的时间和指令条数。

程序	M1		M2	
	指令条数	执行时间	指令条数	执行时间
P1	200×10^6	1000ms	150×10^6	500ms
P2	300×10^3	3ms	420×10^3	6ms

请回答下列问题。

（1）对于 P1，哪台机器的速度快？快多少？对于 P2 呢？

（2）在 M1 上执行 P1 和 P2 的速度分别是多少 MIPS？在 M2 上的执行速度又各是多少？从执行速度来看，对于 P2，哪台机器的速度快？快多少？

（3）假定 M1 和 M2 的时钟频率各是 800MHz 和 1.2GHz，则在 M1 和 M2 上执行 P1 时的平均时钟周期数 CPI 各是多少？

（4）如果某个用户需要大量使用程序 P1，并且该用户主要关心系统的响应时间而不是吞吐率，那么，该用户需要大批购进机器时，应该选择 M1 还是 M2？为什么？（提示：从性价比上考虑）

（5）如果另一个用户也需要购进大批机器，但该用户使用 P1 和 P2 一样多，主要关心的也是响应时间，那么应该选择 M1，还是 M2？为什么？

程序	M1		M2	
	指令条数	执行时间	指令条数	执行时间
P1	200×10^6	1000ms	150×10^6	500ms
P2	300×10^3	3ms	420×10^3	6ms

请回答下列问题。

（1）对于 P1，哪台机器的速度快？快多少？对于 P2 呢？

（2）在 M1 上执行 P1 和 P2 的速度分别是多少 MIPS？在 M2 上的执行速度又各是多少？从执行速度来看，对于 P2，哪台机器的速度快？快多少？

（3）假定 M1 和 M2 的时钟频率各是 800MHz 和 1.2GHz，则在 M1 和 M2 上执行 P1 时的平均时钟周期数 CPI 各是多少？

程序	M1		M2	
	指令条数	执行时间	指令条数	执行时间
P1	200×10^6	1000ms	150×10^6	500ms
P2	300×10^3	3ms	420×10^3	6ms

请回答下列问题。

（1）对于 P1，哪台机器的速度快？快多少？对于 P2 呢？

对于P1，M1所花的执行时间是M2的2倍，M2比M1快1倍；对于程序P2，M2所花的时间是M1的2倍，故M1比M2快1倍

（2）在 M1 上执行 P1 和 P2 的速度分别是多少 MIPS？在 M2 上的执行速度又各是多少？
从执行速度来看，对于 P2，哪台机器的速度快？快多少？

在M1上执行P1的速度为 200M/1s = 200MIPS，P2的速度为 300k / 0.003s = 100 MIPS;

在M2上执行P1的速度为 150M/0.5s = 300MIPS，P2的速度为 420k / 0.006s = 70 MIPS;

从执行速度看，对于P2，100MIPS / 70 MIPS =1.43 M1比M2执行快 0.43倍

（3）假定 M1 和 M2 的时钟频率各是 800MHz 和 1.2GHz，则在 M1 和 M2 上执行 P1 时的平均时钟周期数 CPI 各是多少？

在M1上执行P1，平均每条指令的时钟周期数为 1s * 800 MHz / 200M = 4

在M2上执行P1，平均每条指令的时钟周期数为 0.5s * 1.2 GHz / 150M = 4

程序	M1		M2	
	指令条数	执行时间	指令条数	执行时间
P1	200×10^6	1000ms	150×10^6	500ms
P2	300×10^3	3ms	420×10^3	6ms

（4）如果某个用户需要大量使用程序 P1，并且该用户主要关心系统的响应时间而不是吞吐率，那么，该用户需要大批购进机器时，应该选择 M1 还是 M2？为什么？（提示：从性价比上考虑）

考虑运行P1时M1和M2的性价比，因为该用户关心的系统响应时间，所以性价比中的性能主要考虑执行时间，其性能为执行时间的倒数。故性价比R为 1 / (执行时间 * 价格），R越大说明性价比越高，也即，“执行时间 * 价格 ” 的值越小，性价比越高。

对于程序P1, M1的性价比为 R1 = 1 / (1s * 5000)

M2的性价比为 R2 = 1 / (0.5s * 8000)

R2 > R1 ， M2性价比高，因购买M2

程序	M1		M2	
	指令条数	执行时间	指令条数	执行时间
P1	200×10^6	1000ms	150×10^6	500ms
P2	300×10^3	3ms	420×10^3	6ms

（5）如果另一个用户也需要购进大批机器，但该用户使用 P1 和 P2 一样多，主要关心的也是响应时间，那么应该选择 M1，还是 M2？为什么？

P1 和 P2 需要同等考虑，所以需要考虑综合性能。有多种计算综合性能的方法，如执行时间总和、执行时间算术平均值、执行时间几何平均值等。

若用执行时间总和，则M1的性价比为 R1 = 1 / (1003ms * 5000), M2的性价比为 R2 = 1 / (506ms * 8000) ,R2> R1，选M2

若用算术平均值，执行时间的算术平均值分别为：501.5ms和253ms。因此M1的性价比为 R1 = 1 / (501.5ms * 5000), M2的性价比为 R2 = 1 / (253ms * 8000) ,R2> R1，选M2

若用几何平均值，执行时间的几何平均值为 $\sqrt{}$（$1000 * 3$）= 54.7

因此M1的性价比为 R1 = 1 / (54.7ms * 5000), M2的性价比为 R2 = 1 / (54.7* 8000) ,R1> R2，选M1

几何平均数受极端值的影响较算术平均数小