PA1平时作业

- 姓名：管昀玫
- 学号：2013750
- 专业：计算机科学与技术

问题1

> 用户 CPU 时间与系统响应时间哪个更长?

答:

- 用户 CPU 时间：用户 CPU 时间是指进程在 CPU 上执行代码的时间，包括应用程序代码和用户级别的系统调用。用户 CPU 时间的长短取决于进程执行的代码和计算任务的复杂程度。
- 系统响应时间：系统响应时间是指从发出请求到系统响应完毕的时间，包括用户态和内核态的时间。系统响应时间的长短取决于系统的性能、资源使用情况、任务调度算法等因素。系统响应时间包含了执行时间和等待时间。**系统响应时间大于用户CPU时间。**

如果分时操作系统的时间片一定，那么（用户数越多），则响应时间越长。

假设用户在计算机上启动一个文本编辑器应用程序，计算机需要完成以下操作:

1. 分配内存：计算机需要分配足够的内存空间给应用程序使用。
2. 加载程序：计算机需要加载应用程序代码到内存中，以便执行。
3. 启动应用程序：计算机需要执行应用程序代码，启动应用程序。
4. 响应用户输入：当用户在文本编辑器中输入文本时，计算机需要响应用户输入，并在屏幕上显示文本。

用户 CPU 时间是指计算机在执行应用程序代码时使用的 CPU 时间。例如，当文本编辑器应用程序正在处理用户输入时，计算机正在使用用户 CPU 时间来执行应用程序代码。

系统响应时间是指计算机响应用户请求所需的时间。例如，在启动文本编辑器应用程序时，计算机需要分配内存、加载程序代码、执行应用程序代码等操作。这些操作所需的时间都会被计算在系统响应时间中。

问题2

> 单靠CPI不能反映CPU的性能，为什么?

CPI（Cycles Per Instruction，每指令周期数）是衡量CPU性能的一个指标，它表示执行一条指令所需的CPU周期数。其具体计算公式为:

$$CPI = (CPU\text{时间} \times \text{时钟频率}) / \text{指令条数} = \text{总时钟周期数} / \text{指令条数}$$

然而，仅仅使用CPI并不能完全反映CPU的性能，原因如下:

1. 指令集不同：不同的CPU可能使用不同的指令集，即使两种CPU的CPI相同，它们的性能也可能会有所不同。这是因为不同的指令集需要不同的CPU架构和硬件支持。
2. 等效指令数不同：在不同的CPU架构中，执行同一条高级语言代码的机器码指令数可能不同，因此，在不同的CPU上执行同一份代码所需的CPU周期数也可能不同。
3. 并行执行能力不同：一些CPU具有较强的并行执行能力，能够同时执行多条指令，从而提高了CPU的性能。然而，这种并行执行能力无法通过CPI来衡量。

最简单的例子为：单周期处理器CPI=1，但性能差。

问题3

> 1.57是怎么算出来的?

$$\frac{43\% * 1 + 21\% * 2 + 12\% * 2 + 24\% * 2}{1} = 1.57$$

问题4

> 西文字符有无编码?

西文字符有编码。

目前最常用的字符编码包括ASCII、Unicode和UTF-8。

ASCII（American Standard Code for Information Interchange）是一种最早的西文字符编码，它使用7位二进制数来表示128个字符，包括英文大小写字母、数字、标点符号和一些特殊字符。

Unicode是一种更加全面的字符编码，它可以表示世界上大部分语言所使用的字符，包括汉字、日文、希腊字母等等。Unicode可以使用不同的编码方案来表示字符，包括UTF-8、UTF-16和UTF-32等。其中，UTF-8是最常用的编码方案，它可以使用1至4个字节来表示一个字符，具有兼容性好、节省空间等优点。

问题5

> 为什么同一个实数赋值给float型变量和double型变量，输出结果会有所不同?

答:

这是因为浮点数的存储方式是采用科学计数法，用一个数的指数和尾数来表示这个数。在计算机中，float和double类型分别使用32位和64位的存储空间，因此能够表示的数值范围和精度不同。对于例子中的float a=123456.789e4和double b=123456.789e4，它们的值都是1234567890，表示的都是科学计数法下的1.23456789 × 10^9，但是它们存储的方式不同，因此输出结果不同。

这是一个由于存储方式不同而引起的根本性原因，理由如下:

float型

在计算机内部，float类型使用的是IEEE 754标准的单精度浮点数格式，它的存储空间是32位，其中1位表示符号位，8位表示指数位，23位表示尾数位。在浮点数的存储中，指数位和尾数位的组合表示了一个实数，因此float类型能够表示的实数范围是有限的。

1. float a=123456.789e4是科学计数法表示的一个实数，即1.23456789 × 10^9。
2. 根据IEEE 754标准，对于单精度浮点数，指数位偏移量是127，即指数值的实际值为指数值减去127。因此，将1.23456789 × 10^9转换为单精度浮点数时，需要将指数位调整为二进制的指数值加上127，即:
    - 指数位 = 9 + 127 = 136 = 10001000（二进制）
3. 由于单精度浮点数的尾数位只有23位，因此需要对尾数位进行舍入。对于1.23456789，它的二进制表示是：1.00111100011101011110111000010101（二进制）
    - 这个二进制数最后一位是1，需要向上舍入，得到：1.00111100011101011110111000010110（二进制）
4. 将符号位、指数位和尾数位组合起来，得到最终的32位二进制数：0 10001000 00111100011101011110111 00001011
    - 其中，0表示正数，10001000表示指数位，00111100011101011110111000010110表示尾数位。

5. 将32位二进制数转换为十进制数，得到：1.23456794 × 10^9

使用浮点数进行运算或打印时，计算机会根据浮点数的精度和有效位数对其进行四舍五入和截断。因此，对于浮点数1.23456794 × 10^9，它被截断为被截断为1.234568 × 10^9

同时，由于浮点数在内部表示时使用的是二进制，而我们通常使用的是十进制，因此浮点数在输出时可能会被转换为十进制形式。这个过程可能会导致进一步的舍入误差。在这个例子中，1.234568 × 10^9被转换为1234567936。这个结果也是四舍五入后的结果，与原始浮点数1.23456794 × 10^9有一定的误差。

double型

对于double b=123456.789e4，它被赋值为1.23456789 × 10^9，因为double类型能够精确表示约15-16位有效数字，因此123456.789e4的所有有效数字都能够被表示，最后输出时就是1.23456789 × 10^9，即1234567890。