课上思考题-1

- 姓名：陈睿颖
- 学号：2013544
- 专业：计算机科学与技术

课上思考题

> 用户CPU时间与系统响应时间哪个更长?

用户 CPU 时间是指在 CPU 上执行用户程序的时间，包括用户程序执行系统调用的时间。而系统响应时间是指从请求发出到响应返回的总时间，包括用户等待 CPU 时间、I/O 等待时间、内核运行时间等。

一般来说，用户 CPU 时间与系统响应时间是相互独立的，不存在谁一定更长的情况。不同的操作系统、应用程序、硬件环境等都是影响因素。

举个例子，在一个负载较轻的系统中，用户 CPU 时间可能会更长，因为 CPU 可能有更多的空闲时间来处理用户程序的执行。但在一个负载较重的系统中，系统响应时间可能会更长，因为 CPU 和其他资源都需要在多个进程和系统调用之间进行切换。

因此，二者的大小关系并不确定。

> 指令的CPI、机器的CPI、程序的CPI各能反映哪方面的性能？单靠CPI不能反映CPU性能，为什么？

指令的CPI是指在执行一个指令时，CPU需要的时钟周期数。机器的CPI是指在执行一条机器指令时，CPU需要的时钟周期数。程序的CPI是指在执行整个程序时，CPU需要的时钟周期数。

指令的CPI能够反映CPU指令级并行性的表现，即在执行指令时CPU是否能够充分利用硬件资源，如寄存器、缓存等，从而提高CPU的性能。

机器的CPI则能够反映CPU微架构的性能，即CPU在执行机器指令时能够充分利用硬件资源的效率，如流水线、乱序执行、分支预测等。

程序的CPI则反映的是整个程序在CPU上运行的性能，包括指令级并行性和微架构的性能。程序的CPI会受到指令级并行性、微架构、内存访问等因素的影响。

单靠CPI不能反映CPU性能的原因在于，CPI只是反映了在单条指令或机器指令级别上的性能表现，而现代CPU的性能不仅仅取决于单条指令的执行速度，还取决于指令级并行性、微架构的设计、内存访问速度等多个因素的综合表现。因此，单靠CPI无法全面地反映CPU的性能表现

> Assume we build an optimizing compiler for the load/store machine. The compiler discards 50% of the ALU instructions.
>
> 1. What is the CPI ?
> 2. Assuming a 20 ns clock cycle time (50 MHz clock rate). What is the MIPSrating for optimized code versus unoptimized code? Does the MIPS rating agree with the rating of execution time?

1. 优化前：$CPI = 0.43 \times 1 + 0.21 \times 2 + 0.12 \times 2 + 0.24 \times 2 = 1.57$

   优化后：$CPI = 0.215 \times 1 + (1 - 0.215) \times 2 = 1.73$
2. 假设时钟周期时间为20ns（时钟频率为50MHz），优化后的代码与未优化的代码相比，MIPS评级会有所不同。MIPS评级是指每秒可以执行的百万条指令数。未经优化的代码的MIPS评级为 $50 \div 1.57 = 31.85MIPS$。而优化后的代码中，由于舍弃了一半的ALU指令，因此可以通过更多地使用Load和Store指令来优化程序，使得平均指令执行周期降低到1.73个时钟周期，因此MIPS评级为 $50 \div 1.73 = 28.90MIPS$

> 西文字符有编码吗？

有。在计算机中，每个字符都用一个数字编码来表示，这被称为字符编码。 ASCII码是最常见的字符编码，它将每个字符映射到一个唯一的数字值。例如，字母"A"被映射到数字65，字母"Z"被映射到数字90。

然而，ASCII码只包含128个字符，无法表示其他字符集，例如特殊字符、重音符号和其他语言的字符。因此，为了解决这个问题，出现了更多的字符编码方案，例如ISO-8859和Unicode。Unicode是目前最常用的字符编码，可以表示几乎所有的字符集，包括西文字符和其他语言的字符。

例:将同一实数分别赋值给单精度和双精度类型变量，然后打印输出。

```
#include <stdio.h>
main()
{
float a;
double b;
a = 123456.789e4;
b = 123456.789e4; printf(“%f/n%f/n”,a,b);
}
```

运行结果如下:

1234567936.000000 1234567890.000000

为什么float情况下输出的结果会比原来的大?这到底有没有根本性原因还是随机发生的? 为什么会出现这样的情况?

问题: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呢?

当将实数123456.789e4赋值给float类型的变量a和double类型的变量b时，会发生精度损失，因为float类型只有4个字节（32位），而double类型有8个字节（64位）。在float类型中，有效数字部分只有23位，而在double类型中，有效数字部分有53位。

因此，当将123456.789e4转换为float类型时，它被近似为最接近123456.789e4的32位浮点数1234567936，但在将它转换为double类型时，它被近似为最接近123456.789e4的64位浮点数1234567890，所以float情况下输出的结果会比原来的大。这是由于浮点数在计算机中表示的精度限制所导致的，而不是随机发生的。