1. 假定机器M的时钟频率为1.2GHz，某程序P在机器M上的执行时间为12秒钟。对P优化时，将其所有的乘4指令都换成了一条左移2位的指令，得到优化后的程序P’。已知在M上乘法指令的CPI为5，左移指令的CPI为2，P的执行时间是P’执行时间的1.2倍，则P中有多少条乘法指令被替换成了左移指令被执行？

解：设中有条乘法指令被替换为左移指令，则由题目，的执行时间是秒钟，减少的时间来自对乘法指令的优化，根据，可以得到：



解得：。

1. 图形处理器中经常需要的一种转换是求平方根。浮点（FP）平方根的实现在性能方面有很大差异，特别是在为图形设计的处理器中，尤为明显。假设FP平方根（FPSQR）占用一项关键图形基准测试中20%的执行时间。有一项提议：升级FPSQR硬件，使这一运算速度提高到原来的10倍。另一项提议是让图形处理器中所有FP指令的运行速度提高到原来的1.6倍，FP指令占用该应用程序一半的执行时间。设计团队相信，他们使所有FP指令执行速度提高到1.6倍所需要的工作量与加快平方根运算的工作量相同。试比较这两种设计方案。

解：由Amdahl定律，可以分别计算两种方案的加速比。

第一种方案的加速比为：



第二种方案的加速比为：



所以从系统加速比来看，第二种方案更好。

1. 若有两个基准测试程序P1和P2在机器M1和M2上运行，下表给出了P1和P2在M1和M2上所花的时间和指令条数。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 程序 | M1 | | M2 | |
| 指令条数 | 执行时间(ms) | 指令条数 | 执行时间(ms) |
| P1 | 200×106 | 10000 | 150×106 | 5000 |
| P2 | 300×103 | 3 | 420×103 | 6 |

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？

解：对于，的速度更快，快ms；对于，的速度更快，快ms。

在上执行的速度是：



在上执行的速度是：



在上执行的速度是：



在上执行的速度是：



从来看，对于，的执行速度更快，快。

由，在上执行时的为：



在上执行时的为：



1. 假设我们在对有符号值使用补码运算的32位机器上运行代码。对于有符号值使用的是算术右移，对无符号值使用的是逻辑右移。变量的声明和初始化如下：

int x = foo(); //调用某某函数，给x赋值

int y = bar(); //调用某某函数，给y赋值

unsigned ux = x;

unsigned uy = y;

对于下面每个C表达式

证明对于所有的x和y 值，都为真（等于1）；或者（2）给出使得它为假的x和y值；

1. (x>0) || (x-1<0)
2. (x&7) != 7 || (x<<29<0)
3. (x \* x) >= 0
4. x<0 || -x<=0
5. x>0 || -x>=0
6. x+y == uy+ux
7. x\*~y + uy\*ux == -x
8. x\*4 + y\*8 == (x<<2)+(y<<3)
9. ((x>>2)<<2)<=x

解：不是永真的，因为可能发生溢出，比如当x = 0x80000000时。

为永真，因为7=0x00000007，如果(x&7) == 7，那么x表示为2进制数时的后四位一定是0111，前面28位可以是任意合法的0，1组合，此时(x<<29 < 0)必为真，因为(x<<29)的符号位是1；如果x表示为2进制的最后四位数字不是0111,那么(x&7) != 7已经为真，故永真。

非永真，因为可能发生溢出，比如当x = 0x0000ffff

永真，因为32位有符号整数表示的范围是，当x < 0时，为真，当x >= 0时，-x不会发生溢出，所以也为真，故永真。

非永真，比如当x = 0x80000000时。

非永真，比如当x = -1，y = 1时。

永真，因为~y = -y – 1,而有符号和无符号整数的乘法可以共用一套算法，故uy\*ux = x \* y，带入之后可知永真。

永真，因为计算机内部对于整数乘以2的幂次的操作就是使用左移操作完成的。

永真，因为对于有符号整数采用算术右移，右移之后在最高位填符号位上的数字，而左移不分算术和逻辑，都是在最低位上填0，因此只要x表示为2进制数时最后两位不是00，那么((x>>2)<<2)<x，如果是00，那么((x>>2)<<2)==x，所以永真。

1. 以下是两段C语言程序代码，函数arith()是直接用C语言写的，而optarith()是对arith()函数以某个确定的M和N编译生成的机器代码反编译生成的。根据optarith()，推断函数arith()中的M和N的值各是多少？

#define M

#define N

int arith(int x, int y)

{

int result = 0;

result = x\*M+y/N;

return result;

}

int optarith(int x, int y)

{

int t = x;

x << = 4;

x -= t;

if(y < 0) y+=3;

y>>=2;

return x+y;

}

解：由于 x << 4 = 16 \* x，故，而。

1. 假定在一个程序中定义了变量x、y和i，其中，x和y是float型变量（用IEEE754单精度浮点数表示），i是16位short型变量（用补码表示）。程序执行到某一时刻，x = –0.125、y=7.5、i=100，它们都被写到了主存（按字节编址），其地址分别是100，108和112。请分别画出在大端机器和小端机器上变量x、y和i在内存的存放位置。

解：在大端机器上：

x 的表示如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 100 | 101 | 102 | 103 |
| BE | 00 | 00 | 00 |

y的表示如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 108 | 109 | 110 | 111 |
| 41 | 70 | 00 | 00 |

i的表示如下：

|  |  |
| --- | --- |
| 112 | 113 |
| 00 | C4 |

在小端机器上：

x 的表示如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 100 | 101 | 102 | 103 |
| 00 | 00 | 00 | BE |

y的表示如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 108 | 109 | 110 | 111 |
| 00 | 00 | 70 | 41 |

i的表示如下：

|  |  |
| --- | --- |
| 112 | 113 |
| C4 | 00 |

1. We are running programs on a machine with the following characteristics:

* Values of type int are 32 bits. They are represented in two’s complement, and they are right shifted arithmetically. Values of type unsigned are 32 bits.
* Values of type float are represented using the 32-bit IEEE ﬂoating point format, while values of type double use the 64-bit IEEE ﬂoating point format.
* We generate arbitrary values x, y, and z, and convert them to other forms as follows:

/\* Create some arbitrary values \*/

int x = random();

int y = random();

int z = random();

/\* Convert to other forms \*/

unsigned ux = (unsigned) x;

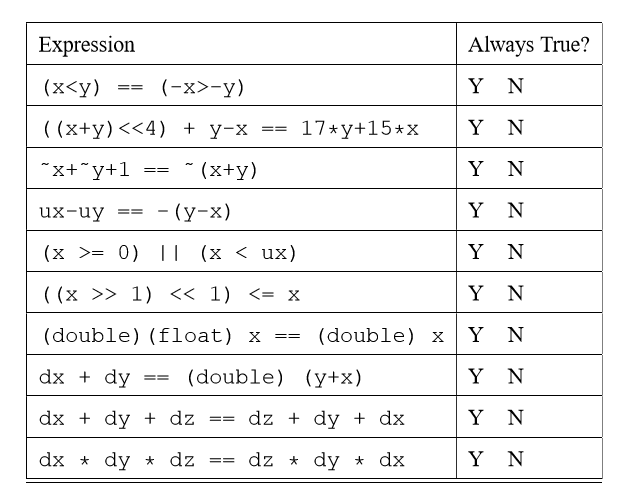
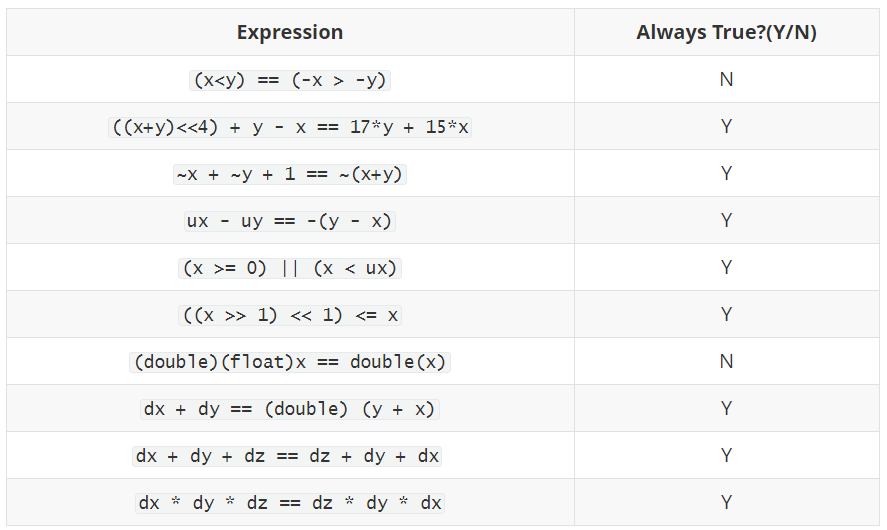
unsigned uy = (unsigned) y;

double dx = (double) x;

double dy = (double) y;

double dz = (double) z;

For each of the following C expressions, you are to indicate whether or not the expression always yields 1. If so, circle “Y”. If not, circle “N” and tell why.

解：如上图，第一个式子是错误的，反例比如：x = 0x80000000，y = 0x00000000。

第七个式子是错误的，因为将int转换为float时，尾数的位数可能超过23，导致舍入，再转换为double后与原来的int取值不等，但是直接转换为double，可以确保转换之后的数值与原来数值相等。