相比较。

** 2.93　遵循位级浮点编码规则，实现具有如下原型的函数：

```
/* Compute |f|.  If f is NaN, then return f. */
float_bits float_absval(float_bits f);
```

对于浮点数 f，这个函数计算 $|f|$。如果 f 是 NaN，你的函数应该简单地返回 f。

测试你的函数，对参数 f 可以取的所有 2^{32} 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。

** 2.94　遵循位级浮点编码规则，实现具有如下原型的函数：

```
/* Compute 2*f.  If f is NaN, then return f. */
float_bits float_twice(float_bits f);
```

对于浮点数 f，这个函数计算 $2.0 \cdot f$。如果 f 是 NaN，你的函数应该简单地返回 f。

测试你的函数，对参数 f 可以取的所有 2^{32} 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。

** 2.95　遵循位级浮点编码规则，实现具有如下原型的函数：

```
/* Compute 0.5*f.  If f is NaN, then return f. */
float_bits float_half(float_bits f);
```

对于浮点数 f，这个函数计算 $0.5 \cdot f$。如果 f 是 NaN，你的函数应该简单地返回 f。

测试你的函数，对参数 f 可以取的所有 2^{32} 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。

** 2.96　遵循位级浮点编码规则，实现具有如下原型的函数：

```
/*
 * Compute (int) f.
 * If conversion causes overflow or f is NaN, return 0x80000000
 */
int float_f2i(float_bits f);
```

对于浮点数 f，这个函数计算(int)f。如果 f 是 NaN，你的函数应该向零舍入。如果 f 不能用整数表示(例如，超出表示范围，或者它是一个 NaN)，那么函数应该返回 0x80000000。

测试你的函数，对参数 f 可以取的所有 2^{32} 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。

** 2.97　遵循位级浮点编码规则，实现具有如下原型的函数：

```
/* Compute (float) i */
float_bits float_i2f(int i);
```

对于函数 i，这个函数计算(float) i 的位级表示。

测试你的函数，对参数 f 可以取的所有 2^{32} 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。

练习题答案

2.1　在我们开始查看机器级程序的时候，理解十六进制和二进制格式之间的关系将是很重要的。虽然本书中介绍了完成这些转换的方法，但是做点练习能够让你更加熟练。

A. 将 0x39A7F8 转换成二进制：

十六进制	3	9	A	7	F	8
二进制	0011	1001	1010	0111	1111	1000

B. 将二进制 1100100101111011 转换成十六进制：

二进制	1100	1001	0111	1011
十六进制	C	9	7	B