3.52 映射参数到寄存器的基本规则非常简单(虽然随着有更多类型的参数出现，这些规则也变得越来越复杂[77])。

A. `double g1(double a, long b, float c, int d);`

寄存器：a 在%xmm0 中，b 在%rdi 中，c 在%xmm1 中，d 在% esi 中

B. `double g2(int a, double *b, float *c, long d);`

寄存器：a 在%edi 中，b 在%rsi 中，c 在%rdx 中，d 在%rcx 中

C. `double g3(double *a, double b, int c, float d);`

寄存器：a 在%rdi 中，b 在%xmm0 中，c 在% esi 中，d 在%xmm1 中

D. `double g4(float a, int *b, float c, double d);`

寄存器：a 在%xmm0 中，b 在%rdi 中，c 在%xmm1 中，d 在%xmm2 中

3.53 从这段汇编代码可以看出有两个整数参数，通过寄存器%rdi 和%rsi 传递，将其命名为 i1 和 i2。类似地，有两个浮点参数，通过寄存器%xmm0 和%xmm1 传递，将其命名为 f1 和 f2。

然后给汇编代码加注释：

```
    Refer to arguments as i1 (%rdi), i2 (%esi)
                          f1 (%xmm0), and f2 (%xmm1)

    double funct1(arg1_t p, arg2_t q, arg3_t r, arg4_t s)
1   funct1:
2     vcvtsi2ssq      %rsi, %xmm2, %xmm2    Get i2 and convert from long to float
3     vaddss  %xmm0, %xmm2, %xmm0           Add f1 (type float)
4     vcvtsi2ss       %edi, %xmm2, %xmm2    Get i1 and convert from int to float
5     vdivss  %xmm0, %xmm2, %xmm0           Compute i1 / (i2 + f1)
6     vunpcklps       %xmm0, %xmm0, %xmm0
7     vcvtps2pd       %xmm0, %xmm0          Convert to double
8     vsubsd  %xmm1, %xmm0, %xmm0           Compute i1 / (i2 + f1) - f2 (double)
9     ret
```

由此可以看出这段代码计算值 i1/(i2+f1)-f2。还可以看到，i1 的类型为 int，i2 的类型为 long，f1 的类型为 float，而 f2 的类型为 double。将参数匹配到命名的值只有一个不确定的地方，来自于加法的交换性——得到两种可能的结果：

```
double funct1a(int p, float q, long r, double s);
double funct1b(int p, long q, float r, double s);
```

3.54 一步步梳理汇编代码，确定每一步计算什么，就很容易找到这道题的答案，如下面的注释所示：

```
    double funct2(double w, int x, float y, long z)
    w in %xmm0, x in %edi, y in %xmm1, z in %rsi
1   funct2:
2     vcvtsi2ss       %edi, %xmm2, %xmm2    Convert x to float
3     vmulss  %xmm1, %xmm2, %xmm1           Multiply by y
4     vunpcklps       %xmm1, %xmm1, %xmm1
5     vcvtps2pd       %xmm1, %xmm2          Convert x*y to double
6     vcvtsi2sdq      %rsi, %xmm1, %xmm1    Convert z to double
7     vdivsd  %xmm1, %xmm0, %xmm0           Compute w/z
8     vsubsd  %xmm0, %xmm2, %xmm0           Subtract from x*y
9     ret                                   Return
```

可以从分析得出结论，该函数计算 y*x-w/z。

3.55 这道题使用的推理与推断标号.LC2 处声明的数字是 1.8 的编码一样，不过例子更简单。

我们看到两个值分别是 0 和 1077936128(0x40400000)。从高位字节可以抽取出指数字段 0x404(1028)，减去偏移量 1023 得到指数为 5。连接两个值的小数位，得到小数字段为 0，加上隐含的开头的 1，得到 1.0。因此这个常数是 $1.0\times 2^5=32.0$。

3.56 A. 在此可以看到从地址.LC1 开始的 16 个字节是一个掩码，它的低 8 个字节是全 1，除了最高位，这是双精度值的符号位。计算这个掩码和%xmm0 的 AND 值时，会清除 x 的符号位，得到绝对