```
1  /*
2   * linux/kernel/math/compare.c
3   *
4   * (C) 1991 Linus Torvalds
5   */
6
7  /*
8   * temporary real comparison routines
9   */
   /*
    * 累加器中临时实数比较子程序。
    */
10
11 #include <linux/math_emu.h> // 协处理器头文件。定义临时实数结构和 387 寄存器操作宏等。
12
   // 复位状态字中的 C3、C2、C1 和 C0 条件位。
13 #define clear_Cx() (I387.swd &= ~0x4500)
14
   // 对临时实数 a 进行规格化处理。即表示成指数、有效数形式。
   // 例如：102.345 表示成 1.02345 X 10^2。 0.0001234 表示成 1.234 X 10^-4。当然，函数中是
   // 二进制表示。
15 static void normalize(temp_real * a)
16 {
17         int i = a->exponent & 0x7fff;           // 取指数值（略去符号位）。
18         int sign = a->exponent & 0x8000;        // 取符号位。
19
   // 如果临时实数 a 的 64 位有效数（尾数）为 0，那么说明 a 等于 0。于是清 a 的指数，并返回。
20         if (!(a->a || a->b)) {
21                 a->exponent = 0;
22                 return;
23         }
   // 如果 a 的尾数最左端有 0 值比特位，那么将尾数左移，同时调整指数值（递减）。直到尾数
   // 的 b 字段最高有效位 MSB 是 1 位置（此时 b 表现为负值）。最后再添加上符号位。
24         while (i && a->b >= 0) {
25                 i--;
26                 __asm__("addl %0,%0 ; adcl %1,%1"
27                         :"=r" (a->a),"=r" (a->b)
28                         :"" (a->a),"1" (a->b));
29         }
30         a->exponent = i | sign;
31 }
32
   // 仿真浮点指令 FTST。
   // 即栈定累加器 ST(0)与 0 比较，并根据比较结果设置条件位。若 ST > 0.0，则 C3，C2，C0
   // 分别为 000；若 ST < 0.0，则条件位为 001；若 ST == 0.0，则条件位是 100；若不可比较，
   // 则条件位为 111。
33 void ftst(const temp_real * a)
34 {
35         temp_real b;
36
   // 首先清状态字中条件标志位，并对比较值 b（ST）进行规格化处理。若 b 不等于零并且设置
   // 了符号位（是负数），则设置条件位 C0。否则设置条件位 C3。
```

```
37          clear_Cx();
38          b = *a;
39          normalize(&b);
40          if (b.a || b.b || b.exponent) {
41                  if (b.exponent < 0)
42                          set_C0();
43          } else
44                  set_C3();
45  }
46
    // 仿真浮点指令 FCOM。
    // 比较两个参数 src1、src2。并根据比较结果设置条件位。若 src1 > src2，则 C3，C2，C0
    // 分别为 000；若 src1 < src2，则条件位为 001；若两者相等，则条件位是 100。
47  void fcom(const temp_real * src1, const temp_real * src2)
48  {
49          temp_real a;
50
51          a = *src1;
52          a.exponent ^= 0x8000;                   // 符号位取反。
53          fadd(&a,src2,&a);                       // 两者相加（即相减）。
54          ftst(&a);                               // 测试结果并设置条件位。
55  }
56
    // 仿真浮点指令 FUCOM（无次序比较）。
    // 用于操作数之一是 NaN 的比较。
57  void fucom(const temp_real * src1, const temp_real * src2)
58  {
59          fcom(src1,src2);
60  }
61
```