```
1  /*
2   * linux/kernel/math/div.c
3   *
4   * (C) 1991 Linus Torvalds
5   */
6
7  /*
8   * temporary real division routine.
9   */
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
   // 将指针 c 指向的 4 字节中内容左移 1 位。
13 static void shift_left(int * c)
14 {
15         __asm__ __volatile__("movl (%0),%%eax ; addl %%eax,(%0)\n\t"
16                 "movl 4(%0),%%eax ; adcl %%eax,4(%0)\n\t"
17                 "movl 8(%0),%%eax ; adcl %%eax,8(%0)\n\t"
18                 "movl 12(%0),%%eax ; adcl %%eax,12(%0)"
19                 ::"r" ((long) c):"ax");
20 }
21
   // 将指针 c 指向的 4 字节中内容右移 1 位。
22 static void shift_right(int * c)
23 {
24         __asm__("shrl $1,12(%0) ; rcrl $1,8(%0) ; rcrl $1,4(%0) ; rcrl $1,(%0)"
25                 ::"r" ((long) c));
26 }
27
   // 减法运算。
   // 16 字节减法运算，a - b ➔ a。最后根据是否有借位（CF=1）设置 ok。若无借位（CF=0）
   // 则 ok = 1。否则 ok=0。
28 static int try_sub(int * a, int * b)
29 {
30         char ok;
31
32         __asm__ __volatile__("movl (%1),%%eax ; subl %%eax,(%2)\n\t"
33                 "movl 4(%1),%%eax ; sbbl %%eax,4(%2)\n\t"
34                 "movl 8(%1),%%eax ; sbbl %%eax,8(%2)\n\t"
35                 "movl 12(%1),%%eax ; sbbl %%eax,12(%2)\n\t"
36                 "setae %%al":"=a" (ok):"c" ((long) a),"d" ((long) b));
37         return ok;
38 }
39
   // 16 字节除法。
   // 参数 a /b ➔ c。利用减法模拟多字节除法。
40 static void div64(int * a, int * b, int * c)
41 {
42         int tmp[4];
43         int i;
44         unsigned int mask = 0;
45
```

```
46          c += 4;
47          for (i = 0 ; i<64 ; i++) {
48                  if (!(mask >>= 1)) {
49                          c--;
50                          mask = 0x80000000;
51                  }
52                  tmp[0] = a[0]; tmp[1] = a[1];
53                  tmp[2] = a[2]; tmp[3] = a[3];
54                  if (try_sub(b,tmp)) {
55                          *c |= mask;
56                          a[0] = tmp[0]; a[1] = tmp[1];
57                          a[2] = tmp[2]; a[3] = tmp[3];
58                  }
59                  shift_right(b);
60          }
61  }
62
    // 仿真浮点指令 FDIV。
63  void fdiv(const temp_real * src1, const temp_real * src2, temp_real * result)
64  {
65          int i,sign;
66          int a[4],b[4],tmp[4] = {0,0,0,0};
67
68          sign = (src1->exponent ^ src2->exponent) & 0x8000;
69          if (!(src2->a || src2->b)) {
70                  set_ZE();
71                  return;
72          }
73          i = (src1->exponent & 0x7fff) - (src2->exponent & 0x7fff) + 16383;
74          if (i<0) {
75                  set_UE();
76                  result->exponent = sign;
77                  result->a = result->b = 0;
78                  return;
79          }
80          a[0] = a[1] = 0;
81          a[2] = src1->a;
82          a[3] = src1->b;
83          b[0] = b[1] = 0;
84          b[2] = src2->a;
85          b[3] = src2->b;
86          while (b[3] >= 0) {
87                  i++;
88                  shift_left(b);
89          }
90          div64(a,b,tmp);
91          if (tmp[0] || tmp[1] || tmp[2] || tmp[3]) {
92                  while (i && tmp[3] >= 0) {
93                          i--;
94                          shift_left(tmp);
95                  }
96                  if (tmp[3] >= 0)
97                          set_DE();
```

```
98          } else
99                  i = 0;
100         if (i>0x7fff) {
101                 set_OE();
102                 return;
103         }
104         if (tmp[0] || tmp[1])
105                 set_PE();
106         result->exponent = i | sign;
107         result->a = tmp[2];
108         result->b = tmp[3];
109 }
110
```