```
/*
 * linux/kernel/math/get_put.c
 *
 * (C) 1991 Linus Torvalds
 */

/*
 * This file handles all accesses to user memory: getting and putting
 * ints/reals/BCD etc. This is the only part that concerns itself with
 * other than temporary real format. All other cals are strictly temp_real.
 */
/*
 * 本程序处理所有对用户内存的访问：获取和存入指令/实数值/BCD 数值等。这是
 * 涉及临时实数以外其他格式仅有的部分。所有其他运算全都使用临时实数格式。
 */
#include <signal.h>        // 信号头文件。定义信号符号，信号结构及信号操作函数原型。

#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。
#include <linux/kernel.h>   // 内核头文件。含有一些内核常用函数的原形定义。
#include <asm/segment.h>    // 段操作头文件。定义了有关段寄存器操作的嵌入式汇编函数。

// 取用户内存中的短实数（单精度实数）。
// 根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，取得短实数
// 所在有效地址（math/ea.c），然后从用户数据区读取相应实数值。最后把用户短实数转换成
// 临时实数（math/convert.c）。
// 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
void get_short_real(temp_real * tmp,
        struct info * info, unsigned short code)
{
        char * addr;
        short_real sr;

        addr = ea(info,code);                           // 计算有效地址。
        sr = get_fs_long((unsigned long *) addr);       // 取用户数据区中的值。
        short_to_temp(&sr,tmp);                         // 转换成临时实数格式。
}

// 取用户内存中的长实数（双精度实数）。
// 首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，取得长
// 实数所在有效地址（math/ea.c），然后从用户数据区读取相应实数值。最后把用户实数值转
// 换成临时实数（math/convert.c）。
// 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
void get_long_real(temp_real * tmp,
        struct info * info, unsigned short code)
{
        char * addr;
        long_real lr;

        addr = ea(info,code);                               // 取指令中的有效地址值。
        lr.a = get_fs_long((unsigned long *) addr);         // 取长 8 字节实数。
        lr.b = get_fs_long(1 + (unsigned long *) addr);
        long_to_temp(&lr,tmp);                              // 转换成临时实数格式。
```

```
39 }
40
   // 取用户内存中的临时实数。
   // 首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，取得临
   // 时实数所在有效地址（math/ea.c），然后从用户数据区读取相应临时实数值。
   // 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
41 void get_temp_real(temp_real * tmp,
42         struct info * info, unsigned short code)
43 {
44         char * addr;
45
46         addr = ea(info,code);                              // 取指令中的有效地址值。
47         tmp->a = get_fs_long((unsigned long *) addr);
48         tmp->b = get_fs_long(1 + (unsigned long *) addr);
49         tmp->exponent = get_fs_word(4 + (unsigned short *) addr);
50 }
51
   // 取用户内存中的短整数并转换成临时实数格式。
   // 临时整数也用 10 字节表示。其中低 8 字节是无符号整数值，高 2 字节表示指数值和符号位。
   // 如果高 2 字节最高有效位为 1，则表示是负数；若最高有效位是 0，表示是正数。
   // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
   // 取得短整数所在有效地址（math/ea.c），然后从用户数据区读取相应整数值，并保存为临时
   // 整数格式。最后把临时整数值转换成临时实数（math/convert.c）。
   // 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
52 void get_short_int(temp_real * tmp,
53         struct info * info, unsigned short code)
54 {
55         char * addr;
56         temp_int ti;
57
58         addr = ea(info,code);                  // 取指令中的有效地址值。
59         ti.a = (signed short) get_fs_word((unsigned short *) addr);
60         ti.b = 0;
61         if (ti.sign = (ti.a < 0))              // 若是负数，则设置临时整数符号位。
62                 ti.a = - ti.a;                 // 临时整数“尾数”部分为无符号数。
63         int_to_real(&ti,tmp);                  // 把临时整数转换成临时实数格式。
64 }
65
   // 取用户内存中的长整数并转换成临时实数格式。
   // 首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，取得长
   // 整数所在有效地址（math/ea.c），然后从用户数据区读取相应整数值，并保存为临时整数格
   // 式。最后把临时整数值转换成临时实数（math/convert.c）。
   // 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
66 void get_long_int(temp_real * tmp,
67         struct info * info, unsigned short code)
68 {
69         char * addr;
70         temp_int ti;
71
72         addr = ea(info,code);                  // 取指令中的有效地址值。
73         ti.a = get_fs_long((unsigned long *) addr);
74         ti.b = 0;
75         if (ti.sign = (ti.a < 0))              // 若是负数，则设置临时整数符号位。
```

```
76                  ti.a = - ti.a;                  // 临时整数“尾数”部分为无符号数。
77          int_to_real(&ti,tmp);                   // 把临时整数转换成临时实数格式。
78  }
79
    // 取用户内存中的 64 位长整数并转换成临时实数格式。
    // 首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，取得
    // 64 位长整数所在有效地址（math/ea.c），然后从用户数据区读取相应整数值，并保存为临
    // 时整数格式。最后再把临时整数值转换成临时实数（math/convert.c）。
    // 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
80  void get_longlong_int(temp_real * tmp,
81          struct info * info, unsigned short code)
82  {
83          char * addr;
84          temp_int ti;
85
86          addr = ea(info,code);                           // 取指令中的有效地址值。
87          ti.a = get_fs_long((unsigned long *) addr);     // 取用户 64 位长整数。
88          ti.b = get_fs_long(1 + (unsigned long *) addr);
89          if (ti.sign = (ti.b < 0))                   // 若是负数则设置临时整数符号位。
90                  __asm__("notl %0 ; notl %1\n\t"     // 同时取反加 1 和进位调整。
91                          "addl $1,%0 ; adcl $0,%1"
92                          :"=r" (ti.a),"=r" (ti.b)
93                          :"" (ti.a),"1" (ti.b));
94          int_to_real(&ti,tmp);                       // 把临时整数转换成临时实数格式。
95  }
96
    // 将一个 64 位整数（例如 N）乘 10。
    // 这个宏用于下面 BCD 码数值转换成临时实数格式过程中。方法是：N<<1 + N<<3。
97  #define MUL10(low,high) \
98  __asm__("addl %0,%0 ; adcl %1,%1\n\t" \
99  "movl %0,%%ecx ; movl %1,%%ebx\n\t" \
100 "addl %0,%0 ; adcl %1,%1\n\t" \
101 "addl %0,%0 ; adcl %1,%1\n\t" \
102 "addl %%ecx,%0 ; adcl %%ebx,%1" \
103 :"=a" (low),"=d" (high) \
104 :"" (low),"1" (high):"cx","bx")
105
    // 64 位加法。
    // 把 32 位的无符号数 val 加到 64 位数 <high,low> 中。
106 #define ADD64(val,low,high) \
107 __asm__("addl %4,%0 ; adcl $0,%1":"=r" (low),"=r" (high) \
108 :"" (low),"1" (high),"r" ((unsigned long) (val)))
109
    // 取用户内存中的 BCD 码数值并转换成临时实数格式。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得 BCD 码所在有效地址（math/ea.c），然后从用户数据区读取 10 字节相应 BCD 码值（其
    // 中 1 字节用于符号），同时转换成临时整数形式。最后把临时整数值转换成临时实数。
    // 参数：tmp - 转换成临时实数后的指针；info - info 结构指针；code - 指令代码。
110 void get_BCD(temp_real * tmp, struct info * info, unsigned short code)
111 {
112         int k;
113         char * addr;
114         temp_int i;
```

```
115         unsigned char c;
116
    // 取得 BCD 码数值所在内存有效地址。然后从最后 1 个 BCD 码字节（最高有效位）开始处理。
    // 先取得 BCD 码数值的符号位，并设置临时整数的符号位。然后把 9 字节的 BCD 码值转换成
    // 临时整数格式，最后再把临时整数值转换成临时实数。
117         addr = ea(info,code);                       // 取有效地址。
118         addr += 9;                                  // 指向最后一个（第 10 个）字节。
119         i.sign = 0x80 & get_fs_byte(addr--);        // 取其中符号位。
120         i.a = i.b = 0;
121         for (k = 0; k < 9; k++) {                   // 转换成临时整数格式。
122                 c = get_fs_byte(addr--);
123                 MUL10(i.a, i.b);
124                 ADD64((c>>4), i.a, i.b);
125                 MUL10(i.a, i.b);
126                 ADD64((c&0xf), i.a, i.b);
127         }
128         int_to_real(&i,tmp);                        // 转换成临时实数格式。
129 }
130
    // 把运算结果以短（单精度）实数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数格式的结果转换成短实数格式并存储到有效
    // 地址 addr 处。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
131 void put_short_real(const temp_real * tmp,
132         struct info * info, unsigned short code)
133 {
134         char * addr;
135         short_real sr;
136
137         addr = ea(info,code);                       // 取有效地址。
138         verify_area(addr,4);                        // 为保存结果验证或分配内存。
139         temp_to_short(tmp,&sr);                     // 结果转换成短实数格式。
140         put_fs_long(sr,(unsigned long *) addr);     // 存储数据到用户内存区。
141 }
142
    // 把运算结果以长（双精度）实数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数格式的结果转换成长实数格式，并存储到有
    // 效地址 addr 处。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
143 void put_long_real(const temp_real * tmp,
144         struct info * info, unsigned short code)
145 {
146         char * addr;
147         long_real lr;
148
149         addr = ea(info,code);                       // 取有效地址。
150         verify_area(addr,8);                        // 为保存结果验证或分配内存。
151         temp_to_long(tmp,&lr);                      // 结果转换成长实数格式。
152         put_fs_long(lr.a, (unsigned long *) addr);  // 存储数据到用户内存区。
153         put_fs_long(lr.b, 1 + (unsigned long *) addr);
154 }
```

```
155
    // 把运算结果以临时实数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数存储到有效地址 addr 处。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
156 void put_temp_real(const temp_real * tmp,
157         struct info * info, unsigned short code)
158 {
159         char * addr;
160
161         addr = ea(info,code);                           // 取有效地址。
162         verify_area(addr,10);                           // 为保存结果验证或分配内存。
163         put_fs_long(tmp->a, (unsigned long *) addr); // 存储数据到用户内存区。
164         put_fs_long(tmp->b, 1 + (unsigned long *) addr);
165         put_fs_word(tmp->exponent, 4 + (short *) addr);
166 }
167
    // 把运算结果以短整数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数格式的结果转换成临时整数格式。如果是负
    // 数则设置整数符号位。最后把整数保存到用户内存中。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
168 void put_short_int(const temp_real * tmp,
169         struct info * info, unsigned short code)
170 {
171         char * addr;
172         temp_int ti;
173
174         addr = ea(info,code);                           // 取有效地址。
175         real_to_int(tmp,&ti);                           // 转换成临时整数格式。
176         verify_area(addr,2);                            // 验证或分配存储内存。
177         if (ti.sign)                                    // 若有符号位，则取负数值。
178                 ti.a = -ti.a;
179         put_fs_word(ti.a,(short *) addr);               // 存储到用户数据区中。
180 }
181
    // 把运算结果以长整数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数格式的结果转换成临时整数格式。如果是负
    // 数则设置整数符号位。最后把整数保存到用户内存中。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
182 void put_long_int(const temp_real * tmp,
183         struct info * info, unsigned short code)
184 {
185         char * addr;
186         temp_int ti;
187
188         addr = ea(info,code);                           // 取有效地址。
189         real_to_int(tmp,&ti);                           // 转换成临时整数格式。
190         verify_area(addr,4);                            // 验证或分配存储内存。
191         if (ti.sign)                                    // 若有符号位，则取负数值。
192                 ti.a = -ti.a;
193         put_fs_long(ti.a,(unsigned long *) addr);       // 存储到用户数据区中。
```

```
194 }
195
    // 把运算结果以 64 位整数格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，然后把临时实数格式的结果转换成临时整数格式。如果是负
    // 数则设置整数符号位。最后把整数保存到用户内存中。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
196 void put_longlong_int(const temp_real * tmp,
197         struct info * info, unsigned short code)
198 {
199         char * addr;
200         temp_int ti;
201
202         addr = ea(info,code);                           // 取有效地址。
203         real_to_int(tmp,&ti);                           // 转换成临时整数格式。
204         verify_area(addr,8);                            // 验证存储区域。
205         if (ti.sign)                                    // 若是负数，则取反加 1。
206                 __asm__("notl %0 ; notl %1\n\t"
207                         "addl $1,%0 ; adcl $0,%1"
208                         :"=r" (ti.a),"=r" (ti.b)
209                         :"" (ti.a),"1" (ti.b));
210         put_fs_long(ti.a,(unsigned long *) addr);       // 存储到用户数据区中。
211         put_fs_long(ti.b,1 + (unsigned long *) addr);
212 }
213
    // 无符号数<high, low>除以 10，余数放在 rem 中。
214 #define DIV10(low,high,rem) \
215 __asm__("divl %6 ; xchgl %1,%2 ; divl %6" \
216         :"=d" (rem),"=a" (low),"=b" (high) \
217         :"" (0),"1" (high),"2" (low),"c" (10))
218
    // 把运算结果以 BCD 码格式保存到用户数据区中。
    // 该函数首先根据浮点指令代码中寻址模式字节中的内容和 info 结构中当前寄存器中的内容，
    // 取得保存结果的有效地址 addr，并验证保存 10 字节 BCD 码的用户空间。然后把临时实数格式
    // 的结果转换成 BCD 码格式的数据并保存到用户内存中。如果是负数则设置最高存储字节的最高
    // 有效位。
    // 参数：tmp - 临时实数格式结果值；info - info 结构指针；code - 指令代码。
219 void put_BCD(const temp_real * tmp,struct info * info, unsigned short code)
220 {
221         int k,rem;
222         char * addr;
223         temp_int i;
224         unsigned char c;
225
226         addr = ea(info,code);                           // 取有效地址。
227         verify_area(addr,10);                           // 验证存储空间容量。
228         real_to_int(tmp,&i);                            // 转换成临时整数格式。
229         if (i.sign)                                 // 若是负数，则设置符号字节最高有效位。
230                 put_fs_byte(0x80, addr+9);
231         else                                            // 否则符号字节设置为 0。
232                 put_fs_byte(0, addr+9);
233         for (k = 0; k < 9; k++) {                       // 临时整数转换成 BCD 码并保存。
234                 DIV10(i.a,i.b,rem);
```

```
235                     c = rem;
236                     DIV10(i.a,i.b,rem);
237                     c += rem<<4;
238                     put_fs_byte(c,addr++);
239             }
240 }
241
```