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

/*
 * Calculate the effective address.
 */
/*
 * 计算有效地址。
 */

#include <stddef.h>          // 标准定义头文件。本程序使用了其中的 offsetof()定义。

#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 <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。

// info 结构中各个寄存器在结构中的偏移位置。offsetof()用于求指定字段在结构中的偏移位
// 置。参见 include/stddef.h 文件。
static int __regoffset[] = {
        offsetof(struct info,___eax),
        offsetof(struct info,___ecx),
        offsetof(struct info,___edx),
        offsetof(struct info,___ebx),
        offsetof(struct info,___esp),
        offsetof(struct info,___ebp),
        offsetof(struct info,___esi),
        offsetof(struct info,___edi)
};

// 取 info 结构中指定位置处寄存器内容。
#define REG(x) (*(long *)(__regoffset[(x)]+(char *) info))

// 求 2 字节寻址模式中第 2 操作数指示字节 SIB（Scale，Index，Base）的值。
static char * sib(struct info * info, int mod)
{
        unsigned char ss,index,base;
        long offset = 0;

// 首先从用户代码段中取得 SIB 字节，然后取出各个字段比特位值。
        base = get_fs_byte((char *) EIP);
        EIP++;
        ss = base >> 6;                         // 比例因子大小 ss。
        index = (base >> 3) & 7;                // 索引值索引代号 index。
        base &= 7;                              // 基地址代号 base。
// 如果索引代号为 0b100，表示无索引偏移值。否则索引偏移值 offset=对应寄存器内容*比例因子。
        if (index == 4)
                offset = 0;
        else
                offset = REG(index);
        offset <<= ss;
```

```
    // 如果上一 MODRM 字节中的 MOD 不为零，或者 Base 不等于 0b101，则表示有偏移值在 base 指定的
    // 寄存器中。因此偏移 offset 需要再加上 base 对应寄存器中的内容。
44          if (mod || base != 5)
45                  offset += REG(base);
    // 如果 MOD=1，则表示偏移值为 1 字节。否则，若 MOD=2，或者 base=0b101，则偏移值为 4 字节。
46          if (mod == 1) {
47                  offset += (signed char) get_fs_byte((char *) EIP);
48                  EIP++;
49          } else if (mod == 2 || base == 5) {
50                  offset += (signed) get_fs_long((unsigned long *) EIP);
51                  EIP += 4;
52          }
    // 最后保存并返回偏移值。
53          I387.foo = offset;
54          I387.fos = 0x17;
55          return (char *) offset;
56  }
57
    // 根据指令中寻址模式字节计算有效地址值。
58  char * ea(struct info * info, unsigned short code)
59  {
60          unsigned char mod,rm;
61          long * tmp = &EAX;
62          int offset = 0;
63
    // 首先取代码中的 MOD 字段和 R/M 字段值。如果 MOD=0b11，表示是单字节指令，没有偏移字段。
    // 如果 R/M 字段=0b100，并且 MOD 不为 0b11，表示是 2 字节地址模式寻址，因此调用 sib()求
    // 出偏移值并返回即可。
64          mod = (code >> 6) & 3;          // MOD 字段。
65          rm = code & 7;                  // R/M 字段。
66          if (rm == 4 && mod != 3)
67                  return sib(info,mod);
    // 如果 R/M 字段为 0b101，并且 MOD 为 0，表示是单字节地址模式编码且后随 32 字节偏移值。
    // 于是取出用户代码中 4 字节偏移值，保存并返回之。
68          if (rm == 5 && !mod) {
69                  offset = get_fs_long((unsigned long *) EIP);
70                  EIP += 4;
71                  I387.foo = offset;
72                  I387.fos = 0x17;
73                  return (char *) offset;
74          }
    // 对于其余情况，则根据 MOD 进行处理。首先取出 R/M 代码对应寄存器内容的值作为指针 tmp。
    // 对于 MOD=0，无偏移值。对于 MOD=1，代码后随 1 字节偏移值。对于 MOD=2，代码后有 4 字节
    // 偏移值。最后保存并返回有效地址值。
75          tmp = & REG(rm);
76          switch (mod) {
77                  case 0: offset = 0; break;
78                  case 1:
79                          offset = (signed char) get_fs_byte((char *) EIP);
80                          EIP++;
81                          break;
82                  case 2:
83                          offset = (signed) get_fs_long((unsigned long *) EIP);
```

```
84                          EIP += 4;
85                          break;
86                  case 3:
87                          math_abort(info,1<<(SIGILL-1));
88          }
89          I387.foo = offset;
90          I387.fos = 0x17;
91          return offset + (char *) *tmp;
92  }
93
```