Pa_1-1

C 语言中的 struct 和 union 关键字都是什么含义，寄存器结构体的参考实现为什么把部分 struct 改成了 union？

①struct 和 union 关键字的含义：

struct: 结构体. 结构体内的每个数据成员都有各自独立的存储空间. 结构体的大小与各数据成员的大小有关.

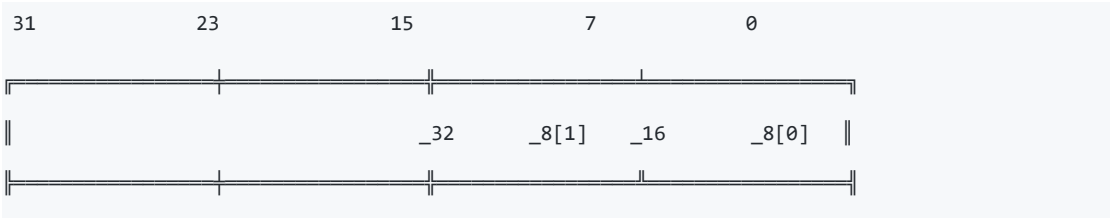
union: 联合体. 联合体只分配和最大数据成员所需存储空间相同的空间, 所有的数据成员都共享这一空间, 修改某一成员的同时可能会影响到其他成员. 联合体的使用会受到系统大端、小端存储方式的影响.

②参考实现中的修改的对应解释：

寄存器的参考实现如下：

```
1    typedef struct {
2          union {
3              union {
4                  union {
5                      uint32_t _32;
6                      uint16_t _16;
7                      uint8_t _8[2];
8                  };
9                  uint32_t val;
10             } gpr[8];
11             struct { // do not change the order of the registers
12                 uint32_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
13             };
14         };
15     } CPU_STATE;
```

最内层的 union (line 4 ~ 8)实现的是单个 32 位寄存器，并定义其低 16 位(line 6)，低 8 位、高 8 位(line 7)，如下所示：



中间的 union(line 3 ~ 10)实现的是 eax, ecx, edx, ebx, esp, ebp, esi, edi 这 8 个 32 位寄存器组，line9 处的 val 变量和内层的 union 声明的变量共享同一空间，相当于 line5 处的_32 变量.

外层的 union(line 2 ~ 14) 将中间层 union 实现的 8 个寄存器和 eax, ecx, edx, ebx, esp, ebp, esi, edi 一一对应，struct(line 11)中的各个 32 位变量和前面实现的各个 32 位寄存器共享内存空间.

最内层的

Pa_1-3

为浮点数加法和乘法各找两个例子：

1）对应输入是规格化或非规格化数，而输出产生了阶码上溢结果为正（负）无穷的情况；

2）对应输入是规格化或非规格化数，而输出产生了阶码下溢结果为正（负）零的情况。是否都能找到？若找不到，说出理由。

1）浮点数加法：

正无穷：

0x7f7fffff + 0x7f000001 = 0x7f800000

负无穷：

0xff7fffff + 0xff000001 = 0xff800000

浮点数乘法:

正无穷：

0x7f000000 * 0x40000000 = 0x7f800000

负无穷：

0xff000000 * 0x40000000 = 0xff800000

2）浮点数加法：

正零：

0x00000001 + 0x80000001 = 0x00000000

负零：

找不到合适的例子. 在 fpu.c 文件的 uint32_t internal_float_add(uint32_t b, uint32_t a)函数中，浮点数的阶码相加得到的结果对符号位的影响的相关代码如下所示：

```
sig_res = sig_a + sig_b;
//如果结果为负数的处理：
if (sign(sig_res))
{
    f.sign = 1;
    sig_res *= -1;
}
else
{
    f.sign = 0;
}
```

要得到负零，首先就要求尾数相加得 0.而代码对尾数相加的实现中，是通过无符号数

（补码）相加的方式得到结果，0 的表示方式是唯一的（即正零），故无法得到符号位是 1 的负零.

浮点数乘法：

正零：

0x00000001 * 0x00000001 = 0x00000000

负零：

0x80000001 * 0x00000001 = 0x80000000