```
            cq_index = cq_tail_ptr & cq_ring->size_mask;
        }

        // update CQ tail
        cq_ring->tail_ptr = cq_tail_ptr;
        //⑭
        mqnic_cq_write_tail_ptr(cq_ring);

        // process ring
        // read tail pointer from NIC
        //⑮
        mqnic_rx_read_tail_ptr(ring);

        ring_clean_tail_ptr = READ_ONCE(ring->clean_tail_ptr);
        ring_index = ring_clean_tail_ptr & ring->size_mask;

        while (ring_clean_tail_ptr != ring->tail_ptr)
        {
            rx_info = &ring->rx_info[ring_index];

            if (rx_info->page)
                break;

            ring_clean_tail_ptr++;
            ring_index = ring_clean_tail_ptr & ring->size_mask;
        }

        // update ring tail
        WRITE_ONCE(ring->clean_tail_ptr, ring_clean_tail_ptr);

        // replenish buffers
        //⑯
        mqnic_refill_rx_buffers(priv, ring);

        return done;
    }
```

对应代码中注释的编号，函数 mqnic_process_rx_cq 的工作流程如下。

① 调用 mqnic_cq_read_head_ptr 函数从硬件获取“接收完成队列”的 head，如果 head 不等于软件记录的此队列的 tail，则说明硬件在“接收完成队列”中添加了新的描述符。

② 将 tail 作为第一个描述符的索引。接下来依次处理“接收完成队列”中所有有效的描述符，所以③到⑬会在一个 while 循环中进行。

③ 计算当前索引指向的描述符的地址，计算方法为“接收完成队列的描述符缓存首地址+描述符索引×两个描述符之间的跨距”。

在前文中已经介绍过“发送完成队列”的描述符所使用的数据结构 struct mqnic_cpl，“接收完成队列”的描述符也使用同样的结构，只是其两个成员的意义有些变化：index 表示的是“接收队列”之前完成此次接收时处理的描述符的索引（硬件会将其持续加 1，所以需要“与”size_mask 后才能使用）；len 表示接收到的数据长度。于是就可以执行下述操作。

④ 使用 ring_index = cpl->index & ring->size_mask;获取“接收队列”中描述符的索引。

⑤ 从“接收队列”描述符对应的 rx_info 结构中获取描述符对应的内存页（数据结构为