MyNote

□□□□

AMBA

AXI

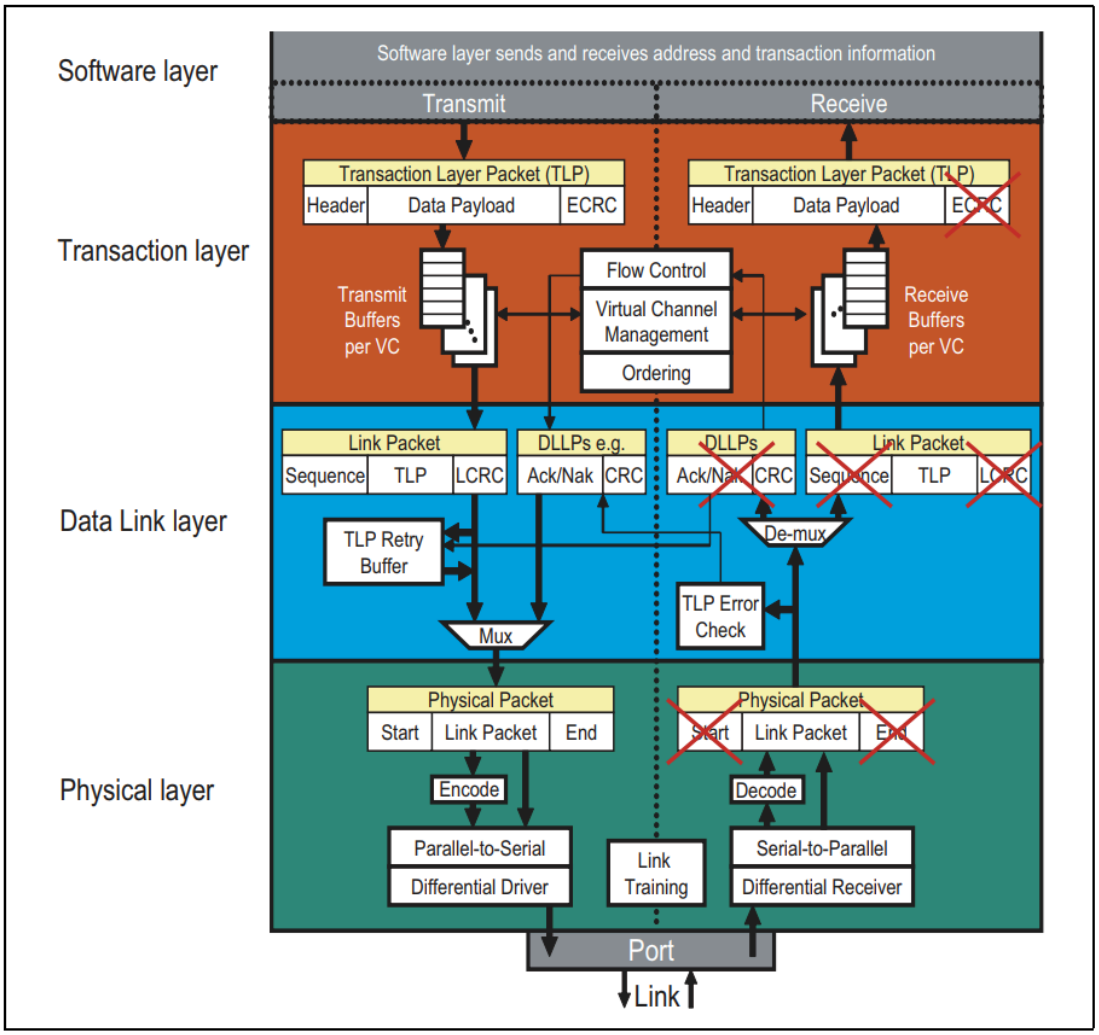
CHI

PCIe

PCIe

□□□

Figure 11-1: PCIe Port Layers



□□□□□

Transaction Layer

ARM N2 PCIe□□□□□

PCIe □□□□□□□

PCIe □□□□

PCIe AER

PCIe Interrupt

PCIe Hot-Plug

PCIe Power Management

Silicon IP

LeetCode

[toc]

记录自己做过的一些题，方便复习。

二分法

35. 搜索插入位置

while 循环，有四中情况，在左、在右、找到了，没找到。

- 注意mid的特殊计算方法。

- 注意区间折半时候加一减一。

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size() - 1;
        while(left<=right) {
            // ⊠⊠⊠⊠⊠⊠⊠left + right ⊠⊠⊠⊠int max
            // ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠
            int mid = left + ((right -left) >> 1);
            if (nums[mid] > target) {
                // ⊠⊠⊠
                right = mid - 1;
            } else if (nums[mid] < target) {
                // ⊠⊠⊠
                left = mid + 1;
            } else if (nums[mid] = target) {
                return mid;
            } else {
                //
                return left + 1;
            }
        }
        //⊠⊠⊠
        return left;

    }
};
```

74. 搜索二维矩阵

方式一：直接两层for循环，注意先行后列；

方法二：二分查找，注意将**mid**换算成二维位置；

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int left = 0;
        int right = matrix.size() * matrix[0].size() - 1;

        while (left <= right) {
            int mid = left + ((right - left) >> 1);
            int mid_value = matrix[mid/matrix[0].size()]
[mid%matrix[0].size()];
            // printf("%d \n", mid_value);

            if (target == mid_value) {
                return true;
            } else if (target > mid_value) {
                //right
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        // for (int x = 0; x < matrix.size(); x++) {
        //     for (int y = 0; y < matrix[0].size(); y++) {
        //         // printf("x-y: %d - %d ⊠ %d\n", x, y, matrix[x]
[y]);
        //         if (target == matrix[x][y]) {
        //             return true;
        //         }
        //     }
        // }
        return false;

    }
};
```

34. 在排序数组中查找元素的第一个和最后一个位置

数组有序，用二分；

找两个位置，用两次二分，注意如果中间的就是目标值，需要继续往左或者右寻找，因为这个不一定是最靠边的；

```cpp
class Solution {
public:
    vector<int>  ans;
    vector<int> searchRange(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size() - 1;

        int first = -1;
        int last = -1;



        // □□□□□□
        if (nums.size() == 0) {
            ans.push_back(-1);
            ans.push_back(-1);
            return ans;
        }

        while (left<=right)
        {
            /* code */
            int mid = left + ((right - left) >> 1);
            if (nums[mid] == target){
                first = mid;
                right = mid -1;
            } else if (nums[mid] > target) {
                //
                right = mid -1;
            } else {
                left = mid + 1;
            }
        }

        left = 0;
        right = nums.size() - 1;

        while (left<=right)
        {
            /* code */
            int mid = left + ((right - left) >> 1);
            if (nums[mid] == target){
                last = mid;
                left = mid +1;
            } else if (nums[mid] > target) {
                //
                right = mid -1;
```

```
            } else {
                left = mid + 1;
            }
        }

        ans.push_back(first);
            ans.push_back(last);
            return ans;


    }
};
```

33. 搜索旋转排序数组

部分有序，也可以用二分法；

先判断哪部分有序，在么一部分内部继续二分查找；

注意要包括左右两边的数字；

```cpp
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left=0,right=nums.size()-1;
        while(left<=right){
            int mid=left+(right-left)/2;
            if(target==nums[mid]) return mid;
            // [illegible]
            if(nums[left]<=nums[mid]){
                // [illegible]
                if(target>=nums[left]&&target<nums[mid]){
                    right=mid-1;
                }else{
                    // [illegible]
                    left=mid+1;
                }
            }else{
                // [illegible]
                if(target>nums[mid]&&target<=nums[right]){
                    left=mid+1;
                }else{
                    right=mid-1;
                }
            }
        }
        return -1;
    }
};
```