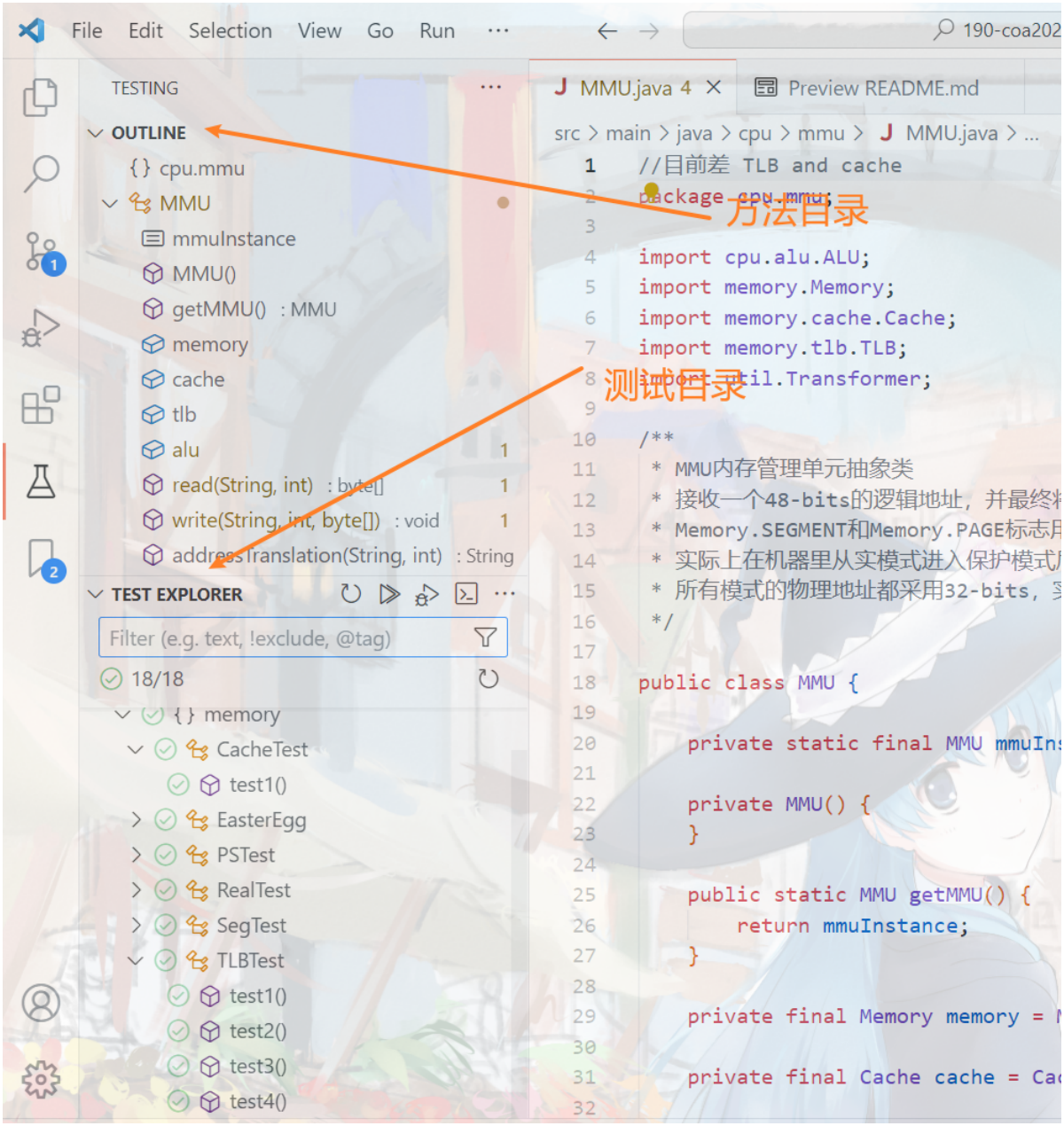
0x01 前置内容（可跳过）

给用 VScode 的 xd 安利几个小玩意，应该可以小小的提升一点生产力。（也可能是俺狭隘了刚刚才开始用。）

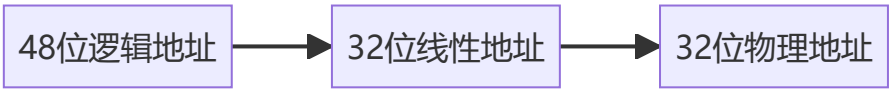
- Outline 大纲：Outline 可以显示当前代码文件所有的类、变量定义、方法定义。

`Ctrl+P`打开输入面板输入 `view Outline`即可打开 Outline 栏，俺一般把它放在跟 Test 一起，然后左边的面板基本就不需要再切换了。



- Bookmarks 插件：这个功能基本跟 Jetbrian 里的书签没什么区别，就是对关键步骤做一些标注，方便跳转。

0x02 地址转化



逻辑地址到线性地址的转化不需要考虑什么含义，可以把它当作一种纯粹的数学变换，且变换方法题目已给出，不多赘述。讲讲线性地址到物理地址的转化。

这里有三个模式：实模式、分段式和段页式，其中分段式本质上可以视为每一页大小为 1 字节的段页式，而实模式和分段式下的线性地址都等于物理地址，所以只讲讲段页式。

目前我们有：

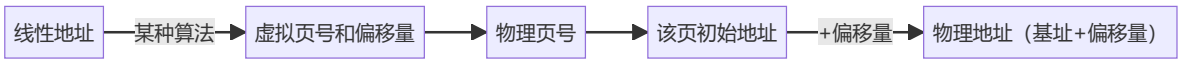
1. 32 位的线性地址
2. 磁盘和主存的单页大小 $PAGE\_SIZE\_B = 4KB$
3. 磁盘 64 MB 容量，划分为页数 $64MB/4KB = 16K$ 页，页表 `PageItem[Disk.DISK_SIZE_B / Memory.PAGE_SIZE_B]`
4. 主存 16 MB 容量，页数 $16MB/4KB = 4K$ 页。

页表存什么？

- 主存的页相关数据有：
  - boolean pageValid[]：判断该页是否可用，在主存加载数据时，找到第一个可用的页，写入数据。
  - memory[]：16 MB 容量，不难想到就是给你写数据的，每一页有 PAGE_SIZE_B = 4 KB 的空间，且主存地址称为**物理地址**，于是乎我们对第 pageNO 页写入 4 KB 数据，也就是**将数据写到物理地址在 $[pageNO * 4KB, (pageNO + 1) * 4KB)$ 这段范围内的主存空间中。**
  - 此处的页称为物理页。
- 硬盘的页相关数据（PageItem 类）有：
  - 32位的 char[] pageFrame：磁盘某页在主存中的物理页号。
  - boolean isInMen：磁盘某页是否在主存中
  - 此处的页称为虚拟页

再来一串逻辑链，它将让你直达终点。

目前我们有 32 位线性地址，需要转化为物理地址。线性地址是干嘛的？Readme 里没讲，google 之后知道，通过它可以算出某个虚拟页号，以及一个偏移量。所以逻辑链为：



Google 可知，线性地址 = 20 位虚拟页号 + 12 位偏移量

一切柳暗花明。

0x03 剩余工作

剩下的都是一些填表之类比较简单活，文档讲得比较清除就不（lan）多（de）讲了。

还需注意：

- 使用 cache 前需判断 available
- 没了

> 如果你认真读完了这篇文，还没有搞懂，很正常，笔者水平奇低。如果你看完了，豁然开朗，说明你真的在上课/读文档时下足了功夫，读完这篇文章只是浪费了你几分钟时间，将这几分钟用来写代码用来思考，也能豁然开朗。

> 行到水穷处，坐看云起时。