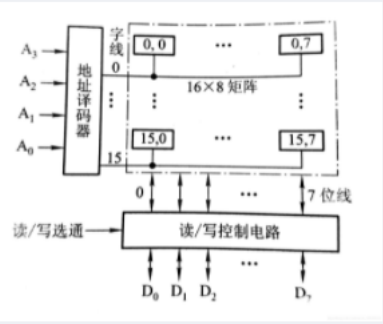
存储管理

- cpu是不能够访问磁盘的，所以说才会有缺页中断啊之类的，只有在内存的数据他才能够加以利用，疑问：什么叫做cpu访问数据呢，是指可以放入cpu的寄存器中吗
- cache主存层次的效率：cache的时间/有效时间，对于cache与主存主要是为了解决cpu与主存速度不匹配的问题
- 注意那几个rom：
  prom：可以改写一次
  EPROM：可以改写多次，但是呢写总是要慢一点
  flash：更快了
  ssd：固态硬盘
- 疑问：操作系统是存储在哪里又是如何被加载的呢？
- 对于dram：如果不说的话默认是刷新周期是2ms，他是电容充放电原理，所以会自己消失一部分
- 我们可以从地址结构的角度理解芯片位扩展和字扩展：
  位扩展的话，对地址结构是无关的，但是如果我们按照字节编址来分析那个地址得话，后面几位可以理解为同一组芯片里的不同芯片号码。如32位字，但是呢编址是按照8位来的，（疑问：那么此时的MDR多少位呢？）我们的芯片也是8位的，那么如果数据线是32位那么就需要位扩展。
  字扩展：那么就是把很多个拼在一起，但是呢他们共用底下的那些作为芯片内的地址空间，而通过上面高位（不一定指的是数字大）来选择组

- drom：地址复用技术要注意，但是相应的他的选择线要有行通选，列通选。
  疑问：是如何硬件进行地址选择的呢？
- 对于低位交叉编址：如果是4体交叉那么连续的读5or8实际上效果是一样的额，他缓不过来的
- 辨析对于多端口和低位还有单体多字：
  低位交叉最灵活，但是如果程序的局部性不好那么就无法很好的发挥作用
  对于多端口，读的话还可以但是，不利于写（会冲突），而且他需要配置多套读写电路
  单体多字，不灵活一次只能读出很多字，而不能从中做选择。而且相应的在cpu中的，寄存器要求也更高。
- 写分配与非写回！未命中时（从分配不分配中选择）
- 改进的clock：第一轮选择修改位和引用位全为0的，第二轮选择引用位为0的并且，在他走过的地方把引用位改为0（修改位不能改的啊大哥不然你以后怎么写回）之后重复这两步
- 线选法和重合法

2.半导体存储芯片的译码驱动方式

1.线选法



线选法

适用容量不大的存储芯片

2.重合法

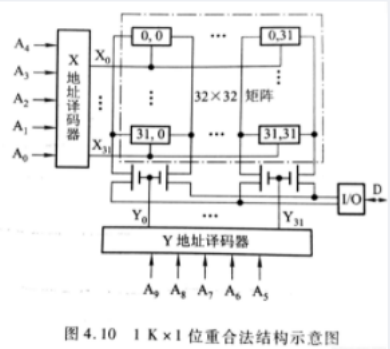


图 4.10 1 K×1 位重合法结构示意图

重合法

操作系统

α. 疑问：操作系统在一个进程缺页了肯定会阻塞他的吧（然后等待页面读入内存再唤醒他吗？）
β. 用户空间也是指的虚地址
γ. 名地址，存储空间
δ. 可重入程序通过减少对换次数来提高效率
ε. 段页式存储：段表项存储的是本段的页表首地址，最终的物理地址查出对应页表项，读出其中的页框号然后与偏移拼接（所以说段页式地址结构是：1.段号（根据段号我们知道了他的页表首地址在哪）2.页号（我们到了页表中可以按照页号来搜索到页表项）3.根据页表项的页框号就能拿到物理地址（再拼接上最后那一部分）
ζ. 区分缺页与越界中断
η. tlb->drom->page fault->tlb->物理页框号

https://www.jianshu.com/p/c320feea72cb