微机原理与接口技术

姓名：陈致蓬

单位：中南大学自动化学院

电话： 15200328617

Email ： ZP.Chen@csu.edu.cn

Homepage:

https://www.scholarmate.com/psnweb/homepage

QQ ： 315566683

第5章
RAM、MMU、ROM

5.1 RAM

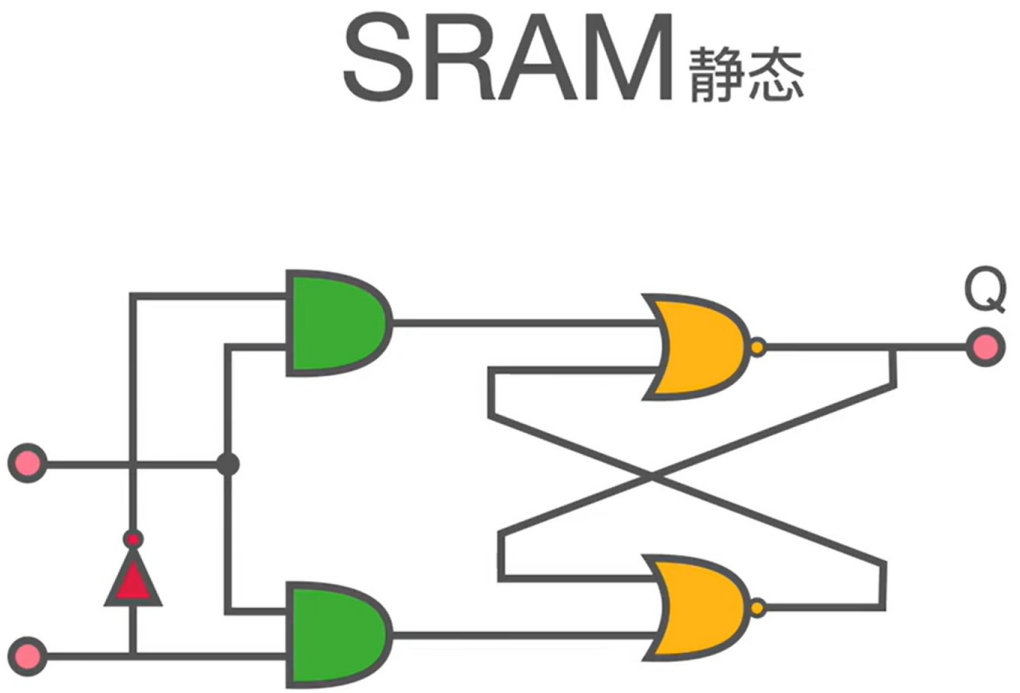
5.2 MMU

5.3 ROM

5.1 RAM
5.1.1 SRAM

SRAM ：

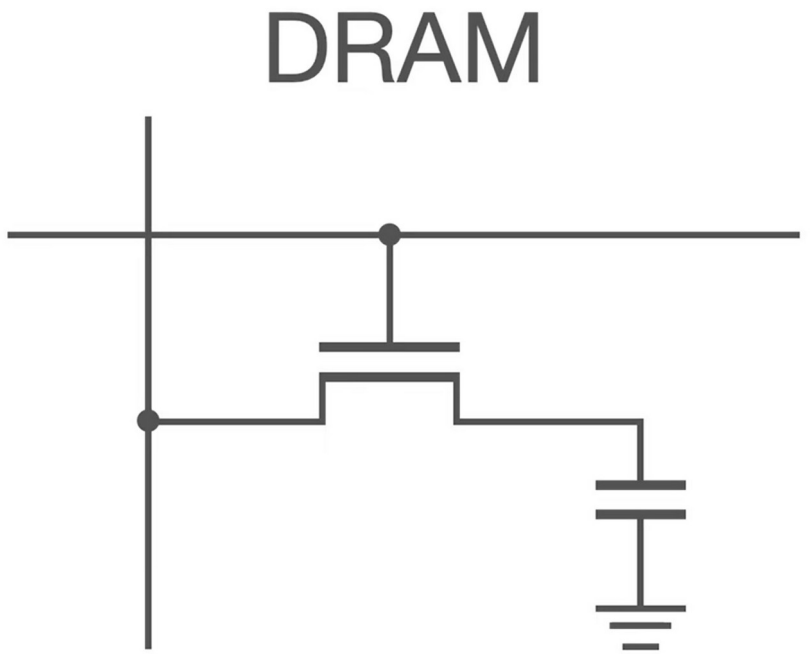
- **缓存采用静态随机存取存储器 (Static Random Access Memory, SRAM) 实现**
- **由 RS 触发器保持数据，无需刷新**
- **速度比 DRAM 快，比寄存器慢，但电路结构更复杂，难以大规模集成，因此用于 CPU 中的缓存**



5.1 RAM
5.1.2 DRAM

DRAM ：

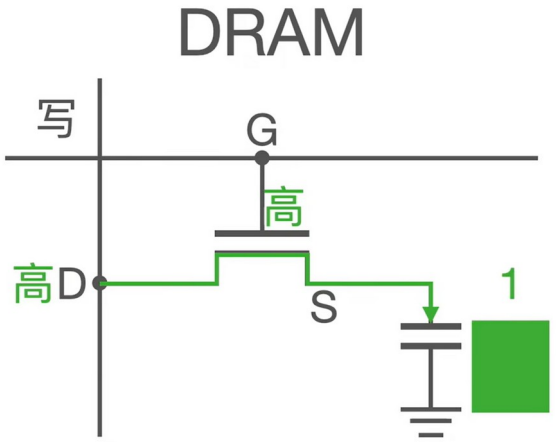
- 采用动态随机存取存储器 (Dynamic Random Access Memory, DRAM) 实现
- 采用电容存储信息，结构简单
- 电容存在漏电流，需要动态刷新给电容充电，因此得名 DRAM
- 由于电容充电和刷新需要时间，因此其相较于 SRAM 要更慢

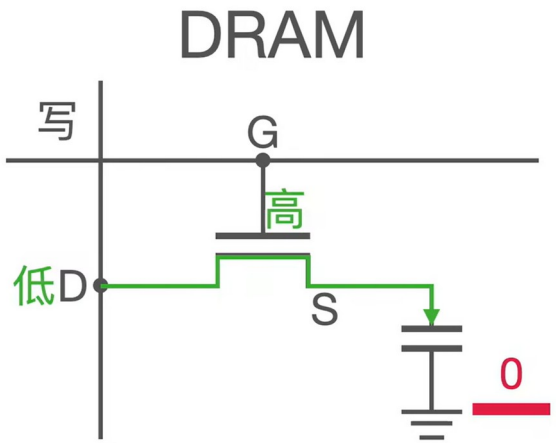


5.2.1 内存电路实现

DRAM 写：

- **G 极通高电平，三极管导通， D 极通高电平，电容充电，写入 1 ，通低电平，电容放电，写入 0**

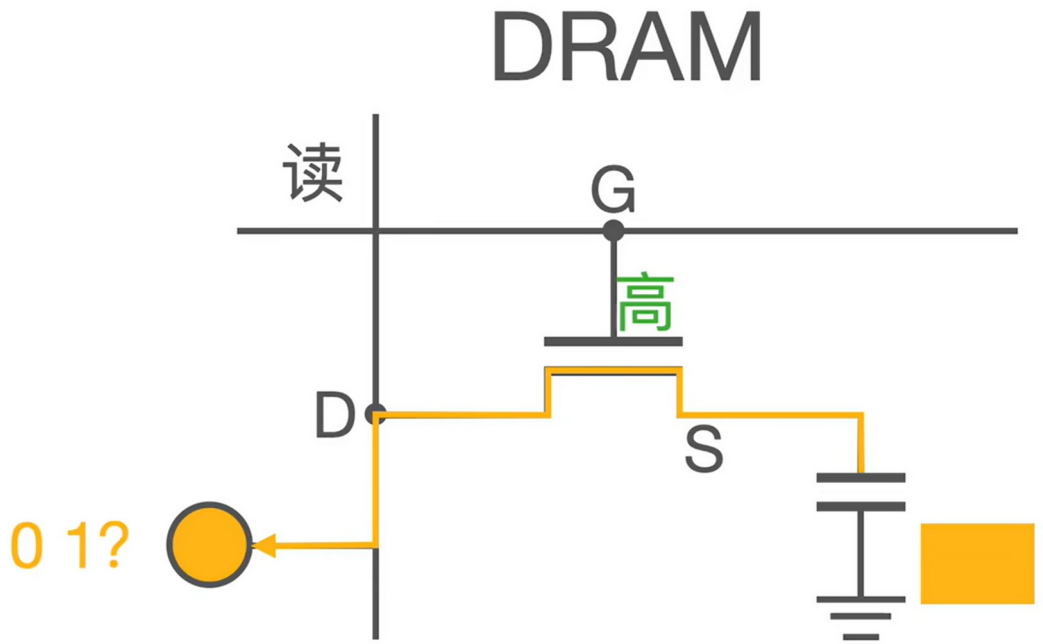




5.2.1 内存电路实现

DRAM 读：

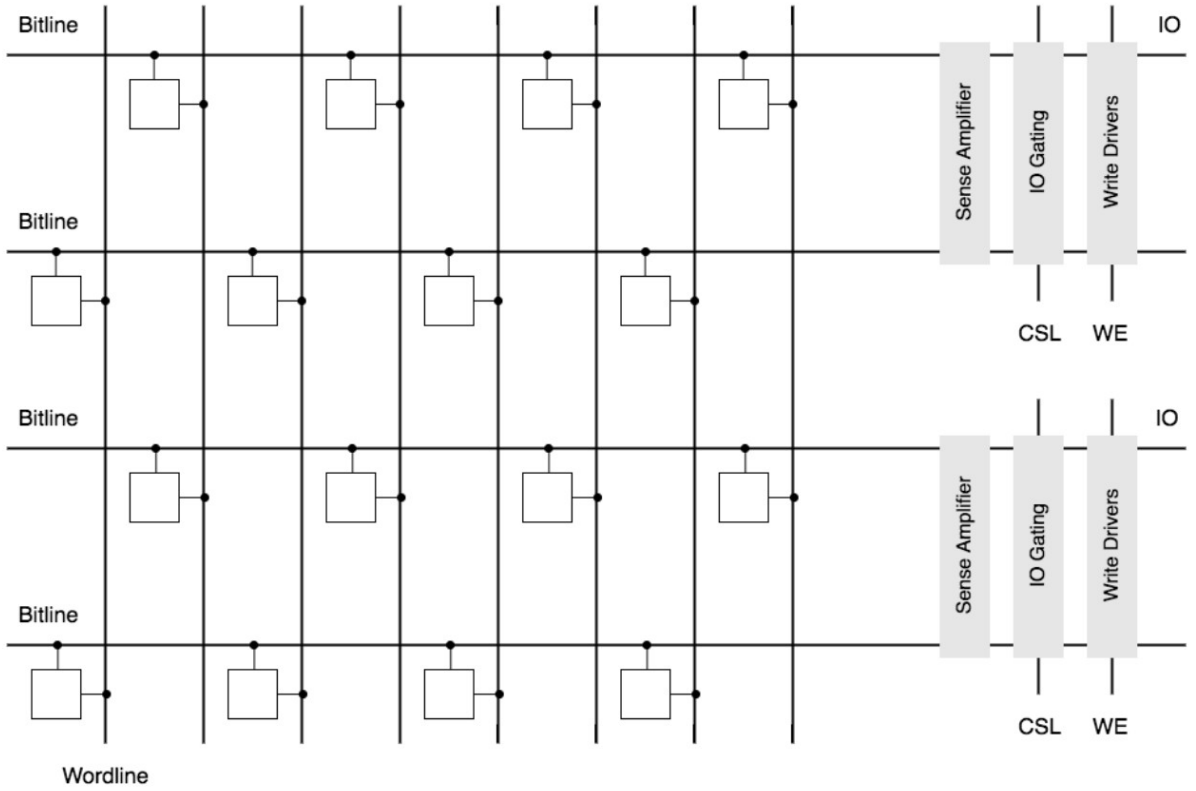
- **G 极通高电平，三极管导通，D 极接信号放大器读取电容中的电压信息，判断存储的值**



5.2.2 内存构成和寻址

内存构成：

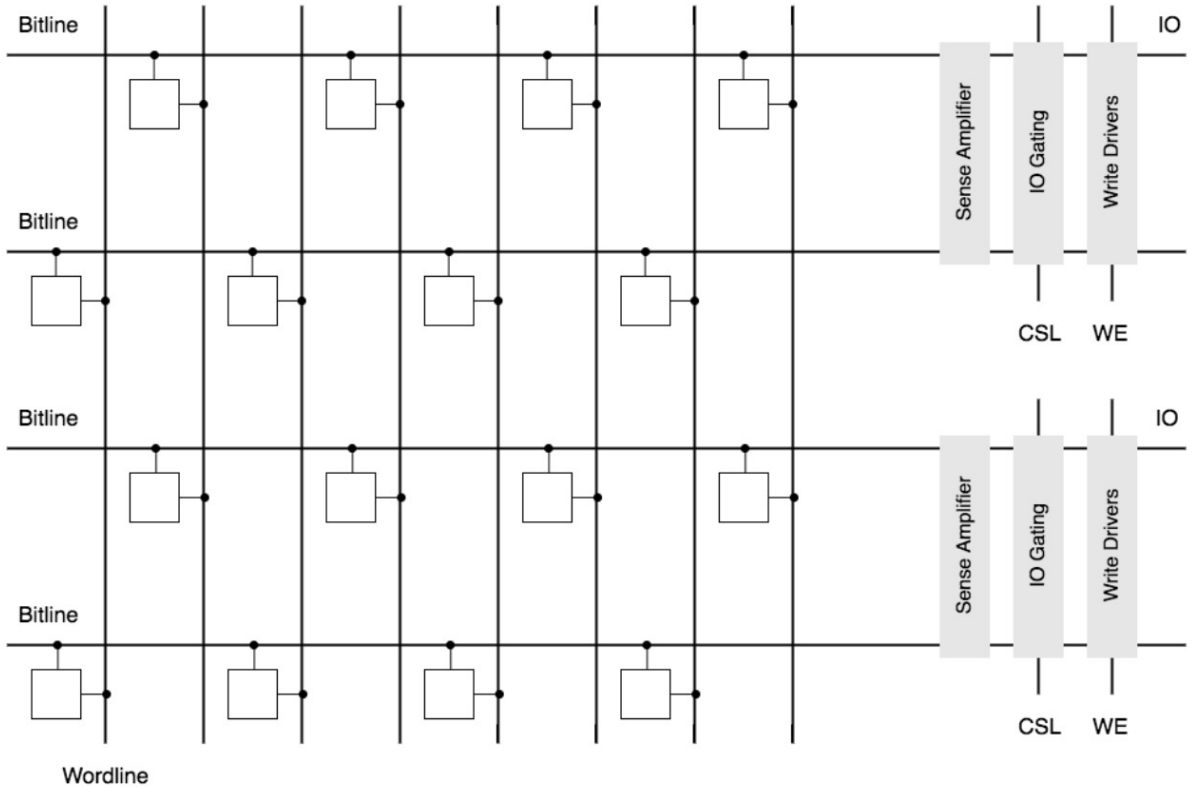
- 由多个 Cell 以特定的方式组成一个 Memory Array ，也称一个 Bank
- 空间 (bit) = m x n x w
  - 字线数 m
  - CSL 或 WE 数 n
  - 一次读写的最小 bit 数 w



5.2.2 内存构成和寻址

内存寻址：

- 将列连成组，构成最小可寻址单元
- 对最小寻址单元进行编址
- 使能相应的字线、 CSL 和 WE ，控制最小寻址空间的读写



5.2.3 程序读写内存的方式

程序在内存上：

- 内存空间分段，将程序分段为连续的代码段、数据段、堆栈段

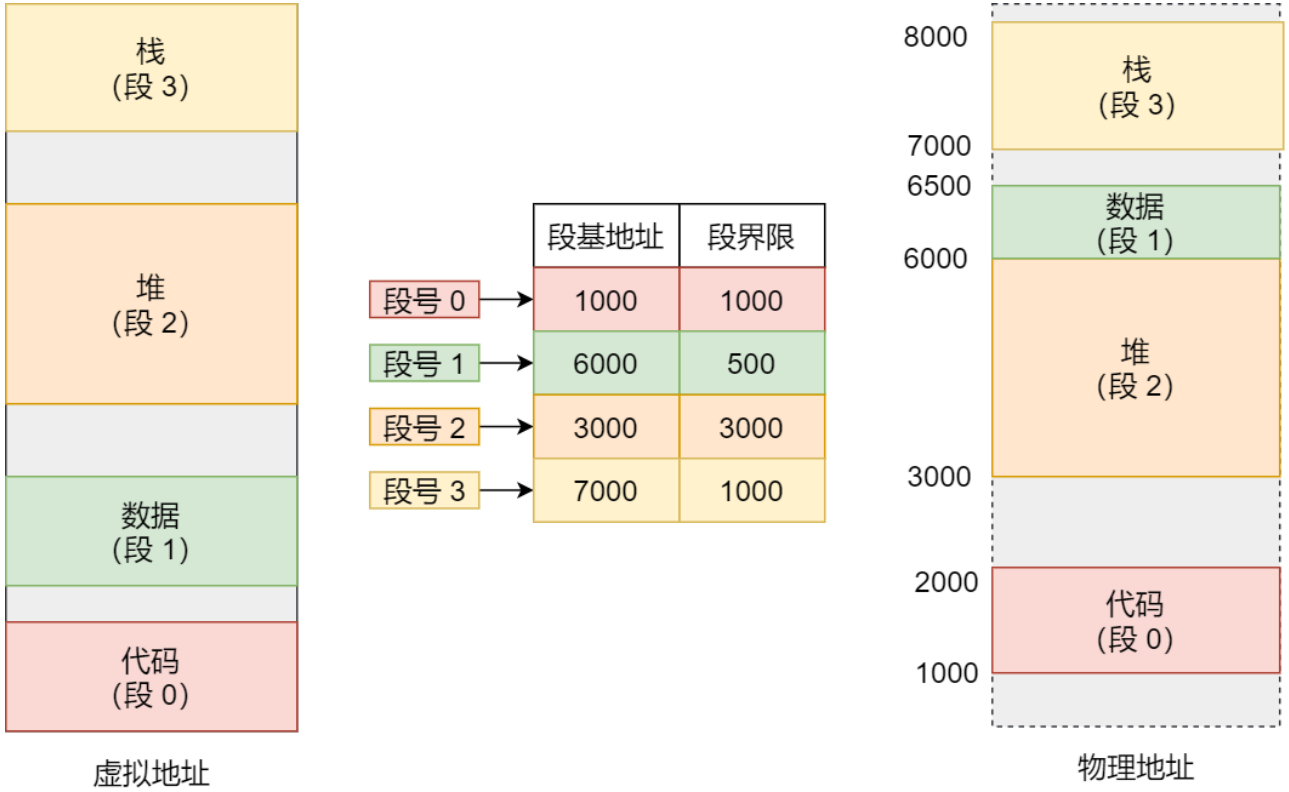
直接读写内存：

- 将程序给段直接分配到实际物理地址上， CPU 直接用物理地址访问内存
- 在多程序同时执行时，可能会出现多程序同时操作同一内存的情况，出现不安全问题
- 因此后来出现了虚拟内存

5.2.3 程序读写内存的方式

虚拟内存（早期）：

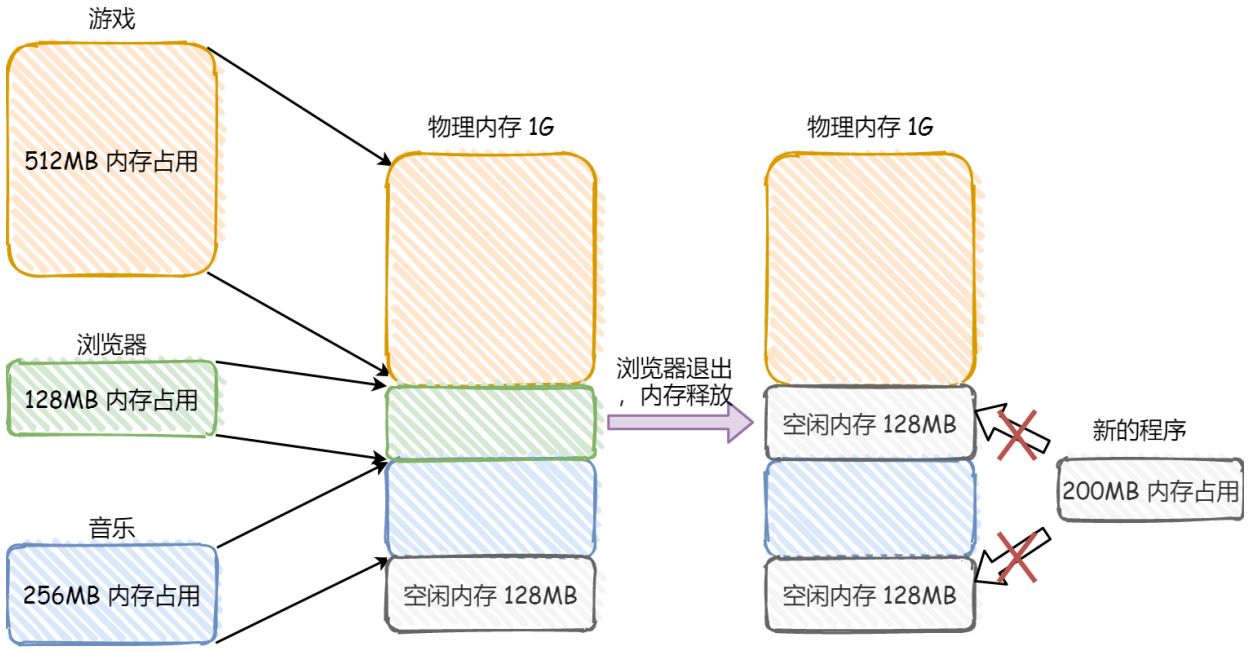
- 为各个程序分配虚拟内存，各个程序在虚拟内存上进行分段
- 将各段映射到连续的物理空间中



5.2.3 程序读写内存的方式

虚拟内存（早期）：

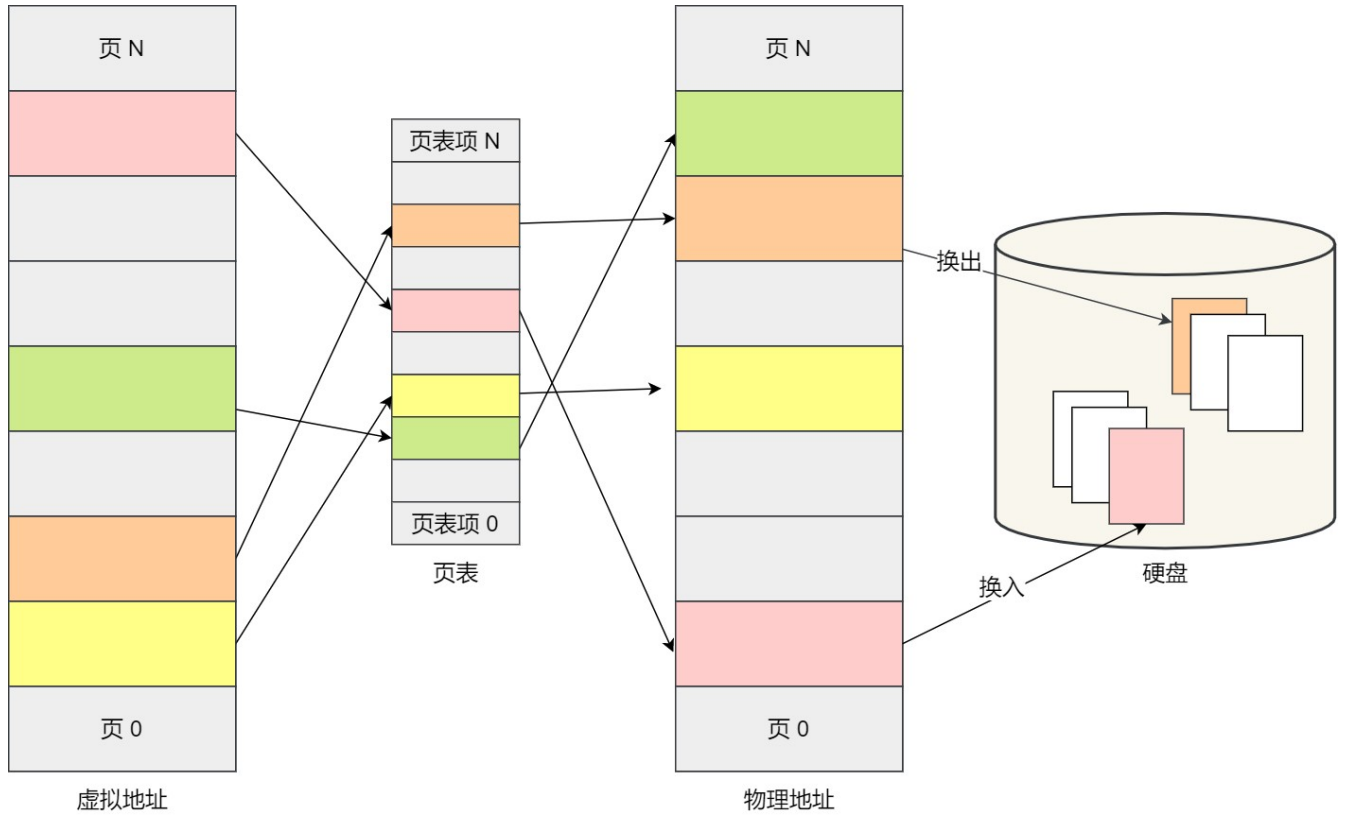
- 内存碎片问题
- Swap 模式，将部分内存中的内容移动到硬盘，再执行新的程序，待需要再使用到移动的内容时，再将其重新移动到内存中
- Swap 模式需要和硬盘交换数据，会拖慢程序运行



5.2.3 程序读写内存的方式

虚拟内存：

- 为了解决内存碎片无法利用的问题，同时减少 Swap 拖慢运行，将内存分为更小的页 (Page) 单元，一般一个页大小为 4K 或更小
- 进行地址映射将连续的虚拟地址映射到物理地址中的各个页中

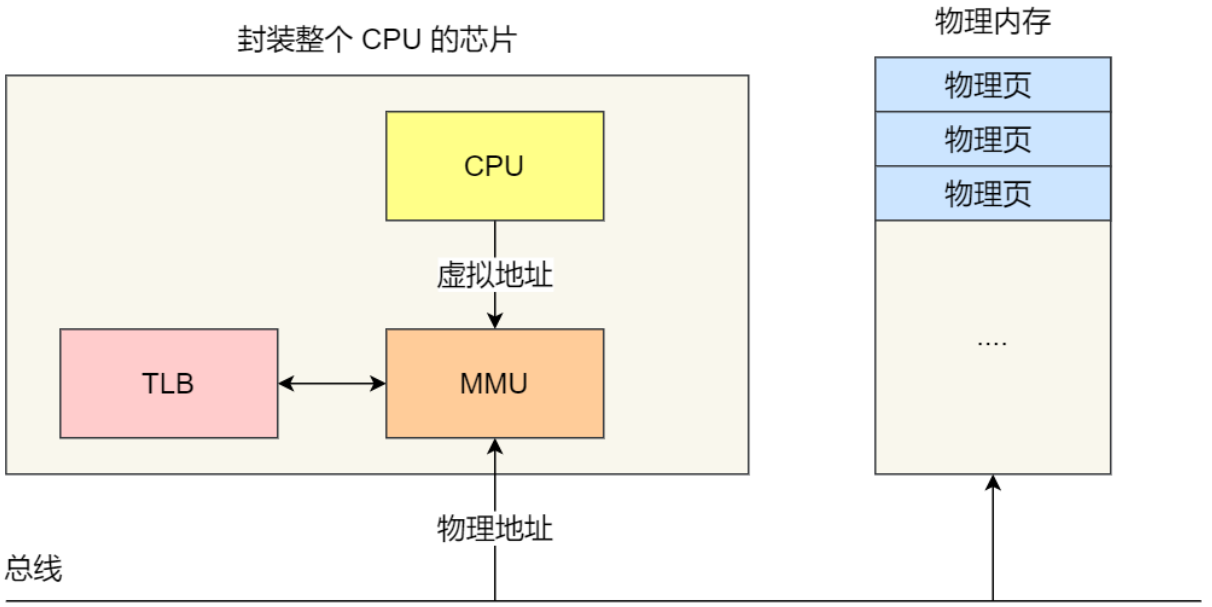


5.2 MMU

5.2.1 MMU 概念简介

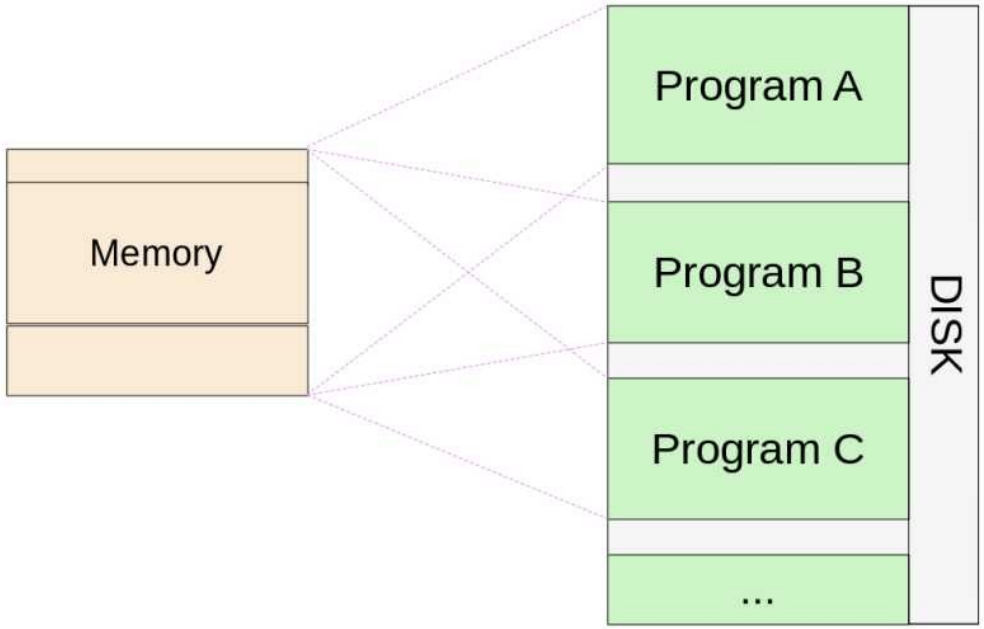
MMU(Memory Management Unit)，即内存管理单元，是现代CPU 架构中不可或缺的一部分，MMU 主要包含以下几个功能：

- 虚实地址翻译
- 访问权限控制
- 引申的物理内存管理



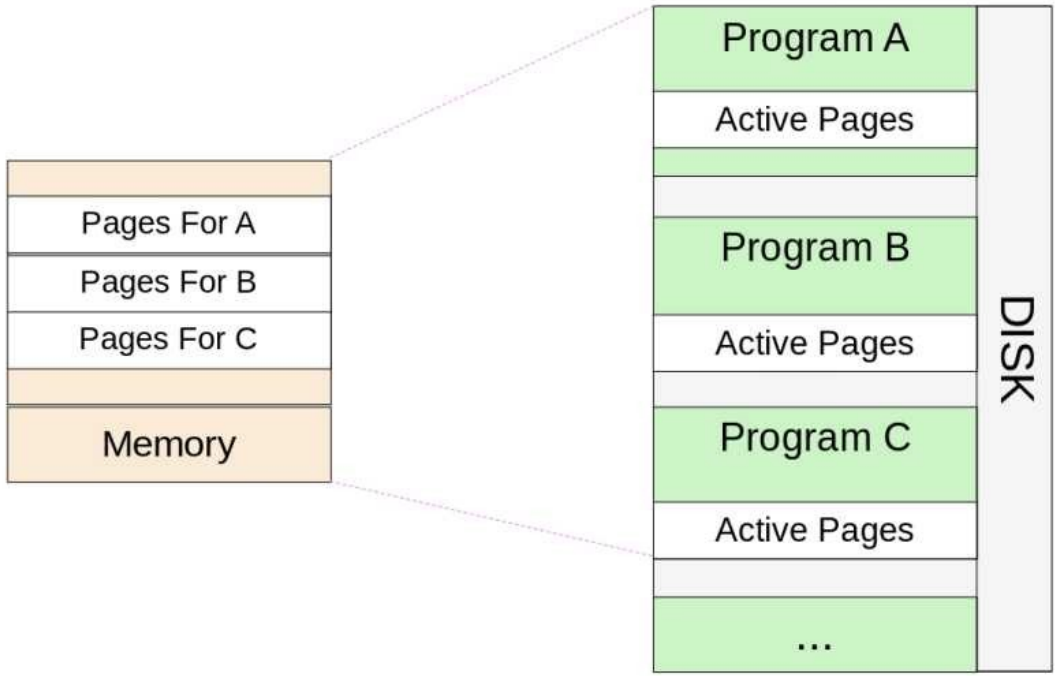
5.2.2 MMU 的由来

- **Swap 模式：早期计算机在执行程序时，将程序从磁盘加载到内存执行中执行，在多用户系统中，当新的用户程序被执行前，需要先将当前用户的程序从内存 swap 到磁盘，然后从磁盘加载新的程序执行，当前用户退出后，再将前一用户程序从磁盘中加载到内存继续执行，每次用户切换伴随程序的 swap ，消耗较大。**



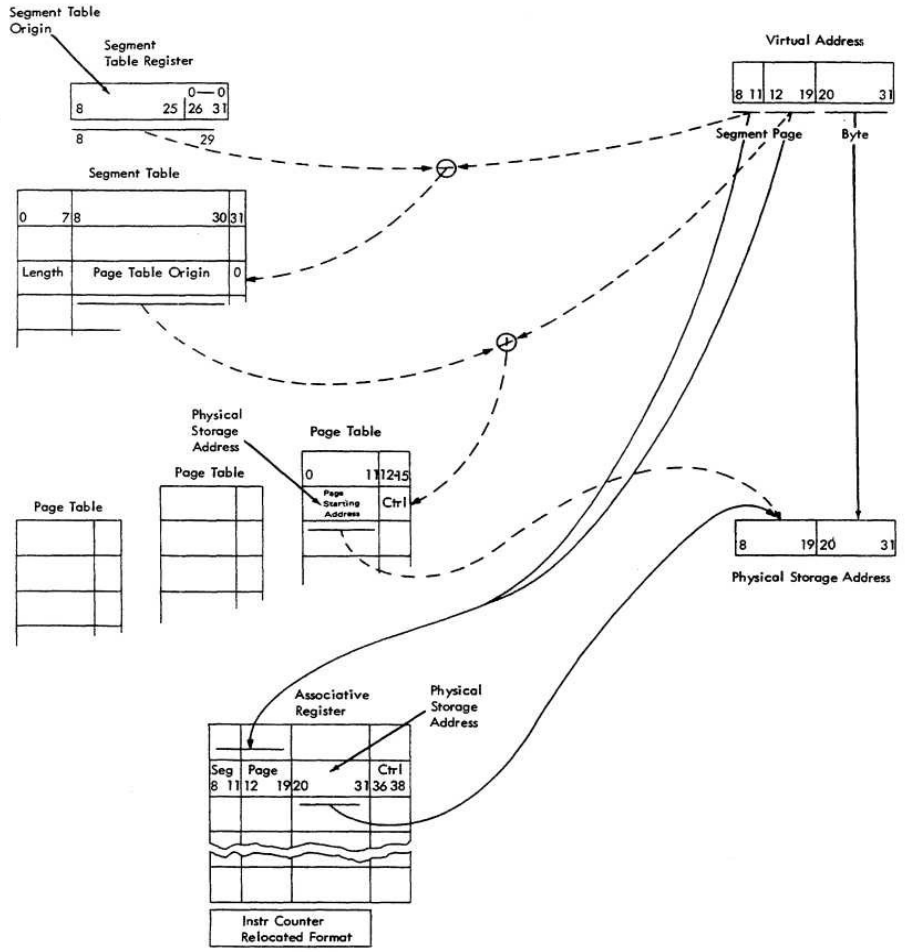
5.2.2 MMU 的由来

- **Page 模式：将内存划分为固定大小的 Page ，一般为 4K 或者更小，这样用户程序按需以 Page 的方式加载到内存，不需要将整个程序加载到内存，这样内存可同时容纳更多程序，而无需按照用户切换进行 swap ，提升了内存利用率和加载时间。**



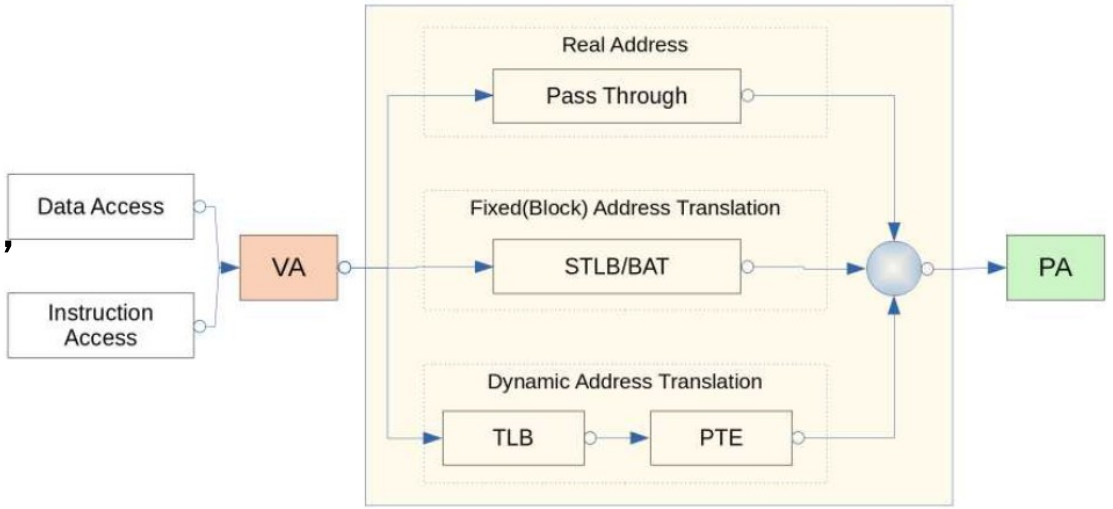
5.2.2 MMU 的由来

- 动态地址转换 (DAT - Dynamic Address Translation) ：在 Page 模式基础上，为用户程序分配虚地址 (VA) ，通过 DAT 转换为物理地址 (PA) 进行访问。



5.2.2 MMU 的由来

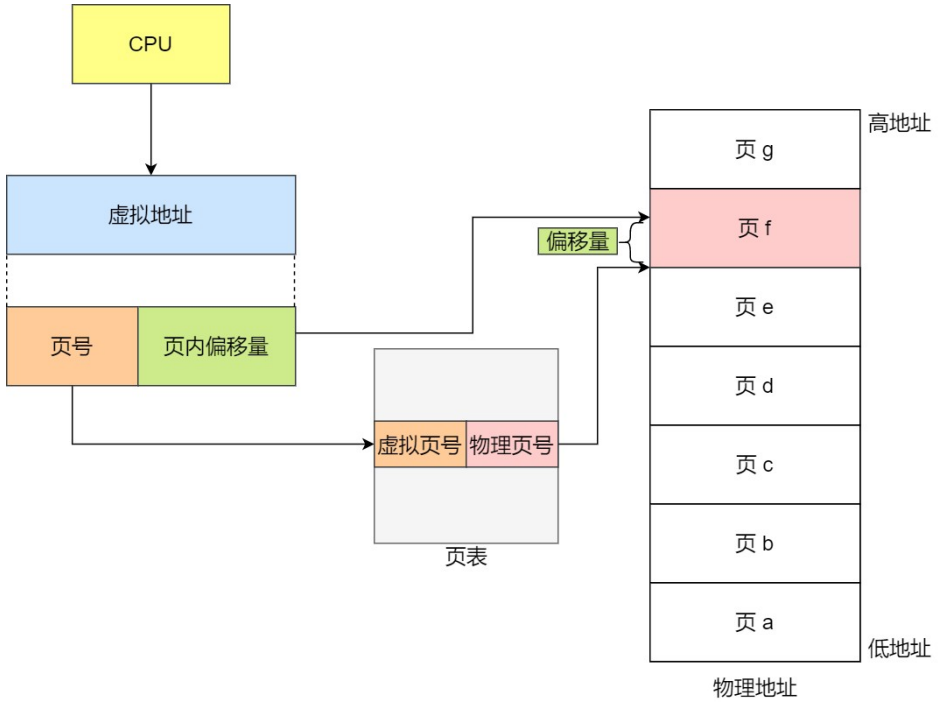
- **现代 MMU ：可进行实地址模式、块地址转换模式和页地址转换模式三种模式的地址转换。**
  - 实地址模式： CPU 状态位中 MMU 使能位清零， MMU 处于关闭状态，此时 CPU 操作的地址不经过转换 (VA=PA) ，直接作为物理地址进行访问。
  - 块地址转换模式：或者成为固定的地址转换或静态配置的地址转换表，这种模式支持配置一些固定的内存地址映射，这种配置没有页表查找过程转换速度快，但是缺乏灵活性，一次配置永久使用。
  - **页地址转换模式：将虚拟地址一部分用于索引分段，一部分用于索引页表，最终得到物理地址的页起始地址，再加上最后 12 位的页偏移量得到真正的物理地址。**
  - **相较于 DAT 模式， MMU ：**
    - **增加了 PTE 表的缓存 TLB**
    - **支持更大的物理地址 (36bit 以上) 或逻辑地址支持多种 PTE 查找方式，如硬件查找和软件查找**



5.2.3 MMU 地址转换

地址翻译：将虚拟地址 (VA) 转换为物理地址 (PA)

- **虚拟地址 (VA) 组成：**
  - 虚拟页序号 VPN （virtual paging number），用于索引页表 (Page Table) 得到页表条目 (Page Table Entry)
  - 偏移量 (Offset)
- **真实地址 (PA) 组成 = 页表条目 (Page Table Entry) + 偏移量 (Offset)，即为 PTE::Offset**



5.2.3 MMU 地址转换

在 32 位的环境下，虚拟地址空间共有 4GB ，假设一个页的大小是 4KB(2^12) ，那么就需要大约 100 万 (2^20) 个页，每个页表项需要 4 个字节大小来存储，那么整个 4GB 空间的映射就需要有 4MB 的内存来存储页表，如果有 100 个程序，则在页表的开销将达到 400M ，如果在 64 位系统中将会更大。

- 为解决这一问题，提出了多级页表

5.2.3 MMU 地址转换

多级页表占用的空间为什么没有更大？（以二级页表为例）

- 局部性原理
- 如果 4GB 的虚拟地址全部都映射到了物理内存上的话，二级分页占用空间确实是更大了，但是，我们往往不会为一个进程分配那么多内存
- 对于大多数程序来说，其使用到的空间远未达到 4GB ，大部分二级页表无需分配
- 对于已分配的页表项，如果存在最近一定时间未访问的页表，在物理内存紧张的情况下，操作系统会将页面换出到硬盘。
- 例如，假设只有 20% 的一级页表项被用到了，那么页表占用的内存空间就只有 4KB(一级页表)20%*4MB(二级页表)=0.804MB ，比单级页表占用 (4M) 更小

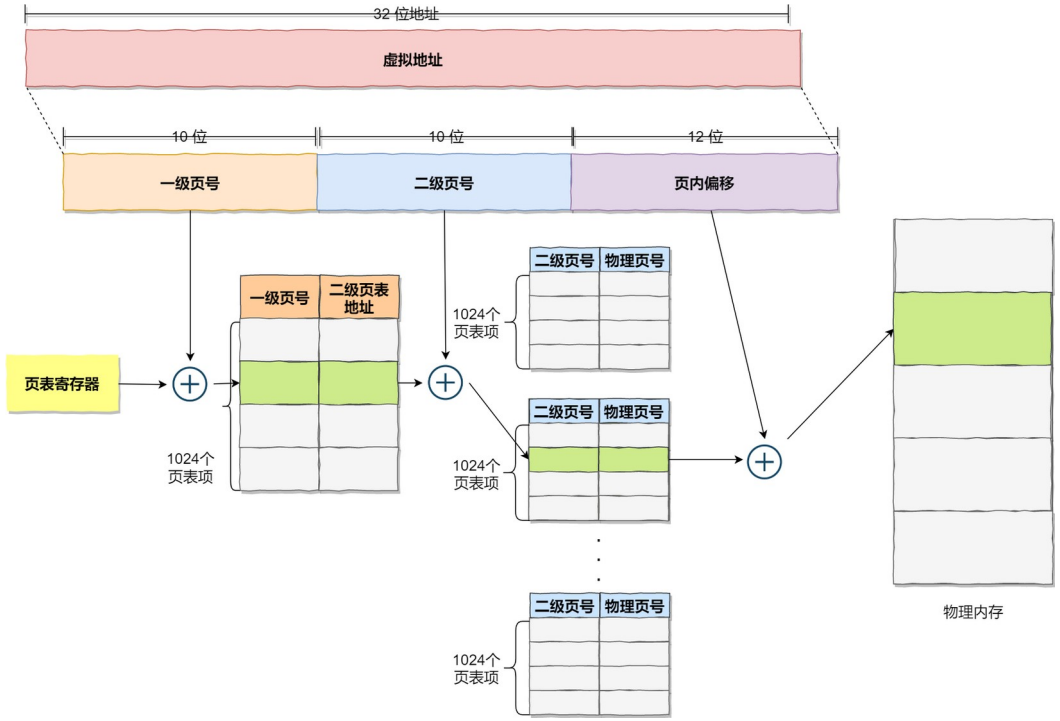
5.2.3 MMU 地址转换

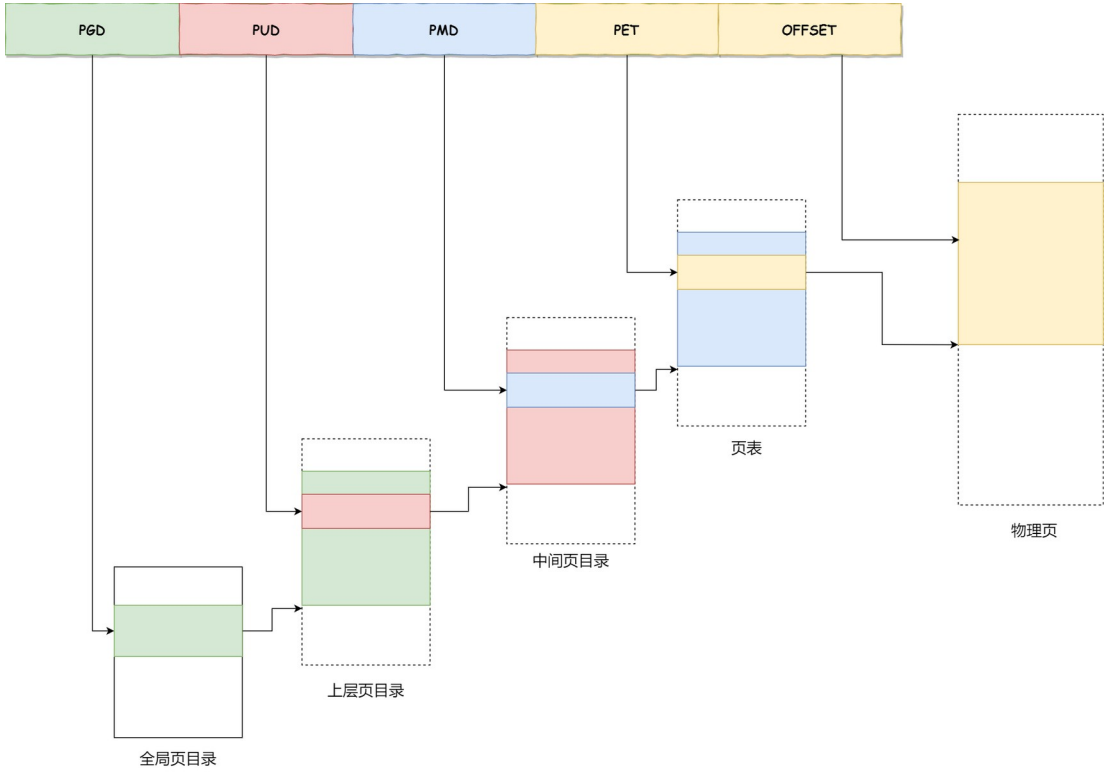
为何单级页表不能部分分配？

- 页表一定要覆盖全部虚拟地址空间
- 以 4G 虚拟地址空间为例，不分级的页表就需要有 100 多万个页表项来映射，而二级分页则只需要 1024 个页表项（此时一级页表覆盖到了全部虚拟地址空间，二级页表在需要时创建）

5.2.3 MMU 地址转换

在 32 位系统中多为 2 级页表，在 64 位中多为 4 级页表

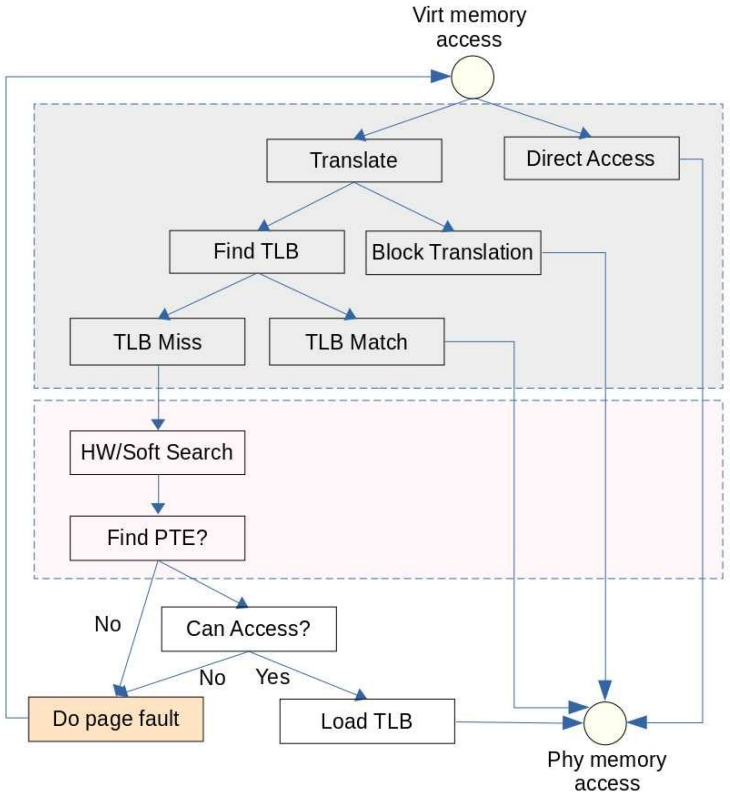




5.2.3 MMU 地址转换

多级页表意味着更多次的转换，如何解决效率问题？

- 局部性原理
- 在一段时间内，整个程序的执行仅限于程序中的某一部分，执行所访问的存储空间也局限于某个内存区域
- 转译后备缓冲器 TLB(Translation lookaside Buffer)，存放程序最常访问的页表项的 Cache
- 在访问虚拟地址时，先查询 TLB 中是否存在，不存在则进行常规页表查询
- Do Page Fault ，分为几种情况：
  - 新申请内存第一次读写，触发物理内存分配
  - 进程 fork 后子进程写内存触发 Copy-On-Write 。
  - 非法内存读写，错误处理



5.3 ROM

5.3.1 ROM 基本概念

ROM ：

- 是 **Read-Only Memory** 的缩写，可译为只读存储器。 **ROM** 可以永久存储数据，并且和 **RAM** 一样，是计算机中主要的存储器。
- 常见的 **ROM** 如游戏卡带、电脑固态硬盘等

5.1.2 ROM 分类

ROM 分类：

- 掩模式只读存储器 (MROM) ：内部数据在制造时嵌入，后期不允许修改或删除
- 可编程只读存储器 (PROM) ：可编程 ROM ，但编程后无法修改
- 可擦除可编程只读存储器 (EPROM) ：利用特殊设备 (例如 PROM 编程器) 可实现多次编程，同时利用特定频率的紫外线可在 40 分钟左右删除 PROM 存储的数据
- 带电可擦可编程只读存储器 (EEPROM) ：支持多次编程和删除 (最多可 10000 次) ，但无需借助紫外线就可以删除内部存储的数据
- 快闪存储器 (FLASH ROM ，简称闪存) ： EEPROM 的高级版本，它的优势在于，用户可以在特定时间内一次性删除或写入 512 个字节的数据。而 EEPROM 一次性只能写入或删除 1 个字节的数据。闪存的读写效率很高，并且可以承受高温和高压，所以非常耐用

谢谢大家！