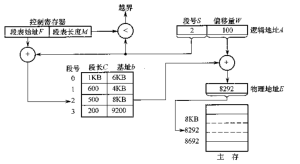
3.1 内存管理概念（下）

基本分段存储管理方式

- 出发点
  - 分页是从计算机角度考虑设计的，目的是为了内存的利用率，提高计算机性能，分页通过硬件机制实现，对用户完全透明
  - 分段是从用户和程序员的角度提出，满足方便编程，信息保护和共享，动态增长及动态链接等多方面的需要
- 分段
  - 按照用户进程中的自然段划分逻辑空间
  - 地址结构 = 段号S+段内偏移量W
  - 页式系统中，页号和页内偏移对用户透明　　段式系统中 段号和段内偏移量必须由用户显示的提供
- 段表
  - 每个进程都有一张逻辑空间与内存空间映射的段白，这个段表项对应进程的一段，段表项记录该段在内存中的始址和长度
  - 段表内容 = 段号 + 段长 + 本段在主存中的地址
- 地址变换机构
  - 逻辑地址A中取出段号S和段内偏移量W
  - 比较段号S和段表长度M，若S>=M,则产生越界中断，否则继续执行
  - 段号S对应的段表项地址 = 段表始址F+段号S*段表项长度，从该段表项中取出段长C，比较段内偏移量与C的大小判断是否出现越界
  - 取出段表项中该段的始址b,计算E = b+W，用得到的物理地址E去访问内存

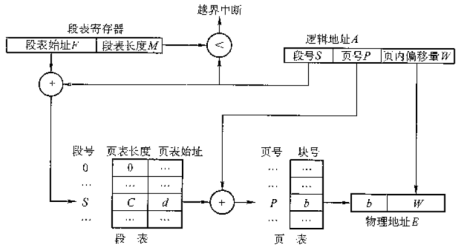
变换过程



- 段的共享与保护
  - 共享：两个作业的段表中响应表项指向被共享段的同一个物理副本来实现的　　纯代码或者可重入代码以及不可修改的数据可以被共享
  - 保护机制
    - 存取控制保护
    - 地址越界保护

段页式管理方式

- 页式存储有效的提高内存利用率，分段存储能反映程序的逻辑结构并有利于段的分享，将这两种方式结合一下　　这种二者结合的方法经常在计算机理论中遇到
- 思想
  - 作业的地址空间首先被分成若干逻辑段，每段有自己的段号
  - 每个段分成若干大小固定的页
  - 对内存空间的管理仍然和分页存储管理一样
- 地址结构　　段号S+页号P+页内偏移量W
- 为了实现地址变换，系统为每个进程建立了一张段表，每个分段有一个页表　　一个进程中，段表只能有一个，页表可以有多个



- 地址变换方式

补充

- 不能被修改的代码称为纯代码或可重入代码（不属于临界资源）
- 分段与分页的区别
  - 分页对用户不可见，分段对用户可见
  - 分页的地址空间是一维的，分段的地址空间是二维的
  - 分页（单级页表）、分段访问一个逻辑地址都需要两次访存，分段存储中也可以引入快表机构
  - 分段更容易实现信息的共享和保护（纯代码问重入代码可以共享）
- 分段与分页优缺点
  - 分页管理
    - 优点：内存空间利用率高，不会产生外部碎片，只会有少量的页内碎片
    - 缺点：不方便按照逻辑模块实现信息的共享和保护
  - 分段管理
    - 优点　　很方便按照逻辑模块实现信息的共享和保护
    - 缺点
      - 如果段长过大，为其分配很大的连续空间会很不方便
      - 段式管理会产生外部碎片