# 综合知识

## 知识点及分值分布



## 计算机组成原理与体系结构

1.1 计算机体系结构分类：Flynn分类法

|  |  |  |  |
| --- | --- | --- | --- |
| 体系结构类型 | 结构 | 特征 | 代表 |
| 单指令流单数据流SISD | 控制部分：1  处理器：1  主存：1 |  | 单片机系统 |
| SIMD | 控制部分：1  处理器：多个  主存：多个 | 各处理器异步执行同一条指令 | 并行计算、向量计算、GPU |
| MISD | 控制部分：多个  处理器：1  主存：多个 | 不可能，没有实际意义，理论模型 |  |
| MIMD | 控制部分：多  处理器：多  主存：多 | 最常见！ | 常见的计算机，服务器 |

指令流：控制部分1就是S，多个就是M；

数据流：处理器1就是S，多就是M；

只有SISD的主存是1个，其他都是多个。

1.2 指令系统分类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 指令 | 寻址方式 | 实现方式 | 其他 |
| CISC（复杂） | 数量多，使用频率差别大，可变长格式 | 多种 |  | 研制周期长 |
| RISC（精简） | 数量少，频率接近，定长 | 少 | 通用寄存器 | 简单 |

CISC是为了完成某种任务（如天气预报系统）专门设计的计算机，他的指令更加专业且全面，所以复杂。

RISC为了更通用，所以精简了常用的东西。寄存器速度快！

1.3 层次化存储结构

cpu寄存器<--->cache缓存<--->内存<--->外存（硬盘、光盘、u盘）

速度：寄存器>cache>内存>外存

容量：寄存器<cache<内存<外存

寄存器与cache的成本太高，不能太多；多层结构为了性价比，cache存一些常用的。

1.4 Cache概念

cache在cpu与主存之间，为了提高gpu的数据输入输出速度提出的。主要是计算存取的平均周期时间：

t = h\*t1 + (1-h)\*t2

t是“cache+主存”的平均周期，t1是cache的，t2是主存的，h是cache的访问命中率。

1.5 局部性原理

计算机在某时段会集中的使用某些指令、读取某些相同空间的数据。

时间局部性 + 空间局部性。

1.6 主存

（1）分类

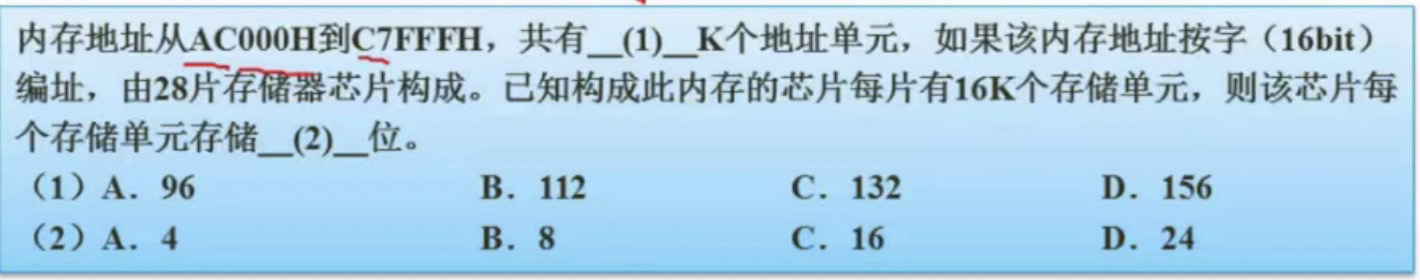
存储器分为：随机存储器RAM和只读存储器ROM。

主存（内存）就是RAM：断电就清除。

ROM断电不丢失，外存就是ROM。

1. 编址

8\*4的存储器就是8个地址，每个地址存4个bit位。



地址个数：大-小+1，16进制的借位：

C7FFFH + 1 = C8000H，C8000H - AC000H = 1C000H = 112

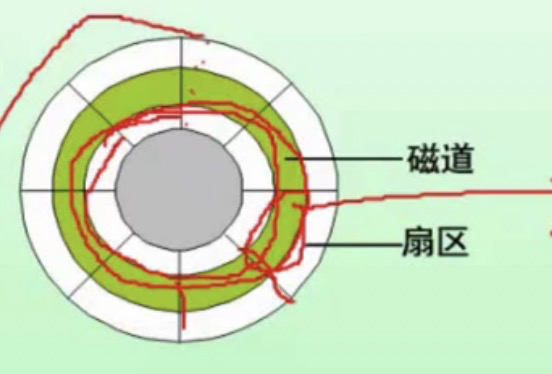
8借位+16 = 24，C=12。16 + 12 \* 8 = 16 + 96 = 112.

K = 1024B = 2的10次方。

112K= 16K \* 7，那28的分布就是7 \* 4个，已知每个地址16bit，一共4个芯片组成，每个也就是16 / 4 = 4bit。

或者直接点：112K \* 16 = 28 \* 16K \* x，求x。

1.7 磁盘结构与参数

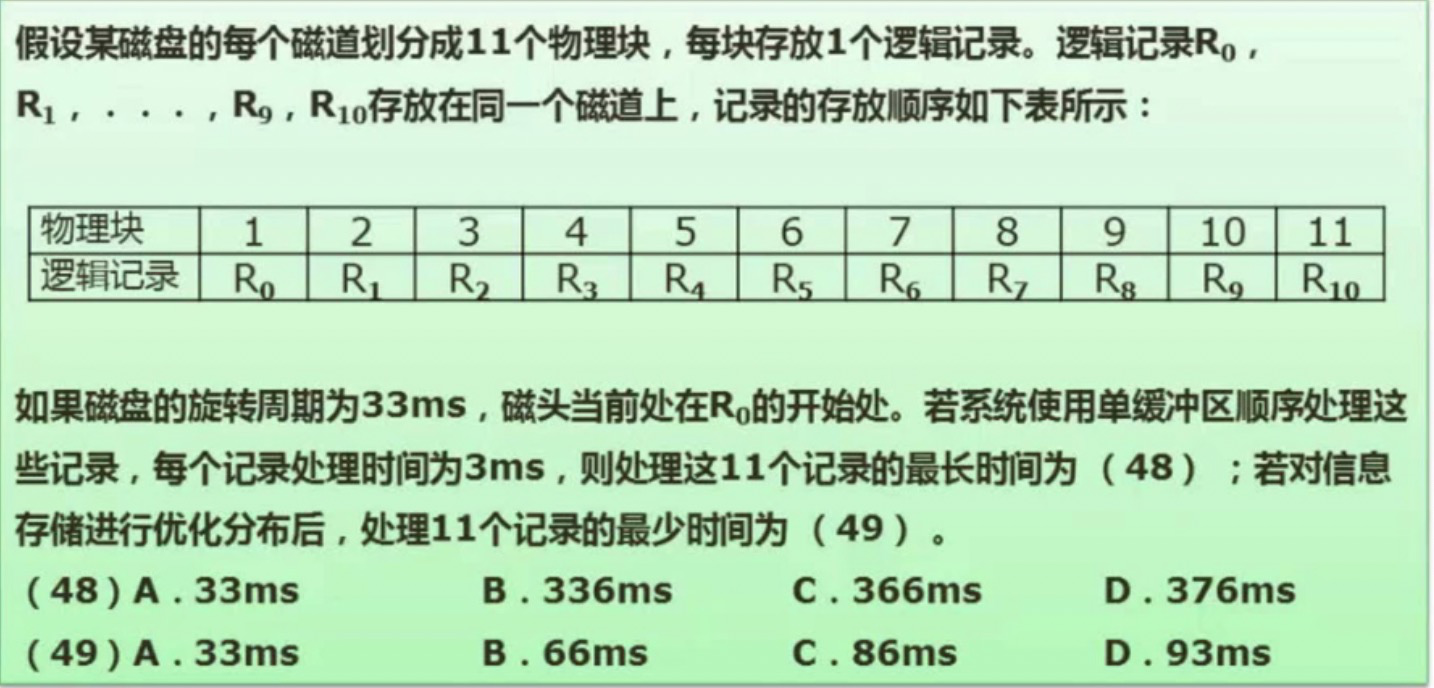


数据存在盘面上，读取是需要磁头寻找到对应的磁道上进行读取。

存取时间 = 寻道时间 + 等待时间

也就是先定位磁道，再旋转到具体的某个扇区上。

例题：



（1）一圈33ms，转一个扇区3ms。

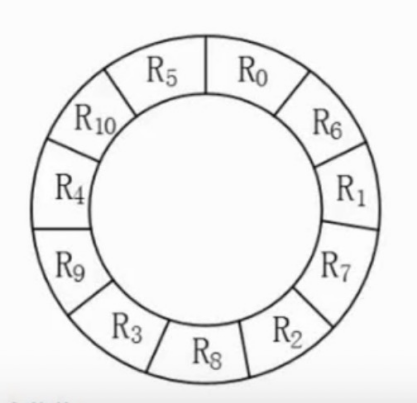
R0为例，3ms处理完后，到达R1开始的位置，但是单缓存区处理R0需要3秒，R0还没处理完，也就是R0还没完成真正的读取。也就是磁盘-->cache-->cpu的过程。可以理解为每个扇区读取的3ms就是读到缓存区的过程，处理过程就是继续读到cpu进行处理的时间。所以R1是不能计算读取的，又因为要顺序读取，且磁盘旋转是不会暂停的，所以要等一圈。

R0的时间：一圈 + 一条记录的时间 = 33ms + 3ms = 36.

R1-R9一样，R10就是3ms + 3ms = 6ms

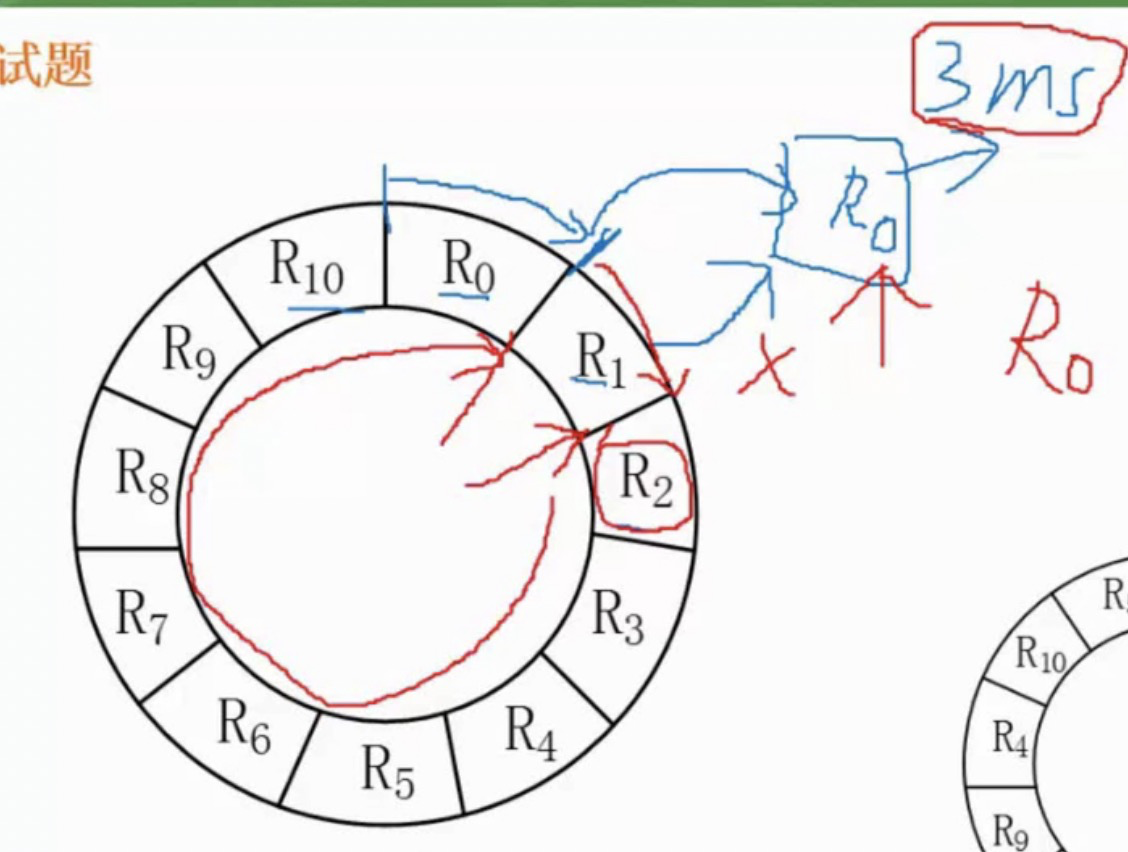
共36 \* 10 + 6 = 366ms。

1. 先求优化位置，6ms也就是2个扇区做一次逻辑连续：



6 \* 11 = 66ms。

考试就是换换时间与扇区个数！



## 系统配置与性能评价

2.1 系统性能设计

1. 性能指标
2. 字长与数据通路宽度：

计算机字长32位或64位代表PC在单次操作中能完成对32或64个bit的数据的处理。也就是一个字是多少位。

1. 主存容量和存取时间；
2. 运算速度：

主要是主频（cpu的时钟周期），决定cpu每次操作的频率快慢；

衡量速度的指标：MIPS，每秒钟可执行的指令的条数，百万级别。

MIPS = 指令条数 /（执行时间 \* 10^6）= 主频/CPI = 主频\*IPC

MFLOPS = 浮点数操作次数 /（执行时间 \* 10^6）

1. 吞吐量与吞吐率；
2. 响应时间与完成时间；
3. 兼容性。

不太重要，因为考核不太方便。下面重要，因为是计算题！

1. 阿姆达尔解决方案

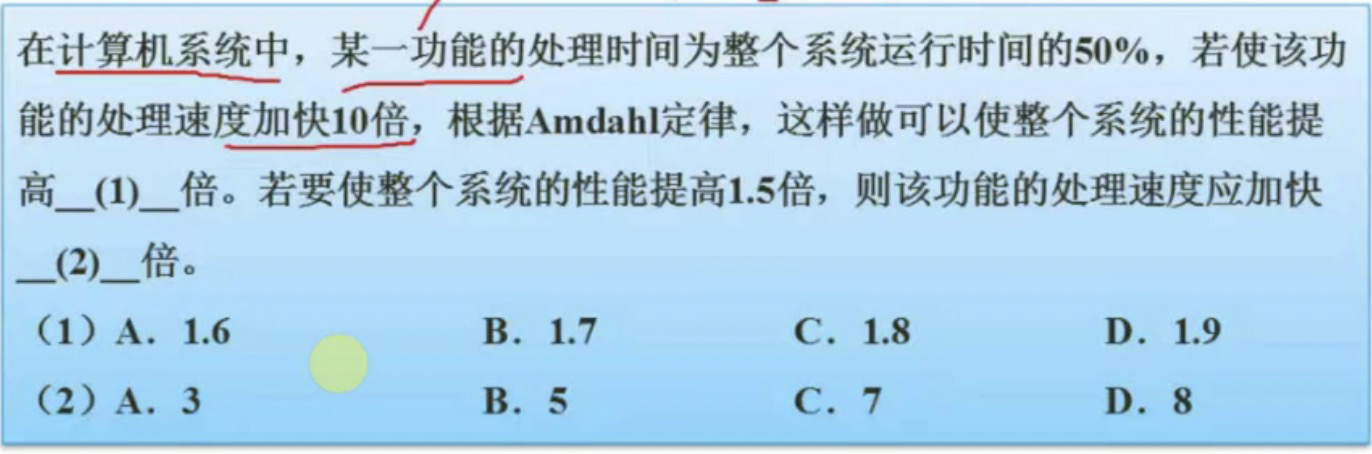
系统的由各组件组成，整体提升各组件效率成本高！28定律，系统中80%的时间是在用20%的功能，这样提升才有效！

加速比计算公式：



Tp是不优化组件的整个任务时间，Ti是优化后的整个任务时间。

不记，直接用数值代入法：



设总时间10，某功能是5，剩下5，加速10倍变为0.5，总时间5.5，（1）加速比为10/5.5=1.8

1. 设加速x倍，10/（5/x + 5）=1.5，x=3

2.2 性能评价方法（选择题，描述该方法的特点）

按时间顺序：

1. 时钟频率法：古老，386与486时代，只考虑cpu处理速度，当时cpu是瓶颈，所以有点道理。但现在并不是cpu完全决定了。
2. 指令执行速度法：单位时间能处理的指令数量。MIPS，但是都是用加法指令来算的，各指令的时间还是有差异的。
3. 等效指令速度法（吉普森混合法）：计算各个指令大致的使用比例与时间，得一个综合时间。但依然局限在了运算能力上。
4. PDR（数据处理速率法）：运算能力 + 存储的交互性。
5. CTP（综合理论性能法91年）：考虑每个处理部件的有效计算率。依然是cpu + 存储。但是瓶颈不在这了，而是IO，硬盘读取慢，换个SSD就好了。
6. 基准程序法：就是各个维度的指标都跑相同的基准程序，横向对比，就是跑分软件！

2.3 性能监测方法

如显示实时的性能。

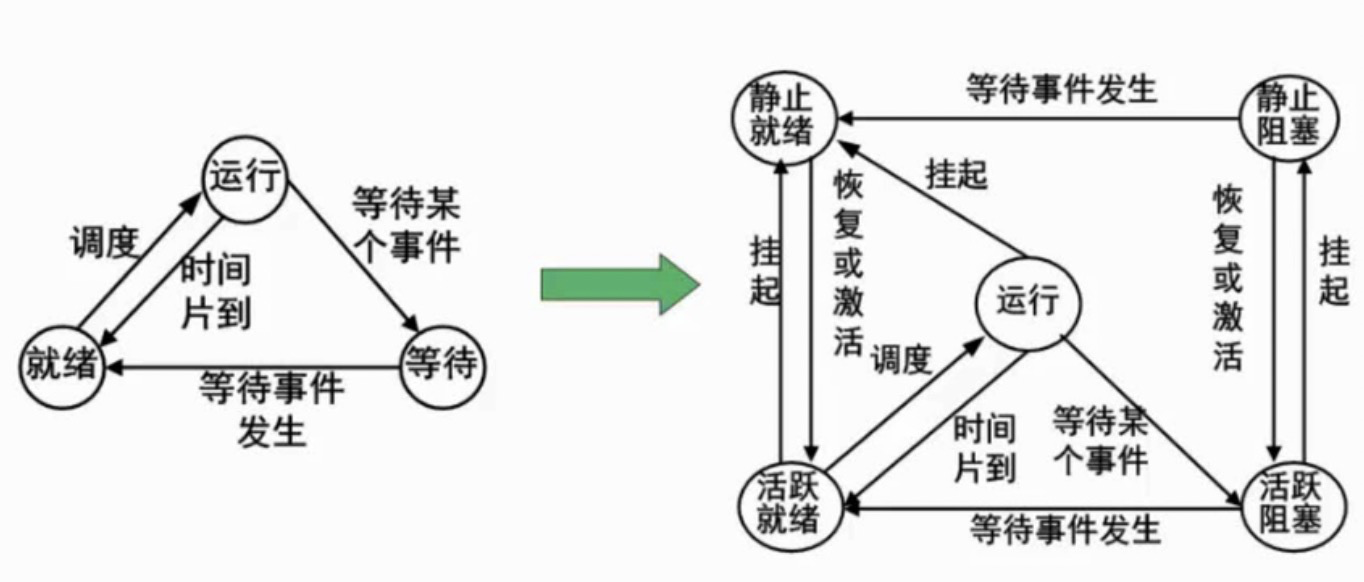
软件监控：消耗当前PC较多系统资源；

硬件监控：适用于高负载的计算机系统，独立的硬件设备，不影响当前PC。

## 操作系统

3.1 进程管理---进程状态转换图

给进程定义的几种状态，便于分配资源，加以管理。



1.三态模型

1. 运行：当前进程需要的所有资源都ok，并且有了cpu资源。
2. 就绪：其他都ok，只缺cpu。
3. 等待：除了cpu没有，其他资源也不够。

运行时缺资源，变为等待cpu资源收回；其他资源齐了，变为就绪，排队等待cpu资源。等待不能直接到运行！

时间片：一会进程可能时间很长，如果cpu都给它，等它完成再执行其他，那太耽误了------->时间片轮询策略。

就是每个进程cpu分配一个时间片，如果没完成，就退到就绪态重新排队。

1. 五态模型

3种不够了，比如人为想先挂起一个进程，无法满足，发展出5态。

运行、活跃就绪、活跃阻塞---对应原来的---运行、就绪、等待。

挂起操作：人为暂停一个进程。

挂起操作后，进入静止就绪状态。如听歌时来个电话，人为暂停去接电话。

静止就绪---执行恢复操作---活跃就绪态，静止就绪不能直接恢复到运行。

进入运行只有一个入口：就是在活跃就绪态排队！

静止阻塞与活跃阻塞同理。

3.2 进程管理---前驱图（常考、与PV操作结合）

要完成的一系列活动，先后的约束关系。

确定哪些任务能并行，哪些任务有先后关系。

3.3 进程管理---进程的同步与互斥

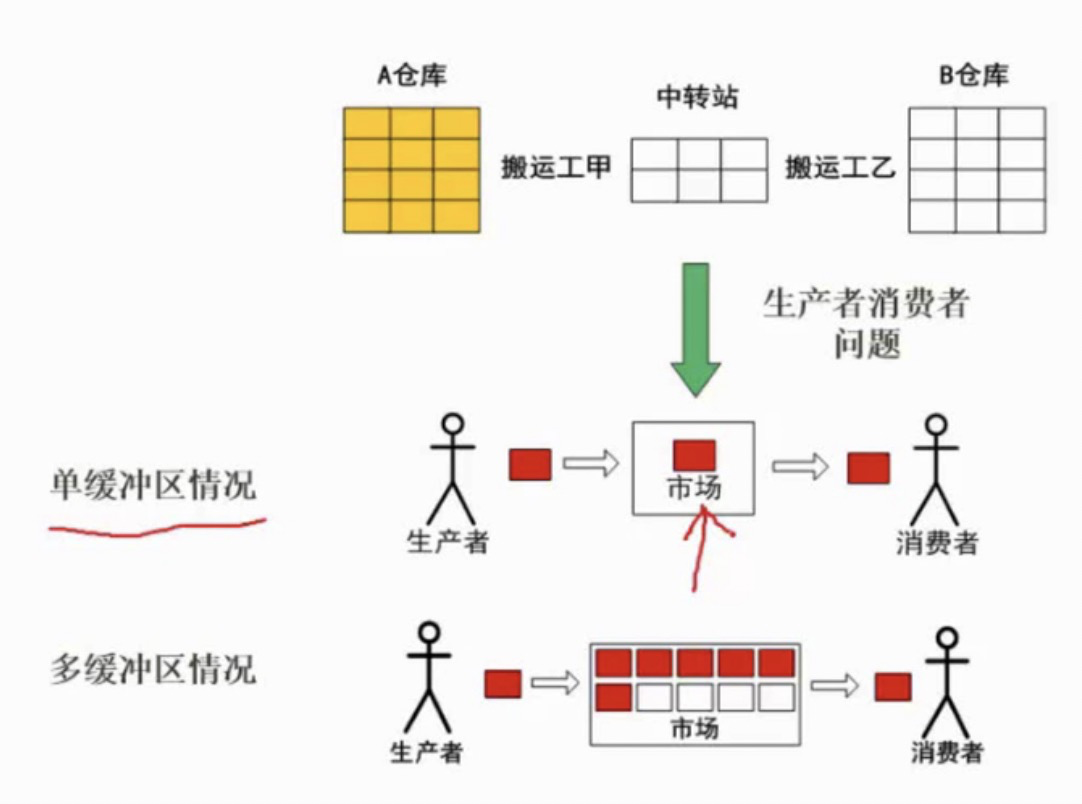
互斥：同一时刻，只允许一个进程使用某个资源。如千军万马过独木桥，独木桥就是资源。一个进程占用资源，其他进程要等待。

同步：速度快的进程在速度差太大时，停下来等等慢的。

生产者-消费者问题：

互斥：市场同一时刻只允许一个人进去操作；

同步：市场放满后，生产者要停下来等消费者。



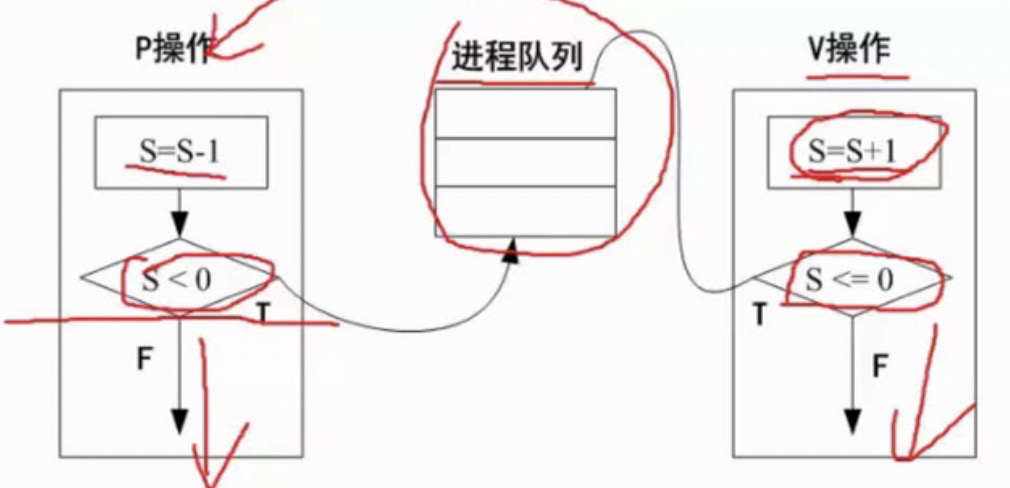
同步与互斥不是反义词。同步-异步，互斥-共享。

3.4 进程管理---PV操作（最难）

临界资源：进程需要互斥访问的资源，如打印机。

临界区：每个进程访问临界资源的那段代码。

信号量：PV操作的特殊变量。



P(S)：自减1，小于0，丢入进程队列，处于就绪状态，阻塞起来；

否则继续执行当前的程序进程。

V(S)：自+1，小于等于0，从队列取一个进程唤醒；否则继续执行当前程序。