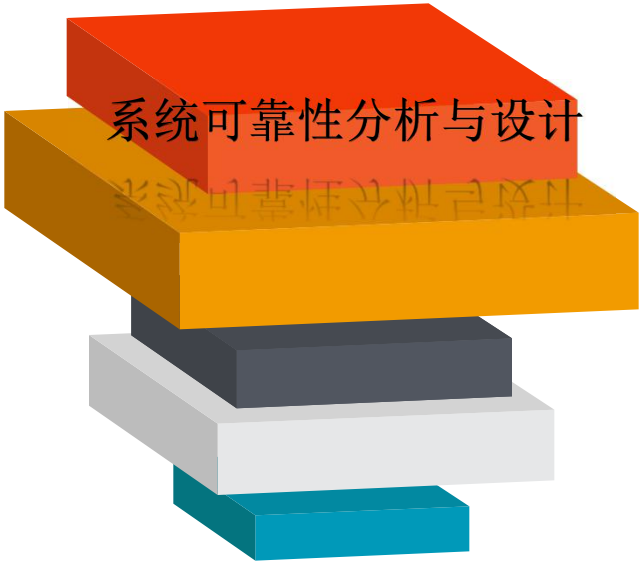
系统架构设计师

DESIGNER:王川林

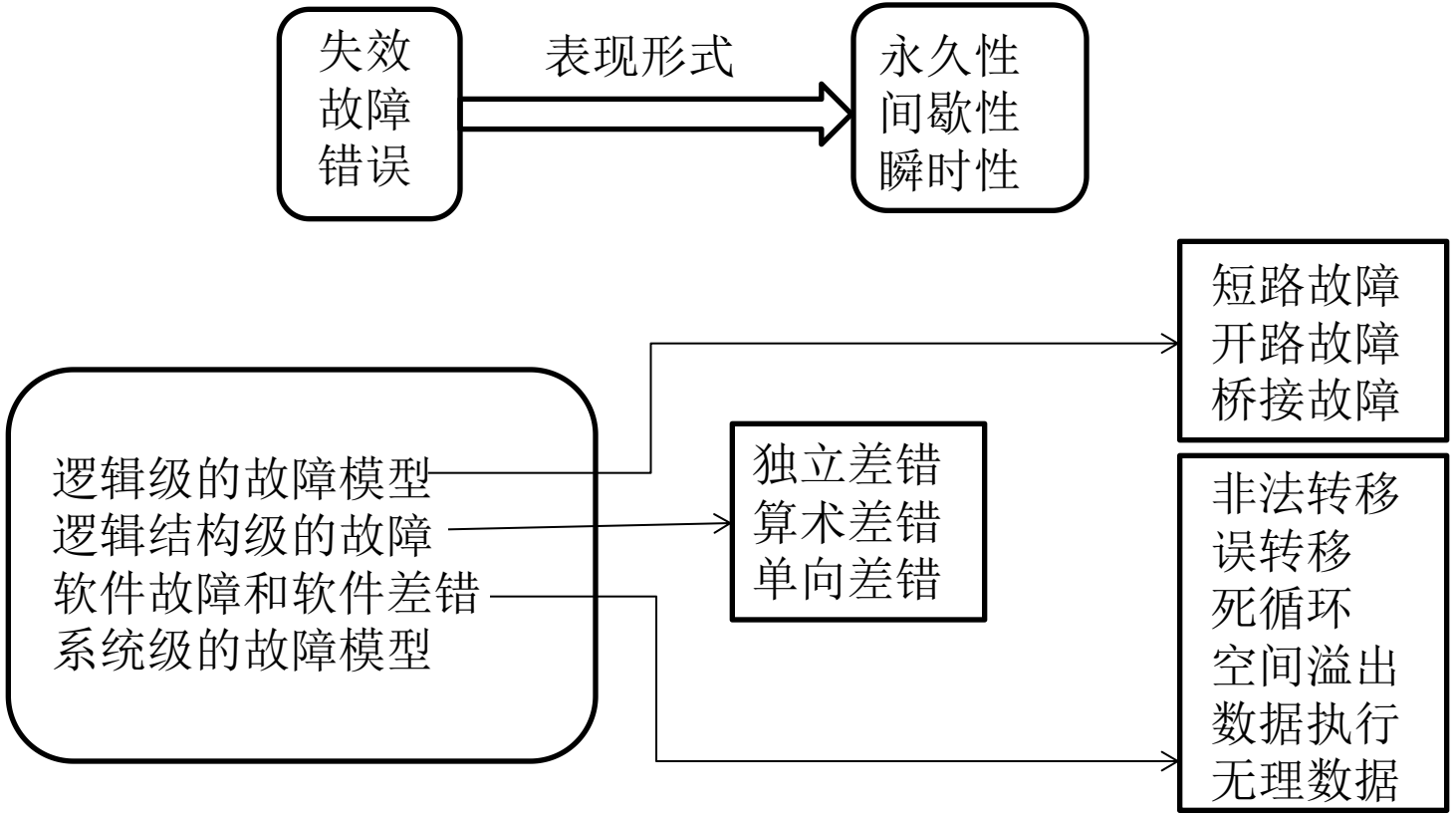
直播课4

第六章 系统可靠性分析与设计



课程内容

- 系统故障模型 ★
- 系统可靠性分析 ★ ★ ★ ★
- 系统容错 ★ ★ ★ ★


失效
故障
错误
表现形式
永久性
间歇性
瞬时性
逻辑级的故障模型
逻辑结构级的故障
软件故障和软件差错
系统级的故障模型
独立差错
算术差错
单向差错
短路故障
开路故障
桥接故障
非法转移
误转移
死循环
空间溢出
数据执行
无理数据

- 平均无故障时间（MTTF）
- 平均故障修复时间（MTTR
- ）
- 平均故障间隔时间（MTBF
）

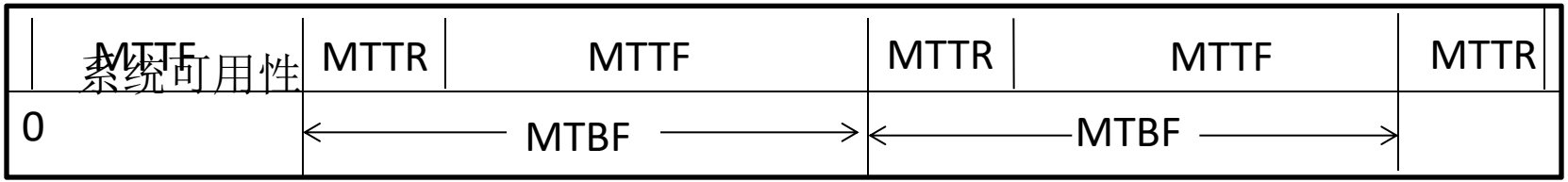
MTTF=1/λ， λ为失效率

MTTR=1/μ， μ为修复率

MTBF=MTTR+MTTF

MTTF/（MTTR+MTTF）×100%

系统可用性



在实际应用中，一般MTTR很小，所以通常认为MTBF≈MTTF

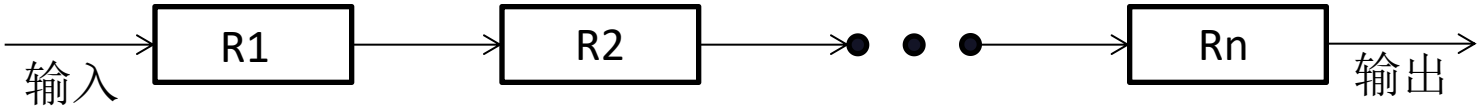
系统可靠性是系统在规定的时间内及规定的环境条件下，完成规定功能的能力，也就是系统无故障运行的概率。

系统可用性是指在某个给定时间点上系统能够按照需求执行的概率

提高可靠性需要强调减少系统中断（故障）的次数，提高可用性需要强调减少从灾难中恢复的时间

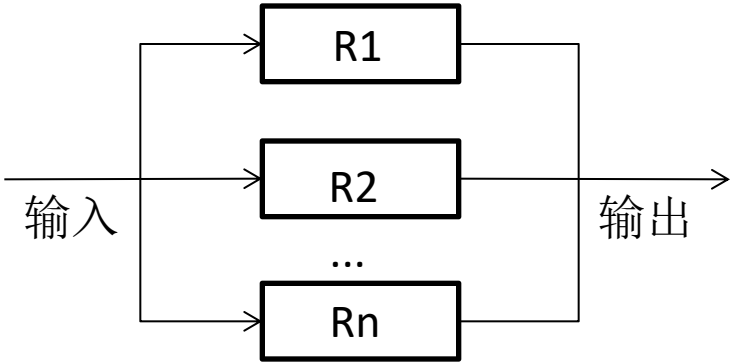
例如，假设同一型号的1000台计算机，在规定的条件下工作1000小时，其中有10台出现故障。
这种计算机千小时的可靠度R为（1000-10）/1000=0.99；
失效率为10/（1000×1000）=1×10^{-5}；
MTTF=1/（1×10^{-5}）=10^5小时。

系统可靠性分析—串联系统与并联系统



$R=R_1 \times R_2 \times \ldots \times R_n$

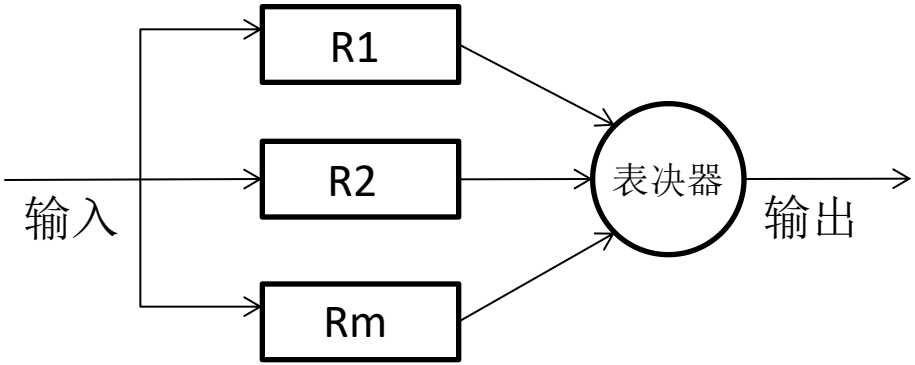
$\lambda=\lambda_1+\lambda_2+\ldots+\lambda_n$



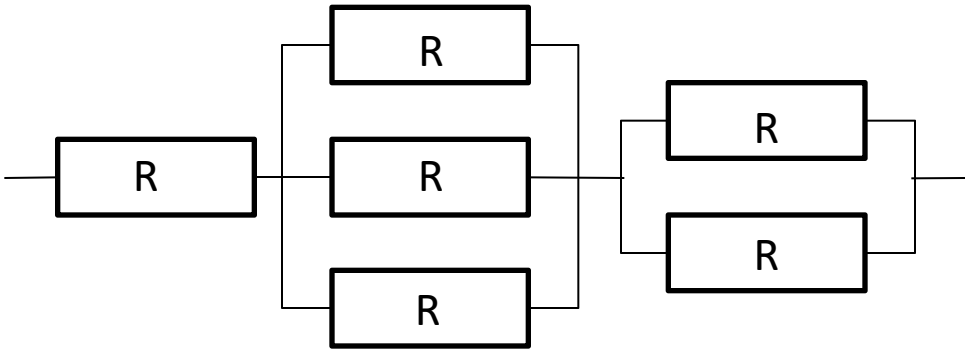
$R=1-(1-R_1) \times (1-R_2) \times \ldots \times (1-R_n)$

$$\mu = \frac{1}{\frac{1}{\lambda}\sum_{j=1}^{n}\frac{1}{j}}$$

系统可靠性分析—模冗余系统与混合系统



$$R = \sum_{i=n+1}^{m} C_m^j \times R_0^i (1-R_0)^{m-i}$$



R×（1-（1-R）^{3}）×（1-（1-R）^{2}）

避错技术
查错技术
改错技术
容错技术

静态冗余(屏蔽冗余、被动冗余；模冗余系统)
动态冗余(主动冗余；备份冗余、集群系统)
混合冗余

结构冗余(硬件冗余、软件冗余)
信息冗余(校验码)
时间冗余(重复多次进行相同的计算)
冗余附加(为实现上述冗余技术所需的资源和技术
)

故障检测
故障定位
故障恢复

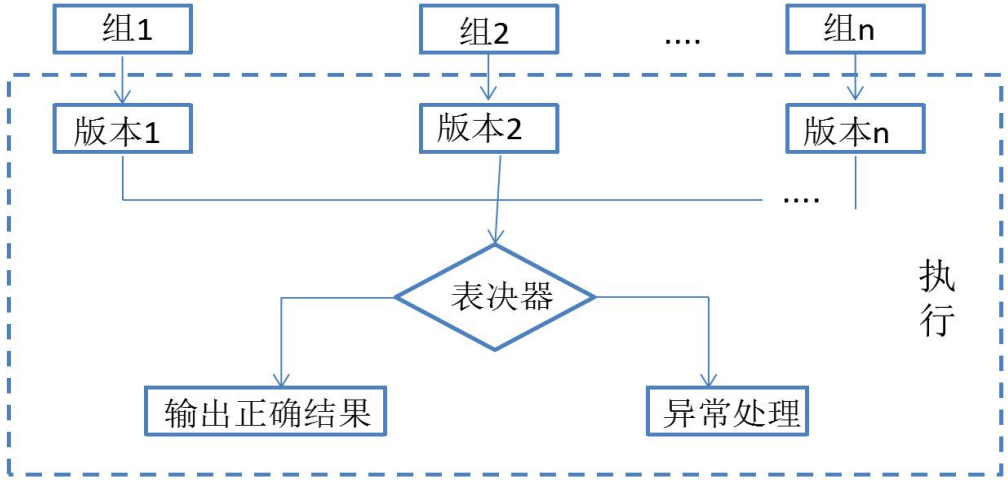
处理故障的步骤：

故障检测
故障屏蔽
故障限制
复执
故障诊断
系统重配置
系统恢复
系统重新启动
修复
系统重组合

- 前向恢复：使当前的计算继续下去，把系统恢复成连贯的正确状态，弥补当前状态的不连贯情况
- 后向恢复：系统恢复到前一个正确状态，继续执行

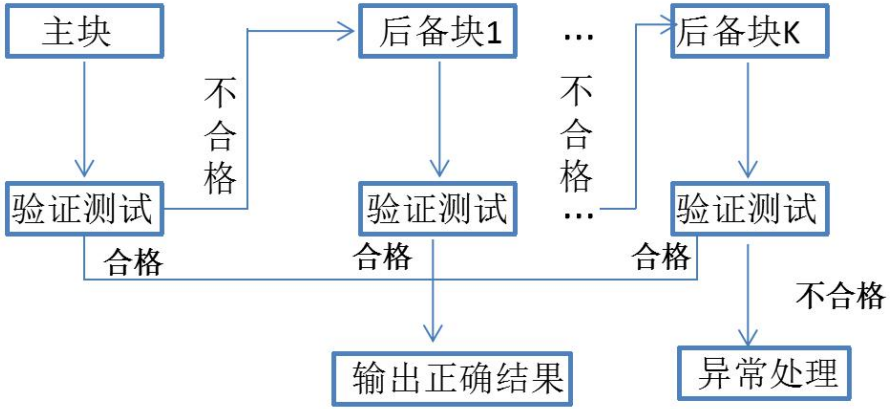
- 后向恢复简单地把变量恢复到检查点的取值；前向恢复将对一些变量的状态进行修改和处理，且这个恢复过程将由程序设计者设计
- 前向恢复适用于可预见的易定义的错误；后向恢复可屏蔽不可预见的错误

系统容错—软件容错—N版本程序设计



- ✓ 与通常软件开发过程不同的是，N版本程序设计增加了三个新的阶段：相异成份规范评审、相异性确认，背对背测试
- ✓ N版本程序的同步、N版本程序之间的通信、表决算法（全等表决、非精确表决、Cosmetie表决）、一致比较问题、数据相异性

系统容错—软件容错—恢复块方法

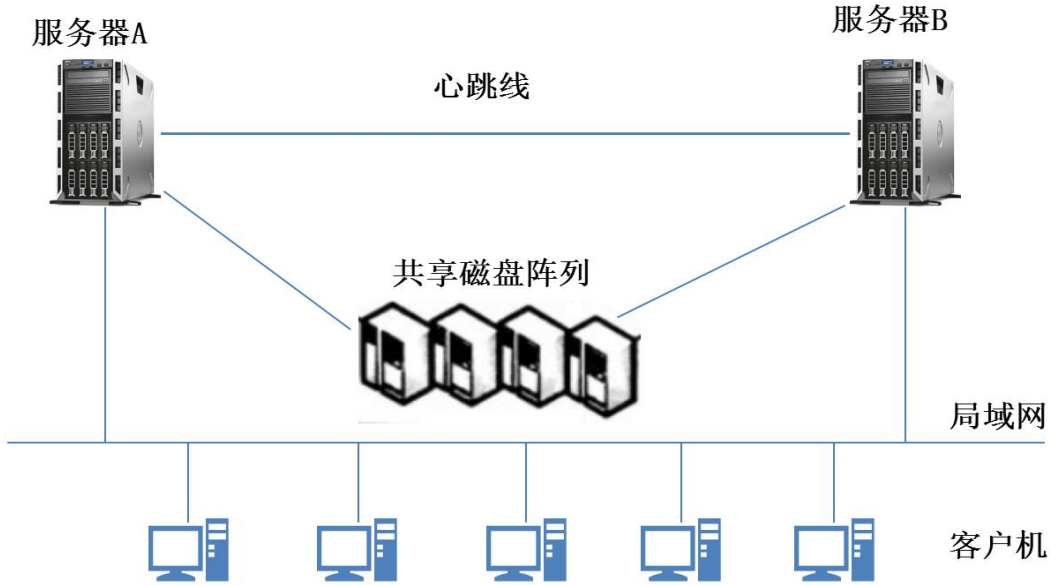


- 设计时应保证实现主块和后备块之间的独立性，避免相关错误的产生，使主块和备份块之间的共性错误降到最低程度。
- 必须保证验证测试程序的正确性。

	恢复块方法	N版本程序设计
硬件运行环境	单机	多机
错误检测方法	验证测试程序	表决
恢复策略	后向恢复	前向恢复
实时性	差	好

- 对于程序中存在的错误和不一致性，通过在程序中包含错误检查代码和错误恢复代码，使得一旦错误发生，程序能撤销错误状态，恢复到一个已知的正确状态中去

- 实现策略：错误检测、破坏估计、错误恢复



双机热备模式（主系统、备用系统）
双机互备模式（同时提供不同的服务，心不跳则接管）
双机双工模式（同时提供相同的服务，集群的一种）

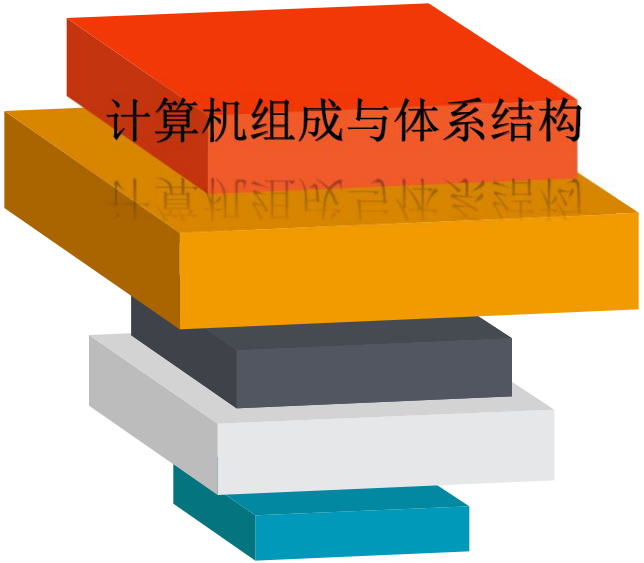
高性能
主机系统

PK

集群系统

可伸缩性
高可用性
可管理性
高性价比
高透明性

第七章 计算机组成与体系结构



Flynn分类法 ★★

CISC与RISC ★★

存储系统 ★★★

嵌入式系统 ★★★★

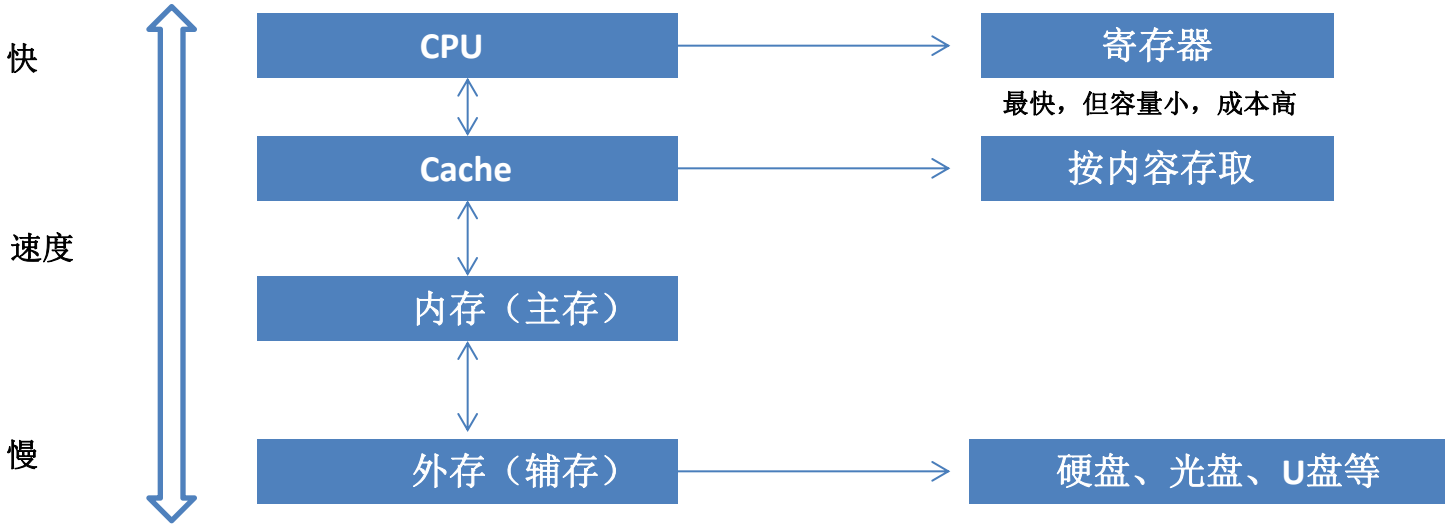
计算机体系结构分类—Flynn

体系结构类型	结构	关键特性	代表
单指令流单数据流SISD	控制部分：一个 处理器：一个 主存模块：一个		单处理器系统
单指令流多数据流SIMD	控制部分：一个 处理器：多个 主存模块：多个	各处理器以异步的形式执行同一条指令	并行处理机 阵列处理机 超级向量处理机
多指令流单数据流MISD	控制部分：多个 处理器：一个 主存模块：多个	被证明不可能 至少是不实际	目前没有，有文献称流水线计算机为此类
多指令流多数据流MIMD	控制部分：多个 处理器：多个 主存模块：多个	能够实现作业、任务、指令等各级全面并行	多处理机系统 多计算机

指令系统类型	指令	寻址方式	实现方式	其它
CISC（复杂）	数量多，使用频率差别大，可变长格式	支持多种	微程序控制技术（微码）	研制周期长
RISC（精简）	数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器只有Load/Store操作内存	支持方式少	增加了通用寄存器；硬布线逻辑控制为主；适合采用流水线	优化编译，有效支持高级语言

DESIGNER:

层细化存储结构



- Cache的功能：提高CPU数据输入输出的速率，突破冯.诺依曼瓶颈，即CPU与存储系统间数据传送带宽限制。
- 在计算机的存储系统体系中，Cache是访问速度最快的层次。
- 使用Cache改善系统性能的依据是程序的局部性原理。

如果以h代表对Cache的访问命中率，t_1表示Cache的周期时间，t_2表示主存储器周期时间，以读操作为例，使用“Cache+主存储器”的系统的平均周期为t_3,则：

$$t_3=h \times t_1+(1-h) \times t_2$$

其中，（1-h)又称为失效率（未命中率）。

DESIGNER:

局部性原理

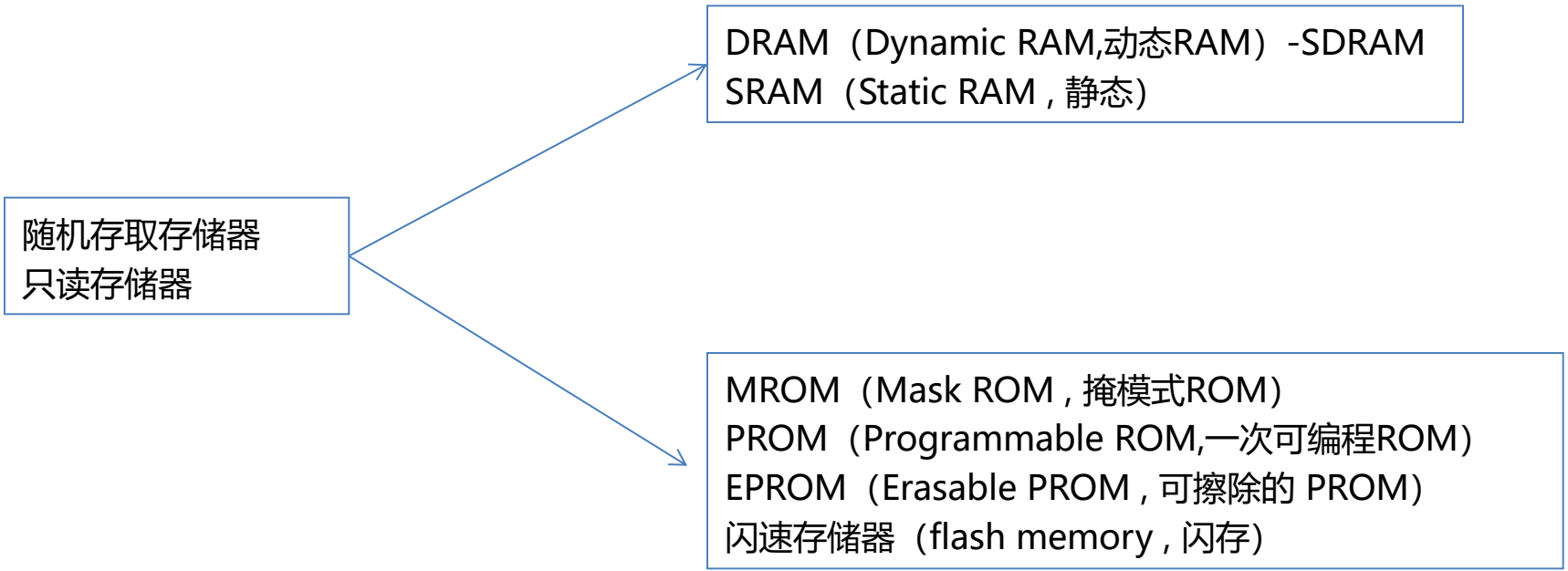
- 时间局部性：指程序中的某条指令一旦执行，不久以后该指令可能再次执行，典型原因是由于程序中存在着大量的循环操作。
- 空间局部性：指一旦程序访问了某个存储单元，不久以后，其附近的存储单元也将被访问，即程序在一段时间内访问的地址可能集中在一定的范围内，其典型情况是程序顺序执行。
- 工作集理论：工作集是进程运行时被频繁访问的页面集合。

例：

```
int ij,s=0,n=10000;
for(i=1;i<=n;i++)
        s+=j;
printf(“结果为：
%d”,s)
```

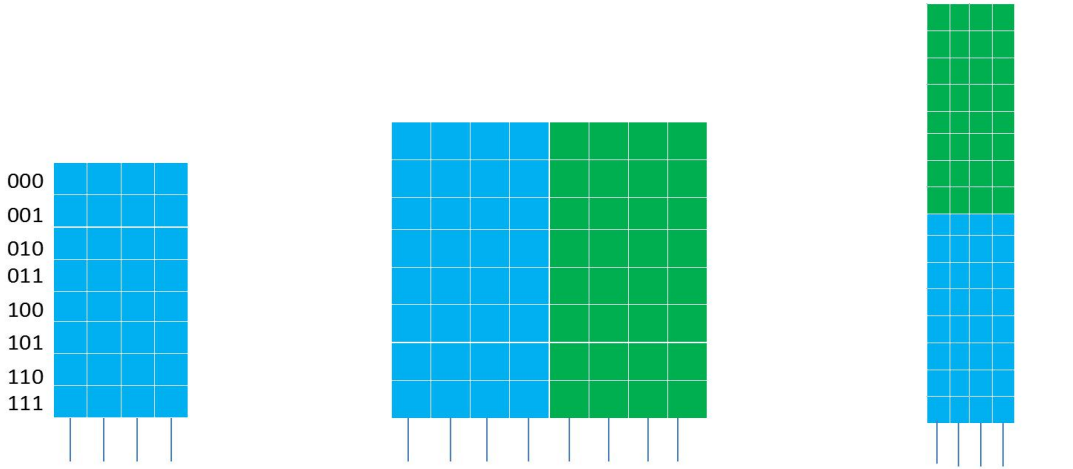
DESIGNER:

主存-分类



DESIGNER:

主存—编址

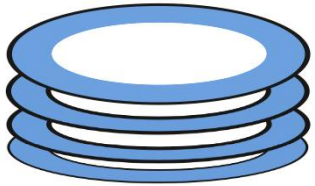


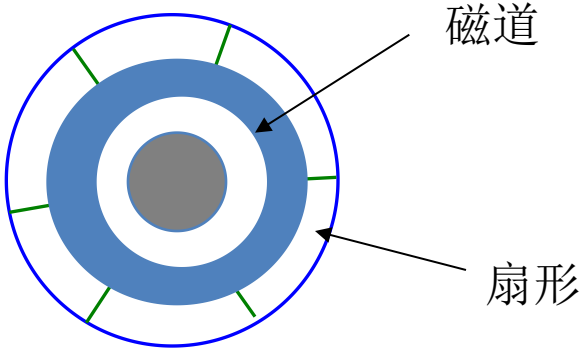
内存地址从AC000H到C7FFFG，共有 （1） K个地址单元，如果该内存地址按字（16bit）编址，由28片存储器芯片构成。已知构成此内存的芯片每片有16K个存储单元，则该芯片每个存储单元存储（2）位。

（1） A.96 B.112 C.132 D.156

（2） A.4 B.8 C.16 D.24

磁盘结构与参数





★存取时间=寻道时间+等待时间（平均定位时间+转动延迟）

注意：寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用时间

DESIGNER:

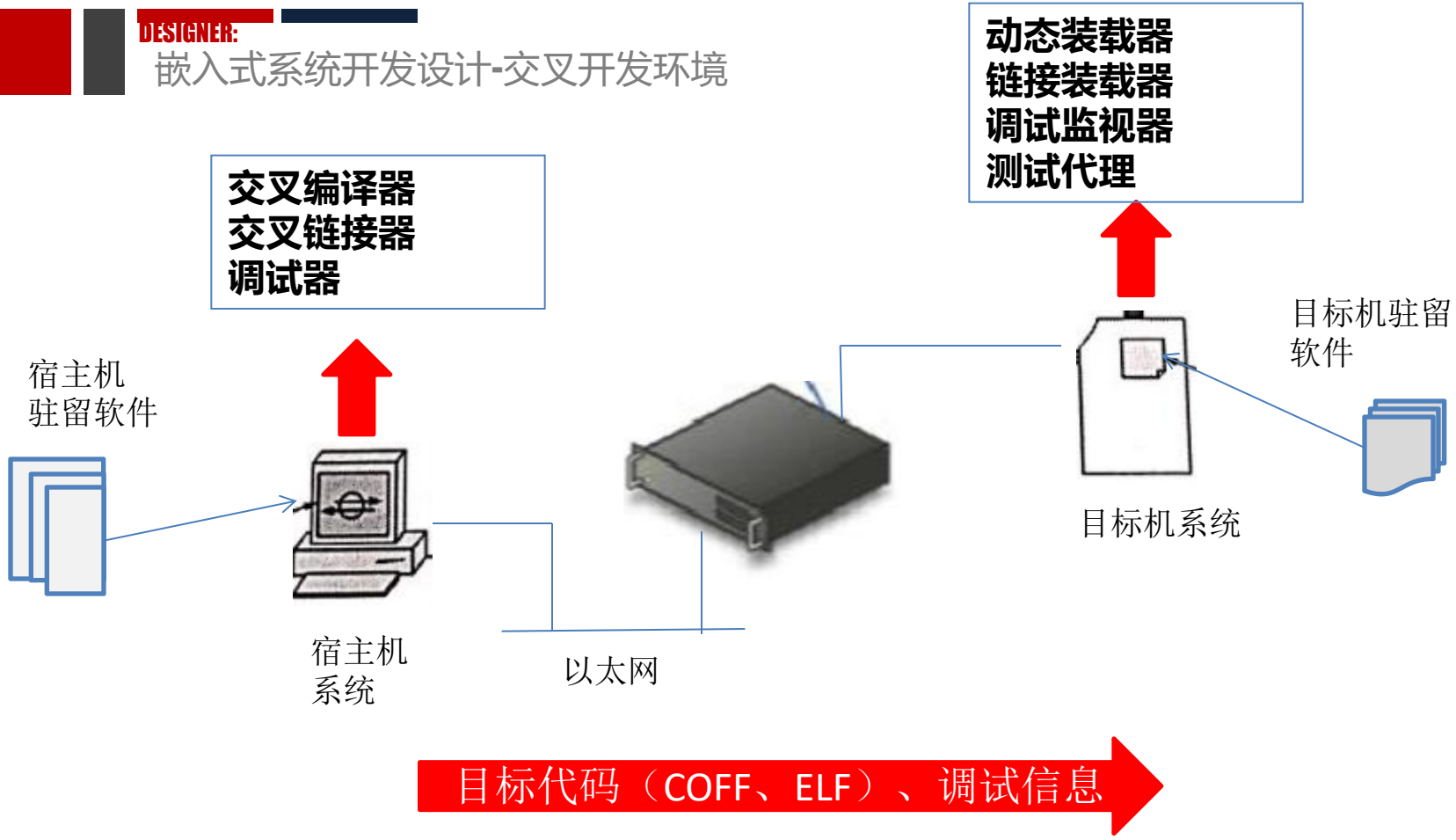
总线

一条总线同一时刻仅允许一个设备发送，但允许多个设备接收

总线的分类

- 数据总线（Data Bus):在CPU与RAM之间来回传送需要处理或是需要储存的数据。
- 地址总线（Address Bus):用来指定在RAM （Random Access Memory)之中储存的数据的地址。
- 控制总线（ Control Bus):将微处理器控制单元（Control Unit)的信号，传送到周边设备，一般常见为USB Bus 和1394 Bus .

嵌入式系统开发设计-交叉开发环境



嵌入式系统初始化过程

片级初始化 → 板级初始化 → 系统初始化

片级初始化

完成嵌入式微处理器的初始化，包括设置嵌入式微处理器 核心寄存器和控制寄存器、嵌入式微处理器核心工作模式和嵌入式微处理器的局部总线模式等。片级初始化把嵌入式微处理器从上电时的默认状态逐步设置成系统所要求的的工作状态。这是一个纯硬件的初始化过程。

板级初始化

完成嵌入式微处理器以外的其他硬件设备的初始化。另外，还需设置某些软件的数据结构和参数，为随后的系统初始化和应用程序的运行建立硬件和软件环境。这是一个同时包含软硬件两部分再内的初始化过程。

系统初始化

该初始化过程以软件初始化为主，主要进行操作系统的初始化。BSP将对嵌入式微处理器的控制权转交给嵌入式操作系统，由操作系统完成余下的初始化操作，包含加载和初始化与硬件无关的设备驱动程序，建立系统内存区，加载并初始化其他系统软件模块，如网络系统、文件系统等。最后，操作系统创建应用程序环境，并将控制权交给应用程序的入口。

历年真题

某计算机系统输入/输出采用双缓冲工作方式，其工作过程如下图所示，假设磁盘块与缓冲区大小相同，每个盘块读入缓冲区的时间T为10μs，缓冲区送用户区的时间M为6μs，系统对每个磁盘块数据的处理时间C为2μs。若用户需要将大小为10个磁盘块的Docl文件逐块从磁盘读入缓冲区，并送用户区进行处理，那么采用双缓冲需要花费的时间为（5）μs，比使用单缓冲节约了（6）μs时间。

2016年(5)

A.100

B.108

C.162

D.180

2016年(6)

A.0

B.8

C.54

D.62

历年真题

【答案】B C

【解析】 单缓冲区执行时间：(10+6+2)+(10-1)*(10+6)=162μs双缓冲区执行时间：(10+6+2)+(10-1)*10=108μs 双缓冲比单缓冲节省162-108=54μs。

把磁盘数据送到缓冲区，花费的时间为Ts，缓冲区的数据送到用户区，花费的时间为Tm，用户进程对这批数据进行计算，花费的时间为Tc

单缓冲下总时间T=max(Ts+Tc)+Tm

双缓冲时间为max(Tc,Ts)

历年真题

试题四 论软件可靠性设计与应用

目前在企业中，以软件为核心的产品得到了广泛的应用。随着系统中软件部分比例的不断增加，使得系统对软件的依赖性越来越强，对软件的可靠性要求也越来越高。软件可靠性与其它质量属性一样，是衡量软件架构的重要指标。 软件工程中已有很多比较成熟的设计技术，如结构化设计、模块化设计、自顶向下设计等，这些技术为保障软件的整体质量发挥了重要作用。在此基础上，为了进一步提高软件的可靠性，通常会采用一些特殊的设计技术，即软件可靠性设计技术。在软件可靠性工程体系中，包含有可靠性模型与预测、可靠性设计和可靠性测试方法等。实践证明，保障软件可靠性最有效、最经济、最重要的手段是在软件设计阶段采取措施进行可靠性控制。

请围绕“软件可靠性设计与应用”论题，依次从以下三个方面进行论述。

1．概要叙述你参与实施的软件开发项目以及你所承担的主要工作。

2．简要叙述影响软件可靠性的因素有哪些。

3．阐述常用的软件可靠性设计技术以及你如何应用到实际项目中，效果如何。

历年真题

影响软件可靠性的主要因素如下：

1.运行环境：一样的软件在不同的运行环境下，其可靠性的表现是不一样的。

2.软件规模：

3.软件内部结构：内部结构越复杂，所包含的软件缺陷就越多。

4.软件的开发方法对软件的可靠性有显著的影响。

5.早前重视软件和可靠性并采取措施开发出来的软件，有较高的可靠性。

历年真题

软件可靠性设计的技术主要有：**容错设计、检错设计和降低复杂度设计**。

容错设计技术：失效后果特别严重的场合，可采用容错设计方法，常用的软件容错技术主要有恢复块设计，N版本程序设计和冗余设计，防卫式程序设计。

恢复块设计：一种动态故障屏蔽技术，采用后向恢复策略，提供具有相同功能的主块和几个后备块，一个块就是一个完整的程序段，主块首先投入运行，结束后进行验证测试，如果没有通过验证测试，系统经现场恢复后由一后备块运行。这一过程可以重复到耗尽所有的后备块，或某个程序故障行为超出了预料，从而导致不可恢复的后果。

历年真题

冗余设计：软件的冗余设计技术实现的原理是在一套完整的软件系统之外，设计一种不同路径、不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替代，从而维持软件系统的正常运行。

防卫式程序设计：不采用任何传统的容错技术就能实现软件容错的方法，防卫式程序设计的基本思想就是通过在程序中包含错误检测代码和错误恢复代码。使得一旦错误发生，程序能撤销错误状态，恢复到一个已知的正确状态中去。其实现策略包括错误检测、破坏估计和错误恢复是三个方面。

检错技术：在软件出现故障后能及时发现并报警，提醒维护人员进行处理。检错技术实现的代价一般低于容错技术和冗余技术，但是明显的缺点是：不能自动解决故障，在出现故障后不进行人工干预，将导致软件系统不能正常运行。

采用检错技术需要考虑的几个要素：检测对象、检测延时、实现方式、处理方式。

历年真题

1.检测对象：包含两个层次的含义，即检测点和检测内容。在设计时应考虑把检测点放在容易出错的地方和出错对软件系统影响较大的地方；检测内容选取那些有代表性的、易于判断的指标。

2.检测延时：从软件发生故障到被自检出来是有一定延时的，这段延时的长短对故障的处理时非常重要的。如果延时时长影响了故障的及时报警，则需要更换检测方式和检测对象。

3.实现方式：有判断返回结果、运行时间的长短、置状态标志位等方式。

4.处理方式：一般采取停止软件系统的运行的方式，但也有部分软件采取不停止或部分停止的方式，这需要视处理情况来确定。