1. **数字表示**

符号位（1）—阶码（k）—尾数（n）v=(-1)^s\*2^E\*M

规格化值：阶码不为全0或者全1， E=e-bias,bias=2的k-1次方-1；M=1+f=1.fn-1fn-2...f0

非规格化的值：阶码全为零，E=1-bias,M=f

特殊值：阶码全为1，尾数全0代表无穷大；位数不全0代表NaN

原码—符号位+正常表示，反码—符号位+负数时取反，补码=反码+1，负数的补码和原码可能相同（1100）

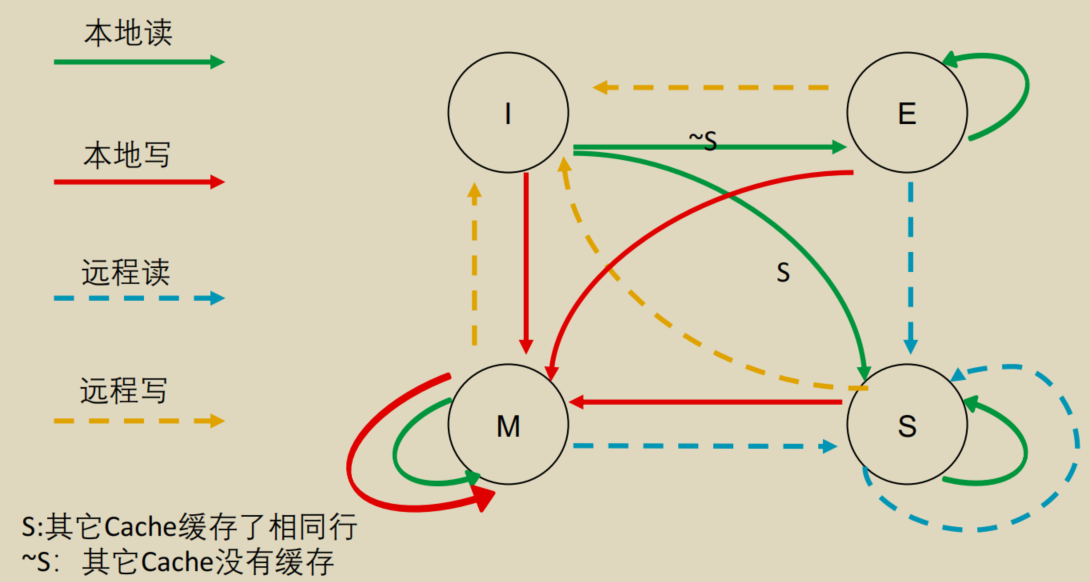
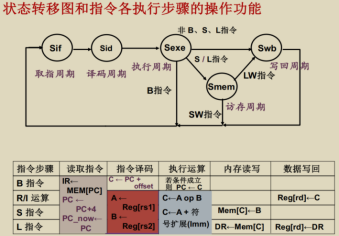
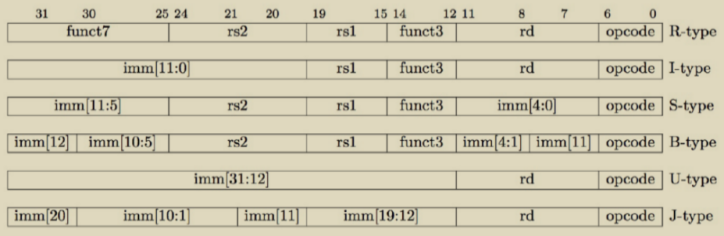
海明码：2的r次方大于等于m+r+1

EEE754 单精度浮点数占用 4 个字节/UTF-8 采用变长字符编码/合理提高码距可以提高发现错误的能力/0的原码和反码都有两个表示，补码就一个/对于100..00可能-x=x

布斯算法：00/11只右移 01：+被乘数再右移10：-被乘数再右移

步骤—012...n(位数) 操作 部分积 初始位（init0）

**指令格式**



R-格式：寄存器指令，指定指令中的3个寄存器,opcode固定add,and or,xor

I-格式：指令中包含立即数，用于带一个常数的算术指令以及加载指令 addi,andi,ori,slli(左移inst[24:20]位),srli(右移inst[24:20]位)，lb,lw

jalr(imm=signext(imm(11:0)),rd=pc+4,pc=(pc+imm)(31:1)-0)

S-格式：store指令sb sw

SB/B-格式：分支指令 bne beq imm=signext(imm[12:1]-0)

1. 格式：大立即数：两个指令

lui (load upper imm)(rd=(inst[31:12]-000000000000)（addi设置低20位，lui高20位，低20位负数的时候高20位+1）

auipc (add upper imm to PC,rd=(inst[31:12]-000000000000)+pc)

UJ/J-格式：唯一指令jal(signext(imm(20:1)-0),rd=pc+4,pc=pc+imm)

伪指令：j跳转，ret返回，li（rd=imm设置寄存器的值）

**执行步骤**

读取指令(IF), 从存储器读来指令并形成下条指令地址 (IM，总线，pc)/指令译码(ID), 指令译码，读寄存器堆为ALU准备数据 (REG)/执行运算(EXE),ALU 执行数据运算或计算存储器地址 (ALU)/存储器读写(MEM),完成存储器的读操作或者写操作 (DM，总线)

写回(WB), 写ALU的结果或存储器读出数据到寄存器堆(REG)/单周期CPU:CPI=1;多周期CPU:CPI=4;流水线CPU:CPI=1

信号含义：PCSel（下一条指令地址来源，是否跳转）ImmSel (指令类型，用于提取立即数)RFWEn （寄存器读写）BrUn （比较时是否无符号）BrEq （是否相等）BrLt （是否小于）BSel（ALU操作B来源） ASel （ALU操作数A来源）ALUSel (ALU运算操作符)MemRW （内存读写使能）WBSel（WB阶段写回寄存器的值来源，mem(sb/sw)/alu/pc+4(jal/jalr)）

流水线需要有装入时间和排空时间。装入时间是指第一个任务进入流水线到输出流水线的时间。排空时间是指第n个（最后一个）任务进入流水线到输出流水线的时间。

衡量流水线性能的主要指标：吞吐率：单位时间执行指令的数量，加速比：与串行执行时速度提高的比率

流水线设计中，流水线的最佳段数选择也是一个重要问题。

PC值多路选择器被放到IF段，这样做的目的是保证对PC值的写操作只出现在一个流水段内，否则当分支转移成功的时候，流水线中两条指令都试图在不同的流水段修改PC值，从而发生写冲突。

**结构冲突**，是指令在重叠执行的过程中，硬件资源满足不了指令重叠执行的要求，发生硬件资源冲突而产生的冲突。

方案一：stall方案二：增加硬件资源 ;寄存器结构冲突：两个独立的读端口和一个独立的写端口;内存结构冲突：将IM与DM分开/分不开就是用双端口存储器/等待一个时钟周期（气泡，相当于一条空指令）

**数据冲突**，是指在同时重叠执行的几条指令中，一条指令依赖于前面指令执行结果数据，但是又得不到时发生的冲突。

**写后读冲突**（RAW：Read After Write） 指令j的执行需要使用指令i的计算结果，但是当它们在流水线中重叠执行时，指令j可能在指令i将其计算结果写入之前就先行对保存该计算结果的寄存器进行了读操作，这样指令j读出的寄存器值就是错误的。

**写后写冲突**（WAW：Write After Write） 指令j和指令i的目的操作数相同，但是当它们在流水线中重叠执行时，指令j可能在指令i将其计算结果写入之前就先行对保存该计算结果的寄存器进行了写操作，这样就导致了寄存器写入顺序的错误，此时，目的寄存器的内容是指令i写入的值，而不是指令j写入的值。RISC-V指令流水不会发生WAW冲突，只有WB阶段写回寄存器。

**读后写冲突**（WAR：Write After Read） 指令j可能在指令i读取某个源寄存器的内容之前就对该寄存器进行了写操作，结果就是导致了指令i后来读取的值是错误的。 RISC-V流水不会发生WAR冲突，RISC-V在ID段完成所有的读操作，而在WB完成所有的写操作。

数据前传：结果可用就直接前传，无需保存到reg,不能解决所有数据冲突，lw,add这种需要等

判断：在译码阶段如果上一条指令是load且写入的寄存器和当前指令某一源寄存器相同，一旦发生就将流水线暂停一个时钟，将lw指令结果通过旁路送到ALU输入端，等价于插入一条nop

另一个解决方案：插入不相关的指令，汇编器调度，静态调度

数据冲突的动态调度：这种方法是由硬件动态调整指令执行顺序以减少暂停的影响，能够简化编译器设计。**动态调度并不能真正消除数据冲突**，但它能在出现数据冲突时尽量避免出现处理器暂停。而静态调度方法则是尽量通过分离有冲突问题的指令使它们不会导致冲突，从而减少暂停的影响。动态调度的主要思想：指令顺序发射——乱序执行——指令乱序流出，动态调度的问题：异常处理的不精确性。在采用动态调度方法的处理机中，在某条指令产生异常情况时，有可能出现其后面的指令已经执行完成的情况，这样异常处理是不精确的。

**解决方案**：结构冲突：增加资源，暂停流水线 ;数据冲突：暂停流水线，使用旁路技术，静态调度（汇编器调度）动态调度（处理器调度）

**控制冲突**，它是指流水线中的分支指令或者其他需要改写PC的指令造成的冲突，全局冲突（数据冲突为局部冲突）

1暂停流水线：一旦发现分支指令（ID结束后，EXE之前）就暂停流水线，即暂停该指令之后的所有指令，直到分支指令达到MEM段结束确定了新的PC值为止。造成性能的降低，减少暂停周期数：在流水线中尽早判断出分支转移是否成功，尽早计算出分支成功转移时的PC值（如分支目标地址），可以增加目标地址加法器和寄存器比较器

2预测-分支不成功：顺序执行下一条指令，预测失败后要清除错误启动的指令

3预测-分支成功，更复杂一些，因为要计算转移目的地址，预测失败后要清除错误启动的指令 （ID完事直接跳）

4动态预测，硬件根据上次分支的结果进行本次预测

分支目标缓冲技术（BTB）使用上次结果/动态预测（2位）连续两次预测错误时才改变预测方向，对多重循环，可进一步提高预测准确率

Stall/bubble:某阶段卡死：前面都1/0stall 后面一个0/1bubble 其余00

某段异常，前面都0/1后面一个0/1（就是当前条的输出）其余00

exe跳转：0/1 0/1 0/0 0/0

**各个阶段寄存器保存的值。**

IF/ID：PC，IR

ID/EXE：PC、A、B，inst ，所有控制信号

EXE/MEM：PC、ALU结果、B，inst ，MEM及WB

MEM/WB：DataD数据：PC+4、ALU结果、存储器读出的结果，AddrD地址（inst）,WB段控制信号

**异常**：CPU运行时会遇到突发的不正常事件 **（来自CPU为异常，来自外设为中断）**

指令执行过程中发生错误，取指令、指令译码、计算、访存/外部设备提出服务请求/多进程运行时与其他进程发生资源冲突

中断程序正常执行流程的事件：来自CPU，称为异常/来自外部设备，称为中断

异常处理的实现：

增加异常原因寄存器，保存异常的原因：错误的指令地址、错误的操作码、错误的运算结果、错误的数据地址，外部中断请求编号

发现异常：增加一个步骤，检查中断寄存器和异常原因寄存器

保存现场：mepc/当前的状态等（状态寄存器）/转异常处理程序（中断服务）/根据异常原因，寻址处理程序的入口（PC的来源）/跳转到处理程序执行（与转移指令相同） /处理完成后，返回主程序执行

流水线CPU异常处理

处理要求：指令流水线中有5条指令正在执行，判断发生异常的位置，保留发生异常的现场 ，执行异常处理程序；硬件：mepc，mcause

精确处理和非精确处理：精确异常处理：mepc中保存有发生异常指令的地址，操作系统处理简单，指令流水情况下实现比较复杂/非精确异常处理：mepc中保存当前PC或者近似的PC：由操作系统处理

**DRAM(动态存储器)**

存储器对性能的影响：假定某台计算机的处理器工作在：主频= 1GHz (机器周期为1 ns) CPI = 1.1 50% 算逻指令, 30% 存取指令, 20% 转移指令,再假定其中10% 的存取指令会发生数据缺失，需要

50个周期的延迟。 CPI = 理想CPI + 每条指令的平均延迟= 1.1 + (0.30 x 0.10 x 50)= 1.1 cycle + 1.5 cycle = 2.6

层次存储器系统：高速度：静态存储器速度高，设置较小容量的高速缓冲存储器/大容量：动态存储器价格适中，速度适中，可作为主存储器/低成本：磁盘存储器价格低廉，作为辅助存储器，暂存CPU访问频率不高的数据和程序，作为虚拟存储器的载体

程序运行的局部性原理：在一小段时间内，最近被访问过的程序和数据很可能再次被访问；在空间上这些被访问的程序和数据往往集中在一小片存储区；在访问顺序上，指令顺序执行比转移执行的可能性大**(**大约**5:1 )**

层次之间应满足的原则：一致性原则：处在不同层次存储器中的同一个信息应保持相同的值。包含性原则：处在内层的信息一定被包含在其外层的存储器中，反之则不成立**,**即内层存储器中的全部信息，是其相邻外层存储器中一部分信息的复制品。

破坏性读出：读操作后，被读单元的内容⼀定被清为零，必须把刚读出的内容⽴即写回去，通常称其为预充电延迟，它影响存储器的⼯作频率，在结束预充电前不能开始下⼀次读。

要定期刷新：在不进⾏读写操作时，DRAM 存储器的各单元处于断路状态，由于漏电的存在，保存在电容CS 上的电荷会慢慢地漏掉，为此必须定时予以补充，通常称其为刷新操作。刷新不是按字处理，⽽是每次刷新⼀⾏，即为连接在同⼀⾏上所有存储单元的电容补充⼀次能量。 刷新有两种常⽤⽅式：集中刷新，停⽌内存读写操作，逐⾏将所有各⾏刷新⼀遍；分散刷新，每⼀次内存读写后，刷新⼀⾏，各⾏轮流进⾏。或在规定的期间内，如 2 ms ，能轮流把所有各⾏刷新⼀遍。

快速分⻚组织的存储器：⾏、列地址要分两次给出，但连续地读写⽤到相同的⾏地址时，

也可以在前⼀次将⾏地址锁存，之后仅送列地址，以节省送地址的时间，⽀持这种运⾏⽅式的被称为快速分⻚组织的存储器。

主存储器的作用和连接：存储正处在运行中的程序和数据**(**或一部分**)** 的部件，通过地址数据控制三类总线与**CPU**、与其它部件连通。

**地址总线**：地址总线用于选择主存储器的一个存储单元（字或字节），其位数决定了能够访问的存储单元的最大数目，称为最大可寻址空间。例如，当按字节寻址时，20位的地址可以访问1MB的存储空间，32位的地址可以访问4GB的存储空间。

**数据总线**：用于在计算机各功能部件之间传送数据，数据总线的位数（总线的宽度）与总线时钟频率的乘积，与该总线所支持的最高数据吞吐（输入/输出）能力成正比。

**控制总线**：用于指明总线的工作周期类型和本次入/出完成的时刻。总线的工作周期可以包括主存储器读周期、主存储器写周期、I/O设备读周期、I/O设备写周期，即用不同的总线周期来区分要用哪个部件（主存或I/O设备）和操作的性质（读或写）；还有直接存

储器访问（DMA）总线周期等。

**SRAM(静态存储器)** 使用触发器存储，非破坏性读出，不用刷新，行列地址同时送，速度快，集成度低，发热量大，容量小，存储成本高，存储密度低，单位面积存储容量小，数据入/出共用管脚，能耗高

**DRAM(动态存储器)** 电容充放电来存储，破坏性读出，要刷新，行列地址分开送，速度慢，集成度高，发热量小，容量大、存储成本低，能耗低

**Cache(告诉缓冲存储器)**

定义：设置于主存和CPU之间的存储器，用高速的静态存储器实现，缓存了CPU频繁访问的信息。特点：高速：与CPU的运行速度基本匹配/透明：完全硬件管理，对程序员透明

时间局部性：最近被访问的信息很可能还要被访问。将最近被访问信息项装入到Cache中。

空间局部性：最近被访问的信息临近的信息也可能被访问。将最近被访问信息项临近的信息一起装入Cache中。

**Cache缺失的四类原因**：1必然缺失（Compulsory Miss）：开机或者是进程切换，首次访问数据块2容量缺失（Capacity Miss）活动数据集超出了Cache的大小3冲突缺失（Conflict Miss）多个内存块映射到同一Cache块/某一Cache组块已满，但空闲的Cache块在其他组4无效缺失，其他进程修改了主存数据

对策：必然缺失：预取；容量缺失，出现在Cache容量太小的时候，增加Cache容量可缓解缺失现象；冲突缺失：两块不同的内存块映射到相同的Cache块，对直接映射的Cache这个问题尤其突出，增加Cache容量有助于缓解冲突，增加相联的路数有助于缓解冲突

替换策略：1最近最少使用LRU满足程序局部性要求，有较高命中率，硬件实现复杂2先进先出FIFO满足时间局部性，实现比较简单3随机替换RAND，实现简单，命中率也不太低

多级Cache：1采用两级或更多级cache来提高命中率，增加Cache层次，增加了用户的选择2将Cache分解为指令Cache和数据Cache

Cache接入系统的体系结构：1侧接法：像入出设备似的连接到总线上，优点是结构简单，成本低，缺点是不利于降低总线占用率。CACHE 接入系统的体系结构 2. 隔断法：把原来的总线打断为两段，使 CACHE 处在两段之间，优点是有利于提高总线利用率，支持总线并发操作，缺点是结构复杂，成本较高。

**一致性保证策略（MESI）**要保证本地cache的数据，其它核cache的数据，内存的数据有一个一致的视图：1修改态（M）：处于这个状态的cache块中的数据已经被修改过，和主存对应的数据已不同，只能从cache中读到正确的数据2独占态（E）：处于本状态的cache块的数据和主存中对应的数据块内容相同，而且在其它cache中没有副本3共享态（S）：处于本状态的cache块的数据和主存中对应的数据块内容相同，而且可能在其它cache中有该块的副本4无效态（I）：处于本状态的cache块中尚未装入数据(状态机)

**虚拟内存**

独立的逻辑地址空间：通过页表将虚地址转换为实地址

实现内存共享：每个进程有独立的逻辑地址空间，建立逻辑地址和物理地址的转换机制

实现内存的保护：页表中存放有访问权限，通过硬件来保证权限（操作系统的“陷阱”操作）

虚拟存储器的目的

多级页表可以节省内存页表占用空间

虚拟索引虚拟标记VIVT：虚拟地址作为查找对象，不需要每次读取或者写入操作的时候把虚拟地址经过MMU转换为物理地址，提高性能；如果cache miss，则把虚拟地址发往MMU，经过MMU转换成物理地址/歧义(ambiguity)：歧义是指不同的数据在cache中具有相同的tag和index，相同的虚拟地址映射不同的物理地址就会出现歧义（例如多进程）解决：进程切换清除缓存/别名(alias)：不同的虚拟地址映射相同的物理地址，而这些虚拟地址的index不同，解决：nocache，仅映射到相同的物理地址

物理索引物理标记PIPT：tag和index都取自物理地址，不会出现歧义和别名，硬件设计上比VIVT复杂很多，硬件成本高，每次都要翻译成物理地址，性能下降，TLB可以缓解这个问题

虚拟索引物理标记VIPT：使用虚拟地址对应的index位查找cache，与此同时(硬件上同时进行)将虚拟地址发到MMU转换成物理地址，比较cacheline对应的tag和物理地址tag域判断是否是cache hit还是cache miss，VIPT以物理地址部分位作为tag，不会存在歧义问题，采用虚拟地址作为index，存在别名问题，不同的虚拟地址对应的物理地址是一样的，因为是物理标记，则缓存行中的标记是一样的，但是，虚拟地址不一样，因此index不一样，占用两个或者多个缓存行/VIPT的别名（一个物理地址映射为多个虚拟地址），不产生别名的条件：Index位数+cacheline大小位数<= Page大小位数

**RISCV虚拟地址**

虚拟地址：VPN1(10)+VPN2(10)+OFFSET(12)

物理地址：PPN1(12)+PPN2(10)+OFFSET(12)

页表项：PPN1(12)+PPN0(10)+RSW(2)+DAGUXWRV(8)

satp寄存器 S 模式控制状态寄存器控制了分页系统 MODE(1)+ASID(9)+PPN(22)

ASID：Address Space Identifier，地址空间标识符（可选），用以降低上下文切换开销

satp寄存器初始化：M模式的程序在第一次进入S模式之前会把0写入satp，以禁用分页

p然后S模式的程序在初始化页表以后会再次进行satp寄存器的写操作

虚拟地址到物理地址的转换

1satp.PPN 给出了一级页表的基址，VA[31:22]给出了一级页号，因此处理器会读取位于地

址(satp.PPN × 4096 + VA[31:22] × 4)的页表项。

2该 PTE 包含二级页表的基址，VA[21:12]给出了二级页号，因此处理器读取位于地址(PTE.PPN × 4096 + VA[21:12] × 4)的叶节点页表项。

3叶节点页表项的 PPN 字段和页内偏移（原始虚址的最低 12 个有效位）组成了最终结果：

物理地址就是(LeafPTE. PPN × 4096 + VA[11: 0])

TLB：如果操作系统修改了页表，TLB就会变得不可用，sfence.vma通知处理器，软件可能已经修改了页表，处理器可以刷新TLB，rs1指示哪个虚拟地址对应的转换被修改了，rs2指示被修改页表的地址空间标识符（一般相当于进程）ASID，如果两者都是x0，整个TLB会被刷新

**缺页处理流程（访问磁盘总是由缺页引起的）**

操作系统从主存中选择⼀⻚准备换出，为调⼊的⻚安排存放空间；

若被换出的⻚是“脏”⻚，需要将其写回磁盘存储；

操作系统申请I/O总线；

获得批准后，设备（磁盘）。 紧发送写命令给I/O跟着传送需要写回的⻚的全部数据。

I/O 加⼊到握⼿协议，并接受数据。

控制器发现发给⾃⼰的写命令 ，根据数据要写⼊的地址，读/写头移动到正确的柱⾯，同时，将数据接收到缓冲区。

寻道结束后，等待相应的扇区旋转到磁头下⾯，将数据写⼊扇区中。

在写⼊数据间隙，计算校验码并写⼊扇区中。

下一步，操作系统继续申请总线（如果还保持总线控制权，则不必申请）。

得到授权后，向磁盘发出读命令。

然后，磁盘识别地址，并转换为相应的地址段。

寻道，将读/写头移动到指定位置。

从指定扇区中读去数据，并进行校验。

磁盘申请I/O总线。

得到授权后，将数据通过总线送到内存。

**外存储器（DISK）**

易失性存储：1静态存储器：SRAM，Cache2动态存储器：DRAM ,特点：快速，掉电后信息丢失，访问粒度小（字节，缓存块）

非易失性存储器：1磁盘，磁带：磁表面存储器2光盘3SSD，固态存储器 ,特点：慢速，掉电后信息不丢失，访问粒度大（以数据块为访问单位）

**磁盘访问时间 = 寻道时间 + 旋转延迟 + 传输时间 + 磁盘控制器延迟**,传输时间=数据大小/传输速度，旋转延迟=半周的时间

**RAID(冗余磁盘阵列)**用N 个低价磁盘构成一个统一管理的阵列，以取

代特贵单一磁盘，RAID目标：N个磁盘的容量/1/N的访问时间/更高的性价比/采用冗余技术提高存储信息的可用性

**RAID0**:RAID0 将由RAID模拟的单个虚拟磁盘划分成带，每带*k*个扇区。第0带为第0到第*k –* 1扇区，第1带为第*k*扇区到第2*k* – 1扇区，等等。对*k=*1，每个带为1个扇区；对*k=*2，每带有2个扇区；等等。RAID 0以交叉循环的方式将数据写到连续的带中，下图描述的就是有4个磁盘驱动器的RAID盘。这种在多个驱动器上分布数据的方式叫作**分带**。如果软件发出从带的边界开始读四个连续带的数据块的命令，RAID控制器将把这个命令分解成四个单独的读命令，四个驱动器每个一个，让它们并行执行。这样，就实现了对软件透明的并行I/O操作。1. **适合数据请求量⽐较⼤的情况 2. 没有冗余，可靠性差，不算真正的RAID** X->X

**RAID1：**它复制了所有的磁盘，所以有四块主磁盘和四块辅助磁盘。每个对磁盘的写操作都进行两次，而每次读操作则可以读任意一个备份，把负载均衡分布到不同的驱动器上。这样，写操作的性能并不比单个磁盘好，但读磁盘的性能却比单个磁盘高了两倍，读写不同。容错性能就更好了，如果一个驱动器崩溃的话，只要简单的用备份驱动器代替就行了。恢复整个磁盘的操作包括两个步骤：装上一个新的驱动器，然后将整个备份驱动器的内容拷贝到新的驱动器上。**1. 冗余备份，可靠性⾼2. 写性能不⾼，但读性能却提⾼了两倍3. 成本较⾼**

**RAID2：**RAID2的工作单位为字，可能的话甚至可以是字节。首先我们可以想象将单个虚拟磁盘上的字节分解成一对4位的半字节，对每个半字节加上3位海明码形成7位字，即其中1、2、4位做校验位。然后，用下图所示的七个驱动器的磁头和旋转同步，就可能将整个海 明码字写在七个驱动器上，每个驱动器一位。**1. 驱动器必须同步旋转2. 驱动器个数要⾜够多3. 需要多个控制器**

**RAID3：**RAID3是RAID 2的一个简化版本，它只需对每个字计算一个校验位，写到一个校验驱动器上。和RAID2相同，驱动器之间必须严格同步，因为一个字被分布到多个驱动器中。

**1. 驱动器之间要严格同步2. 对整个磁盘崩溃的错误，能够进⾏恢复**

**RAID4：**RAID 4和RAID 0类似，将对带的校验写在额外的驱动器上。例如，若带的长度是*k*个字节，将所有的带异或到一起，产生一个*k*字节长的校验带。如果其中一块磁盘崩溃的话，它的内容可以从校验磁盘上重新计算出来。**1. 不对字进⾏校验，也不需要驱动器同步 2. 可以防⽌整块盘崩溃，但对盘上部分字节数据出错的纠错性能相当差3. 校验盘负载沉重**

**RAID5：**RAID 5为减少校验盘的负载，将校验位循环均匀分布到所有的驱动器上。**1.如果RAID 5的磁盘崩溃的话， 修复磁盘内容的将是⼀个复杂的过程**。RAID5 阵列能够对整个磁盘崩溃的错误进行恢复。**二位校验(a+b) X->(a+b/a)X**

**固态盘:**用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元（FLASH芯片、DRAM芯片）组成。按照每个存储单元能够存储的位数分为SLC，MLC，TLC，QLC,多比特单元/采用格雷码编码.固态盘存储单元的擦除次数,有限次擦除,随着擦除次数的增加，存储单元不能可靠地保持状态（存储数据）,SLC：100,000次,MLC：10,000次,TLC：1,000次SSD主要由SSD控制器，FLASH存储阵列，板上DRAM（可选），以及跟HOST接口（诸如SATA，SAS, PCIe等）组成。三个重要组成部分：主存芯片，闪存芯片，固件算法。

固态盘存储介质的组织：一个 package，即一个存储芯片，包含多个Die （典型1，2，4个），一个Die包含1个或者2个plane，可并行操作，每一个plane包含多个block，**block是最小的擦除单位**，每一个block里面有多个页，**页是最小的读写单位**，与磁盘不一样，**不会写入到原来的page，写入之前需要进行擦除操作，写入的时候不会在原来的page中写入，会在一个新的页面（可以在同一个块，也可以在不同的块，可能不同的plane，甚至不同的die上面）**这样，需要维护上层管理软件的逻辑地址和底层的物理地址之间的映射关系，

**FTL**做逻辑地址到物理地址的翻译，除了做**地址转换**，FTL还帮助完成磨损均衡，写入之前必须要进行擦除，但是每一个块的擦除的次数有限，写入的时候需要挑选位于擦除次数最少的块中的页面，完成磨损均衡

**垃圾回收步骤**:选择回收块 -> 移动有效页 -> 擦除块,垃圾回收开销 = 擦除开销 + 页面移动开销

**磨损均衡:**尽量让不同闪存单元的磨损程度接近，以提高设备整体寿命,动态磨损均衡：在回收和分配时选择，从GC选中回收的块中选择用的次数比较少的闪存块继续用,静态磨损均衡：在回收和分配时选择，会定期将静态数据占用的数据块搬移再利用

**I/O方式：程序直接控制**：CPU直接使用输入/输出指令来控制外部设备

CPU方：查询接⼝状态（循环等待） 直到接⼝已经接收到该字符读字符

外设⽅：往接⼝数据缓冲中送字符，处理完后，置状态寄存器，等待下⼀个字符

**程序中断**：外部设备请求，CPU响应，CPU与外设并行工作

CPU和外部设备同时工作：外部设备发起，CPU暂停正在执行的程序，进行响应，处理完成后，继续执行原来的程序，可以提高CPU的效率，可以同时管理多个外部设备

中断源：外中断：I/O设备等；异常(内中断)：处理器硬件故障、程序“出错”，Trap；中断触发器；中断状态寄存器

中断优先级：响应中断的顺序；禁止中断与中断屏蔽：中断允许触发器（EI、DI），有选择封锁

**中断过程**:中断请求：中断源设备设置中断触发器，每个中断源有1个中断触发器，同时可设置1个中断屏蔽触发器;中断响应：响应条件：允许中断、当前指令结束、优先级，响应实现：硬件实现的中断隐指令，保存断点;中断处理：保存现场信息，运行中断服务程序，中断返回

**全过程**：关中断，保存断点，判中断源转中断服务，开中断，执行中断服务程序，关中断，恢复断点，开中断，返回断点

中断设备接口组成：中断请求寄存器，中断屏蔽寄存器，优先级排队线路，数据缓冲寄存器 ，中断控制和工作状态逻辑，设备选择器，中断向量表

**直接存储访问（DMA）**：专用输入/输出控制器

**通道**：I/O通道是计算机系统中代替CPU管理控制外设的独立部件，是一种能执行有限I/O指令集合—通道命令的I/O处理机。一对多的连接关系，适应不同速度、不同种类的外部设备，可并行工作

通道功能：根据CPU要求选择某一指定外设与系统相连，向该外设发出操作命令，进行初始化，指出外设读/写信息的位置以及与外设交换信息的主存缓冲区地址，控制外设与主存之间的数据交换，指定数据传送结束时的操作内容，检查外设的状态

通道类型：字节多路通道：简单的共享通道，分时处理，面向低、中速字符设备；选择通道：选择一台外设置**独占整个通道（DMA设备可以独占使用内存总线）**，以成组传送方式传送数据块，效率高，适合快速设备；

**外围处理机：**通道型处理机：共享内存；外围处理机；通用计算机，独立完成输入/输出功能，通过通道方式与主机进行交互

**设计输入/输出系统：**性能：考虑吞吐量和延迟，适应各种不同类别的设备的性能的差异，从操作系统、驱动程序等各方面综合考虑，考虑到设备性能的提高/可扩展性：允许更多的设备接入到输入输出系统/可适应性：设备有无，设备故障

**输入/输出系统**：I/O设备多，功能复杂速度不一，多种控制方式，解决速度不一的问题，尽量少地占用CPU资源，操作系统管理，硬件直接支持，与不同的设备有直接的依赖关系（驱动程序），尽量使设备使用统一的标准——虚拟设备

**总线:**共享的信息通道，用于连接计算机多个子系统部件,好处：解决外部设备“杂”的问题，容易增添新的设备，使用相同总线标准的外设容易在不同计算机间兼容/降低成本：总线可供多个设备共享/简化设计/不足：容易成为信息通道的瓶颈，总线带宽限制了整条总线的吞吐量/总线的最高速度主要由下列因素决定：总线长度，总线负载的设备数，负载设备的特性（延迟是否差异较大？数据传输率差异较大？）

**总线分类:**单总线计算机（主板总线）：使用一条总线，处理器/主存储器/IO设备之间通信/优点：简单，成本低/缺点：速度慢/双总线系统：使用2条总线，处理器-主存储器总线与IO总线,IO总线通过适配器和处理器-主存总线相连/三总线系统：使用3条总线，处理器-主存储器总线/主板总线IO总线，主板总线连接到处理器-主存总线，I/O总线连接到主板总线/优点：大大减少处理器—主存总线负载，现代PC

处理器-主存总线 （专用）： 传输距离短、速度高，主存储器专用，保证主存储器-处理器之间的高带宽，直接和处理器连接，优化处理使之适应Cache块传送/输入/输出总线（行业标准）：通常距离较长，速度较慢，需要适应多种输入/输出设备，和处理器-主存总线通过桥连接（或通过主板总线）/主板总线（行业标准或专门设计）：主板：连接各部件器件的底盘，应允许处理器、主存储器和输入/输出设备互连，应有价格优势：所有组件连接在一条总线上

**总线组成:**控制线: 总线请求信号及数据接收信号，指明数据线上传输信息的类型/数据线：在源设备和目标设备间传送信息，数据和地址，复杂的命令/**地址线**：指定主存单元和 IO 设备接口电路的地址

总线通信协议：定义总线传输中的事件顺序和时序要求/异步总线传输： 控制信号（请求，应答）作为总控信号/同步总线传输：使用共同的时钟信号/总线事务包括两个部分：发起命令 （和地址），传输数据

**总线仲裁:**总线仲裁的基本要求：某总线主设备使用总线前应发出总线请求/只有得到授权后，主设备才能使用总线/使用完毕后，主设备应通知仲裁器;总线仲裁器在以下两方面取得平衡：**优先权**：优先级高的设备应该得到优先服务/**公平性**：最低优先级的设备也不能永远被排除在总线服务之外

**集中仲裁**：**菊链仲裁**：所有设备共用一个总线请求信号，缺点：无法保证公平性，低优先级设备可能得不到总线使用权，总线授权信号的逐级传递限制总线的速度;**集中平行仲裁**：通过集中的仲裁器进行，用于几乎所有处理器-主存总线和一些高速输入/输出总线

**分布仲裁**： 路口没有交通警察，所有车辆先停下，确认其他方向没有来车后通行，通过自我选择进行分布式仲裁：每个使用总线的设备将自己的标识放在总线上/碰撞检测：以太网

**优先级仲裁**：例如，救护车在道路上有高优先级

**同步总线**：控制线中包含有一根时钟信号线，传输协议根据时钟信号制定，优点：逻辑简单、高速，缺点：总线上所有设备必须按时钟频率工作，为防止时钟信号扭曲，高速工作时，总线距离必须足够短

**异步总线**：不使用统一的时钟，可适应设备的不同速度，不用担心时钟信号扭曲，距离可较长，使用握手协议

**典型的异步协议：**

1. 主存储器收到外部设备发出的ReadReq信号，从数据总线读到地址，

并发出Ack信号。

2. 外部设备发现Ack信号为高 =>释放ReadReq和数据

3. 主存发现ReadReq信号为低，将 Ack信号置低

4. 主存读出数据后，将数据送总线，并将DataRdy置高

5. 外部设备发现DataRdy为高，读数据，并发出Ack信号

6. 主存发现Ack为高，将DataRdy拉低，并释放数据线

7. 外部设备发现DataRdy为低，拉低 Ack信号，指示传送结束

**///增加总线的宽度**：可增加每个周期传送数据的量，提高了成本

**///分别设置数据总线和地址总线**：可同时传送数据和地址，提高了成本

**///采用成组传送方式**：一个总线事务传送多个数据，每次只需要在开始的时候传送一个地址，直到数据传送完毕才释放总线，代价：复杂度提高，延长后续总线请求的等待时间///

多主设备总线提高事务数量：仲裁重叠：在当前事务时，为下一总线事务进行仲裁/总线占用：在没有其他主设备请求总线的情况下，某主设备一直占用总线，完成多个总线事务/地址、数据传送重叠，在现代内存总线上应用了上述全部技术

**PCI总线:**外部组件互连总线/时钟频率：33MHz或66MHz（CLK） /集中仲裁方式（REQ#、GNT#）和上一事务重/32位地址和数据线互用（AD）V2.1 为64位/总线协议:总线周期：内存读、内存写、内存成组读等（C/BE#）/地址握手和保持（FRAME#、IRDY#）/数据宽度（C/BE# ）/通过IRDY#和TRDY#握手信号传输变长的数据块/最大带宽达133MB（33MHz）或528MB（66MHz)/所有信号在时钟正边沿采样/集中平行仲裁和上一事务重叠/所有事务可无限制成组传送/地址段起始于FRAME#信号有效/第一时钟周期主设备发出cmd和address/数据传送:当主设备准备好传输数据，主设备发出IRDY# 信号/从设备准备好传输数据，发出TRDY#信号/上述两个信号均有效时的时钟上升沿开始传送数据

**PCI优化**

尽量使总线有效传输:可采用类似RISC技术，仲裁和数据传输并行进行

总线占用:为上一主设备保留总线授权，直到有其他主设备申请使用总线/得到授权的主设备可在不仲裁的情况下直接开始下一传送过程

仲裁时长:主设备和从设备尽力延长传输流（使用xRDY）/从设备使用STOP (abort or retry)信号终止连接/主设备通过FRAME信号终止连接/仲裁器通过GNT信号终止连接

延迟(挂起, 时段分离)事务:对慢速设备，在请求后暂时释放总线

*提⾼性能/降低成本:*总线宽度：将地址和数据线分开/互⽤地址和数据线,数据宽度：越宽越快（32位）/越窄越廉价（8位),⼤⼩：多字可减少总线开销 送单字传送简单,主设备：多主设备（仲裁）/单主设备,时钟：同步/异步,协议：并⾏/串⾏

**接口电路和外部设备**

接口基本功能：1提供主机识别使用的I/O设备的支持，为每个设备规定几个地址码或编号2建立主机和设备之间的控制与通信机制，接收处理器（主设备）的命令，并提交给外部设备，同时为主设备提供外部设备的状态3提供主机和设备之间信息交换过程中的数据缓冲机构4提供主机和设备之间信息交换过程中的其他特别需求支持，屏蔽外部设备的差异5. 数据输入输出缓冲6使用同步帧进行实时设备同步（注意：仲裁不是接口的工作）

**通用可编程接口电路**：1通用，能有多种用法与入/出功能2可编程，能通过指令指定接口的功能和运行控制参数3接口内部组成，设备识别电路，数据缓冲寄存器（输入/输出），控制寄存器，状态寄存器，中断电路，其他电路

串行通信：1同步传送，采用同步信号，内同步：同步字符，外同步：硬件同步信号2异步传送，起始位、停止位，波特率3全双工，通信双方有各自的接收和发送部件，两条数据线

**RISC-V 不通过专用 I/O 访问指令对外设进行访问，而是将外设映射到地址空间上，使用 load/store 指令访问外设。**

