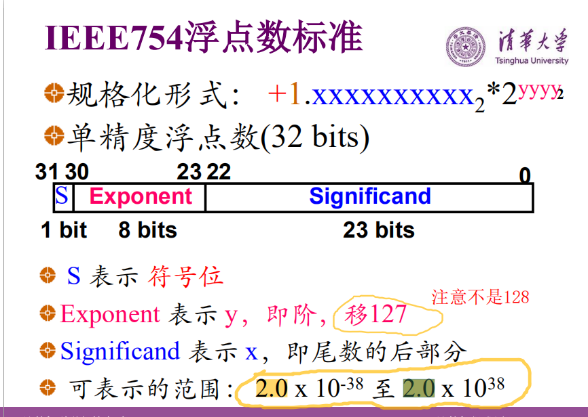
笔记内容：面向912考试，基于912真题和THU期末题以及之前做的重点笔记综合而成的更加凝练的笔记。

适合于最后冲刺阶段经常翻看。

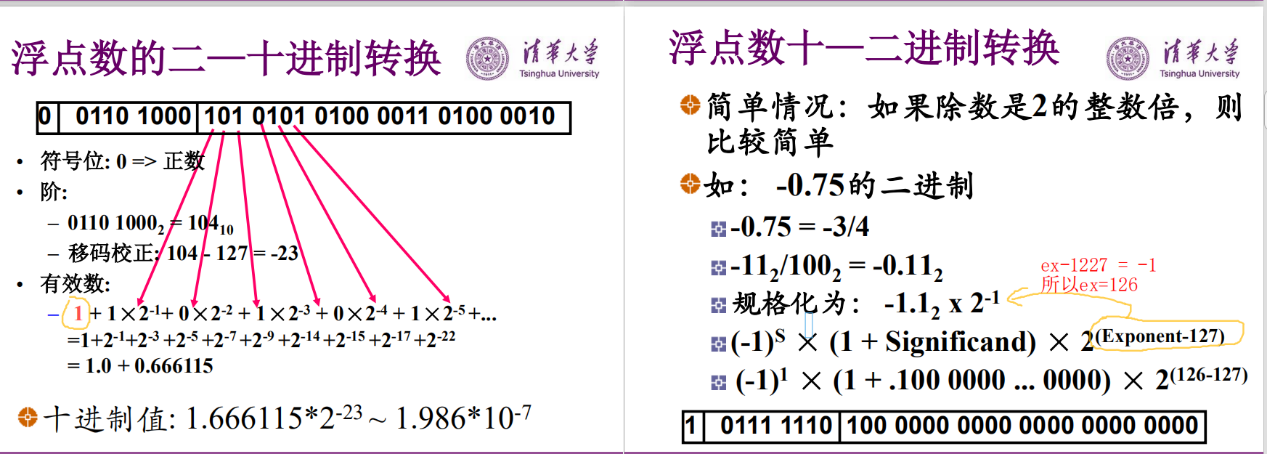
**运算器**

IEEE规格化单精度浮点数能表示的正数范围： ~

最小正数为 ，最大正数为



**二进制的浮点数转十进制的浮点数**，**十进制的浮点数转二进制的浮点数**



【例】十进制单精度浮点数-27.625在IEEE754浮点标准下表示：\_\_\_\_\_\_\_（16进制）very 重要！！！！！

27 = 11011

0.625 = 0.101

-27.625 = -11011.101

规格化：-1.1011101 × 2^4

(-1)^1 × （1+.1011101）× 2^(131-127)

s = 1 ，exponent = 131 = 10000011 ，significand = 10111010

IEEE754浮点标准下表示：1 10000011 10111010000000000000000 = 0x**C1DD0000**

**浮点数float y,若y\*y仍然在浮点数表示范围内，则y大于0 。** ❌说法错误

容易举出反例：令y=-1.0，y\*y=1.0在浮点数的表示范围内，此时y<0。

浮点数的表示范围很大的，几乎相当于负无穷到正无穷了，

**整数Int x<0,则必有-x>0。**❌说法错误

int 整数的取值范围为 [ -2^31 ,  2^31 - 1 ]

最小负数即为 -2^31 ，其只有补码形式，即为0x8000 0000

所以说对于 int a=0x80000000，-a依然是0x80000000。即 a<0，-a<0

但是如果题目变成：Int x >0,则必有 -x < 0。则就是正确的。

**假设x类型是C语言中的int，若x>0，则x\*x>0 。**❌说法错误

当x\*x 超过int上限表示的范围，即溢出了，那么就会得到不可预期的结果。

**C 语言若 int x,y 若 x>y，则-x<-y。**❌说法错误

反例： -2^31 + 1  >  -2^31，但是   -（-2^31） =  -2^31 不变，仍然等于其本身

-( -2^31 + 1 ) 是一个正数，其显然大于一个负数，所以存在 x>y，则-x> -y

冯诺依曼结构体系中把程序也当做数据放在内存中。

**求补码**

* 正数的补码就是其原码
* 负数的补码等于抛开符号位逐位按位取反，最后＋1
* 大端存储：数据低位保存内存高地址字节中，数据高位保存在内存低地址字节中。
* 小端存储：数据高位保存内存高地址字节中，数据低位保存在内存低地址字节中。

**tips**：按小端存储，就是把算出来的数据两位一组前后倒过来，大端存储不变

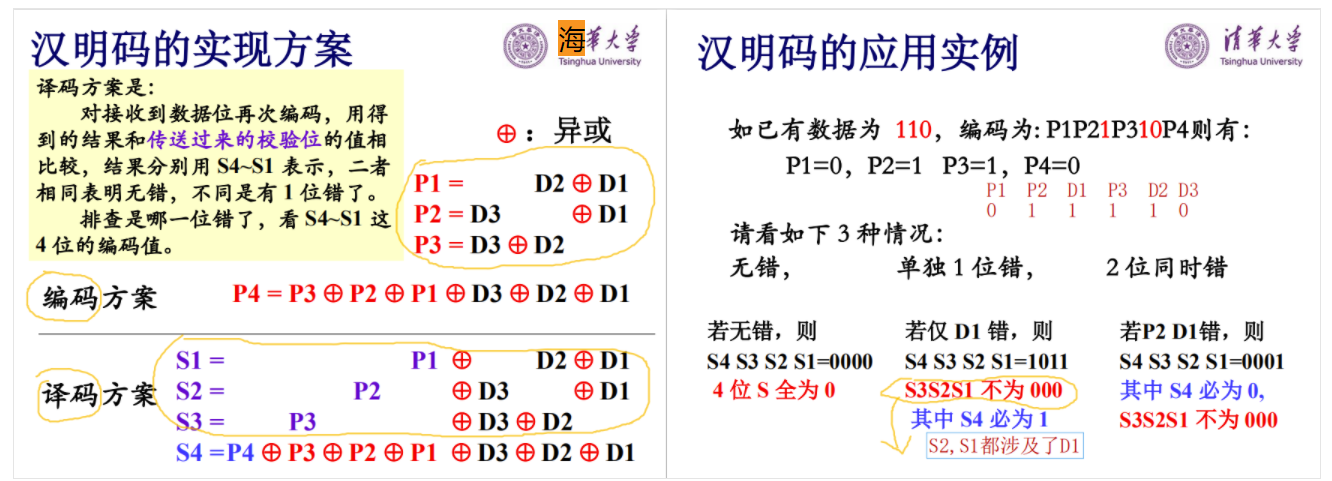
**汉明校验码**

能发现并改正 k+r 位中任何一位出错；

能发现 k+r 位中任何二位同时出错，但是不能纠正两位错误。

k 与 r 之间应该满足什么样的关系 ：**2^(r-1) ≥ k+r**

用 r-1 位校验码为出错位编码，再单独设一位用于区分 1 位还是 2 位同时出错，更实用。



【例】若海明码 **P1P2** D1 **P3** D2 D3 **P4** 为**01** 0 **1** 1 0 **0**，则该海明码有（）位错误（0 位，一位，二位），正确的 D1D2D3 为

**P1 P2** D1 **P3** D2 D3 **P4**

**0 1** 0 **1** 1 0 **0**

**--------------------------------------------------**

S1 = p1 ⊕ D2 ⊕ D1 = 1

S2 = p2 ⊕ D3 ⊕ D1 = 1

S3 = p3 ⊕ D3 ⊕ D2 = 0

S4 = p4 ⊕ p3 ⊕ p2 ⊕ p1 ⊕ D3 ⊕ D2 ⊕ D1 = 1

S4 S3 S2 S1 = 1011

S4 = 1，故有一位错，出错的位位S1和S2的交集即D1

将收到的海明码中的D1反转过来就可以了，原D1D2D2 = 010 -> 110

正确的D1D2D3应为 110

**指令、处理器**

计算机运行的最小单位是指令。（不是什么微指令）

常见的指令寻址方式：立即数寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、变址寻址、基址寻址、堆栈寻址、相对寻址。

程序执行的时间由指令数量、时钟周期长度和每条指令的周期数目（CPI）等因素共同决定。单单提高CPI，cpu 主频等，无法判断是否能提高程序执行的快慢。

指令由操作码和操作数地址构成。

处理机组合逻辑电路进行算术运算，时序逻辑电路可以用于数据暂存，组合逻辑电路可以用于分支选择。

指令流水线可能发生的冲突有**三类**： 结构冲突、数据冲突、控制冲突。

**各类冲突原因**：

**结构冲突**：指令在重叠执行过程中，需要用到相同的资源，但是硬件资源满足不了重叠执行的要求，从而产生所谓的结构冲突。

**数据冲突**：指令在重叠执行过程中，一条指令依赖于前面指令执行结果数据，但是又在自己该需要的时候得不到时发生的冲突。

**控制冲突**：流水线的分支指令或者其他需要改写PC的指令造成的冲突。

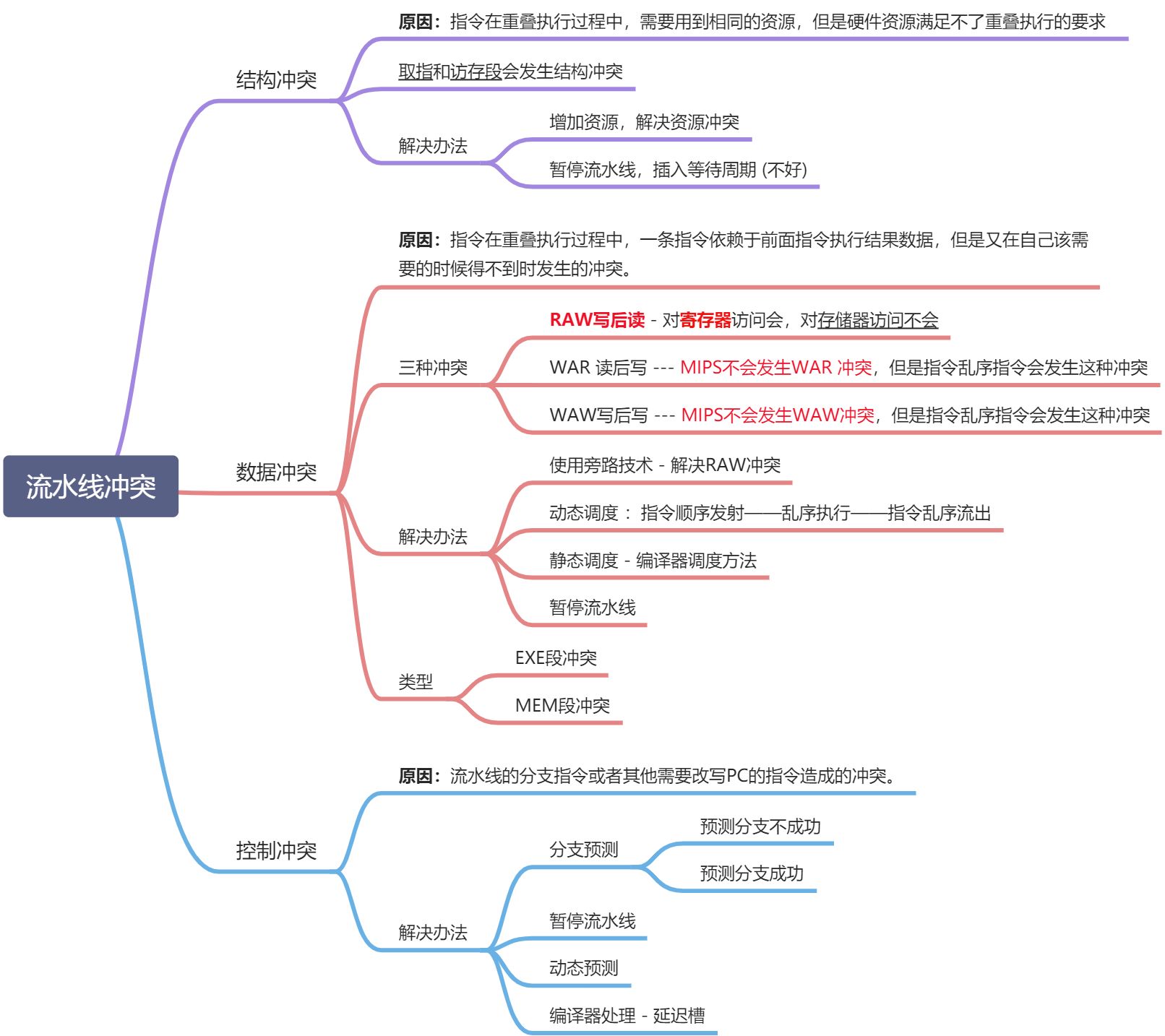
解决**结构冲突**的2种方法：增加资源、暂停流水线，插入等待周期、

解决**数据冲突**的4种方法：使用旁路技术、暂停流水线、静态调度 — 编译器调度方法、动态调度 — 指令顺序发射，乱序执行，指令乱序流出。

解决**控制冲突**的4种方法：分支预测、暂停流水线、动态预测、编译器处理。

**MIPS**五级流水线只会发生RAW（读后写）冲突。

MIPS五级流水线，全部指令都是选用5个步骤完成，不同指令的执行时间相同



**单周期、多周期、流水线对比**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **单周期** | **多周期** | **流水线** |
| **特点** | 1.每条指令占用一个时钟周期  2.逻辑设计简单，时序设计也简单 | 1.指令的执行划分为多个步骤  2.每个步骤占用一个CPU周期  3.不同指令的指令周期不同  4.指令串行执行  5.提高了整体性能 | 1.全部指令都是选用5个步骤完成，执行时间相同  2.在不过多增加硬件投入的情况下，提高系统效率  3.单任务的速度不提高，但提高系统的吞吐率  4.尖峰CPI = 1 |
| **缺点** | 1.各组成部件的利用率不高  2.各部件大部分时间在等待  3.时钟周期应满足执行时间最长指令的要求，也即Load指令 | 1.各部件利用率依然偏低  2.CPI > 1 | 会有冲突问题   * 结构冲突 * 数据冲突 * 控制冲突 |
| **CPI** | CPI =1 | CPI > 1 | 尖峰CPI = 1 |

**流水线计算大题**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **取指 IF** | | **译码 ID** | | **执行EXE** | | **访存 MEM** | | **写回 WB** |
|  |  | 内存延迟 | 锁存器输入输出 | 寄存器输出延迟(读寄存器) | 锁存器输入输出 | ALU延迟 | 锁存器输入输出 | 内存延迟 | 锁存器输入输出 | 寄存器输入延迟(写寄存器) |
| **单周期** | **R型指令** | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ❌ | | ✔ |
| **Lw指令** | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ✔ |
| **Sw指令** | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ❌ |
| **J型指令** | ✔ | ❌ | ❌ | | ❌ | | ❌ | | ❌ |
| **Branch** | ✔ | ❌ | ✔ | ❌ | ✔ | ❌ | ❌ | | ❌ |

**对于单周期，**如果题目给了PC的输入延迟，还需在计算每一个指令的延迟上加上PC的输入延迟。

**单周期**的最短时钟周期(指令延迟、时钟周期)为所给的指令中执行时间最长的指令，也即LW指令。

**单周期指令延迟时间(LW) =** PC输入延迟 + IF段内存延迟 + ID段寄存器输出延迟 + ALU延迟 + MEM访存延迟 + WB段寄存器输入延迟

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **取指 IF**(写回PC并行) | | **译码 ID** | | **执行EXE** | | **访存 MEM** | | **写回 WB** |
|  |  | 内存延迟 | 锁存器输入输出 | 寄存器输出延迟(读寄存器) | 锁存器输入输出 | ALU延迟 | 锁存器输入输出 | 内存延迟 | 锁存器输入输出 | 寄存器输入延迟(写寄存器) |
| **多周期、流水线** | **R型指令** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ❌ | | ✔ |
| **Lw指令** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ |
| **Sw指令** | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ❌ | ❌ |
| **J指令** | ✔ | ✔ | 写回PC | | ❌ | | ❌ | | ❌ |
| **Branch** | ✔ | ✔ | ⭐ | ✔ | ✔ | ✔ | ❌ | | ❌ |
|  | | ⭐处为【寄存器输出延迟】 与【ALU计算pc+imm并行】 | | | | | | | | |

MIPS中，**多周期**和**流水线**的都分为5段执行，并且每一段相对于单周期来说还多了一个锁存器作为每一段的缓存。

**多周期**和**流水线**的时钟周期都是5段中执行时间最长的一段，通常为【访存MEM】段 = 内存延迟时间 + 锁存器输入输出延迟

**多周期的指令延迟** = 每种类型的指令执行的段数 ✖ 时钟周期 （如R型段数为4，Lw为5段，J为2段）

* 最短指令延迟 ：J 指令延迟 = 2 ✖ 时钟周期
* 最长指令延迟 ： Lw指令延迟 = 5 ✖ 时钟周期

**流水线的指令延迟** = 5 ✖ 时钟周期

**存储器**

**静态动态存储器**

SRAM和DRAM都是电易失性存储器，但是FLASH是非易失性存储器。

静态存储器使用触发器，不需要定期刷新。

动态存储器使用电容，需要定期刷新。

**高速缓冲存储器Cache**

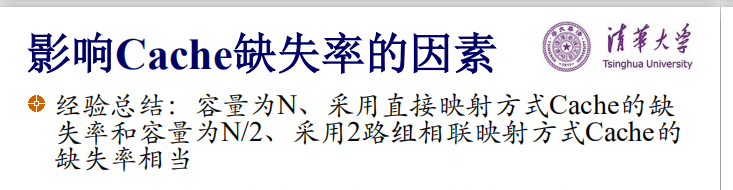
缓存原理利用了程序的局部性 。

程序局部性原理包括空间局部性和时间局部性。

缓存缺失的四类原因：必然缺失、容量缺失、冲突缺失、无效缺失。

高速缓存器的几种映射方式：全相连映射、直接映射、组相连映射

命中率从高到底分别是 ：全相连=n路组相联 > 4 路组相连 > 2 路组相连 > 直接映射



**I/O**

**I/O 通道的类型**分别是字节多路通道、选择通道、数组多路通道。

**DMA使用总线的方式**为独占总线方式和周期窃取方式。

**总线**是用于连接计算机多个子系统的共享的信息通道。

**总线仲裁**是多个设备需要使用 总线时，如何安排总线。

**总线仲裁的两种方式**为集中仲裁和分布仲裁。其中集中仲裁又包括菊链仲裁、集中平行仲裁。

**菊链仲裁方式**是链式询问，从离总线仲裁器最近的设备开始询问，也即越靠近总线仲裁器的设备优先级越高。

**中断处理过程**

包括关中断、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、返回断点等步骤。

