**第一章**

**1、计算机设计目标：性能 性价比 性能功耗比**

性能Performance per second

大型机时代

性能价格比Performance per dollar

PC机时代

性能功耗比Performance per watt

移动计算PMD时代

**2、计算分类：**SISD单指令流单数据流 SIMD向量机、多媒体扩展处理、GPU

MISD紧耦合MIMD: SMP、DSM松散耦合MIMD: 机群cluster

MIMD 无商业产品

**3晶体管动态功耗 含义、因素**

晶体管状态转换(1->0或者0->1)所需要的能耗

动态能耗∝容性负载×电压^2

动态功耗∝1/2×容性负载×电压^2×开关频率

1. **系统可靠性 MTBF MTTF MTTR 计算**

系统/模块的可信度由可靠性和可用性来衡量

模块可靠性：从某个参考时刻开始的连续服务时间

MTTF ( Mean time to failure)：平均无故障时间

MTTF的倒数是故障率，经常用每十亿小时故障次数来衡量

MTTR ( Mean time to repair)：故障平均修复时间，服务中断度量

MTBF(Mean time between failures) 平均故障间隔时间= MTTF + MTTR

模块可用性：当模块在服务与服务中断两种状态中切换，服务占据的比率

模块可用性=MTTF/MTTF+MTTR

**5、计算机设计遵循基本原则**

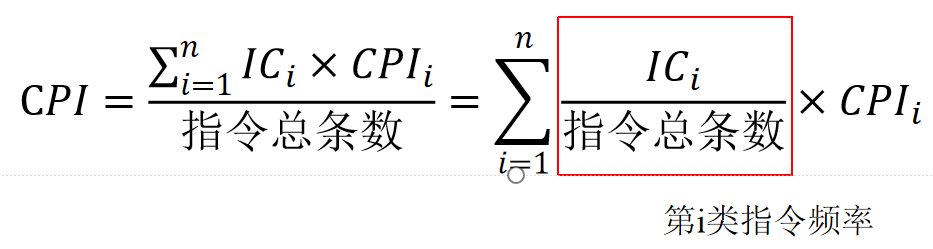
局部性原理、利用并行性、聚焦一般情况

**6、等效CPI 、Amdahl定律 计算**

新执行时间=原执行时间×{(1−升级比例)+升级比例/升级加速比}

CPI=程序的CPU时钟周期数 / 指令条数

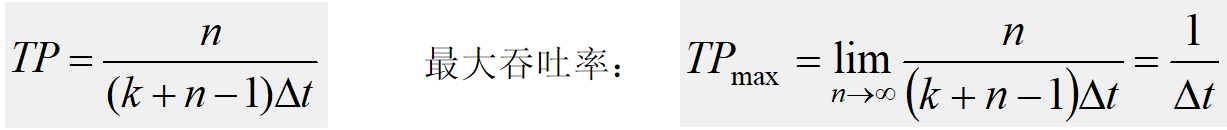
CPU时间=指令条数×CPI × 时钟周期时间

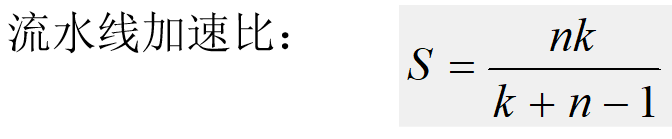


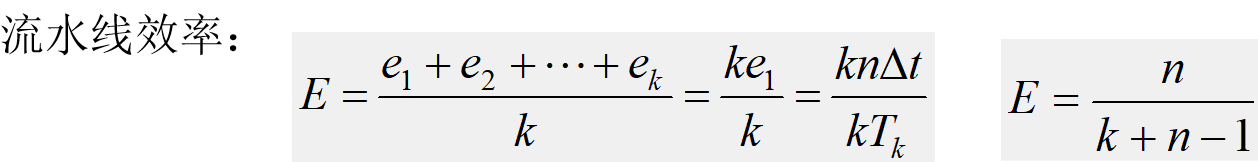
**第二章**

**1、RISC主流指令集：ARM、RISC-V、MIPS LoongArch**

**2、流水线性能指标：吞吐率 加速比 效率**







1. **分支延迟转移**

在条件指令后面插入一条不相关的指令

**4、分支预测：动态、静态、转移方向**

分支预测：预测分支指令/条件指令的方向——转移成功或转移不成功

动态分支预测：根据程序行为动态预测转移方向，转移方向动态变化

静态分支预测：预测方向固定，根据编译阶段的信息来预测方向，根据早期运行情况来预测转移方向——单条分支指令往往会倾向于转移成功或转移不成功，并非以等概率转移成功或转移不成功。

1. **多周期5段RISC流水线分析**

**第三章**

**1、名称相关**：两条指令使用相同的寄存器名称和相同的内存地址，却没有信息流动交换。

不是真实的数据相关

**2、反相关(Anti-dependence)**: 指令j要写某个寄存器或者内存单元，而指令i要读相同位置或相同寄存器，并且保持指令顺序(指令i在指令j之前)

**3、输出相关(Output dependence)**：两条指令i和j写同一个寄存器或者同一个内存单元，需要保持指令顺序

结构冒险、控制相关

数据冒险：WAW WAR RAW

相关预测：

gshare预测器

**竞赛预测器：全局 局部 选择器**

竞赛预测器使用多个预测器，一个基于全局信息(多条分支指令转移历史)，另外一个基于局部信息(某条分支指令最近若干次执行时转移情况)。再用一个选择器针对特定分支指令选择合适的预测器。

**4、Tomasulo 算法指令动态调度分析 例子**

**5、带ROB的Tomasulo动态调度分析 例子**

寄存器重命名WAW WAR

**6、处理器微结构：**

顺序发射、顺序提交、乱序执行

**7、循环展开局限性**

随循环展开次数增加，平均每个迭代步的循环开销(计算循环变量、分支语句)减少量，变得越来越少

循环展开会让增加代码的长度，增加指令的条数

----对于比较长的循环语句，这会导致Cache的不命中率增加

寄存器压力：循环展开会增加寄存器的需求数量

----如果不能分配足够的寄存器，会损失一些或全部的循环展开好处

**8、多发射过程**

现代微体系结构

动态调度+多发射+猜测+分支预测

两种方法来实现多发射

分配保留栈，半个时钟周期更新流水线控制表，一个时钟发射2条指令

设计逻辑电路处理指令间存在的相关性

可以将两者结合

**第四章**

**1、存储器层次结构->存储墙**

**2、Cache高级优化方法：**

减少命中时间

小容量L1 Cache； 组内块(路)预测

增加Cache带宽

流水线结构Cache；多缓存组Cache；无阻塞Cache

减少不命中时间开销

关键字优先；合并写缓冲器

减少不命中率

编译器优化

利用并行性减少不命中时间开销和不命中率

硬件预取；编译器预取

提高Cache带宽：分组结构、无阻塞、流水线结构

Cache映射: 直接相联 组相联 全相联

**3、Cache替换方法：FIFO LRU 随机**

LRU(近期最久没用)算法：基本思想是近期被用到的块大概率还会被再次使用，所以选择近期最久没用的块当做被替换的块

FIFO先进先出算法：最先进来的块被替换

**4、Cache更新方法比较：写回法、写直达法**

Cache数据更新策略：

写直达法write-through：在更新Cache数据时，同步更新内存数据

写回法write-back：只有当Cache块被替换时，才更新内存的相应数据

两种策略比较：

1、如果数据改变多次，写直达法多次会写内存，写回法则不会。

2、写直达法与内存通信量大，写回法与内存通信量小，比较适合嵌入式应用

3、写直达法能较好保持数据一致性，更加适合多级Cache。只需要保持与相邻级Cache数据一致，不用一直追溯到内存

4、写直达法实现比较容易

**第五章**

**1、挖掘数据级并行性三种结构：**

**GPU**

GPU主要用来加快处理图形、游戏、视频，现在可以将其扩展到更广的应用(AI 计算、深度学习训练)。

GPU的主要思想：

混合执行模式

CPU是主机，GPU是设备

开发了一种类似C的编程语言，集成开发环境CUDA(Compute Unified Device Architecture)

将各种形式的GPU并行统一为CUDA线程(最低层次的并行性，CUDA编程的基元，一个CUDA线程处理一个元素)

编程模式：SIMT(Single Instruction Multiple Thread)

**向量机**

将数据元素集合读入向量寄存器

操作基于向量寄存器

将结果分散写回存储器

***向量寄存器***

用于隐藏存储器延迟

以杠杆作用方式扩大存储器带宽

**SIMD多媒体扩展**

**2、SIMD多媒体扩展：**SSE MMX AVX

**3、GPU的调度器：**线程调度器、线程块调度器

**4、SIMD多媒体扩展与向量机不同**

与向量指令相比较，SIMD多媒体扩展：

将操作数数量固定，在X86结构的多媒体扩展集MMX，SSE和AVX中增加了几百条指令。

没有复杂的寻址方式(比如步幅访问、集中分散访问)

没有掩模寄存器用于条件执行向量元素。

***5、*屋脊线Roofline性能模型**

基本思想

画出峰值浮点运算吞吐量随算术密度变化的曲线

将浮点性能、存储器性能、算术强度关系反映到2D图形中

这是一种比较各种SIMD体系结构性能的直观可视化方法

***算术密度Arithmetic intensity***

每访问存储器一个字节，包含的算术运算量

向量机：集中-分散操作

**第六章**

**1、消息传递系统、共享存储器系统**

**2、LLC**

共享最后一级Cache (Last Level Cache, LLC)

**3、Cache一致性**

Cache一致性：在共享存储器系统中，维持数据在存储器和多个处理器或多个核的私有Cache中的数据副本一致

**4、监听协议和目录协议、适用场景、优劣**

***监听协议Snoopy protocol***

当某个处理器写一个共享数据时，将写无效信号通过信道广播给所有的处理器。每个处理器监听信道，看无效信号是否与自己的Cache副本有关，如果是则将副本作废。

常用于SMP系统，需要广播，适合于共享总线结构。

总线是独占共享资源，延迟会随处理器数量增多而增加，限制了连接的处理器数量

***目录协议Directory protocol***

为每个存储行维持一目录项, 记录所有当前持有此行数据备份的处理器号以及是否已被改写等信息。

当一个处理器核写数引起数据不一致时, 它就根据目录的内容只向持该数据备份的处理器发出写无效/信号, 避免了广播。

扩展性比较好，可以连接较多数量的处理器，多用于DSM系统

**5、互连网络**

多核处理器通过互连网络将处理器核、Cache、内存控制器、IO 接口等模块连接起来。

**6、硬件多线程**

允许多个线程以重叠执行方式共享单个处理器的功能部件，以提高硬件资源的利用效率。

**7、粗粒度多线程**

只有遇到长停顿，比如最后一级Cache不命中，才切换进程

减少进程切换频率，加快单个进程执行，简化硬件

不能隐藏短停顿，比如数据相关

流水线清空装入开销大

**8、细粒度多线程**

每个时钟周期切换一次进程。多线程指令交替执行

如果某个进程出现停顿，更换另外一个进程

可以隐藏流水线执行中长短停顿，但是会推迟单个进程执行

**9、SMT同时多线程**

在多发射、动态调度处理器中，挖掘指令并行性的基础上，挖掘线程并行性

在多发射、动态调度处理器中，受指令间相关性的限制，单个线程不能充分利用处理器功能部件并行性

从多个线程调度指令，一次多发射包含来自不同线程的指令，当功能部件可以用，执行来自不同线程的指令。从而，让多线程指令并行执行

线程内数据相关性由寄存器重名来解决