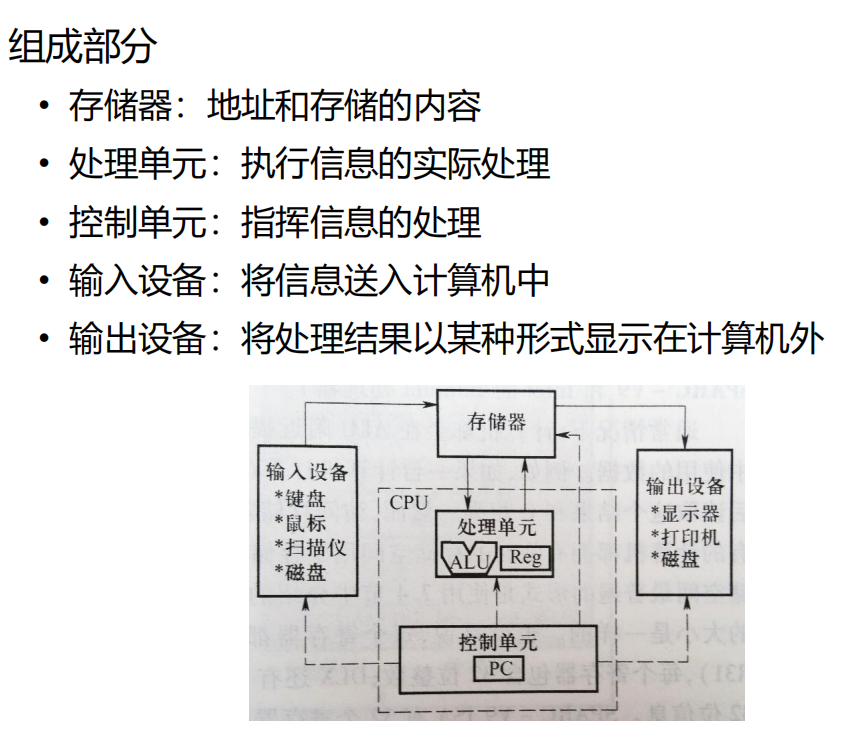
Lecture01 Introduction

1.结构（Architecture）可见，如计算机有无乘法指令

组织（Organization）不可见，如如何实现该乘法指令

2.冯诺依曼结构

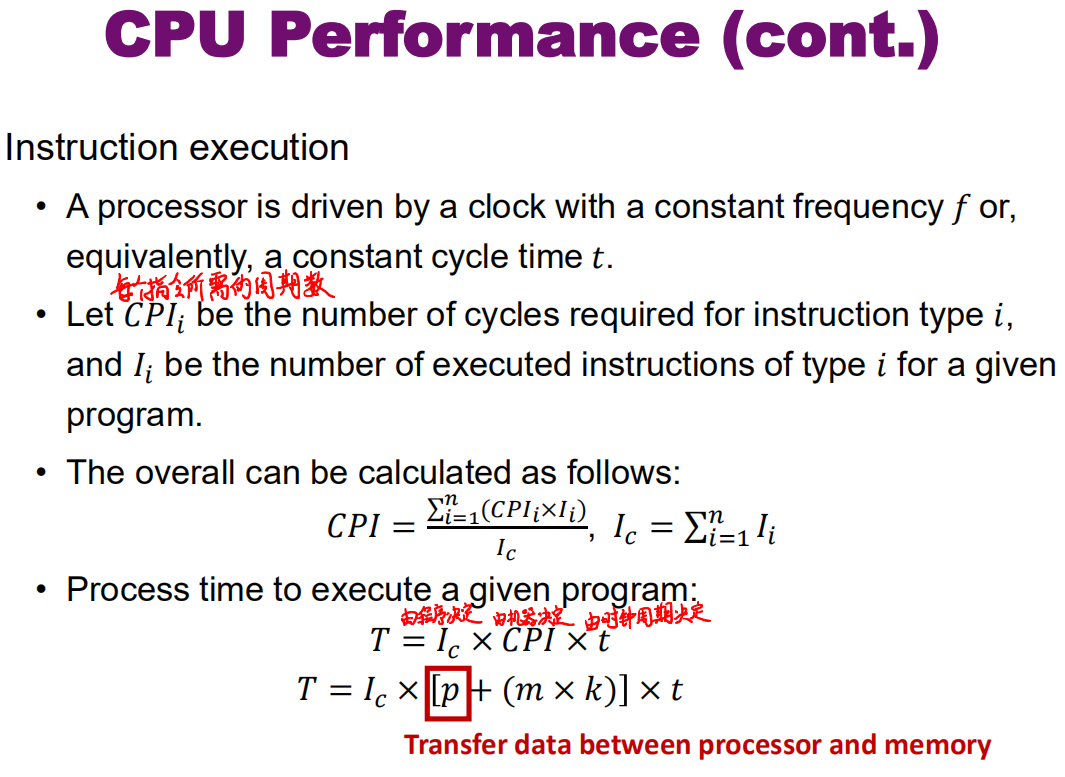


3.摩尔定律：可以放在一个芯片上的晶体管的数目每18个月翻一番

影响：计算机成本下降；便携性；运行速度提高；省电、降温；连接可靠

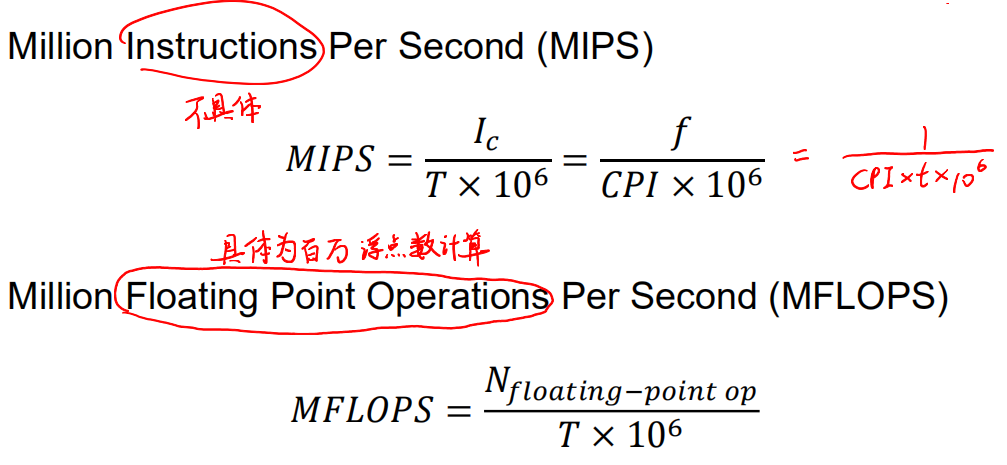
4.计算机性能提高主要目标/驱动是CPU速度的提高

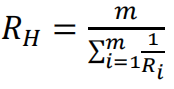
5.CPU性能评估



CPI=p+(m×k) p:译码和执行指令所需周期数；m:内存访问次数；k:访问内存所需周期数

指令数×平均每条指令所需周期×周期时间



对评估结果取平均值：算术平均值；调和平均值

Lecture02 A Top-Level View of Computer

1.内存问题：

（1）数据传输速率跟不上CPU速度要求

解决：增加缓存或其他缓冲方案，以降低内存访问频率，

提高数据传输速率；

增加一次性检索到的比特位数；

使用内存层次结构而不是依赖单个内存组件

（2）约束：

容量:越大越好

速度:跟上处理器

成本:相对于其他部件来说是合理的

约束之间的关系：访问时间越短，每比特的成本越高

2.I/O问题：I/O性能跟不上CPU速度的提高，速度差距大

解决：缓冲区（Buffer）I/O模块；新的接口技术。

3.（1）CPU问题：闲置

解决：中断（Interrupt）

中断处理顺序：按序/按优先级

（2）CPU特性：

从一条指令执行到下一条指令按顺序执行(除非显式修改)；

数据和指令存储在一个读写存储器中；

这个内存的内容可以按位置寻址，而不考虑包含在那里的数据类型。

4.bus：总线是连接两个或多个设备的通信通道

三种：数据线、地址线、控制线

数据线:在系统模块之间移动数据

地址线:在数据总线和地址I/O端口上指定数据的源或目的地

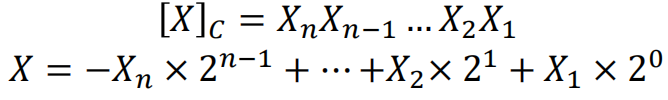
控制线:控制数据和地址线的访问和使用

共享的传输介质简化了互连布局和处理器的控制

Lecture03 Integer, Floating-point and Decimal Representation

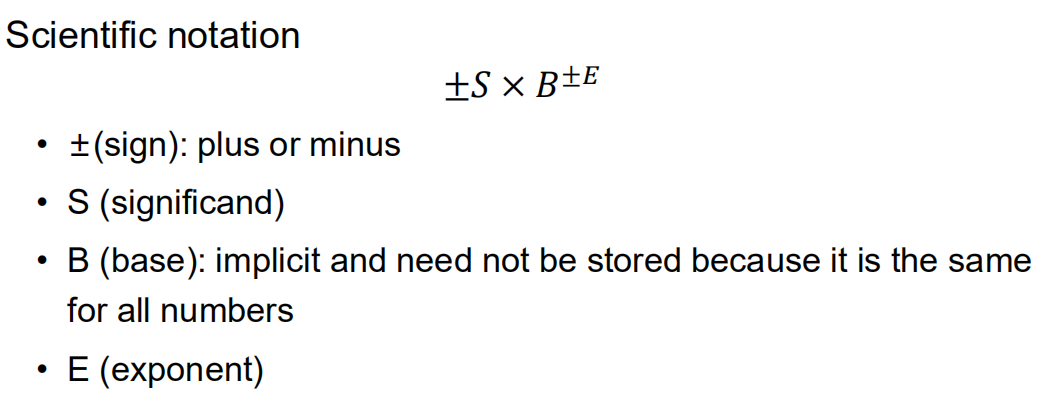
1.二进制k位最多只能表示2^k个不同值

2.二进制补码整数转十进制数值





3.浮点数



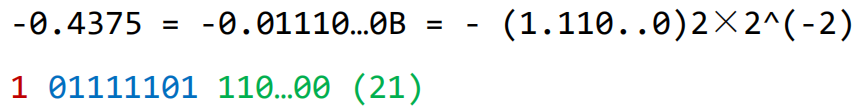
尾数，真正的第一位是1不需要存，存入的是1.xxx即首1后面的数字，23位

指数，存入的是真实指数+127之后的无符号整数，8位

基底全为2，不需存储

符号位，1位

e.g.十进制小数转二进制浮点数



‘-’→符号位为1

0.4375乘2取整数部分按序排列为二进制小数中小数点后的数字，浮动小数点直至整数位为1，得到以2为底的指数-2，+127=125，转为8位二进制无符号整数01111101，小数点后的数右侧补零成为23位尾数。

4.NBCD码

四位01串表示0~9：0000~1001

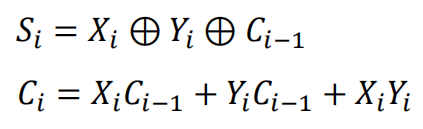
四位01串表示符号位：‘+’：1100/0，‘-’：1101/1

Lecture04 Integer Arithmetic

1.ALU输入：存储在寄存器中的待处理数据，控制信号

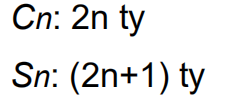
输出：存储在寄存器中的标志位数据

2.全加器：计算一位加法



1ty(三个与运算同时)+1ty(三元或运算)=2ty

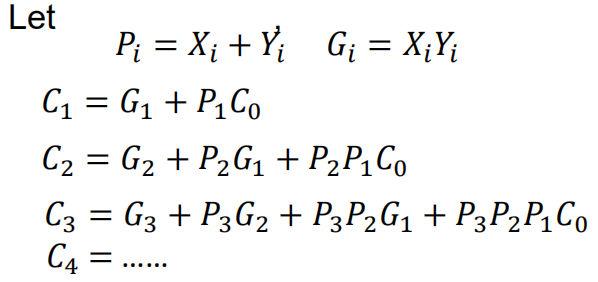
3ty+3ty=6ty(两个异或运算)



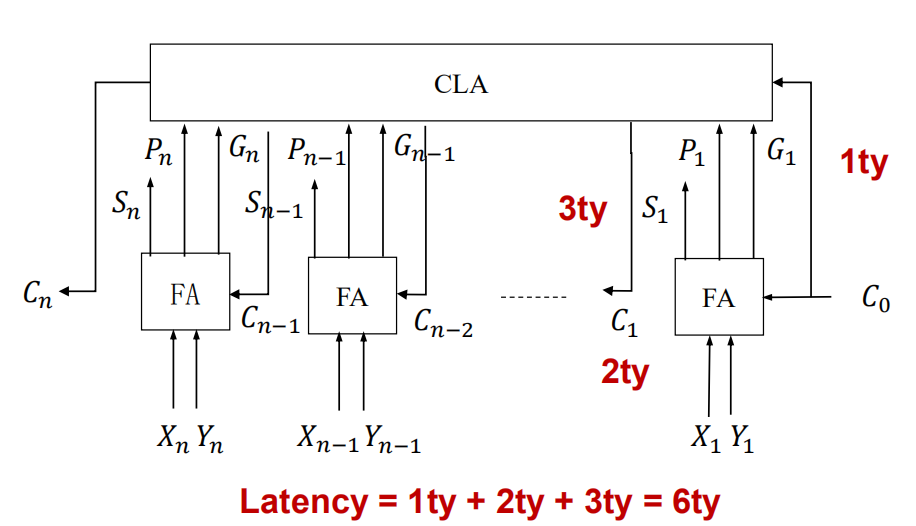
2(n-1)（等Cn-1传过来）+3（XnYn已经做了异或）=2n+1，n>=3

线性增长

3.超前进位加法器



Xi异或Yi的结果（需要的3ty和前两部操作同时完成）和Ci-1异或，还需要3ty，计算全部结果Si



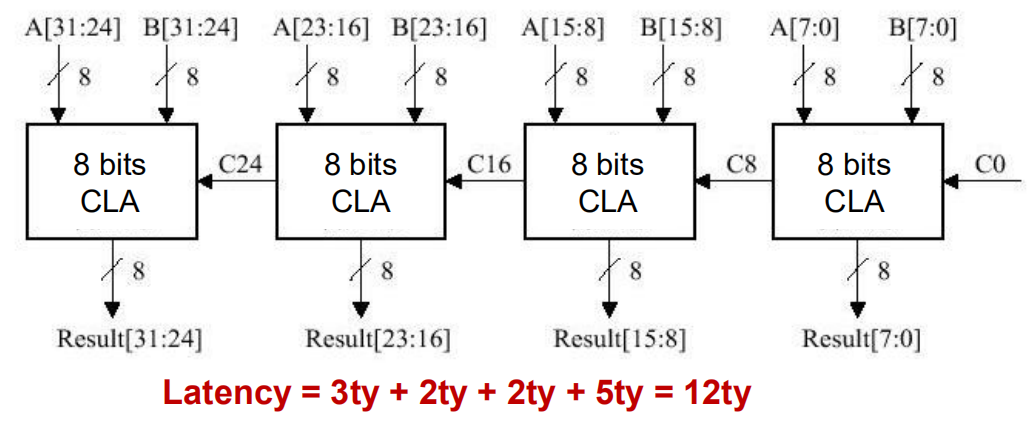
计算全部Ci

（先与后或）

计算全部Pi（XY或）

和Gi（XY与）

4.部分超前进位加法器



不想分析了，12ty记住吧

Lecture07 Internal Memory

1.寻址单元：可以被唯一表识的单个元

地址空间：可唯一标识的单元总数

寻址能力：存储在每个单元中的信息的位数

位：bit（b）字节：byte，1 byte=8 bit 字：由若干字节构成，字的位数称为字长，32位计算机字长为32

2.半导体存储器类型：

--Read-Write Memory

RAM（随机访问存储器）：

随机指访问不同位置耗时相同

具有易失性，必须有电，断电即消失

分为DRAM（动态），状态不稳定，需要不断刷新，漏电后充电，结构简单集成度高，便宜，用于大容量的主运存

和SRAM（静态），状态稳定，不需刷新、充电，结构复杂占体积大，更贵，用于更快的Cache

改进的DRAM还分为SDRAM（同步DRAM，CPU和内存共用一个时钟信号）

和DDR SDRAM（双倍速率SDRAM，一个时钟周期上升沿下降沿各传一次数据）

--Read-Only Memory

ROM（只读存储器）：

只读指只能写一次，接下来只能读不能改，也是随机访问的

非易失，断电依然在

用于计算机系统程序、函数表、固定不变的数据库等

容错性差，出错成本高

PROM（可编程ROM）：

还是只能写一次，但是出厂时没写，有第一次写的权限，用电写入

批量生产，便宜

--Read-Mostly Memory

EPROM（可擦除的PROM）：

以读操作为绝大多数，也可以偶尔的写多次

非易失，断电依然在

用电读和写

紫外线照射可擦，擦除耗时久（20分钟），擦完可重写

更贵，但是可以多次更新

EEPROM（用电擦除的EPROM）：

可随时写入，无需擦除先前内容，类似于RAM的直接覆盖

以字节（byte）为单位进行更新操作，写入速度慢（每字节几百微秒）

比EPROM更贵，密度也更低

Flash Memory（闪存的EEPROM）：

电擦除，先擦后才能写，不能直接覆盖

擦除以块为单位而不能以字节单位，与EEPROM寻址方式不同

密度高于EEPROM，到达和EPROM相同

EPROM和EEPROM在成本（价格）和功能（密度）上的中间体

3.寻址模式：字节寻址（8位）；字寻址（字长）

4.存储阵列与地址译码器

5.内存刷新逻辑

集中式刷新：暂停读写，按行刷新，其间不能操作，内存会一顿一顿

分散式刷新：完成读写操作后，在每个存储周期中刷新，增加每个读写周期的时间

异步刷新：每64ms内在当前内存行不在被使用的时候刷新，效率高最常用

6.模块管理

位扩展：扩展寻址单元位数，寻址能力增加

地址线不变，数据线增加

8个4K×1，扩展后变成4K×8，增加了位数

字扩展：扩展寻址单元个数，地址空间增多

地址线增加，数据线不变

8个4K×1，扩展后变成32K×1，增加了个数

同时扩展：寻址单元位数和个数都增加，数据线和地址线都增加

8个4K×1可以扩展成16K×2，位数个数都增加

7.主存=RAM+ROM；主存容量=RAM容量

Lecture08 Cache

1.操作：Check、Hit、Miss

2.Check的判断：Cache中存储了tag，通过tag算出所存储的数据对应到内存中的地址，check时检查所需要的数据的地址是否在就行

3.加入Cache的原因、为什么整块搬运：

局部性原理：时间局部性：较短时间内反复使用同一块数据

一次搬来，多次使用

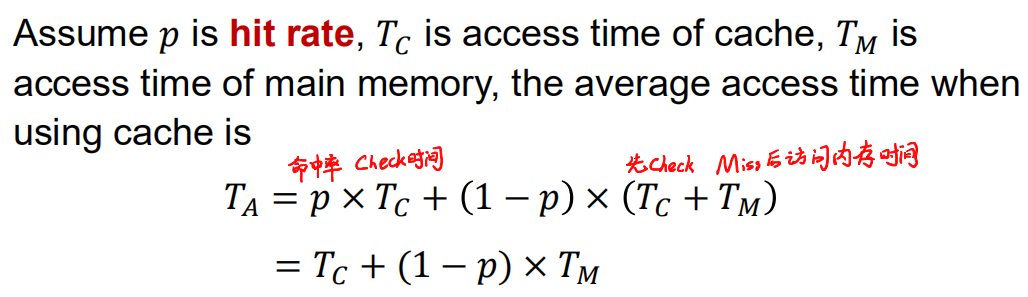
空间局部性：使用相近位置的数据

顺序局部性：按序访问数据

连同周围的数据一起搬，减少后续miss，

且整块搬运（t0+kt搬）比分段搬运（k(t0+t搬)）总时间短

4.平均获取时间

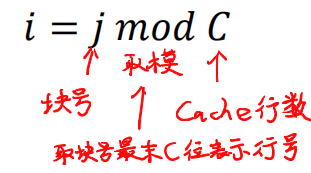


命中率p越大、Tc越小即Cache速度越快，性能越好

p=cache容量/内存容量，要使加入cache后的时间Ta>直接访问内存时间Tm，要求p>Tc/Tm

p随cache容量增长的增速先快后慢

5.映射策略

直接映射：内存中的每一块对应cache中唯一的一行  


取块号最末log C位表示行号

把内存分成j块，对应j个块号，用j对cache行数C取模（相除取余），二进制块号的末log C位即为映射到的行号

Cache中每行有一个tag，在直接映射时存储了块号高几位

内存容量决定地址长度，cache行数决定行号的位数（行号不用存在cache中），cache每行容量决定块内地址长度，地址中剩下的位数即为tag的位数，tag也被存在cache中。

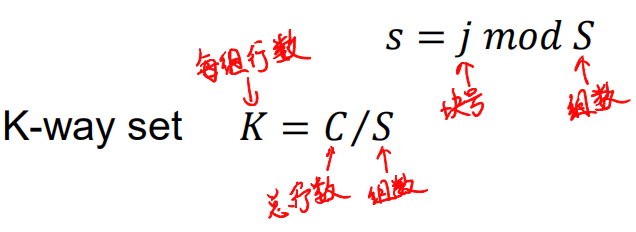
简单，映射和check快；容易抖动，适合大容量的cache

关联映射：内存中的每一块可以放在任意一行

Tag储存整个块号

避免抖动，替换策略复杂，check慢，适合小容量的cache

组关联映射：把cache分成若干组，规定内存块的组号（直接映射策略），在组内行位置任意（关联映射策略）



内存容量决定地址长度，cache行数/k路的结果为组数决定组号的位数（组号不用存在cache中），cache每行容量决定块内地址长度，地址中剩下的位数即为tag的位数，tag也被存在cache中。

比较：相关性（表示每一个内存块在cache中存放位置的可能情况数）

直接映射=1，关联映射=cache行数，组关联映射=每组行数

直接映射命中率最低，tag最短，check最快

关联映射命中率最高，tag最长，check最慢

6.替换策略（要求掌握两个）

LRU：最近最少使用的先出

FIFO：先进先出

7.改写策略：cache被改写时需要把新内容写回主存

写入（write through）：cache中内容每次被改写都写回一次

保证了始终一致，但主存和cache访问过于频繁

写回（write back）：cache在被替换时如果内容有被改写才写回主存

最小化了写操作，但主存和cache会出现短暂的不一致

7.多个cache

Ta =T1 + (1-p1)T2 +(1-p2)Tm（1-p1是一级不命中的概率，1-p2是两级都不命中的概率）

Lecture09 External Memory