COA Review

导论

概念

- 冯氏结构工作方式

  预先存放到MEM、自动/逐条取指令并执行

- 冯氏结构的改进

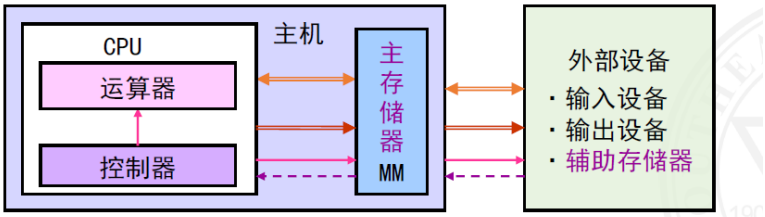
  ### 冯式结构的改进

  - **改进1：从“以运算器为中心”到“以存储器为中心”**

    - 运算与I/O并行（缓冲技术+DMA技术）
    - 提高性能

  - **改进2：从“单存储器”到“多种存储器共存”**

    - 效率差异化的存储结构，扩展存储空间 → 提高性价比
    - CPU（访存时）只直接访问主存

  冯式结构的改进：以存储器为中心 + 多种存储器共存

- 程序执行的过程
  - 工作方式
    - 采用存储程序的方式，程序和数据预先存放在存储器中
    - 机器工作时，自动、逐条地取出指令并执行

- 程序执行
  - 顺序型：指令在存储器中按执行顺序存放，由指令计数器（PC）指明下一条指令的地址，一般按顺序递增
  - 转移型：根据运算结果或外界条件，改变下一条指令的地址

主存储器

概念

- SRAM和DRAM
  - 比较与应用

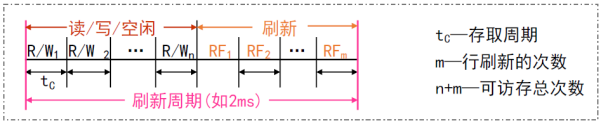
SRAM和DRAM的特性比较

对比项	SRAM	DRAM
存储信息	触发器	电容
破坏性读出	非	是
需要刷新	不要	需要
送行列地址	同时送	分两次送
运行速度	快	慢
集成度	低	高
发热量	大	小
存储成本	高	低
应用	Cache	主存

  - DRAM刷新

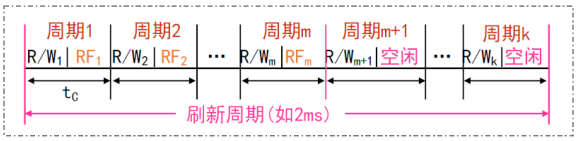
- **DRAM芯片的刷新**
  - 缘由：读出是破坏性的，读出后要立即对单元进行“重写”；信息用电容的充电状态保存，必须在电荷漏掉以前就进行充电。
  - 刷新（再生）：每隔一定时间给全部基本存储元的存储电容补充一次电荷。
  - 机制：采用“读出”方式进行刷新，接在数据线上的读出放大器也是一个再生放大器，在读出的同时，又使单元存储信息自动得以恢复。
  - 刷新方式：集中式、分散式、异步式、透明刷新。
  - 行刷新操作：同一行的所有存储元可同时刷新。
  - 刷新周期：从上一次整个存储器刷新结束到下一次对整个存储器全部刷新一遍为止的时间间隔。

    - 集中式
      - 集中式刷新：行刷新操作集中在一起连续操作。



特点：存在“死区”（不能进行读/写操作的时间段）。

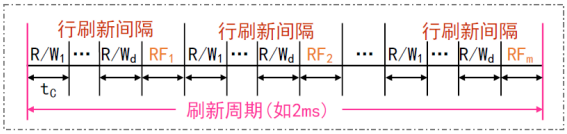
    - 分散式
      - 分散式刷新：行刷新操作分散在每个存储周期中。



特点：避免了“死区”，但增加存取周期。

    - 异步式

- 异步式刷新：行刷新操作均匀分布在刷新周期中。



特点：可忽略“死区”，存取周期不变（最常用）。

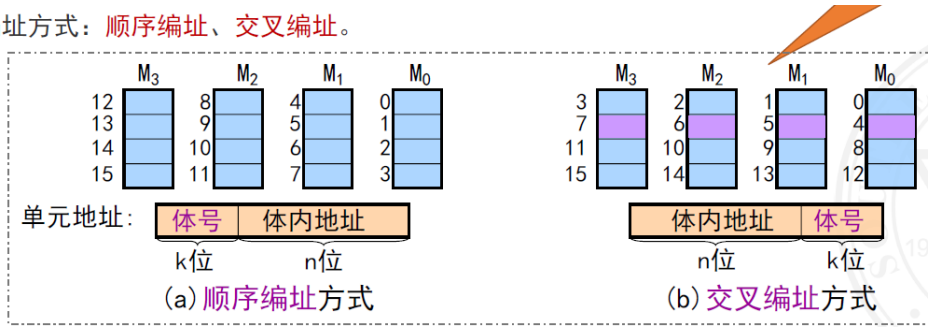
  - 透明式

    - 透明刷新：利用取址周期后的译码时间，此时存储器为空闲阶段，进行刷新操作。
      - ✓ 特点：不占用CPU时间。

- 编址方式

  - 顺序编制与交叉编址

    编址方式：顺序编址、交叉编址。



    交叉编址的同一列可被同时选中

公式

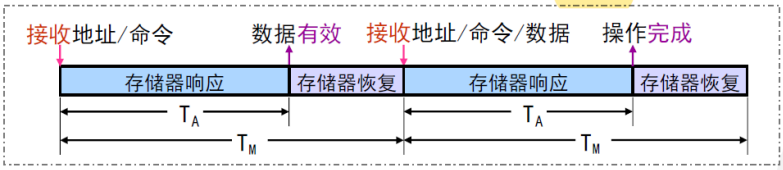
- 存储容量

  - 容量：存储单元个数 × 存储单元长度 = 存储字数 × 存储字长。

存储系统

概念

- 技术指标
  - 存储容量（S）：可存储的二进制位数，单位常为字节（B）。
  - 存取速度（B）：常用存取时间、存取周期表示。
    - ✓ 存取时间（T_A）：指存储器从收到命令到完成操作所需的时间。
    - ✓ 存取周期（T_M）：指连续访问存储器的最小时间间隔，$T_M=T_A+T_{恢复}$。



  - 传输速度：常用存储器带宽（B_M）表示。
    - ✓ 定义：指存储器的最大数据传输率，单位常为Mbps。
    - ✓ 公式：$B_M=W/T_M$，W为数据宽度（数据引脚位数）。

思考：某1K×8位SRAM芯片的存储周期为100ns，该芯片的带宽是多少？

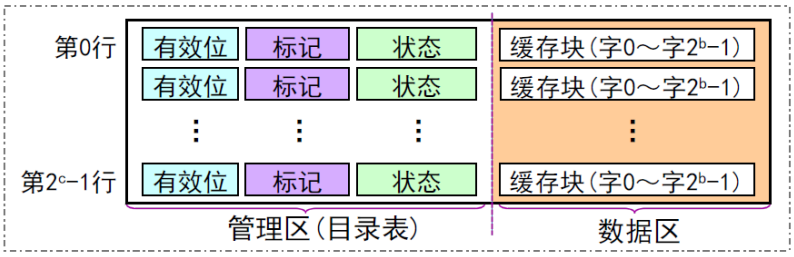
- 层次结构中的数据传递
  - ✓ CPU <==> Cache：以字为单位。
  - ✓ Cache <==> 主存：以块为单位，一个块由若干个字组成，是定长的。

✓ 辅存只与主存进行数据交换。

✓ 主存 <==> 辅存：以段、页或两者结合为单位。

- 核心问题（实现）
  - 地址映射
  - 替换算法
  - 写策略

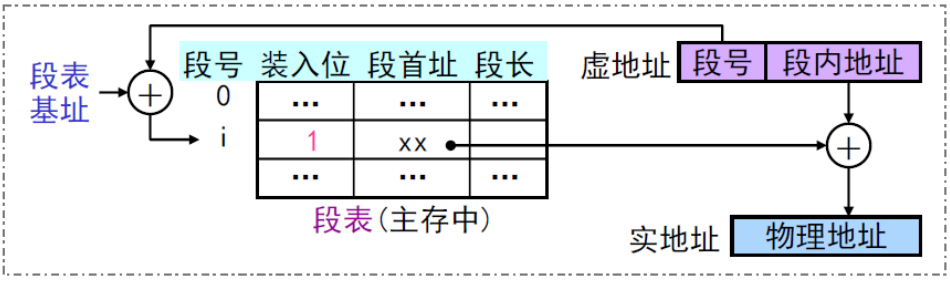
- Cache空间组织
  - 结构：Cache是行的数组，每行包含缓存块信息、管理信息。



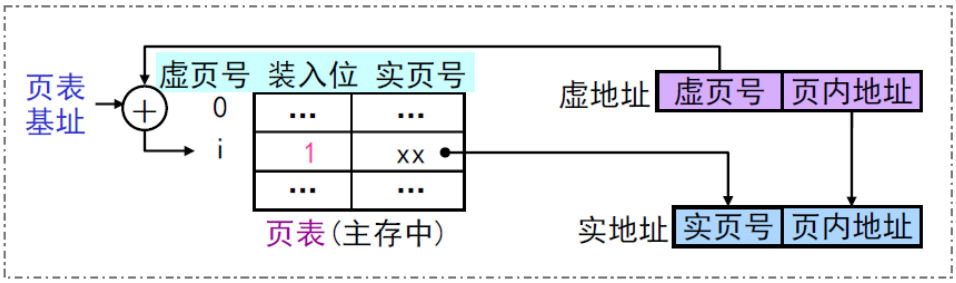
  - 容量：Cache数据区的容量称为Cache容量；Cache数据区、管理区容量之和称为Cache总容量。

Physic Address: Tag + Index + Byte-Offset

- 虚拟存储空间
  - 段式
    - 管理实现：用段表来管理。
      - ✓ 段表的结构：内容上，<段号，段内地址>；形式上，段表行数＝程序的段数。
      - ✓ 段表的组织：按段号进行索引。
    - 段表的存放位置：主存中。

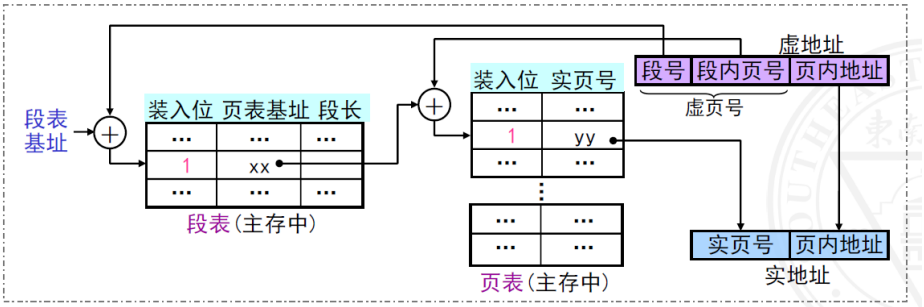


  - 页式
    - 管理方法：虚存、主存空间按页大小划分成若干个页，主存空间以页为单位进行分配。
    - 管理实现：用页表来管理。
      - ✓ 段表的结构：内容上，<页号，页内地址>；形式上，页表行数＝程序的页数。
      - ✓ 段表的组织：按虚页号进行索引。
    - 页表的存放位置：主存中。



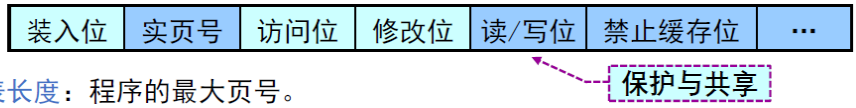
页式管理是基址寻址：虚地址中给出虚页号+页表基址 ---> 页表中找实页号

- 段页式
  - 管理方法：虚存空间先分段、再分页，主存空间只分页，主存空间以页为单位进行分配。
  - 管理实现：用一个段表、一组页表（每个段表1个页表）来管理。
  - 地址变换方法：两次查表（访存），步骤同段式、页式。
  - 特点：具有段式及页式优点，但需2次访存。

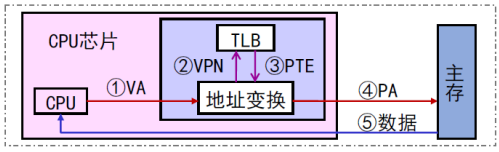


在页的基础上编段

- 页表（PT）的组织
  - 页表的索引方法：虚页号。
  - 页表项（Page Table Entry，PTE）的组织：无需标记字段。



  - 页表长度：程序的最大页号。
- 快表（TLB）的组织
  - TLB（Translation Lookaside Buffer）：旁路快表缓冲，也称为页表缓冲、地址变换高速缓存、快表。
    - ✓ 存储管理方法：组/全相联映射、LRU算法、写回法策略。
    - ✓ TLB条目组织：管理信息＋页表项信息。
  - 地址变换过程：先访问TLB（CPU的Cache中），TLB缺失时才访问页表。



公式

- 带宽（衡量传输速度）

公式：$B_M=W/T_M$，W为数据宽度（数据引脚位数）。

- Cache性能指标
  - ✓ 命中率：$H = N_c/(N_c+N_m)$ ＝ 命中Cache次数/访存总次数。
  - ✓ 平均访问时间：指Cache-主存层次完成访存操作平均所用时间。

$$T_A = H \cdot T_c + (1-H)(T_c+T_m) = T_c + (1-H)T_m = T_{命中} + (1-H)T_{缺失}$$

层间透明交换　不命中时访存的平均开销　缺失导致的停顿时间

T_c：访问Cache的时间
T_m：访问主存的时间

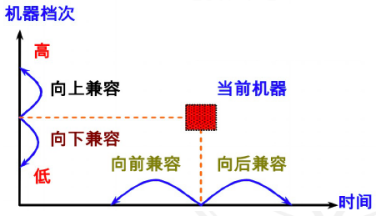
- * 辅助存储器的基本参数

- 存储密度
  - 存储密度：Ds＝道密度×位密度
    - ✓ 道密度(TPI)：磁盘半径方向单位长度包含的磁道数。
    - ✓ 位密度(BPI)：在每一个磁道内单位长度内所能记录的二进制信息。
- 存储容量
  - 存储容量：S＝磁道数×磁道长度×记录密度
    - ✓ 记录密度：由于磁盘需要格式化后才能使用，一般记录密度≤位密度。
- 寻址时间
  - 寻址时间：Ta＝平均寻道时间＋平均等待时间
    - ✓ 平均寻道时间：寻找磁道的平均时间。
    - ✓ 平均等待时间：寻找区域的平均时间，磁盘转半圈的时间。
- 数据传输率
  - 数据传输率：Dr＝磁道容量×磁盘转速

指令集

概念

- 指令系统的要求
  - **指令系统的要求**
    - 完备性
      - ✓ 指令系统直接提供的指令足够使用，相关功能不需用软件实现。（乘法指令 vs 加法指令）
    - 有效性
      - ✓ 指令级程序占据的存储空间小、执行速度快、能够高效运行。（访存效率 + 执行效率）
    - 规整性
      - ✓ 对称性、匀齐性、执行格式与数据格式的一致性。（变长指令 vs 定长指令）
    - 兼容性
      - ✓ 向上兼容：低档机器的软件能在高档机器上运行。
      - ✓ 向前兼容：早期机器的软件能在最新机器上运行。



- 各种字长

概念	含义	特征
存储字长	存储器中一个存储单元（存储地址）所存储的二进制信息的位数	可寻址最小单元，例如8位
指令字长	一条指令中所包含二进制信息的位数	m×存储字长，例如32位
数据字长	计算机各类型数据存储所占的二进制位数	n×存储字长，例如64位double
机器字长	CPU一次能处理数据的最大位数	k×存储字长，常数，例如32/64位机

- * 数据与指令的存放、寄存器长度

- 存放位置
  - ✓ 寄存器：寄存器长度 = 机器字长 = 最长数据的长度。
  - ✓ 存储器：一般以字节（8位）为最小单位存储，访问时可按字节、半字、字或双字进行。

- 一些寻址方式的别名

  例如2：将（存储器）间接寻址直接称为间接寻址，而将（寄存器）间接寻址成为寄存器间接寻址。

- 变址寻址

  若机器有基址寄存器存在，有效地址EA=基址寄存器内容+变址寄存器内容+形式地址A。

  变址寄存器的内容由用户设定，在程序执行过程中可变，但形式地址A的内容不可变。

公式

方法

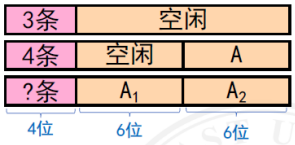
- 求解指令数

  例如：某指令系统中，有零指令、单指令、双指令3种指令格式，指令字长都为16位，每个地址码为6位。

  （1）若操作码采用定长编码，且已定义3条零地址指令、4条单地址指令，则双地址指令最多有多少条？

  （2）若操作码采用变长编码，且已定义48条零地址指令、14条双地址指令，则单地址指令最多有多少条？

  解：（1）操作码长度=16-6-6=4位，双地址指令的条数≤$2^4-3-4=9$。

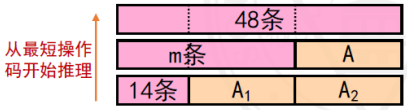


  （2）零地址、单地址、双地址指令的操作码长度分别为16位、10位和4位。

  前4位编码中，零地址和单地址指令可用编码数≤$2^4-14=2$，设单地址指令最多为m条，

  则前10位编码中，零地址指令可用的编码有$2\times2^6-m$个，即$2\times2^6-m=\lceil 48/2^6 \rceil$，

  解方程得，单地址指令最多有$m=2\times2^6-1=127$条。



  思考：定长编码和变长编码各自的优劣？

  设零地址N_0，单地址N_1，双地址N_2，则满足

$$N_0 + N_1 \cdot 2^6 + N_2 \cdot 2^{12} = 2^{16}$$

中央处理器

概念

- * 应用问题的产生与解决

应用（问题）的产生和解决

不仅取决于

- 算法
- 程序编写

而且取决于

- 语言处理系统
- 操作系统
- 指令集架构（ISA）
- 微体系结构

- 四个基本操作

✓ 操作的具体内容是一个包含基本操作的序列。

①REG间传送 — $R_Y \leftarrow (R_X)$

②存储器读 — $MDR \leftarrow M[(MAR)]$

③存储器写 — $M[(MAR)] \leftarrow (MDR)$

④算术逻辑运算 — $R_D \leftarrow (R_{S1})$ OP (R_{S2})

- 数据通路 - 哈佛结构

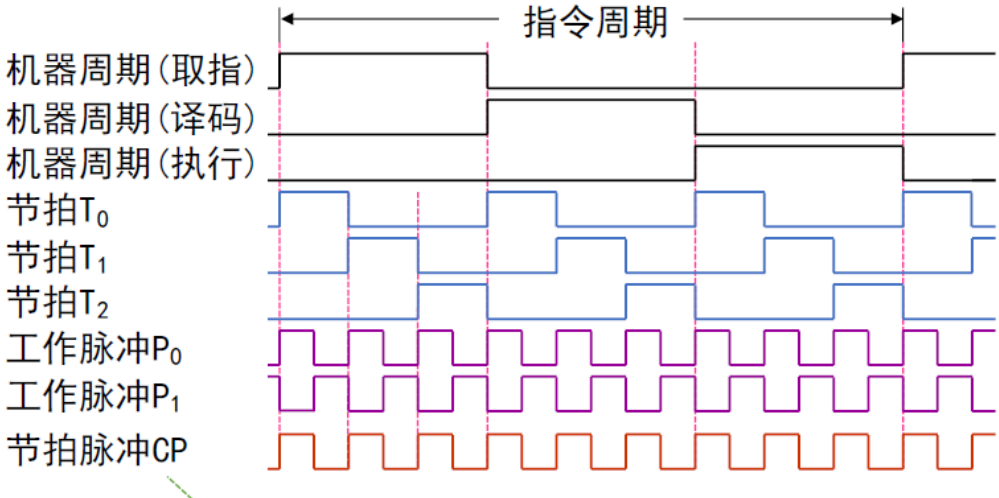
数据通路的功能部件（哈佛结构）

– 哈佛结构：一种将程序存储和数据存储分开的存储器结构，即程序存储器和数据存储器各自独立。

✓ 目的：减轻程序运行时的访存瓶颈。

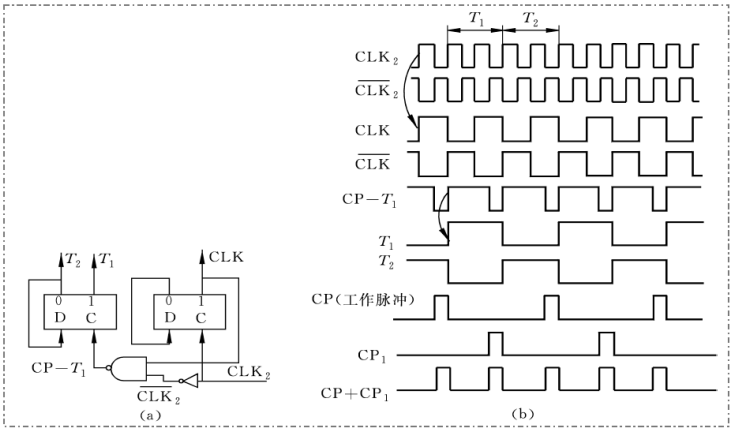
✓ 实现：CPU内的缓存中区分指令缓存和数据缓存，CPU外部仍采用冯·诺依曼结构。

- 时序信号



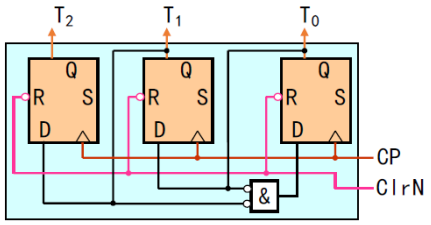
- 时序信号形成 - 变频

- **时序信号的形成电路（变换）**
  - 变换：分频、倍频、组合运算、延迟等。

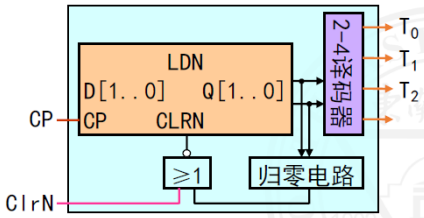


通过组合/时序逻辑电路实现

- **时序信号的形成电路（环形信号发生器）**
  - 功能：周期性地产生节拍信号序列，可以是变长或定长。
    - ✓ 节拍间的关系（以三个节拍为例）：①$T_0=\overline{T_1}\cdot\overline{T_2}$，②$T_1=\overline{T_0}\cdot\overline{T_2}$，③ $T_2=\overline{T_0}\cdot\overline{T_1}$。
    - ✓ 以移位寄存器为主： T_0的输入$D=T_2+\overline{T_0}\cdot\overline{T_1}=\overline{T_0}\cdot\overline{T_1}$。
    - ✓ 以计数器和译码器为主：循环计数，选择输出节拍。



以移位寄存器为主

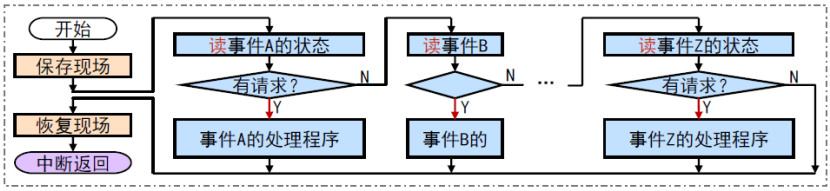


以计数器和译码器为主

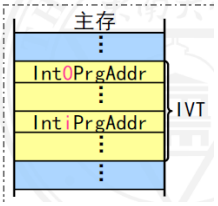
- 中断判断方法

中断源的判别方法

- 非向量方式：所有事件共用一个处理程序，入口地址固定。
  - ✓ 实现方法：PC←入口地址，其余子任务由处理程序完成。
  - ✓ 判优逻辑：事件的紧急程度由查找顺序决定。



- 向量方式：每个事件有一个处理程序，各入口地址保存在中断向量表中。
  - ✓ 中断向量表（IVT）：由操作系统管理，放在主存中。
  - ✓ 实现方法：判优、查表、PC←入口地址（全部由硬件实现）。
  - ✓ 判优逻辑：查找相关的寄存器、选择事件。



- CPU运行效率

CPU的运算效率

- CPU的主频：指CPU内部的主时钟脉冲的频率，记为f。
  - ✓ 单位：Hz（赫兹），1GHz=1×10³MHz=1×10⁶KHz=1×10⁹Hz。
  - ✓ （主）时钟周期：CPU主时钟脉冲的宽度，常记为T_C（$T_C=1/f$）。
- CPU时间：$T_{CPU}=I_N \times CPI \times T_C=\left(\sum_{i=1}^{n} I_i \times \mathrm{CPI}_i\right)\times T_C$
  - ✓ I_N：程序执行指令数（注意：和代码所含指令数存在差异）。 ➡ 工作量
  - ✓ CPI（Cycles Per Instruction）：一条指令执行所需的平均时钟周期数。
  - ✓ T_C：CPU主时钟周期（$=1/f$）。 ➡ 速度
  - ✓ n：指令系统的指令类型数。
  - ✓ CPI_i：第i种指令的CPI。
  - ✓ I_i：程序中第i种指令的执行次数。

- 流水线性能

程序执行时间

- $T_{串行}=n\cdot(m\Delta t)$
- $T_{流水}=m\Delta t+(n-1)\Delta t$

注：n为指令数、 m和Δt为段数和拍长

流水线的性能（假设流水线有m段、拍长为Δt，共执行n条指令）

- 吞吐率：单位时间内完成/输出的指令条数或结果数量。

$$T_P=\frac{n}{T_{流水}}=\frac{n}{m\Delta t+(n-1)\Delta t}$$

最大吞吐率：当n>>m时，$T_{P\ max}=1/\Delta t$，即拍长的倒数。

- 加速比：流水方式相对于串行方式的速度比。

$$S=\frac{串行方式执行时间}{流水方式执行时间}=\frac{n\cdot m\Delta t}{m\Delta t+(n-1)\Delta t}=\frac{nm}{m+n-1}$$

最大加速比：当n>>m时，$S_{max}=m$，即流水线段数。

- 效率：部件使用时间与整个执行时间的比值（平均值）。

$$E=\frac{n个任务所占时空区}{n个任务总的时空区}=\frac{n\cdot m\Delta t}{m(m+n-1)\Delta t}=\frac{n}{m+n-1}=\frac{S_P}{m}=T_P\cdot\Delta t$$

最高效率：当n>>m时，$E_{max}=1$。

-