第一章：**计算机系统概论**

1. **计算机的性能指标**
2. 冯诺依曼型计算机
3. 计算机硬件

第二章： **运算方法和运算器**

1.计算机中使用的数据可分成两大类：

2. 浮点数的表示方法

3. **十进制数串的表示方法**
4. **数的机器码表示**
5. 补码加法
6. 溢出
7. **基本的二进制加法/减法器**
8. **浮点加法**

第三章：内部存储器

1.存储器的分级
2.SRAM和DRAM
3.只读存储器与闪速存储器
4.存储器容量扩展的方法（3种）
5. **并行存储器**
6.cache存储器

第四章：指令系统

1.指令分类
2. 指令系统
3.计算机语言分为高级语言和低级语言。
4. **指令格式**
5. **指令字长度与机器字长**
6. **操作数类型**(4种)
7. **指令的寻址方式**（2种）
8. 操作数基本寻址方式
9. **指令的分类**
10. RISC指令系统的最大特点是

第五章 中央处理器

1.CPU的功能（4个）
2. CPU的基本组成
3. CPU中的主要寄存器（6类）
4.操作控制器
5.指令周期、机器周期、时钟周期
6.时序产生器
6.微命令、微操作、微指令、微程序
7.微程序控制器组成
8. **机器指令与微指令的关系**
9. **微程序设计技术**
10. **硬连线控制器**
11. 流水CPU

第六章：**总线系统**

1.定义
2.总线的分类
3.总线的特性
3. 总线带宽
4. 总线的连接方式(2种)
5.信息传送的方式
6.总线的仲裁
7. **总线数据传送模式**（4种）

第七章：外围设备

1. **磁盘阵列RAID**

第一章：计算机系统概论

1. 计算机的性能指标

- **吞吐量**：表征一台计算机在某一时间间隔内能够处理的信息量。
- **响应时间**：表征从输入有效到系统产生响应之间的时间度量，用时间单位来度量。
- **利用率**：在给定的时间间隔内系统被实际使用的时间所占的比率。
- **处理器字长**：指处理机运算器中一次能够完成二进制数运算的位数。
- **存储器容量**：存储器中所有存储单元的总数
- **存储器带宽**：单位时间内从存储器读出的二进制信息量，B/s
- **主频/时钟周期**
- **CPU执行时间**：表示一段程序执行过程中所占用的CPU时间。
- **CPI**：每条指令的周期，即执行一条指令所需的平均周期数
- **MIPS**：每秒百万指令数，即单位时间内执行的指令数
- **MFLOPS**：每秒百万次浮点操作次数，用来衡量机器浮点操作的性能

2. 冯诺依曼型计算机

- 设计思想：存储程序并按地址顺序执行
- 冯诺依曼结构与哈佛型结构的对比
  - 冯诺依曼结构：指令、数据在同一个存储器
  - 哈佛型结构：指令、数据分别放在两个存储器

3. 计算机硬件

控制器,运算器,存储器,输入设备,输出设备 (以运算器为中心)

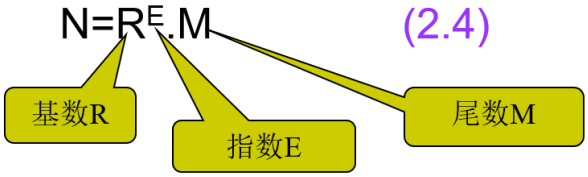
第二章： 运算方法和运算器

1.计算机中使用的数据可分成两大类：

- **符号数据**:非数字符号的表示（ASCII、汉字、图形等）
- **数值数据**:数字数据的表示方式
  - **定点表示**：小数点位置固定
  - **浮点表示**：小数点位置不固定

2. 浮点数的表示方法

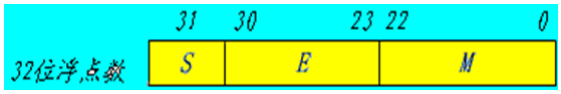
- 格式



  - M： **尾数**，是一个纯小数。
  - *e*： 比例因子的指数，称为浮点数的**指数**，是一个整数。
  - *R*： 比例因子的**基数**，对于二进计数值的机器是一个常数，一般规定 R 为2。

  一个机器浮点数由阶码和尾数及其符号位组成（**尾数**：用定点小数表示，给出有效数字的位数决定了浮点数的表示精度；**阶码**：用整数形式表示，指明小数点在数据中的位置，决定了浮点数的表示范围。）

- **IEEE754标准**（以单精度32 为例）



  - 规则中,尾数用原码,指数用移码(便于对阶和比较)
  - 32位的浮点数（单精度浮点数）：
    - 数的符号位，1位，在最高位，“0”表示正数，“1”表示负数。
    - M是尾数， 23位，在低位部分，采用纯小数表示
    - E是阶码，8位，采用移码表示。移码比较大小方便。
  - **规格化：**为提高数据的表示精度，当尾数的值不为 0 时，尾数域的最高有效位应为1,否则以修改阶码同时左右移小数点的办法，使其变成这一表示形式，这称为浮点数的**规格化表示。**
    - 一个规格化的32位浮点数x的真值表示为：

      $x=(-1)^S\times(1.M)\times2^{E-127}$　e=E-127

3. 十进制数串的表示方法

- **十进制编码**：BCD码或ASCII码的低4位
- **字符串形式**：一个字节存放一个十进制的数位或符号位。为了指明这样一个数，需要给出该数在主存中的起始地址和位数(串的长度)。
- **压缩的十进制数串形式**：一个字节存放两个十进制的数位。它比前一种形式节省存储空间，又便于直接完成十进制数的算术运算，是广泛采用的较为理想的方法。

4. 数的机器码表示

- 原码
  - 有正0和负0之分。
  - **实现乘除运算规则简单。进行加减运算十分麻烦。**
- 补码
  - 高位表明正负
  - 无正零和负零之分
  - 补码表示对加减法运算十分方便，乘除运算复杂。
  - **补码求法**
    - 正数，与原码相同

        - 负数，符号位不变，反码最后一位+1
        - 举例

          x 1 = - 1110 $[x1]_{反}$=1 0001 $[x1]_{补}$=1 0010

- 反码
    - 反码表示有正0和负0之分
    - **反码求法**
        - 正数，与原码相同
        - 负数，负数的反码符号位为1，数值位是将原码的数值按位取反，就得到该数的反码表示。
- 移码
    - 用在阶码中

5. 补码加法

- 公式

  $[x+y]_{补}=[x]_{补}+[y]_{补}$

  $[x-y]_{补}=[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}$

- 从$[y]_{补}$求$[-y]_{补}$的法则是：对$[y]_{补}$包括符号位“**求反且最末位加1**”,即可得到$[-y]_{补}$。
    - 举例

      $[x]_{补}$ = **0**1101

      $[-x]_{补}$ = **1**0011

6. 溢出

- 概念：在运算过程中如出现大于字长绝对值的现象,称为“溢出”。
- 为了判断“溢出”是否发生，检测方法：
    - 双符号位法(“变形补码”)

      Sf1 Sf2

        - 0 0 正确（正数）
        - 0 1 上溢 （正溢出）
        - 1 0 下溢（负溢出）
        - 1 1 正确（负数）

7.基本的二进制加法/减法器

- 半加器 不考虑进位
- 全加器 考虑进位
- 先行进位加法器: 设计出专门的电路，使得每一位的进位能够并行地产生而与低位的运算情况无关。

8. 浮点加法

- 运算步骤
    - **0 操作数的检查,看有无简化操作的可能**；

      如果判知两个操作数 x 或 y 中有一个数为0，即可知没有必要再进行后续的一系列操作以节省运算时间。
    - **比较阶码大小并完成对阶**；

两浮点数进行加减，首先要看两数的阶码是否相同。若二数阶码相同表示小数点是对齐的，可以进行尾数的加减运算；若二数阶码不同表示小数点位置没有对齐，此时必须使二数阶码相同这个过程叫作**对阶。**在对阶时总是使**小阶向大阶看齐**，即小阶的尾数向右移位(相当于小数点左移)，每右移一位其阶码加1，直到阶码相等。

- **尾数进行加或减运算；**

  不论加法运算还是减法运算，都按加法进行操作，其方法与定点加减法运算完全一样。

- **结果规格化并进行舍入处理**

  在浮点加减运算时，尾数求和的结果也可以得到01.ф...ф或10.ф...ф，即两符号位不等，这在定点加减法运算中称为溢出，是不允许的。但在浮点运算中,它表明尾数求和结果的绝对值大于1，向左破坏了规格化。此时将运算结果右移以实现规格化表示称为**向右规格化。**规则是：尾数右移1位阶码加1。当尾数不是1.M时需向左规格化。

  对阶或向右规格化时尾数要向右移位，这样被右移的尾数的低位部分会被丢掉，从而造成一定误差，因此要进行**舍入处理。**简单的舍入方法有两种：一种是"0舍1入"法。另一种是"恒置一"法。

- IEEE754标准的四种舍入处理方法：
  - **就近舍入** 其实质就是通常所说的“四舍五入”。
  - **朝0舍入** 即朝数轴原点方向舍入，就是简单的截尾。这种方法容易导致误差积累。
  - **朝 + ∞舍入** 对正数来说,只要多余位不全为0则向最低有效位进1；对负数来说则是简单的截尾。
  - **朝 - ∞舍入** 处理方法正好与 朝 + ∞舍入情况相反。对正数来说，则是简单截尾；对负数来说，只要多余位不全为0则向最低有效位进1。
- 浮点数的溢出

  浮点数的溢出是以其**阶码溢出**表现出来的。在加\减运算过程中要检查是否产生了溢出：若阶码正常，加(减)运算正常结束；若阶码溢出，则要进行相应处理。另外对尾数的溢出也需要处理。

第三章：内部存储器

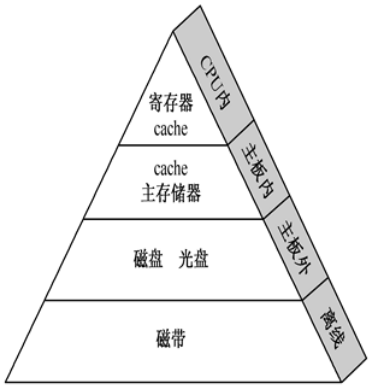
1.存储器的分级

对存储器的要求是**容量大、速度快、成本低。**为了解决了这三方面的矛盾，计算机采用多级存储体系结构，即cache、主存和外存。CPU能直接方问内存(cache、主存），但不能直接访问外存。

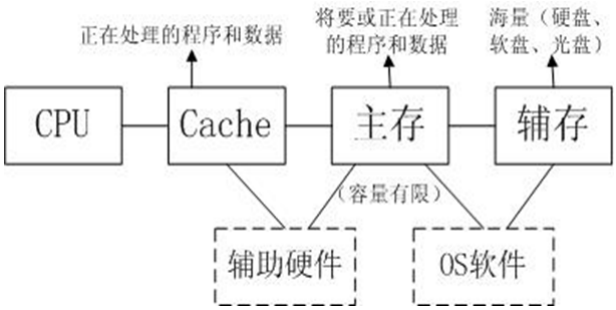
高速缓冲存储器简称cache，它是计算机系统中的一个高速小容量半导体存储器。

主存储器简称主存，是计算机系统的主要存储器，用来存放计算机运行期间的大量程序和数据。

外存储器简称外存、辅存，它是大容量辅助存储器。



- 分层存储器系统之间的连接关系



- 主存储器的技术指标

指标	含　义	表　现	单 位
存储容量	在一个存储器中可以容纳的存储单元总数	存储空间的 大小	KB.MB.GB.TB
存取时间	一次读操作命令发出到该操作完成，将数据读出到数据总线上所经历的时间	主存的速度	ns
存储周期	连续启动两次读操作所 需间隔的最小时间	主存的速度	ns
存储器 带宽	单位时间里存储器所存取的信息量	数据传输速 率技术指标	位/秒，字节/秒

2.SRAM和DRAM

静态读写存储器(SRAM)：存取速度快。

动态读写存储器(DRAM)：存储容量大。

- SRAM与DRAM比较

比较内容	SRAM	DRAM
存储信息0和1的方式	双稳态触发器	极间电容上的电荷
电源不掉电时	信息稳定	信息会丢失
刷新	不需要	需要
集成度	低	高
容量	小	大
价格	高	低
速度	快	慢
适用场合	Cache	主存

3.只读存储器与闪速存储器

- ROM(只读存储器)
  - 掩模ROM
  - 可编程ROM
    - PROM 一次性编程。

    - EPROM 叫做光擦除可编程可读存储器。可多次编程。
    - E^2PROM 叫做电擦除可编程只读存储器。可多次编程。
- FLASH存储器：也叫闪速存储器，它是高密度、非失易失性的读/写存储器。它既有RAM的优点，又有ROM的优点。
- 几种非易失性存储器的比较

存储器	类别	擦除方式	能否单字节修改	写机制
MROM	只读	不允许	否	掩膜位写
PROM	写一次读多次	不允许	否	电信号
EPROM	写多次读多次	紫外线擦除，脱机改写	否	电信号
E^2PROM	写多次读多次	电擦除，在线改写	能	电信号
Flash Memory	写多次读多次	电擦除，在线改写	否	电信号

4.存储器容量扩展的方法（3种）

- **位扩展**
    - 给定的芯片字长位数较短，不满足设计要求的存储器字长，此时需要用多片给定芯片扩展字长位数。
    - 三组信号线中，地址线和控制线公用而数据线单独分开连接。

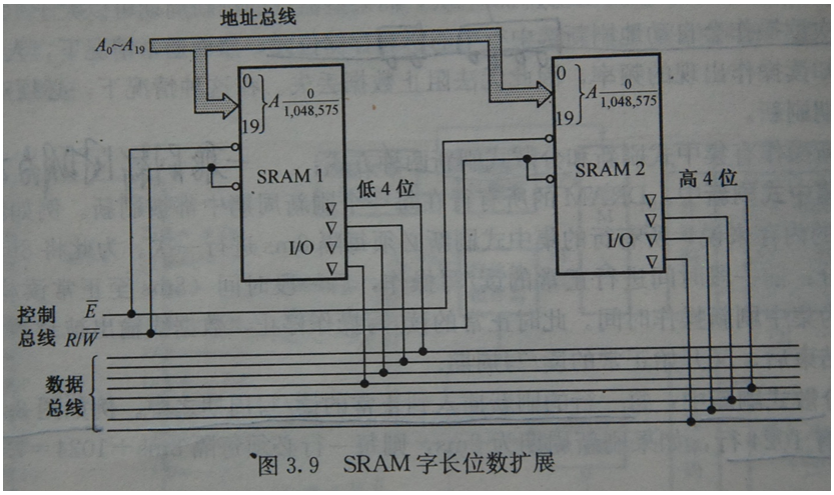


图 3.9　SRAM 字长位数扩展

- **字扩展**
    - 给定的芯片存储容量较小（字数少），不满足设计要求的总存储容量，此时需要用多片给定芯片来扩展字数。
    - 三组信号组中给定芯片的地址总线和数据总线公用，控制总线中R/W公用，使能端EN不能公用，它由地址总线的高位段译码来决定片选信号。
        - 存储器地址线的低若干位连接各芯片的地址线
        - 存储器地址线的高若干位作用于各芯片的片选信号CS#。

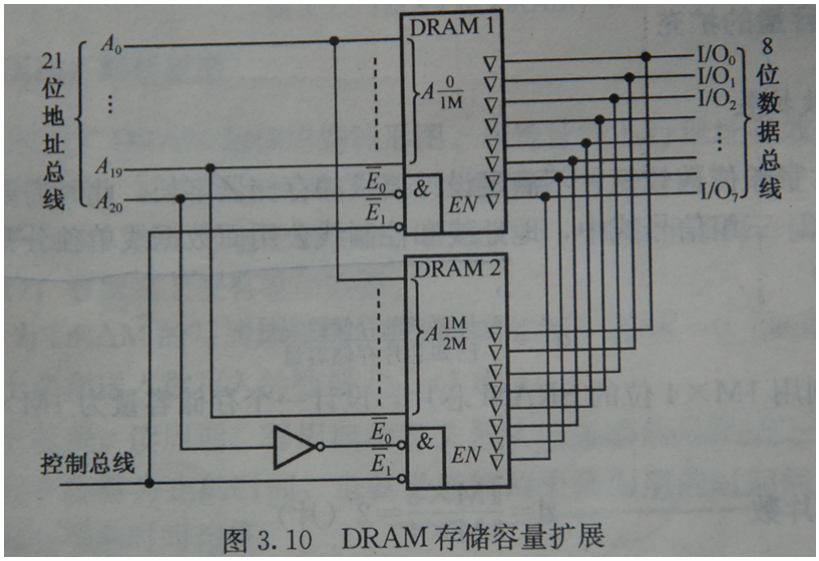


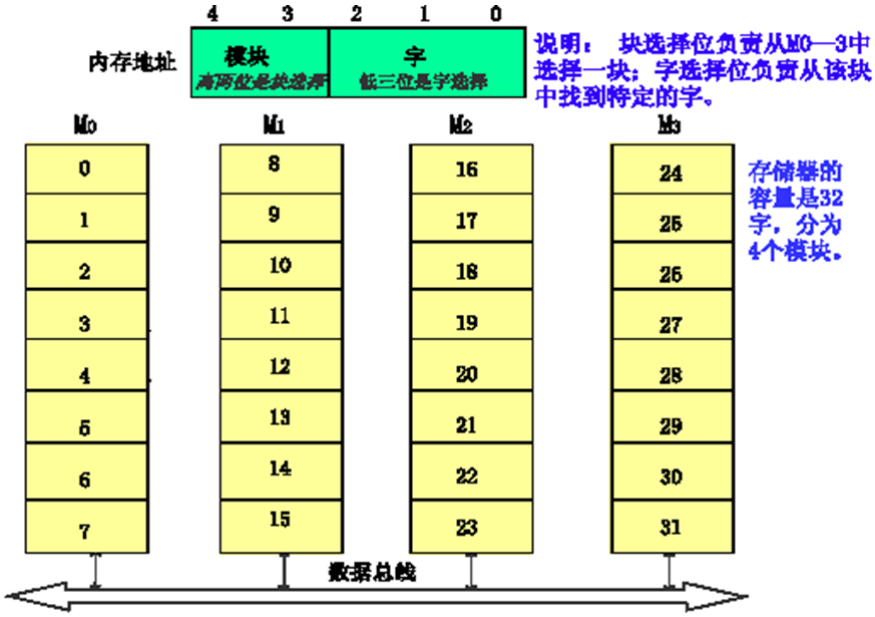
图 3.10　DRAM 存储容量扩展

- **字位扩展**

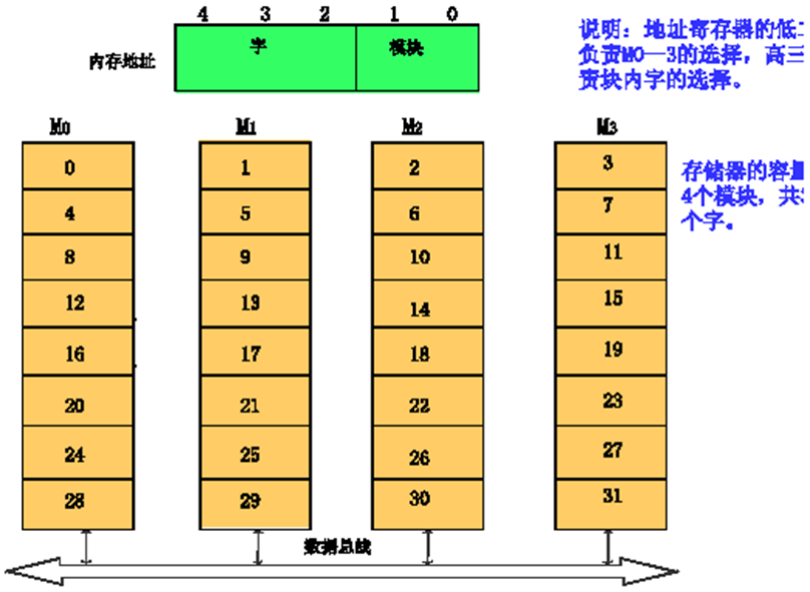
5.并行存储器

- **双端口存储器**
  - 双端口存储器由于同一个存储器具有两组相互独立的读写控制电路而得名。
  - **无冲突读写控制**：当两个端口的地址不相同时，在两个端口上进行读写操作，一定不会发生冲突。
  - **有冲突读写控制**：当两个端口同时存取存储器同一存储单元时，便发生读写冲突。
- **多模块交叉存储器**

  存储器的模块化组织
  - 顺序方式：某个模块进行存取时，其他模块不工作
    - 高位选模块，低位选块内地址
    - 优点：某一模块出现故障时，其他模块可以照常工作，通过增添模块来扩充存储器容量比较方便。
    - 缺点：但各模块串行工作，存储器的带宽受到了限制。



  - 交叉方式：
    - 高位选块内地址，低位选模块
    - 同一个模块内的地址都是不连续的。对连续字的成块传送可实现多模块流水式并行存取，大大提高存储器的带宽。



6.cache存储器

6.1 cache的功能

- Cache是介于CPU和主存之间的小容量存储器，存取速度比主存快。它是为了解决CPU和主存之间速度不匹配而采用的一项重要技术。
- 一般采用高速的SRAM构成。全由硬件调度，对用户透明。

6.2 cache的基本原理

- 当CPU读取主存中一个字时，便发出此字的内存地址到cache和主存。此时cache控制逻辑依据地址判断此字当前是否在 cache中：若是，此字立即传送给CPU；若非，则用主存读周期把此字从主存读出送到CPU，与此同时，把含有这个字的整个数据块从主存读出送到cache中。

6.3 cache的设计依据

CPU这次访问过的数据，**下次**有很大的可能也是访问**附近的**数据。

6.4 cache的命中率

- 若t_c表示命中时的cache访问时间，t_m表示未命中时的主存访问时间，1-h表示未命中率，则cache/主存系统的平均访问时间t_a为： $\mathbf{t_a=ht_c+(1-h)t_m}$
- 追求的目标是：以较小的硬件代价使cache/主存系统的平均访问时间t_a越接近t_c越好。

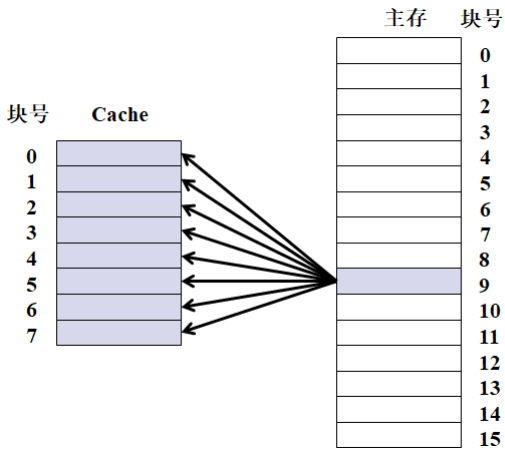
6.5 主存与Cache的地址映射（3种）

地址映射即是应用某种方法把主存地址定位到cache中。

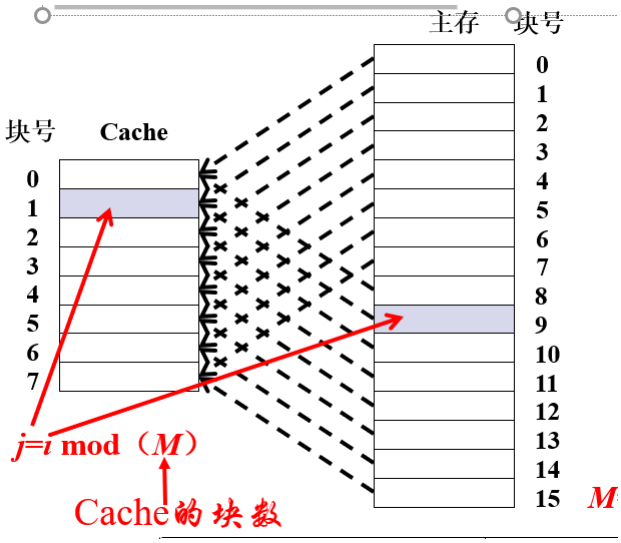
- **全相联映射方式**
  - 优点:

    可使主存的一个块直接拷贝到cache中的任意一行上，非常灵活。
  - 缺点:

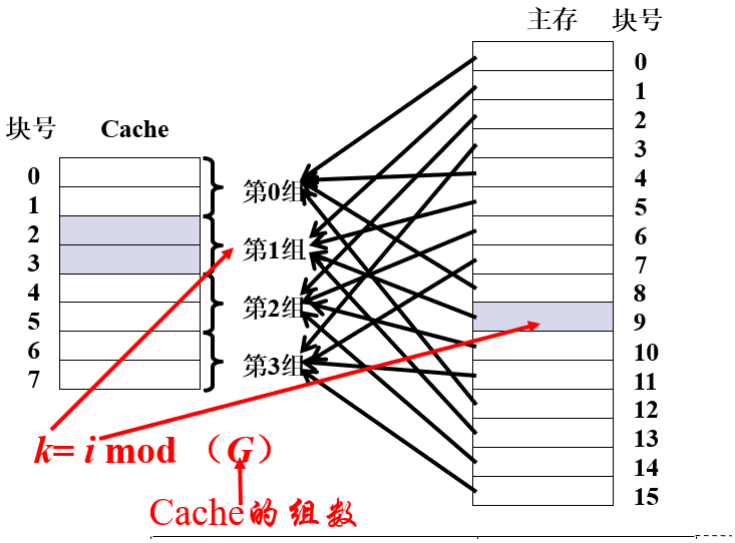
    它的主要缺点是比较器电路难于设计和实现，因此只适合于小容量cache采用。
  - 全相联是指主存中的任一块可以被放置到Cache中的任意一个位置的方法。



- **直接映射方式**
  - 优点:直接映射方式的优点是硬件简单，成本低。
  - 缺点:缺点是每个主存块只有一个固定的行位置可存放,容易产生冲突。因此适合大容量cache采用。
  - 直接映象是指主存中的每一块只能被放置到Cache中的唯一的一个位置 。

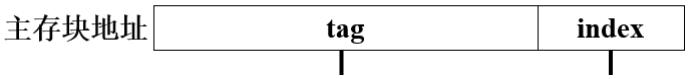


- **组相联映射方式**
  - 全相联和直接映射的折中
  - 将cache分成u组，每组v行，主存块存放到哪个组是固定的，至于存到该组哪一行是灵活的.
  - 组相联是指主存中的每一块可以被放置到Cache中唯一的一个组中的任何位置（Cache被等分为若干组，每组由若干个块构成）。



6.6 查找算法

- 通过查找目录表来实现。
- Cache中设有一个目录表，该表所包含的项数与Cache的块数相同，每一项对应于Cache中的一块，用于指出当前该块中存放的信息是哪个主存块的。它实际上记录了该主存块的块地址的高位部分，称为标识（tag）。每个主存块能唯一地由其标识来确定。



6.7 替换算法（3种）

对直接映射的cache来说，只要把此特定位置上的原主存块换出cache即可。对全相联和组相联cache来说， 就要从允许存放新主存块的若干特定行中选取一行换出。

- 最不经常使用(LFU)算法
  - LFU算法将一段时间内被访问次数最少的那行数据换出。每行设置一个计数器。从0开始计数，每访问一次， 被访行的计数器增1。当需要替换时，将计数值最小的行换出，同时将这些行的计数器都清零。
  - 这种算法将计数周期限定在对这些特定行两次替换之间的间隔时间内，不能严格反映近期访问情况。
- 近期最少使用(LRU)算法
  - LRU算法将近期内长久未被访问过的行换出。每行也设置一个计数器，cache每命中一次，命中行计数器清零，其它各行计数器增1。当需要替换时，将计数值最大的行换出。
  - 这种算法保护了刚拷贝到cache中的新数据行，有较高的命中率。
- 随机替换
  - 随机替换策略从特定的行位置中随机地选取一行换出。在硬件上容易实现，且速度也比前两种策略快。
  - 缺点是降低了命中率和cache工作效率。

6.8 写操作策略（3种）

cache的内容应当与主存内容保持一致。当CPU对cache进行写操作时更改了cache的内容，所以需要更改的数据回写到相应主存中。

- 写回法
  - 当CPU写cache命中时，只修改cache的内容，而不立即写入主存；**只有当此行被换出时才写回主存**。这种方法减少了访问主存的次数,但是存在不一致性的隐患。
  - 实现这种方法时，每个cache行必须配置一个修改位，以反映此行是否被CPU修改过。
- 全写法
  - 当写cache命中时，cache与主存同时发生写修改，因而较好地维护了cache与主存的内容的一致性。
  - 当写cache未命中时，直接向主存进行写入。cache中每行无需设置一个修改位以及相应的判断逻辑。缺点是降低了cache的功效。
- 写一次法
  - 基于写回法并结合全写法的写策略，写命中与写未命中的处理方法与写回法基本相同，只是第一次写命中时要同时写入主存。这样使其它使用该块数据的能即时进行标识或作废处理，以便于维护系统全部cache的一致性

第四章：指令系统

1.指令分类

从计算机组成的层次结构来说，计算机的指令有微指令、机器指令和宏指令之分。

- 微指令：微程序级的命令，它属于硬件。
- 机器指令（指令）：介于微指令与宏指令之间，每条指令可完成一个独立的算术运算或逻辑运算。
- 宏指令：由若干条机器指令组成的软件指令，它属于软件。

2. 指令系统

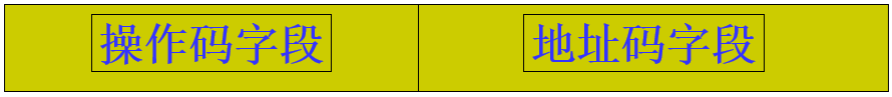
- 概念：指令系统：一台计算机中所有机器指令的集合
- 分类
  - 复杂指令系统计算机(CISC)
  - 精简指令系统计算机（RISC）
- 一个完善的指令系统应满足如下四方面的要求：
  - **完备性** 指令丰富、功能齐全、使用方便。
  - **有效性** 程序执行时占据存储空间小、执行速度快
  - **规整性**
    - 对称性(所有寄存器、存储单元同等对待)、匀齐性（一种操作支持多种数据类型）
    - 指令格式和数据格式的一 致性（指令长度和数据长度通常是字节的整数倍）：
  - **兼容性** 系列机软件 “向上兼容”

3.计算机语言分为高级语言和低级语言。

- **高级语言**如C，FORTRAN等，其语句和用法与具体机器的指令系统无关。
- **低级语言**分机器语言（二进制语言）和汇编语言（符号语言），这两种语言都是**面向机器**的语言，和具体机器的指令系统密切相关。
  - 机器语言用指令代码编写程序
  - 汇编语言用指令助记符来编写程序

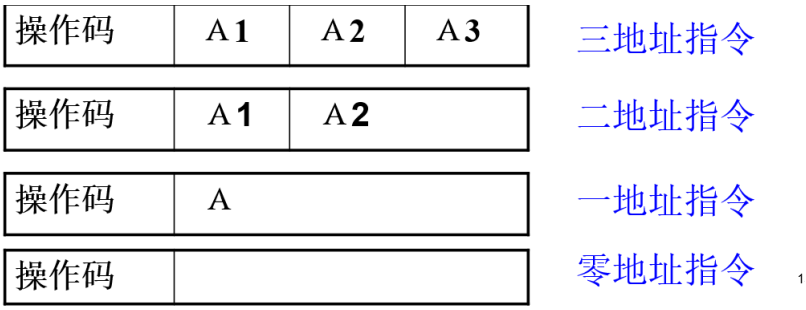
4. 指令格式

指令字用二进制代码表示的结构形式，包括两个方面：



11

- **操作码**：表示该指令应进行什么性质的操作
- **地址码**：用来描述指令的操作对象



1

- 在二地址和三地址指令格式中，从操作数的物理位置来说，又可归结为三种类型。
  - **存储器存储器(SS)型指令**：操作时都是涉及内存单元，参与操作的数都放在内存里，从内存某单元中取操作数，操作结果存放至内存另一单元中，因此机器执行这种指令需要多次访问内

存。
    - **寄存器寄存器 (RR) 型指令**：需要多个通用寄存器或个别专用寄存器，从寄存器中取操作数，把操作结果放到另一寄存器。机器执行寄存器-寄存器型指令的速度很快，因为执行这类指令，不需要访问内存。
    - **寄存器存储器 (RS) 型指令**：执行此类指令时，既要访问内存单元，又要访问寄存器。

5. 指令字长度与机器字长

- **指令字长度**：一个指令字中包含二进制代码的位数。
- **机器字长：**计算机能直接处理的二进制数据的位数，它决定了计算机的运算精度。
- 指令字长度分为：单字长、半字长、双字长三种形式。
- **多字长指令**的优缺点
    - 优点提供足够的地址位来解决访问内存任何单元的寻址问题；
    - 缺点必须两次或多次访问内存以取出一整条指令，降低了CPU的运算速度，又占用了更多的存储空间。
- **等长指令字结构**：各种指令字长度是相等的，指令字结构简单，且指令字长度是不变的；
- **变长指令字结构**：各种指令字长度随指令功能而异，结构灵活，能充分利用指令长度，但指令的控制较复杂 。

6. 操作数类型(4种)

- 地址数据：地址实际上也是一种形式的数据，被认为是无符号整数。
- 数值数据：计算机中普遍使用的三种类型的数值数据，定点整数或小数；浮点数；压缩十进制。
- 字符数据：文本数据或字符串，目前广泛使用ASCII码。
- 逻辑数据：一个单元中有几位二进制bit项组成，每个bit的值可以是1或0。当数据以这种方式看待时，称为逻辑性数据

7. 指令的寻址方式（2种）

- **顺序寻址方式**

  必须使用**程序计数器**（又称**指令指针寄存器**）PC来计数指令的顺序号，该顺序号就是指令在内存中的地址

- **跳跃寻址方式**

  所谓**跳跃**，是指**下条指令的地址码**不是由PC给出，而是**由本条指令给出。**

8. 操作数基本寻址方式

形成操作数的有效地址的方法，称为**操作数的寻址方式。**

- **隐含寻址**：在指令中**不明显的给出**而是**隐含**着操作数的地址。
- **立即寻址**: 指令的地址字段指出的不是操作数的地址，而是**操作数本身。**
- **直接寻址**:操作数的地址直接给出，而不需要经过某种变换
- **间接寻址**：指令地址字段中的形式地址A不是操作数D的真正地址，而是操作数地址的指示器，A单元的内容才是操作数的有效地址。
- **寄存器寻址**：操作数不在存储器中，而是在CPU内某一通用寄存器中。
- **寄存器间接寻址**：寄存器中**存放的**不是操作数，而是操作数的存储器地址
- **偏移寻址**：偏移寻址是直接寻址和寄存器寻址方式的结合。常用的三种偏移寻址是相对寻址、基址寻址、变址寻址。
    - **相对寻址方式**：隐含引用的专用寄存器是**程序计数器(PC)**，即有效地址EA = A + (PC)。
    - **基址寻址方式**：被引用的专用寄存器（**基址寄存器**）含有一个存储器地址。它的优点是**可以扩大寻址能力。**

    - **变址寻址方式**：把CPU中某个专用寄存器**（变址寄存器）**的内容与偏移量D相加来形成操作数有效地址。I变址寻址方式的目的而在于实现程序块的**规律性变化。**
- **段寻址方式**：这种寻址方式的实质还是基址寻址。
- **堆栈寻址方式** ： 堆栈有寄存器堆栈和存储器堆栈两种形式，都以**先进后出**的方式存储数据。

9. 指令的分类

- **数据传送指令**
- **算术运算指令**
- **逻辑运算指令**
- **程序控制指令**
- **输入输出指令**
- **字符串处理指令**
- **特权指令**
- **其他指令**

10. RISC指令系统的最大特点是

- ①指令条数少；
- ②指令长度固定，指令格式和寻址方式种类少；
- ③只有取数/存数指令访问存储器，其余指令的操作均在寄存器之间进行。

第五章 中央处理器

1.CPU的功能（4个）

- **指令控制** 。程序的顺序控制称为指令控制。由于程序是一个指令序列，这些指令的相互顺序不能任意颠倒，必须严格按程序规定的顺序进行。
- **操作控制**。一条指令的功能往往是由若干个操作信号的组合来实现的，因此，CPU管理并产生由内存取出的每条指令的操作信号，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。
- **时间控制** 。对各种操作实施时间上的定时称为时间控制。在计算机中，各种指令的操作信号以及一条指令的整个执行过程都受到时间的严格定时。
- **数据加工** 。数据加工就是对数据进行算术运算和逻辑运算处理。

2. CPU的基本组成

运算器、cache和控制器三大部分。

- 运算器
    - 功能
        - 执行所有的算术运算；
        - 执行所有的逻辑运算，并进行逻辑测试，如零值测试或两个值的比较。
- 控制器
    - 功能
        - 从内存中取出一条指令，并指出下一条指令在内存中的位置；
        - 对指令进行译码或测试，并产生相应的操作控制信号，以便启动规定的动作；
        - 指挥并控制CPU、数据cache和输入/输出设备之间数据流动的方向。

3. CPU中的主要寄存器（6类）

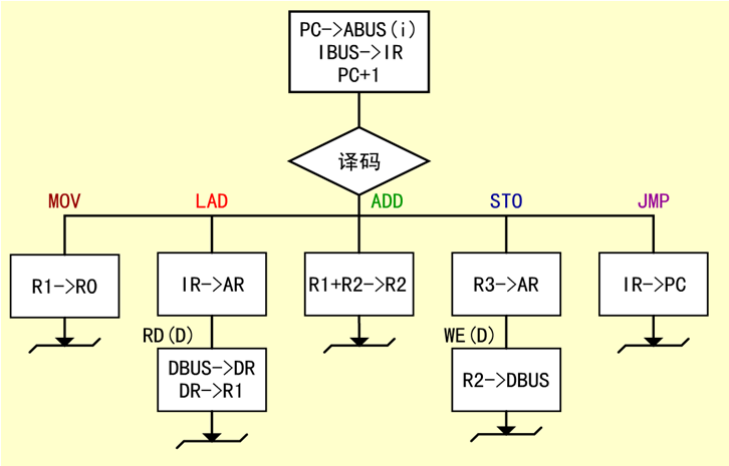
- **数据缓冲寄存器（DR）**用来暂时存放ALU的运算结果，或由数据存储器读出的一个数据字，或来自外部接口的一个数据字。
    - 作用
        - 作为ALU运算结果和通用寄存器之间信息传送中时间上的缓冲
        - 补偿CPU和内存、外围设备之间在操作速度上的差别
- **指令寄存器（IR）**指令寄存器用来保存当前正在执行的一条指令。
- **程序计数器（PC）**又称为**指令计数器**。由于大多数指令都是按顺序来执行的，所以修改的过程通常只是简单的对PC加1。但是当遇到转移指令如JMP指令时，那么后继指令的地址（即PC的内容）必须从指令寄存器中的地址字段取得。
- **数据地址寄存器（AR）** 用来保存当前CPU所访问的内存单元的地址。
- **通用寄存器（R0~R3）**当算术逻辑单元（ALU）执行算术或逻辑运算时，为ALU提供一个工作区。
- **状态字寄存器（PSW）**状态字寄存器保存由算术指令和逻辑指令运算或测试结果建立的各种条件代码，如运算结果进位标志C，运算结果溢出标志V，运算结果为零标志Z，运算结果为负标志N，等等。这些标志位通常分别由 1位触发器保存。

4.操作控制器

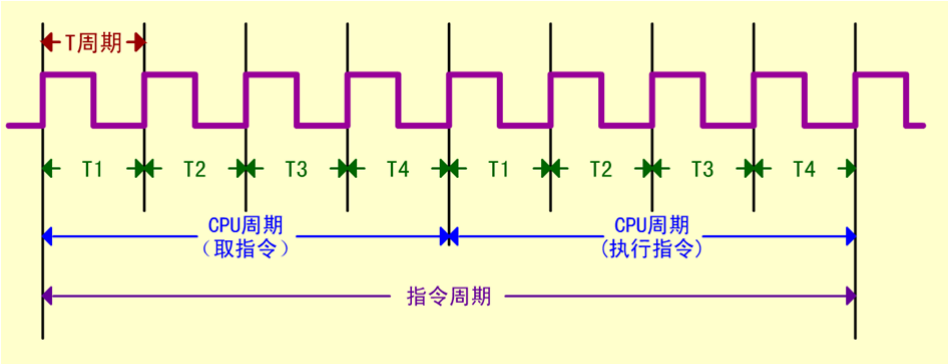
- 分类
    - **硬布线控制器**: 采用时序逻辑技术来实现
    - **微程序控制器**:采用存储逻辑（组合逻辑）来实现 重点介绍
    - 第三种是前两种方式的组合。

5.指令周期、机器周期、时钟周期

- **指令周期**：指取指令、分析指令到执行完该指令所需的全部时间。也可以说是CPU从内存取出一条指令并执行这条指令的时间总和。一个取值周期+一个以上的执行周期
    - 由于各种指令的操作功能不同，因此各种指令的指令周期是不尽相同的。
    - **方框图表示指令周期**
        - **方框** 代表一个CPU周期，方框中的内容表示数据通路的操作或某种控制操作。
        - **菱形** 通常用来表示某种判别或测试，不过时间上它依附于紧接它的前面一个方框的CPU周期，而不单独占用一个CPU周期。



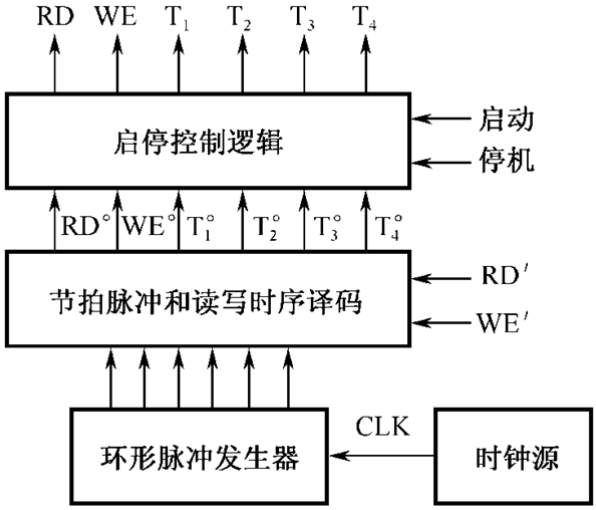
- **机器周期**通常又称**CPU周期**，通常把一条指令周期划分为若干个机器周期，每个机器周期完成一个基本操作。一个机器周期中，包含若干个**时钟周期**（节拍脉冲或T周期）。
- **时钟周期**：把一个机器周期分为若干个相等的时间段，每一个时间段称为一个节拍。



6.时序产生器

- 作用：CPU中的控制器用它指挥机器的工作。CPU可以用时序信号/周期信息来辨认从内存中取出的是指令（取指）还是数据（执行）。
- 体制： 组成计算机硬件的器件特性决定了时序信号的基本体制是电位—脉冲制
  - 硬布线控制器中，时序信号往往采用**主状态周期-节拍电位-节拍脉冲**三级体制。
  - 在微程序控制器中，时序信号比较简单，一般采用**节拍电位-节拍脉冲**二级体制。
- 组成

  微程序控制器中使用的时序信号产生器由时钟源、环形脉冲发生器、节拍脉冲读写时序译码逻辑、启停控制逻辑等部分组成。

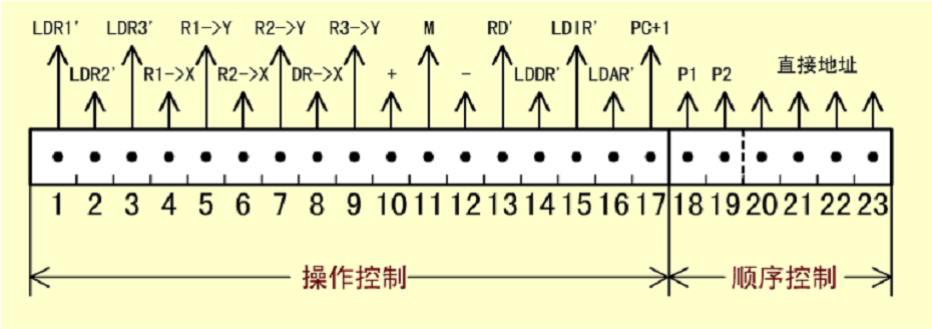


- **控制方式**

  同步控制、异步控制、联合控制三种方式
  - 同步控制
  - 异步控制
  - 联合控制

6.微命令、微操作、微指令、微程序

- 微命令：控制部件通过控制线向执行部件发出的各种控制命令。
- 微操作：执行部件接受微命令后所进行的操作。
  - 微操作在执行部件中是最基本的操作。微操作可分为相容的和互斥的两种：
    - **互斥的微操作**：是指不能同时或不能在同一个CPU周期内并行执行的微操作。
    - **相容的微操作**：是指能够同时或在同一个CPU周期并行执行的微操作。
- 微指令：在机器的一个CPU周期中，一组实现一定操作功能的微命令的组合。微指令存储在控制器中的控制存储器中。
  - 基本格式：微指令字长为23位，由**操作控制**和**顺序控制**两大部分组成。



- 操作控制字段，又称微操作码字段，用以产生某一步操作所需的各个微操作控制信号。某位为1，表明发微指令。
- 顺序控制字段，又称微地址码字段，用以控制产生下一条要执行的微指令地址。

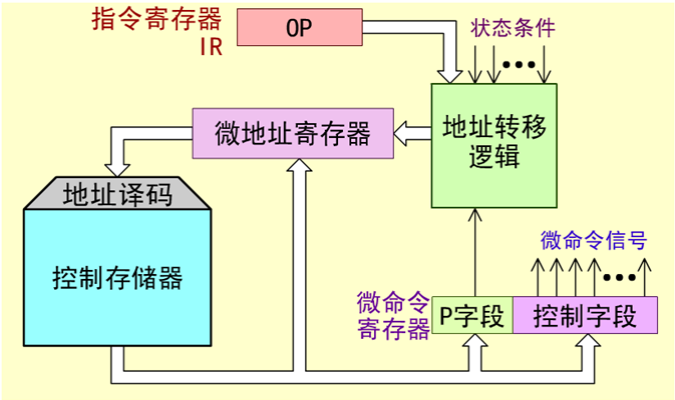
- 微程序：实现一条机器指令功能的许多条微指令组成的序列。

机器指令-》微程序-》微指令-》微命令-》微操作

7.微程序控制器组成

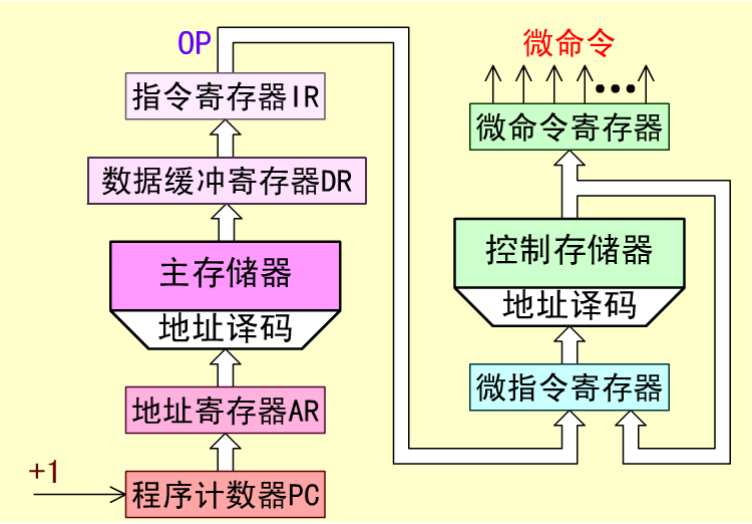
它主要由控制存储器、微指令寄存器和地址转移逻辑三大部分组成。

- 控制存储器(μCM)：控制存储器是微程序控制器的**核心部件**，用来存放实现全部指令系统的微程序，它是一种只读存储器。
- 微指令寄存器(μIR)：微指令寄存器用来存放由控制存储器读出的一条微指令信息。其中微地址寄存器决定将要访问的下一条微指令的地址，而微命令寄存器则保存一条微指令的操作控制字段和判别测试字段的信息。
- 地址转移逻辑



8.机器指令与微指令的关系

- 一条机器指令对应一个微程序，这个微程序由若干条微指令序列组成。因此，一条机器指令的功能是由若干条微指令组成的序列来实现的。
- 一条机器指令所完成的操作划分成若干条微指令来完成，由微指令进行解释和执行。
- 指令、程序和地址与**内存储器**有关；微指令、微程序和微地址与**控制存储器**有关。

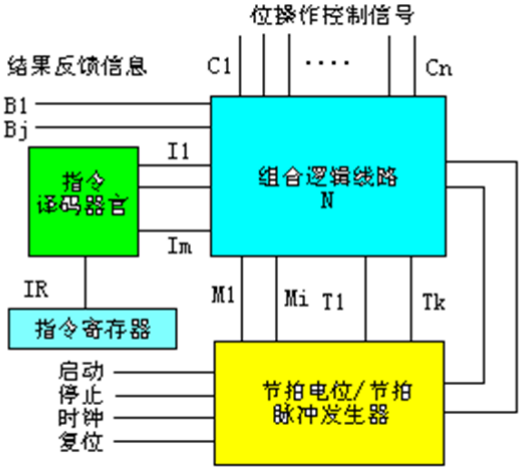


9.微程序设计技术

- **微命令编码**: 对微指令中的操作控制字段采用的表示方法。
  - 直接表示法: 操作控制字段中的每一位代表一个微命令。
  - 编码表示法：把一组相斥性的微命令信号组成一个小组，然后通过小组译码器对每一个微命令信号进行译码，译码输出作为操作控制信号。
  - 混合编码法：这种方法是把直接表示法与字段编码法混合使用，以便能综合考虑指令字长、灵活性、执行微程序速度等方面的要求。
- **微地址的形成方法**
  - 计数器方式
  - 多路转移方式：在多路转移方式中，当微程序不产生分支时，后继微地址直接由顺序控制字段给出；
- **微指令格式**
  - 水平型微指令：一次能定义并执行多个并行操作微命令的微指令，叫做水平型微指令。
  - 垂直型微指令：微指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能，称为垂直型微指令。
  - 水平型微指令和垂直型微指令的比较：
    - 水平型微指令并行操作能力强，效率高，灵活性强，垂直型微指令则较差。
    - 水平型微指令执行一条指令的时间短，垂直型微指令执行时间长。
    - 由水平型微指令解释指令的微程序，有微指令字较长而微程序短的特点。垂直型微指令则相反。
    - 水平型微指令用户难以掌握，而垂直型微指令与指令比较相似，相对来说，比较容易掌握。

10. 硬连线控制器

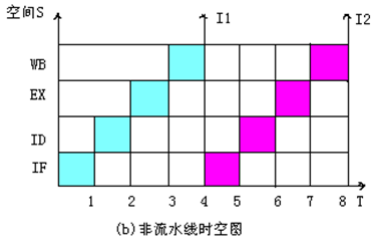
- 这种逻辑电路是一种由门电路和触发器构成的复杂树形逻辑网络，故称之为**硬连线控制器**。
- 组成
  - 输入信号来源：
    - 来自指令操码译码器的输出I_m
    - 来自执行部件的反馈信息B_j
    - 来自时序产生器的时序信号，包括节拍电位信号*M*和节拍脉冲信号*T*。
  - 输出信号： 就是微操作控制信号，它用来对执行部件进行控制。
  - 硬连线控制器的基本原理：$C=f(I_m,M_i,T_k,B_j)$



11. 流水CPU

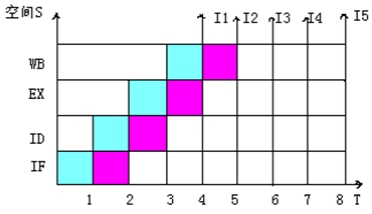
- **计算机的并行处理技术**（3种）
  - 时间并行：时间并行指时间重叠，在并行性概念中引入时间因素，让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。时间并行性概念的实现方式就是采用**流水处理部件。**
  - 空间并行：空间并行指资源重复，在并行性概念中引入空间因素，以“数量取胜” 为原则来大幅度提高计算机的处理速度。
  - 时间并行+空间并行：指时间重叠和资源重复的综合应用 ，既采用时间并行性又采用空间并行性。
- **流水CPU的结构**

  通常由三部分组成：**指令部件、指令队列、执行部件。**
- **流水CPU的时空图**
  - 非流水CPU



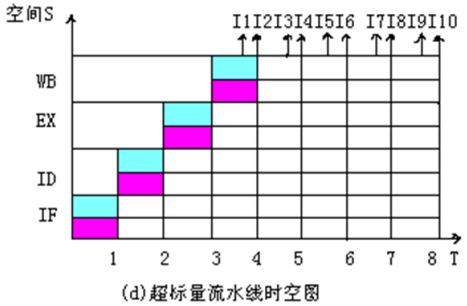
(b)非流水线时空图

  - 流水CPU



(c)标量流水线时空图

  - 超标量流水线



(d)超标量流水线时空图

    - 采用时间和空间并行技术
- **流水线分类**

    - **指令流水线**：指令步骤的并行。将指令流处理过程分为取指令、译码、执行、写回等几个并行处理过程段。目前几乎所有高性能计算机都采用指令流水线。
    - **算术流水线：**运算操作步骤的并行。如流水加法器、流水乘法器、流水除法器等。现代计算机中已广泛采用了流水的算术运算器。
    - **处理机流水线**：又称为宏流水线，是指程序步骤的并行。
- **流水线中的主要问题**
    - **资源相关**：资源相关是指多条指令进入流水线后在同一机器时钟周期内争用同一个功能部件所发生的冲突。
        - 解决方法
            - 产生冲突的指令停顿一拍后再启动。
            - 增设一个存储器，将指令和数据分别放在两个存储器中。
    - **数据相关**：在一个程序中，如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的。
        - RAW(Read After Write)：后面指令用到前面指令所写的数据。
        - WAW(Write After Write)：两条指令写同一个单元。在简单流水线中没有此类相关，因为不会乱序执行。
        - WAR(Write After Read)：后面指令覆盖前面指令所读的单元。在简单流水线中没有此类相关。
        - 解决方法
            - 可以推后后继指令对相关单元的读操作
            - 设置相关的直接通路
    - **控制相关**：控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能为顺序取下条指令；也可能转移到新的目标地址取指令，从而使流水线发生断流。
        - 解决方法：
            - 延迟转移法
            - 转移预测法

第六章：总线系统

1.定义

- 总线(BUS)是构成计算机系统的**互连机构**，是多个系统功能部件之间进行**数据传送的公共通路。**
- 借助于总线连接，计算机在各系统功能部件之间实现**地址**、**数据**和**控制信息**的交换。在**争用资源**的基础上进行工作。

2.总线的分类

- **内部总线**：CPU内连接各寄存器及运算器部件之间的总线。
- **系统总线**：计算机系统中CPU和其他高速功能部件（如存储器、通道等）相互连接的总线。
- **I/O总线**：中低速I/O设备相互连接的总线。

3.总线的特性

- **物理特性**：总线的**物理连接方式。**根数、插头、插座形状，引脚线的排列方式
- **功能特性**：描述总线中每一根线的功能。地址、数据、控制三类
- **电气特性**：定义每一根线上信号的传递方向及有效电平范围。单/双向，电平高有效/低有效及范围
- **时间特性**：规定了总线上各信号有效的时序关系，每根总线在什么时间有效。

3. 总线带宽

- 总线本身所能达到的**最高传输速率**
- 是衡量总线性能的重要指标，单位是MB/s或GB/S。

4. 总线的连接方式(2种)

- **单总线结构**：使用一条单一的系统总线来连接CPU、内存和I/O设备。
    - 单总线结构简单、容易扩充。
    - 多部件共用一根总线，分时工作,传输效率较低。同一时间只能允许**两个**设备之间传送数据，信息传送的效率和吞吐量受到限制。
    - 处理器结构变化对总线产生影响
- **多总线结构**：在CPU、主存、I/O之间互联采用多条总线。

5.信息传送的方式

- 串行传送
- 并行传送
- 分时传送
    - **总线复用方式**，某个传输线上既传送地址信息，又传送数据信息。l为此必须划分时间片，以便在不同的时间间隔中完成传送地址和传送数据的任务。
    - 共享总线的部件分时使用总线。

6.总线的仲裁

为了解决多个功能模块争用总线的问题，设置**总线仲裁**部件。一般采用**优先级**或**公平策略**进行仲裁。

按照总线仲裁电路的位置不同，仲裁方式分为集中式和分布式两种。

- **集中式仲裁**
    - 集中式仲裁中每个功能模块有两条线连到中央仲裁器：
        - 一条是送往仲裁器的总线请求信号线BR(Bus Request)
        - 一条是仲裁器送出的总线授权信号线BG(Bus grant)
    - 集中式仲裁采用的三种查询方式：
        - **链式查询方式（菊花链查询）**：链式查询是通过接口的优先级排队电路来实现
        - **计数器定时查询方式**
        - **独立请求方式**
- **分布式仲裁**
    - 不需要中央仲裁器，每个功能设备都有自己的仲裁号以及仲裁器。
    - 分布式仲裁是以优先级仲裁策略为基础。

7.总线数据传送模式（4种）

- **读、写操作**
- **块传送操作**：只需给出块的起始地址，然后对固定块长度的数据一个接一个地读出或写入。
- **写后读、读修改写操作**：这是两种组合操作。只给出地址一次（表示同一地址），或进行先写后读操作，或进行先读后写操作。
- **广播、广集操作**：一个主方对多个从方进行写操作，这种操作称为广播。与广播相反的操作称为广集，它将选定的多个从方数据在总线上完成AND或OR操作，用以检测多个中断源。

第七章：外围设备

1.磁盘阵列RAID

RAID称廉价冗余磁盘阵列，它的设计理念是用多个小容量磁盘代替一个大容量磁盘，并用分布数据的方法能够同时从多个磁盘中存取数据，因而改善了I/O性能，增加了存储容量，现已在超级或大型计算机中使用。

第八章：输入输出系统

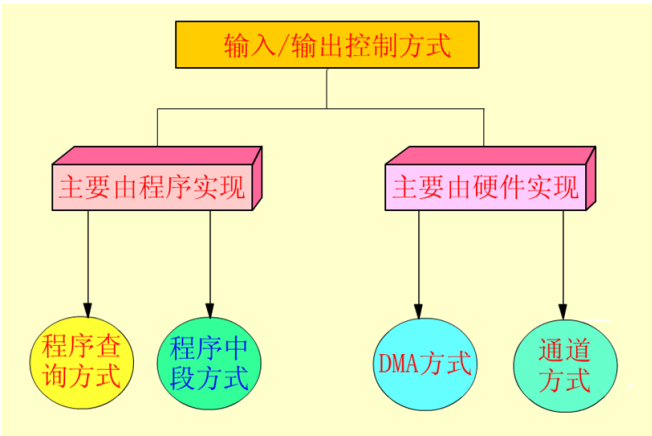
1.CPU管理外围设备的方式（4种）

- **程序查询方式**

  在这种方式中，数据在CPU和外围设备之间的传输完全靠计算机程序控制，是在CPU主动控制下进行的。

  - 优点：CPU的操作和外围设备的操作能够同步，而且硬件结构比较简单。
  - 缺点：由于外围设备动作很慢，程序进入查询循环时，CPU花很长时间等待外设完成，此时不能处理其他业务。因此当前除单片机外，很少使用程序查询方式。
- **程序中断方式**
- **直接内存访问(DMA)方式**
  - 是一种完全由硬件执行I/O交换的工作方式。
- 在该方式中DMA控制器从CPU完全接管对总线的控制权。数据交换不经过CPU而直接在主存和外围设备之间进行，以便高速传送数据。
  - DMA方式一般用于高速传送**成组数据**。
- 优点：
  - 数据传送速度很高
    - 传送速率仅受到内存访问时间的限制。
  - 与中断方式相比
    - 需要更多的硬件
    - 适用于内存和高速外围设备之间大批数据交换的场合。
- **通道方式**
  - 通道的出现进一步提高了CPU的效率。这是因为，CPU将部分权力下放给通道。这种提高CPU效率的办法是以花费更多硬件为代价的。
- 总结

  程序查询方式和程序中断方式适用于**数据传输率**比较低的外围设备，而DMA方式、和通道方式适用于数据传输率比较高的设备。

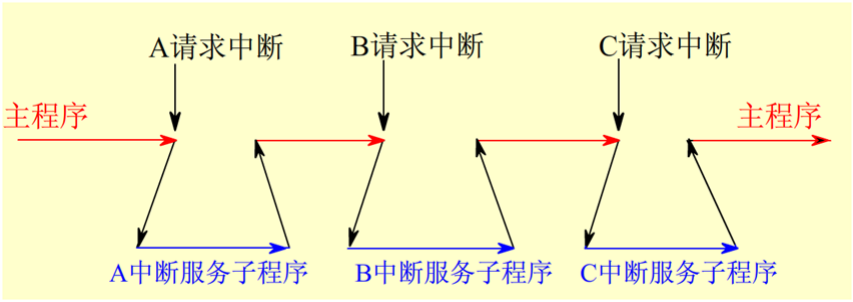


2.设备编址

- **统一编址**
  - I/O设备中的控制寄存器、数据寄存器、状态寄存器等和内存单元一样看待，联合在一起编排地址。
  - 用访问内存的指令去访问I/O设备，不需要专门的I/O指令。
- **独立编址**
  - 内存地址和I/O设备地址是分开的，不占用内存空间。
  - 访问I/O设备有专门的I/O指令组。

3. 中断

- 概念：中断是指CPU暂时中止现行程序，转去处理随机发生的紧急事件，处理完后自动返回原程序的功能和技术。

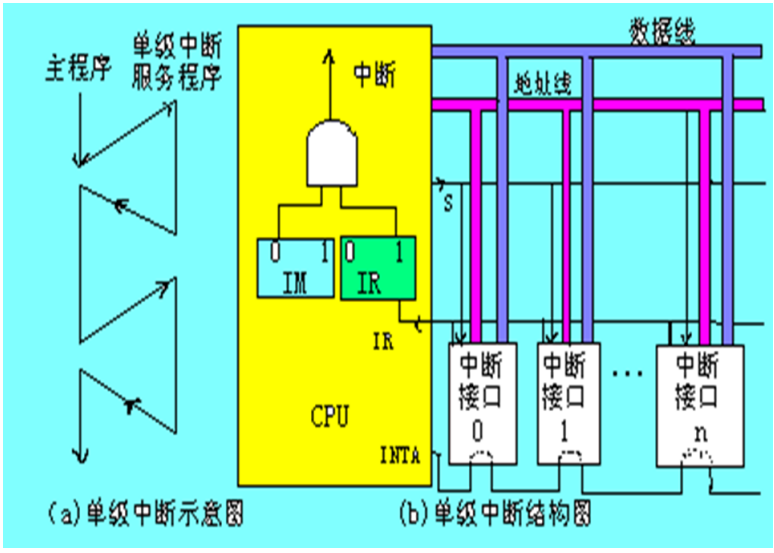


- 中断处理过程是由**硬件和软件**结合来完成的。
  - **中断周期**由硬件实现
  - **中断服务程序**由机器指令序列实现。
- **中断源**：能够向CPU发出中断请求的事件。
- **程序中断方式的基本接口**

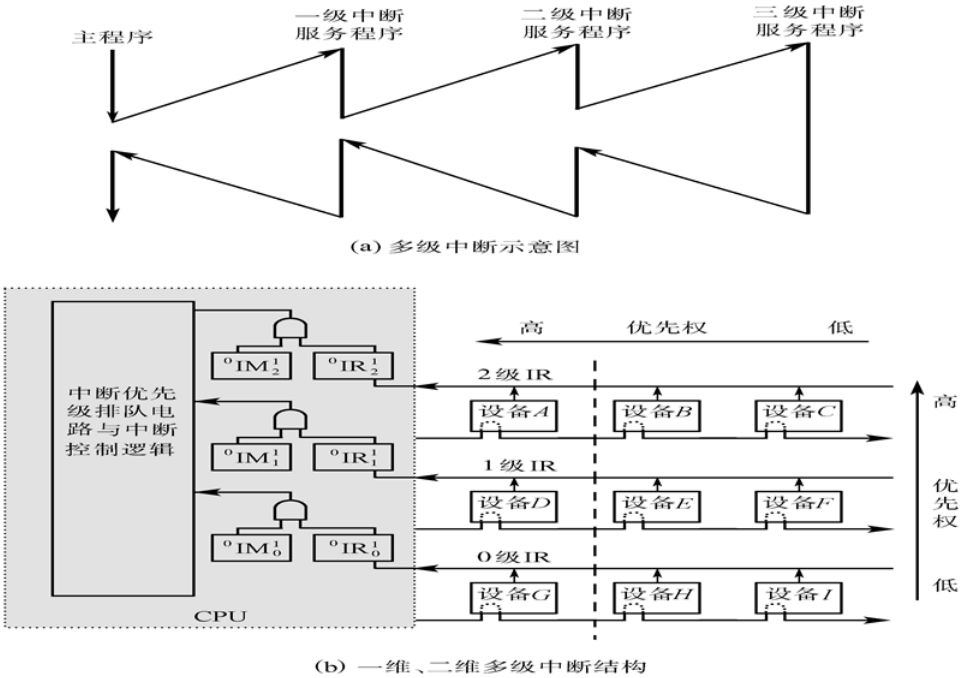
  程序中断由外设接口的状态和CPU两方面来控制：
  - 在接口方面，有决定是否向CPU发出中断请求的机构，主要是接口中的**“准备就绪”标志(RD)**和**“允许中断”标志(EI)**两个触发器；
    - **准备就绪的标志(RD)**：一旦设备做好一次数据的接收或发送，便发出一个设备动作完毕信号 ，使RD标志置“1”。在中断方式中，该标志用作为中断源触发器，简称中断触发器。
    - **允许中断触发器(EI)**：可以用程序指令来置位。EI为“1”时，某设备可以向CPU发出中断请求；EI为“0”时，不能向CPU发出中断请求，这意味着某中断源的中断请求被禁止。设置EI标志的目的，就是通过软件来控制是否允许某设备发出中断请求。
  - 在CPU方面，有决定是否受理中断请求的机构，主要是**“中断请求”标志(IR)**和**“中断屏蔽”标志(IM)**两个触发器。
    - **中断请求触发器(IR)**：它暂存中断请求线上由设备发出的中断请求信号。当IR标志为“1”时，表示设备发出了中断请求。
    - **中断屏蔽触发器(IM)**：**是CPU是否受理中断或批准中断的标志。IM标志为“0”时，CPU可以受理外界的中断请求，反之，IM标志为“1”时，CPU不受理外界的中断。

4.单级中断与多级中断

- 单级中断
  - 在单级中断系统中，所有中断源属于同一级，离CPU越近，优先级越高
  - 不允许其他中断源再打断中断服务程序，即使优先权比它高的中断源也不能再打断。



- 多级中断
  - **多级中断系统**是指计算机系统中有相当多的中断源，根据各中断事件的轻重缓急程度不同而分成若干级别，每一中断级分配给一个优先权。
  - 优先权高的中断级可以打断优先权低的中断服务程序，以程序嵌套方式工作。多级中断可以嵌套，但**同一级的中断不允许嵌套**
  - 多级中断可分为一维多级中断和二维多级中断
    - **一维多级中断**是指每一级中断里只有一个中断源
    - **二维多级中断**是指每一级中断里又有多个中断源。
  - 一个系统有n级中断，则CPU中有n个IR，n个IM，某级中断被响应后，则**关闭本级和低于本级**的IM，开放更高级的IM。



5. DMA控制器与CPU分时使用内存通常采用以下三种方法：

- 停止CPU访问内存
- 周期挪用
- DMA与CPU交替访内存

6. 通道的类型

- **选择通道**

  选择通道又称高速通道，在物理上它可以连接多个设备，但是这些设备不能同时工作，在某一段时间内通道只能选择一个设备进行工作。

- **多路通道**

  多路通道又称为多路转换通道，在同一时间能处理多个I/O设备的数据传输。它又分为数组多路通道和字节多路通道。

第九章:操作系统支持

1.虚拟存储器概念

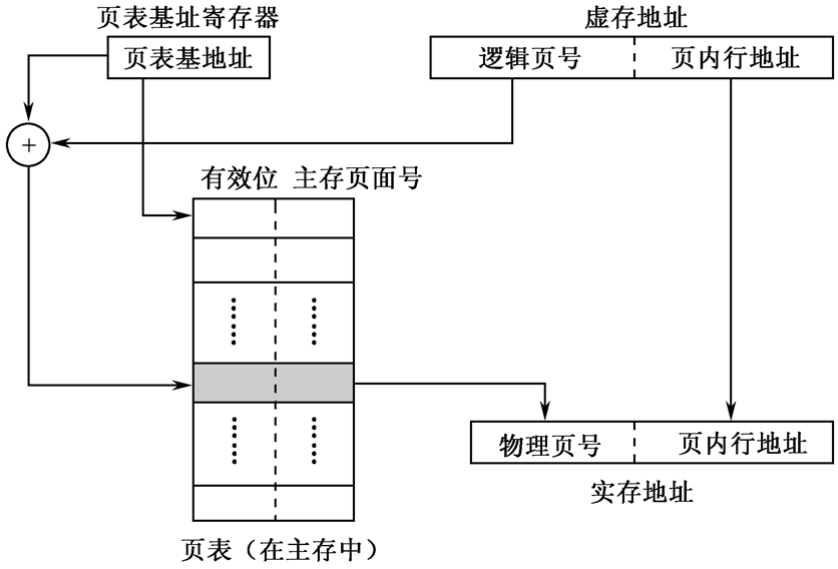
- 虚拟存储器只是一个容量非常大的存储器的**逻辑模型**，不是任何实际的物理存储器。
- 逻辑地址：用户编制程序时使用的地址称为虚地址或逻辑地址
- 物理地址：而计算机物理内存的访问地址则称为实地址或物理地址，其对应的存储空间称为物理存储空间或主存空间。
- 再定位：程序进行虚地址到实地址转换的过程称为程序的再定位。

2. cache-主存和主存-辅存这两个存储层次相同点和不同点

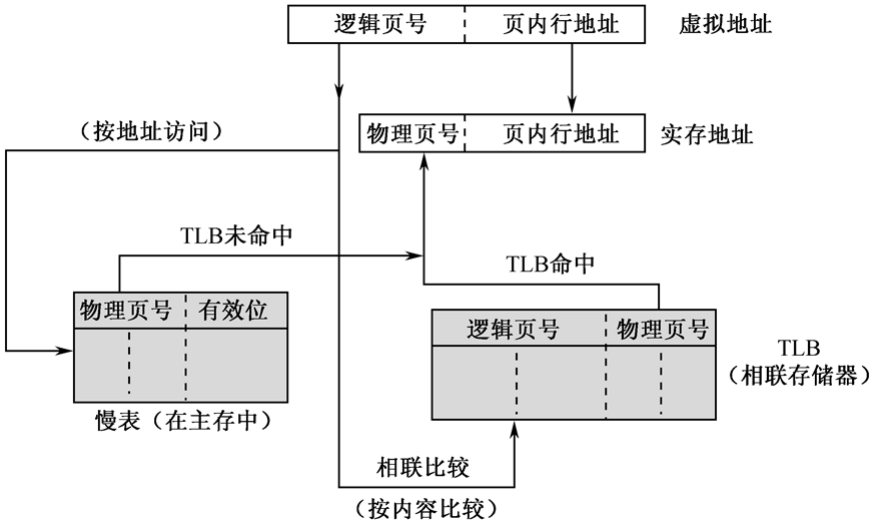
- 相同点：
  - **出发点相同**：二者都是为了提高存储系统的性能价格比而构造的分层存储体系，都力图使存储系统的性能接近高速存储器，而价格和容量接近低速存储器。
  - **原理相同**：都是利用了程序运行时的局部性原理把最近常用的信息块从相对慢速而大容量的存储器调入相对高速而小容量的存储器。
- 不同点：
  - **侧重点不同**：cache主要解决主存与CPU的速度差异问题；虚存主要是解决存储容量问题，另外还包括存储管理、主存分配和存储保护等方面。
  - **数据通路不同**：CPU与cache和主存之间均有直接访问通路，cache不命中时可直接访问主存；而虚存所依赖的辅存与CPU之间不存在直接的数据通路，当主存不命中时只能通过调页解决，CPU最终还是要访问主存。
  - **透明性不同**：cache的管理完全由硬件完成，对系统程序员和应用程序员均透明；而虚存管理由软件和硬件共同完成，由于软件的介入，虚存对实现存储管理的系统程序员不透明，而只对应用程序员透明。
  - **未命中时的损失不同**：由于主存的存取速度通常比辅存的存取速度快上千倍，故主存未命中时系统的性能损失要远大于cache未命中时的损失。

3. 页式虚存地址映射

- 页式虚拟存储系统中，虚地址空间被分成等长大小的页，称为逻辑页；主存空间也被分成同样大小的页，称为物理页。相应地，虚地址分为两个字段：高字段为逻辑页号，低字段为页内地址（偏移量）；实存地址也分两个字段：高字段为物理页号，低字段为页内地址。通过页表可以把虚地址（逻辑地址）转换成物理地址。
- 在大多数系统中，每个进程对应一个页表。页表本身则放在主存中。
- 地址映射



- 为了避免页表保存或调入主存储器时对主存访问次数的增多，可以对页表本身实行二级缓存，把页表中的最活跃的部分存放在高速存储器中，组成**快表**。这个专用于页表缓存的高速存储部件通常称为转换后援缓冲器(TLB)。保存在主存中的完整页表则称为**慢表**。TLB的作用与Cache作用类似，通常由相联存储器实现，存储慢表中部分信息的副本。

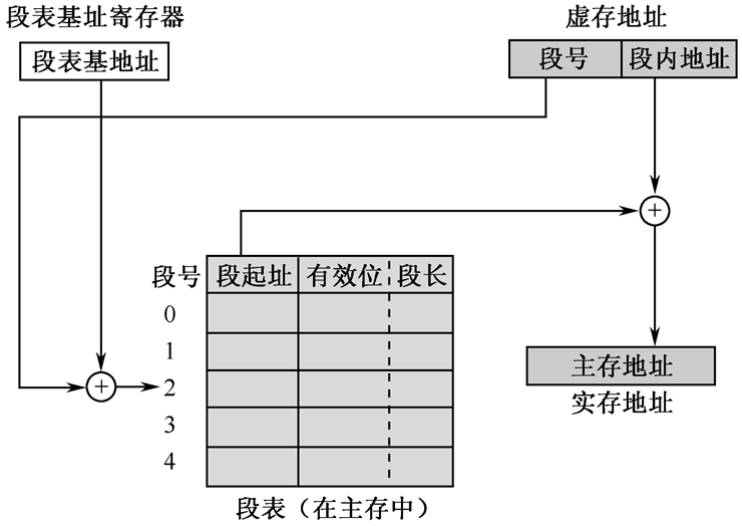


- **页式虚拟存储器的特点**
  - 优点
    - 页长固定、便于构造页表、易于管理。
    - 不存在外碎片，空间利用率高。
  - 缺点：

    页长与程序的逻辑大小不相关，不利于编程时的独立性，并给存储共享、存储保护等操作造成麻烦。

4. 段式虚拟存储器

- 段是按照程序的自然分界(逻辑结构)划分的长度可以动态改变的区域。
- 在段式虚拟存储系统中，虚地址由段号和段内地址（偏移量）组成。虚地址到实主存地址的变换通过段表实现。每个程序设置一个段表，段表的每一个表项对应一个段。每个表项至少包含下面三个字段：有效位、段起址、段长。
- 段表本身也是一个段，一般是驻留在主存中。
- 段式虚地址向实存地址的变换过程见图：



- 段式虚拟存储器特点
  - 优点：
    - 段的逻辑独立性使其易于编译、管理、修改和保护，也便于多道程序共享。
    - 段长可以根据需要动态改变。
  - 段式虚拟存储器也有一些缺点：
    - 因为段的长度不固定，主存空间分配比较麻烦。
    - 容易在段间留下许多外碎片，造成存储空间利用率降低。
    - 必须用加法操作通过段起址与段内偏移量的求和运算求得物理地址。因此，段式存储管理比页式存储管理方式需要更多的硬件支持。

5. 段页式虚拟存储器

6.虚存的替换算法

虚拟存储器的替换算法与cache的替换算法类似，有FIFO算法、LRU算法、LFU算法等。