# 计算机组成

## 历史

1. 电子管计算机(1946-1958),使用电子管作为基本元器件，使用汇编语言和机器语言编程，因为美国军方需要计算炮弹弹道而产生的。
2. 晶体管计算机（1958-1964），使用晶体管作为基本元器件。
3. 中小规模集成电路（1964-1971），使用集成电路作为基本元器件，主存使用半导体，计算机语言与操作系统得到发展。
4. 第四代：（1971至今），使用超大集成电路作为基本元器件。

## 现代计算机硬件的构成思想(冯诺依曼思想)

1. 运算器(ALU)：也称为算术逻辑单元ALU（Arithmetic Logic Unit）。一般计算机采用的二进制运算器，也有十进制运算器，能完成二进制和十进制运算。

功能：完成算术运算和逻辑运算。算术运算是指加、减、乘、除及它们的复合运算。

逻辑运算是指“与”、“或”、“非”等逻辑比较和逻辑判断等。任何复杂运算都转化为基本的算术与逻辑运算，然后在运算器中完成。CPU组成之一。

组成：

①算术逻辑单元：

②累加器：

③状态寄存器：

④通用寄存器：

1. 控制器(CU)：Controller Unit。

功能：从内存中取出指令和执行指令。

组成：

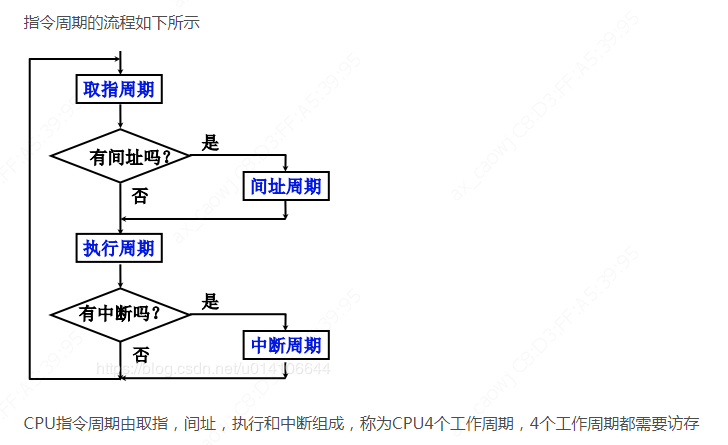
①指令寄存器(IR)：保存当前执行或即将执行的指令。

②程序计数器(PC)：寄存器，保存当前正在执行的指令的地址，在每个指令被获取后，程序计数器执行顺序中的下一个指令地址。 程序计数器作用是为了保证程序能够连续执行。

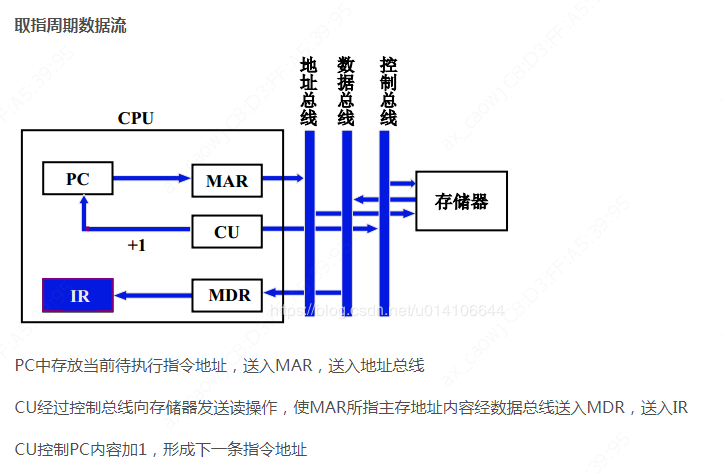
③操作控制器(OC)：

1. 主存储器(内存)：
2. 输入设备：
3. 输出设备：

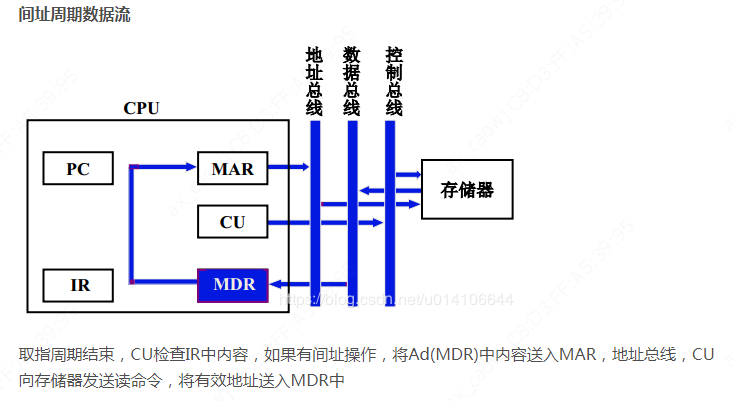
## 指令周期



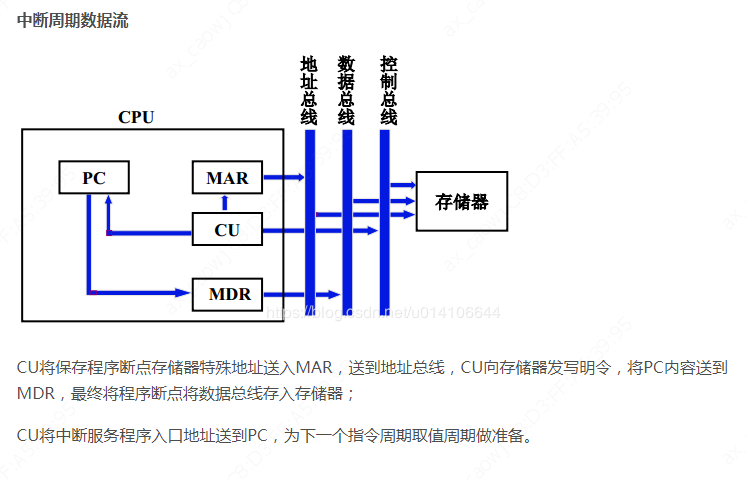
1. 取指令阶段：将一条指令从主存中取到指令寄存器的过程。



1. 指令译码阶段：对取回的指令进行拆分和解释。



1. 指令执行阶段：完成指令所规定的各种操作，具体实现指令的功能(例如根据指令地址码，得到操作数在主存中的地址，并从主存中读取该操作数用于运算，并将结果写入寄存器或主存中)。如果要求完成一个加法运算，算术逻辑单元ALU将被连接到一组输入和一组输出，输入端提供需要相加的数值，输出端将含有最后的运算结果。
2. 循环阶段：检测有无溢出、溢出、中断等信息，若没有则进行下一个指令周期。



## 硬件相关

### DMA

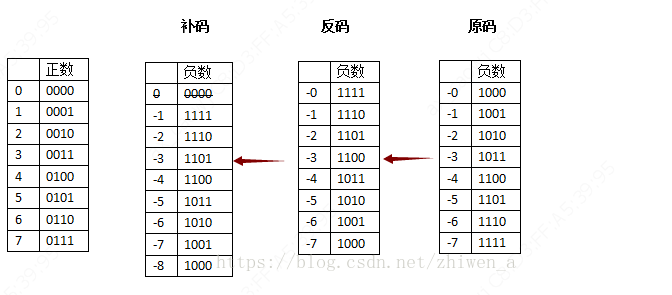
Direct Memory Access：直接内存存取。是指外部设备不通过CPU而直接与系统内存交换数据的接口技术。

DMA功能：通常系统总线是由CPU管理的，在ＤＭＡ方式时，就希望CPU把这些总线让出来，即CPU连到这些总线上的线处于第三态(高阻状态)，而由DMA控制器接管，控制传送的字节数，判断DMA是否结束，以及发出DMA结束信号，而在结束DMA传输后，DMA控制器应立即把总线控制权再交回给CPU。

DMA作用：减轻CPU负担， 提升数据读取速度。

# 信息的表示及处理

## 整数的表示及处理



### 原码

原码若是负数，最高位为1，反之为0。例如：0100表示4，1100表示-4。

由于CPU只有加法器没有减法器，那么一个数与它的相反数相加不会为0，因此不使用原码对数字进行计算。例如0010(2) + 1010(-2) = 1100(-4)。

### 补码

为了解决一个数与它的相反数相加不会为0的问题。计算机使用补码相加解决减法问题。

计算：正数的补码等于原码，负数的补码等于反码+1。例如0110（6）-0010（2）=0110（6）+1110（14)=10100（20=16+4），最高位的1，计算机将它放入psw寄存器。

最高位负数是1而不是0的原因：把1110（14）的最高位看作符号位后就是（-2）的补码。

### 反码

为了解决一个数与它的相反数相加不会为0的问题。

计算：正数的反码等于原码， 负数反码等于除符号位按位取反。

缺点：无法区分0000（+0）与1111（-0）的问题。

## 浮点数的表示及处理

采用IEEE浮点表示法

## 字符串的表示及处理

### Unicode字符集

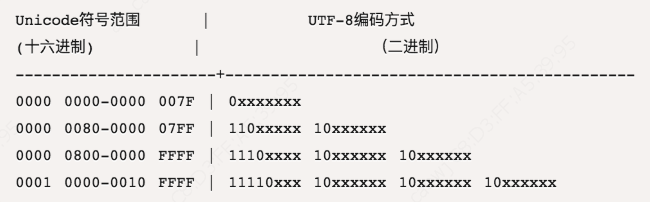
Unicode是一种字符集， 并未规定其二进制如何存储。

UTF(Unicode Transformation Format)Unicode转换格式，是规定Unicode如何存储的实现方式。

UTF-8表示使用1~6个字节不定长度大小来存储。

UTF-16表示使用2个或4个字节长度来存储。

UTF-32表示使用4个字节定长来存储。



以汉字严为例，演示如何实现 UTF-8 编码。

严的 Unicode 是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（0000 0800 - 0000 FFFF），因此严的 UTF-8 编码需要三个字节，即格式是1110xxxx 10xxxxxx 10xxxxxx。然后，从严的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，严的 UTF-8 编码是11100100 10111000 10100101，转换成十六进制就是E4B8A5

|  |  |  |  |
| --- | --- | --- | --- |
| 例子 | 字符集 | 编码方式 | 存储方式 |
| 严(E4B8A5) | Unicode | UTF-8,根据Unicode字符集进行转换。 | EF BB BF E4 B8 A5;(大端存储)EF BB BF表示使用UTF-8编码, E4 B8 A5为真正编码。 |
| 严(E4B8A5) | Unicode | UTF-8无BOM编码 | E4 B8 A5;(大端存储)E4 B8 A5为真正编码。无需EF BB BF表示编码方式。 |
| 严(4E25) | Unicode | Unicode  (默认USC-2 little endian) 使用2个字节存储,无需转换。 | FF FE 25 4E; FF FE表示小端存储类型，25 4E为真正编码。 |
| 严(4E25) | Unicode | Unicode big endian  (USC-2 big endian ) 使用2个字节存储，无需转换。 | FE FF 4E 25 ; FE FF表示大端存储类型，4E 25为真正编码。 |

ANSI是默认的编码方式。对于英文文件是ASCII编码，对于简体中文文件是GB2312编码（只针对 Windows 简体中文版，如果是繁体中文版会采用 Big5 码。

# 计算机术语

## 局部性

在现代计算机系统的各个层次，从硬件到操作系统、应用程序等，设计上都利用了局部性原理来提高访问性能。比如缓存机制，CPU指令顺序处理等。

局部性通常有两种形式：时间局部性和空间局部性。

空间局部性：一旦程序访问了某个存储单元，则不久之后其附近的存储单元也将被访问。通过一个地址能获取多个值，则具有良好的空间局部性（数组、链表）。

时间局部性：程序中的某条指令一旦执行，则不久之后该指令可能再次被执行。重复引用同一个变量时，则具有良好的时间局部性(循环)。