第二章微处理器的系统结构

CPU的功能和组成

微处理器可以分为：**通用型微处理器**和**嵌入式微处理器**(Micro-processor Unit，MPU)。通用型微处理器指的是通用计算机中的CPU，嵌入式微处理器指单片机、ARM、嵌入式DSP处理器（Digital Signal Processor）等。

相比通用型微处理器指令形式为复杂指令集（CISC），嵌入式微处理器的指令为嵌入式精简指令集（RISC）。

CPU的主要性能参数：

1. CPU的主频，即CPU内核工作的时钟频率（CPU Clock Speed）。时钟频率速度是指同步电路中时钟的基础频率，它以“若干次周期每秒”来度量，量度单位采用SI单位赫兹（Hz）。
2. 外频，是CPU外部的工作频率，是由主板提供的基准时钟频率。
3. FSB频率，是连接CPU和主板芯片组中的北桥芯片的前端总线（Front Side Bus）上的数据传输频率。
4. CPU的主频和外频间存在这样的关系：主频=外频×倍频

CPU的功能

1. 指令顺序控制
   控制 程序中指令的执行顺序。程序中的各指令之间是有严格顺序的，必须严格按程序规定的顺序执行，才能保证计算机工作的正确性。因此，保证系统按照顺序执行程序是CPU的首要任务。
2. 操作控制
   一条指令的功能往往是由计算机中的部件执行一序列的操作来实现的。CPU要根据指令的功能，产生相应的操作控制信号，发给相应的部件，从而控制这些部件按指令的要求进行动作。
3. 时间控制
   时间控制就是对各种操作实施时间上的定时。在一条指令的执行过程中，在什么时间做什么操作均应受到严格的控制。另一方面，一条指令的整个执行过程也要受到时间的严格控制。只有这样，计算机才能有条不紊地自动工作。
4. 数据加工
   即对数据进行算术运算和逻辑运算处理，完成数据的加工处理，这是CPU的根本任务。因为原始信息只有经过加工处理后才能对人们有用。

CPU内核结构

（一） 运算器
【运算单元】

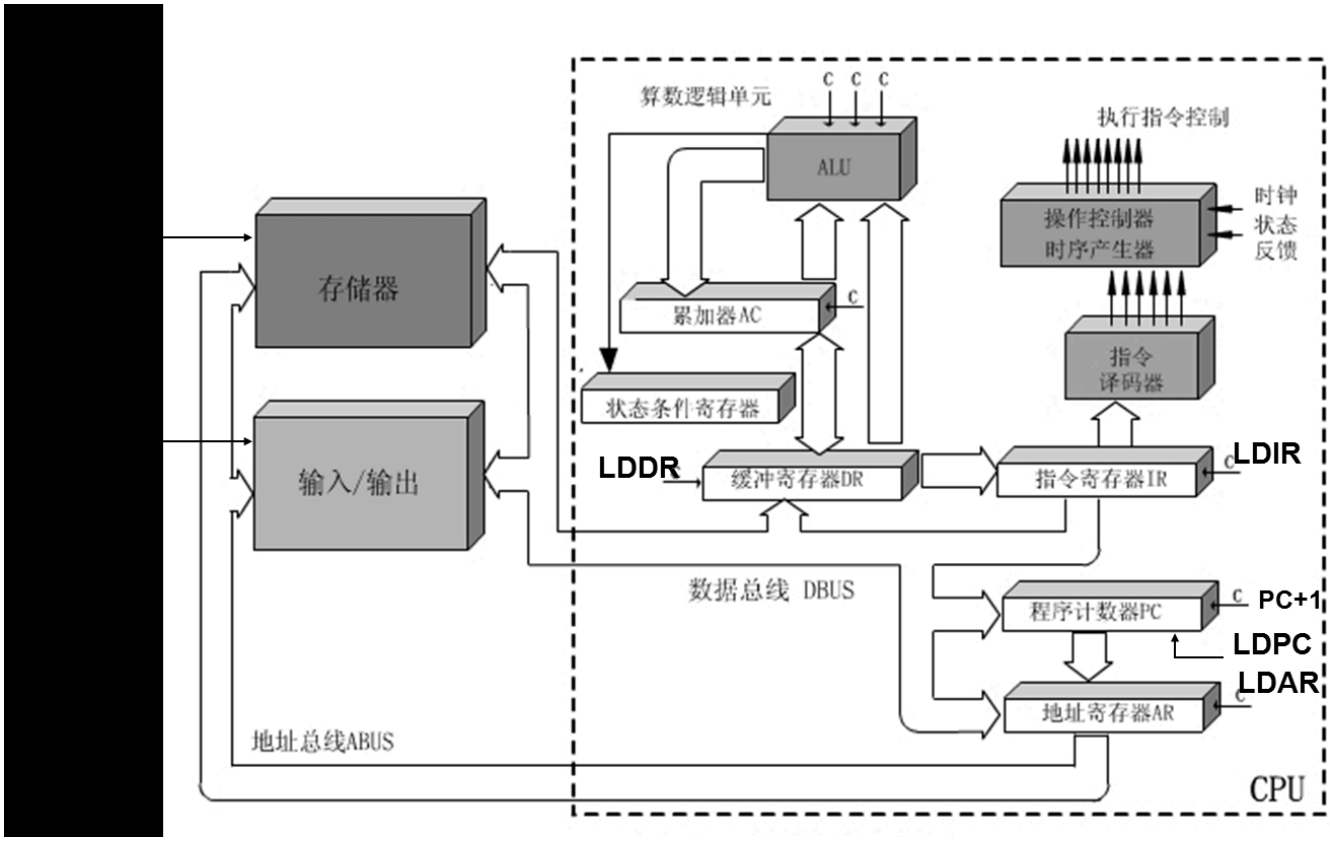
- 算术逻辑运算单元ALU（Arithmetic and Logic Unit）
  ALU主要完成对二进制数据的定点算术运算（加减乘除）逻辑运算（与或非异或）以及移位操作
- 浮点运算单元FPU（Floating Point Unit）
  FPU主要负责浮点运算和高精度整数运算。有些FPU还具有向量运算的功能，另外一些则有专门的向量处理单元

【寄存器组】
通用寄存器组和专用寄存器

（二） 控制器
运算器只能完成运算，而控制器用于控制着整个CPU的工作

1. 指令控制器
2. 时序控制器
3. 总线控制器
4. 中断控制器

CPU的基本组成



1. 运算部件
   运算部件又叫算术逻辑单元（ALU），它可以对数据进行最基本的算术和逻辑运算，如加、减、乘、除、与、或、异或等。
2. 寄存器组
   CPU内部有多个寄存器。寄存器就是用于暂存信息的小型存储器，它们按功能的区别分为：（1）指令寄存器（IR）；（2）地址寄存器（AR）；（3）数据寄存器（DR）；（4）累加寄存器（AC）；（5）状态条件寄存器。
3. 程序计数器（PC）
   程序计数器也叫指令计数器，它实际上也是一个寄存器，**它总是指出下一条要执行的指令在存储器中的地址。**在顺序执行指令的情况下，当它把一条指令的地址码送到地址总线后，程序计数器的内容就自动调整，这条指令是几个字节，它就加几，从而又指向下一条要执行的指令地址。如果遇到跳转指令，就会把新的地址码置入程序计数器，从而改变指令执行的顺序。
4. 指令译码器（ID）
   指令译码器对指令进行译码，并控制时序逻辑电路。
5. 控制器电路（C）
   控制器电路根据译码器的分析，产生执行这条指令所需要的全部时序和控制信号，送到CPU内部和外部各部件进行控制。
   目前，新型的CPU都增加了高速缓存器（Cache），高速缓存单元的主要功能是快速进行指令或数据存储，在CPU内部开辟一个高速缓存空间，这样指令和数据可以暂时存放在CPU内部的高速缓存中，减少了指令在CPU和内存之间的传输次数

（6.）高速缓存器

CPU缓存（Cache Memory）位于CPU与内存之间的临时存储器，它的容量比内存小但交换速度快。在缓存中的数据是内存中的一小部分，但这一小部分是短时间内CPU即将访问的，当CPU调用大量数据时，就可避开内存直接从缓存中调用，从而加快读取速度。这样整个内存储器（缓存+内存）就变成了既有缓存的高速度，又有内存的大容量缓存对CPU的性能影响很大，主要是因为CPU的数据交换顺序和CPU与缓存间的带宽引起的。
缓存的工作原理是当CPU要读取一个数据时，首先从缓存中查找，如果找到就立即读取并送给CPU处理；如果没有找到，就用相对慢的速度从内存中读取并送给CPU处理，同时把这个数据所在的**数据块**调入缓存中，以后对整块数据的读取都从缓存中进行，不必再调用内存。

CPU的主要寄存器

1. **指令寄存器（IR）**
   指令寄存器用来用于存放当前正在执行的指令码。当前指令执行完了，下一条指令才能存入，否则一直保持着。当执行一条指令时，先把它从内存取到缓冲寄存器中，然后再传送至指令寄存器。指令划分为操作码和地址码字段，由二进制数字组成。为了执行任何给定的指令，必须对操作码进行测试，以便识别所要求的操作，一个叫做“指令译码器”的部件就是做这项工作的。指令寄存器中操作码字段的输出就是指令译码器的输入，操作码一经译码后，即可向操作控制器发出具体操作的特定信号。

2. **地址寄存器（AR）**
   地址寄存器用来暂存正在执行的指令在存储单元中的地址，或I/O接口的地址。由于在内存和CPU之间存在着操作速度上的差别，所以必须使用地址寄存器来保持地址信息，直到内存的读/写操作完成为止。
   当CPU和内存进行信息交换，要使用地址寄存器和缓冲寄存器。同样，如果我们把外围设备的设备地址作为像内存的地址单元那样来看待，那么，当CPU和外围设备交换信息时，同样需要使用地址寄存器和缓冲寄存器。

3. 数据寄存器（DR）
   数据寄存器用于暂存通过数据总线从存储器中取出的指令或操作数，也可以暂存准备往存储器中存储的数据，数据寄存器的作用是：
   作为CPU和内存、外部设备之间信息传送的中转站；
   补偿CPU和内存、外围设备之间在操作速度上的差别；
   在单累加器结构的运算器中，数据存储器还可兼作为操作数寄存器

4. 累加寄存器（AC）
   当运算器的算术逻辑单元（ALU）执行全部算术和逻辑运算时，为ALU提供一个工作区。例如，在执行一个加法运算前，先将一个操作数暂时存放在AC中，再从内存中取出另一个操作数，然后同AC的内容相加，所得结果送回AC中，而AC中原有的内容随即被破坏。所以，顾名思义，累加寄存器是暂时存放ALU运算的结果信息。显然，**运算器中至少要有一个累加寄存器。**
   由于运算器的结构不同，可采用多个累加寄存器，如有些计算机中有2个，4个，8个，甚至更多。当使用多个累加器时，就变成通用寄存器结构，其中任何一个可存放源操作数，也可存放结果操作数。

5. 状态条件寄存器
   状态条件寄存器用来寄存CPU执行完上一条指令后，处理结果的某些特征（或者状态），例如运算结果连位标志（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。除此之外，状态条件寄存器还保存中断和系统工作状态等信息，以便使CPU和系统能及时了解机器运行状态和程序运行状态。因此，状态条件寄存器是一个由各种状态条件标志拼凑而成的寄存器。

6. 程序计数器（PC）
   程序计数器又称指令计数器，它是指出下一条要执行的指令在存储器中的地址。在程序开始执行前，必须将

它的起始地址，即程序的第一条指令所在的内存单元地址送入PC，因此PC的内容即是从内存提取的第一条指令的地址。当执行指令时，CPU将自动修改PC的内容，以便使其保持的总是将要执行的下一条指令的地址。由于大多数指令都是按顺序来执行的，所以修改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**后继指令地址（即PC的内容）必须从指令寄存器中的地址字段取得**。在这种情况下，下一条从内存取出的指令将有转移指令来规定，而不是像通常一样按顺序来取得。

操作控制器和时序产生器

通常把许多寄存器之间传送信息的通路，称为 “数据通路”，信息从什么地方开始，中间经过哪个寄存器或多路开关，最后传送到哪个寄存器，都要加以控制。在各寄存器之间建立数据通路的任务，是由称为“操作控制器”的部件来完成的，**操作控制器的功能，就是根据指令操作码和时序信号，产生各种操作控制信号，以便正确地建立数据通路，从而完成取指令和执行指令的控制。**
根据设计方法不同，操作控制器可分为组合逻辑型，存储逻辑型，组合逻辑与存储逻辑结合型三种。第一种称为硬布线控制器，它是采用组合逻辑技术来实现的；第二种称为微程序控制器，它是采用存储逻辑来实现的；第三种称为可编程控制器，它是吸收前两种的设计思想来实现的。
时序产生器的作用，就是对各种操作实施时间上的控制。

硬布线控制器和微程序控制器
微程序控制器的控制功能是在**存放微程序存储器和存放当前正在执行的微指令的寄存器直接控制下实现的，**而硬布线控制的功能则由**逻辑门组合实现**。微程序控制器的电路比较规整，各条指令信号的差别集中在**控制存储器内容**上因此，无论是增加或修改指令都只要增加或修改控制存储器内容即可，若控制存储器是ROM，则要更换芯片，在设计阶段可以先用RAM或EPROM来实现，验证正确后或成批生产时，再用ROM代替。

硬布线控制器的控制信号**先用逻辑式列出，经化简后用电路来实现**，因此，显得零乱复杂，当需要修改指令或增加指令时就必须重新设计电路，非常麻烦而且有时甚至无法改变。因此，微操作控制取代了硬布线控制并得到了广泛应用，尤其是**指令复杂的计算机，一般都采用微程序来实现控制功能。**（例如X86的arm架构）

CPU的分类

初期的CPU （冯·诺伊曼计算机模型 ）
一、CPU控制程序执行过程
系统内存用于存放程序和数据。程序由一系列指令组成，这些指令是有序存放的，指令号表明了它的执行顺序。什么时候执行哪一条指令由CPU中的控制单元决定。数据表示用户需要处理的信息，它包括用户的具体数据和这个数据在内存系统中的地址。
二、CPU指令执行流程

- 流水线CPU

  由摩尔定律的预测，越来越大规模的集成电路问世，原来那些由于过于复杂而实现不了的微处理器架构可能得以实现。**晶体管数量的增加允许微处理器架构可以具有更多的资源来实现高性能的处理器。**一个早先的设计可以只拥有一个加法器，而后来的设计则可能拥有两个加法器，这就使得更多的操作可以并行执行。不可避免地，为了充分利用更多指令之间的并行执行，又存在很多需要改进的方面。在并行执行更多任务的方法中，最成功的方法就是采用流水线技术。

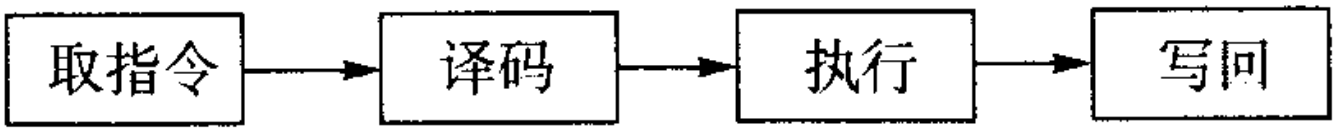
  处理器架构定义了软件的运行方式，其中就期望在程序执行过程中每一个时钟周期执行一条指令。在程序中，很多指令可以并行执行或者至少重叠执行。微处理器架构可以利用这些优势来提高处理器性能，但是为了保证软件兼容性，又要维持指令的顺序执行。通过允许不同指令的重叠执行，流水线可以提供更高的处理器性能。
  流水线的实质是通过在同一时间做多件事情来提高机器的性能，因此**指令流水线是一种可以将多条指令的执行过程相互重叠的实现技巧**，目前它是提高处理器处理速度的关键技术之一。

- 多媒体CPU

  随着微软Windows系统的普遍采用，音、视频信号在个人电脑应用中已非常普及，需要提高CPU处理多媒体数据的效率，加速电脑的多媒体应用，Pentium MMX CPU是英特尔在Pentium内核基础上改进的，其最大的特点是增加了57条MMX扩展指令集。这些指令专门用来处理音视频相关的计算。大部分多媒体应用程序需要 SIMD（Singl Instruction Stream Multipie Data Stream，单指令流多数据流）类型的体系结构

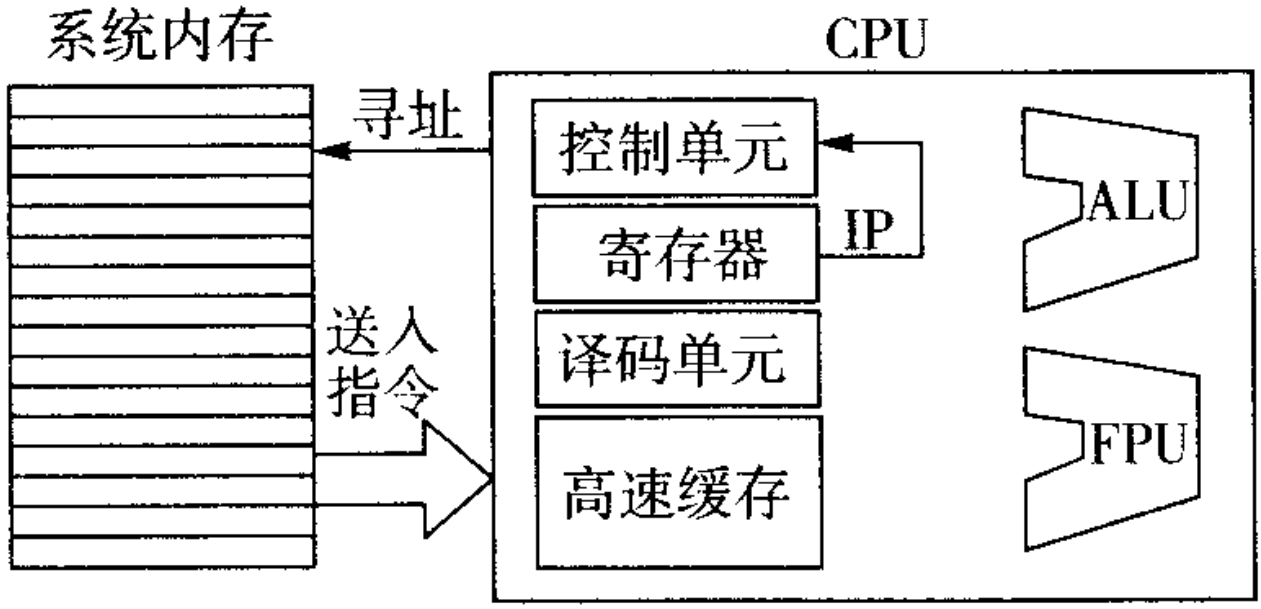
CPU指令执行流程

一条程序指令可以包含许多CPU操作。CPU的工作就是执行指令，它的工作过程是：**控制器中的指令指针给出指令存放的内存地址，指令读取器从内存读取指令并存放到指令寄存器。然后传输给指令译码器，指令译码器分析指令并决定完成指令需要多少步骤。如果有数据需要处理，算术逻辑运算单元将按指令要求工作，做加法、减法或其他运算。**
指令执行流程由**“取指令”、“指令译码”、“指令执行”、“结果写回”**四种基本操作构成，这个过程是不断重复进行的

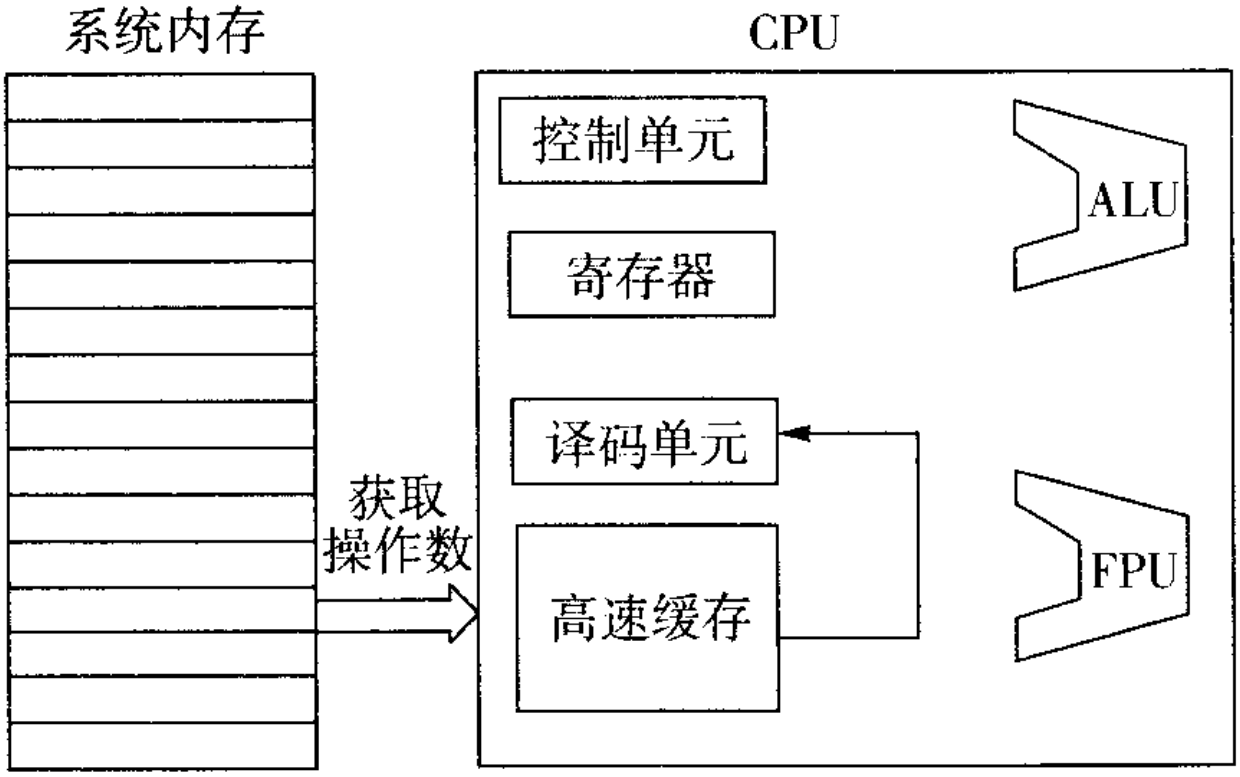


1. 取指令（IF）

   在CPU内部有一个指令寄存器（IP），它保存着当前所处理指令的内存单元地址。当CPU开始工作时，便按照指令寄存器地址，**通过地址总线，查找到指令在内存单元的位置，然后利用数据总线将内存单元的指令传送到CPU内部的指令高速缓存。**

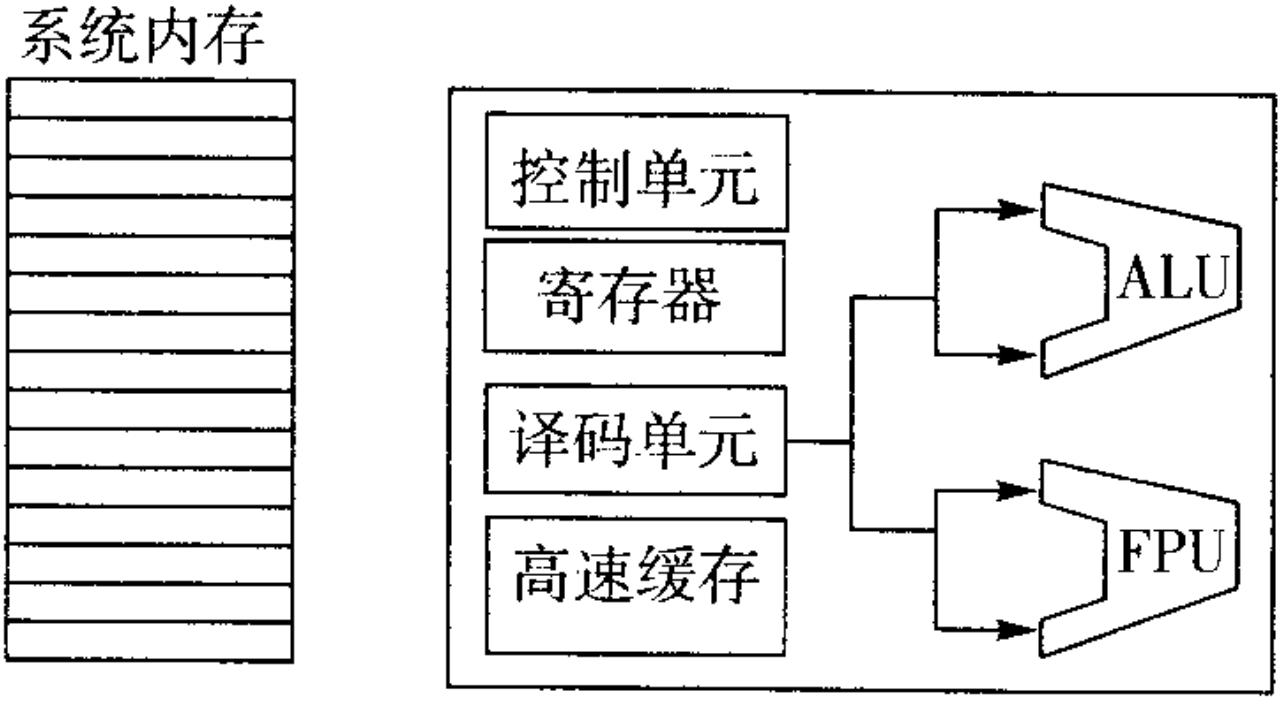


2. 指令译码（ID）
   CPU内部的**译码单元将解释指令的类型与内容**，并且判定这条指令的作用对象（操作数），并且**将操作数从内存单元读入CPU内部的高速缓存中。**译码实际上就是将二进制指令代码翻译成为特定的CPU电路微操作，然后由控制器传送给算术逻辑单元 。



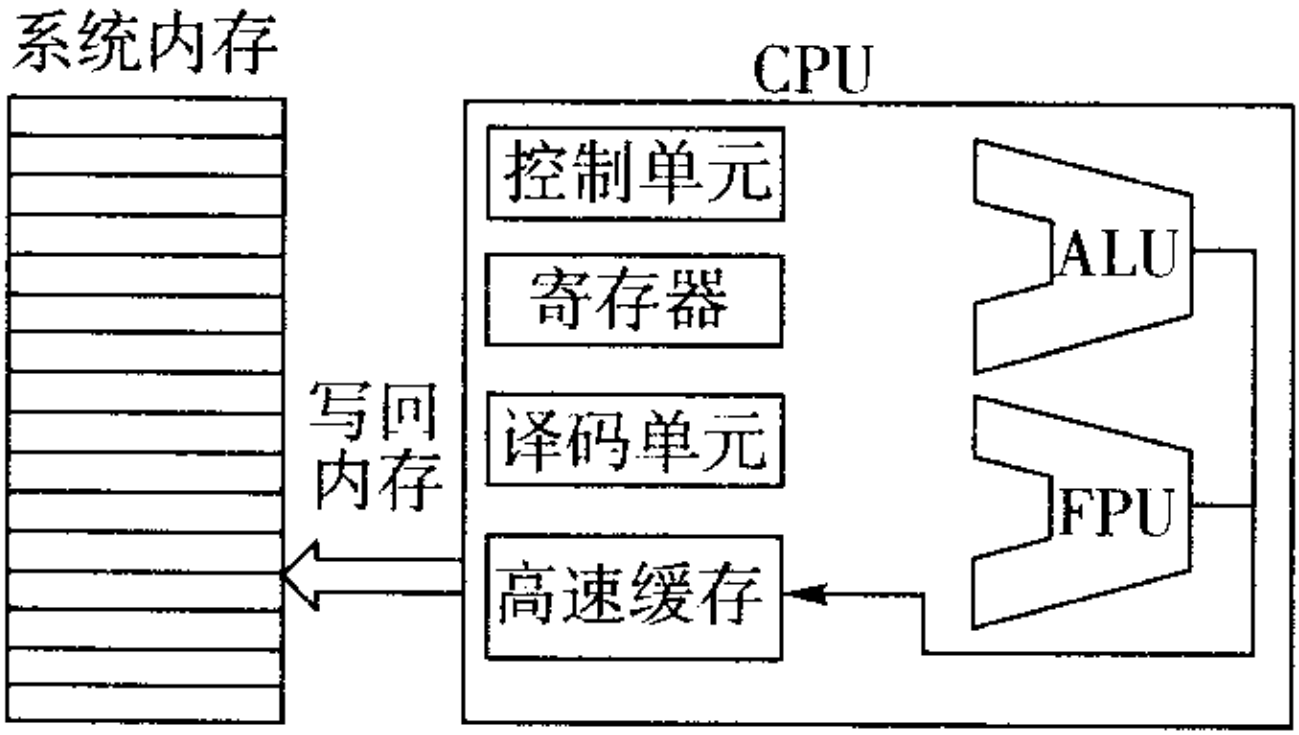
3. 指令执行（IE）

控制器根据不同的操作对象，将指令送入不同处理单元。如果是整数运算、逻辑运算、内存单元存取、一般控制指令等，则送入算术逻辑单元处理。如果操作对象是浮点数（如三角函数运算），则送入浮点处理单元进行处理。如果在运算过程中需要相应的用户数据，则CPU首先从数据高速缓存读取相应数据。如果数据高速缓存没有用户需要的数据，则CPU通过数据通道接收必要的数据。运算完成后输出运算结果。



4. 写回（WB）

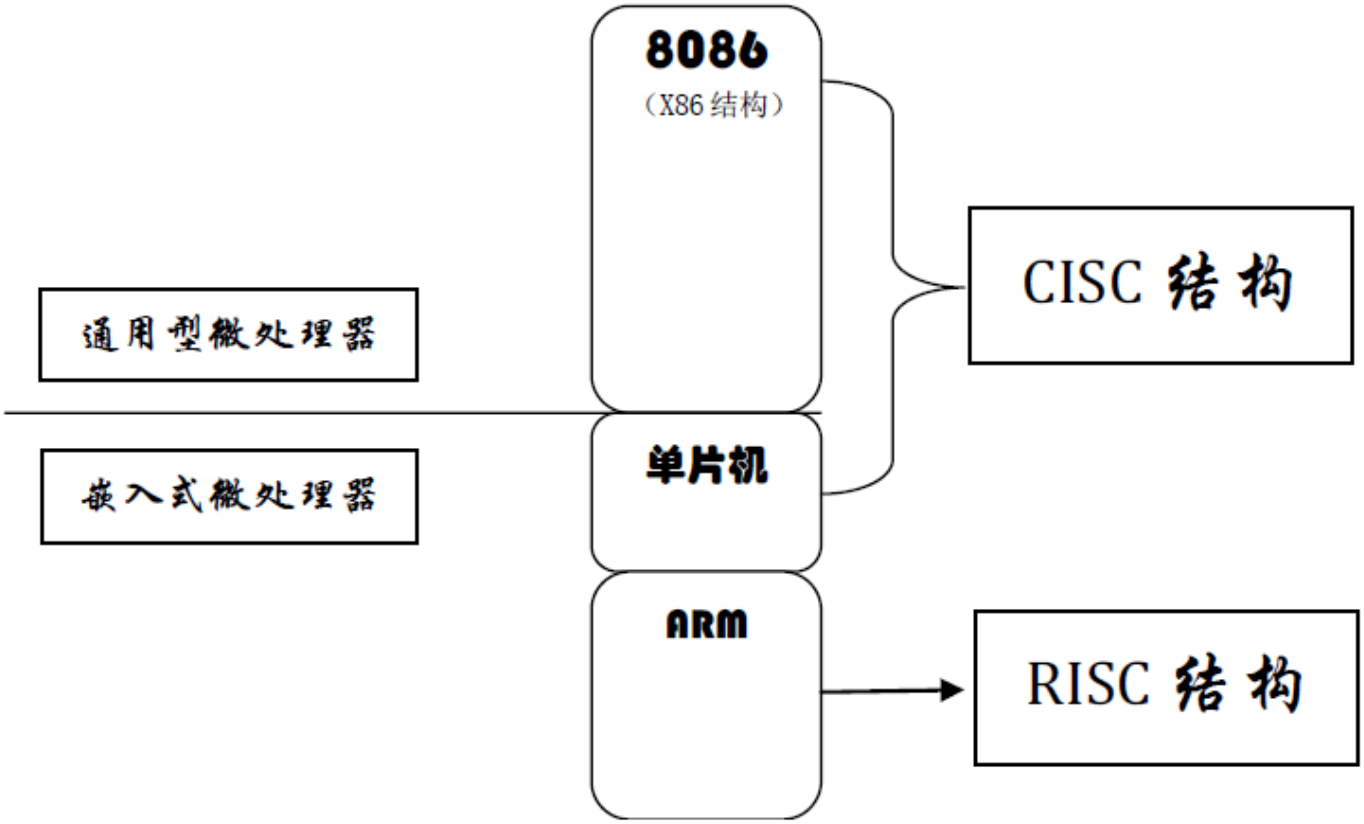
将执行单元处理结果写回到高速缓存或内存单元中。



在CPU解释和执行指令之后，控制单元告诉指令读取器从内存单元中读取下一条指令。这个过程不断重复执行，最终产生用户在显示器上所看到的结果。

微处理器设计技术结构

设计技术虽然属于CPU体系结构的第一个层次，但是它与功能单元有密切的联系，**差别在于设计技术是以整体的观点来解决问题，而功能单元是怎样去实现这种设计技术。CPU的实际目标一是怎样提高CPU的系统性能，二是怎样降低CPU的复杂程度。这是两个相互冲突的要求，从目前流行的设计技术来看主要偏重于第一个目标的实现。**
目前CPU设计的两种技术，即CISC结构和RISC结构。



8086、单片机及ARM的分类

复杂指令集计算机（CISC）及特点

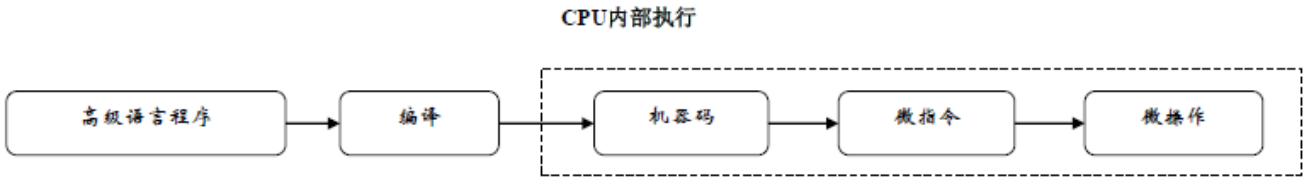
1. CISC的产生

   计算机的工作就是**取指令、执行指令。**一条指令一般由操作码和操作数（即地址码）组成，往往涉及到以下几个问题：**指令有多少位，是定长还是变长指令；操作码需几位，位数是固定量还是浮动量；操作数的地址的结构和寻址方式等。**种种因素使计算机指令产生简单指令和复杂指令之分。

   为了提高CPU的速度和功能，越来越多的复杂指令被加入到指令系统中。但是，很快又出现了一个问题：**一个指令系统的指令数是受指令操作码的位数所限制的**，如果操作码为8位，那么指令数最多为256条。改变操作码的宽度就可以加长指令的宽度，但是这会造成指令系统的不兼容现象。指令系统设计人员想出了一种方案：**操作码扩展。**

2. **CISC体系的特点**（想到微，就是CISC）

   指令长短不一：CISC（复杂指令集计算机）中有许多简单和复杂的指令，简单的指令只有1个时钟周期，而复杂指令达到了20个时钟周期以上。最长的8086指令IDIV（带符号整数除法）达到了190个时钟周期，这使CPU译码单元工作加重。由于历史的原因，x86指令长度不一致给CPU流水线技术带来了很多困难。
   庞大的指令集：在早期程序编制中，指令越多，就可以减少编程所需要的代码行数，减轻程序员的负担。由于x86系列指令集不断扩充，目前汇编指令总数已经达到500多条，而指令的组合形式达到几万种之多，这给CPU译码工作带来了严重的负担。
   微指令译码结构：在CISC结构CPU中，所有机器指令必须在CPU内部译码为微指令代码，微指令集存放在CPU的ROM中。当机器指令读入CPU内后，经过译码单元将一条复杂的x86指令译码为多个微指令代码，再送到CPU执行单元进行操作。因此，从本质上说，CISC结构CPU的译码过程是软件工作过程，它必然影响CPU的运行速度。



（微指令）

微指令就是把同时发出的控制信号的有关信息汇集起来形成的。将一条指令分成若干条微指令，按次序执行就可以实现指令的功能。若干条微指令可以构成一个微程序，而一个微程序就对应了一条机器指令。因此，一条机器指令的功能是若干条微指令组成的序列来实现的。简言之，一条机器指令所完成的操作分成若干条微指令来完成，由微指令进行解释和执行。微指令的编译方法是决定微指令格式的主要因素。微指令格式大体分成两类：**水平型微指令和垂直型微指令。**
机器指令是CPU能直接识别并执行的指令，它的表现形式是二进制编码。机器指令通常由操作码和操作数组成，**操作码指出该指令所要完成的操作，即指令的功能，操作数指出参与运算的对象，以及运算结果所存放的位置等**

软件功能硬件化：CISC通过增强指令的功能，把原本由软件实现的功能改用硬件实现。这样，一些常用的、简单的指令就不必经过译码，或者经过简单的译码就可以直接送到CPU执行单元进行处理，但是这也增加了CPU的复杂程度。
优化目标程序：CISC系统非常重视优化目标程序，这样可以缩短程序执行时间，减少程序的开销。例如：传输指令（MOV）在CPU中的使用频率占40％左右，执行时间占整个程序的30％左右，因此增强传输指令的功能有助于改进CPU执行效率。

3. **CISC体系的优点与缺点**

CISC体系的**优点**：**CPU指令集向下兼容性好**，新设计的CPU只需增加较少的晶体管就可以执行同样的指令集。新指令系统可以包含早期系统的指令集。**微操作指令的格式与高级语言相匹配**，因而编译器不一定要重新编写。
CISC体系的**缺点**：**CISC指令系统过于复杂，指令规模过于庞大。** 由于CISC结构控制复杂，不规整，不符合超大规模集成电路（VLSI）发展的方向，所以在CPU中必须最大化的使用**微指令技术**来实现CISC功能。在CISC中，虽然增加了硬件指令，但并不能保证整个程序执行时间的缩短。因为这些复杂指令要消耗较多的CPU周期数，但又不常用。

4. CISC体系的应用

8086微处理器，属于CISC体系结构，与Intel 8008兼容，指令长短不一，有较多的寻址方式，将在第三章节详细介绍 。

将CPU、存储器、I/O接口、总线等集成在一片超大规模集成电路芯片上，称为单片微型计算机，简称微控制器或单片机（MCU），其典型产品代表为**Intel MCS51**系列单片机，这将在后面的章节详细阐述。

精简指令集计算机（RISC）及特点

1. RISC技术的发展
1975年，IBM的设计师约翰·科克（如John Cocke）研究了当时的计算机系统，发现其中**占总指令数仅20％的简单指令却在程序调用中占了80％**，而占指令数80％的复杂指令却只有20％的机会用到。由此，他提出RISC的概念。事实证明RISC是成功的。20世纪70年代末，第一代RISC CPU由IBM 801实现。80年代末，各公司RISC CPU如雨后春笋般大量出现，占据了大量市场。RISC 最大特点是指令长度固定，指令格式种类少，寻址方式种类少，大多数是简单指令，并且都能在一个时钟周期内完成。RISC 易于设计超标量与流

水线，寄存器数量多，大量操作在寄存器之间进行。不过，RISC 必须经过编译程序的处理，才能发挥它的效率。

2. RISC设计思想
RISC的**主要特点**是：大多数指令在一个时钟周期内完成；采用装载—存储（Load/Store）结构；尽量将运算的数据存放在寄存器中，**从而减少访问内存的次数**；**操作由硬件完成**，而不是通过微指令完成；减少了指令和寻址方式的种类；固定指令格式；注重译码优化；面向寄存器设计指令系统；注重流水线的效率设计；重视优化编译设计。

采用装载-存储（Load/Store）结构：只允许Load和Store指令执行内存操作，其余指令均对寄存器操作。
延迟转移技术：在转移指令之后插入一条或几条有效的指令。当程序执行时，要等这些插入的指令执行完成之后，才执行转移指令，因此，转移指令好象被延迟执行了，这种技术称为延迟转移技术。 。
重叠寄存器窗口技术：调用指令（Call）、返回指令（Return）都需要传递大量参数，访问大量内存。减少访问内存次数的方法是：在CPU中设置一个数量较大的寄存器堆，并把它分为很多个窗口。每个进程使用其中的三个窗口，而在这些窗口中有一个窗口和其他进程共同使用，还有一个窗口与下一进程共同使用，目的是实现参数传递和数据共享。
指令流调整：通过寄存器置换、编译优化的方法，消除数据相关，从而使流水线的运行效率达到最高。
采用多级指令流水线结构：采用流水线技术可使每一时刻都有多条指令重叠执行。

3. RISC体系的优点与缺点

RISC指令的**优点**：在使用相同的芯片技术和相同运行时钟下，RISC系统的运行速度将是CISC的2~4倍。
RISC的指令比较简单、对称、均匀。RISC的寻址方式简单，只有装载/存储（Load/Store）指令能够访问内存，其他指令均在通用寄存器之间进行操作。
RISC指令的**缺点**：

必须精心选择通用寄存器，充分发挥每个通用寄存器的作用，尽量减少访问内存的次数。
优化编译器要做数据和控制相关性的分析，要调整指令的执行序列，并与硬件配合实现指令延迟技术和指令取消技术等。
要设计复杂的子程序库，因为在CISC中的一条指令在RISC中要用一段子程序来实现。所以，RISC的子程序库通常要比CISC的大得多

多指令的操作使得程序开发者必须小心地选用合适的编译器，而且编写的代码量会变得非常大。另外就是RISC体系的CPU需要更快的L1Cache（一级缓存）。RISC系统速度很快，但是编译一个软件时间很长。RISC系统是把最常用的那些指令设法尽快执行，不常用的指令就不怎么照顾了，甚至取消了。这样前期工作指令的统计筛选就显得特别重要。因此，RISC系统的指令集肯定都是通过大量统计筛选出来的，是最优化的，这样做嵌入式专用系统特别有优势。但是，要实现一个通用的CPU芯片，还是把RISC和CISC结合起来更好一些，以便适应多种应用的需求。

4. **RISC体系的性能特点**

- 性能特点一：由于指令集简化后，流水线以及常用指令均可用硬件执行；
- 性能特点二：采用大量的寄存器，使大部分指令操作都在寄存器之间进行，提高了处理速度；
- 性能特点三：采用缓存—主机—外存三级存储结构，使取数与存数指令分开执行，使处理器可以完成尽可能多的工作，且不因从存储器存取信息而放慢处理速度。
- 应用特点：由于RISC处理器指令简单、采用硬布线控制逻辑、处理能力强、速度快，世界上绝大部分UNIX工作站和服务器厂商均采用RISC芯片作CPU用。如原DEC的Alpha21364、IBM的PowerPC G4、HP的PA—8900、SGI的R12000A和SUN Microsystem公司的Ultra SPARC

5. RISC应用领域

基于ARM技术设计的微处理器应用占据了32位RISC微处理器大部分市场份额，ARM微处理器及技术的应用已渗入到如下各个领域：
① 工业控制领域：作为32位的RISC架构，基于ARM核的微控制器芯片不但占据了高端微控制器市场的大部分份额，同时也逐渐向低端微控制器应用领域扩展。**ARM微控制器的低功耗和高性价比，向传统的8位/16位微控制器提出了挑战。**
② 无线通信领域：很多无线通信设备采用了ARM技术，ARM以其高性能和低成本，在该领域的地位日益巩固。
③ 网络应用：随着宽带技术的推广，采用ARM技术的ADSL（Asymmetric Digital Subscriber Line非对称数字用户环路）芯片正逐步获得竞争优势。此外，ARM在语音及视频处理上进行了优化，并获得广泛支持，也对DSP的应用领域提出了挑战。
④ **消费类电子产品**：ARM技术在目前流行的数字音频播放器、数字机顶盒和游戏机中得到广泛采用。
⑤ 成像和安全产品：现在流行的数码相机和打印机中大部分采用ARM技术。**移动电话、手持式设备中的系统管理、基带信号处理、安全管理等也广泛采用了ARM微处理器。**

CISC与RISC体系的比较

- **指令系统**：RISC设计者把主要精力放在那些经常使用的指令上，尽量使它们具有简单高效的特色。对不常用的功能，常通过组合指令来实现。而CISC的指令系统比较丰富，有专用指令来完成特定的功能。
- **存储器操作**：RISC对存储器操作有限制，使控制简单化；而CISC机器的存储器操作指令多，操作直接。
- **程序**：RISC汇编语言程序一般需要较大的内存空间，实现特殊功能时程序复杂，不易设计；而CISC汇编语言程序编程相对简单，科学计算及复杂操作的程序设计相对容易，效率较高。
- **中断**：RISC微处理器在一条指令执行的适当地方可以响应中断；而CISC微处理器是在一条指令执行结束后响应中断。
- **CPU**：由于RISC CPU包含较少的单元电路，因而面积小、功耗低；而CISC CPU包含丰富的电路单元，因而功能强、面积大、功耗大。
- **设计周期**：RISC微处理器结构简单，布局紧凑，设计周期短，且易于采用最新技术；CISC微处理器结构复杂，设计周期长。
- **易用性**：RISC微处理器结构简单，指令规整，性能容易把握，易学易用；CISC微处理器结构复杂，功能强大，实现特殊功能容易。

在今天，大部分的**CISC**处理器都是基于**混合的CISC - RISC体系结构**的。这种混合体系结构**使用了一个译码器**，用于在执行前将CISC指令转换成RISC指令。这些RISC指令接下来会由一个RISC核来处理，这个RISC核可以快速执行一些简单的指令。同时，这个RISC核也采用了性能增强技术，比如分支预测技术和流水线技术。按照传统观念，只能在RISC设计中采用这样的RISC核，因为要想使这种核高效运行，则必须使用定长指令。
这种混合体系结构的几个典型例子包括Pentium处理器和Athlon系列处理器。尽管这些CISC处理器和基于RISC的处理器的执行方式完全不同，但是它们还是兼容基于它们的前任CISC处理器开发的软件。不过，CISC - RISC混合体系结构仍然耗费太多的能量，因而仍然不是移动应用及嵌入式应用的最好选择。

微处理器的六种体系结构

按照**指令的串行执行和并行执行**，可将CPU分为六种体系结构：（**从时间节点上来说**）

1. 随机逻辑体系结构（RISC构架）
   随机逻辑（**硬连**逻辑）体系结构用布尔逻辑函数来表示控制单元的输入和输出之间的关系。使用指令集结构驱动硬件逻辑方程，再由硬件逻辑方程反馈到指令集结构。一方面在复杂的随机逻辑中要求大量的不同的设计，另一方面其设计成果很少能用到新的CPU设计中，不易在逻辑中优化复杂的指令集，因此随机逻辑体系结构的CPU设计只能用于一般的结构简单的指令集。

2. **微**码体系结构（CISC构架）
   尽量将指令集与其硬件的设计分开，把每条指令分割为许多微步骤或微指令，使硬件设计转化为实现每条指令所需要的微码（一系列的微指令）的一种软件设计，其微码不但可以进行独立于硬件设计之外的设计和验证，还可修改移植到新研发的CPU中，采用微码结构使复杂的硬件设计转变为利用一套相对简单的微指令集而进行的软件编写工作。

3. 流水线体系结构
   采用流水线的处理方法，将多条指令在不同时间段并行执行。在执行每条指令完成第一步的同时，前一条指令正在执行第二步，此条指令的前一条指令正在执行第三步，依此类推。**流水线体系结构要求每条指令在执行过程中具有相同的类型和步骤，这样才能增强CPU的性能。**

4. 超流水线体系结构
   以增加流水线级数的方法来缩短机器周期，进一步提高CPU对指令运行的并行性。相同的时间内超级流水线执行了更多的机器指令。采用简单指令以加快执行速度是所有流水线的共同特点，但超级流水线配置了多个功能部件和指令译码电路，采用多条流水线并行处理，还有多个寄存器端口和总线，可以同时执行多个操作，因此比普通流水线执行的更快，在一个机器周期内可以执行多条指令。

5. 超标量体系结构 (superscalar architectures)，它能够在一个时钟周期执行多个指令。在超标量体系结构设计中，处理器或指令编译器能够判断指令能独立于其它顺序指令而执行，还是依赖于另一指令，必须跟其按顺序执行。处理器然后使用多个执行单元同时执行两个或更多独立指令。超标量体系结构设计有时称“**第二代RISC**”。

6. 片上多核处理器（Chip Multi-Processor，CMP）就是**将多个计算内核集成在一个处理器芯片中，从而提高计算能力。**按计算内核的对等与否，CMP可分为同构多核和异构多核。计算内核相同，地位对等的称为同构多核，现在Intel和AMD主推的双核处理器，就是同构的双核处理器。计算内核不同，地位不对等的称为异构多核，异构多核多采用“主处理核+协处理核”的设计，IBM、索尼和东芝等联手设计推出的Cell处理器正是这种异构架构的典范。

   - 多核之间的通信：

     CMP处理器的各CPU核心执行的程序之间需要进行数据的共享与同步，因此其硬件结构必须支持核间通信。高效的通信机制是CMP处理器高性能的重要保障，目前比较主流的片上高效通信机制有两种，**一种是基于总线共享的Cache结构，一种是基于片上的互连结构。**总线共享Cache结构是指每个CPU内核拥有共享的二级或三级Cache，用于保存比较常用的数据，并通过连接核心的总线进行通信。这种系统的优点是结构简单，通信速度高，缺点是基于总线的结构可扩展性较差。基于片上互连的结构是指每个CPU核心具有独立的处理单元和Cache，**各个CPU核心通过交叉开关或片上网络等方式连接在一起，核心间通过消息通信。**这种结构的优点是可扩展性好，数据带宽有保证；缺点是硬件结构复杂，且软件改动较大。

   - 多核处理器并行结构：

     多核处理器是指在一枚处理器中集成两个或多个完整的计算引擎(内核)。多核技术的开发源于工程师们认识到，仅仅提高单核芯片的速度会产生过多热量且无法带来相应的性能改善。即便是没有热量问题，其性价比也令人难以接受，速度稍快的处理器价格要高很多。
     多核服务器CPU，用CMP(单芯片多处理器)技术来替代复杂性较高的单线程CPU 。

   - 多核处理器的优势：

     一种应用模式是一个程序采用了线程级并行编程，那么这个程序在运行时可以把并行的线程同时交付给两个核心分别处理，因而程序运行速度得到极大提高。

更常见的日常应用程序，例如Office、IE等，同样也是采用线程级并行编程，可以在运行时同时调用多个线程协同工作，所以在双核处理器上运行速度也会得到较大提升。

对于已经采用并行编程的软件，不管是专业软件，还是日常应用软件，在多核处理器上的运行速度都会大大提高。

日常应用中的另一种模式是**同时运行多个程序**。许多程序没有采用并行编程，例如一些文件压缩软件、部分游戏软件等等。对于这些单线程的程序，单独运行在多核处理器上与单独运行在同样参数的单核处理器上没有明显差别。但是，由于日常使用的最基本的程序——操作系统——是支持并行处理的，所以，当在多核处理器上同时运行多个单线程程序的时候，操作系统会把多个程序的指令分别发送给多个核心，从而使得同时完成多个程序的速度大大加快。

单线程序进行并行优化设计，使得多核处理器的优势能得到进一步的发挥 。

- 多核的关键技术：

1、核结构研究

- 同构还是异构？CMP的构成分成同构和异构两类，同构是指内部核的结构是相同的，而异构是指内部的核结构是不同的。为此，面对不同的应用研究核结构的实现对未来微处理器的性能至关重要。核本身的结构，关系到整个芯片的面积、功耗和性能。怎样继承和发展传统处理器成果，直接影响多核的性能和实现周期。同时根据Amdahl定理，程序的加速比决定于串行部分的性能，所以，从理论上来看似乎异构微处理器的结构具有更好的性能。
- 核所用的指令系统对系统的实现也是很重要的，采用多核之间采用相同的指令系统还是不同的指令系统，能否运行操作系统等，也将是研究的内容之一。

2、程序执行模型
处理器设计的首要问题是选择程序执行模型。程序执行模型的适用性决定多核处理器能否以最低的代价提供最高的性能。程序执行模型是编译器设计人员与系统实现人员之间的接口。
3、Cache设计——**缓冲**
多级Cache设计与一致性问题。处理器和主存间的速度差距对CMP来说是个突出的矛盾，因此必须使用多级Cache来缓解。目前有共享一级Cache的CMP、共享二级Cache的CMP以及共享主存的CMP。通常，**CMP采用共享二级Cache的CMP结构**，即每个处理器核心拥有私有的一级Cache，且所有处理器核心共享二级Cache。

4、核间通信技术
CMP处理器的各CPU核心执行的程序之间有时需要进行数据共享与同步，因此其硬件结构必须支持核间通信。高效的通信机制是CMP处理器高性能的重要保障，目前比较主流的片上高效通信机制有两种，**一种是基于总线共享的Cache结构，一种是基于片上的互连结构。**
5、总线设计
传统微处理器中，Cache不命中或访存事件都会对CPU的执行效率产生负面影响，而总线接口单元（BIU）的工作效率会决定此影响的程度。当多个CPU核心同时要求访问内存或多个CPU核心内私有Cache同时出现Cache不命中事件时，BIU对这多个访问请求的仲裁机制以及对外存储访问的转换机制的效率决定了CMP系统的整体性能。

6、操作系统设计
任务调度、中断处理、同步互斥。对于多核CPU，优化操作系统任务调度算法是保证效率的关键。一般任务调度算法有全局队列调度和局部队列调度。前者是指操作系统维护一个全局的任务等待队列，当系统中有一个CPU核心空闲时，操作系统就从全局任务等待队列中选取就绪任务开始在此核心上执行。
7、低功耗设计

低功耗设计是一个多层次问题，需要同时在操作系统级、算法级、结构级、电路级等多个层次上进行研究。每个层次的低功耗设计方法实现的效果不同——抽象层次越高，功耗和温度降低的效果越明显。

8、存储器墙

为了使芯片内核充分地工作，最起码的要求是芯片能提供与芯片性能相匹配的存储器带宽，虽然内部Cache的容量能解决一些问题，但随着性能的进一步提高，必须有其他一些手段来提高存储器接口的带宽。

9、可靠性及安全性设计

处理器的安全性方面存在着很大的隐患。一方面，处理器结构自身的可靠性低下，由于超微细化与时钟设计的高速化、低电源电压化，设计上的安全系数越来越难以保证，故障的发生率逐渐走高。另一方面，来自第三方的恶意攻击越来越多，手段越来越先进，已成为具有普遍性的社会问题。现在，可靠性与安全性的提高在计算机体系结构研究领域备受注目。