**嵌入式发展史简述及一些概念**

**一、 MPU、MCU、SoC、Application Processors的概念**

<https://www.microcontrollertips.com/microcontrollers-vs-microprocessors-whats-difference/>

<https://www.microcontrollertips.com/difference-microprocessors-application-processors/>

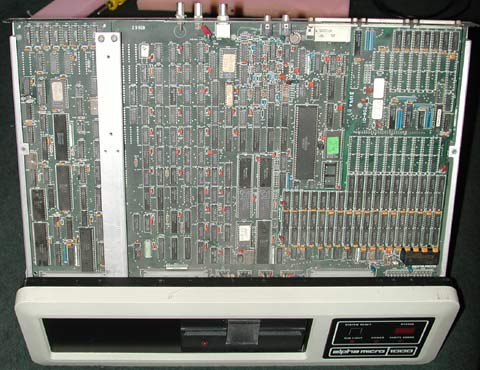
在一个电子系统中，处理器占据最重要的位置，被称为中内处理器单元：CPU(Central Processing Unit)。它从IO设备读取数据，处理，然后显示出来。

CPU的发展有两个路线：MPU、MCU。

MPU只是一个处理器，需要搭配内存等非常多的其他外设才可以构成一个系统；

MCU内部有处理器、内存、Flash及其他模块，仅仅需要搭配少量外设就可以构成一个系统。

MPU这一路线，在上世纪80年代非常流行，那时的微型计算机、游戏机都是使用MPU。比如Motorola公司的68000芯片就是一款主流的微处理器(MPU)。下图是一款Alpha Micro AM-1000系列的微型计算机电路板，电路板右侧中间最大的芯片就是MPU 68000。



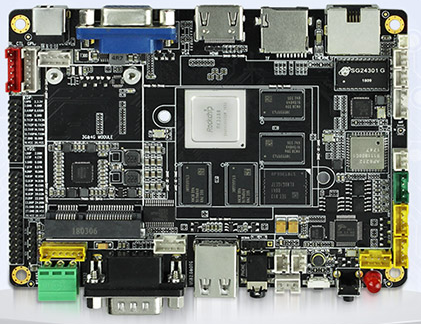
后来MPU逐渐式微，可以认为MPU发现到现在只剩下了intel、AMD公司x86系列CPU。这类CPU也越来越复杂，称之为“微处理器”似乎不恰当了。在个人电脑领域，可以看到类似下图的主板。主板上插上CPU、内存条、声卡等等就构成了一台计算机。整个主板体积庞大、制造复杂：



MCU这一路线，就是把CPU、内存、Flash都集中在一个芯片上，它再搭配其他外设备就可以构建一个完整的系统。MCU的发展经历过8位8051单片机、16位AVR单片机、32位STM单片机等等：



我们使用的51单片机、STM32单片机，一般都只是用来处理比较简单的事务，应用场景比较单一。我们使用的手机，里面的主芯片更加复杂，它既含有CPU，也含有用于数据处理的DSP、用于图形显示的GPU。使用这类主芯片设计出来的电路板更加复杂，它的性能甚至不弱于个人电脑：



为了跟个人电脑相区分，这些使用含有CPU电子产品，被称为嵌入式系统、嵌入式产品、嵌入式板卡。

嵌入式板卡就是一台形状不一样的电脑，跟电脑相比，可以引入几个概念：

**1. CPU(Central Processing Unit)：**

中央处理器，在PC机它是一个独立的芯片。

在嵌入系统中，它是芯片里的一个单元，跟其他模块比如USB、UART、音频组成一个芯片。

**2. MPU(Mircro Processor Unit)：**

微处理器单元，其作用等同于在PC上使用的CPU，它也只仅仅是一个处理器，需要配合内存、Flash等外设才可以使用。

现在，除了个人电脑上的CPU，基本上找不到MPU了。并且我们一般不把电脑上的CPU当作MPU，毕竟它也是挺大的，并不“微小”。

**3. MCU(Micro Controller Unit)：**

微控制器单元，有时又称为单片机。

MCU内部集成了处理器和各类模块，比如USB控制器、UART控制器、内存、Flash等等。只需要外接少量的器件，就可以搭建一个电子系统。

C51芯片、STM32等芯片，都是MCU。

MCU芯片内部的内存或Flash，容量在几KB、几百KB、几MB的量级，一般不再需要外接内存或Flash。

**4. Application Processors：**

手机中的主芯片跟MCU类似，也是集成了处理器和各类模块。但是它的性能已经极大提升，可以外接几GB的内存、几GB的Flash。

在手机中，这个主芯片一般用来处理显示、输入，运行用户的程序，所以称它为“Application Processors”。

“Application Processors”的概念可以扩展到其他场景，不再局限于手机。

跟MCU进行比较，Application Processors有以下不同：

a. 集成了更多的模块：

Application Processors内部集成了更多的模块，比如用于数据处理的DSP、用于图形显示的GPU，甚至有多个处理器。

这里又引入一个概念“片上系统”(SoC，System on Chip)，SoC的本意是在一个芯片上就可以搭建完整的系统。

但是这个概念在日常使用中比较宽泛：MCU芯片也可以称为SoC，Application Processors也可以称为SoC，即使它们还必须外接内存/Flash等外设才可以运行。

在以前的文档中涉及SoC时，意指比较复杂的系统；这时候MCU不属于SoC，因为MCU比较简单。但是时代在发展，MCU也越来越复杂了，所以把MCU也当作SoC也是可以的。

在手机电路板中，可用空间非常小。Application Processors还需要搭配内存芯片才可以使用，于是发展出了一种名为package-on-package (PoP)的工艺：在电路板上先焊接Application Processors，在Application Processors上面再焊接内存芯片，即2个芯片叠在一起。

b. 运行的操作系统不同：

MCU上一般不运行操作系统，或是运行一些资源耗费较小的小型实时操作系统(RTOS)。

MCU一般用来处理实时性要求高的事情，处理一些比较简单的事情。

Application Processors基本上都会运行比较复杂的操作系统(比如Linux)，在操作系统上运行多个APP。

**二、 哈弗架构与冯诺伊曼架构**



CPU架构可以分为哈弗架构与冯诺伊曼架构。哈弗架构中指令与数据分开存放，CPU可以同时读入指令、读写数据。冯诺伊曼架构中指令、数据混合存放，CPU依次读取指令、读写数据，不可同时操作指令和数据。

ARM公司的芯片，ARM7及之前的芯片是冯诺伊曼架构，ARM7之后使用“改进的哈弗架构”。“改进的哈弗结构”如下所示：



在“改进的哈弗架构”里，指令和数据在外部存储器中混合存放；CPU运行时，从指令cache中获得指令，从数据cache中读写数据。

**三、 指令集：CISC和RISC**

CISC和RISC的区别

作者：gongxsh00

来源：CSDN

原文：<https://blog.csdn.net/gongxsh00/article/details/81048671>

CISC(Complex Instruction Set Computers，复杂指令集计算集)和RISC(Reduced Instruction Set Computers，精减指令集计算集)是两大类主流的CPU指令集类型。

其中CISC以Intel、AMD的X86 CPU为代表，而RISC以ARM、IBM Power为代表。开源的RISC-V也是RISC指令集。

RISC的设计初衷针对CISC CPU复杂的弊端，选择一些可以在单个CPU周期完成的指令，以降低CPU的复杂度，将复杂性交给编译器。

举一个例子，下图是实现这样的乘法运算：a = a \* b。它需要4个步骤：读出a的值、读出b的值、相乘、写结果到a。



使用CISC提供的乘法指令，只需要一条指令即可完成这4步操作。当然，这一个指令需要多个CPU周期才可以完成。

而RISC不提供“一站式”的乘法指令，需调用四条单CPU周期指令完成两数相乘：内存a加载到寄存器，内存b加载到寄存器，两个寄存器中数相乘，寄存器结果存入内存a。

按照此思路，早期的设计出的RISC指令集，指令数是比CISC少些。后来，很多RISC的指令集中指令数反超了CISC。因此，应该根据指令的复杂度而非数量来区分两种指令集。

当然，CISC也是要通过操作内存、寄存器、运算器来完成复杂指令的。它在实现时，是将复杂指令转换成了一个微程序，微程序在制造CPU时就已存储于微服务存储器。一个微程序包含若干条微指令（也称微码），执行复杂指令时，实际上是在执行一个微程序。这也带来两种指令集的一个差别，微程序的执行是不可被打断的，而RISC指令之间可以被打断，所以理论上RISC可更快响应中断。

在此，总结一下CISC和RISC的主要区别：

1. 指令能力：

CISC的指令能力强，单多数指令使用率低却增加了CPU的复杂度，指令是可变长格式；

RISC的指令大部分为单周期指令，指令长度固定。

RISC对内存只有load/store操作，数据的运算都是在CPU内部实现。

2. 寻址方式：

CISC支持多种寻址方式；RISC支持的寻址方式少；

3. 实现方式：

CISC通过微程序控制技术实现；

RISC增加了通用寄存器，硬布线逻辑控制为主，采用流水线方式执行。

4. 研发周期：

CISC的研制周期长

RISC硬件简单，需要优化编译器。

ARM公司的芯片都使用RISC指令集，对内存只有load/store操作，数据的处理是在CPU寄存器上进行。

术语：

Execute state: 执行状态

ARMv8 架构与指令集.学习笔记 - 沉思 - CSDN博客.html

<https://blog.csdn.net/forever_2015/article/details/50285865>

AArch32和AArch64之间的切换只能通过发生异常或者系统Reset来实现.（A32 -> T32之间是通过BX指令切换的）

一篇文章读懂Armv8 AArch64 - 简书.html

<https://www.jianshu.com/p/65afe6e37764>

CISC和RISC的区别

<https://blog.csdn.net/gongxsh00/article/details/81048671>