**68000**

摩托罗拉68000型中央处理器，Motorola 68000（MC68000）

摩托罗拉68000是摩托罗拉1984年推出的32位微处理器，也是最早推出的32位微处理器。推出后，性能超群，并获得如日中天的苹果公司青睐，在自己的划时代个人电脑“Mac”中采用该芯片。但80386推出后，日渐没落。

目录

• [简介](http://www.techcn.com.cn/index.php?doc-view-141595.html#1)

• [历史](http://www.techcn.com.cn/index.php?doc-view-141595.html#3)

• [突破应用：苹果Macintoshi(1984年)](http://www.techcn.com.cn/index.php?doc-view-141595.html#5)

• [其他制造商](http://www.techcn.com.cn/index.php?doc-view-141595.html#7)

* • [应用](http://www.techcn.com.cn/index.php?doc-view-141595.html#9)
* • [作为微控制器内核](http://www.techcn.com.cn/index.php?doc-view-141595.html#11)
* • [架构](http://www.techcn.com.cn/index.php?doc-view-141595.html#13)
* • [指令集细节](http://www.techcn.com.cn/index.php?doc-view-141595.html#15)
* • [参考文献](http://www.techcn.com.cn/index.php?doc-view-141595.html#17)

简介[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

摩托罗拉68000型中央处理器，或称MC68000，是由美国摩托罗拉公司（其半导体部门现已独立成为飞思卡尔公司）出品的一款16/32位CISC（复杂指令集）微处理器。作为M68K处理器系列的第一个成员，MC68000于1979年投放市场。由于内部使用32位总线和寄存器，它在软件层（指令集）上基本与随后的纯32位产品保持兼容。目前这款微处理器仍在嵌入式领域得到应用。

XC68000，即MC68000的预发布版本，1979年

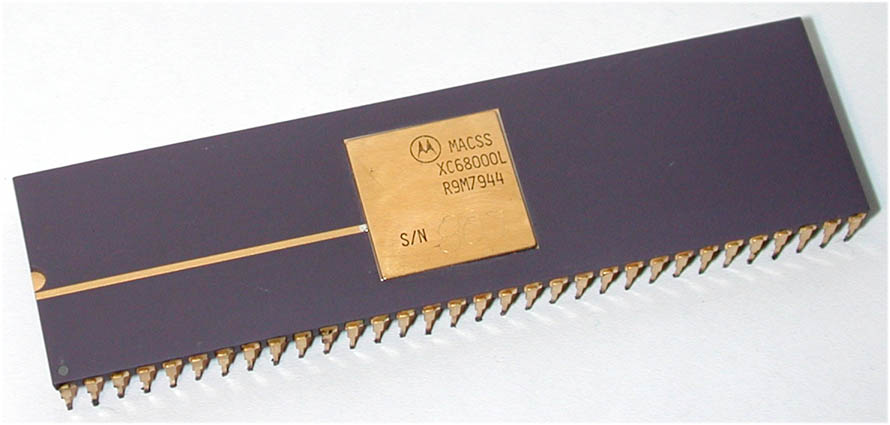
历史[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

摩托罗拉公司于1976年启动MACSS项目（Motorola Advanced Computer System on Silicon，摩托罗拉硅晶高级计算机系统），打算开发一款与以前产品完全不兼容的全新微处理器。根据计划，新CPU应该是对当时摩托罗拉主流8位CPU6800的一个高端互补产品，而不会考虑两者间的兼容性。不过，当68000设计出来后，它还是被保留了一个可兼容6800外设的总线协议模式，并且专门有8位数据总线的产品被生产出来。当然，设计人员还是更在意于其向后兼容性，这为68000确立在32位CPU领域的领先优势奠定了基础。比如，68000使用32位寄存器和内部总线，尽管其本身的结构很少直接操作长字。小型计算机诸如PDP-11和VAX—二者采用了类似的微编码—对68000的设计有深刻的影响。

在20世纪70年代中期，8位微处理器生产商纷纷竞争导入下一代16位CPU。国家半导体在1973年到1975年间首先开发出了IMP-16和PACE，但其速度并不理想。英特尔公司于1977年推出8086，迅速受到欢迎。此时，为确保竞争上的领先，摩托罗拉认识到其MACSS项目必须跳过16位系统，而直接推出16/32位混合型CPU。到1979年，摩托罗拉68000，即MC68000，才告姗姗来迟。由于比8086晚两年，其晶体管数目更多，并因其易用性得到了好评。

最初的MC68000使用3.5微米HMOS技术（即高性能N通道金属氧化物半导体，CMOS的前身）制造。1979年发布了工程样品，次年产品型面世，速度有4、6、8、10兆赫兹多种。最快的16.67MHz版本到80年代末才面市。

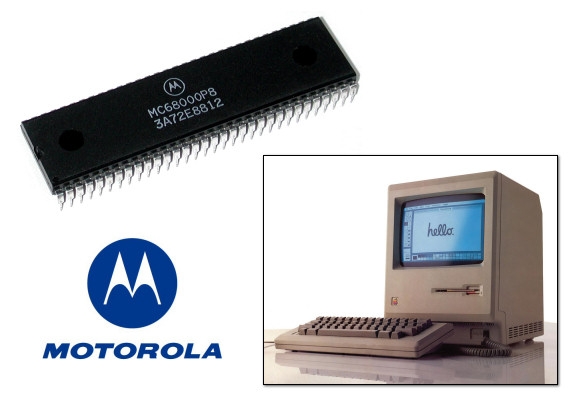
MC68000在早期得到了很多高端产品的青睐。在升阳公司的Sun workstation等多种Unix工作站中，MC68000一度占统治地位。市场领先的其他一些计算机，包括Amiga（阿米加）、Atari ST（雅达利ST）、Apple Lisa （苹果Lisa）和 Macintosh（麦金托什），以及第一代激光打印机，如苹果公司的LaserWriter，都使用MC68000。1982年，摩托罗拉进一步更新了MC68000的指令集以支持虚拟内存，并使其能够满足由Popek和Goldberg于1974年提出的虚拟化标准。



为支持低成本系统和使用较少内存的应用，摩托罗拉于1982年还推出了面向8位外部数据总线的MC68008。1982年以后，摩托罗拉开始把更多的注意力投向68020和88000。

突破应用：苹果Macintoshi(1984年)[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

　　摩托罗拉1980年推出68000时，是当时市场上最强大的处理器之一，最初它应用于一些UNIX工作站和服务器，其中包括SUN-1。但当时16/32位处理器并没有在个人电脑中产生多大影响，直到1984年，苹果推出了Macintosh，在苹果1990年代推出PowerPC之前，苹果一直都在使用68000处理器。



　　在摩托罗拉于20世纪80年代中降低了68000的价格之后，这款处理器还应用到了

　　Atari ST和Amiga计算机产品线、Sega Genesis视频游戏机和一些商业用机器中。如今68000内核仍然在一些嵌入式微处理器中使用，而这些处理器广泛应用于自动控制、显示器和洗衣机等设备中。

其他制造商[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

由日立公司设计，1985年两公司联合推出了使用CMOS技术的68HC000。68HC000的速度有8到20MHz多个版本。尽管除了使用CMOS电路，68HC000与基于HMOS的MC68000完全一致，但正是因此其能耗得到大幅下降。MC68000在25摄氏度环境下能耗大约为1.35瓦，而8MHz的68HC000则为0.13瓦，较高频率的版本则能耗也相应提高（HMOS技术则不同：其在CPU空闲时仍会耗电，所以功耗与频率基本无关）。后于1990年摩托罗拉又推出了MC68008的CMOS版本，并将其改进为可兼容8/16位两种总线模式。

其他HMOS版68000的生产商包括：Mostek、Rockwell（洛克维尔）、Signetics、Thomson（汤姆逊）和东芝。东芝也生产CMOS版68000（TMP68HC000）。

应用[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

68000首先被应用于许多高端产品中，如多用户微型计算机WICAT 150及阿尔法微系统的一些早期计算机等。单用户工作站如惠普公司的HP 9000/200、太阳微系统的Sun-1；图形终端如DEC的VAXstation 100和Silicon Graphics的IRIS 1000均使用68000。许多Unix系统也开始使用68K系列CPU。



在80年代中期，68000又成为首先应用于个人/家庭计算机的CPU。先后有苹果的Apple Lisa与麦金托什、Amiga的海军准将（Commodore）、雅达利ST及夏普的Sharp X68000采用68000。

68000最主要的成功在于控制器领域。早在1981年Imagen公司就把68000用作其激光打印机Imprint-10外部控制器的CPU。惠普于1984年发布的第一款LaserJet打印机亦使用一片8MHz的68000作为内置控制器。类似的基于68000的整合控制器也被广泛用在其他多款打印机中。到90年代，68000仍在许多低端打印机中被使用。

除了传统商业和家用计算应用，68000在工业控制系统中也取得了巨大成功。Allen-Bradley、德州仪器及西门子公司生产的可编程逻辑控制器（PLC）即使用68000。一般来说，这类工业系统的用户更重视产品的可用性，而不是向家庭用户一样过于在意其是否过时。所以，仍有许多使用68000的系统，在被安装二十多年后，在生产一线提供着持续而可靠的服务。

随着技术的进步，68000在计算机单机市场渐被淘汰，但其应用仍活跃于消费和嵌入式领域。游戏机制造商使用68000作为许多街机和家用游戏机的处理器。雅达利在1983年推出的Food Fight便是使用68000的代表街机游戏。世嘉的System 16、卡普空的CPS-1和CPS-2以及SNK的Neo Geo也都使用68000。到了90年代，尽管街机游戏开始使用更加强大的CPU，68000仍被用作声音控制器。



在80年代末到90年代初，一些游戏机厂商使用68000作为家用游戏平台的中央处理器。这包括世嘉的Mega Drive（MD）和Neo Geo家用版。后来，68000还在世嘉的32位CPU游戏机Sega Saturn中用作声音控制器。

基于68000的683XX系列微控制器则广泛应用于许多应用领域中，包括网络和电话设备、电视机机顶盒、实验室与医疗设备等。思科、3com等公司曾在他们生产的通讯设备中使用MC68302及其衍生产品。Palm公司曾使用DragonBall系列CPU（68K系列的后续）作为其PDA的处理器，直到基于RISC的ARM处理器开始统治PDA市场。此外，AlphaSmart公司在其便携式文字处理器中使用DragonBall。

德州仪器在一些高端图形计算器，如TI-89和TI-92中，使用68000。这些设备早期使用以68EC000为内核的专门化微控制器，后来则改用封装好的MC68SEC000。

作为微控制器内核[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

在被纯32位CPU取代后，68000开始被用作许多微控制器的内核。1989年，摩托罗拉发布了MC68302通讯处理器。两年后又推出独立处理器芯片MC68EC000。1996年的MC68SEC000便使用了这款芯片。

1996年摩托罗拉停产HMOS MC68000和MC68008，但其分离出来的半导体部门，飞思卡尔公司，仍在生产MC68HC000等68K家族系列产品。68000架构的后代，680x0、CPU32以及Coldfire系列，也仍在生产。

架构[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

 地址总线  
68000地址总线为24位，故支持16MB最大物理内存。在使用32位长字对地址进行存储和计算时，高位的一个字节会被自动忽略。这种设计使得其具备相当的向前兼容性，可以直接运行为后续的纯32位CPU编写的软件。也因此，根据现今的定义，68000应称得上是一款32位CPU。摩托罗拉使用32位内部总线的目的在于希望能够在68000上编写可以被将来的后续产品直接使用的软件，而相关指令不必作位数上的调整。

然而，编程人员还是有可能编写出无法与后续产品兼容的软件。倘若这种24位软件丢弃高位字节，或将该字节用作寻址以外的目的，它就有可能在32位68K系列CPU上运行失败。这就是说，对于希望支持向前兼容的软件，必须始终使用32位长字寻址，并且将最高位字节置零。

 内部寄存器  
68000包含8个32位通用寄存器（D0-D7），及8个32位地址寄存器（A0-A7）。最后一个地址寄存器，即A7，也作为标准栈指针使用，在编程中可以使用SP作为同义词。这组寄存器在规模上恰到好处：既可以对中断快速反应（只有十多个寄存器要保存），也有足够的寄存器来进行快速计算。

尽管两种寄存器并存有时会比较麻烦，但在实践中并非难于掌握。据称，这还使得CPU的设计者们可以通过对地址寄存器组使用辅助计算单元，从而实现较高程度的并行机制。

存储内容高位字节在前（Big Endian模式），与x86相反。

 状态寄存器  
68000比较、算术和逻辑操作会在状态寄存器SR的低端字节（又称CCR）中设置一些标志位，以供之后的条件跳转使用。这些标志位是：得零（Z）、进位（C）、溢出（V）、扩展（X）、负数（N）。尽管许多时候值是相同的，X与C依然是两个不同的标志位。这就允许算术、逻辑和移位操作的多余位与逻辑控制/连接造成的进位区别开。

 指令集  
68000的指令集基本上是正交的。大部分指令被划分成操作和地址模式两部分，并且大部分地址模式都对几乎全部指令可用。这种近似正交性在编程人员当中毁誉参半。

编程者会清楚地发现，他/她所书写的指令可能被汇编成几种不同的二进制操作码。这实际上是一种不错的妥协：一方面，在便利性上与纯粹的正交指令系统相仿；一方面，CPU设计者可以有更多的自由来设计操作码表。

对于一台16位时代的机器而言，由56条指令构成的最小指令集仍显巨大。此外，许多指令和寻址模式会在指令后边加入地址/寻址模式码。

许多设计者确信MC68000体系结构应基于成本考量使用较精简的指令码，特别是使用编译器自动生成时。这种认识为对其设计上的成功加分不少，并且使之成为一种经久不衰的体系结构。这一信条持续地保证了整个系列指令集的设计优势，直到ARM体系结构引入同样精简的Thumb指令集。

 特权级  
68K系列CPU包含两个特权级。超级用户（supervisor）模式和用户（user）模式。后者相比于前者只是禁用了中断级控制。中断总会使CPU进入超级用户态。超级用户标志位存储于状态寄存器SR中，并对用户可见。

超级用户态下会有一个分离的栈指针用于中断处理。

 中断  
68000可以识别7级中断，从级别1到级别7。7级中断严格按优先级排列，一个高级中断总是能嵌套于一个低级中断。可以使用专门的特权指令在SR内设置最小中断级别，从而屏蔽所有小于此级别的中断。但如果设置为0，表示不接受中断。级别7不可被屏蔽，即NMI。级别1总是可以被高级中断打断。

硬件中断源将中断信号以编码方式通过三条输入线传送给CPU。一般会使用专门的中断控制器来汇总各外部设备，并将中断信号按级编码与CPU硬连。中断控制器可以使用简单如74LS148优先级编码器，复杂如MC68901多功能外设（支持可编程中断控制、通用异步收发装置、定时器及并行输入输出等）等各种电路模块。

在内存低1K位置存储中断向量表，共支持256条中断向量。部分中断向量有特殊用途：向量1为初始栈地址；向量2为初始代码地址；向量3到15用于错误报告，包括总线错误、寻址错误、非法指令、除零异常、优先权违反等。从向量24起处理真正的中断，包括伪中断、针对级别1到级别7的默认处理向量，多达15个自陷向量，以及用户定义向量。

由于必须在重启时保证向量1和2的内容有效，所以68000系统通常包含在地址底部使用非易失性存储器（如ROM）来存储一些例程向量和启动代码。但是，一个通用计算机的操作系统会期望在运行时改变向量内容。解决办法是将ROM内的向量指向RAM的分支表，或使用早期在8位CPU中广泛使用的换页技术（Bank switching）。

由于包含一条非特权指令MOVE from SR，允许一般用户只读地访问某些特权状态，68000并未完全满足波佩克与戈德堡虚拟化需求。该需求指出了为构建某一CPU之等价虚拟机而对CPU提出的若干要求。

MC68000对虚拟内存缺乏方便的支持。一款支持虚拟内存的CPU应能在内存访问失败后自陷并恢复。不过，68000确实提供了一个总线错误异常来使CPU自陷，尽管还不能保存足够的状态信息以便于异常处理之后的恢复。为此，一些Unix工作站通过使用两块68000来解决虚拟内存问题。两块CPU的运行时钟存在相位差。当第一块遇到寻址异常后，特殊的硬件会设法向第二块发出中断，以防止其也访问错误地址。中断例程在第二块CPU上处理完内存换页后，会按之前的状态重启第一块CPU，从而再次使两CPU同步。

不过，以上这些问题在MC68010被彻底解决。在MC68010中，总线异常和地址错误均会使大量状态信息压入系统栈，以便于之后的恢复。MOVE from SR也被修正为特权指令。原本用于访问SR低端字节的代码可由新指令MOVE from CCR取代。

指令集细节[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

 标准寻址模式  
68000提供多种寻址模式，并统称为有效寻址（EA）。在CPU参考手册中，经常会有诸如MOVE <ea>,<ea>这样的表记方式。这表示在目的操作数和源操作数上可分别使用一种（但通常不是全部）有效寻址。

寄存器直接寻址   
数据寄存器直接寻址，如D0。   
地址寄存器直接寻址，如A0。通常不使用A7。   
寄存器间接寻址   
简单间址，如(A0)。先取得A0所存储的地址，再在内存中该地址处取出数据。   
自增间址，如(A0)+。只能用于源操作数域。先取得A0所存储的地址，再在内存中该地址处取出数据，然后A0的内容（地址）自增一定长度（因指令后缀而定）。这个操作其实相当于x86中的出栈指令POP（注意栈的方向和内存方向相反）。在这里，A0被用作一个用户自定义的栈指针，与系统使用的A7/A7'不同。   
自减间址，如-(A0)。可用于源或目的操作数域。先将A0的内容（地址）自减一定长度（因指令后缀而定）；然后取得A0所存储的地址，再在内存中该地址处存入数据，然后这个操作其实相当于x86中的入栈指令PUSH。[1]   
偏移间址，如2AFF(A0)。前边的偏移量为16位。实际取得的地址为(A0)+2AFF。   
加索引的偏移间址，如E3(A0, D0)。前边的偏移量为8位。实际取得的地址为(A0)+(D0)+E3。   
程序计数器位移   
长位移，如2AFF(PC)。前边的偏移量为16位。PC变为(PC)+2AFF。   
短位移，如E3(PC, D0)。前边的偏移量为8位。PC变为(PC)+(D0)+E3。   
绝对内存寻址   
直接使用内存地址值，如$4000，表示目标地址在内存地址为4000处。注意$表示后边所跟数字为16进制，%表示2进制。   
在实际编程中，可以使用代码中的标号来充当内存地址值。汇编器会自动翻译为数字地址。   
注意不应与立即数混淆。使用立即数时应在值前再加一个#。$4000表示16进制地址4000；#$4000表示16进制立即数4000；4000表示10进制地址4000；#4000表示10进制立即数4000。   
 指令后缀  
大部分指令都有表示操作长度的后缀：.B、.W或.L，分别表示这一操作在字节（8比特）上、字（16比特）上或长字（32比特）上进行。运算的过程和结果都会受到后缀的影响。在运算过程中，只有属于操作长度范围的部分才会参与运算。比方说，执行MOVE.B D2,D1会把D1的最低一字节复制到D2的最低一字节处，而两者的其余字节均不受影响。对于CCR，各标志位的值会由操作长度的最高有效位决定。如果某次ADD.B使得结果的第7Bit位为1，则CCR的N位会置1；如果某次ADD.L使得结果的第7Bit位为1，并且第15Bit不为1，则CCR的N位不会置1。在后一种情况下，CCR的N位只受第15Bit，即一个字的最高有效位影响。另一个例子是在自增/自减寻址中，自增/自减的长度因操作后缀而异。如果操作为MOVE.B，则自增/自减1；W为2；L为4。

 常用指令  
大部分68000的指令都是二元的。目的操作数在前，源操作数在后。

移动指令   
MOVE：标准移动指令。另有一些其它移动指令供选择：MOVEA（移动到地址寄存器An，不会影响CCR，但后缀不可为.B）、MOVEQ（移动一个8位数到目标寄存器，因为可将数值直接写入指令故称快速移动）、MOVEM（移动寄存器组到指定堆栈，多用于中断/子程序处理的第一步）等。   
LEA：移动一个地址值到目标寄存器。LEA $2000,A0表示将A0的值设为16进制的2000。应注意此时不加立即数标志#。   
LINK/ULNK：建立/取消堆栈帧。这个指令用于翻译高级语言的函数调用。LINK An, #-x可将An作为函数栈指针，并SP所指向的用户栈内取得x大小的空间存储局部变量，之后SP仍将指向栈顶，而An指向栈顶+x的位置。x的值在编译时由编译器自动算出。   
算数指令：ADD（加）、SUB（减）、MULU /MULS（无/有符号乘）、DIVU/DIVS（无/有符号除）、NEG （取补）、及CMP（类似于减但只会影响标志位而不改变操作数）。算数指令也多成组提供。对于ADD，还有ADDA（加地址，不会影响标志位）、ADDI（加立即数）、ADDQ（快速加，加数不大于8以便于直接放在操作码中，比x86的INC指令书写麻烦但功能更强）、ADDX（影响进位符，用于大数运算）等。   
BCD码算数指令：ABCD（加）和SBCD（减）。   
移位指令   
逻辑移位（移位后用0补充）：LSL（左移）、LSR（右移）   
算数移位（移位后用原来最高/最低有效位补充）：ASL（左移）、ASR（右移）   
循环移位（移位后用所移动位补充）：ROL（左移）、ROR（右移）   
循环扩展移位（移位至CCR的X位，同时用之前的X位补充）：ROXL（左移）、ROXR（右移）   
位操作指令：BSET（置1）、BCLR （清0）和BTST （如果测试位在改变前为0则将CCR的Z设为1）。   
多任务处理：TAS（测试并置位）。这个指令用于实现信号灯等同步机制。   
流程控制：JMP（无责任跳转）、 JSR （跳转至子例程）、BSR （按相对地址跳转至子例程，多用于跳转至用户定义例程）、RTS （从子例程返回）、RTE （从异常/中断中返回）、 TRAP（自陷，即软中断）、CHK（检查地址是否越界，如是则触发异常）。   
JMP指令只是纯粹跳转，不会将下一条指令地址压栈；JSR和BSR会将下一条指令压栈。   
RTS将栈指针指向内容弹出给PC。   
条件测试并跳转：Bcc系列指令。cc定义了所测试的条件位。常用的如：   
BNE：不等于时跳转。   
BEQ：等于时跳转。   
BRA：无条件跳转。 

参考文献[回目录](http://www.techcn.com.cn/index.php?doc-view-141595.html" \l "section)

<http://zh.wikipedia.org/zh-cn/%E6%91%A9%E6%89%98%E7%BD%97%E6%8B%89_68000>  
<http://www.cpu-world.com/CPUs/68000/index.html>  
<http://en.wikipedia.org/wiki/Motorola_68000>