重点大学计算机教材

HZ BOOKS

64位微处理器及其编程

王占杰 编译　　于泽源 刘日升 审

机械工业出版社
China Machine Press

重点大学计算机教材

64位微处理器及其编程

王占杰 编译　于泽源 刘日升 审

机械工业出版社
China Machine Press

本书主要介绍AMD64位微处理器的基本组成、体系结构、新的特点、存储模式、编程模式、指令系统、中断系统以及系统管理模式，并详细介绍硬件的工作原理、系统资源访问、软件设计方法及应用等方面的有关知识。

本书适合用作高等院校计算机科学与工程专业的高年级本科生或研究生的教材，也可用作计算机、电子工程和信息系统等专业工程技术人员的参考书。

图书在版编目（CIP）数据

64位微处理器及其编程/王占杰编著. -北京：机械工业出版社，2006，1
（重点大学计算机教材）
ISBN 7-111-17113-6

Ⅰ.6… Ⅱ.王… Ⅲ.微处理器-程序设计-高等学校-教材 Ⅳ.TP332

中国版本图书馆CIP数据核字（2005）第092260号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
策划编辑：温莉芳
责任编辑：舒　立　　隋　曦
北京京北制版厂印刷 · 新华书店北京发行所发行
2006年1月第1版第1次印刷
787mm × 1020mm 1/16 · 22印张
印数：0 001-5 000册
定价：38.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

序　言

电脑的出现深刻地改变了这个世界，也改变了人们的工作、学习和生活方式。人们对于信息技术不断增长的需求强烈地推动着电脑以更低的价格提供更高的性能。承担这一关键使命的是电脑心脏——微处理器。微处理器技术的不断创新和突破一直是推动信息技术发展的动力和源泉。

AMD在2003年4月推出了全球第一款兼容32位的64位x86处理器（AMD64），标志着64位计算普及时代的全面到来。使64位技术从高端走向实际应用，从服务器走向台式机和笔记本电脑。Linux和Solaris操作系统成功地支持AMD64计算，支持AMD64技术的64位Windows也已正式推出，64位计算正成为主流的桌面计算技术。

伴随着AMD64技术应用在我国的迅速普及，广大高校师生和科研工作者迫切需要了解和掌握AMD64技术，以便能充分利用领先的成果进行应用开发和技术创新。大连理工大学的专家们及时地编写了这本AMD64位架构及其编程的教材，相信它会对我国高校的计算机教学、科研和64位应用软件开发以及我国信息技术和产业发展大有帮助。

Su Qingxin

苏清新

AMD中国

2005年9月

前　言

本书是以AMD64有关技术文献和资料为基础编译而成的。在编译过程中，为满足教学的要求，按64位微处理器的结构特点和编程的基本需求进行了组织，集中介绍64位微处理器的新功能、新特点以及64位模式下的编程技术。前两章介绍了AMD64架构的特点，与传统的操作模式进行了比较，描述了寄存器的扩展和新的操作模式。第3章介绍了64位架构的存储模式，第4章至第8章介绍了64位编程模式、64位通用指令和系统指令，并对某些指令的工作过程进行了描述，以便更深入地理解64位微处理器的工作原理。此外，还详细论述了计算机指令集结构设计中的一些问题，包括寻址技术、指令集的功能设计、操作数的类型和大小、指令格式等。第9、10章介绍了64位架构的段式虚拟内存、异常与中断处理的有关知识，以及64位的寻址方式和中断处理方式。第11、12、13章介绍了机器检验、处理器初始化、系统管理模式与任务管理以及系统调试和性能监控等。

本书对指令分类以及高性能计算方法、向量计算、指令级并行处理进行了讨论，并论述如何利用这些技术开发软件，包括指令调度、分支处理技术和超长指令字技术。本书也论述了高速缓存的基本知识，降低高速缓存失效的方法，减少高速缓存失效开销的方法以及减少命中时间的方法，并对主存和虚拟存储器进行讨论。

在本书的初期文献翻译工作中，刘艳博、曹江波、金全凯、周小兵、王媛、赵殿奎、肖侯亮、查峰、刘力协、傅一凡、李贤、宋丹、赵丹等计算机网络与控制研究室的全体人员参加了翻译整理工作，在此表示感谢。

本书由于泽源教授、刘日升教授主审，提出了许多宝贵的修改意见。在本书编写过程中，得到了美国AMD公司和AMD中国分公司苏清新博士以及有关技术人员的大力支持，并得到了软件学院沈宏书教授、电信学院及计算机系领导和师生的多方面帮助，在此一并表示衷心的感谢。

由于作者水平有限，书中难免有错误和不妥之处，敬请读者批评指正。

王占杰

2005年9月

目　录

第1章 AMD64架构概述

1.1 综述

计算机从诞生到现在已有了飞速的发展，回顾计算机的发展史可以看出，计算机系统性能的不断提高，不仅依靠器件的改革，还要靠计算机系统结构的改进。所谓计算机系统结构就是指把各个功能部件组合成一个系统，这些部件可以是硬件、软件或两者的混合体。另外，对于CPU而言，重要的指标还有寻址空间上的提升。传统32位处理器只拥有32条地址线，可管理4GB的内存。而64位微处理器则拥有64条地址线，可以管理16TB的内存空间！计算机运行的快与慢，要看它的运行频率有多快和每次处理的数据量。目前32位处理器在运行过程中，每次可以处理掉32位二进制数，即4字节。而64位的处理器则每次可处理64位数据，即8字节。频率相同时，性能的提高就十分明显了。简单的说性能就是：时钟速度×每个时钟处理的指令数目。

64位架构是简单而功能强大的64位结构，它向后兼容扩展了x86架构的工业标准。它增加了64位寻址并扩展了寄存器资源，以便支持重新编译的更高性能的64位程序。同时它还支持传统的16位和32位的应用程序和操作系统而不必修改和重新编译它们。Intel 64位技术和AMD64实际上是同一架构，两者相互兼容。64位是系统结构的基础，基于它的新型处理器能对庞大的已有软件系统和要求高性能应用的新型64位软件提供无缝的高效的支持。AMD公司率先推出了AMD64架构系统，以满足市场的需要。

那些需要大容量虚拟地址和物理地址的应用加强了对64位x86架构的需要，例如：高性能服务器、数据库管理系统和CAD工具。这些应用都能从64位寻址和增加寄存器数目中获益。而在传统的x86架构中，少量可用的寄存器数目限制了计算密集应用的性能，在增加寄存器的数量后则提高了这方面应用的性能。

1.1.1 AMD64架构的特征

AMD64架构引入了以下新的特征：

1）寄存器的扩展（见图1-1）

- 8个新的通用寄存器（GPR）
- 所有的16个GPR都是64位字宽
- 8个新的128位XMM 寄存器
- 对于所有的GPR都用统一的字节寄存器寻址
- 访问扩展寄存器的64位指令前缀

2）长模式（见表1-1）

- 升到64位虚拟地址（最大支持）
- 64位指令指针（RIP）

- 新的“相对指针的寻址”模式
- 线性地址空间

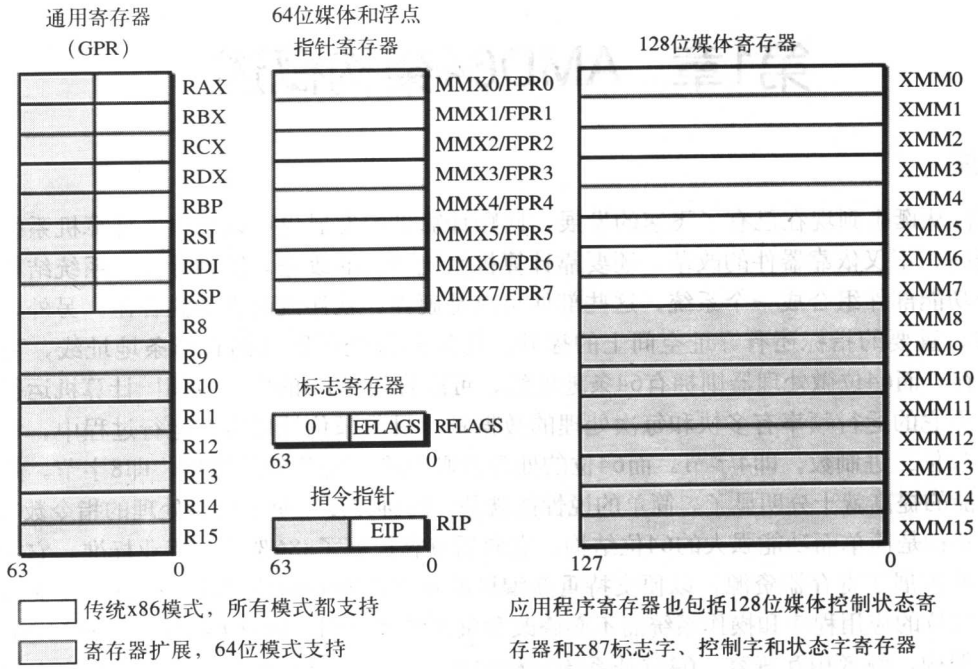

图1-1 应用编程寄存器集

表1-1 操作模式

操作系统		操作系统要求	应用程序重编译要求	默认值		寄存器扩展	典型的GPR宽度（位）
				地址值大小（位）	操作数大小（位）		
长模式	64位模式	64位操作系统	是	64	32	是	64
	兼容模式		否	32		否	32
				16	16		16
传统模式	保护模式	传统32位操作系统	否	32	32	否	32
				16	16		
	虚拟8086模式			16	16		16
	实模式	传统16位操作系统					

一、寄存器

表1-2对在各种操作模式下可用于应用软件的寄存器和栈资源进行了比较。表的左边几列示出了传统的x86资源，这些资源在AMD64架构的传统模式和兼容模式下是可用的。右边的几列

示出了64位模式下可比较的资源。灰色阴影部分展示了这些模式的不同之处，这些寄存器的不同（不包括栈宽不同）反映了表1-1展示的寄存器扩展。

表1-2 各种操作模式下的应用寄存器和栈

寄存器或栈	传统和兼容模式			64位模式①		
	名字	数量	大小（位）	名字	数量	大小（位）
通用寄存器（GPR）②	EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP	8	32	RAX,RBX, RCX, RDX, RBP, RSI, RDI, RSP, R8-R15	16	64
128位MMX寄存器	XMM0-XMM7	8	128	XMM0-XMM15	16	128
64位MMX寄存器	MMX0-MMX7③	8	64	MMX0-MMX7③	8	64
x87寄存器	FPR0-FPR7③	8	80	FPR0-FPR7③	8	80
指令指针	EIP	1	32	RIP	1	64
标志②	EFLAGS	1	32	RFLAGS	1	64
栈②	—		16或32	—		64

① 灰色阴影部分展示了各个模式之间的不同。这些不同（栈宽度不同除外）就是AMD64位体系结构寄存器的扩展部分。

② GPR的列表只列出了32位寄存器。32位寄存器的16位和8位的映像也是可以访问的。

③ 如图1-1所示，MMX0~MMX7寄存器也被映像到FPR0和FPR7物理寄存器上。x87栈寄存器ST(0)~ST(7)，是FPR0和FPR7物理寄存器的逻辑映像。

如表1-2所示，x86架构（在AMD64架构中叫做传统模式）支持8个GPR。事实上，在常规使用中至少有4个寄存器（EBP、ESI、EDI和ESP）的用途是不能发挥正常作用的，因为当执行许多指令时它们提供了专用的功能。AMD64架构增加了8个新的GPR并且把这些寄存器的宽度从32位增加到64位，这使得编译器能充分地提高软件的性能。编译器在用寄存器存储变量时有了更大的弹性。通过在GPR中局部地完成工作，编译器也能最小程度地降低存储器的通信量，从而提高了性能。

二、系统寄存器

图1-2示出了为AMD64架构定义的系统寄存器。系统软件使用这些寄存器管理处理器运行环境、定义系统资源特征以及监控软件执行。除了RFLAGS寄存器之外，只有特权软件才能读写系统寄存器。

除描述符表寄存器和任务寄存器外，AMD64架构将所有的系统寄存器定义为64位宽。描述符表和任务寄存器被AMD64架构定义为包括64位基址域，还有其他的域。

如图1-2所示，系统寄存器包括:

- 控制寄存器——用于控制系统运行和一些系统特性。
- 系统标志寄存器——RFLAGS寄存器，包括系统状态标志和屏蔽位。它也用于激活虚拟8086模式和控制应用程序访问I/O设备和中断。
- 描述符表寄存器——包括描述符表在内存中的位置和大小。在保护模式中描述符表保存了分段使用的数据结构。

图1-2 系统寄存器

- 任务寄存器——包括任务状态段（TSS）在内存中的大小和位置。硬件多任务机制使用任务状态段来保存一个给定任务的状态信息。TSS也保存其他的数据，比如当需要切换到更高特权级而使用的内部级别的堆栈指针。
- 调试寄存器——用于控制软件调试机制，以及为调试用途或向应用程序回馈信息。

定义为系统寄存器也是一定数目的模型指定寄存器，它包括在AMD64架构定义中，如图1-2所示。

- 扩展特征使能寄存器（EFER）——EFER寄存器用于激活和报告不受CRn控制寄存器控制的特殊的特征状态。特别是EFER寄存器用于控制长模式的激活。
- 系统配置寄存器（SYSCFG）——SYSCFG寄存器用于激活和配置系统总线特征。
- 系统连接寄存器——系统连接指令使用这些寄存器来指定操作系统的入口指针、堆栈位置和系统数据结构指针。
- 存储类型寄存器——存储类型寄存器可以用于表述系统内存的特性。系统软件给出了存储类型控制指令和数据如何高速缓存、内存读写是怎样的顺序。
- 调试扩展寄存器——这些寄存器控制软件调试报告的附加特征。

- 性能监控寄存器——用于为处理器和系统事件计数，以及事件的持续时间。
- 机器检查寄存器——机器检查寄存器控制处理器对不可恢复故障的响应。它们还用来报告这样的故障返回到系统的有用信息。

1.1.2 AMD64架构指令集的特点

AMD64架构指令集支持所有的x86指令集，并且增加了一些新的支持长模式的指令（参见表1-1操作模式）。这些应用编程指令的分类和描述如下:

- 通用指令——实际上这些指令是指在所有的编程中都用到的那些基本的x86整数指令。其中大多数指令都是对保存在GPR和内存中的数据进行装载、存储或操作。一些指令通过跳转到程序中的其他位置来改变程序的流程。
- 128位媒体指令——这些指令是SIMD的扩展（SSE和SSE2）指令，这些指令主要加载、存储或操作存在128位XXM寄存器中的数据。它们能够对标量和向量的数据类型进行整数和浮点运算。因为这些向量指令能同时独立地对多个数据集的数据执行同一条指令，所以称它们为单指令流多数据流（SIMD）指令。对于那些高性能的媒体和操纵数据块的科学计算，这些指令很有用。
- 64位媒体指令——这些指令是指多媒体扩展指令（MMX™ 技术）和AMD 3DNow!™ 技术指令。它们主要加载、存储或操纵保存在64位MMX寄存器中的数据。与上面介绍的128位的媒体指令相对应，它们能够对标量和向量的数据类型进行整数和浮点运算。因此它们也是SIMD指令，同样可用于操纵数据块的媒体应用中。
- x87浮点指令——这是用在传统x87中的那些浮点指令。它们加载、存储或操纵保存在x87寄存器中的数据。

某些应用程序的指令跨越上述两个或两个以上的子集。例如，有一些指令在通用寄存器和XMM或MMX寄存器之间传送数据。尽管这些运算不是同时的，许多整数向量指令能在XMM或MMX上进行运算。假如某些指令是跨两个或两个以上子集的，那么它们将在其应用的所在子集中描述。

一、媒体指令

媒体应用，例如图像处理、音乐合成、语音识别、高速视频和3D图像分析，都有如下特征:

- 它们处理大量的数据。
- 它们通常对数据重复执行大量相同序列的操作。
- 这些数据通常是小数值的，如8位的像素值、16位的音频采样值和32位的浮点格式的对象坐标。

128位和64位媒体指令是为了加速这方面的应用而设计的。这些指令采用SIMD（单指令流多数据流）向量并行处理形式。这些向量技术有如下特征:

- 一个单独的寄存器能存储若干独立的数据块。例如，一个128位XMM寄存器能存16个8位整数或4个32位单精度浮点数。
- 向量指令能同时独立地操作一个寄存器中的所有数据。例如，一个（操纵128位XMM寄存器中的两个向量操作数的字节元素）PADDB指令能同时运行16个加法并在一个操作周期

返回16个独立的结果。

128位和64位指令含有一些特殊指令，而这些指令通常完成的操作是建立在媒体应用基础上的，这就使得SIMD向量技术更进了一步。例如两个像素亮点值相加的图像应用，一定要防止加操作回绕到一个小值，即结果在目的寄存器产生了溢出，溢出会引起不可预料的后果，如在预期产生亮点的地方产生了黑点。128位和64位媒体指令具有饱和算法指令，使得对这种类型的操作变得很容易。那个能覆盖回绕的结果，不管其有没有上溢或者下溢、都强制被转化为目的寄存器中的最大或最小值。

二、浮点指令

AMD64架构利用3种不同的寄存器提供了3种浮点指令集:

- 128位媒体指令除了支持整数操作外，还支持32位单精度和64位双精度浮点操作。对向量和标量数据的操作都支持，并带浮点异常报告机制。这些浮点操作符合IEEE-754标准。
- 64位媒体指令（3DNow!™技术指令的子集）支持单精度浮点操作。对向量和标量数据都支持，但不支持浮点异常报告。
- x87浮点指令支持单精度，双精度和80位扩展精度指令操作。只支持标量数据的操作，并带浮点异常报告机制。x87浮点指令包含一些特殊的指令，这些指令在三角和对数运算方面比先前的指令更胜一筹。单精度和双精度浮点操作符合IEEE-754标准。

通过用128位的媒体指令能达到浮点的最大性能。向量指令能同时支持多达4个单精度（或两个双精度）操作。在64位模式下，AMD64架构将原XMM寄存器的数目加倍，从8个增到了16个。

通过用64位媒体指令和x87指令，应用程序获得了一些附加的好处。由这些指令所支持的单独的寄存器组减缓了用于128位媒体指令的XMM寄存器的压力。它提供给应用程序3个不同的浮点寄存器组。此外，对某些AMD64架构的高端实现还用单独的执行单元支持128位媒体指令、64媒体指令和x87指令。

1.1.3 REX指令前缀

指令前缀就是指令操作码前面的字节，它能修改指令的操作和操作数。指令前缀有两种类型:

1）传统前缀

2）REX前缀

传统前缀被分为5组，每组前缀的值都是惟一的。REX前缀被分为一组，在这个组中，前缀的值表明扩展寄存器的组合特性有效，REX前缀使AMD64架构扩展寄存器在64位模式下可用。

一、传统前缀

表1-3展示了传统前缀。它们被分为5组。每个前缀都有一个惟一的16进制的数。传统前缀可以出现在指令的任何顺序中，但是每个组中只有一个前缀能用在一条指令中。用一个组中的多重前缀得到的结果是无意义的。

用这些前缀有一些限制。例如地址大小前缀只能改变一个内存操作数的大小，并且一个内存操作数能在一个指令中重载。通常，操作数大小前缀不能用于x87浮点指针指令，并且当用于128位媒体指令和64位媒体指令时，这些前缀以一种特殊的方式来修改操作码。重复前缀只有与

某几个串指令联合使用时才能引发重复执行，并且当用于128位媒体指令和64位媒体指令时，这些前缀以一种特殊的方式来修改操作码。锁前缀只能与一小部分通用指令联合使用。表1-3对指令前缀的功能进行了概述。

表1-3 传统指令前缀

前缀组	助记符	前缀码（十六进制）	描述
操作数大小替换	无	66	改变内存和寄存器操作数的默认大小
地址大小替换	无	67	改变内存操作数的默认地址大小
段替换	CS	2E	内存操作数强制使用CS段
	DS	3E	内存操作数强制使用DS段
	ES	26	内存操作数强制使用ES段
	FS	64	内存操作数强制使用FS段
	GS	65	内存操作数强制使用GS段
	SS	36	内存操作数强制使用SS段
加锁	LOCK	F0	使一些在内存上的读-改变-写操作自动发生
重复	REP	F3	重复执行一个字符串操作（INS, MOVS, OUTS, LODS和STOS）直到rCX寄存器为0
	REPE或REPZ		重复一个字符串比较或者字符串扫描操作（CMPSx和SCASx）直到xCX寄存器为0或者0标记符（ZF）被清0
	REPNE或REPNZ	F2	重复一个字符串比较或字符串扫描操作（CMPSx和SCASx）直到xCX寄存器为0或者0标记符被置为1

注：当128位或者64位媒体指令一同使用时，这些指令能通过一种特殊的方式来修改操作码。

操作数大小和地址大小前缀。在一个指令接着一个指令的基础上，操作数大小和地址大小前缀允许数据和地址大小的混合。在任何操作模式下，一个指令的默认地址大小可以通过使用67H地址大小前缀替换。

在64位模式下，大多数通用指令的默认操作数大小是32位的。REX前缀指定了一个64位操作数大小，并且66H前缀指定一个16位操作数大小。REX前缀优先于66H前缀。

在64位模式下，默认的地址大小是64位的。地址大小可以替换成32位。16位地址在64位模式下不被支持。在兼容模式下，地址大小前缀与在传统x86体系结构下的工作方式相同。

段替换前缀。对于大多数内存操作数，DS段是默认段。许多指令允许用一个段替换前缀对默认数据段进行替换。当在下列情况下访问数据时，数据段的替换将被忽略。

- 当一个栈把数据压入栈和从栈中取出时，引用SS段。
- 当一个串的目的操作数是一个内存地址时，用ES段。

从CS段取出的指令不能被替换。尽管如此，CS段替换前缀可以作为一个数据对象访问指令，也可访问当前代码段的数据。

锁前缀。锁前缀引发了某些“读-修改-写”指令，这些指令自动引发内存访问调用。为了这样做，这些机制是由系统实现决定的（例如，该机制可能涉及高速缓存行的锁，它包括引用

内存操作数的副本、总线信号或总线上的包消息）。这个前缀在多处理器系统中可以实现对共享内存中数据的互斥使用。

这个前缀只能与写内存操作数的指令联合使用，这些指令如：ADC、ADD、AND、BTC、BTR、BTS、CMPXCHG、CMPXCHG8B、DEC、INC、NEG、NOT、OR、SBB、SUB、XADD、XCHG和XOR。如果LOCK与其他指令联合使用将发生一个无效操作数码异常。

重复前缀。有两个重复前缀操作码，F3H和F2H。字节码F3H是最普通的并且通常被编译器看作是两个不同的指令。字节码F2H仅同CMPSx和SCASx指令一起使用：

- REP（F3H）——这个通用重复前缀将重复与它相关的串指令，重复的次数由rCX指定。当rCX达到0时，重复停止。这个前缀同INS、LODS、MOVS、OUTS和STOS指令联合使用。
- REPE或REPZ（F3H）——REP前缀的这个版本重复与它相关的串指令，重复的次数由rCX指定。当rCX达到0或ZF复位为0时，重复停止。这个前缀仅同CMPSx和SCASx指令联合使用。

rCX计数器的尺寸由有效地址大小决定。

- REPE或REPZ（F2H）——REPE或REPZ前缀重复与它相关的串指令，重复的次数由rCX指定。当rCX达到0或当零标志位置1时，重复停止。该前缀只能同CMPSx和SCASx指令联合使用。

rCX计数器的大小由有效地址的位数决定。

二、REX前缀

REX前缀是一组新的指令前缀，能用于64位模式下。它们使AMD64架构扩展寄存器可用。REX前缀有以下特征：

- 使用扩展GPR寄存器。
- 使用XMM寄存器。
- 使用64位（四倍字）操作数。
- 使用扩展控制寄存器和调试寄存器。

根据扩展寄存器特殊组合需求的不同，REX前缀字节有一个从40H到4FH的值。个别情况下，访问一个64位的GPR或扩展GPR或XMM寄存器时，REX前缀是必须的。某些指令默认的操作数大小就是64位的，这时无需REX前缀来访问扩展的64位GPR。

一个指令只能有一个REX前缀，一个这样的前缀就可以表达64位模式的扩展寄存器的所有特征。如果需要，这个前缀在一个指令的第1个操作数的前面。REX前缀的其他位置被忽略。以15字节为极限的传统指令大小也适用于那些包括REX前缀的指令。

1. *支持的前缀*

1）下面前缀可用于128位或64位媒体指令：

地址大小替换——67H前缀仅在内存中影响操作数。该前缀被别的128位或64位媒体指令所忽略。

操作数大小替换——66H前缀用于形成特定的128位或64位媒体指令操作码。该前缀被其他所有媒体指令所忽略。

段替换——2EH（CS）、36H（SS）、3EH（DS）、26H（ES）、64H（FS）和65H（GS）前