**Intel的64位扩展技术简介**

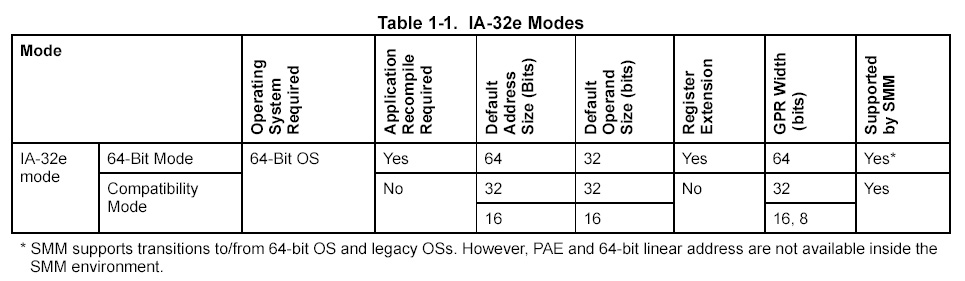
杨全胜翻译

【译者注】Intel公司在IDF2004上展示了他们的IA-32e架构新的处理器，这是一个利用兼容IA-32架构的64位扩展技术开发的64位微处理器，为了让大家尽早了解该项技术，本人特根据Intel公司提供的64-Bit Extension Technology Software Developer's Guide的部分内容翻译编辑了本文。

1. **操作模式**

具有64位扩展技术的处理器能运行在传统IA-32模式或IA-32e模式。传统的IA-32模式允许处理器运行在保护模式、实地址模式或虚拟8086模式。

IA-32E模式是处理器在运行64位操作系统的时候使用的一种模式。带有64位扩展技术的处理器将初始进入传统的、页式地址、保护模式，然后，当IA32-EFER寄存器中的某位被设置并且PAE(Physical Address Extensions，物理地址扩展)模式被使能。下表显示了64位扩展技术所支持的操作模式和他们之间的区别。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e1.jpg)

1. **IA-32e模式**

IA-32e模式有两个子模式：64位模式和兼容模式。IA-32e模式只能在装载64位操作系统的情况下进入。

1. **64位模式**

64位模式用于运行在64位操作系统中的64位应用程序它支持以下的特性：

* 支持64位线性地址结构；然而支持64位扩展技术的IA-32处理器将用少于64位地址来实现
* 寄存器扩展后，可以使用新的操作码前缀来访问（REX）
* 现有的通用寄存器被加宽到64位(RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP)
* 8个新的通用寄存器(R8–R15)
* 8个新的128位流SIMD扩展（SSE）寄存器(XMM8–XMM15)
* 一个64位的指令指针（RIP）
* 一个新的RIP相关数据寻址模式
* 对单一的代码、数据和栈空间能用平板地址空间
* 扩展的和新的指令
* 支持大于64GB的物理地址；然而支持64位扩展技术的IA-32处理器的实际物理地址要特殊实现
* 新的中断优先级控制机制

64位模式能够在以代码段为基础的操作系统中被使用。它的缺省地址大小是64位；它的缺省操作宽度大小是32位。注意这些缺省设置能够在使用新的REX操作码前缀的指令－指令对中被超越。当操作在64位模式下时，REX前缀允许指定一个64位操作数。利用这个机制，很多现有的指令被修改或重新定义来允许使用64位寄存器和64位地址。

1. **兼容模式**

    兼容模式允许传统的16位和32位应用程序无需重新编译就可以运行在64位操作系统下（然而运行在虚拟8086模式下或使用硬件任务管理中的传统应用程序将无法工作）。就像64位模式那样，操作系统在一个专门的代码段使能兼容模式。这意味着64位应用程序能运行在处理器中（64位模式）的同时，32位应用程序（没有为64位重编译的）运行再兼容模式。

兼容模式像传统的保护模式。应用程序只能存取线性地址空间中的第一个4GB，处理标准IA-32指令前缀和寄存器。在兼容模式下不提供REX前缀。（REX前缀编码已经处理成传统IA-32指令）兼容模式也必须使用16位和32位地址和操作数。和传统保护模式一样，兼容模式也允许应用程序使用PAE（物理地址扩展）处理64GB的物理存储。

下列传统保护模式下的项目，在兼容模式下不支持。

* 虚拟8086模式，任务切换和栈参数拷贝特性在兼容模式下不可用
* 从操作系统的角度看：使用64位机制替代32位机制来处理系统数据结构，地址变换，中断和异常处理等结构和事务。

1. **传统模式为什么Intel现在做这件事情？**

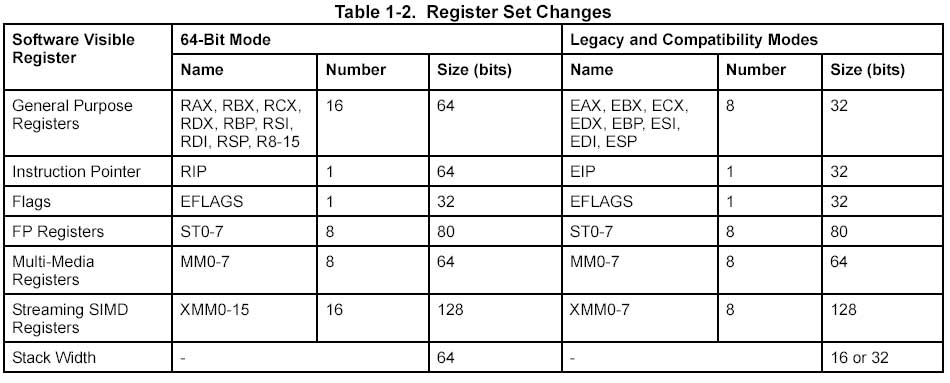
传统模式包括保护模式，实地址模式和虚拟8086模式。现有的为这些模式中的任何一种模式而编写的软件都完全能兼容地运行在具有64为扩展技术的IA-32处理器中。

1. **系统管理模式**

系统管理模式（SMM)提供与传统IA-32架构中的系统管理中断（SMI）处理程序相同的执行环境。SMM支持从一个模式到另一个操作模式（包括IA-32e和传统模式）的转换。一个SMI处理程序能够通过PSE机制处理任何的物理存储页。然而由于不支持PAE，SMM环境不支持64位线性地址。提交给SMI的事务，处理器将转换到SMM，并根据SMM存储映射（save map）将存储器的状态存储到SM RAM中。因此，一个SMI处理程序将执行在和传统IA-32架构中一样的环境中。

1. **寄存器组的改变**

下表比较了运行在64位模式的应用程序和运行在传统的IA-32环境的应用程序中寄存器数据结构的不同。传统环境包括那些存在于现有IA-32处理器、支持64位扩展技术的处理器中的传统模式以及IA-32e兼容模式中的环境。兼容模式应用程序不能在64位模式或64位操作系统中运行，因此应用程序需要运行在传统IA-32保护模式环境中。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e2.jpg)

1. **通用寄存器（General-Purpose Registers,GPRs)**

IA-32结构运行在传统或兼容模式时，有8个通用寄存器。AX, BX, CX, DX, DI, SI, BP, SP对16位操作数有效，EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP对32位操作数有效。

在64位模式，缺省的操作数是32位的，然而GPRs可以针对32位和64位操作数。如果是32位操作数 EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D - R15D可用，如果是64位操作数RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8-R15可用，R8-R15是8个新的GPRs。所有的这些寄存器能够具有字节、字、双字和四字四个级别。这些级别的划分主要是看REX前缀。

在64位模式，将限制指令存取字节寄存器，指令不能同时使用传统的高字节（比如AH, BH. CH, DH）和新的字节寄存器（比如RAX寄存器的低字节）。然而指令将可同时用传统低字节（比如AL,BL,CL或DL）和新的字节寄存器（比如R8寄存器或RBP）。这种结构将强迫大家遵守以上的限制，并将任何代REX前缀的指令对高字节(AH, BH, CH, DH)的使用转换到低字节(BPL, SPL, DIL, SIL; 这些是RBP,R SP, RDI 和 RSI的低8位) 的使用。

在64位模式下，操作数的大小决定了目标GPR的有效位数：

* 64位操作数产生一个64位的结果到目标通用寄存器
* 32位操作数产生32位的结果，采用0-扩充的方法将64位结果写到目标通用寄存器
* 8位和16位操作数产生一个8位或者16位的结果。目标通用寄存器的高56位或48位在操作中不会被修改。如果一个8位或16位操作数的结果被用作64位地址计算，则会对其进行符号扩展，扩展到64位。

因为64位通用寄存器的高32位在32位模式中没有定义，所以当从64位模式转换到任何一种32位模式（比如传统模式或兼容模式）时，高32位的数据将不被保留。同样，在64位转换到32位模式之后，软件也不必要用这些没有定义的高字节位来存放数据。这些值回从一个硬件实现转换到下一个，或从一个周期转换到下一个。

1. **流SIMD扩展（SSE)寄存器**

在兼容和传统模式下，SSE寄存器组由8个128位的传统寄存器XMM0-XMM7组成。在64位模式，有了8个附加的128位SSE寄存器，XMM8-XMM15。通过使用REX指令前缀访问这些存储器。XMM寄存器能够在任何模式下，在SSE, SSE2, SSE3指令中使用。

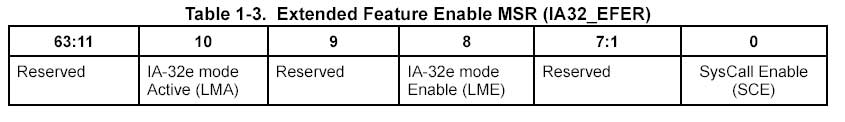
1. **系统寄存器**

64位引入新的寄存器也改变了现有的系统寄存器。他们是：

* **MSRs.**扩展特性允许MSR(IA-32\_EFER)包含控制，允许与禁止64位扩展技术特性的那些位
* **控制寄存器。**所有的控制寄存器扩充到64位，增加了一个新的控制寄存器（任务优先级寄存器CR8或TPR)
* **描述符表寄存器。**全局描述符表寄存器（GDTR）和中断描述符表寄存器(IDTR)被扩展到10字节，以便他们能够包含全部64位地址。局部描述符表寄存器（LDTR）和任务寄存器也别扩展来包含64位地址。
* **调试寄存器。**调试寄存器扩展到64位。

1. **扩展特性允许寄存器(IA-32\_EFER）**

扩展特性语序寄存器（IA-32\_EFER）包含处理器扩展特性控制位。该寄存器在地址C0000080H。下表是该寄存器的各位细节。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e3.jpg)

* **LMA(IA-32e模式激活，位10）：**该位是只读状态位，任何对该位的写入操作都将会被忽略。当IA-32e模式和页式管理被允许后，处理器将该位置1，这表明处理器运行在兼容模式或64位模式，具体在那个模式就要看代码段描述符的L位和D位的值。LMA＝0时，处理器运行在传统模式，在这个模式下，处理器处理器的行为如同标准32位的IA-32处理器。
* **LME（IA-32e模式允许，位8）**：设置该位为1可以使处理器的能力转换到IA-32e模式，但是IA-32e模式并没有真正被激活，只有当软件使能PAE模式进行页式管理。当PAE页式管理被允许，并且LME被设置为1，处理器将设置LMA位为1，这指明IA-32e模式不仅被允许，同时被激活。IA32\_EFER的其他所有保留位必须位0。
* **SCE（Syscall/Sysret允许，位0）**：这位设置为1将支持Syscall/Sysret。Syscall/Sysret只在64为模式下被支持。操作系统负责为64位操作来使能它。

1. **控制寄存器**

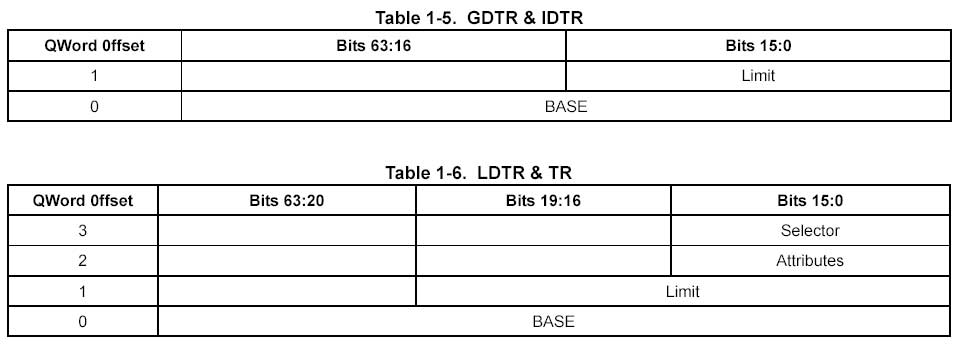
控制机存器CR0-CR4在64位扩展模式下被扩展到64位。在64位模式，MOV CRn指令读或写这些寄存器的全部64位。操作数宽度前缀被忽略。兼容和传统模式，控制寄存器的高32位被全部填0，读控制寄存器也只返回低32位。

在64位模式，CR0和CR4的高32位被保留并且必须被写0。对高32位的任何一位进行写的结果是引起一般性保护异常，＃GP（0）。CR2的所有64位都可通过软件来写。CR3的位[51:40]被保留，必须为0。然而MOV CRn指令不检查写到CR2或CR3的地址是否在线性地址或物理地址的实现界限内。

64为扩展结构引入了一个新控制寄存器—CR8，它被定义为任务优先级寄存器（TPR)。操作系统能够基于中断的优先级别，使用TPR来控制是否允许外部中断来中断处理器。

1. **描述符表寄存器**

四个系统描述符表寄存器(GDTR, IDTR, LDTR和TR) 被扩展到能容下64位基地址。这允许运行在IA-32e模式的操作系统能够将描述符表定位在可用的线性地址空间的任何地方。下表给出了这四个寄存器。在所有的情况下，基地址必须符合范式。线性和物理地址位数能够用执行CPUID通过EAX设置80000008H来决定。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e5.jpg)

1. **调试寄存器**

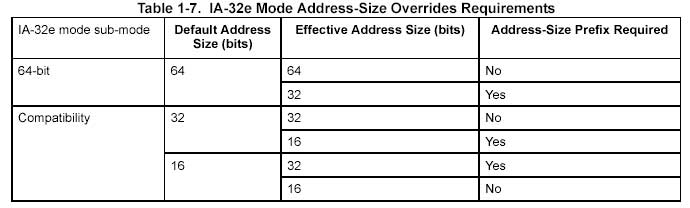
在64位模式下，调试寄存器DR0-DR7是64位的。MOV DRn指令读或写所有的64个寄存器位。操作数宽度前缀被忽略。

在IA-32e平台上所有16位模式或32位模式（传统模式或兼容模式）写调试寄存器时高32位全部填0，读调试寄存器的时候只返回低32位。在64位模式下，DR6和DR7的高32位保留并必须是0，对高32位的任何一位写1都会引发＃GP(0)异常。

DR0-DR3的所有64位都是软件能写的。然而MOV DRn指令不检查写到DR0-DR3的地址在线性地址的限制内。只在处理器产生有效地址的时候支持地址匹配。

1. **指令集变化**
2. **地址宽度和操作数宽度前缀**

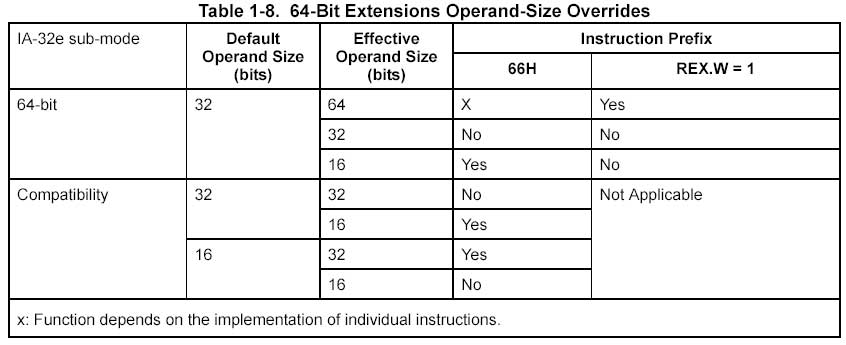
64位模式中，缺省的地址宽度是64位，缺省的操作数宽度是32位。地址宽度和操作数宽度前缀允许32位和64位数据和地址在指令序列中混用。下表（1－7）显示了在IA-32e模式下需要指令前缀地址宽度。注意，在64位模式下不支持16位地址。在敬爱内容和传统模式下，地址宽度函数的功能和在IA-32传动架构中一样。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e7.jpg)

下表(1-8)显示了66H指令前缀和REX.W前缀的有效组合来指定IA-32e操作模式下的操作数宽度问题。

在64位模式下， 缺省的操作数宽度是32位，REX前缀包括4位域来指定16个不同的值。REX前缀的W位域指定为REX.W。REX.W＝1时前缀表明操作数位64为操作数。注意，软件依然能使用操作数宽度66H前缀来切换到16位操作宽度。然而如果同时用REX.W和66H前缀，REX.W的优先权要高。

在SSE/SSE2/SSE3 SIMD指令的情况下，66H, F2H和F3H前缀作为操作码扩展，并被认为是指令的一部分。在这些情况下，有效的REX.W前缀和66H代码扩展前缀之间没有相互关系。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e8.jpg)

1. **REX前缀**

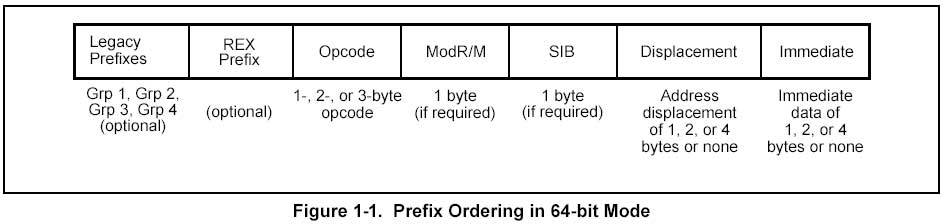
REX前缀是64位模式下引入的新的指令前缀字节，他作以下工作：

* 指定新的GPRs和SSE寄存器
* 指定64位代码宽度
* 指定扩展的控制寄存器（只给系统软件使用）

不是所有的指令都需要REX前缀。这个前缀只在指令引用扩展的寄存器或使用64位操作数的时候才有必要。如果该前缀放在不需要的地方将会被忽略。

一个指令只能有一个REX前缀。这个前缀一旦使用，就必须直接放在操作码字节或两字节操作码扩展前缀之前。 其他位置的REX前缀将被忽略。

包含有REX前缀的指令依然要遵循传统的15字节的指令宽度的限制。下图描述了REX前缀如何符合指令的字节次序的。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32et1.jpg)

1. **控制和调试寄存器的新编码**

在64位模式下，有为控制机存器和调试寄存器指定的附加的编码。当ModRM寄存器的域编码一个控制或调试寄存器的时候，REX.R位被用来修改这些域。这些编码允许处理器访问CR8-CR15和DR8-DR15。

在64位模式中附加了一个控制寄存器（CR8）。CR8成为任务优先级寄存器(TPR)。在IA-32e技术的首次实现的时候，CR9-CR15和DR8-DR15都没有实现，对它们的访问将引起无效代码异常（＃UD）。

1. **新的指令**

下面的新指令在带有64位扩展的64位模式下被引入。

* SWAPGS 指令
* SYSCALL and SYSRET 指令
* CDQE 指令
* CMPSQ 指令
* CMPXCHG16B 指令
* LODSQ 指令
* MOVSQ 指令
* MOVZX(64-bits) 指令
* STOSQ 指令

1. **堆栈指针**

在64位模式，堆栈指针为64位。堆栈大小不是像兼容模式或传统模式中那样靠SS段描述符中的某位来控制，也不通过指令前缀来指示。

对隐式堆栈引用将忽略地址大小的指示。除远分支以外，所有隐式引用RSP的指令在64位模式下缺省为64位操作数。影响到的指令包括：PUSH, POP, PUSHF, POPF, ENTER, 和LEAVE。使用这些指令在64位模式下将不可能产生32位堆栈值的压栈和退栈。如果使用66H操作数前缀，将支持16位的压栈和退栈。

当寄存器RAX－RSP被用作操作数的时候，64位模式缺省的操作尺寸无需REX前缀作为这些指令的先导。如果式R8-R15作为操作数，则REX依然是需要的。这是因为前缀在访问新扩展寄存器中是需要的。

1. **分支转移**

64位扩展技术扩充2个分支机制来适应64位线性地址空间的分支。他们是：

* 64位模式下近分支转移被重新定义
* 在64位模式和兼容模式下，64位调用门描述符定义成远调用

64位模式下，所有近分支转移(CALL, RET, JCC, JCXZ, JMP 和 LOOP)被强迫为64位。这些指令被更新为提供64位的RIP值而无需REX前缀。下面的近转移被有效的操作数宽度所控制：

* 指令指针的宽度的截断
* 由于CALL或RET引起的退栈压栈或退栈的大小
* 由于CALL或RET而引起的堆栈指针增加或减少的大小
* 间接转移操作数大小

在64位模式下，以上的所有操作都被强制为64位而不管操作数前缀（操作数大小的前缀被忽略）。然而相对转移的位移区域依然受到32位的限制；近转移的地址大小没有被强制为64位。

地址大小影响到JCXZ和LOOP中RCX的大小；他们也影响到内存间接转移的地址计算。这样的地址缺省是64位，但是他们可以通过地址宽度前缀转换到32位宽度。

软件会用远转移来改变优先级。传统IA-32结构提供调用门机制来允许软件去从一个优先级转到另一个优先级，尽管调用门也可以不改变优先级而只是做转移。当调用门使用的时候，直接或间接的选择器指针会指向一个门描述符（指令重的便宜被忽略）目的代码段的偏移可以从调用门描述符中获得。IA-32e模式重新定义了32位调用门描述符的类型值，使其成为64位调用门描述符，并扩展64位描述符使其能够容纳64位的偏移。64位模式调用门描述符允许远转移访问有效的线性地址空间的任何地方。这些调用门也控制代码段选择器（CS)，允许转换到特权级和缺省尺寸并作为门转换的结果。

因为通常情况下是指定32位的，唯一在64位模式下指定完全64位绝对RIP的是间接分支转移，由于这个原因，直接远分支转移被从64位模式的指令集中删除了。

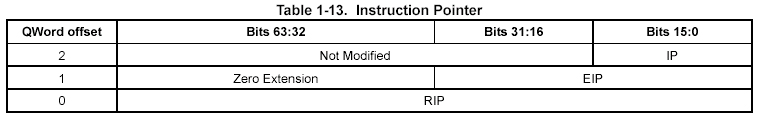
IA-32e模式扩充了SYSENTER和SYSEXIT指令的语义，以便他们操作在64位存储空间。IA-32e也引入了两个新的指令：SYSCALL和SYSRET，他们只在64位模式有效。

1. **存储组织**
2. **64位模式下的地址计算**

在64位模式（如果没有地址大小的转变），有效地址计算的大小是64位的。一个有效地址计算使用一个64位的基和索引寄存器以及符号扩展变换成64位。

对于64位模式下平面地址空间，线性地址等同于有效地址。在使用FS和GS段的非0为基的事务中，这个规则不被使用。在64位模式下，有效地址成分被加进来，并且有效地址在加64位基地址之前被缩短。地址映射模式在64位模式时，基地址从不会被缩短。

在IA-32e模式下，指令指针被扩展到64位来支持64位代码偏移。64位指令指针在调用中将值赋给RIP。下表描述了RIP、EIP和IP之间的不同。

[](http://www.x64asm.com/Technology/IntelEm64T/ia32e13.jpg)

    通常，替换和直接在64位模式下不被扩展到64位。他们在有效地址计算中依然被限制在32位和符号扩展。然而，在64位模式提供了MOV指令的64位替换和直接形式的支持。  
    所有的在IA-32e模式下的16位和32位地址计算用0扩展来形成64位地址。地址计算搜现是缩短到当前模式的有效地址宽度，就像地址宽度前缀的指定那样。其结果是用0扩展得到完全的64位地址宽度。因为这个，16位和32位应用程序运行在兼容模式只能存取64位模式有效地址的低4GB。同样，在64位模式产生一个32位地址只能访问64位模式有效地址的低4GB。

1. **规范的寻址**

一个规范形式的地址有地址位63直到更有效的实现位，宏结构设置其为全1或全0。

IA-32e模式定义一个64位的线性地址，但实现的时候支持的位数要少些。第一个具有64位扩展技术的IA-32e结构的处理器将支持48位线性地址。这意味着规范的地址必须将位63到位48全填0或全填1，填0还是填1要看位47是0还是1。

尽管实现并不用先行地址的全部64位，他们需要检查位63知道更有效的实现位来看是否地址是规范形式。如果一个线性存储引用不是规范形式，该实现将会产生一个异常。在很多情况下，会产生一个一般保护异常（＃GP）。然而，在显示或隐式对战应用的情况下，会产生一个堆栈错（＃SS）。隐式堆栈引用指令包括PUSH/POP指令和使用RSP/RBP寄存器来作为缺省堆栈段寄存器的指令。在这些情况下，一个规范错误式＃SF，如果一个指令使用RSP/RBP作为基寄存器并且有段超越给出一个非SS段，将引起一个一般保护错误（＃GP)的规范错误。隐式堆栈引用包括所有PUSH/POP类型指令和任何使用RSP或RBP作为一个基寄存器。规范地址形式的检查将在特权检查之后页面和边界检查之前完成。