misterliwei的专栏

莫等闲，白了少年头，空悲切。

个人资料

misterliwei

访问：441765次
积分：5728
等级：BLOG 5
排名：第3377名

原创：88篇
转载：2篇

保护模式、实地址模式及V8086模式下的指令格式（上）

标签：express extension reference 扩展 branch signal

2010-05-02 18:14 2128人阅读 评论(0) 收藏 举报

保护模式、实地址模式及V8086模式下的指令格式（上）

本文译自《intel指令手册2.1节》,该手册地址为：
http://www.intel.com/products/processor/manuals/index.htm

Intel64和IA32架构的指令编码格式是图2-1格式的子集。指令有这几个部分组成：可选的指令前缀（顺序不限）、主操作码字节（最多3个字节）以及寻址方式说明（addressing-form specifier）字节（如果需要的话）。寻址方式说明由ModR/M字节、SIB字节、位移量（如果需要的话）和立即数（如果需要的话）组成。

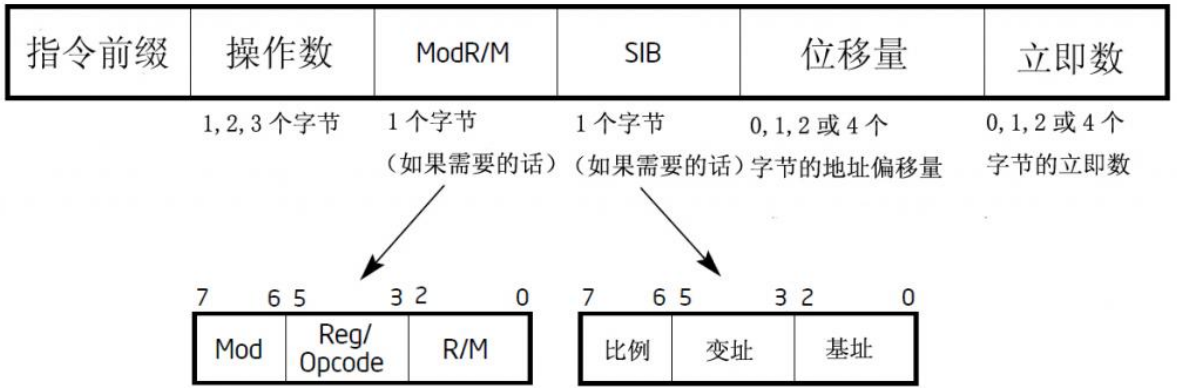

图2-1Intel64和IA-32架构指令格式

2.1.1 指令前缀

指令前缀分成四块，每块都有一组前缀代码。对于每个指令来说，四块中每块只能使用一个前缀。这四块的前缀的顺序是不限定的。

•第一块

—锁前缀和重复前缀：

•LOCK锁前缀代码为F0H

•REPNE/REPNZ前缀代码为F2H。该前缀仅用于字符串和输入/输出指令(F2H在某些指令中还被用作强制前缀(mandatory prefix)).

• REP或者REPE/REPZ前缀代码为F3H。该前缀仅用于字符串和输入/输出指令(F3H在某些指令中还被用作强制前缀).

•第二块

—段超越前缀

• 2EH——CS段超越（在分支指令中使用未定义）

• 36H——SS段超越（在分支指令中使用未定义）

• 3EH——DS段超越（在分支指令中使用未定义）
• 26H——ES段超越（在分支指令中使用未定义）
• 64H——FS段超越（在分支指令中使用未定义）
• 65H——GS段超越（在分支指令中使用未定义）
—分支预测（branch hints）前缀
• 2EH——分支被使用(只用于条件转移JCC指令)
• 3EH——分支不被使用(只用于条件转移JCC指令)

• 第三块
•调整操作数大小(operand-size override)前缀代码为66H(在一些指令中66H被用作强制前缀)

• 第四块
•67H——调整地址大小(address-size override)前缀

锁前缀（F0H）在多处理器情况下会迫使一个操作必须互斥地使用共享内存。（详见《“Instruction Set Reference, A-M,”》第三章“LOCK—Assert LOCK# Signal Prefix”）

重复前缀（F2H,F3H）会使指令对字符串中的每一个元素都重复执行。该前缀只在字符串和I/O指令(MOVS,CMPS,SCAS,LODS,STOS,INS,OUTS)中使用。在其他未定义的操作码前使用重复前缀没有定义，这种使用会导致不可预见的后果。
一些指令会将F2H/F3H作为强制前缀来表示不同的功能（express distinct functionality）。强制前缀通常应置于其他可选前缀后面（例外情况在2.2.1“REX Prefixes”中讨论）。

分支预测前缀（2EH,3EH）允许程序预测一个分支最有可能的路径。此前缀只在条件转移指令（JCC）中使用。在其他未定义的操作码前使用分支预测前缀没有定义，这种使用会导致不可预见的后果。

方案业务连接透明化（中）

最新评论

IoAllocateDriverObjectExtension
misterliwei: @xzyee:是的。然后根据反编译写的伪代码。

IoAllocateDriverObjectExtension
xzyee: 好！这代码是从反编译码得到的？

WINDOWS内核对象
misterliwei: @mr_dong110:可以，保留作者信息就行。

WINDOWS内核对象
mr_dong110: 博主，你写的文章内容 很好 我可以 转载吗？

数据库读写分离和垂直分库、水平
misterliwei: @u014665416:一个从表中读取

有方法读取一个已被其他进程打开
changshenglugu: -_-|||马了再说

创建、分离、重新附加并修复一个
changshenglugu: 看不懂怎么办，先收藏了

有方法读取一个已被其他进程打开
LiAuH: 感谢楼主，顺便mark

MS SQL SERVER 2005 MDF文件
u010548682: @tianyi_iflytek:请问这个后来是怎么解决的，我也是用了这个程序修改之后就提示不是主文件...

GetWindowText函数的一个注意点
D_T: 正在找没取出的原因 好评

调整操作数大小前缀（66H）允许程序在16位和32位操作数之间切换操作数。这两种大小都可以作为缺省值，使用本前缀可以切换为非缺省的那个大小值。

一些SSE2/SSE3/SSSE3/SSE4指令和主操作码为3字节的指令可能会将66H作为强制前缀来表示不同的功能。强制前缀通常应置于其他可选前缀后面（例外情况在2.2.1“REX Prefixes”中讨论）。
66H前缀的其他使用保留，如果使用可能会导致不可预见的后果。

调整地址大小前缀（67H）允许程序在16位和32位地址之间切换。这两种大小都可以作为缺省值，使用本前缀可以切换为非缺省的那个大小值。当指令的操作数是内存中不存在时，使用该前缀和/或者未定义的操作码是未定义的，这种使用会导致不可预见的后果。

.

2.1.2 操作码

主操作码可以是1个,2个或3个字节长。有时会有另外的3位操作码存在于ModR/M字节中。主操作码中还可以为定义更小的段(fields)，比如操作方向、位移量（displacements）大小、寄存器编码、条件码(condition codes)或者符号扩展(sign extension)。操作码中的段随操作的类型不同而有很大的不同。

通用指令和SIMD指令的2个字节的操作码格式组成如下：

•一个逃逸码(escape opcode)0FH作为主操作码和一个第二操作码字节。或者

•一个强制前缀（66H、F2H或F3H），一个逃逸码字节和一个第二操作码字节。

举例说明,CVTDQ2PD的操作码序列为F3 0F E6。第一个字节就是强制前缀（不是重复前缀）。

通用指令和SIMD指令的3个字节的操作码格式组成如下：

•一个逃逸码0FH作为主操作码，加上两个额外的操作码字节。或者

•一个强制前缀（66H、F2H或F3H），一个逃逸码字节，加上两个额外的操作码字节。

举例说明，XMM寄存器的PHADDW指令的操作码序列为66 0F 38 01。第一个字节就是强制前缀。

所有有效的操作码列于附录A和附录B.

2.1.3 ModR/M和SIB字节

很多引用内存操作数的指令需要一个说明寻址方式的字节（称为ModR/M）紧跟在主操作码后面。ModR/M字节包含下面三个信息段（译注：见图2-1）：

•Mod段和r/m段形成32种可能值：8种寄存器和24种寻址方式。

•Reg/Opcode段或者指定一个寄存器或者是主操作码的3位附加信息。Reg/opcode段的具体作用由主操作码决定。

•r/m段可以指定一个寄存器作为操作数，或者和mod段组合起来指定一种编码方式。有时候，这种mod段和r/m段的组合可以被用来表示一些指令的操作码信息。

一些ModR/M字节还需要第二个寻址字节（SIB）。32位的基址变址（base-plus-index）和比例变址(scale-plus-index)寻址方式需要一个SIB字节。SIB字节包含下面段（译注：见图2-1）：

•比例段指出比例倍数(scale factor)

•变址段指出变址寄存器的寄存器数字

•基址段指出基址寄存器的寄存器数字

参看2.1.5节ModR/M和SIB字节的编码

2.1.4位移量和立即数字节

一些寻址方式在ModR/M(或者SIB，如果有的话)后面会有一个位移量。如果有位移量的话，它可以是1、2或4字节。如果一个指令指定一个立即数，那么这个立即数会跟在位移量字节后面，立即数可以是1、2和4字节。

2.1.5 ModR/M和SIB字节组成的寻址方式

由ModR/M和SIB字节组成的值和其对应的寻址方式列于表2-1到表2-3：由ModR/M字节指定的16位寻址方式列于表2-1；32位寻址方式列于表2-2；表2-3列出了由SIB字节指定的32位寻址方式。假如ModR/M字节中reg/opcode段表示一个扩展操作码，有效编码见附录B.

在表2-1和表2-2中，有效地址（Effective Address）列中列出了由Mod和R/M组成的可以赋给指令的第一个操作数的32种有效地址。前面24种指出了一个内存的位置；后面的8种（Mod=11B）是指出了通用寄存器、MMX寄存器和XMM寄存器。

表2-1和表2-2中的Mod和R/M列给出了要获得第一列这种有效地址方式的二进制编码。比如，看Mod=11B，R/M=000B那一行，该行标明的通用寄存器为EAX,AX和AL;MMX寄存器为MM0;XMM寄存器为XMM0.具体到底使用哪个寄存器是由操作码字节和操作数大小属性决定。

我们再来看表2-1或2-2任一个表的第7行（标有“REG=”的那行）。该行指出如何使用3位的Reg/Opcode段来获得第二个操作数的位置。第二个操作数必须是通用寄存器、MMX寄存器或者XMM寄存器。第1到5行列出了表中值所对应的寄存器。和上面一样，使用哪种寄存器是由操作码字节和操作数大小属性决定的。

如果指令不需要第二个操作数，Reg/Opcode段可以被用作操作码的扩展。由表的第6行（标有“/DIGIT(OPCODE)”的那行）标出。注意该行值是十进制形式的。

表2-1和表2-2的主体部分（在标有“Value of ModR/M Byte (in Hexadecimal)”的下面）包含一个由32*8数组所表示的256个ModR/M值（16进制）。第3/4/5位由字节所在列指出，第0/1/2位和第6/7位由行指出。下图演示如何计算出表中值的。

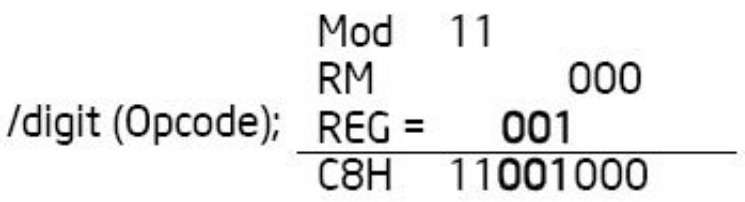

图2-2 解释如何得到ModR/M(C8H)字节

表 2-1 十六位寻址方式 (ModR/M)

r8(/r) r16(/r) r32(/r) mm(/r) xmm(/r) (In decimal) /digit (Opcode) (In binary) REG =			AL AX EAX MM0 XMM0 0 000	CL CX ECX MM1 XMM1 1 001	DL DX EDX MM2 XMM2 2 010	BL BX EBX MM3 XMM3 3 011	AH SP ESP MM4 XMM4 4 100	CH BP1 EBP MM5 XMM5 5 101	DH SI ESI MM6 XMM6 6 110	BH DI EDI MM7 XMM7 7 111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
[BX+SI]	00	000	00	08	10	18	20	28	30	38
[BX+DI]		001	01	09	11	19	21	29	31	39
[BP+SI]		010	02	0A	12	1A	22	2A	32	3A
[BP+DI]		011	03	0B	13	1B	23	2B	33	3B
[SI]		100	04	0C	14	1C	24	2C	34	3C
[DI]		101	05	0D	15	1D	25	2D	35	3D
disp16[2]		110	06	0E	16	1E	26	2E	36	3E
[BX]		111	07	0F	17	1F	27	2F	37	3F
[BX+SI]+disp8[3]	01	000	40	48	50	58	60	68	70	78
[BX+DI]+disp8		001	41	49	51	59	61	69	71	79
[BP+SI]+disp8		010	42	4A	52	5A	62	6A	72	7A
[BP+DI]+disp8		011	43	4B	53	5B	63	6B	73	7B
[SI]+disp8		100	44	4C	54	5C	64	6C	74	7C
[DI]+disp8		101	45	4D	55	5D	65	6D	75	7D
[BP]+disp8		110	46	4E	56	5E	66	6E	76	7E
[BX]+disp8		111	47	4F	57	5F	67	6F	77	7F
[BX+SI]+disp16	10	000	80	88	90	98	A0	A8	B0	B8
[BX+DI]+disp16		001	81	89	91	99	A1	A9	B1	B9

[BP+SI]+disp16		010	82	8A	92	9A	A2	AA	B2	BA
[BP+DI]+disp16		011	83	8B	93	9B	A3	AB	B3	BB
[SI]+disp16		100	84	8C	94	9C	A4	AC	B4	BC
[DI]+disp16		101	85	8D	95	9D	A5	AD	B5	BD
[BP]+disp16		110	86	8E	96	9E	A6	AE	B6	BE
[BX]+disp16		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM1/XMM1		001	C1	C9	D1	D9	EQ	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AHMM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

1. 包含 BP 基址的有效地址取缺省段寄存器为 SS;其他的有效地址的缺省寄存器为 DS.
2. disp16 表示在 ModR/M 字节后面有一个 16 位的偏移值，这个值是要加到变址的。
3. disp8 表示在 ModR/M 字节后面有一个 8 位的偏移值，这个值是有符号扩展的，并要加到变址的。

表 2-2 32 位寻址方式 (ModR/M)

r8(/r) r16(/r) r32(/r) mm(/r) xmm(/r) (In decimal) /digit (Opcode) (In binary) REG =			AL AX EAX MM0 XMM0 0 000	CL CX ECX MM1 XMM1 1 001	DL DX EDX MM2 XMM2 2 010	BL BX EBX MM3 XMM3 3 011	AH SP ESP MM4 XMM4 4 100	CH BP EBP MM5 XMM5 5 101	DH SI ESI MM6 XMM6 6 110	BH DI EDI MM7 XMM7 7 111
Effective Address	**Mod**	**R/M**	**Value of ModR/M Byte (in Hexadecimal)**							
[EAX]	00	000	00	08	10	18	20	28	30	38
[ECX]		001	01	09	11	19	21	29	31	39
[EDX]		010	02	0A	12	1A	22	2A	32	3A
[EBX]		011	03	0B	13	1B	23	2B	33	3B
[--][--][1]		100	04	0C	14	1C	24	2C	34	3C
disp32[2]		101	05	0D	15	1D	25	2D	35	3D
[ESI]		110	06	0E	16	1E	26	2E	36	3E
[EDI]		111	07	0F	17	1F	27	2F	37	3F
[EAX]+disp8[3]	01	000	40	48	50	58	60	68	70	78
[ECX]+disp8		001	41	49	51	59	61	69	71	79

[EDX]+disp8		010	42	4A	53	5A	62	6A	72	7A
[EBX]+disp8		011	43	4B	54	5B	63	6B	73	7B
[--][--]+disp8		100	44	4C	55	5C	64	6C	74	7C
[EBP]+disp8		101	45	4D	56	5D	65	6D	75	7D
[ESI]+disp8		110	46	4E	57	5E	66	6E	76	7E
[EDI]+disp8		111	47	4F		5F	67	6F	77	7F
[EAX]+disp32	10	000	80	88	90	98	A0	A8	B0	B8
[ECX]+disp32		001	81	89	91	99	A1	A9	B1	B9
[EDX]+disp32		010	82	8A	92	9A	A2	AA	B2	BA
[EBX]+disp32		011	83	8B	93	9B	A3	AB	B3	BB
[--][--]+disp32		100	84	8C	94	9C	A4	AC	B4	BC
[EBP]+disp32		101	85	8D	95	9D	A5	AD	B5	BD
[ESI]+disp32		110	86	8E	96	9E	A6	AE	B6	BE
[EDI]+disp32		111	87	8F	97	9F	A7	AF	B7	BF
EAX/AX/AL/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
ECX/CX/CL/MM/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
EDX/DX/DL/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
EBX/BX/BL/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
ESP/SP/AH/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
EBP/BP/CH/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
ESI/SI/DH/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
EDI/DI/BH/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

1. [—][—]表示在 ModR/M 字节后面有一个 SIB。
2. disp32 表示在 ModR/M 字节（或者 SIB, 如果有的）后面有一个 32 位的偏移值，这个值是要加到变址的。
3. disp8 表示在 ModR/M 字节（或者 SIB, 如果有的）后面有一个 8 位的偏移值，这个值是有符号扩展的，并要加到变址的。

表2-3显示的是256中SIB字节的可能值(16进制)。用作基址的通用寄存器列在表的上部，并同时显示了在SIB字节中的对应值。表的主体部分的行是作为变址的寄存器和比例倍数(由SIB字节的第6/7位表示).

表 2-3 32 位寻址方式 (SIB)

r32 (In decimal) Base = (In binary) Base =			EAX 0 000	ECX 1 001	EDX 2 010	EBX 3 011	ESP 4 100	[*] 5 101	ESI 6 110	EDI 7 111
Scaled Index	**SS**	**Index**	**Value of SIB Byte (in Hexadecimal)**							
[EAX]	00	000	00	01	02	03	04	05	06	07

[ECX]		001	08	09	0A	0B	0C	0D	0E	0F
[EDX]		010	10	11	12	13	14	15	16	17
[EBX]		011	18	19	1A	1B	1C	1D	1E	1F
none		100	20	21	22	23	24	25	26	27
[EBP]		101	28	29	2A	2B	2C	2D	2E	2F
[ESI]		110	30	31	32	33	34	35	36	37
[EDI]		111	38	39	3A	3B	3C	3D	3E	3F
[EAX*2]	01	000	40	41	42	43	44	45	46	47
[ECX*2]		001	48	49	4A	4B	4C	4D	4E	4F
[EDX*2]		010	50	51	52	53	54	55	56	57
[EBX*2]		011	58	59	5A	5B	5C	5D	5E	5F
none		100	60	61	62	63	64	65	66	67
[EBP*2]		101	68	69	6A	6B	6C	6D	6E	6F
[ESI*2]		110	70	71	72	73	74	75	76	77
[EDI*2]		111	78	79	7A	7B	7C	7D	7E	7F
[EAX*4]	10	000	80	81	82	83	84	85	86	87
[ECX*4]		001	88	89	8A	8B	8C	8D	8E	8F
[EDX*4]		010	90	91	92	93	94	95	96	97
[EBX*4]		011	98	89	9A	9B	9C	9D	9E	9F
none		100	A0	A1	A2	A3	A4	A5	A6	A7
[EBP*4]		101	A8	A9	AA	AB	AC	AD	AE	AF
[ESI*4]		110	B0	B1	B2	B3	B4	B5	B6	B7
[EDI*4]		111	B8	B9	BA	BB	BC	BD	BE	BF
[EAX*8]	11	000	C0	C1	C2	C3	C4	C5	C6	C7
[ECX*8]		001	C8	C9	CA	CB	CC	CD	CE	CF
[EDX*8]		010	D0	D1	D2	D3	D4	D5	D6	D7
[EBX*8]		011	D8	D9	DA	DB	DC	DD	DE	DF
none		100	E0	E1	E2	E3	E4	E5	E6	E7
[EBP*8]		101	E8	E9	EA	EB	EC	ED	EE	EF
[ESI*8]		110	F0	F1	F2	F3	F4	F5	F6	F7
[EDI*8]		111	F8	F9	FA	FB	FC	FD	FE	FF

1. [*]表示如果 MOD 是 00B 时，disp32 是没有基址的；不为 00B 时，[*]表示 disp8 或者 disp32+[EBP]。总结如下：

MOD 位	有效地址
00	[scaled index] + disp32
01	[scaled index] + disp8 + [EBP]
10	[scaled index] + disp32 + [EBP]

参考文献

1. http://linux.chinaunix.net/bbs/thread-1050480-1-1.html。
2. http://blog.ftofficer.com/2010/04/n-forms-of-call-instructions/
3. http://baike.baidu.com/view/889427.htm?fr=ala0_1

上一篇 SQL SERVER 数值类型的存储格式及转换
下一篇 保护模式、实地址模式及V8086模式下的指令格式（下）

猜你在找

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery
BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity
Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack
FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide
Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase
Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap