VMP学习笔记之X86指令之Opcode快速入门（二）

参考资料：

本文大量内容抄袭看雪作者：waiWH的VMP系列

1、名称：**X86指令内幕 —— 序**

网址：https://blog.csdn.net/xfcyhuang/article/details/6228024

2、名称：**谈谈vmp的还原(1)**

网址：https://bbs.pediy.com/thread-225278.htm

学习目的：

1、VMP代码自带反汇编引擎，需要一定的Opcode指令基础。

正文：

1、快速入门Opcode

Intel-64和IA-32架构指令编码是图2-1所示格式的子集.一条指令包括可选的指令前缀(顺序任意),主操作码(最多3字节),由ModR/M和SIB字节(可选) 组成的地址格式描述符(如果需要的话),偏移量(可选)以及立即数(可选).

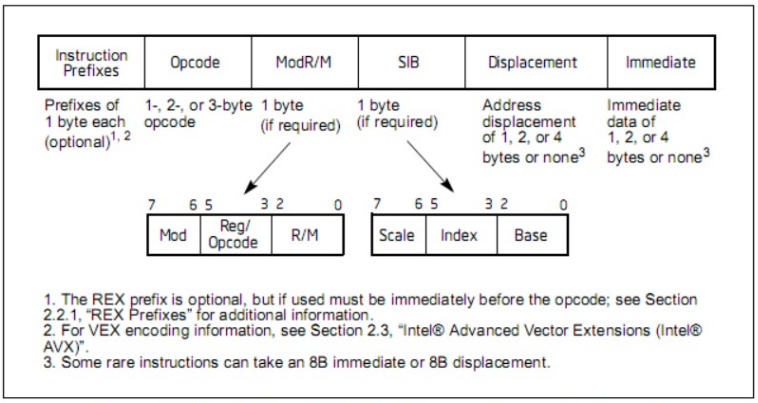


Figure 2-1. Intel 64 and IA-32 Architectures Instruction Format

总结：

1、根据上图所示我们知道一条汇编指令分为以下几个部分：

1、Legacy Prefix（可选）
2、REX prefix（用于64位模式，这里不做讲解）
3、Opcode（必须有）
4、ModRM（可选）
5、SIB（可选）
6、Displacement（可选）
7、Immediate（可选）

2、Opcode结构顺序是不能被打乱的

2、 指令前缀讲解（可选）

1、基础理论知识

指令前缀分为四组,每一组包含一些允许的前缀码.对于任何指令,前缀可以从这四组(组1,2,3,4)里的挑选,并且它们不区分次序.

- 组1

— 锁定和重复前缀:

- F0H - LOCK
- F2H - REPNE/REPNZ,仅用于串操作和I/O指令,也可被用作某些指令的强制性前缀
- F3H - REP或REPE/REPZ,仅用于串操作和I/O指令,也可被用作某些指令的强制性前缀
- 组2

— 段重载前缀:

- 2EH—CS 段重载(用于任意分支指令时保留)
- 36H—SS 段重载(用于任意分支指令时保留)
- 3EH—DS 段重载(用于任意分支指令时保留)
- 26H—ES 段重载(用于任意分支指令时保留)
- 64H—FS 段重载(用于任意分支指令时保留)
- 65H—GS 段重载(用于任意分支指令时保留)

— 分支提示:

- 2EH—分支不被接受(仅用于Jcc指令中)
- 3EH—分支被接受(仅用于Jcc指令中)
- 组3
- 66H—操作数大小重载前缀,也可被用作某些指令的强制性前缀.
- 组4

- 67H—地址尺寸重载前缀

LOCK前缀(F0H)在多处理器环境下强制执行独占共享内存操作.详见《Instruction Set Reference, A-M》第三章"LOCK – 断言LOCK#信号前缀".

重复前缀(F2H,F3H)将会重复操作字符串的每一个元素.只有MOVS,CMPS,SCAS,LODS,STOS,INS,OUTS等字符串操作或I/O指令才能使用这些前缀. 对Intel 64 或 IA-32 其他指令使用重复前缀和/或未定义的操作码是被保留的,将会引起不可预知的行为.

某些指令可能使用F2H,F3H作为强制性前缀来表示特定的功能.强制性前缀应当位于其他可选的前缀之后(例外的情形请查看第2.2.1节,"REX前缀")

分支提示前缀(2EH,3EH)允许程序给处理器一个最有可能的执行分支提示.这些前缀只能用于条件指令(Jcc).在Intel 64 或 IA-32 其他指令中使用分支预测前缀或者未定义的操作码是被保留的,将引起不可预知的行为.

操作数大小重载前缀允许程序在16位和32位操作数大小间切换.它们中任一个都可以是默认值,而使用这个前缀则选择非默认值.

某些SSE2/SSE3/SSSE3/SSE4和使用3字节操作码的指令可能使用66H作为强制性前缀来表示特定的功能. 强制性前缀应当位于其他可选的前缀之后(例外的情形请查看第2.2.1节,"REX前缀") . 66H前缀的其他用法是被保留的, 将引起不可预知的行为.

地址尺寸重载前缀(67H)允许程序在16位和32位地址间切换.它们中的任何一个都可以是默认值,使用这个前缀选择非默认值.当指令中的操作数不在内存中,使用这个前缀或未定义的操作码时,操作被保留,可能引起不可预知的行为.

3、 OPcode详解（必备）

主操作码，必须有。指令不定长1、2、3都有，例如8B C0　mov eax,eax 其中8B就是主操作码。例如90 NOP就是只有主操作码没有ModR/M和SIB字节的。

4、ModR/M 和 SIB 字节（可选）

1、基础知识（非常重要的字节）

许多涉及内存操作数的指令都有一个紧挨着主操作码的寻址格式说明字节(叫做ModR/M字节),ModR/M字节包含3个域信息:

- mod域与r/m域组成32个可能的值:8个寄存器和24个寻址模式.
- reg/opcode域确定寄存器号或者附加的3位操作码.reg/opcode域的用途由主操作码确定.
- r/m域确定一个寄存器为操作数或者和mod域一起编码寻址模式.有时候有些指令使用特定的mod域和r/m域组合来表示操作码信息.

某些ModR/M字节编码需要第二寻址字节(SIB).基址+索引或者比例+索引形式的32位寻址需要SIB字节.SIB字节包括下列域:

- scale 域指定比例因子.
- index域指定索引寄存器号.
- base 域指定基址寄存器号.

ModR/M和SIB编码详见第2.1.5节.

总结：

1、SIB就是对ModR/M的补充辅助。这些都是可选的

5、 针对前面内容进行实践

0、 参考书籍

1、查ModRM、SIB表使用的是：参考文档：Intel开发者手册 第二卷 指令集手册 第2章

2、查OPcode使用的是：参考文档：IntelCPU机器指令中文版手册

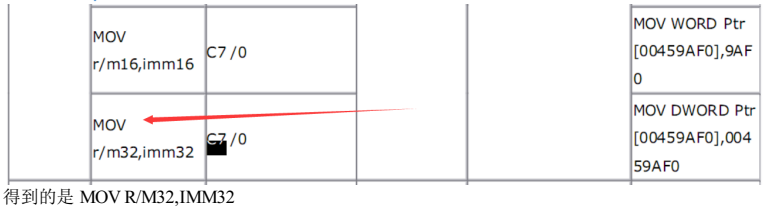
1、举例说明：

26:C784C8 44332211 78563412　MOV DWORD PTR ES:[EAX+ECX*8+0x11223344],0x12345678

2、指令拆解：

1、Legacy Prefix（可选）	26H—ES 段重载(用于任意分支指令时保留)
2、Opcode（必须有）	C7
3、ModRM（可选）	84
4、SIB（可选）	C8
5、Displacement（可选）	44332211
6、Immediate（可选）	78563412

3、解析Opcode找到主操作码（查表）



得到的是 MOV R/M32,IMM32

4、分析ModRM（查表）

ModRM结构如下：

--	位	描述
ModRM.mod	[7:6]	提供寻址模式: 11 = register !11 = memory
ModRM.reg	[5:3]	提供 register 或者对 Opcode 进行补充
ModRM.r/m	[2:0]	提供 register 或者 memory 依赖于 ModRM.mod

转换成二进制如下：

84=10 000 100

结构	描述	内容
ModRM.mod	寻址模式	10
ModRM.reg	寄存器	000
ModRM.r/m	寄存器或则地址	100

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] [ECX] [EDX] [EBX] [--][--][1] disp32[2] [ESI] [EDI]	00	000 001 010 011 100 101 110 111	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	10 11 12 13 14 15 16 17	18 19 1A 1B 1C 1D 1E 1F	20 21 22 23 24 25 26 27	28 29 2A 2B 2C 2D 2E 2F	30 31 32 33 34 35 36 37	38 39 3A 3B 3C 3D 3E 3F
[EAX]+disp8[3] [ECX]+disp8 [EDX]+disp8 [EBX]+disp8 [--][--]+disp8 [EBP]+disp8 [ESI]+disp8 [EDI]+disp8	01	000 001 010 011 100 101 110 111	40 41 42 43 44 45 46 47	48 49 4A 4B 4C 4D 4E 4F	50 51 52 53 54 55 56 57	58 59 5A 5B 5C 5D 5E 5F	60 61 62 63 64 65 66 67	68 69 6A 6B 6C 6D 6E 6F	70 71 72 73 74 75 76 77	78 79 7A 7B 7C 7D 7E 7F
[EAX]+disp32 [ECX]+disp32 [EDX]+disp32 [EBX]+disp32 [--][--]+disp32 [EBP]+disp32 [ESI]+disp32 [EDI]+disp32	10	000 001 010 011 100 101 110 111	80 81 82 83 84 85 86 87	88 89 8A 8B 8C 8D 8E 8F	90 91 92 93 94 95 96 97	98 99 9A 9B 9C 9D 9E 9F	A0 A1 A2 A3 A4 A5 A6 A7	A8 A9 AA AB AC AD AE AF	B0 B1 B2 B3 B4 B5 B6 B7	B8 B9 BA BB BC BD BE BF
EAX/AX/AL/MM0/XMM0 ECX/CX/CL/MM/XMM1 EDX/DX/DL/MM2/XMM2 EBX/BX/BL/MM3/XMM3 ESP/SP/AH/MM4/XMM4 EBP/BP/CH/MM5/XMM5 ESI/SI/DH/MM6/XMM6 EDI/DI/BH/MM7/XMM7	11	000 001 010 011 100 101 110 111	C0 C1 C2 C3 C4 C5 C6 C7	C8 C9 CA CB CC CD CE CF	D0 D1 D2 D3 D4 D5 D6 D7	D8 D9 DA DB DC DD DE DF	E0 E1 E2 E3 E4 E5 E6 E7	E8 E9 EA EB EC ED EE EF	F0 F1 F2 F3 F4 F5 F6 F7	F8 F9 FA FB FC FD FE FF

注:

1. “[--][--]”记号表示ModR/M 后跟随有一个SIB字节.
2. “disp32”记号表示ModR/M(或者SIB,如果出现的话) 后跟随一个32位的偏移量,该偏移量被加至有效地址.
3. “disp8” 记号表示ModR/M(或者SIB,如果出现的话) 后跟随一个8位的偏移量,该偏移量将被符号扩展,然后被加至有效地址.

表2-3囊括了SIB 的256个可能值(十六进制形式) . 可以作为基的通用寄存器通过表的上部列出,也列出了相应的base域值. 表的主体的每行列出了索引(index SIB的3,4,5位)对应的寄存器及倍率因子(scaling factor SIBbyte的6,7位).

总结：

1、ModRM.mod 提供寻址模式: 11 = register（寄存器） 11 != memory（地址）

2、R/M = 100并且mod！=11 的时候表示存在SIB表

5、分析SIB（查表）

根据上文找到的地址发现是[--][--]”,表示有SIB表

SIB结构如下：

--	位	描述
SIB.scale	[7:6]	提供 scale: 00 = 1, 01 = 2, 10 = 4, 11 = 8
SIB.index	[5:3]	提供 index 寄存器
SIB.base	[2:0]	提供 base 寄存器

转换成2进制如下：

C8=11 001 000

结构	描述	内容
SIB.scale	提供 index 寄存器乘数因子 scale	11
SIB.index	提供 index 寄存器寻址	001
SIB.base	提供 base 寄存器寻址	000

Table 2-3. 32-Bit Addressing Forms with the SIB Byte

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	99	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

注:

1. “[*]”记号表示:若MOD = 00B表示没有基,且带有一个32位的偏移量；否则表示disp8或disp32 + [EBP].即提供如下的寻址方式:

MOD　　有效地址

00　[scaled index] + disp32

01　[scaled index] + disp8 + [EBP]

10　[scaled index] + disp32 + [EBP]

44332211 - Displacement：此为小端模式，即为 0x11223344

78563412 - Immediate：小端模式，即为 0x12345678

综上，得到：

MOV DWORD PTR ES:[EAX+ECX*8+0x11223344],0x12345678