



CHAPTER 2 INSTRUCTION FORMAT

This chapter describes the instruction format for all IA-32 processors.

2.1. GENERAL INSTRUCTION FORMAT

All IA-32 instruction encodings are subsets of the general instruction format shown in Figure 2-1. Instructions consist of optional instruction prefixes (in any order), one or two primary opcode bytes, an addressing-form specifier (if required) consisting of the ModR/M byte and sometimes the SIB (Scale-Index-Base) byte, a displacement (if required), and an immediate data field (if required).

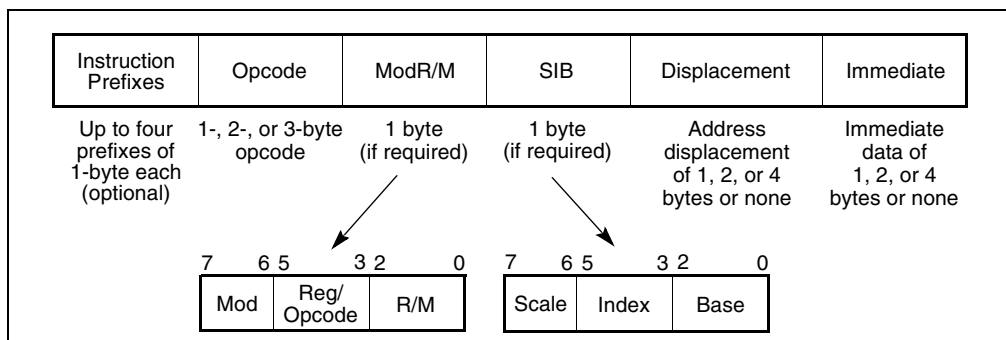


Figure 2-1. IA-32 Instruction Format

2.2. INSTRUCTION PREFIXES

The instruction prefixes are divided into four groups, each with a set of allowable prefix codes:

- Group 1
 - Lock and repeat prefixes:
 - F0H—LOCK.
 - F2H—REPNE/REPZ (used only with string instructions).
 - F3H—REP or REPE/REPZ (use only with string instructions).
- Group 2
 - Segment override prefixes:
 - 2EH—CS segment override (use with any branch instruction is reserved).

- 36H—SS segment override prefix (use with any branch instruction is reserved).
- 3EH—DS segment override prefix (use with any branch instruction is reserved).
- 26H—ES segment override prefix (use with any branch instruction is reserved).
- 64H—FS segment override prefix (use with any branch instruction is reserved).
- 65H—GS segment override prefix (use with any branch instruction is reserved).
- Branch hints:
 - 2EH—Branch not taken (used only with *Jcc* instructions).
 - 3EH—Branch taken (used only with *Jcc* instructions).
- Group 3
 - 66H—Operand-size override prefix.
- Group 4
 - 67H—Address-size override prefix.

For each instruction, one prefix may be used from each of these groups and be placed in any order. Using redundant prefixes (more than one prefix from a group) is reserved and may cause unpredictable behavior.

The LOCK prefix forces an atomic operation to insure exclusive use of shared memory in a multiprocessor environment. See “LOCK—Assert LOCK# Signal Prefix” in Chapter 3, *Instruction Set Reference*, for a detailed description of this prefix and the instructions with which it can be used.

The repeat prefixes cause an instruction to be repeated for each element of a string. They can be used only with the string instructions: MOVSB, CMPSB, SCASB, LODSB, STOSB, INSB, and OUTSB. Use of the repeat prefixes with other IA-32 instructions is reserved and may cause unpredictable behavior (see the note below).

The branch hint prefixes allow a program to give a hint to the processor about the most likely code path that will be taken at a branch. These prefixes can only be used with the conditional branch instructions (*Jcc*). Use of these prefixes with other IA-32 instructions is reserved and may cause unpredictable behavior. The branch hint prefixes were introduced in the Pentium 4 and Intel Xeon processors as part of the SSE2 extensions.

The operand-size override prefix allows a program to switch between 16- and 32-bit operand sizes. Either operand size can be the default. This prefix selects the non-default size. Use of this prefix with MMX, SSE, and/or SSE2 instructions is reserved and may cause unpredictable behavior (see the note below).

The address-size override prefix allows a program to switch between 16- and 32-bit addressing. Either address size can be the default. This prefix selects the non-default size. Using this prefix when the operands for an instruction do not reside in memory is reserved and may cause unpredictable behavior.

NOTE

Some of the SSE and SSE2 instructions have three-byte opcodes. For these three-byte opcodes, the third opcode byte may be F2H, F3H, or 66H. For example, the SSE2 instruction CVTQ2PD has the three-byte opcode F3 OF E6. The third opcode byte of these three-byte opcodes should not be thought of as a prefix, even though it has the same encoding as the operand size prefix (66H) or one of the repeat prefixes (F2H and F3H). As described above, using the operand size and repeat prefixes with SSE and SSE2 instructions is reserved. It should also be noted that execution of SSE2 instructions on an Intel processor that does not support SSE2 (CPUID Feature flag register EDX bit 26 is clear) will result in unpredictable code execution.

2.3. OPCODE

The primary opcode is 1, 2, or 3 bytes. An additional 3-bit opcode field is sometimes encoded in the ModR/M byte. Smaller encoding fields can be defined within the primary opcode. These fields define the direction of the operation, the size of displacements, the register encoding, condition codes, or sign extension. The encoding of fields in the opcode varies, depending on the class of operation.

2.4. MODR/M AND SIB BYTES

Most instructions that refer to an operand in memory have an addressing-form specifier byte (called the ModR/M byte) following the primary opcode. The ModR/M byte contains three fields of information:

- The *mod* field combines with the *r/m* field to form 32 possible values: eight registers and 24 addressing modes.
- The *reg/opcode* field specifies either a register number or three more bits of opcode information. The purpose of the *reg/opcode* field is specified in the primary opcode.
- The *r/m* field can specify a register as an operand or can be combined with the *mod* field to encode an addressing mode.

Certain encodings of the ModR/M byte require a second addressing byte, the SIB byte, to fully specify the addressing form. The base-plus-index and scale-plus-index forms of 32-bit addressing require the SIB byte. The SIB byte includes the following fields:

- The *scale* field specifies the scale factor.
- The *index* field specifies the register number of the index register.
- The *base* field specifies the register number of the base register.

See Section 2.6., “Addressing-Mode Encoding of ModR/M and SIB Bytes”, for the encodings of the ModR/M and SIB bytes.

2.5. DISPLACEMENT AND IMMEDIATE BYTES

Some addressing forms include a displacement immediately following the ModR/M byte (or the SIB byte if one is present). If a displacement is required, it can be 1, 2, or 4 bytes.

If the instruction specifies an immediate operand, the operand always follows any displacement bytes. An immediate operand can be 1, 2 or 4 bytes.

2.6. ADDRESSING-MODE ENCODING OF MODR/M AND SIB BYTES

The values and the corresponding addressing forms of the ModR/M and SIB bytes are shown in Tables 2-1 through 2-3. The 16-bit addressing forms specified by the ModR/M byte are in Table 2-1, and the 32-bit addressing forms specified by the ModR/M byte are in Table 2-2. Table 2-3 shows the 32-bit addressing forms specified by the SIB byte.

In Tables 2-1 and 2-2, the first column (labeled “Effective Address”) lists 32 different effective addresses that can be assigned to one operand of an instruction by using the Mod and R/M fields of the ModR/M byte. The first 24 effective addresses give the different ways of specifying a memory location; the last eight (specified by the Mod field encoding 11B) give the ways of specifying the general-purpose, MMX, and XMM registers. Each of the register encodings list five possible registers. For example, the first register-encoding (selected by the R/M field encoding of 000B) indicates the general-purpose registers EAX, AX or AL, MMX register MM0, or XMM register XMM0. Which of these five registers is used is determined by the opcode byte and the operand-size attribute, which select either the EAX register (32 bits) or AX register (16 bits).

The second and third columns in Tables 2-1 and 2-2 gives the binary encodings of the Mod and R/M fields in the ModR/M byte, respectively, required to obtain the associated effective address listed in the first column. All 32 possible combinations of the Mod and R/M fields are listed.

Across the top of Tables 2-1 and 2-2, the eight possible values of the 3-bit Reg/Opcode field are listed, in decimal (sixth row from top) and in binary (seventh row from top). The seventh row is labeled “REG=”, which represents the use of these 3 bits to give the location of a second operand, which must be a general-purpose, MMX, or XMM register. If the instruction does not require a second operand to be specified, then the 3 bits of the Reg/Opcode field may be used as an extension of the opcode, which is represented by the sixth row, labeled “/digit (Opcode)”. The five rows above give the byte, word, and doubleword general-purpose registers, the MMX registers, and the XMM registers that correspond to the register numbers, with the same assignments as for the R/M field when Mod field encoding is 11B. As with the R/M field register options, which of the five possible registers is used is determined by the opcode byte along with the operand-size attribute.

The body of Tables 2-1 and 2-2 (under the label “Value of ModR/M Byte (in Hexadecimal)”) contains a 32 by 8 array giving all of the 256 values of the ModR/M byte, in hexadecimal. Bits 3, 4 and 5 are specified by the column of the table in which a byte resides, and the row specifies bits 0, 1 and 2, and also bits 6 and 7.

Table 2-1. 16-Bit Addressing Forms with the ModR/M Byte

			AL	CL	DL	BL	AH	CH	DH	BH
			AX	CX	DX	BX	SP	BP ¹	SI	DI
			EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
			MM0	MM1	MM2	MM3	MM4	MM5	MM6	MM7
			XMM0	XMM1	XMM2	XMM3	XMM4	XMM5	XMM6	XMM7
			0	1	2	3	4	5	6	7
			000	001	010	011	100	101	110	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	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/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

NOTES:

1. The default segment register is SS for the effective addresses containing a BP index, DS for other effective addresses.
2. The disp16 nomenclature denotes a 16-bit displacement that follows the ModR/M byte and that is added to the index.
3. The disp8 nomenclature denotes an 8-bit displacement that follows the ModR/M byte and that is sign-extended and added to the index.

Table 2-2. 32-Bit Addressing Forms with the ModR/M Byte

<i>r8(/r)</i> <i>r16(/r)</i> <i>r32(/r)</i> <i>mm(/r)</i> <i>xmm(/r)</i> <i>/digit (Opcode)</i> <i>REG =</i>	AL AX EAX MM0 XMM0	CL CX ECX MM1 XMM1	DL DX EDX MM2 XMM2	BL BX EBX MM3 XMM3	AH SP ESP MM4 XMM4	CH BP EBP MM5 XMM5	DH SI ESI MM6 XMM6	BH DI EDI MM7 XMM7		
	0 000	1 001	2 010	3 011	4 100	5 101	6 110	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

NOTES:

1. The [--][--] nomenclature means a SIB follows the ModR/M byte.
2. The disp32 nomenclature denotes a 32-bit displacement that follows ModR/M byte (or the SIB byte if one is present) and that is added to the index.
3. The disp8 nomenclature denotes an 8-bit displacement that follows ModR/M byte (or the SIB byte if one is present) and that is sign-extended and added to the index.



Table 2-3 is organized similarly to Tables 2-1 and 2-2, except that its body gives the 256 possible values of the SIB byte, in hexadecimal. Which of the 8 general-purpose registers will be used as base is indicated across the top of the table, along with the corresponding values of the base field (bits 0, 1 and 2) in decimal and binary. The rows indicate which register is used as the index (determined by bits 3, 4 and 5) along with the scaling factor (determined by bits 6 and 7).

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

r32			EAX	ECX	EDX	EBX	ESP	[*]	ESI	EDI
Base =			0	1	2	3	4	5	6	7
Base =			000	001	010	011	100	101	110	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	

NOTE:

- The [*] nomenclature means a disp32 with no base if MOD is 00, [EBP] otherwise. This provides the following addressing modes:

disp32[index] (MOD=00).
 disp8[EBP][index](MOD=01).
 disp32[EBP][index](MOD=10).

INSTRUCTION FORMAT

