X86 Opcode and Instruction Reference Home

Other editions: coder64, coder, geek32, geek64, geek

32-bit ModR/M Byte | 32-bit SIB Byte
16-bit ModR/M Byte

one-byte opcodes index:

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
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
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
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

two-byte opcodes (0F..) index:

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
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
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
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

General notes:

1. *90 NOP*
    a. 90 NOP is not really aliased to XCHG eAX, eAX instruction. This is important in 64-bit mode where the implicit zero-extension to RAX does not happen
2. *SAL*
    a. sandpile.org -- IA-32 architecture -- opcode groups
3. *D6 and F1 opcodes*
    a. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 3: System Programming Guide, Interrupt and Exception Handling
4. *SALC*
    a. sandpile.org -- IA-32 architecture -- one byte opcodes
    b. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
5. *FSTP1*
    a. Christian Ludloff wrote: "While FSTP (D9 /3, mod < 11b), FSTP8 (DF /2, mod = 11b), and FSTP9 (DF /3, mod = 11b) do signal stack underflow, FSTP1 (D9 /3, mod = 11b) does not."
6. *FNENI and FNDISI*
    a. INTEL 80287 PROGRAMMER'S REFERENCE MANUAL 1987, Processor Control Instructions: "The 8087 instructions FENI and FDISI perform no function in the 80287. If these opcodes are detected in an 80286/80287 instruction stream, the 80287 will perform no specific operation and no internal states will be affected."
7. *FNSETPM*
    a. INTEL 80387 PROGRAMMER'S REFERENCE MANUAL 1987, 6.1.2 Independent of CPU Addressing Modes: "Unlike the 80287, the 80387 is not sensitive to the addressing and memory management of the CPU. The 80387 operates the same regardless of whether the 80386 CPU is operating in real-address mode, in protected mode, or in virtual 8086 mode."
8. *FFREEP*
    a. INTEL 80287 PROGRAMMER'S REFERENCE MANUAL 1987, Table A-2. Machine Instruction Decoding Guide: "If the 80287 encounters one of these encodings (DF /1, mod = 11b) in the instruction stream, it will execute it as follows: FFREE ST(i) and pop stack"
    b. Intel Architecture Optimization Reference Manual PIII, Table C-1 Pentium II and Pentium III Processors Instruction to Decoder Specification
    c. AMD Athlon Processor x86 Code Optimization Guide, Chapter 9, Use FFREEP Macro to Pop One Register from the FPU Stack
    d. sandpile.org -- IA-32 architecture -- ESC (FP) opcodes
9. *X87 aliases*
    a. sandpile.org -- IA-32 architecture -- ESC (FP) opcodes
10. *INT1, ICEBP*
    a. sandpile.org -- IA-32 architecture -- one byte opcodes
    b. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes

   c. Christian Ludloff wrote: "Unlike INT 1 (CDh,01h), INT1 (F1h) doesn't perform the IOPL or DPL check and it can't be redirected via the TSS32.IRB."
11. *REP prefixes*
   a. Flags aren't updated until after the last iteration to make the operation faster
12. *TEST*
   a. sandpile.org -- IA-32 architecture -- opcode groups
   b. Christian Ludloff wrote: "While the latest Intel manuals still omit this de-facto standard, the recent x86-64 manuals from AMD document it."
   c. AMD64 Architecture Programmer's Manual Volume 3, Table One-Byte and Two-Byte Opcode ModRM Extensions
13. *SMSW r32/64*
   a. Some processors support reading whole CR0 register, causing a security flaw.
14. *0F0D NOP*
   a. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
   b. AMD architecture maps 3DNow! PREFETCH instructions here
15. *Hintable NOP*
   a. See U.S. Patent 5,701,442
   b. sandpile.org -- IA-32 architecture -- opcode groups
16. *MOV from/to CRn, DRn, TRn*
   a. Christian Ludloff wrote: "For the MOVs from/to CRx/DRx/TRx, mod=00b/01b/10b is aliased to 11b."
   b. AMD64 Architecture Programmer's Manual Volume 3, System Instruction Reference: "This instruction is always treated as a register-to-register instruction, regardless of the encoding of the MOD field in the MODR/M byte."
17. *GETSEC Leaf Functions*
   a. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z: "The GETSEC instruction supports multiple leaf functions. Leaf functions are selected by the value in EAX at the time GETSEC is executed." The following leaf functions are available: CAPABILITIES, ENTERACCS, EXITAC, SENTER, SEXIT, PARAMETERS, SMCTRL, WAKEUP. GETSEC instruction operands are specific to selected leaf function.
18. *SETcc*
   a. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: "The reg field in the ModR/M byte is unused."
19. *CMPXCHG with memory operand*
   a. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: "This instruction can be used with a LOCK prefix …. To simplify the interface to the processor's bus, the destination operand receives a write cycle without regard to the result of the comparison."
   b. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: "CMPXCHG always does a read-modify-write on the memory operand."

20. *0FB9 UD*
    a. Intel 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, Two-byte Opcode Map
    b. sandpile.org -- IA-32 architecture -- two byte opcodes
21. *CMPXCHG8B, CMPXCHG16B*
    a. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: "This instruction can be used with a LOCK prefix …. To simplify the interface to the processor's bus, the destination operand receives a write cycle without regard to the result of the comparison."
    b. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: "The CMPXCHG8B and CMPXCHG16B instructions always do a read-modify-write on the memory operand."
    c. CMPXCHG16B is invalid on early steppings of AMD64 architecture.
22. *BSWAP r16*
    a. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: "When the BSWAP instruction references a 16-bit register, the result is undefined."
    b. AMD64 Architecture Programmer's Manual Volume 3: General-Purpose and System Instructions: "The result of applying the BSWAP instruction to a 16-bit register is undefined."
23. *MASKMOVQ*
    a. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2A: Instruction Set Reference, A-M: "This instruction causes a transition from x87 FPU to MMX technology state."
24. *Intel VMX*
    a. Intel VMX is not binary-compatible with AMD SVM
25. *Intel SSE4*
    a. AMD64 architecture does not support SSE4 instructions but PTEST as part of SSE5

Notes for the Ring Level, used in case of *f* mark:

1. rFlags.IOPL
2. CR4.TSD[bit 2]
3. CR4.PCE[bit 8]

32-bit ModR/M Byte

r8(/r)	AL	CL	DL	BL	AH	CH	DH	BH
r16(/r)	AX	CX	DX	BX	SP	BP	SI	DI
r32(/r)	EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
mm(/r)	MM0	MM1	MM2	MM3	MM4	MM5	MM6	MM7

xmm(/r)			XMM0	XMM1	XMM2	XMM3	XMM4	XMM5	XMM6	XMM7
sreg			ES	CS	SS	DS	FS	GS	res.	res.
eee			CR0	invd	CR2	CR3	CR4	invd	invd	invd
eee			DR0	DR1	DR2	DR3	DR4[1]	DR5[1]	DR6	DR7
(In decimal) /digit (Opcode)			0	1	2	3	4	5	6	7
(In binary) REG =			000	001	010	011	100	101	110	111
Effective Address	Mod	R/M	Value of ModR/M Byte (in Hex)							
[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
[sib]		100	04	0C	14	1C	24	2C	34	3C
disp32		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	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	52	5A	62	6A	72	7A
[EBX]+disp8		011	43	4B	53	5B	63	6B	73	7B
[sib]+disp8		100	44	4C	54	5C	64	6C	74	7C
[EBP]+disp8		101	45	4D	55	5D	65	6D	75	7D
[ESI]+disp8		110	46	4E	56	5E	66	6E	76	7E
[EDI]+disp8		111	47	4F	57	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
[sib]+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
AL/AX/EAX/ST0/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
CL/CX/ECX/ST1/MM1/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
DL/DX/EDX/ST2/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
BL/BX/EBX/ST3/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
AH/SP/ESP/ST4/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
CH/BP/EBP/ST5/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
DH/SI/ESI/ST6/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
BH/DI/EDI/ST7/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

32-bit SIB Byte

r32			EAX	ECX	EDX	EBX	ESP	→[1]	ESI	EDI
(In decimal) Base =			0	1	2	3	4	5	6	7
(In binary) 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

SIB Note 1

Mod bits	base
00	disp32
01	EBP+disp8
10	EBP+disp32

16-bit ModR/M Byte

r8(/r)			AL	CL	DL	BL	AH	CH	DH	BH
r16(/r)			AX	CX	DX	BX	SP	BP	SI	DI
r32(/r)			EAX	ECX	EDX	EBX	ESP	EBP	ESI	EDI
mm(/r)			MM0	MM1	MM2	MM3	MM4	MM5	MM6	MM7
xmm(/r)			XMM0	XMM1	XMM2	XMM3	XMM4	XMM5	XMM6	XMM7
sreg			ES	CS	SS	DS	FS	GS	res.	res.
eee			CR0	invd	CR2	CR3	CR4	invd	invd	invd
eee			DR0	DR1	DR2	DR3	DR4[1]	DR5[1]	DR6	DR7
(In decimal) /digit (Opcode)			0	1	2	3	4	5	6	7
(In binary) REG =			000	001	010	011	100	101	110	111
Effective Address	Mod	R/M	Value of ModR/M Byte (in Hex)							
[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		110	06	0E	16	1E	26	2E	36	3E
[BX]		111	07	0F	17	1F	27	2F	37	3F
[BX+SI]+disp8	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
AL/AX/EAX/ST0/MM0/XMM0	11	000	C0	C8	D0	D8	E0	E8	F0	F8
CL/CX/ECX/ST1/MM1/XMM1		001	C1	C9	D1	D9	E1	E9	F1	F9
DL/DX/EDX/ST2/MM2/XMM2		010	C2	CA	D2	DA	E2	EA	F2	FA
BL/BX/EBX/ST3/MM3/XMM3		011	C3	CB	D3	DB	E3	EB	F3	FB
AH/SP/ESP/ST4/MM4/XMM4		100	C4	CC	D4	DC	E4	EC	F4	FC
CH/BP/EBP/ST5/MM5/XMM5		101	C5	CD	D5	DD	E5	ED	F5	FD
DH/SI/ESI/ST6/MM6/XMM6		110	C6	CE	D6	DE	E6	EE	F6	FE
BH/DI/EDI/ST7/MM7/XMM7		111	C7	CF	D7	DF	E7	EF	F7	FF

ModR/M Note 1: Debug Registers DR4 and DR5

References to debug registers DR4 and DR5 cause an undefined opcode (#UD) exception to be generated when CR4.DE[bit 3] (Debugging Extensions) set; when clear, processor aliases references to registers DR4 and DR5 to DR6 and DR7 for compatibility with software written to run on earlier IA-32 processors.

Your Notes: