



**B**

# **Instruction Formats and Encodings**

**I**





# APPENDIX B

## INSTRUCTION FORMATS AND ENCODINGS

This appendix shows the formats and encodings of the Intel Architecture instructions. The main format and encoding tables are Tables B-10, B-14, B-20, and B-23.

### B.1. MACHINE INSTRUCTION FORMAT

All Intel Architecture instructions are encoded using subsets of the general machine instruction format shown in Figure B-1. Each instruction consists of an opcode, a register and/or address mode specifier (if required) consisting of the ModR/M byte and sometimes the scale-index-base (SIB) byte, a displacement (if required), and an immediate data field (if required).

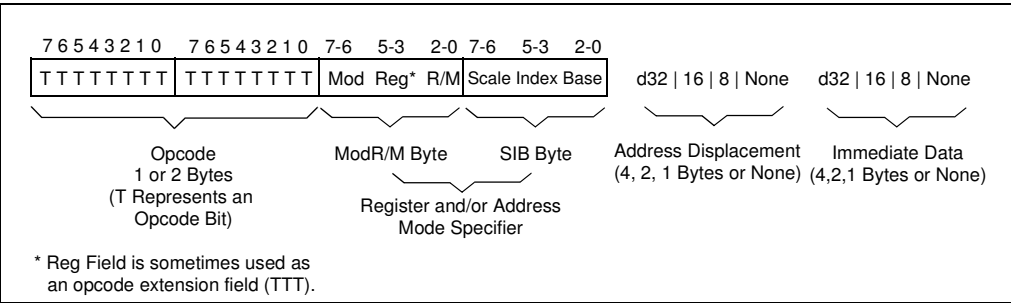


Figure B-1. General Machine Instruction Format

The primary opcode for an instruction is encoded in one or two bytes of the instruction. Some instructions also use an opcode extension field encoded in bits 5, 4, and 3 of the ModR/M byte. Within the primary opcode, smaller encoding fields may be defined. These fields vary according to the class of operation being performed. The fields define such information as register encoding, conditional test performed, or sign extension of immediate byte.

Almost all instructions that refer to a register and/or memory operand have a register and/or address mode byte following the opcode. This byte, the ModR/M byte, consists of the mod field, the reg field, and the R/M field. Certain encodings of the ModR/M byte indicate that a second address mode byte, the SIB byte, must be used.

If the selected addressing mode specifies a displacement, the displacement value is placed immediately following the ModR/M byte or SIB byte. If a displacement is present, the possible sizes are 8, 16, or 32 bits.

If the instruction specifies an immediate operand, the immediate value follows any displacement bytes. An immediate operand, if specified, is always the last field of the instruction.



Table B-1 lists several smaller fields or bits that appear in certain instructions, sometimes within the opcode bytes themselves. The following tables describe these fields and bits and list the allowable values. All of these fields (except the d bit) are shown in the integer instruction formats given in Table B-10.

Table B-1. Special Fields Within Instruction Encodings

Field Name	Description	Number of Bits
reg	General-register specifier (refer to Table B-2 or B-3)	3
w	Specifies if data is byte or full-sized, where full-sized is either 16 or 32 bits (refer to Table B-4)	1
s	Specifies sign extension of an immediate data field (refer to Table B-5)	1
sreg2	Segment register specifier for CS, SS, DS, ES (refer to Table B-6)	2
sreg3	Segment register specifier for CS, SS, DS, ES, FS, GS (refer to Table B-6)	3
eee	Specifies a special-purpose (control or debug) register (refer to Table B-7)	3
tttn	For conditional instructions, specifies a condition asserted or a condition negated (refer to Table B-8)	4
d	Specifies direction of data operation (refer to Table B-9)	1

B.1.1. Reg Field (reg)

The reg field in the ModR/M byte specifies a general-purpose register operand. The group of registers specified is modified by the presence of and state of the w bit in an encoding (refer to Table B-4). Table B-2 shows the encoding of the reg field when the w bit is not present in an encoding, and Table B-3 shows the encoding of the reg field when the w bit is present.

Table B-2. Encoding of reg Field When w Field is Not Present in Instruction

reg Field	Register Selected during 16-Bit Data Operations	Register Selected during 32-Bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI



**Table B-3. Encoding of reg Field When w Field is Present in Instruction**

Register Specified by reg Field during 16-Bit Data Operations			Register Specified by reg Field during 32-Bit Data Operations		
reg	Function of w Field		reg	Function of w Field	
	When w = 0	When w = 1		When w = 0	When w = 1
000	AL	AX	000	AL	EAX
001	CL	CX	001	CL	ECX
010	DL	DX	010	DL	EDX
011	BL	BX	011	BL	EBX
100	AH	SP	100	AH	ESP
101	CH	BP	101	CH	EBP
110	DH	SI	110	DH	ESI
111	BH	DI	111	BH	EDI

### B.1.2. Encoding of Operand Size Bit (w)

The current operand-size attribute determines whether the processor is performing 16- or 32-bit operations. Within the constraints of the current operand-size attribute, the operand-size bit (w) can be used to indicate operations on 8-bit operands or the full operand size specified with the operand-size attribute (16 bits or 32 bits). Table B-4 shows the encoding of the w bit depending on the current operand-size attribute.

**Table B-4. Encoding of Operand Size (w) Bit**

w Bit	Operand Size When Operand-Size Attribute is 16 bits	Operand Size When Operand-Size Attribute is 32 bits
0	8 Bits	8 Bits
1	16 Bits	32 Bits

### B.1.3. Sign Extend (s) Bit

The sign-extend (s) bit occurs primarily in instructions with immediate data fields that are being extended from 8 bits to 16 or 32 bits. Table B-5 shows the encoding of the s bit.

**Table B-5. Encoding of Sign-Extend (s) Bit**

s	Effect on 8-Bit Immediate Data	Effect on 16- or 32-Bit Immediate Data
0	None	None
1	Sign-extend to fill 16-bit or 32-bit destination	None

### B.1.4. Segment Register Field (sreg)

When an instruction operates on a segment register, the reg field in the ModR/M byte is called the sreg field and is used to specify the segment register. Table B-6 shows the encoding of the sreg field. This field is sometimes a 2-bit field (sreg2) and other times a 3-bit field (sreg3).

**Table B-6. Encoding of the Segment Register (sreg) Field**

2-Bit sreg2 Field	Segment Register Selected	3-Bit sreg3 Field	Segment Register Selected
00	ES	000	ES
01	CS	001	CS
10	SS	010	SS
11	DS	011	DS
		100	FS
		101	GS
		110	Reserved*
		111	Reserved*

\* Do not use reserved encodings.

### B.1.5. Special-Purpose Register (eee) Field

When the control or debug registers are referenced in an instruction they are encoded in the eee field, which is located in bits 5, 4, and 3 of the ModR/M byte. Table B-7 shows the encoding of the eee field.

**Table B-7. Encoding of Special-Purpose Register (eee) Field**

eee	Control Register	Debug Register
000	CR0	DR0
001	Reserved*	DR1
010	CR2	DR2
011	CR3	DR3
100	CR4	Reserved*
101	Reserved*	Reserved*
110	Reserved*	DR6
111	Reserved*	DR7

\* Do not use reserved encodings.

### B.1.6. Condition Test Field (ttn)

For conditional instructions (such as conditional jumps and set on condition), the condition test field (ttn) is encoded for the condition being tested for. The ttt part of the field gives the condition to test and the n part indicates whether to use the condition ( $n = 0$ ) or its negation ( $n = 1$ ). For 1-byte primary opcodes, the ttn field is located in bits 3,2,1, and 0 of the opcode byte; for 2-byte primary opcodes, the ttn field is located in bits 3,2,1, and 0 of the second opcode byte. Table B-8 shows the encoding of the ttn field.

**Table B-8. Encoding of Conditional Test (ttn) Field**

t t t n	Mnemonic	Condition
0000	O	Overflow
0001	NO	No overflow
0010	B, NAE	Below, Not above or equal
0011	NB, AE	Not below, Above or equal
0100	E, Z	Equal, Zero
0101	NE, NZ	Not equal, Not zero
0110	BE, NA	Below or equal, Not above
0111	NBE, A	Not below or equal, Above
1000	S	Sign
1001	NS	Not sign
1010	P, PE	Parity, Parity Even
1011	NP, PO	Not parity, Parity Odd
1100	L, NGE	Less than, Not greater than or equal to
1101	NL, GE	Not less than, Greater than or equal to
1110	LE, NG	Less than or equal to, Not greater than
1111	NLE, G	Not less than or equal to, Greater than

### B.1.7. Direction (d) Bit

In many two-operand instructions, a direction bit (d) indicates which operand is considered the source and which is the destination. Table B-9 shows the encoding of the d bit. When used for integer instructions, the d bit is located at bit 1 of a 1-byte primary opcode. This bit does not appear as the symbol “d” in Table B-10; instead, the actual encoding of the bit as 1 or 0 is given. When used for floating-point instructions (in Table B-23), the d bit is shown as bit 2 of the first byte of the primary opcode.

**Table B-9. Encoding of Operation Direction (d) Bit**

d	Source	Destination
0	reg Field	ModR/M or SIB Byte
1	ModR/M or SIB Byte	reg Field

## B.2. INTEGER INSTRUCTION FORMATS AND ENCODINGS

Table B-10 shows the formats and encodings of the integer instructions.

**Table B-10. Integer Instruction Formats and Encodings**

Instruction and Format	Encoding
<b>AAA – ASCII Adjust after Addition</b>	0011 0111
<b>AAD – ASCII Adjust AX before Division</b>	1101 0101 : 0000 1010
<b>AAM – ASCII Adjust AX after Multiply</b>	1101 0100 : 0000 1010
<b>AAS – ASCII Adjust AL after Subtraction</b>	0011 1111
<b>ADC – ADD with Carry</b>	
register1 to register2	0001 000w : 11 reg1 reg2
register2 to register1	0001 001w : 11 reg1 reg2
memory to register	0001 001w : mod reg r/m
register to memory	0001 000w : mod reg r/m
immediate to register	1000 00sw : 11 010 reg : immediate data
immediate to AL, AX, or EAX	0001 010w : immediate data
immediate to memory	1000 00sw : mod 010 r/m : immediate data
<b>ADD – Add</b>	
register1 to register2	0000 000w : 11 reg1 reg2
register2 to register1	0000 001w : 11 reg1 reg2
memory to register	0000 001w : mod reg r/m
register to memory	0000 000w : mod reg r/m
immediate to register	1000 00sw : 11 000 reg : immediate data
immediate to AL, AX, or EAX	0000 010w : immediate data
immediate to memory	1000 00sw : mod 000 r/m : immediate data





Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>AND – Logical AND</b>	
register1 to register2	0010 000w : 11 reg1 reg2
register2 to register1	0010 001w : 11 reg1 reg2
memory to register	0010 001w : mod reg r/m
register to memory	0010 000w : mod reg r/m
immediate to register	
immediate to AL, AX, or EAX	0010 010w : immediate data
immediate to memory	1000 00sw : mod 100 r/m : immediate data
<b>ARPL – Adjust RPL Field of Selector</b>	
from register	0110 0011 : 11 reg1 reg2
from memory	0110 0011 : mod reg r/m
<b>BOUND – Check Array Against Bounds</b>	0110 0010 : mod reg r/m
<b>BSF – Bit Scan Forward</b>	
register1, register2	0000 1111 : 1011 1100 : 11 reg2 reg1
memory, register	0000 1111 : 1011 1100 : mod reg r/m
<b>BSR – Bit Scan Reverse</b>	
register1, register2	0000 1111 : 1011 1101 : 11 reg2 reg1
memory, register	0000 1111 : 1011 1101 : mod reg r/m
<b>BSWAP – Byte Swap</b>	0000 1111 : 1100 1 reg
<b>BT – Bit Test</b>	
register, immediate	0000 1111 : 1011 1010 : 11 100 reg: imm8 data
memory, immediate	0000 1111 : 1011 1010 : mod 100 r/m : imm8 data
register1, register2	0000 1111 : 1010 0011 : 11 reg2 reg1
memory, reg	0000 1111 : 1010 0011 : mod reg r/m
<b>BTC – Bit Test and Complement</b>	
register, immediate	0000 1111 : 1011 1010 : 11 111 reg: imm8 data
memory, immediate	0000 1111 : 1011 1010 : mod 111 r/m : imm8 data
register1, register2	0000 1111 : 1011 1011 : 11 reg2 reg1
memory, reg	0000 1111 : 1011 1011 : mod reg r/m
<b>BTR – Bit Test and Reset</b>	
register, immediate	0000 1111 : 1011 1010 : 11 110 reg: imm8 data
memory, immediate	0000 1111 : 1011 1010 : mod 110 r/m : imm8 data
register1, register2	0000 1111 : 1011 0011 : 11 reg2 reg1
memory, reg	0000 1111 : 1011 0011 : mod reg r/m

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>BTS – Bit Test and Set</b>	
register, immediate	0000 1111 : 1011 1010 : 11 101 reg: imm8 data
memory, immediate	0000 1111 : 1011 1010 : mod 101 r/m : imm8 data
register1, register2	0000 1111 : 1010 1011 : 11 reg2 reg1
memory, reg	0000 1111 : 1010 1011 : mod reg r/m
<b>CALL – Call Procedure (in same segment)</b>	
direct	1110 1000 : full displacement
register indirect	1111 1111 : 11 010 reg
memory indirect	1111 1111 : mod 010 r/m
<b>CALL – Call Procedure (in other segment)</b>	
direct	1001 1010 : unsigned full offset, selector
indirect	1111 1111 : mod 011 r/m
<b>CBW – Convert Byte to Word</b>	1001 1000
<b>CDQ – Convert Doubleword to Qword</b>	1001 1001
<b>CLC – Clear Carry Flag</b>	1111 1000
<b>CLD – Clear Direction Flag</b>	1111 1100
<b>CLI – Clear Interrupt Flag</b>	1111 1010
<b>CLTS – Clear Task-Switched Flag in CR0</b>	0000 1111 : 0000 0110
<b>CMC – Complement Carry Flag</b>	1111 0101
<b>CMOVcc – Conditional Move</b>	
register2 to register1	0000 1111: 0100 ttn : 11 reg1 reg2
memory to register	0000 1111: 0100 ttn : mod mem r/m
<b>CMP – Compare Two Operands</b>	
register1 with register2	0011 100w : 11 reg1 reg2
register2 with register1	0011 101w : 11 reg1 reg2
memory with register	0011 100w : mod reg r/m
register with memory	0011 101w : mod reg r/m
immediate with register	1000 00sw : 11 111 reg : immediate data
immediate with AL, AX, or EAX	0011 110w : immediate data
immediate with memory	1000 00sw : mod 111 r/m
<b>CMPS/CMPSB/CMPSW/CMPSD – Compare String Operands</b>	1010 011w
<b>CMPXCHG – Compare and Exchange</b>	
register1, register2	0000 1111 : 1011 000w : 11 reg2 reg1
memory, register	0000 1111 : 1011 000w : mod reg r/m



## INSTRUCTION FORMATS AND ENCODINGS

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>CMPXCHG8B – Compare and Exchange 8 Bytes</b> memory, register	0000 1111 : 1100 0111 : mod reg r/m
<b>CPUID – CPU Identification</b>	0000 1111 : 1010 0010
<b>CWD – Convert Word to Doubleword</b>	1001 1001
<b>CWDE – Convert Word to Doubleword</b>	1001 1000
<b>DAA – Decimal Adjust AL after Addition</b>	0010 0111
<b>DAS – Decimal Adjust AL after Subtraction</b>	0010 1111
<b>DEC – Decrement by 1</b> register	1111 111w : 11 001 reg
register (alternate encoding)	0100 1 reg
memory	1111 111w : mod 001 r/m
<b>DIV – Unsigned Divide</b> AL, AX, or EAX by register	1111 011w : 11 110 reg
AL, AX, or EAX by memory	1111 011w : mod 110 r/m
<b>ENTER – Make Stack Frame for High Level Procedure</b>	1100 1000 : 16-bit displacement : 8-bit level (L)
<b>HLT – Halt</b>	1111 0100
<b>IDIV – Signed Divide</b> AL, AX, or EAX by register	1111 011w : 11 111 reg
AL, AX, or EAX by memory	1111 011w : mod 111 r/m
<b>IMUL – Signed Multiply</b> AL, AX, or EAX with register	1111 011w : 11 101 reg
AL, AX, or EAX with memory	1111 011w : mod 101 reg
register1 with register2	0000 1111 : 1010 1111 : 11 : reg1 reg2
register with memory	0000 1111 : 1010 1111 : mod reg r/m
register1 with immediate to register2	0110 10s1 : 11 reg1 reg2 : immediate data
memory with immediate to register	0110 10s1 : mod reg r/m : immediate data
<b>IN – Input From Port</b> fixed port	1110 010w : port number
variable port	1110 110w
<b>INC – Increment by 1</b> reg	1111 111w : 11 000 reg
reg (alternate encoding)	0100 0 reg
memory	1111 111w : mod 000 r/m
<b>INS – Input from DX Port</b>	0110 110w

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>INT n – Interrupt Type n</b>	1100 1101 : type
<b>INT – Single-Step Interrupt 3</b>	1100 1100
<b>INTO – Interrupt 4 on Overflow</b>	1100 1110
<b>INVD – Invalidate Cache</b>	0000 1111 : 0000 1000
<b>INVLPG – Invalidate TLB Entry</b>	0000 1111 : 0000 0001 : mod 111 r/m
<b>IRET/IRETD – Interrupt Return</b>	1100 1111
<b>Jcc – Jump if Condition is Met</b>	
8-bit displacement	0111 ttn : 8-bit displacement
full displacement	0000 1111 : 1000 ttn : full displacement
<b>JCXZ/JECXZ – Jump on CX/ECX Zero</b> Address-size prefix differentiates JCXZ and JECXZ	1110 0011 : 8-bit displacement
<b>JMP – Unconditional Jump (to same segment)</b>	
short	1110 1011 : 8-bit displacement
direct	1110 1001 : full displacement
register indirect	1111 1111 : 11 100 reg
memory indirect	1111 1111 : mod 100 r/m
<b>JMP – Unconditional Jump (to other segment)</b>	
direct intersegment	1110 1010 : unsigned full offset, selector
indirect intersegment	1111 1111 : mod 101 r/m
<b>LAHF – Load Flags into AH Register</b>	1001 1111
<b>LAR – Load Access Rights Byte</b>	
from register	0000 1111 : 0000 0010 : 11 reg1 reg2
from memory	0000 1111 : 0000 0010 : mod reg r/m
<b>LDS – Load Pointer to DS</b>	1100 0101 : mod reg r/m
<b>LEA – Load Effective Address</b>	1000 1101 : mod reg r/m
<b>LEAVE – High Level Procedure Exit</b>	1100 1001
<b>LES – Load Pointer to ES</b>	1100 0100 : mod reg r/m
<b>LFS – Load Pointer to FS</b>	0000 1111 : 1011 0100 : mod reg r/m
<b>LGDT – Load Global Descriptor Table Register</b>	0000 1111 : 0000 0001 : mod 010 r/m
<b>LGS – Load Pointer to GS</b>	0000 1111 : 1011 0101 : mod reg r/m
<b>LIDT – Load Interrupt Descriptor Table Register</b>	
<b>LLDT – Load Local Descriptor Table Register</b>	
LDTR from register	0000 1111 : 0000 0000 : 11 010 reg
LDTR from memory	0000 1111 : 0000 0000 : mod 010 r/m



Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>LMSW – Load Machine Status Word</b>	
from register	0000 1111 : 0000 0001 : 11 110 reg
from memory	0000 1111 : 0000 0001 : mod 110 r/m
<b>LOCK – Assert LOCK# Signal Prefix</b>	1111 0000
<b>LODS/LODSB/LODSW/LODSD – Load String Operand</b>	1010 110w
<b>LOOP – Loop Count</b>	1110 0010 : 8-bit displacement
<b>LOOPZ/LOOPE – Loop Count while Zero/Equal</b>	1110 0001 : 8-bit displacement
<b>LOOPNZ/LOOPNE – Loop Count while not Zero/Equal</b>	1110 0000 : 8-bit displacement
<b>LSL – Load Segment Limit</b>	
from register	0000 1111 : 0000 0011 : 11 reg1 reg2
from memory	0000 1111 : 0000 0011 : mod reg r/m
<b>LSS – Load Pointer to SS</b>	0000 1111 : 1011 0010 : mod reg r/m
<b>LTR – Load Task Register</b>	
from register	0000 1111 : 0000 0000 : 11 011 reg
from memory	0000 1111 : 0000 0000 : mod 011 r/m
<b>MOV – Move Data</b>	
register1 to register2	1000 100w : 11 reg1 reg2
register2 to register1	1000 101w : 11 reg1 reg2
memory to reg	1000 101w : mod reg r/m
reg to memory	1000 100w : mod reg r/m
immediate to register	1100 011w : 11 000 reg : immediate data
immediate to register (alternate encoding)	1011 w reg : immediate data
immediate to memory	1100 011w : mod 000 r/m : immediate data
memory to AL, AX, or EAX	1010 000w : full displacement
AL, AX, or EAX to memory	1010 001w : full displacement
<b>MOV – Move to/from Control Registers</b>	
CR0 from register	0000 1111 : 0010 0010 : 11 000 reg
CR2 from register	0000 1111 : 0010 0010 : 11 010reg
CR3 from register	0000 1111 : 0010 0010 : 11 011 reg
CR4 from register	0000 1111 : 0010 0010 : 11 100 reg
register from CR0-CR4	0000 1111 : 0010 0000 : 11 eee reg

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>MOV – Move to/from Debug Registers</b>	
DR0-DR3 from register	0000 1111 : 0010 0011 : 11 eee reg
DR4-DR5 from register	0000 1111 : 0010 0011 : 11 eee reg
DR6-DR7 from register	0000 1111 : 0010 0011 : 11 eee reg
register from DR6-DR7	0000 1111 : 0010 0001 : 11 eee reg
register from DR4-DR5	0000 1111 : 0010 0001 : 11 eee reg
register from DR0-DR3	0000 1111 : 0010 0001 : 11 eee reg
<b>MOV – Move to/from Segment Registers</b>	
register to segment register	1000 1110 : 11 sreg3 reg
register to SS	1000 1110 : 11 sreg3 reg
memory to segment reg	1000 1110 : mod sreg3 r/m
memory to SS	1000 1110 : mod sreg3 r/m
segment register to register	1000 1100 : 11 sreg3 reg
segment register to memory	1000 1100 : mod sreg3 r/m
<b>MOVS/MOVSb/MOVSsw/MOVSd – Move Data from String to String</b>	1010 010w
<b>MOVsx – Move with Sign-Extend</b>	
register2 to register1	0000 1111 : 1011 111w : 11 reg1 reg2
memory to reg	0000 1111 : 1011 111w : mod reg r/m
<b>MOVzx – Move with Zero-Extend</b>	
register2 to register1	0000 1111 : 1011 011w : 11 reg1 reg2
memory to register	0000 1111 : 1011 011w : mod reg r/m
<b>MUL – Unsigned Multiply</b>	
AL, AX, or EAX with register	1111 011w : 11 100 reg
AL, AX, or EAX with memory	1111 011w : mod 100 reg
<b>NEG – Two's Complement Negation</b>	
register	1111 011w : 11 011 reg
memory	1111 011w : mod 011 r/m
<b>NOP – No Operation</b>	1001 0000
<b>NOT – One's Complement Negation</b>	
register	1111 011w : 11 010 reg
memory	1111 011w : mod 010 r/m



## INSTRUCTION FORMATS AND ENCODINGS

**Table B-10. Integer Instruction Formats and Encodings (Contd.)**

Instruction and Format	Encoding
<b>OR – Logical Inclusive OR</b>	
register1 to register2	0000 100w : 11 reg1 reg2
register2 to register1	0000 101w : 11 reg1 reg2
memory to register	0000 101w : mod reg r/m
register to memory	0000 100w : mod reg r/m
immediate to register	1000 00sw : 11 001 reg : immediate data
immediate to AL, AX, or EAX	0000 110w : immediate data
immediate to memory	1000 00sw : mod 001 r/m : immediate data
<b>OUT – Output to Port</b>	
fixed port	1110 011w : port number
variable port	1110 111w
<b>OUTS – Output to DX Port</b>	0110 111w
<b>POP – Pop a Word from the Stack</b>	
register	1000 1111 : 11 000 reg
register (alternate encoding)	0101 1 reg
memory	1000 1111 : mod 000 r/m
<b>POP – Pop a Segment Register from the Stack</b>	
segment register CS, DS, ES	000 sreg2 111
segment register SS	000 sreg2 111
segment register FS, GS	0000 1111: 10 sreg3 001
<b>POPA/POPAD – Pop All General Registers</b>	0110 0001
<b>POPF/POPFD – Pop Stack into FLAGS or EFLAGS Register</b>	1001 1101
<b>PUSH – Push Operand onto the Stack</b>	
register	1111 1111 : 11 110 reg
register (alternate encoding)	0101 0 reg
memory	1111 1111 : mod 110 r/m
immediate	0110 10s0 : immediate data
<b>PUSH – Push Segment Register onto the Stack</b>	
segment register CS,DS,ES,SS	000 sreg2 110
segment register FS,GS	0000 1111: 10 sreg3 000
<b>PUSHA/PUSHAD – Push All General Registers</b>	0110 0000
<b>PUSHF/PUSHFD – Push Flags Register onto the Stack</b>	1001 1100

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>RCL – Rotate thru Carry Left</b>	
register by 1	1101 000w : 11 010 reg
memory by 1	1101 000w : mod 010 r/m
register by CL	1101 001w : 11 010 reg
memory by CL	1101 001w : mod 010 r/m
register by immediate count	1100 000w : 11 010 reg : imm8 data
memory by immediate count	1100 000w : mod 010 r/m : imm8 data
<b>RCR – Rotate thru Carry Right</b>	
register by 1	1101 000w : 11 011 reg
memory by 1	1101 000w : mod 011 r/m
register by CL	1101 001w : 11 011 reg
memory by CL	1101 001w : mod 011 r/m
register by immediate count	1100 000w : 11 011 reg : imm8 data
memory by immediate count	1100 000w : mod 011 r/m : imm8 data
<b>RDMSR – Read from Model-Specific Register</b>	0000 1111 : 0011 0010
<b>RDPMS – Read Performance Monitoring Counters</b>	0000 1111 : 0011 0011
<b>RDTSC – Read Time-Stamp Counter</b>	0000 1111 : 0011 0001
<b>REP INS – Input String</b>	1111 0011 : 0110 110w
<b>REP LODS – Load String</b>	1111 0011 : 1010 110w
<b>REP MOVS – Move String</b>	1111 0011 : 1010 010w
<b>REP OUTS – Output String</b>	1111 0011 : 0110 111w
<b>REP STOS – Store String</b>	1111 0011 : 1010 101w
<b>REPE CMPS – Compare String</b>	1111 0011 : 1010 011w
<b>REPE SCAS – Scan String</b>	1111 0011 : 1010 111w
<b>REPNE CMPS – Compare String</b>	1111 0010 : 1010 011w
<b>REPNE SCAS – Scan String</b>	1111 0010 : 1010 111w
<b>RET – Return from Procedure (to same segment)</b>	
no argument	1100 0011
adding immediate to SP	1100 0010 : 16-bit displacement
<b>RET – Return from Procedure (to other segment)</b>	
intersegment	1100 1011
adding immediate to SP	1100 1010 : 16-bit displacement





## INSTRUCTION FORMATS AND ENCODINGS

**Table B-10. Integer Instruction Formats and Encodings (Contd.)**

Instruction and Format	Encoding
<b>ROL – Rotate Left</b>	
register by 1	1101 000w : 11 000 reg
memory by 1	1101 000w : mod 000 r/m
register by CL	1101 001w : 11 000 reg
memory by CL	1101 001w : mod 000 r/m
register by immediate count	1100 000w : 11 000 reg : imm8 data
memory by immediate count	1100 000w : mod 000 r/m : imm8 data
<b>ROR – Rotate Right</b>	
register by 1	1101 000w : 11 001 reg
memory by 1	1101 000w : mod 001 r/m
register by CL	1101 001w : 11 001 reg
memory by CL	1101 001w : mod 001 r/m
register by immediate count	1100 000w : 11 001 reg : imm8 data
memory by immediate count	1100 000w : mod 001 r/m : imm8 data
<b>RSM – Resume from System Management Mode</b>	0000 1111 : 1010 1010
<b>SAHF – Store AH into Flags</b>	1001 1110
<b>SAL – Shift Arithmetic Left</b>	same instruction as SHL
<b>SAR – Shift Arithmetic Right</b>	
register by 1	1101 000w : 11 111 reg
memory by 1	1101 000w : mod 111 r/m
register by CL	1101 001w : 11 111 reg
memory by CL	1101 001w : mod 111 r/m
register by immediate count	1100 000w : 11 111 reg : imm8 data
memory by immediate count	1100 000w : mod 111 r/m : imm8 data
<b>SBB – Integer Subtraction with Borrow</b>	
register1 to register2	0001 100w : 11 reg1 reg2
register2 to register1	0001 101w : 11 reg1 reg2
memory to register	0001 101w : mod reg r/m
register to memory	0001 100w : mod reg r/m
immediate to register	1000 00sw : 11 011 reg : immediate data
immediate to AL, AX, or EAX	0001 110w : immediate data
immediate to memory	1000 00sw : mod 011 r/m : immediate data
<b>SCAS/SCASB/SCASW/SCASD – Scan String</b>	1101 111w

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>SETcc – Byte Set on Condition</b>	
register	0000 1111 : 1001 ttn : 11 000 reg
memory	0000 1111 : 1001 ttn : mod 000 r/m
<b>SGDT – Store Global Descriptor Table Register</b>	0000 1111 : 0000 0001 : mod 000 r/m
<b>SHL – Shift Left</b>	
register by 1	1101 000w : 11 100 reg
memory by 1	1101 000w : mod 100 r/m
register by CL	1101 001w : 11 100 reg
memory by CL	1101 001w : mod 100 r/m
register by immediate count	1100 000w : 11 100 reg : imm8 data
memory by immediate count	1100 000w : mod 100 r/m : imm8 data
<b>SHLD – Double Precision Shift Left</b>	
register by immediate count	0000 1111 : 1010 0100 : 11 reg2 reg1 : imm8
memory by immediate count	0000 1111 : 1010 0100 : mod reg r/m : imm8
register by CL	0000 1111 : 1010 0101 : 11 reg2 reg1
memory by CL	0000 1111 : 1010 0101 : mod reg r/m
<b>SHR – Shift Right</b>	
register by 1	1101 000w : 11 101 reg
memory by 1	1101 000w : mod 101 r/m
register by CL	1101 001w : 11 101 reg
memory by CL	1101 001w : mod 101 r/m
register by immediate count	1100 000w : 11 101 reg : imm8 data
memory by immediate count	1100 000w : mod 101 r/m : imm8 data
<b>SHRD – Double Precision Shift Right</b>	
register by immediate count	0000 1111 : 1010 1100 : 11 reg2 reg1 : imm8
memory by immediate count	0000 1111 : 1010 1100 : mod reg r/m : imm8
register by CL	0000 1111 : 1010 1101 : 11 reg2 reg1
memory by CL	0000 1111 : 1010 1101 : mod reg r/m
<b>SIDT – Store Interrupt Descriptor Table Register</b>	0000 1111 : 0000 0001 : mod 001 r/m
<b>SLDT – Store Local Descriptor Table Register</b>	
to register	0000 1111 : 0000 0000 : 11 000 reg
to memory	0000 1111 : 0000 0000 : mod 000 r/m



## INSTRUCTION FORMATS AND ENCODINGS

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>SMSW – Store Machine Status Word</b>	
to register	0000 1111 : 0000 0001 : 11 100 reg
to memory	0000 1111 : 0000 0001 : mod 100 r/m
<b>STC – Set Carry Flag</b>	1111 1001
<b>STD – Set Direction Flag</b>	1111 1101
<b>STI – Set Interrupt Flag</b>	1111 1011
<b>STOS/STOSB/STOSW/STOSD – Store String Data</b>	1010 101w
<b>STR – Store Task Register</b>	
to register	0000 1111 : 0000 0000 : 11 001 reg
to memory	0000 1111 : 0000 0000 : mod 001 r/m
<b>SUB – Integer Subtraction</b>	
register1 to register2	0010 100w : 11 reg1 reg2
register2 to register1	0010 101w : 11 reg1 reg2
memory to register	0010 101w : mod reg r/m
register to memory	0010 100w : mod reg r/m
immediate to register	1000 00sw : 11 101 reg : immediate data
immediate to AL, AX, or EAX	0010 110w : immediate data
immediate to memory	1000 00sw : mod 101 r/m : immediate data
<b>TEST – Logical Compare</b>	
register1 and register2	1000 010w : 11 reg1 reg2
memory and register	1000 010w : mod reg r/m
immediate and register	1111 011w : 11 000 reg : immediate data
immediate and AL, AX, or EAX	1010 100w : immediate data
immediate and memory	1111 011w : mod 000 r/m : immediate data
<b>UD2 – Undefined instruction</b>	0000 FFFF : 0000 1011
<b>VERR – Verify a Segment for Reading</b>	
register	0000 1111 : 0000 0000 : 11 100 reg
memory	0000 1111 : 0000 0000 : mod 100 r/m
<b>VERW – Verify a Segment for Writing</b>	
register	0000 1111 : 0000 0000 : 11 101 reg
memory	0000 1111 : 0000 0000 : mod 101 r/m
<b>WAIT – Wait</b>	1001 1011
<b>WBINVD – Writeback and Invalidate Data Cache</b>	0000 1111 : 0000 1001
<b>WRMSR – Write to Model-Specific Register</b>	0000 1111 : 0011 0000

Table B-10. Integer Instruction Formats and Encodings (Contd.)

Instruction and Format	Encoding
<b>XADD – Exchange and Add</b>	
register1, register2	0000 1111 : 1100 000w : 11 reg2 reg1
memory, reg	0000 1111 : 1100 000w : mod reg r/m
<b>XCHG – Exchange Register/Memory with Register</b>	
register1 with register2	1000 011w : 11 reg1 reg2
AL, AX, or EAX with reg	1001 0 reg
memory with reg	1000 011w : mod reg r/m
<b>XLAT/XLATB – Table Look-up Translation</b>	1101 0111
<b>XOR – Logical Exclusive OR</b>	
register1 to register2	0011 000w : 11 reg1 reg2
register2 to register1	0011 001w : 11 reg1 reg2
memory to register	0011 001w : mod reg r/m
register to memory	0011 000w : mod reg r/m
immediate to register	1000 00sw : 11 110 reg : immediate data
immediate to AL, AX, or EAX	0011 010w : immediate data
immediate to memory	1000 00sw : mod 110 r/m : immediate data
<b>Prefix Bytes</b>	
address size	0110 0111
LOCK	1111 0000
operand size	0110 0110
CS segment override	0010 1110
DS segment override	0011 1110
ES segment override	0010 0110
FS segment override	0110 0100
GS segment override	0110 0101
SS segment override	0011 0110