General, base, and Index Registers			Bits for Segment Registers	
bits	W=0	W=1	000	ES
000	AL	AX/EAX	001	CS
001	CL	CX/ECX	010	SS
010	DL	DX/EDX	011	DS
011	BL	BX/EBX	100	FS
100	AH	SP	101	GS
101	CH	BP		
110	DH	SI		
111	BH	DI		

r/m	Mod = 00	Mod=01 or 10	Mod = 11 W =0	Mod = 11 W =1
000	BX+SI	DS:[BX+SI+disp]	AL	AX
001	BX+DI	DS:[BX+DI+disp]	CL	CX
010	BP+SI	SS:[BP+SI+disp]	DL	DX
011	BP+DI	SS:[BP+DI+disp]	BL	BX
100	SI	DS:[SI+disp]	AH	SP
101	DI	DS:[DI+disp]	CH	BP
110	Direct	SS:[BP+disp]	DH	SI
111	BX	DS+[BX+disp]	BH	DI

Instruction	Encoding (bin)[b]
adc reg32, reg32	0110 0110 0001 00x1 [11-reg-r/m]
adc reg8, mem8	0001 0010 [mod-reg-r/m]
adc reg16, mem16	0001 0011 [mod-reg-r/m]
adc reg32, mem32	0110 0110 0001 0011 [mod-reg-r/m]
adc mem8, reg8	0001 0000 [mod-reg-r/m]
adc mem16, reg16	0001 0001 [mod-reg-r/m]
adc mem32, reg32	0110 0110 0001 0001 [mod-reg-r/m]
adc reg8, imm8	1000 00x0 [11-010-r/m] [imm]
adc reg16, imm16	1000 00s0 [11-010-r/m] [imm]
adc reg32, imm32	0110 0110 1000 00s0 [11-010-r/m] [imm]
adc mem8, imm8	1000 00x0 [mod-010-r/m] [imm]
adc mem16, imm16	1000 00s1 [mod-010-r/m] [imm]
adc mem32, imm32	0110 0110 1000 00s1 [mod-010-r/m] [imm]
adc al, imm	0001 0100 [imm]
adc ax, imm	0001 0101 [imm]
adc eax, imm	0110 0110 0001 0101 [imm]
add reg8, reg8	0000 00x0 [11-reg-r/m]
add reg16, reg16	0000 00x1 [11-reg-r/m]
add reg32, reg32	0110 0110 0000 00x1 [11-reg-r/m]

Instruction	Encoding (bin)[b]
add reg8, mem8	0000 0010 [mod-reg-r/m]
add reg16, mem16	0000 0011 [mod-reg-r/m]
add reg32, mem32	0110 0110 0000 0011 [mod-reg-r/m]
add mem8, reg8	0000 0000 [mod-reg-r/m]
add mem16, reg16	0000 0001 [mod-reg-r/m]
add mem32, reg32	0110 0110 0000 0001 [mod-reg-r/m]
add reg8, imm8	1000 00x0 [11-000-r/m] [imm]
add reg16, imm16	1000 00s0 [11-000-r/m] [imm]
add reg32, imm32	0110 0110 1000 00s0 [11-000-r/m] [imm]
add mem8, imm8	1000 00x0 [mod-000-r/m] [imm]
add mem16, imm16	1000 00s1 [mod-000-r/m] [imm]
add mem32, imm32	0110 0110 1000 00s1 [mod-000-r/m] [imm]
add al, imm	0000 0100 [imm]
add ax, imm	0000 0101 [imm]
add eax, imm	0110 0110 0000 0101 [imm]
and reg8, reg8	0010 00x0 [11-reg-r/m]
and reg16, reg16	0010 00x1 [11-reg-r/m]
and reg32, reg32	0110 0110 0010 00x1 [11-reg-/rm]
and reg8, mem8	0010 0010 [mod-reg-r/m]

Instruction	Encoding (bin)[b]
and reg16, mem16	0010 0011 [mod-reg-r/m]
and reg32, mem32	0110 0110 0010 0011 [mod-reg-r/m]
and mem8, reg8	0010 0000 [mod-reg-r/m]
and mem16, reg16	0010 0001 [mod-reg-r/m]
and mem32, reg32	0110 0110 0010 0001 [mod-reg-r/m]
and reg8, imm8	1000 00x0 [11-100-r/m] [imm]
and reg16, imm16	1000 00s1 [11-100-r/m] [imm]
and reg32, imm32	0110 0110 1000 00s1 [11-100-r/m] [imm]
and mem8, imm8	1000 00x0 [mod-100-r/m] [imm]
and mem16, imm16	1000 00s1 [mod-100-r/m] [imm]
and mem32, imm32	0110 0110 1000 00s1 [mod-100-r/m] [imm]
and al, imm	0010 0100 [imm]
and ax, imm	0010 0101 [imm]
and eax, imm	0110 0110 0010 0101 [imm]
bound reg16, mem32	0110 0010 [mod-reg-r/m]
bound reg32, mem64	0110 0110 0110 0010 [mod-reg-r/m]
bsf reg16, reg16	0000 1111 1011 1100 [11-reg-r/m]

Instruction	Encoding (bin)[b]
bsf reg32, reg32	0110 0110 0000 1111 1011 1100 [11-reg-r/m]
bsf reg16, mem16	0000 1111 1011 1100 [mod-reg-r/m]
bsf reg32, mem32	0110 0110 0000 1111 1011 1100 [mod-reg-r/m]
bsr reg16, reg16	0000 1111 1011 1101 [11-reg-r/m]
bsr reg32, reg32	0110 0110 0000 1111 1011 1101 [11-reg-r/m]
bsr reg16, mem16	0000 1111 1011 1101 [mod-reg-r/m]
bsr reg32, mem32	0110 0110 0000 1111 1011 1101 [mod-reg-r/m]
bswap reg32	0000 1111 11001rrr
bt reg16, reg16	0000 1111 1010 0011 [11-reg-r/m]
bt reg32, reg32	0110 0110 0000 1111 1010 0011 [11-reg-r/m]
bt mem16, reg16	0000 1111 1010 0011 [mod-reg-r/m]
bt mem32, reg32	0110 0110 0000 1111 1010 0011 [mod-reg-r/m]
bt reg16, imm	0000 1111 1011 1010 [11-100-r/m] [imm8]
bt reg32, imm	0110 0110 0000 1111 1011 1010 [11-100-r/m] [imm8]
bt mem16, imm	0000 1111 1011 1010 [mod-100-r/m]
bt mem32, imm	0110 0110 0000 1111 1011 1010 [mod-100-r/m]
btc reg16, reg16	0000 1111 1011 1011 [11-reg-r/m]
btc reg32, reg32	0110 0110 0000 1111 1011 1011 [11-reg-r/m]
btc mem16, reg16	0000 1111 1011 1011 [mod-reg-r/m]
btc mem32, reg32	0110 0110 0000 1111 1011 1011 [mod-reg-r/m]
btc reg16, imm	0000 1111 1011 1010 [11-111-r/m] [imm8]
btc reg32, imm	0110 0110 0000 1111 1011 1010 [11-111-r/m] [imm8]
btc mem16, imm	0000 1111 1011 1010 [mod-111-r/m] [imm8]
btc mem32, imm	0110 0110 0000 1111 1011 1010 [mod-111-r/m] [imm8]
btr reg16, reg16	0000 1111 1011 0011 [11-reg-r/m]
btr reg32, reg32	0110 0110 0000 1111 1011 0011 [11-reg-r/m]
btr mem16, reg16	0000 1111 1011 0011 [mod-reg-r/m]
btr mem32, reg32	0110 0110 0000 1111 1011 0011 [mod-reg-r/m]
btr reg16, imm	0000 1111 1011 1010 [11-110-r/m] [imm8]
btr reg32, imm	0110 0110 0000 1111 1011 1010 [11-110-r/m] [imm8]
btr mem16, imm	0000 1111 1011 1010 [mod-110-r/m] [imm8]
btr mem32, imm	0110 0110 0000 1111 1011 1010 [mod-110-r/m] [imm8]
bts reg16, reg16	0000 1111 1010 1011 [11-reg-r/m]
bts reg32, reg32	0110 0110 0000 1111 1010 1011 [11-reg-r/m]
bts mem16, reg16	0000 1111 1010 1011 [mod-reg-r/m]
bts mem32, reg32	0110 0110 0000 1111 1010 1011 [mod-reg-r/m]
bts reg16, imm	0000 1111 1011 1010 [11-101-r/m] [imm8]
bts reg32, imm	0110 0110 0000 1111 1011 1010 [11-101-r/m] [imm8]
bts mem16, imm	0000 1111 1011 1010 [mod-101-r/m] [imm8]
bts mem32, imm	0110 0110 0000 1111 1011 1010 [mod-101-r/m] [imm8]
call near	1110 1000 [disp16]

Instruction	Encoding (bin)[b]
call far	1001 1010 [offset] [segment]
call reg16	1111 1111 [11-010-r/m]
call mem16	1111 1111 [mod-010-r/m]
call mem32	1111 1111 [mod-011-r/m]
cbw	1001 1000
cdq	0110 0110 1001 1001
clc	1111 1000
cld	1111 1100
cli	1111 1010
cmc	1111 0101
cmp reg8, reg8	0011 10x0 [11-reg-r/m]
cmp reg16, reg16	0011 10x1 [11-reg-r/m]
cmp reg32, reg32	0110 0110 0011 10x1 [11-reg-/rm]
cmp reg8, mem8	0011 1010 [mod-reg-r/m]
cmp reg16, mem16	0011 1011 [mod-reg-r/m]
cmp reg32, mem32	0110 0110 0011 1011 [mod-reg-r/m]
cmp mem8, reg8	0011 1000 [mod-reg-r/m]
cmp mem16, reg16	0011 1001 [mod-reg-r/m]
cmp mem32, reg32	0110 0110 0011 1001 [mod-reg-r/m]
cmp reg8, imm8	1000 00x0 [11-111-r/m] [imm]
cmp reg16, imm16	1000 00s0 [11-111-r/m] [imm]
cmp reg32, imm32	0110 0110 1000 00s0 [11-111-r/m] [imm]
cmp mem8, imm8	1000 00x0 [mod-111-r/m] [imm]
cmp mem16, imm16	1000 00s1 [mod-111-r/m] [imm]
cmp mem32, imm32	0110 0110 1000 00s1 [mod-111-r/m] [imm]
cmp al, imm	0011 1100 [imm]
cmp ax, imm	0011 1101 [imm]
cmp eax, imm	0110 0110 0011 1101 [imm]
cmpsb	1010 0110
cmpsw	1010 0111
cmpsd	0110 0110 1010 0111
repe cmpsb	1111 0011 1010 0110
repne cmpsb	1111 0010 1010 0110
repe cmpsw	1111 0011 1010 0111
repne cmpsw	1111 0010 1010 0111
repe cmpsd	0110 0110 1111 0011 1010 0111
repne cmpsd	0110 0110 1111 0010 1010 0111
cmpxchg reg8, reg8	0000 1111 1011 0000 [11-reg-r/m] Note: r/m is first register operand.
cmpxchg reg16, reg16	0000 1111 1011 0001 [11-reg-r/m]
cmpxchg reg32, reg32	0110 0110 0000 1111 1011 0001 [11-reg-r/m]
cmpxchg mem8, reg8	0000 1111 1011 0000 [mod-reg-r/m]
cmpxchg mem16, reg16	0000 1111 1011 0001 [mod-reg-r/m]
cmpxchg mem32, reg32	0110 0110 0000 1111 1011 0001 [mod-reg-r/m]
cmpxchg8b mem64	0000 1111 1100 0111 [mod-001-r/m]
cpuid	0000 1111 1010 0010
cwd	1001 1001
cwde	0110 0110 1001 1000
daa	0010 0111
das	0010 1111
dec reg8	1111 1110 [11-001-r/m]
dec reg16	0100 1rrr
dec reg16 (alternate encoding)	1111 1111 [11-001-r/m]
dec reg32	0110 0110 0100 1rrr
dec reg32 (alternate encoding)	0110 0110 1111 1111 [11-001-r/m]
dec mem8	1111 1110 [mod-001-r/m]
dec mem16	1111 1111 [mod-001-r/m]
dec mem32	0110 0110 1111 1111 [mod-001-r/m]
div reg8	1111 0110 [11-110-r/m]
div reg16	1111 0111 [11-110-r/m]
div reg32	0110 0110 1111 0111 [11-110-r/m]
div mem8	1111 0110 [mod-110-r/m]
div mem16	1111 0111 [mod-110-r/m]
div mem32	0110 0110 1111 0111 [mod-110-r/m]

Instruction	Encoding (bin)[b]
enter local, 0	1100 1000 [locals-imm16] 0000 0000
enter local, 1	1100 1000 [locals-imm16] 0000 0001
enter local, lex	1100 1000 [locals:imm16] [lex:imm8]
hlt	1111 0100
idiv reg8	1111 0110 [11-111-r/m]
idiv reg16	1111 0111 [11-111-r/m]
idiv reg32	0110 0110 1111 0111 [11-111-r/m]
idiv mem8	1111 0110 [mod-111-r/m]
idiv mem16	1111 0111 [mod-111-r/m] [disp]
idiv mem32	0110 0110 1111 0111 [mod-111-r/m]
imul reg8	1111 0110 [11-101-r/m]
imul reg16	1111 0111 [11-101-r/m]
imul reg32	0110 0110 1111 0111 [11-101-r/m]
imul mem8	1111 0110 [mod-101-r/m]
imul mem16	1111 0111 [mod-101-r/m]
imul mem32	0110 0110 1111 0111 [mod-101-r/m]
imul reg16, reg16, imm8 imul reg16, imm8 (Second form assumes reg and r/m are the same, instruction sign extends eight bit immediate operand to 16 bits)	0110 1011 [11-reg-r/m] [imm8] (1st reg operand is specified by reg field, 2nd reg operand is specified by r/m field)
imul reg16, reg16, imm imul reg16, imm	0110 1001 [11-reg-r/m] [imm16]
imul reg32, reg32, imm8 imul reg32, imm8	0110 0110 0110 1011 [11-reg-r/m] [imm8]
imul reg32, reg32, imm imul reg32, imm	0110 0110 0110 1001 [11-reg-r/m] [imm32]
imul reg16,mem16,imm8	0110 1011 [11-reg-r/m] [imm8]
imul reg16,mem16,imm	0110 1001 [11-reg-r/m] [imm16]
imul reg32, mem32, imm8	0110 0110 0110 1011 [11-reg-r/m] [imm8]
imul reg32, mem32, imm	0110 0110 0110 1001 [11-reg-r/m] [imm32]
imul reg16, reg16	0000 1111 1010 1111 [11-reg-r/m] (reg is dest operand)
imul reg32, reg32	0110 0110 0000 1111 1010 1111 [11-reg-r/m] (reg is dest operand)
imul reg16, mem16	0000 1111 1010 1111 [mod-reg-r/m]
imul reg32, mem32	0110 0110 0000 1111 1010 1111 [mod-reg-r/m]
in al, port	1110 0100 [port8]
in ax, port	1110 0101 [port8]
in eax, port	0110 0110 1110 0101 [port8]
in al, dx	1110 1100
in ax, dx	1110 1101
in eax, dx	0110 0110 1110 1101
inc reg8	1111 1110 [11-000-r/m]
inc reg16	0100 0rrr
inc reg16 (alternate encoding)	1111 1111 [11-000-r/m]
inc reg32	0110 0110 0100 0rrr
inc reg32 (alternate encoding)	0110 0110 1111 1111 [11-000-r/m]
inc mem8	1111 1110 [mod-000-r/m]
inc mem16	1111 1110 [mod-000-r/m] [disp]
inc mem32	0110 0110 1111 1110 [mod-000-r/m]
insb	1010 1010
insw	1010 1011
insd	0110 0110 1010 1011
rep insb	1111 0010 1010 1010
rep insw	1111 0010 1010 1011
rep insd	0110 0110 1111 0010 1010 1011
int nn	1100 1101 [imm8]
int 03	1100 1100
into	1100 1110
iret	1100 1111
iretd	0110 0110 1100 1111
ja short	0111 0111 [disp8]
ja near	0000 1111 1000 0111 [disp16]
jae short	0111 0011 [disp8]
jae near	0000 1111 1000 0011 [disp16]
jb short	0111 0010 [disp8]

Instruction	Encoding (bin)[b]
jb near	0000 1111 1000 0010 [disp16]
jbe short	0111 0110 [disp8]
jbe near	0000 1111 1000 0110 [disp16]
jc short	0111 0010 [disp8]
jc near	0000 1111 1000 0010 [disp16]
je short	0111 0100 [disp8]
je near	0000 1111 1000 0100 [disp16]
jg short	0111 1111 [disp8]
jg near	0000 1111 1000 1111 [disp16]
jge short	0111 1101 [disp8]
jge near	0000 1111 1000 1101 [disp16]
jl short	0111 1100 [disp8]
jl near	0000 1111 1000 1100 [disp16]
jle short	0111 1110 [disp8]
jle near	0000 1111 1000 1110 [disp16]
jna short	0111 0110 [disp8]
jna near	0000 1111 1000 0110 [disp16]
jnae short	0111 0010 [disp8]
jnae near	0000 1111 1000 0010 [disp16]
jnb short	0111 0011 [disp8]
jnb near	0000 1111 1000 0011 [disp16]
jnbe short	0111 0111 [disp8]
jnbe near	0000 1111 1000 0111 [disp16]
jnc short	0111 0011 [disp8]
jnc near	0000 1111 1000 0011 [disp16]
jne short	0111 0101 [disp8]
jne near	0000 1111 1000 0101 [disp16]
jng short	0111 1110 [disp8]
jng near	0000 1111 1000 1110 [disp16]
jnge short	0111 1100 [disp8]
jnge near	0000 1111 1000 1100 [disp16]
jnl short	0111 1101 [disp8]
jnl near	0000 1111 1000 1101 [disp16]
jnle short	0111 1111 [disp8]
jnle near	0000 1111 1000 1111 [disp16]
jno short	0111 0001 [disp8]
jno near	0000 1111 1000 0001 [disp16]
jnp short	0111 1011 [disp8]
jnp near	0000 1111 1000 1011 [disp16]
jns short	0111 1001 [disp8]
jns near	0000 1111 1000 1001 [disp16]
jnz short	0111 0101 [disp8]
jnz near	0000 1111 1000 0101 [disp16]
jo short	0111 0000 [disp8]
jo near	0000 1111 1000 0000 [disp16]
jp short	0111 1010 [disp8]
jp near	0000 1111 1000 1010 [disp16]
jpe short	0111 1010 [disp8]
jpe near	0000 1111 1000 1010 [disp16]
jpo short	0111 1011 [disp8]
jpo near	0000 1111 1000 1011 [disp16]
js short	0111 1000 [disp8]
js near	0000 1111 1000 1000 [disp16]
jz short	0111 0100 [disp8]
jz near	0000 1111 1000 0100 [disp16]
jcxz short	1110 0011 [disp8]
jecxz short	0110 0110 1110 0011 [disp8]
jmp short	1110 1011 [disp8]
jmp near	1110 1001 [disp16]
jmp reg16	1111 1111 [11-100-r/m]

Instruction	Encoding (bin)[b]
jmp mem16	1111 1111 [mod-100-r/m]
jmp far	1110 1010 [offset16] [segment16]
jmp mem32	1111 1111 [mod-101-r/m]
lahf	1001 1111
lds reg, mem32	1100 0101 [mod-reg-r/m]
lea reg, mem	1000 1101 [mod-101-r/m]
leave	1100 1001
les reg, mem32	1100 0100 [mod-reg-r/m]
lfs reg, mem32	0000 1111 1011 0100 [mod-reg-r/m]
lgs reg, mem32	0000 1111 1011 0101 [mod-reg-r/m]
lodsb	1010 1100
lodsw	1010 1101
loadsd	0110 0110 1010 1101
loop short	1110 0010 [disp8]
loope short loopz short	1110 0001 [disp8]
loopne short loopnz short	1110 0000 [disp8]
lss reg, mem32	0000 1111 1011 0010 [mod-reg-r/m]
mov reg8, reg8	1000 1000 [11-reg-r/m] (r/m specifies destination reg)
mov reg8, reg8 (alternate encoding)	1000 1010 [11-reg-r/m] (reg specifies destination reg)
mov reg16, reg16	1000 1001 [11-reg-r/m] (r/m specifies destination reg)
mov reg16, reg16 (alternate encoding)	1000 1011 [11-reg-r/m] (reg specifies destination reg)
mov reg32, reg32	0110 0110 1000 1001 [11-reg-r/m] (r/m specifies destination reg)
mov reg32, reg32 (alternate encoding)	0110 0110 1000 1011 [11-reg-r/m] (reg specifies destination reg)
mov mem, reg8	1000 1000 [mod-reg-r/m]
mov reg8, mem	1000 1010 [mod-reg-r/m]
mov mem, reg16	1000 1001 [mod-reg-r/m]
mov reg16, mem	1000 1011 [mod-reg-r/m]
mov mem, reg32	0110 0110 1000 1001 [mod-reg-r/m]
mov reg16, mem	0110 0110 1000 1011 [mod-reg-r/m]
mov reg8, imm	1011 0rrr [imm8]
mov reg8, imm (alternate encoding)	1100 0110 [11-000-r/m] [imm8]
mov reg16, imm	1011 1rrr [imm16]
mov reg16, imm (alternate encoding)	1100 0111 [11-000-r/m] [imm16]
mov reg32, imm	0110 0110 1011 1rrr [imm32]
mov reg32, imm (alternate encoding)	0110 0110 1100 0111 [11-000-r/m] [imm32]
mov mem8, imm	1100 0110 [mod-000-r/m] [imm8]
mov mem16, imm	1100 0111 [mod-000-r/m] [imm16]
mov mem32, imm	1100 0111 [mod-000-r/m] [imm32]
mov al, disp	1010 0000 [disp]
mov ax, disp	1010 0001 [disp]
mov eax, disp	0110 0110 1010 0001 [disp]
mov disp, al	1010 0010 [disp]
mov disp, ax	1010 0011 [disp]
mov disp, eax	0110 0110 1010 0011 [disp]
mov segreg, reg16	1000 1110 [11-sreg-r/m]
mov segreg, mem	1000 1110 [mod-reg-r/m]
mov reg16, segreg	1000 1100 [11-sreg-r/m]
mov mem, segreg	1000 1100 [mod-reg-r/m]
movsb	1010 0100
movsw	1010 0101
movsd	0110 0110 1010 0101
rep movsb	1111 0010 1010 0100
rep movsw	1111 0010 1010 0101
rep movsd	0110 0110 1111 0010 1010 0101
movsx reg16, reg8	0000 1111 1011 1110 [11-reg-r/m] (dest is reg operand)
movsx reg32, reg8	0110 0110 0000 1111 1011 1110 [11-reg-r/m]
movsx reg32, reg16	0110 0110 0000 1111 1011 1111 [11-reg-r/m]
movsx reg16, mem8	0000 1111 1011 1110 [mod-reg-r/m]

Instruction	Encoding (bin)[b]
movsx reg32, mem8	0110 0110 0000 1111 1011 1110 [mod-reg-r/m]
movsx reg32, mem16	0110 0110 0000 1111 1011 1111 [mod-reg-r/m]
movzx reg16, reg8	0000 1111 1011 0110 [11-reg-r/m] (dest is reg operand)
movzx reg32, reg8	0110 0110 0000 1111 1011 0110 [11-reg-r/m]
movzx reg32, reg16	0110 0110 0000 1111 1011 0111 [11-reg-r/m]
movzx reg16, mem8	0000 1111 1011 0110 [mod-reg-r/m]
movzx reg32, mem8	0110 0110 0000 1111 1011 0110 [mod-reg-r/m]
movzx reg32, mem16	0110 0110 0000 1111 1011 0111 [mod-reg-r/m]
mul reg8	1111 0110 [11-100-r/m]
mul reg16	1111 0111 [11-100-r/m]
mul reg32	0110 0110 1111 0111 [11-100-r/m]
mul mem8	1111 0110 [mod-100-r/m]
mul mem16	1111 0111 [mod-100-r/m]
mul mem32	0110 0110 1111 0111 [mod-100-r/m]
neg reg8	1111 0110 [11-011-r/m]
neg reg16	1111 0111 [11-011-r/m]

Instruction	Encoding (bin)[b]
neg reg32	0110 0110 1111 0111 [11-011-r/m]
neg mem8	1111 0110 [mod-011-r/m]
neg mem16	1111 0111 [mod-011-r/m]
neg mem32	0110 0110 1111 0111 [mod-011-r/m]
nop (same as xchg ax, ax)	1001 0000
not reg8	1111 0110 [11-010-r/m]
not reg16	1111 0111 [11-010-r/m]
not reg32	0110 0110 1111 0111 [11-010-r/m]
not mem8	1111 0110 [mod-010-r/m]
not mem16	1111 0111 [mod-010-r/m]
not mem32	0110 0110 1111 0111 [mod-010-r/m]
or reg8, reg8	0000 10x0 [11-reg-r/m]
or reg16, reg16	0000 10x1 [11-reg-r/m]
or reg32, reg32	0110 0110 0000 10x1 [11-reg-r/m]
or reg8, mem8	0000 1010 [mod-reg-r/m]
or reg16, mem16	0000 1011 [mod-reg-r/m]
or reg32, mem32	0110 0110 0000 1011 [mod-reg-r/m]
or mem8, reg8	0000 1000 [mod-reg-r/m]
or mem16, reg16	0000 1001 [mod-reg-r/m]
or mem32, reg32	0110 0110 0000 1001 [mod-reg-r/m]
or reg8, imm8	1000 00x0 [11-001-r/m] [imm]

Instruction	Encoding (bin)[b]
or reg16, imm16	1000 00s0 [11-001-r/m] [imm]
or reg32, imm32	0110 0110 1000 00s0 [11-001-r/m] [imm]
or mem8, imm8	1000 00x0 [mod-001-r/m] [imm]
or mem16, imm16	1000 00s1 [mod-001-r/m] [imm]
or mem32, imm32	0110 0110 1000 00s1 [mod-001-r/m] [imm]
or al, imm	0000 1100 [imm]
or ax, imm	0000 10101 [imm]
or eax, imm	0110 0110 0000 1101 [imm]
out port, al	1110 0110 [port8]
out port, ax	1110 0111 [port8]
out port, eax	0110 0110 1110 0111 [port8]
out dx, al	1110 1110
out dx, ax	1110 1111
out dx, eax	0110 0110 1110 1111
outsb	1010 1010
outsw	1010 1011
outsd	0110 0110 1010 1011
rep outsb	1111 0010 1010 1010
rep outsw	1111 0010 1010 1011
rep outsd	0110 0110 1111 0010 1010 1011
pop reg16	0101 1rrr
pop reg16 (alternate encoding)	1000 1111 [11-000-r/m]

Instruction	Encoding (bin)[b]
pop reg32	0110 0110 0101 1rrr
pop reg32 (alternate encoding)	0110 0110 1000 1111 [11-000-r/m]
pop mem16	1000 1111 [mod-000-r/m]
pop mem32	1000 1111 [mod-000-r/m]
pop es	0000 0111
pop ss	0001 0111
pop ds	0001 1111
pop fs	0000 1111 1010 0001
pop gs	0000 1111 1010 1001
popa	0110 0001
popad	0110 0110 0110 0001
popf	1001 1101
popfd	0110 0110 1001 1101
push reg16	0101 0rrr
push reg16 (alternate encoding)	1111 1111 [11-110-r/m]
push reg32	0110 0110 0101 0rrr
push reg32 (alternate encoding)	0110 0110 1111 1111 [11-110-r/m]
push mem16	1111 1111 [mod-110-r/m]
push mem32	1111 1111 [mod-110-r/m]
push cs	0000 1110
push ds	0001 1110
push es	0000 0110
push ss	0001 0110
push fs	0000 1111 1010 0000
push gs	0000 1111 1010 1000
push imm8->16	0110 1000 [imm8] (sign extends value to 16 bits)
push imm16	0110 1010 [imm16]

Instruction	Encoding (bin)[b]
push imm32	0110 0110 0110 1010 [imm32]
pusha	0110 0000
pushad	0110 0110 0110 0000
pushf	1001 1100
pushfd	0110 0110 1001 1100
rcl reg8, 1	1101 0000 [11-010-r/m]
rcl reg16, 1	1101 0001 [11-010-r/m]
rcl reg32, 1	0110 0110 1101 0001 [11-010-r/m]
rcl mem8, 1	1101 0000 [mod-010-r/m]
rcl mem16, 1	1101 0001 [mod-010-r/m]
rcl mem32, 1	0110 0110 1101 0001 [mod-010-r/m]
rcl reg8, cl	1101 0010 [11-010-r/m]
rcl reg16, cl	1101 0011 [11-010-r/m]
rcl reg32, cl	0110 0110 1101 0011 [11-010-r/m]
rcl mem8, cl	1101 0010 [mod-010-r/m]
rcl mem16, cl	1101 0011 [mod-010-r/m]
rcl mem32, cl	0110 0110 1101 0011 [mod-010-r/m]
rcl reg8, imm8	1100 0000 [11-010-r/m] [imm8]
rcl reg16, imm8	1100 0001 [11-010-r/m] [imm8]
rcl reg32, imm8	0110 0110 1100 0001 [11-010-r/m] [imm8]
rcl mem8, imm8	1100 0000 [mod-010-r/m] [imm8]

Instruction	Encoding (bin)[b]
rcl mem16, imm8	1100 0001 [mod-010-r/m] [imm8]
rcl mem32, imm8	0110 0110 1100 0001 [mod-010-r/m] [imm8]
rcr reg8, 1	1101 0000 [11-011-r/m]
rcr reg16, 1	1101 0001 [11-011-r/m]
rcr reg32, 1	0110 0110 1101 0001 [11-011-r/m]
rcr mem8, 1	1101 0000 [mod-011-r/m]
rcr mem16, 1	1101 0001 [mod-011-r/m]
rcr mem32, 1	0110 0110 1101 0001 [mod-011-r/m]
rcr reg8, cl	1101 0010 [11-011-r/m]
rcr reg16, cl	1101 0011 [11-011-r/m]
rcr reg32, cl	0110 0110 1101 0011 [11-011-r/m]
rcr mem8, cl	1101 0010 [mod-011-r/m]
rcr mem16, cl	1101 0011 [mod-011-r/m]
rcr mem32, cl	0110 0110 1101 0011 [mod-011-r/m]
rcr reg8, imm8	1100 0000 [11-011-r/m] [imm8]
rcr reg16, imm8	1100 0001 [11-011-r/m] [imm8]
rcr reg32, imm8	0110 0110 1100 0001 [11-011-r/m] [imm8]
rcr mem8, imm8	1100 0000 [mod-011-r/m] [imm8]
rcr mem16, imm8	1100 0001 [mod-011-r/m] [imm8]

Instruction	Encoding (bin)[b]
rcr mem32, imm8	0110 0110 1100 0001 [mod-011-r/m] [imm8]
ret retn	1100 0011
ret imm16 retn imm16	1100 0010 [imm16]
ret retf	1100 1011
ret imm16 retf imm16	1100 1010 [imm16]
rol reg8, 1	1101 0000 [11-000-r/m]
rol reg16, 1	1101 0001 [11-000-r/m]
rol reg32, 1	0110 0110 1101 0001 [11-000-r/m]
rol mem8, 1	1101 0000 [mod-000-r/m]
rol mem16, 1	1101 0001 [mod-000-r/m]
rol mem32, 1	0110 0110 1101 0001 [mod-000-r/m]
rol reg8, cl	1101 0010 [11-000-r/m]
rol reg16, cl	1101 0011 [11-000-r/m]
rol reg32, cl	0110 0110 1101 0011 [11-000-r/m]
rol mem8, cl	1101 0010 [mod-000-r/m]
rol mem16, cl	1101 0011 [mod-000-r/m]
rol mem32, cl	0110 0110 1101 0011 [mod-000-r/m]
rol reg8, imm8	1100 0000 [11-000-r/m] [imm8]
rol reg16, imm8	1100 0001 [11-000-r/m] [imm8]
rol reg32, imm8	0110 0110 1100 0001 [11-000-r/m] [imm8]
rol mem8, imm8	1100 0000 [mod-000-r/m] [imm8]
rol mem16, imm8	1100 0001 [mod-000-r/m] [imm8]
rol mem32, imm8	0110 0110 1100 0001 [mod-000-r/m] [imm8]
ror reg8, 1	1101 0000 [11-001-r/m]
ror reg16, 1	1101 0001 [11-001-r/m]
ror reg32, 1	0110 0110 1101 0001 [11-001-r/m]
ror mem8, 1	1101 0000 [mod-001-r/m]
ror mem16, 1	1101 0001 [mod-001-r/m]
ror mem32, 1	0110 0110 1101 0001 [mod-001-r/m]
ror reg8, cl	1101 0010 [11-001-r/m]
ror reg16, cl	1101 0011 [11-001-r/m]
ror reg32, cl	0110 0110 1101 0011 [11-001-r/m]
ror mem8, cl	1101 0010 [mod-001-r/m]
ror mem16, cl	1101 0011 [mod-001-r/m]
ror mem32, cl	0110 0110 1101 0011 [mod-001-r/m]
ror reg8, imm8	1100 0000 [11-001-r/m] [imm8]
ror reg16, imm8	1100 0001 [11-001-r/m] [imm8]
ror reg32, imm8	0110 0110 1100 0001 [11-001-r/m] [imm8]
ror mem8, imm8	1100 0000 [mod-001-r/m] [imm8]
ror mem16, imm8	1100 0001 [mod-001-r/m] [imm8]
ror mem32, imm8	0110 0110 1100 0001 [mod-001-r/m] [imm8]
sahf	1001 1110
sal reg8, 1 (Same instruction as shl)	1101 0000 [11-100-r/m]
sal reg16, 1	1101 0001 [11-100-r/m]
sal reg32, 1	0110 0110 1101 0001 [11-100-r/m]
sal mem8, 1	1101 0000 [mod-100-r/m]
sal mem16, 1	1101 0001 [mod-100-r/m]
sal mem32, 1	0110 0110 1101 0001 [mod-100-r/m]
sal reg8, cl	1101 0010 [11-100-r/m]
sal reg16, cl	1101 0011 [11-100-r/m]
sal reg32, cl	0110 0110 1101 0011 [11-100-r/m]
sal mem8, cl	1101 0010 [mod-100-r/m]
sal mem16, cl	1101 0011 [mod-100-r/m]
sal mem32, cl	0110 0110 1101 0011 [mod-100-r/m]
sal reg8, imm8	1100 0000 [11-100-r/m] [imm8]
sal reg16, imm8	1100 0001 [11-100-r/m] [imm8]
sal reg32, imm8	0110 0110 1100 0001 [11-100-r/m] [imm8]
sal mem8, imm8	1100 0000 [mod-100-r/m] [imm8]

Instruction	Encoding (bin)[b]
sal mem16, imm8	1100 0001 [mod-100-r/m] [imm8]
sal mem32, imm8	0110 0110 1100 0001 [mod-100-r/m] [imm8]
sar reg8, 1	1101 0000 [11-111-r/m]
sar reg16, 1	1101 0001 [11-111-r/m]
sar reg32, 1	0110 0110 1101 0001 [11-111-r/m]
sar mem8, 1	1101 0000 [mod-111-r/m]
sar mem16, 1	1101 0001 [mod-111-r/m]
sar mem32, 1	0110 0110 1101 0001 [mod-111-r/m]
sar reg8, cl	1101 0010 [11-111-r/m]
sar reg16, cl	1101 0011 [11-111-r/m]
sar reg32, cl	0110 0110 1101 0011 [11-111-r/m]
sar mem8, cl	1101 0010 [mod-111-r/m]
sar mem16, cl	1101 0011 [mod-111-r/m]
sar mem32, cl	0110 0110 1101 0011 [mod-111-r/m]
sar reg8, imm8	1100 0000 [11-111-r/m] [imm8]
sar reg16, imm8	1100 0001 [11-111-r/m] [imm8]
sar reg32, imm8	0110 0110 1100 0001 [11-111-r/m] [imm8]
sar mem8, imm8	1100 0000 [mod-111-r/m] [imm8]
sar mem16, imm8	1100 0001 [mod-111-r/m] [imm8]
sar mem32, imm8	0110 0110 1100 0001 [mod-111-r/m] [imm8]
sbb reg8, reg8	0001 10x0 [11-reg-r/m]
sbb reg16, reg16	0001 10x1 [11-reg-r/m]
sbb reg32, reg32	0110 0110 0001 10x1 [11-reg-r/m]
sbb reg8, mem8	0001 1010 [mod-reg-r/m]
sbb reg16, mem16	0001 1011 [mod-reg-r/m]
sbb reg32, mem32	0110 0110 0001 1011 [mod-reg-r/m]
sbb mem8, reg8	0001 1000 [mod-reg-r/m]
sbb mem16, reg16	0001 1001 [mod-reg-r/m]
sbb mem32, reg32	0110 0110 0001 1001 [mod-reg-r/m]
sbb reg8, imm8	1000 00x0 [11-011-r/m] [imm]
sbb reg16, imm16	1000 00s1 [11-011-r/m] [imm]
sbb reg32, imm32	0110 0110 1000 00s1 [11-011-r/m] [imm]
sbb mem8, imm8	1000 00x0 [mod-011-r/m] [imm]
sbb mem16, imm16	1000 00s1 [mod-011-r/m] [imm]
sbb mem32, imm32	0110 0110 1000 00s1 [mod-011-r/m] [imm]
sbb al, imm	0001 1100 [imm]
sbb ax, imm	0001 1101 [imm]
sbb eax, imm	0110 0110 0001 1101 [imm]
scasb	1010 0100
scasw	1010 0101
scasd	0110 0110 1010 0101
rep scasb	1111 0010 1010 0100
rep scasw	1111 0010 1010 0101
rep scasd	0110 0110 1111 0010 1010 0101
seta reg8	0000 1111 1001 0111 [11-000-r/m][c]
seta mem8	0000 1111 1001 0011 [mod-000-r/m]
setae reg8	0000 1111 1001 0011 [11-000-r/m]
setae mem8	0000 1111 1001 0011 [mod-000-r/m]
setb reg8	0000 1111 1001 0010 [11-000-r/m]
setb mem8	0000 1111 1001 0010 [mod-000-r/m]
setbe reg8	0000 1111 1001 0110 [11-000-r/m]
setbe mem8	0000 1111 1001 0110 [mod-000-r/m]
setc reg8	0000 1111 1001 0010 [11-000-r/m]
setc mem8	0000 1111 1001 0010 [mod-000-r/m]
sete reg8	0000 1111 1001 0100 [11-000-r/m]

Instruction	Encoding (bin)[b]
sete mem8	0000 1111 1001 0100 [mod-000-r/m]
setg reg8	0000 1111 1001 1111 [11-000-r/m]
setg mem8	0000 1111 1001 1111 [mod-000-r/m]
setge reg8	0000 1111 1001 1101 [11-000-r/m]
setge mem8	0000 1111 1001 1101 [mod-000-r/m]
setl reg8	0000 1111 1001 1100 [11-000-r/m]
setl mem8	0000 1111 1001 1100 [mod-000-r/m]
setle reg8	0000 1111 1001 1110 [11-000-r/m]
setle mem8	0000 1111 1001 1110 [mod-000-r/m]
setna reg8	0000 1111 1001 0110 [11-000-r/m]
setna mem8	0000 1111 1001 0110 [mod-000-r/m]
setnae reg8	0000 1111 1001 0010 [11-000-r/m]
setnae mem8	0000 1111 1001 0010 [mod-000-r/m]
setnb reg8	0000 1111 1001 0011 [11-000-r/m]
setnb mem8	0000 1111 1001 0011 [mod-000-r/m]
setnbe reg8	0000 1111 1001 0111 [11-000-r/m]
setnbe mem8	0000 1111 1001 0111 [mod-000-r/m]
setnc reg8	0000 1111 1001 0011 [11-000-r/m]
setnc mem8	0000 1111 1001 0011 [mod-000-r/m]
setne reg8	0000 1111 1001 0101 [11-000-r/m]
setne mem8	0000 1111 1001 0101 [mod-000-r/m]
setng reg8	0000 1111 1001 1110 [11-000-r/m]
setng mem8	0000 1111 1001 1110 [mod-000-r/m]
setnge reg8	0000 1111 1001 1100 [11-000-r/m]
setnge mem8	0000 1111 1001 1100 [mod-000-r/m]
setnl reg8	0000 1111 1001 1101 [11-000-r/m]
setnl mem8	0000 1111 1001 1101 [mod-000-r/m]
setnle reg8	0000 1111 1001 1111 [11-000-r/m]
setnle mem8	0000 1111 1001 1111 [mod-000-r/m]
setno reg8	0000 1111 1001 0001 [11-000-r/m]
setno mem8	0000 1111 1001 0001 [mod-000-r/m]
setnp reg8	0000 1111 1001 1011 [11-000-r/m]
setnp mem8	0000 1111 1001 1011 [mod-000-r/m]
setns reg8	0000 1111 1001 1001 [11-000-r/m]
setns mem8	0000 1111 1001 1001 [mod-000-r/m]
setnz reg8	0000 1111 1001 0101 [11-000-r/m]
setnz mem8	0000 1111 1001 0101 [mod-000-r/m]
seto reg8	0000 1111 1001 0000 [11-000-r/m]
seto mem8	0000 1111 1001 0000 [mod-000-r/m]
setp reg8	0000 1111 1001 1010 [11-000-r/m]
setp mem8	0000 1111 1001 1010 [mod-000-r/m]
setpe reg8	0000 1111 1001 1010 [11-000-r/m]
setpe mem8	0000 1111 1001 1010 [mod-000-r/m]
setpo reg8	0000 1111 1001 1011 [11-000-r/m]
setpo mem8	0000 1111 1001 1011 [mod-000-r/m]
sets reg8	0000 1111 1001 1000 [11-000-r/m]
sets mem8	0000 1111 1001 1000 [mod-000-r/m]
setz reg8	0000 1111 1001 0100 [11-000-r/m]
setz mem8	0000 1111 1001 0100 [mod-000-r/m]
shl reg8, 1	1101 0000 [11-100-r/m]
shl reg16, 1	1101 0001 [11-100-r/m]
shl reg32, 1	0110 0110 1101 0001 [11-100-r/m]

Instruction	Encoding (bin)[b]
shl mem8, 1	1101 0000 [mod-100-r/m]
shl mem16, 1	1101 0001 [mod-100-r/m]
shl mem32, 1	0110 0110 1101 0001 [mod-100-r/m]
shl reg8, cl	1101 0010 [11-100-r/m]
shl reg16, cl	1101 0011 [11-100-r/m]
shl reg32, cl	0110 0110 1101 0011 [11-100-r/m]
shl mem8, cl	1101 0010 [mod-100-r/m]
shl mem16, cl	1101 0011 [mod-100-r/m]
shl mem32, cl	0110 0110 1101 0011 [mod-100-r/m]
shl reg8, imm8	1100 0000 [11-100-r/m] [imm8]
shl reg16, imm8	1100 0001 [11-100-r/m] [imm8]
shl reg32, imm8	0110 0110 1100 0001 [11-100-r/m] [imm8]
shl mem8, imm8	1100 0000 [mod-100-r/m] [imm8]
shl mem16, imm8	1100 0001 [mod-100-r/m] [imm8]
shl mem32, imm8	0110 0110 1100 0001 [mod-100-r/m] [imm8]
shld reg16, reg16, imm8 r/m is 1st operand, reg is second operand.	0000 1111 1010 0100 [11-reg-r/m] [imm8]
shld reg32, reg32, imm8 r/m is 1st operand, reg is second operand.	0110 0110 0000 1111 1010 0100 [11-reg-r/m] [imm8]
shld mem16, reg16, imm8	0000 1111 1010 0100 [mod-reg-r/m] [imm8]
shld mem32, reg32, imm8	0110 0110 0000 1111 1010 0100 [mod-reg-r/m] [imm8]
shld reg16, reg16, cl r/m is 1st operand, reg is second operand.	0000 1111 1010 0101 [11-reg-r/m]
shld reg32, reg32, cl r/m is 1st operand, reg is second operand.	0110 0110 0000 1111 1010 0101 [11-reg-r/m]
shld mem16, reg16, cl	0000 1111 1010 0101 [mod-reg-r/m]
shld mem32, reg32, cl	0110 0110 0000 1111 1010 0101 [mod-reg-r/m]
shr reg8, 1	1101 0000 [11-101-r/m]
shr reg16, 1	1101 0001 [11-101-r/m]
shr reg32, 1	0110 0110 1101 0001 [11-101-r/m]
shr mem8, 1	1101 0000 [mod-101-r/m]
shr mem16, 1	1101 0001 [mod-101-r/m]
shr mem32, 1	0110 0110 1101 0001 [mod-101-r/m]
shr reg8, cl	1101 0010 [11-101-r/m]
shr reg16, cl	1101 0011 [11-101-r/m]
shr reg32, cl	0110 0110 1101 0011 [11-101-r/m]
shr mem8, cl	1101 0010 [mod-101-r/m]
shr mem16, cl	1101 0011 [mod-101-r/m]
shr mem32, cl	0110 0110 1101 0011 [mod-101-r/m]
shr reg8, imm8	1100 0000 [11-101-r/m] [imm8]
shr reg16, imm8	1100 0001 [11-101-r/m] [imm8]
shr reg32, imm8	0110 0110 1100 0001 [11-101-r/m] [imm8]
shr mem8, imm8	1100 0000 [mod-101-r/m] [imm8]
shr mem16, imm8	1100 0001 [mod-101-r/m] [imm8]
shr mem32, imm8	0110 0110 1100 0001 [mod-101-r/m] [imm8]
shrd reg16, reg16, imm8 r/m is 1st operand, reg is second operand.	0000 1111 1010 1100 [11-reg-r/m] [imm8]
shrd reg32, reg32, imm8 r/m is 1st operand, reg is second operand.	0110 0110 0000 1111 1010 1100 [11-reg-r/m] [imm8]
shrd mem16, reg16, imm8	0000 1111 1010 1100 [mod-reg-r/m] [imm8]
shrd mem32, reg32, imm8	0110 0110 0000 1111 1010 1100 [mod-reg-r/m] [imm8]
shrd reg16, reg16, cl r/m is 1st operand, reg is second operand.	0000 1111 1010 1101 [11-reg-r/m]
shrd reg32, reg32, cl r/m is 1st operand, reg is second operand.	0110 0110 0000 1111 1010 1101 [11-reg-r/m]
shrd mem16, reg16, cl	0000 1111 1010 1101 [disp]
shld mem32, reg32, cl	0110 0110 0000 1111 1010 1101 [mod-reg-r/m]

Instruction	Encoding (bin)[b]
stc	1111 1001
std	1111 1101
sti	1111 1011
stosb	1010 1010
stosw	1010 1011
stosd	0110 0110 1010 1011
rep stosb	1111 0010 1010 1010
rep stosw	1111 0010 1010 1011
rep stosd	0110 0110 1111 0010 1010 1011
sub reg8, reg8	0010 10x0 [11-reg-r/m]
sub reg16, reg16	0010 10x1 [11-reg-r/m]
sub reg32, reg32	0110 0110 0010 10x1 [11-reg-r/m]
sub reg8, mem8	0010 1010 [mod-reg-r/m]
sub reg16, mem16	0010 1011 [mod-reg-r/m]
sub reg32, mem32	0110 0110 0010 1011 [mod-reg-r/m]
sub mem8, reg8	0010 1000 [mod-reg-r/m]
sub mem16, reg16	0010 1001 [mod-reg-r/m]
sub mem32, reg32	0110 0110 0010 1001 [mod-reg-r/m]
sub reg8, imm8	1000 00x0 [11-101-r/m] [imm]
sub reg16, imm16	1000 00s1 [11-101-r/m] [imm]
sub reg32, imm32	0110 0110 1000 00s1 [11-101-r/m] [imm]
sub mem8, imm8	1000 00x0 [mod-101-r/m] [imm]
sub mem16, imm16	1000 00s1 [mod-101-r/m] [imm]
sub mem32, imm32	0110 0110 1000 00s1 [mod-101-r/m] [imm]
sub al, imm	0010 1100 [imm]
sub ax, imm	0010 1101 [imm]
sub eax, imm	0110 0110 0010 1101 [imm]
test reg8, reg8	1000 0100 [11-reg-r/m]
test reg16, reg16	1000 0101 [11-reg-r/m]
test reg32, reg32	0110 0110 1000 0101 [11-reg-r/m]
test reg8, mem8	1000 0110 [mod-reg-r/m]
test reg16, mem16	1000 0111 [mod-reg-r/m]
test reg32, mem32	0110 0110 1000 0111 [mod-reg-r/m]
test reg8, imm8	1111 0110 [11-000-r/m] [imm]
test reg16, imm16	1111 0111 [11-000-r/m] [imm]
test reg32, imm32	0110 0110 1111 0111 [11-000-r/m] [imm]
test mem8, imm8	1111 0110 [mod-000-r/m] [imm]
test mem16, imm16	1111 0111 [mod-000-r/m] [imm]
test mem32, imm32	0110 0110 1111 0111 [mod-000-r/m] [imm]
test al, imm	1010 1000 [imm]
test ax, imm	1010 1001 [imm]
test eax, imm	0110 0110 1010 1001 [imm]
xadd reg8, reg8 r/m is first operand, reg is second operand.	0000 1111 1100 0000 [11-reg-r/m]
xadd reg16, reg16	0000 1111 1100 0001 [11-reg-r/m]
xadd reg32, reg32	0110 0110 0000 1111 1100 0001 [11-reg-r/m]
xadd mem8, reg8	0000 1111 1100 0000 [mod-reg-r/m]
xadd mem16, reg16	0000 1111 1100 0001 [mod-reg-r/m]
xadd mem32, reg32	0110 0110 0000 1111 1100 0001 [mod-reg-r/m]
xchg reg8, reg8	1000 0110 [11-reg-r/m]
xchg reg16, reg16	1000 0111 [11-reg-r/m]
xchg reg32, reg32	0110 0110 1000 0111 [11-reg-r/m]
xchg mem8, reg8[f]	1000 0110 [11-reg-r/m]
xchg mem16, reg16	1000 0111 [11-reg-r/m]
xchg mem32, reg32	0110 0110 1000 0111 [11-reg-r/m]
xchg ax, reg16	1001 0rrr
xchg ax, reg32	0110 0110 1001 0rrr
xlat	1101 0111
xor reg8, reg8	0011 00x0 [11-reg-r/m]
xor reg16, reg16	0011 00x1 [11-reg-r/m]

Instruction	Encoding (bin)[b]
xor reg32, reg32	0110 0110 0011 00x1 [11-reg-r/m]
xor reg8, mem8	0011 0010 [mod-reg-r/m]
xor reg16, mem16	0011 0011 [mod-reg-r/m]
xor reg32, mem32	0110 0110 0011 0011 [mod-reg-r/m]
xor mem8, reg8	0011 0000 [mod-reg-r/m]
xor mem16, reg16	0011 0001 [mod-reg-r/m]
xor mem32, reg32	0110 0110 0011 0001 [mod-reg-r/m]
xor reg8, imm8	1000 00x0 [11-110-r/m] [imm]
xor reg16, imm16	1000 00s1 [11-110-r/m] [imm]
xor reg32, imm32	0110 0110 1000 00s1 [11-110-r/m] [imm]
xor mem8, imm8	1000 00x0 [mod-110-r/m] [imm]
xor mem16, imm16	1000 00s1 [mod-110-r/m] [imm]
xor mem32, imm32	0110 0110 1000 00s1 [mod-110-r/m] [imm]
xor al, imm	0011 0100 [imm]
xor ax, imm	0011 0101 [imm]
xor eax, imm	0110 0110 0011 0101 [imm]

Instruction	Encoding (bin)[b]
aaa	0011 0111
aad	1101 0101 0000 1010
aam	1101 0100 0000 1010
aas	0011 1111
adc reg8, reg8	0001 00x0 [11-reg-r/m]
adc reg16, reg16	0001 00x1 [11-reg-r/m]

Key to special bits in encodings:

x: Don't care. Can be zero or one.

s: Sign extension bit for immediate operands. If zero, immediate operand is 16 or 32 bits depending on destination operand size. If s bit is one, then the immediate operand is eight bits and the CPU sign extends to 16 or 32 bits, as appropriate.

rrr: Same as reg field in [mod-reg-r/m] byte.

Other Notes:

[disp] This field can be zero, one, two, or four bytes long as required by the instruction.

[imm] This field is one byte long if the operand is an eight bit operand or if the *s* bit in the instruction opcode is one. It is two or four bytes long if the *s* bit contains zero and the destination operand is 16 or 32 bits, respectively.

[mod-reg-r/m]: Instructions that have a mod-reg-r/m byte may have a scaled index byte (sib) and a zero, one, two, or four byte displacement. See Appendix E for details concerning the encoding of this portion of the instruction.

reg,reg Many instructions allow two operands using a [mod-reg-r/m] byte. A single *direction* bit in the opcode determines whether the instruction treats the *reg* operand as the destination or the mod-r/m operand as the destination (e.g., mov reg,mem vs. mov mem,reg). Such instructions also allow two register operands. It turns out there are two encodings for each such reg-reg instruction. That is, you can encode an instruction like mov ax, bx with ax encoded in the reg field and bx encoded in the mod-r/m field, or you can encode it with bx encoded in the reg field and ax encoded in the mod-r/m field. Such instructions always have an *x* bit in the opcode. If the *x* bit is zero, the destination is the register specified by the mod-r/m field. If the *x* bit is one, the destination is the register specified by the reg field. Other types of instructions support multiple encodings for similar reasons.