

**ref.x86asm.net**

**coder64**

edition

pf	OF	po	so	o	proc	st	m	r	l	l	mnemonic	op1	op2	op3	op4	iext	tested	modif	def	undef	f	f	values	description, notes
		00	r							L	ADD	r/m8	r8						o..szapc	o..szapc				Add
		01	r							L	ADD	r/m16/32/64	r16/32/64						o..szapc	o..szapc				Add
		02	r								ADD	r8	r/m8						o..szapc	o..szapc				Add
		03	r								ADD	r16/32/64	r/m16/32/64						o..szapc	o..szapc				Add
		04									ADD	AL	imm8						o..szapc	o..szapc				Add
		05									ADD	rAX	imm16/32						o..szapc	o..szapc				Add
		06						E			invalid													Invalid Instruction in 64-Bit Mode
		07						E			invalid													Invalid Instruction in 64-Bit Mode
		08	r							L	OR	r/m8	r8						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		09	r							L	OR	r/m16/32/64	r16/32/64						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		0A	r								OR	r8	r/m8						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		0B	r								OR	r16/32/64	r/m16/32/64						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		0C									OR	AL	imm8						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		0D									OR	rAX	imm16/32						o..szapc	o..szapc	....a..	o.....c		Logical Inclusive OR
		0E						E			invalid													Invalid Instruction in 64-Bit Mode
		0F									two-byte													
		10	r							L	ADC	r/m8	r8				.....c		o..szapc	o..szapc				Add with Carry
		11	r							L	ADC	r/m16/32/64	r16/32/64				.....c		o..szapc	o..szapc				Add with Carry
		12	r								ADC	r8	r/m8				.....c		o..szapc	o..szapc				Add with Carry
		13	r								ADC	r16/32/64	r/m16/32/64				.....c		o..szapc	o..szapc				Add with Carry
		14									ADC	AL	imm8				.....c		o..szapc	o..szapc				Add with Carry
		15									ADC	rAX	imm16/32				.....c		o..szapc	o..szapc				Add with Carry
		16						E			invalid													Invalid Instruction in 64-Bit Mode
		17						E			invalid													Invalid Instruction in 64-Bit Mode
		18	r							L	SBB	r/m8	r8				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		19	r							L	SBB	r/m16/32/64	r16/32/64				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		1A	r								SBB	r8	r/m8				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		1B	r								SBB	r16/32/64	r/m16/32/64				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		1C									SBB	AL	imm8				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		1D									SBB	rAX	imm16/32				.....c		o..szapc	o..szapc				Integer Subtraction with Borrow
		1E						E			invalid													Invalid Instruction in 64-Bit Mode
		1F						E			invalid													Invalid Instruction in 64-Bit Mode
		20	r							L	AND	r/m8	r8						o..szapc	o..szapc	....a..	o.....c		Logical AND
		21	r							L	AND	r/m16/32/64	r16/32/64						o..szapc	o..szapc	....a..	o.....c		Logical AND
		22	r								AND	r8	r/m8						o..szapc	o..szapc	....a..	o.....c		Logical AND
		23	r								AND	r16/32/64	r/m16/32/64						o..szapc	o..szapc	....a..	o.....c		Logical AND
		24									AND	AL	imm8						o..szapc	o..szapc	....a..	o.....c		Logical AND
		25									AND	rAX	imm16/32						o..szapc	o..szapc	....a..	o.....c		Logical AND
		26						E			null													Null Prefix in 64-bit Mode
		27						E			invalid													Invalid Instruction in 64-Bit Mode
		28	r							L	SUB	r/m8	r8						o..szapc	o..szapc				Subtract
		29	r							L	SUB	r/m16/32/64	r16/32/64						o..szapc	o..szapc				Subtract
		2A	r								SUB	r8	r/m8						o..szapc	o..szapc				Subtract
		2B	r								SUB	r16/32/64	r/m16/32/64						o..szapc	o..szapc				Subtract
		2C									SUB	AL	imm8						o..szapc	o..szapc				Subtract
		2D									SUB	rAX	imm16/32						o..szapc	o..szapc				Subtract
		2E						E			undefined													(branch hint prefixes have no effect in 64-bit mode)
		2E						E			null													Null Prefix in 64-bit Mode
		2F						E			invalid													Invalid Instruction in 64-Bit Mode
		30	r							L	XOR	r/m8	r8						o..szapc	o..szapc	....a..	o.....c		Logical Exclusive OR
		31	r							L	XOR	r/m16/32/64	r16/32/64						o..szapc	o..szapc	....a..	o.....c		Logical Exclusive OR

pf	OF	po	so	o	proc	st	m	r	l	l	mnemonic	opl	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes			
		32		r							XOR	<b>r8</b>	r/m8						o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR		
		33		r							XOR	<b>r16/32/64</b>	r/m16/32/64						o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR		
		34									XOR	<b>AL</b>	imm8						o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR		
		35									XOR	<b>rAX</b>	imm16/32						o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR		
		36					E				<i>null</i>												Null Prefix in 64-bit Mode		
		37					E				<i>invalid</i>													Invalid Instruction in 64-Bit Mode	
		38		r							CMP	r/m8	r8						o..szapc	o..szapc				Compare Two Operands	
		39		r							CMP	r/m16/32/64	r16/32/64						o..szapc	o..szapc				Compare Two Operands	
		3A		r							CMP	r8	r/m8						o..szapc	o..szapc				Compare Two Operands	
		3B		r							CMP	r16/32/64	r/m16/32/64						o..szapc	o..szapc				Compare Two Operands	
		3C									CMP	AL	imm8						o..szapc	o..szapc				Compare Two Operands	
		3D									CMP	rAX	imm16/32						o..szapc	o..szapc				Compare Two Operands	
		3E					E				<i>undefined</i>													(branch hint prefixes have no effect in 64-bit mode)	
		3E					E				<i>null</i>													Null Prefix in 64-bit Mode	
		3F					E				<i>invalid</i>													Invalid Instruction in 64-Bit Mode	
		40					E				REX														
		41					E				REX.B														Extension of the r/m field, base field, or opcode reg field
		42					E				REX.X														Extension of the SIB index field
		43					E				REX.XB														
		44					E				REX.R														Extension of the ModR/M reg field
		45					E				REX.RB														
		46					E				REX.RX														
		47					E				REX.RXB														
		48					E				REX.W														64 Bit Operand Size
		49					E				REX.WB														
		4A					E				REX.WX														
		4B					E				REX.WXB														
		4C					E				REX.WR														
		4D					E				REX.WRB														
		4E					E				REX.WRX														
		4F					E				REX.WRXB														
		50+r					E				PUSH	r64/16													Push Word, Doubleword or Quadword Onto the Stack
		58+r					E				POP	r64/16													Pop a Value from the Stack
		60					E				<i>invalid</i>														Invalid Instruction in 64-Bit Mode
		61					E				<i>invalid</i>														Invalid Instruction in 64-Bit Mode
		62					E				<i>invalid</i>														Invalid Instruction in 64-Bit Mode
		63		r			E				MOVSXD	r32/64	r/m32												Move with Sign-Extension
		64					E				<i>undefined</i>														(branch hint prefixes have no effect in 64-bit mode)
		65									GS	GS													GS segment override prefix
		65									<i>undefined</i>														(used only with Jcc instructions)
		66									<i>no mnemonic</i>														Operand-size override prefix
		66				M					<i>no mnemonic</i>						sse2								Precision-size override prefix
		67									<i>no mnemonic</i>														Address-size override prefix
		68									PUSH	imm16/32													Push Word, Doubleword or Quadword Onto the Stack
		69									IMUL	r16/32/64	r/m16/32/64	imm16/32					o..szapc	o.....c	...szap.				Signed Multiply
		6A									PUSH	imm8													Push Word, Doubleword or Quadword Onto the Stack
		6B									IMUL	r16/32/64	r/m16/32/64	imm8					o..szapc	o.....c	...szap.				Signed Multiply
		6C					f <sup>1</sup>				INS	m8	DX				.d.....								Input from Port to String
							f <sup>1</sup>				INSB	m8	DX				.d.....								Input from Port to String
		6D					f <sup>1</sup>				INS	m16	DX				.d.....								Input from Port to String
							f <sup>1</sup>				INSW	m16	DX				.d.....								Input from Port to String

pf	OF	po	so	o	proc	st	m	rl	l	mnemonic	op1	op2	op3	op4	ixext	tested f	modif f	def f	undef f	f values	description, notes	
		6D							f <sup>1</sup>	INS	m16/32	DX				.d.....					Input from Port to String	
										INSD	m32	DX										
		6E							f <sup>1</sup>	OUTS	DX	m8				.d.....					Output String to Port	
										OUTSB	DX	m8										
		6F							f <sup>1</sup>	OUTS	DX	m16				.d.....					Output String to Port	
										OUTSW	DX	m16										
		6F							f <sup>1</sup>	OUTS	DX	m16/32				.d.....					Output String to Port	
										OUTSD	DX	m32										
		70								JO	rel8					o.....					Jump short if overflow (OF=1)	
		71								JNO	rel8					o.....					Jump short if not overflow (OF=0)	
		72								JB	rel8					.....c					Jump short if below/not above or equal/carry (CF=1)	
										JNAE	rel8											
										JC	rel8											
		73								JNB	rel8					.....c					Jump short if not below/above or equal/not carry (CF=0)	
										JAE	rel8											
										JNC	rel8											
		74								JZ	rel8					....z...					Jump short if zero/equal (ZF=0)	
										JE	rel8											
		75								JNZ	rel8					....z...					Jump short if not zero/not equal (ZF=1)	
										JNE	rel8											
		76								JBE	rel8					....z.c					Jump short if below or equal/not above (CF=1 AND ZF=1)	
										JNA	rel8											
		77								JNBE	rel8					....z.c					Jump short if not below or equal/above (CF=0 AND ZF=0)	
										JA	rel8											
		78								JS	rel8					...s...					Jump short if sign (SF=1)	
		79								JNS	rel8					...s...					Jump short if not sign (SF=0)	
		7A								JP	rel8					.....p.					Jump short if parity/parity even (PF=1)	
										JPE	rel8											
		7B								JNP	rel8					.....p.					Jump short if not parity/parity odd	
										JPO	rel8											
		7C								JL	rel8					o..s...					Jump short if less/not greater (SF!=OF)	
										JNGE	rel8											
		7D								JNL	rel8					o..s...					Jump short if not less/greater or equal (SF=OF)	
										JGE	rel8											
		7E								JLE	rel8					o..sz...					Jump short if less or equal/not greater ((ZF=1) OR (SF!=OF))	
										JNG	rel8											
		7F								JNLE	rel8					o..sz...					Jump short if not less nor equal/greater ((ZF=0) AND (SF=OF))	
										JG	rel8											
		80	0							L ADD	r/m8	imm8					o..szapc	o..szapc			Add	
		80	1							L OR	r/m8	imm8					o..szapc	o..sz.pc	.....a..	o.....c	Logical Inclusive OR	
		80	2							L ADC	r/m8	imm8				.....c	o..szapc	o..szapc			Add with Carry	
		80	3							L SBB	r/m8	imm8				.....c	o..szapc	o..szapc			Integer Subtraction with Borrow	
		80	4							L AND	r/m8	imm8					o..szapc	o..sz.pc	.....a..	o.....c	Logical AND	
		80	5							L SUB	r/m8	imm8					o..szapc	o..szapc			Subtract	
		80	6							L XOR	r/m8	imm8					o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR	
		80	7							CMP	r/m8	imm8					o..szapc	o..szapc			Compare Two Operands	
		81	0							L ADD	r/m16/32/64	imm16/32					o..szapc	o..szapc			Add	
		81	1							L OR	r/m16/32/64	imm16/32					o..szapc	o..sz.pc	.....a..	o.....c	Logical Inclusive OR	
		81	2							L ADC	r/m16/32/64	imm16/32				.....c	o..szapc	o..szapc			Add with Carry	
		81	3							L SBB	r/m16/32/64	imm16/32				.....c	o..szapc	o..szapc			Integer Subtraction with Borrow	
		81	4							L AND	r/m16/32/64	imm16/32					o..szapc	o..sz.pc	.....a..	o.....c	Logical AND	
		81	5							L SUB	r/m16/32/64	imm16/32					o..szapc	o..szapc			Subtract	
		81	6							L XOR	r/m16/32/64	imm16/32					o..szapc	o..sz.pc	.....a..	o.....c	Logical Exclusive OR	



pf	OF	po	so	o	proc	st	m	r	l	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes	
										CMPSD	m32	m32										
										CMPSQ	m64	m64										
		A8								TEST	AL	imm8						o..szapc	o..sz.pc	.....a..	o.....c	Logical Compare
		A9								TEST	rAX	imm16/32						o..szapc	o..sz.pc	.....a..	o.....c	Logical Compare
		AA								STOS	m8	AL				.d.....						Store String
										STOSB	m8	AL										
		AB						E		STOS	m16/32/64	rAX										
										STOSW	m16	AX				.d.....						Store String
										STOSD	m32	EAX										
										STOSQ	m64	RAX										
		AC								LODS	AL	m8				.d.....						Load String
										LODSB	AL	m8										
		AD						E		LODS	rAX	m16/32/64				.d.....						Load String
										LODSW	AX	m16										
										LODSD	EAX	m32										
										LODSQ	RAX	m64										
		AE								SCAS	m8	AL				.d.....	o..szapc	o..szapc				Scan String
										SCASB	m8	AL										
		AF						E		SCAS	m16/32/64	rAX				.d.....	o..szapc	o..szapc				Scan String
										SCASW	m16	AX										
										SCASD	m32	EAX										
										SCASQ	m64	RAX										
		B0+r								MOV	r8	imm8										Move
		B8+r								MOV	r16/32/64	imm16/32/64										Move
		C0	0							ROL	r/m8	imm8						o..szapc	o..szapc	o.....		Rotate
		C0	1							ROR	r/m8	imm8						o..szapc	o..szapc	o.....		Rotate
		C0	2							RCL	r/m8	imm8				.....c	o..szapc	o..szapc	o.....			Rotate
		C0	3							RCR	r/m8	imm8				.....c	o..szapc	o..szapc	o.....			Rotate
		C0	4							SHL	r/m8	imm8						o..szapc	o..sz.pc	o....a.c		Shift
										SAL	r/m8	imm8										
		C0	5							SHR	r/m8	imm8						o..szapc	o..sz.pc	o....a.c		Shift
		C0	6			U <sup>5</sup>				SAL	r/m8	imm8						o..szapc	o..sz.pc	o....a.c		Shift
										SHL	r/m8	imm8										
		C0	7							SAR	r/m8	imm8						o..szapc	o..sz.pc	o....a..		Shift
		C1	0							ROL	r/m16/32/64	imm8						o..szapc	o..szapc	o.....		Rotate
		C1	1							ROR	r/m16/32/64	imm8						o..szapc	o..szapc	o.....		Rotate
		C1	2							RCL	r/m16/32/64	imm8				.....c	o..szapc	o..szapc	o.....			Rotate
		C1	3							RCR	r/m16/32/64	imm8				.....c	o..szapc	o..szapc	o.....			Rotate
		C1	4							SHL	r/m16/32/64	imm8						o..szapc	o..sz.pc	o....a.c		Shift
										SAL	r/m16/32/64	imm8										
		C1	5							SHR	r/m16/32/64	imm8						o..szapc	o..sz.pc	o....a.c		Shift
		C1	6			U <sup>5</sup>				SAL	r/m16/32/64	imm8						o..szapc	o..sz.pc	o....a.c		Shift
										SHL	r/m16/32/64	imm8										
		C1	7							SAR	r/m16/32/64	imm8						o..szapc	o..sz.pc	o....a..		Shift
		C2								RETN	imm16											Return from procedure
		C3								RETN												Return from procedure
		C4						E		invalid												Invalid Instruction in 64-Bit Mode
		C5						E		invalid												Invalid Instruction in 64-Bit Mode
		C6	0							MOV	r/m8	imm8										Move
		C7	0							MOV	r/m16/32/64	imm16/32										Move
		C8								ENTER	rBP	imm16	imm8									Make Stack Frame for Procedure Parameters
		C9								LEAVE	rBP											High Level Procedure Exit

pf	Of	po	so	o	proc	st	m	rl	l	mnemonic	opl	op2	op3	op4	ixt	tested f	modif f	def f	undef f	f values	description, notes	
										f	RETf	imm16										Return from procedure
										f	RETf											Return from procedure
										f	INT	3				..i.....	..i.....		..i.....			Call to Interrupt Procedure
										f	INT	imm8				..i.....	..i.....		..i.....			Call to Interrupt Procedure
										f	INTO					o.....	..i.....	..i.....		..i.....		Call to Interrupt Procedure
									E f		IRET											Interrupt Return
											IRETD						odiszapc	odiszapc				
											IRETQ											
		D0	0								ROL	r/m8	1					o..szapc	o..szapc			Rotate
		D0	1								ROR	r/m8	1					o..szapc	o..szapc			Rotate
		D0	2								RCL	r/m8	1			.....c		o..szapc	o..szapc			Rotate
		D0	3								RCR	r/m8	1			.....c		o..szapc	o..szapc			Rotate
		D0	4								SHL	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
											SAL	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
		D0	5								SHR	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
		D0	6				U <sup>5</sup>				SAL	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
											SHL	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
		D0	7								SAR	r/m8	1					o..szapc	o..sz.pc	.....a..		Shift
		D1	0								ROL	r/m16/32/64	1					o..szapc	o..szapc			Rotate
		D1	1								ROR	r/m16/32/64	1					o..szapc	o..szapc			Rotate
		D1	2								RCL	r/m16/32/64	1			.....c		o..szapc	o..szapc			Rotate
		D1	3								RCR	r/m16/32/64	1			.....c		o..szapc	o..szapc			Rotate
		D1	4								SHL	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
											SAL	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
		D1	5								SHR	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
		D1	6				U <sup>5</sup>				SAL	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
											SHL	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
		D1	7								SAR	r/m16/32/64	1					o..szapc	o..sz.pc	.....a..		Shift
		D2	0								ROL	r/m8	CL					o..szapc	o..szapc	o.....		Rotate
		D2	1								ROR	r/m8	CL					o..szapc	o..szapc	o.....		Rotate
		D2	2								RCL	r/m8	CL			.....c		o..szapc	o..szapc	o.....		Rotate
		D2	3								RCR	r/m8	CL			.....c		o..szapc	o..szapc	o.....		Rotate
		D2	4								SHL	r/m8	CL					o..szapc	o..sz.pc	o....a.c		Shift
											SAL	r/m8	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D2	5								SHR	r/m8	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D2	6				U <sup>5</sup>				SAL	r/m8	CL					o..szapc	o..sz.pc	o....a.c		Shift
											SHL	r/m8	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D2	7								SAR	r/m8	CL					o..szapc	o..sz.pc	o....a..		Shift
		D3	0								ROL	r/m16/32/64	CL					o..szapc	o..szapc	o.....		Rotate
		D3	1								ROR	r/m16/32/64	CL					o..szapc	o..szapc	o.....		Rotate
		D3	2								RCL	r/m16/32/64	CL			.....c		o..szapc	o..szapc	o.....		Rotate
		D3	3								RCR	r/m16/32/64	CL			.....c		o..szapc	o..szapc	o.....		Rotate
		D3	4								SHL	r/m16/32/64	CL					o..szapc	o..sz.pc	o....a.c		Shift
											SAL	r/m16/32/64	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D3	5								SHR	r/m16/32/64	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D3	6				U <sup>5</sup>				SAL	r/m16/32/64	CL					o..szapc	o..sz.pc	o....a.c		Shift
											SHL	r/m16/32/64	CL					o..szapc	o..sz.pc	o....a.c		Shift
		D3	7								SAR	r/m16/32/64	CL					o..szapc	o..sz.pc	.....a..		Shift
		D4						E			invalid											Invalid Instruction in 64-Bit Mode
		D5						E			invalid											Invalid Instruction in 64-Bit Mode
		D6						E			invalid											Invalid Instruction in 64-Bit Mode

pf	of	so	o	proc	st	m	r	l	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes
		D7							XLAT	<b>AL</b>	m8									Table Look-up Translation
									XLATB	<b>AL</b>	m8									
		E0							LOOPNZ	<b>rCX</b>	rel8				....z...					Decrement count; Jump short if count!=0 and ZF=0
									LOOPNE	<b>rCX</b>	rel8									
		E1							LOOPZ	<b>rCX</b>	rel8				....z...					Decrement count; Jump short if count!=0 and ZF=1
									LOOPE	<b>rCX</b>	rel8									
		E2							LOOP	<b>rCX</b>	rel8									Decrement count; Jump short if count!=0
		E3				E			JECXZ	rel8	ECX									Jump short if rCX register is 0
									JRCXZ	rel8	RCX									
		E4				f <sup>1</sup>			IN	<b>AL</b>	imm8									Input from Port
		E5				f <sup>1</sup>			IN	<b>eAX</b>	imm8									Input from Port
		E6				f <sup>1</sup>			OUT	<b>imm8</b>	AL									Output to Port
		E7				f <sup>1</sup>			OUT	<b>imm8</b>	eAX									Output to Port
		E8		D <sup>23</sup>	E				CALL	rel32										Call Procedure
		E9		D <sup>23</sup>	E				JMP	rel32										Jump
		EA				E			<i>invalid</i>											Invalid Instruction in 64-Bit Mode
		EB							JMP	rel8										Jump
		EC				f <sup>1</sup>			IN	<b>AL</b>	DX									Input from Port
		ED				f <sup>1</sup>			IN	<b>eAX</b>	DX									Input from Port
		EE				f <sup>1</sup>			OUT	<b>DX</b>	AL									Output to Port
		EF				f <sup>1</sup>			OUT	<b>DX</b>	eAX									Output to Port
		F0							LOCK											Assert LOCK# Signal Prefix
		F1		D <sup>6</sup>					<i>undefined</i>											Undefined and Reserved; Does not Generate #UD
		F1		U <sup>8</sup>					INT1							..i.....	..i.....		..i.....	Call to Interrupt Procedure
									ICEBP											
		F2							REPZ	<b>rCX</b>					....z...					Repeat String Operation Prefix
									REPNE	<b>rCX</b>										
		F2		U					REP	<b>rCX</b>										Repeat String Operation Prefix
		F2		M					<i>no mnemonic</i>					sse2						Scalar Double-precision Prefix
		F3							REPZ	<b>rCX</b>					....z...					Repeat String Operation Prefix
									REPE	<b>rCX</b>										
		F3							REP	<b>rCX</b>										Repeat String Operation Prefix
		F3		M					<i>no mnemonic</i>					sse						Scalar Single-precision Prefix
		F4							HLT											Halt
		F5							CMC						.....c	.....c	.....c			Complement Carry Flag
		F6	0						TEST	r/m8	imm8				o..szapc	o..sz.pc	....a..	o.....c		Logical Compare
		F6	1	U <sup>9</sup>					TEST	r/m8	imm8				o..szapc	o..sz.pc	....a..	o.....c		Logical Compare
		F6	2						NOT	<b>r/m8</b>										One's Complement Negation
		F6	3						NEG	<b>r/m8</b>					o..szapc	o..szapc				Two's Complement Negation
		F6	4						MUL	<b>AX</b>	AL	r/m8			o..szapc	o.....c	...szap.			Unsigned Multiply
		F6	5						IMUL	<b>AX</b>	AL	r/m8			o..szapc	o.....c	...szap.			Signed Multiply
		F6	6						DIV	<b>AL</b>	<b>AH</b>	AX	r/m8		o..szapc		o..szapc			Unsigned Divide
		F6	7						IDIV	<b>AL</b>	<b>AH</b>	AX	r/m8		o..szapc		o..szapc			Signed Divide
		F7	0						TEST	r/m16/32/64	imm16/32/64				o..szapc	o..sz.pc	....a..	o.....c		Logical Compare
		F7	1	U <sup>9</sup>					TEST	r/m16/32/64	imm16/32/64				o..szapc	o..sz.pc	....a..	o.....c		Logical Compare
		F7	2						NOT	<b>r/m16/32/64</b>										One's Complement Negation
		F7	3						NEG	<b>r/m16/32/64</b>					o..szapc	o..szapc				Two's Complement Negation
		F7	4						MUL	<b>rDX</b>	<b>rAX</b>	r/m16/32/64			o..szapc	o.....c	...szap.			Unsigned Multiply
		F7	5						IMUL	<b>rDX</b>	<b>rAX</b>	r/m16/32/64			o..szapc	o.....c	...szap.			Signed Multiply

pf	OF	po	so	o	proc	st	m	r	l	l	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes
	F7		6								DIV	<b>rDX</b>	<b>rAX</b>	r/m16/32/64				o..szapc		o..szapc		Unsigned Divide
	F7		7								IDIV	<b>rDX</b>	<b>rAX</b>	r/m16/32/64				o..szapc		o..szapc		Signed Divide
	F8										CLC							.....c	.....c		.....c	Clear Carry Flag
	F9										STC							.....c	.....c		.....c	Set Carry Flag
	FA							f <sup>1</sup>			CLI							..i.....	..i.....		..i.....	Clear Interrupt Flag
	FB							f <sup>1</sup>			STI							..i.....	..i.....		..I.....	Set Interrupt Flag
	FC										CLD							.d.....	.d.....		.d.....	Clear Direction Flag
	FD										STD							.d.....	.d.....		.D.....	Set Direction Flag
	FE		0								INC	<b>r/m8</b>						o..szap.	o..szap.			Increment by 1
	FE		1								DEC	<b>r/m8</b>						o..szap.	o..szap.			Decrement by 1
	FF		0								INC	<b>r/m16/32/64</b>						o..szap.	o..szap.			Increment by 1
	FF		1								DEC	<b>r/m16/32/64</b>						o..szap.	o..szap.			Decrement by 1
	FF		2		D <sup>23</sup>	E					CALL	<b>r/m64</b>										Call Procedure
	FF		3		D <sup>10</sup>						CALLF	<b>r/m16:16/32/64</b>										Call Procedure
	FF		4		D <sup>23</sup>	E					JMP	<b>r/m64</b>										Jump
	FF		5		D <sup>10</sup>						JMPF	<b>r/m16:16/32/64</b>										Jump
	FF		6			E					PUSH	<b>r/m64/16</b>										Pop a Value from the Stack





pf	OF	po	so	o	proc	st	m	r	l	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes
	0F	8E					D <sup>23</sup>	E		JLE	rel32					o..sz...					Jump short if less or equal/not greater ((ZF=1) OR (SF!=OF))
										JNG	rel32										
	0F	8F					D <sup>23</sup>	E		JNLE	rel32					o..sz...					Jump short if not less nor equal/greater ((ZF=0) AND (SF=OF))
										JG	rel32										
	0F	90		0			D <sup>18</sup>			SETO	r/m8					o.....					Set Byte on Condition - overflow (OF=1)
	0F	91		0			D <sup>18</sup>			SETNO	r/m8					o.....					Set Byte on Condition - not overflow (OF=0)
										SETB	r/m8										
	0F	92		0			D <sup>18</sup>			SETNAE	r/m8					.....c					Set Byte on Condition - below/not above or equal/carry (CF=1)
										SETC	r/m8										
										SETNB	r/m8										
	0F	93		0			D <sup>18</sup>			SETAE	r/m8					.....c					Set Byte on Condition - not below/above or equal/not carry (CF=0)
										SETNC	r/m8										
										SETZ	r/m8					....z...					
	0F	94		0			D <sup>18</sup>			SETE	r/m8										Set Byte on Condition - zero/equal (ZF=0)
										SETNZ	r/m8					....z...					
	0F	95		0			D <sup>18</sup>			SETNE	r/m8										Set Byte on Condition - not zero/not equal (ZF=1)
										SETBE	r/m8					....z..c					
	0F	96		0			D <sup>18</sup>			SETNA	r/m8										Set Byte on Condition - below or equal/not above (CF=1 AND ZF=1)
										SETNBE	r/m8					....z..c					
	0F	97		0			D <sup>18</sup>			SETA	r/m8										Set Byte on Condition - not below or equal/above (CF=0 AND ZF=0)
										SETS	r/m8					...s....					
	0F	98		0			D <sup>18</sup>			SETNS	r/m8					...s....					Set Byte on Condition - not sign (SF=0)
										SETP	r/m8					.....p.					
	0F	9A		0			D <sup>18</sup>			SETPE	r/m8										Set Byte on Condition - parity/parity even (PF=1)
										SETNP	r/m8					.....p.					
	0F	9B		0			D <sup>18</sup>			SETPO	r/m8										Set Byte on Condition - not parity/parity odd
										SETL	r/m8					o..s....					
	0F	9C		0			D <sup>18</sup>			SETNGE	r/m8										Set Byte on Condition - less/not greater (SF!=OF)
										SETNL	r/m8					o..s....					
	0F	9D		0			D <sup>18</sup>			SETGE	r/m8										Set Byte on Condition - not less/greater or equal (SF=OF)
										SETLE	r/m8					o..sz...					
	0F	9E		0			D <sup>18</sup>			SETNG	r/m8										Set Byte on Condition - less or equal/not greater ((ZF=1) OR (SF!=OF))
										SETNLE	r/m8					o..sz...					
	0F	9F		0			D <sup>18</sup>			SETG	r/m8										Set Byte on Condition - not less nor equal/greater ((ZF=0) AND (SF=OF))
	0F	A0								PUSH	FS										Push Word, Doubleword or Quadword Onto the Stack
	0F	A1								POP	FS										Pop a Value from the Stack
	0F	A2								CPUID	...										CPU Identification
	0F	A3								BT	r/m16/32/64	r16/32/64				o..szapc	.....c	o..szap.			Bit Test
	0F	A4								SHLD	r/m16/32/64	r16/32/64	imm8			o..szapc	o..sz.pc	o....a.c			Double Precision Shift Left
	0F	A5								SHLD	r/m16/32/64	r16/32/64	CL			o..szapc	o..sz.pc	o....a.c			Double Precision Shift Left
	0F	A8								PUSH	GS										Push Word, Doubleword or Quadword Onto the Stack
	0F	A9								POP	GS										Pop a Value from the Stack
	0F	AA						S		RSM						odiszapc	odiszapc				Resume from System Management Mode
	0F	AB							L	BTS	r/m16/32/64	r16/32/64				o..szapc	.....c	o..szap.			Bit Test and Set
	0F	AC								SHRD	r/m16/32/64	r16/32/64	imm8			o..szapc	o..sz.pc	o....a.c			Double Precision Shift Right
	0F	AD								SHRD	r/m16/32/64	r16/32/64	CL			o..szapc	o..sz.pc	o....a.c			Double Precision Shift Right
	0F	AF		r						IMUL	r16/32/64	r/m16/32/64				o..szapc	o.....c	...szap.			Signed Multiply
	0F	B0		r					L	CMPXCHG	r/m8	AL	r8			o..szapc	o..szapc				Compare and Exchange
	0F	B1		r					L	CMPXCHG	r/m16/32/64	rAX	r16/32/64			o..szapc	o..szapc				Compare and Exchange
	0F	B2		r			D <sup>19</sup>			LSS	SS	r16/32/64	m16:16/32/64								Load Far Pointer
	0F	B3							L	BTR	r/m16/32/64	r16/32/64				o..szapc	.....c	o..szap.			Bit Test and Reset



## General notes:

1. a. OPCODE.LST, Revision 4.51, 15 Oct 1999 © Potemkin's Hackers Group 1994...1999
2. a. The microarchitecture of Intel and AMD CPU's, By Agner Fog, Copyright © 1996 - 2006.
3. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 2B: Instruction Set Reference, N-Z, PAUSE instruction
4. a. LAHF nad SAHF are invalid on early steppings of EM64T architecture; that's why they need CPUID.8000001H:ECX.LAHF-SAHF[bit 0]
5. a. sandpile.org -- IA-32 architecture -- opcode groups
6. a. Intel® 64 and IA-32 Architecture Software Developer's Manual Volume 3: System Programming Guide, Interrupt and Exception Handling
7. a. sandpile.org -- IA-32 architecture -- one byte opcodes  
b. AMD64 Architecture Programmer's Manual Volume 3, Table One-Bytes Opcodes
8. 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."
9. 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
10. a. AMD64 architecture does not enable 64-bit offset: AMD64 Architecture Programmer's Manual Volume 3: If the operand-size is 32 or 64 bits, the operand is a 16-bit selector followed by a 32-bit offset.
11. a. sandpile.org -- IA-32 architecture -- two byte opcodes  
b. www.x86.org - The LOADALL Instruction
12. a. On AMD64 architecture, SYSCALL is valid also in legacy mode
13. 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
14. a. AMD64 Architecture Programmer's Manual Volume 3, System Instruction Reference: If CPUID.8000001H:ECX.4, CR8 can be read and written in legacy mode using a LOCK prefix instead of a REX prefix to specify the additional opcode bit.
15. 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.
16. a. On AMD64 architecture, SYSENTER is valid only in legacy mode.
17. a. On AMD64 architecture, SYSEXIT is not valid in long mode.
18. a. AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: The reg field in the ModR/M byte is unused.
19. a. AMD64 architecture does not enable 64-bit operands: AMD64 Architecture Programmers Manual Volume 3: General-Purpose and System Instructions: Executing LFS, LGS, or LSS with a 64-bit operand size only loads a 32-bit general purpose register and the specified segment register
20. 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. a. On AMD64 architecture, BSF and BSR instructions act differently if the content of the source operand is 0
22. a. CMPXCHG16B is invalid on early steppings of AMD64 architecture
23. a. Use of operand-size prefix in 64-bit mode may result in implementation-dependent behaviour; on AMD64 architecture, this prefix acts as expected

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

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