X86 Opcode Reference
64-bit Edition

general, system, x87 FPU, MMX, SSE(1), SSE2, SSE3, SSSE3 opcodes

ISBN 978-80-254-2350-9

X86 Opcode Reference, 64-bit Edition
general, system, x87 FPU, MMX, SSE(1), SSE2, SSE3, SSSE3 opcodes

First Edition, July 2008

Errata:
http://ref.x86asm.net/errata/64/opcode

Karel Lejska
Bayerova 8
Brno 60200
Czech Republic

For comments, suggestions, questions or bug reports, please contact
mazegen@gmail.com

Credits: Christian Ludloff, Martin Mocko (*vid*), Anthony Lopes, *Aquila*, *EliCZ*, *Cephexin*

ISBN 978-80-254-2350-9

Quick Guide

- *mnemonic*: Instruction mnemonic itself. If the mnemonic is set up using *italic*, there is no oficial mnemonic and the present one is just a suggested one
- *op1–op4*: Up to four instruction operands. Implicate operands are set up using *italic*. Modified operands are **bold**. Implicate *[RSP]* operand is not indicated. If the *op4* column contains only three dots '...', there are more than four operands
- *pf*: Prefix value, or if Primary opcode is present, fixed extraordinary prefix
- *0F*: Dedicated for *0x0F* two-byte prefix
- *po*: Primary opcode. Second opcode byte in case of multi-byte opcodes. *+r* means a register code, from 0 through 7, added to the value
- *so*: Secondary opcode. Fixed appended value to the primary opcode
- *o*: Register/Opcode field. Either the value of an opcode extension (values from 0 through 7) or *r* indicates that the ModR/M byte contains a register operand and an r/m operand
- *proc*: Indicates the instruction's introductory processor. If the column is empty, it means that the instruction is supported since first implementation of Intel EM64T architecture.
- *st*: Indicates how is the instruction documented in the Intel manuals. *D* means fully documented. *M* means documented only marginally. *U* undocumented at all. Empty column means *D*
- *m*: Indicates the mode in which is the instruction valid. Virtual-8086 Mode and SMM is not taken into account. *R* applies for real , protected and 64-bit mode. *P* applies for protected and 64-bit mode. *E* applies for 64-bit mode. If this column is empty, it means *R*
- *rl*: The ring level, which is the instruction valid from (3 or 0). *f* indicates that the level depends on further flag(s)
- *x*: For general instructions, *L* indicates that the instruction is basically valid with *LOCK* (*0xF0*) prefix. For x87 FPU instructions, *s* incidates that the opcode performs additional pu**s**h of a value to the register stack, *p* incidates that the opcode performs additional **p**op of the register stack, *P* pops twice
- *iext*: The instruction extension group, which was the opcode released on
- *tested f*, *modif f*, *def f*, *undef f*: For RFlags register, indicates these flags using *odiszapc* pattern. Present flag fits in with the appropriate group. For x87 FPU flags, indicates these flags using *1234* x87 FPU flag pattern. Present flag fits in with the appropriate group.
- *f values*: For RFlags register, indicates the values of flags, which are always set or cleared, using case–sensitive *odiszapc* flag pattern. Lower–case flag means cleared flag, upper–case means set flag. For x87 FPU flags, indicates these flags using *1234* x87 FPU flag pattern. Present flag holds its value
- *description, notes*: Generic description

Visit http://ref.x86asm.net for detailed guide.

One-byte General and System Instructions

pf	0F	po	so	o	proc	st	m	rl	x	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	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
		08		r					L	OR	**r/m8,**	r8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		09		r					L	OR	**r/m16/32/64,**	r16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0A		r						OR	**r8,**	r/m8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0B		r						OR	**r16/32/64,**	r/m16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0C								OR	**AL,**	imm8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0D								OR	**rAX,**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0F								*Two-byte Instructions*											
		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
		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
		20		r					L	AND	**r/m8,**	r8					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		21		r					L	AND	**r/m16/32/64,**	r16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		22		r						AND	**r8,**	r/m8					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		23		r						AND	**r16/32/64,**	r/m16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		24								AND	**AL,**	imm8					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		25								AND	**rAX,**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
	26						E			*null*											Null Prefix 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
		30		r					L	XOR	**r/m8,**	r8					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		31		r					L	XOR	**r/m16/32/64,**	r16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR

pf	0F	po	so	o	proc	st	m	rl	x	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes
		32		r						XOR	**r8,**	r/m8					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		33		r						XOR	**r16/32/64,**	r/m16/32/64					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		34								XOR	**AL,**	imm8					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		35								XOR	**rAX,**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
36							E			*null*											Null Prefix 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			*undefined*											(branch hint prefixes have no effect in 64-bit mode)
3E							E			*null*											Null Prefix in 64-bit Mode
40							E			REX											Access to new 8-bit registers
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											REX.X and REX.B combination
44							E			REX.R											Extension of the ModR/M reg field
45							E			REX.RB											REX.R and REX.B combination
46							E			REX.RX											REX.R and REX.X combination
47							E			REX.RXB											REX.R, REX.X and REX.B combination
48							E			REX.W											64 Bit Operand Size
49							E			REX.WB											REX.W and REX.B combination
4A							E			REX.WX											REX.W and REX.X combination
4B							E			REX.WXB											REX.W, REX.X and REX.B combination
4C							E			REX.WR											REX.W and REX.R combination
4D							E			REX.WRB											REX.W, REX.R and REX.B combination
4E							E			REX.WRX											REX.W, REX.R and REX.X combination
4F							E			REX.WRXB											REX.W, REX.R, REX.X and REX.B combination
		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
		63		r			E			MOVSXD	**r32/64,**	r/m32									Move with Sign-Extension
64							E			*undefined*											(branch hint prefixes have no effect in 64-bit mode)
65										GS	*GS*										GS segment override prefix
65										*undefined*											(used only with Jcc instructions)
66										*no mnemonic*											Operand-size override prefix
66						M				*no mnemonic*					sse2						Precision-size override prefix
67										*no mnemonic*											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[1]		INS INSB	**m8,** ***m8,***	DX *DX*				.d......					Input from Port to String