imul stc cmp subss fsqrt stosd jnle psubq unpcklps mov bswap cmovna pmaxsw sysenter inc repnz sbb fstp outsw out shrd jnle phaddsw psadbw idiv finit leave smsw retn fincstp fcmovnbe fxam mfence cmovpe str fnsave mulss add sal cvtpi2pd ror push verr sgdt nop cwde fldz aaa phaddd pushad movsxd emms hlt rdtsc daa fbld fiadd psllq fldlg2 cvttss2si int xor bts sub fstenv fsubr bsf fnclex setnge pmovmskb movq wait xlatb cdq movddup haddpd fabs movlhps movntpd movntdq addsubpd not lidt lss comisd lsl jnae scasw clflush divps jp subps orpd rcpps xchg rcl pop sar jrcxz pmuludq

X86 Opcode Reference
32-bit Edition

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

Advertisement

ISBN 978-80-254-2349-3

9 788025 423493

X86 Opcode Reference, 32-bit Edition
general, system, x87 FPU, MMX, SSE(1), SSE2, SSE3, SSSE3 opcodes

Copyright © MazeGen

First Edition, July 2008

Errata:
http://ref.x86asm.net/errata/32/opcode

Karel Lejska
Bayerova 8
Brno 60200
Czech Republic

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

For booking a computer-related ad in this reference, please contact mazegen@gmail.com

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

ISBN 978-80-254-2349-3

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 *SS:[eSP]* 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 8086 processor.
- *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 and protected mode. *P* applies for protected 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 EFlags 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 EFlags 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,**	r16/32					o..szapc	o..szapc			Add
		02		r						ADD	**r8,**	r/m8					o..szapc	o..szapc			Add
		03		r						ADD	**r16/32,**	r/m16/32					o..szapc	o..szapc			Add
		04								ADD	**AL,**	imm8					o..szapc	o..szapc			Add
		05								ADD	**eAX,**	imm16/32					o..szapc	o..szapc			Add
		06								PUSH	ES										Push Word, Doubleword or Quadword Onto the Stack
		07								POP	**ES**										Pop a Value from the Stack
		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,**	r16/32					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,**	r/m16/32					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	**eAX,**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0E								PUSH	CS										Push Word, Doubleword or Quadword Onto the Stack
		0F			02+					*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,**	r16/32				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,**	r/m16/32				c	o..szapc	o..szapc			Add with Carry
		14								ADC	**AL,**	imm8				c	o..szapc	o..szapc			Add with Carry
		15								ADC	**eAX,**	imm16/32				c	o..szapc	o..szapc			Add with Carry
		16								PUSH	SS										Push Word, Doubleword or Quadword Onto the Stack
		17								POP	**SS**										Pop a Value from the Stack
		18		r					L	SBB	**r/m8,**	r8				c	o..szapc	o..szapc			Integer Subtraction with Borrow
		19		r					L	SBB	**r/m16/32,**	r16/32				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,**	r/m16/32				c	o..szapc	o..szapc			Integer Subtraction with Borrow
		1C								SBB	**AL,**	imm8				c	o..szapc	o..szapc			Integer Subtraction with Borrow
		1D								SBB	**eAX,**	imm16/32				c	o..szapc	o..szapc			Integer Subtraction with Borrow
		1E								PUSH	DS										Push Word, Doubleword or Quadword Onto the Stack
		1F								POP	**DS**										Pop a Value from the Stack
		20		r					L	AND	**r/m8,**	r8					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
		21		r					L	AND	**r/m16/32,**	r16/32					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,**	r/m16/32					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	**eAX,**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical AND
26										ES	*ES*										ES segment override prefix
26					P4+					*undefined*											(use with any branch instruction is reserved)
		27								DAA	**AL**					.....a.c	o..szapc	...szapc	o.......		Decimal Adjust AL after Addition
		28		r					L	SUB	**r/m8,**	r8					o..szapc	o..szapc			Subtract

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
		29		r					L	SUB	**r/m16/32**,	r16/32					o..szapc	o..szapc			Subtract
		2A		r						SUB	**r8**,	r/m8					o..szapc	o..szapc			Subtract
		2B		r						SUB	**r16/32**,	r/m16/32					o..szapc	o..szapc			Subtract
		2C								SUB	**AL**,	imm8					o..szapc	o..szapc			Subtract
		2D								SUB	**eAX**,	imm16/32					o..szapc	o..szapc			Subtract
2E										CS	*CS*										CS segment override prefix
2E					P4+					*NTAKEN*											Branch not taken prefix (used only with Jcc instructions)
		2F								DAS	***AL***					.....a.c	o..szapc	...szapc	o.......		Decimal Adjust AL after Subtraction
		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**,	r16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		32		r						XOR	**r8**,	r/m8					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
		33		r						XOR	**r16/32**,	r/m16/32					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	**eAX**,	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Exclusive OR
36										SS	*SS*										SS segment override prefix
36					P4+					*undefined*											(use with any branch instruction is reserved)
		37								AAA	**AL**,	***AH***				.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust After Addition
		38		r						CMP	r/m8,	r8					o..szapc	o..szapc			Compare Two Operands
		39		r						CMP	r/m16/32,	r16/32					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,	r/m16/32					o..szapc	o..szapc			Compare Two Operands
		3C								CMP	AL,	imm8					o..szapc	o..szapc			Compare Two Operands
		3D								CMP	eAX,	imm16/32					o..szapc	o..szapc			Compare Two Operands
3E										DS	*DS*										DS segment override prefix
3E					P4+					*TAKEN*											Branch taken prefix (used only with Jcc instructions)
		3F								AAS	**AL**,	***AH***				.....a..	o..szapc	.....a.c	o..sz.p.		ASCII Adjust AL After Subtraction
		40+r								INC	**r16/32**						o..szap.	o..szap.			Increment by 1
		48+r								DEC	**r16/32**						o..szap.	o..szap.			Decrement by 1
		50+r								PUSH	r16/32										Push Word, Doubleword or Quadword Onto the Stack
		58+r								POP	**r16/32**										Pop a Value from the Stack
		60			01+					PUSHA	*AX*,	*CX*,	*DX*,	...							Push All General-Purpose Registers
		60			03+					PUSHA PUSHAD	*AX*, *EAX*,	*CX*, *ECX*,	*DX*, *EDX*,	... ...							Push All General-Purpose Registers
		61			01+					POPA	***DI***,	***SI***,	***BP***,	...							Pop All General-Purpose Registers
		61			03+					POPA POPAD	***DI***, ***EDI***,	***SI***, ***ESI***,	***BP***, ***EBP***,	... ...							Pop All General-Purpose Registers
		62		r	01+			f		BOUND	r16/32,	m16/32&16/32,	*eFlags*				..i.....	..i.....		..i.....	Check Array Index Against Bounds
		63		r	02+					ARPL	r/m16,	r16					....z...	....z...			Adjust RPL Field of Segment Selector
64					03+					FS	*FS*										FS segment override prefix
64					P4+					*undefined*											(used only with Jcc instructions)
65					03+					GS	*GS*										GS segment override prefix
65					P4+					*undefined*											(used only with Jcc instructions)
66										*no mnemonic*											Operand-size override prefix
66					P4+	M				*no mnemonic*					sse2						Precision-size override prefix
67										*no mnemonic*											Address-size override prefix