8086 INSTRUCTION SET

AAA		ASCII adjust addition
AAD		ASCII adjust division
AAM		ASCII adjust multiply
AAS		ASCII adjust subtraction
ADC	dt,sc	Add with carry
ADD	dt,sc	Add
AND	dt,sc	Logical AND
CALL	proc	Call a procedure
CBW		Convert byte to word
CLC		Clear carry flag
CDL		Clear direction flag
CLI		Clear interrupt flag
СМС		Complement carry flag
СМР	dt,sc	Compare
CMPS	[dt,sc]	Compare string
CMPSB	[dt,sc]	Compare bytes
CMPSW	[dt,sc]	Compare words
CWD		Convert word to dble word
DAA		Decimal adjust addition
DAS		Decimal adjust substrac.
DEC	dt	Decrement
DIV	sc	Unsigned divide
ESC	code, sc	Escape
HLT		Halt
IDIV	sc	Integer divide
IMUL	se	Integer multiply
IN	ac,port	Input from port
INC	dt	Increment
INT	type	Interrupt
INTO		Interrupt if overflow
IRET		Return from interrupt
JA	slabel	Jump if above
JAE	slabel	Jump if above or equal
JB	slabel	Jump if below
JBE	slabel	Jump if below or equal
JC	slabel	ump if carry
JCXZ	slabel	Jump if CX is zero
JE	slabel	Jump if equal
JG	slabel	Jump if greater
JGE	slabel	Jump if greater or
		equal

JNB slabel Jump if not below JNBE slabel Jump if below or equal JNC slabel Jump if no carry JNE slabel Jump if not equal JNG slabel Jump if not greater JNGE slabel Jump if not greater or equal JNL slabel Jump if not less			
JMP label Jump if not above JNA slabel Jump if not above or equal JNB slabel Jump if not below JNBE slabel Jump if not below JNBE slabel Jump if not carry JNC slabel Jump if not equal JNC slabel Jump if not greater JNG slabel Jump if not greater JNG slabel Jump if not greater or equal JNG slabel Jump if not less JNL slabel Jump if not less or equal JNZ slabel Jump if not overflow JNZ slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LOOP slabel Loop LOOPE slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSB [dt,sc] Move words	JL	slabel	Jump if less
JNA slabel Jump if not above JNB slabel Jump if not above or equal JNB slabel Jump if not below JNBE slabel Jump if not below or equal JNC slabel Jump if no carry JNE slabel Jump if not greater JNG slabel Jump if not greater or equal JNG slabel Jump if not greater or equal JNL slabel Jump if not less or equal JNL slabel Jump if not less or equal JNZ slabel Jump if not overflow JNP slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LOAD AM from flags LOAD Horse Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LOOP slabel Loop LOOPE slabel Loop LOOPE slabel Loop if equal LOOPI slabel Loop if reto LOOPI slabel Loop if not equal LOOPI slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words	JLE	slabel	Jump if less or equal
JNAE slabel Jump if not above or equal JNB slabel Jump if not below JNBE slabel Jump if not carry JNE slabel Jump if no carry JNE slabel Jump if not greater JNG slabel Jump if not greater or equal JNG slabel Jump if not greater or equal JNL slabel Jump if not less or equal JNZ slabel Jump if not verflow JNZ slabel Jump if not overflow JNP slabel Jump if not overflow JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load string LODS [so] Load string LODS [so] Load words LODP slabel Loop Loop slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move words	JMP	label	Jump
JNB slabel Jump if not below JNBE slabel Jump if below or equal JNC slabel Jump if no carry JNE slabel Jump if not greater JNG slabel Jump if not greater or equal JNG slabel Jump if not greater or equal JNL slabel Jump if not less or equal JNZ slabel Jump if not overflow JNZ slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if sign JZ slabel Jump if sign JZ slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LODP slabel Loop LOOPE slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if requal LOOPX slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move words	JNA	slabel	Jump if not above
JNBE slabel Jump if below or equal JNC slabel Jump if no carry JNE slabel Jump if not equal JNG slabel Jump if not greater JNGE slabel Jump if not greater or equal JNL slabel Jump if not less or equal JNL slabel Jump if not less or equal JNZ slabel Jump if not overflow JNP slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LODP slabel Loop LOOPE slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if reto LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move words	JNAE	slabel	Jump if not above or equa
JNC slabel Jump if no carry JNG slabel Jump if not equal JNG slabel Jump if not greater JNG slabel Jump if not greater or equal JNL slabel Jump if not less JNLE slabel Jump if not less or equal JNZ slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not parity JNS slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LODP slabel Loop LOOPE slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNB	slabel	Jump if not below
JNE slabel Jump if not equal JNG slabel Jump if not greater JNGE slabel Jump if not greater or equal JNL slabel Jump if not less JNLE slabel Jump if not less or equal JNZ slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity JPE slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load string LODS [so] Load string LODSB [so] Load bytes LODS [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNBE	slabel	Jump if below or equal
JNG slabel Jump if not greater or equal JNL slabel Jump if not less JNLE slabel Jump if not less or equal JNZ slabel Jump if not verflow JNP slabel Jump if not overflow JNP slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LOOP slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNC	slabel	Jump if no carry
JNGE slabel Jump if not greater or equal JNL slabel Jump if not less or equal JNZ slabel Jump if not vero Jib slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNE	slabel	Jump if not equal
JNL slabel Jump if not less JNLE slabel Jump if not less or equal JNZ slabel Jump if not overflow JNP slabel Jump if not overflow JNP slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if requal LOOPX slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move words	JNG	slabel	Jump if not greater
JNLE slabel Jump if not less or equal JNZ slabel Jump if not zero Jib slabel Jump if not overflow JNP slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity even JS slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNGE	slabel	Jump if not greater or equa
JNZ slabel Jump if not zero Jib slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LODP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move words MOVSW [dt,sc] Move words	JNL	slabel	Jump if not less
Jib slabel Jump if not overflow JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNLE	slabel	Jump if not less or equal
JNP slabel Jump if not parity JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if not equal LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JNZ	slabel	Jump if not zero
JNS slabel Jump if not sign JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load words LODP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	Jib	slabel	Jump if not overflow
JO slabel Jump if overflow JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSB [dt,sc] Move words	JNP	slabel	Jump if not parity
JPO slabel Jump if parity odd JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move words	JNS	slabel	Jump if not sign
JP slabel Jump if parity JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNZ slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSB [dt,sc] Move words	70	slabel	Jump if overflow
JPE slabel Jump if parity even JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JP0	slabel	Jump if parity odd
JS slabel Jump if sign JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JP	slabel	Jump if parity
JZ slabel Jump if zero LAHF Load AM from flags LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move words	JPE	slabel	Jump if parity even
LAHF LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words		slabel	Jump if sign
LDS dt,sc Load pointer using DS LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	JZ	slabel	Jump if zero
LEA dt,sc Load effective address LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	LAHF		Load AM from flags
LES dt,sc Load pointer using ES LOCK Lock bus LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move bytes MOVSW [dt,sc] Move words	LDS	dt,sc	Load pointer using DS
LOCK LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move words	LEA	dt,sc	Load effective address
LODS [so] Load string LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words		dt,sc	Load pointer using ES
LODSB [so] Load bytes LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words	LOCK		Lock bus
LODSW [so] Load words LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words	LODS	[so]	Load string
LOOP slabel Loop LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			Load bytes
LOOPE slabel Loop if equal LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			Load words
LOOPZ slabel Loop if zero LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			•
LOOPNE slabel Loop if not equal LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words		slabel	
LOOPNZ slabel Loop if not zero MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			
MOV dt,sc Move MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			·
MOVS [dt,sc] Move string MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words			•
MOVSB [dt,sc] Move bytes MOVSW [dt,sc] Move words		•	
MOVSW [dt,sc] Move words			-
	1		•
MUL se Unsigned multiply			
	MUL	se	Unsigned multiply

NEG	dt	Negate
NOP		No operation
NOT	dt	Logical NOT
OR	dt,sc	Logical OR
OUT	port,ac	output to port
POP	dt	Pop word off stack
POPF		Pop flags off stack
PUSH	sc	Push word onto stack
PUSHF		Push flags onto stack
RCL	dt,cnt	Rotate left through carry
RCR	dt,cnt	Rotate right through carry
REP		Repeat string operation
REPE		Repeat while equal
REPZ		Repeat while zero
REPNE		Repeat while not equal
REPNZ		Repeat while not zero
RET	[pop]	Return from procedure
ROL	dt,cnt	Rotate left
ROR	dt,cnt	Rotate right
SAME'		Store AH into flags
SAL	dt,cnt	Shift arithmetic left
SHL	dt,cnt	Shift logical left
SAR	dt,cnt	Shift arithmetic right
SHE	dt,sc	Subtract with borrow
SCAS	[dt]	Scan string
SCASB	[dt]	Scan byte
SCASW	[dt]"	Scan word
SHR	dt,cnt	Shift logical right
STC		Set carry flag
STD		Set direction flag
STI		Set interrupt flag
ST0S	[dt]	Store string
STOSH	[dt]	Store byte
ST0SW	[dt]	Store word
SUB	dt,sc	Substraction
TEST	dt,sc	Test (logical AND)
WAIT		Wait for 8087
XCHG	dt,sc	Exchange
XLAT	table	Translate
XLATB	table	Translate
XOR	dt,sc	Logical exclusive OR