Command	Detail	Technical	Opcode	Clocks	Bytes	O D I T S Z A P C	Addressing modes
AAA	ASCII Adjust for Addition (For BCD)		37	4	1	U - - - U U X U X	
AAD	ASCII Adjust for Division		D5 0A	60	2	U - - - X X U X U	
AAM	ASCII Adjust for Multiply		D4 0A	83	1	U - - - X X U X U	
AAS	ASCII Adjust for Subtraction		3F	4	1	U - - - U U X U X	
ADC dest, src	Add with Carry	ADC AX,SI		3+	2-6	X - - - X X X X X	r,r r,m m,r r,i m,i a,i
ADD dest,src	The sum of the two operands	ADD CX,DX		3+	2-6	X - - - X X X X X	r,r r,m m,r r,i m,i ,a,i
AND dest,src	logical and	AND AL,BL		3+	2-6	0 - - - X X U X 0	r,r r,m m,r r,i m,i a,i
CALL procedure-name	call sub	CALL label… CALL AX		19+	2-5	- - - - - - - - -	II III memptr regptr memptr32
CBW	Convert Byte to Word... Sign Extend AL to AX	CBW	98	2	1	- - - - - - - - -	r,r r,m m,r r,i m,i a,i
CLC	Clear Carry flag	CF=0	F8	2	1	- - - - - - - - 0	
CLD	Clear direction flag	DF=0	FC	2	1	- - 0 - - - - - -	
CLI	Clear Interrupt-enable flag	IF=0	FA	2	1	- - 0 - - - - - -	
CMC	Complement carry flag	CF=!CF	F5	2	1	- - - - - - - - X	
CMP dest, src	Compare	CMP ah,40… CMP bx,cx		3+	2-6	X - - - X X X X X	r,r r,m m,r r,i m,i a,i
CMPS	Compare String... DS:SI to ES:DI… result in ZF	REPE CMPS 10,KEY	A6/A7	22+	1	X - - - X X X X X	ds,ss
CWD	Conv Word to DblW... sign extend AX to DX:AX	DX:AX=AX		5	1	- - - - - - - - -	
DAA	Decimal Adjust for Addition		27	4	1	X - - - X X X X X	
DAS	Decimal Adjust for Subtraction			4	1	U - - - X X X X X	
DEC dest	Decrement			2	1	X - - - X X X X -	rr r m
DIV source	divide			80+	2-5	U - - - U U U U U	r rr m mm
ESC externa/-opcode, src	Escape (for external cpu) (0-63)			2+	2-4	- - - - - - - - -	i,m i,r
HLT	Halt (stop CPU until reset / nmi)		F4	2	1	- - - - - - - - -	
IDIV source	Integer Divide	IDIV BL... IDIV CX		101+	2-4	U - - - U U U U U	r rr m mm
IMUL source	Integer Multiply			80+	2-4	X - - - U U U U U	r rr m mm
IN accumulator,port	Input byte or word (use DX for 16 bit port)	in ax,dx… in ax,1Fh		8+	1-2	- - - - - - - - -	a,i a,DX
INC dest	Increment			2+	1-4	X - - - X X X X -	rr r m
INT interrupt-type	Interrupt (0-255)			51/52	1-2	- - 0 0 - - - - -	i
INTO	Interrupt if overflow flag set			61	1	- - 0 0 - - - - -	
INTR*	External Maskable Interrupt			61	0?!	- - 0 0 - - - - -	
IRET	Interrupt Return		CF	24	1	R R R R R R R R R	
JA/JNBE short-label	above/not below nor equal	(CF OR ZF)=O		4/16	2	- - - - - - - - -	I
JAE/JNB short-label	above or equal/ not below	(CF OR ZF)=O		4/16	2	- - - - - - - - -	I
JB/JNAE short-label	below / not above nor equal	CF=1		4/16	2	- - - - - - - - -	I
JBE/JNA short-label	below or equal/ not above	(CF OR ZF)=1		4/16	2	- - - - - - - - -	I
JC short-label	carry	CF=1		4/16	2	- - - - - - - - -	I
JCXZ short-label	Jump If CX Zero (see loop)			4/16	2	- - - - - - - - -	I
JE/JZ short-label	equal/zero	ZF=1		4/16	2	- - - - - - - - -	I
JG/JNLE short-label	greater/ not less nor equal	((SF XOR OF) OR ZF)=O		4/16	2	- - - - - - - - -	I
JGE/JNL short-label	greater or equal/not less	(SF XOR OF)=O		4/16	2	- - - - - - - - -	I
JLE/JNG short-label	less or equal/ not greater	((SF XOR OF) OR ZF)=1		4/16	2	- - - - - - - - -	I
JLIJNGE short-label	less/not greater nor equal	(SF XOR OF)=1		4/16	2	- - - - - - - - -	I
JMP target	Jump to label (byte or word target)			11+	2	- - - - - - - - -	I II III memptr regptr memptr32
JNC short-label	not carry	CF=O		4/16	2	- - - - - - - - -	I
JNE/JNZ short-label	not equal/ not zero	ZF=O		4/16	2	- - - - - - - - -	I
JNO short-label	not overflow	OF=O		4/16	2	- - - - - - - - -	I
JNP/JPO short-label	not parity / parity odd	PF=O		4/16	2	- - - - - - - - -	I
JNS short-label	not sign	SF=O		4/16	2	- - - - - - - - -	I
JO short-label	overflow	OF=1		4/16	2	- - - - - - - - -	I
JP/JPE short-label	parity/ parity equal	PF=1		4/16	2	- - - - - - - - -	I
JS short-label	sign	SF=1		4/16	2	- - - - - - - - -	I
LAHF	Load AH from flags (see SAHF)	AH=F		4	1	- - - - - - - - -	
LDS dest,source	Load Pointer using DS(Set DS and Dest)	lds di,[es:si]		16+	2-4	- - - - - - - - -	rr,mmm
LEA dest, source	Load effective address		8D n	2+	2-4	- - - - - - - - -	rr,mm
LES dest, source	Load Pointer using ES			16+	2-4	- - - - - - - - -	rr,mmm
LOCK	Lock bus signal		F0	2	1	- - - - - - - - -	
LODSB/W	Load String Copy DS:SI to AL/AX		AC/AD	12+	1	- - - - - - - - -	ss
LOOP short-label	dec CX and loop if CX!=0 (See JCXZ)		E2	17/5	2	- - - - - - - - -	I
LOOPNZ short-label (or LOOPNE)	dec CX and loop if nz		E0	19/5	2	- - - - - - - - -	I
LOOPZ short-label (or LOOPE)	dec CX and loop if z		E1	18/6	2	- - - - - - - - -	I
MOV dest,source	Move			2+	2-6	- - - - - - - - -	m,a a,m r,r r,m m,r r,i m,i sr,rr sr,mm rr,sr mm,sr
MOVSB/W (like LDI)	Move String	Move DS:SI to ES:DI	A4/A5	18+	1	- - - - - - - - -	ds,ss
MUL source	Multiply AX= AL*byte / DX:AX = AX*word	MUL BL... MUL addr		70+	2-4	X - - - U U U U X	r rr m mm
NMI*				50	0?!	- - 0 0 - - - - -	
NEG dest	Negate	NEG AL.. NEG BX... NEG addr		3+	2-4	X - - - X X X 1*	r rr m
NOP	No Operation		90	3	1	- - - - - - - - -	
NOT dest	inverts the bits	NOT AL... NOT AX... NOT addr		3+	2-4	- - - - - - - - -	r rr m
OR dest,src	inclusive or			3+	2-6	0 - - - X X U X 0	r,r r,m m,r r,i m,i a,i
OUT port,accumulator	Output byte or word (use DX for 16 bit port)	out dx,ax… out 1Fh,ax		8-10	1-2	- - - - - - - - -	i,a DX,a
POP dest	Pop a word off the stack	POP DX... POP DX... POP WORD PTR [addr]		8+	1-4	- - - - - - - - -	rr sr mm
POPF	Pop flags off stack		9D	8-10	1	R R R R R R R R R	
PUSH source	Push word onto stack	PUSH DX... PUSH DX... PUSH WORD PTR [addr]		10+	1-4	- - - - - - - - -	rr sr mm
PUSHF	Push flags onto stack		9C	10	1	- - - - - - - - -	
RCL dest, count	Rotate through Carry Left	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
RCR dest, count	Rotate through Carry Right	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
REP cmd	Repeat cmd CX times	Cmd = INS/LODS/MOVS/OUTS/STOS	F3	2	1	- - - - - - - - -	
REPE / **REPZ** cmd	Repeat While Zero	cmd =CMPS/SCAS	F3	2	1	- - - - - - - - -	
REPNE / REPNZ cmd	Repeat While Not Zero	cmd =CMPS/SCAS	F2	2	1	- - - - - - - - -	
RET optional-pop-value	Return from sub (and pop n bytes off stack)	N= 0-64k		8-17	1-3	- - - - - - - - -	- i ii
ROL dest, count	Rotate Left (no carry)	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
ROR dest, count	Rotate Right (no carry)	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
SAHF	Store AH into flags (see LAHF)		9E	4	1	- - - - R R R R R	
SAL dest, count	Shift Arithmetic Left	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
SAR dest, count	Shift Arithmetic Right	count must be 1 on 086		2+	2-4	X - - - X X U X X	r,1 r,CL m,1 m,CL
SBB dest, src	Subtract with Borrow			3	2-6	X - - - X X X X X	r,r r,m m,r a,i r,i m,i
SCASB/W	Scan String... cmp AL/AX to ES:DI… result in ZF	SCAS	AE/AF	15+	1	X - - - X X X X X	ds
Segment*	Override to specific segment	MOV SS:parameter,AX		2	1	- - - - - - - - -	
SHL dest, count	Shift Logical Left	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
SHR dest, src	Shift Logical Right	count must be 1 on 086		2+	2-4	X - - - - - - - X	r,1 r,CL m,1 m,CL
SINGLE STEP	Trap flag interrupt	Interrupt if TF=1		2+	2-4	- - 0 0 - - - - -	
STC	Set Carry Flag	CF=1	F9	2	1	- - - - - - - - 1	
STD	Set direction flag	DF=0	FD	2	1	- - 1 - - - - - -	
STI	Set interrupt enable flag	IF=1	FB	2	1	- - 1 - - - - - -	
STOSB/W dest-string	Store String Byte/Word in ES:DI	STOS	AA/AB	11+	1	- - - - - - - - -	ds
SUB dest,src	Subtraction	SUB CX,BX		3+	2-6	X - - - X X X X X	r,r r,m m,r a,i r,i m,i
TEST dest, src	sets flags like AND without changing regs		9B	3+	2-6	0 - - - X X U X 0	r,r r,m m,r a,i r,i m,i
WAIT	Wait while test/busy not active (high)			3+5n	1	- - - - - - - - -	
XCHG dest,src	Exchange byte/word register	XCHG AX,BX... XCHG AL,BL... XCHG addr,AX		3+	1-4	- - - - - - - - -	a,r m,r r,r
XLAT	Translate (Read from LUT)... AL = [DS:BX+AL]	XLAT	D7	11	1	- - - - - - - - -	
XOR dest, src	Exclusive Or			3+	2-6	0 - - - X X U X 0	r,r r,m m,r a,i r,i m,i

186 Commands	Detail	Technical	Opcode	Clocks	Bytes	O D I T S Z A P C	Addressing modes
BOUND	Check Array Index Against Bounds		62	10		- - - - - Z - - -	
ENTER	Make Stack Frame for Procedure Parameters		C8	10/15+4		- - - - - - - - -	
INS	Input from Port to String		6C	9/29		- - - - - - - - -	
INSB	Input from Port to String		6D	9/29		- - - - - - - - -	
INSW	Input from Port to String		6D	9/29		- - - - - - - - -	
LEAVE	High Level Procedure Exit		C9			- - - - - - - - -	
OUTS	Output String to Port			8/28		- - - - - - - - -	
OUTSB	Output String to Port			8/28		- - - - - - - - -	
OUTSW	Output String to Port			8/28		- - - - - - - - -	
POPA	Pop all General Registers	DI,SI,BP,skipped,BX,DX,CX,AX		24		- - - - - - - - -	
PUSHA	Push all General Registers	AX,CX,DX,BX,SP,BP,SI,DI	60	18		- - - - - - - - -	
PUSHW							

Mnemonic	Description	Example	Valid Regs	Flags affected
AAA	ASCII Adjust for Addition. Treats AL as an unpacked binary coded decimal number	AAA		o s z A p C
AAD	ASCII Adjust for Division. AL=AL+(AH*10), AH=0.	AAD		o S Z a P c
AAM	ASCII Adjust for Multiplication. We can use the normal MUL command then use AAM	AAM		o S Z a P c
AAS	ASCII Adjust for Subtraction. This treats AL as an unpacked binary coded decimal number	AAS		o s z A p C
ADC dest,src	Add src and the carry flag to dest.	ADC CX,1000h		O S Z A P C
ADD dest,src	Add src to dest.	ADD CX,1000h		O S Z A P C
AND dest,src	Logical AND of bits in dest with Accumulator scr.	AND AX,1100h		O S Z A P C
CALL dest	Call Subroutine at address dest.	CALL 1000h		- - - - - - -
CBW	Convert the 8 bit byte in AL into a 16 bit word in AX.	CBW		- - - - - - -
CLC	Clear the Carry Flag. C flag will be set to Zero.	CLC		- - - - - - C
CLD	Clear the Direction Flag. D flag will be set to Zero. This is used for 'String functions'.	CLD		D - - - - - -
CLI	Clear the Interrupt enable flag. I flag will be set to 0. This disables maskable interrupts.	CLI		I - - - - - -
CMC	Complement the Carry flag. If C=1 it will now be 0. If it was 0 it will now be 1.	CMC		- - - - - - C
CMP dest,src	Compare the Byte or Word dest to src. This sets the flags the same as "SUB dest,src" would.	CMP AL,32		O S Z A P C
CMPSBCMPSW	Compare DS:SI to ES:DI. This command can work in bytes or words. Sets flags like CMP	REPZ CMPSB		O S Z A P C
CWD	Convert the 16 bit word in AX into a 32 bit doubleword in DX.AX. This 'Sign Extends' AX	CWD		- - - - - - -
DAA	Decimal Adjust for Addition. This treats AL as a packed binary coded decimal number.	DAA		O S Z A P C
DAS	Decimal Adjust for Subtraction. This treats AL as a packed binary coded decimal number.	DAS		O S Z A P C
DEC Dest	Divide Unsigned number AX or DX.AX by src.	DEC AL		O S Z A P -
DIV src	Divide Unsigned number AX or DX.AX by src. AL=AX/src (8 bit) or AX=DX.AX/src (16 bit)	DIV CX		o s z a p c
ESC #,src	This command is for working with multiple processors - it's not something you will need.	ESC 1,AH		- - - - - - -
HLT	Stop the CPU until an interrupt occurs	HLT		- - - - - - -
IDIV src	Divide Signed number AX or DX.AX by src. AL=AX / src (8 bit) or AX=DX.AX / src (16 bit)	IDIV CX		o s z a p c
IMUL src	Multiply Signed number AX or DX.AX by src. AX=AL*src (8 bit) or DX.AX=AX*src (16 bit)	IMUL CX		O s z a p C
IN dest,port	Read in an 8 bit byte or 16 bit word into dest (either AX, AL or AH). Use DX for 16 bit port num	IN AX,F0h		- - - - - - -
INC Dest	Increase Dest by one. This is faster than using ADD with a value of 1.	INC AL		O S Z A P -
INT #	Causes software interrupt #. The flags are pushed onto the stack before call	INT 33h		- - - - - - -
INTO	INTO will cause Interrupt 4 if the Overflow flag (O) is set, otherwise it will have no effect.	INTO		- - - - - - -
IRET	Restore the flags from the stack and return from an Interrupt.	IRET		O S Z A P C
Jcc addr	Jump to 8 bit offset addr if condition cc is true.	JO ErrorHandle		- - - - - - -
JCXZ addr	Jump to 8 bit offset addr if CX=0.	JCXZ NoLoop		- - - - - - -
JMP addr	Jump to address addr.	JMP BX		- - - - - - -
LAHF	Load AH from the Flags. This only transfers the main flags: SZ-A-P-C	LAHF		- - - - - - -
LDS reg,addr	Load a full 32 bit pointer into DS segment register and register reg.	LDS BX,TestPointer	AX, BX, CX, DX, SI, DI	- - - - - - -
LEA reg,src	Load the effective address src into reg.	LEA CX,[BX+DI]	AX, BX, CX, DX,SI, DI,	- - - - - - -
LES reg,addr	Load a full 32 bit pointer into ES segment register and register reg.	LES AX,MyLabel	AX,BX,CX,DX,SI,DI	- - - - - - -
LOCK	Enable the LOCK signal. This is for multiprocessor systems.	LOCK		- - - - - - -
LODSBLODSW	Load from DS:SI into AX or AL. This command can work in bytes or words.	LODSB		- - - - - - -
LOOP addr	Decrease CX and jump to label addr if CX is not zero.	LOOP LoopLabel		- - - - - - -
LOOPNZ addr **LOOPNE addr**	Decrease CX and jump to label addr if CX is not zero and the Zero flag is not set.	LOOPNZ LoopLabel		- - - - - - -
LOOPZ addr **LOOPE addr**	Decrease CX and jump to label addr if CX is not zero and the Zero flag is set.	LOOPZ LoopLabel		- - - - - - -
MOVSB **MOVSW**	Move a byte or word from DS:SI to ES:DI. This command can be combined with repeat command REP, to repeat CX times.	REPZ MOVSB		- - - - - - -
MUL src	Multiply unsigned number AX or DX.AX by src.AX=AL*src (8 bit) or DX.AX=AX*src (16 bit)	MUL CX		O s z a p C
NEG dest	Negate dest (Twos Complement of the number).	NEG AL		- - - - - - -
NOP	No Operation. This command has no effect on any registers or memory.	NOP		- - - - - - -
NOT dest	Invert/Flip all the bits of dest.	NOT dest		- - - - - - -
OR dest,src	Logically ORs the src and dest parameter together.	OR AX,BX		O S Z a P C
OUT port,src	Send an 8 bit byte or 16 bit word from src (either AX, AL or AH) to hardware port number port.	OUT 100,AL		- - - - - - -
POP reg	Pop a pair of bytes off the stack into 16 bit register reg.	POP ES	AX, BX, CX, DX, SI, DI	- - - - - - -
POPF	Pop a pair of bytes off the stack into the 16 bit Flags register.	POPF		O D I T S Z A P C
PUSH reg	Push a pair of bytes from 16 bit register reg onto the top of the stack.	PUSH AX		- - - - - - -
PUSHF	Push a pair of bytes off the stack into the 16 bit Flags register.	PUSHF		- - - - - - -
RCL dest,count	Rotate bits in Destination dest to the Left by count bits, with the carry flag acting as an extra bit.	RCL AX,1		O - - - - - C
RCR dest,count	Rotate bits in Destination dest to the Right by count bits, with the carry flag acting as an extra bit	RCR AX,1		O - - - - - C
REP stringop	Repeat string operation stringop while CX>0. Decrease CX after each iteration	REP LODSW		- - - - - - -
REPE stringop **REPZ stringop**	Repeat string operation stringop while the Z flag is set and CX>0. Decrease CX each time	REPZ CMPSB		- - - - - - -
REPNE stringop	Repeat string operation stringop while the Z flag is not set and CX>0. Decrease CX each time	REPNZ CMPSB		- - - - - - -
RET	Return from a subroutine.	RET		- - - - - - -
ROL dest,count	Rotate bits in Destination dest to the Left by count bits	ROL AX,1		O - - - - - C
ROR dest,count	Rotate bits in Destination dest to the Right by count bits	ROR AL,1		O - - - - - C
SAHF	Store AH to the Flags. This only transfers the main flags: SZ-A-P-C .	SAHF		- S Z A P C
SAL dest,count	Shift the bits for Arithmetic in Destination dest to the Left by count bits.	SAL AX,1		O - - - - - C
SAR dest,count	Shift the bits for Arithmetic in Destination dest to the Right by count bits.	SAR AX,1		O - - - - - C
SBB dest,src	Subtract src and the Borrow (carry flag) from dest.	SBB AL,BL		O S Z A P C
SCASBSCASW	Scan ES:DI and compare to AX or AL. This command can work in bytes or words. (Like CMP)	REPZ SCASB		O S Z A P C
SHL dest,count	Shift the bits logically Left in destination dest by count bits.	SHL AX,1		O - - - - - C
SHR dest,count	Shift the bits logically Right in destination dest by count bits.	SHR AX,1		O - - - - - C
STC	Set the Carry Flag. C flag will be set to 1.	STC		- - - - - - C
STD	Set the Direction Flag. D flag will be set to 1. This is used for 'String functions'.	STD		D - - - - - - -
STI	Set the Interrupt enable flag. I flag will be set to 1. This enables maskable interrupts.	STI		I - - - - - - -
STOSBSTOSW	Store AX or AL to ES:DI. This command can work in bytes or words.	REP STOSB		- - - - - - -
SUB dest,src	Subtract src from dest.	SUB AX,BX		O S Z A P C
TEST dest,src	Test dest, setting the flags in the same way a logical "AND src" would. Dest unchanged	TEST BX,64h		O S Z A P C
WAIT	Wait until the busy pin of the CPU is inactive.	WAIT		O S Z A P C
XCHG reg1,reg2	Exchange the contents of registers reg1 and reg2.	XCHG BH,AL		- - - - -
XLAT	Translate AL using lookup table DS:BX. AL is read from memory address [DS:BX+AL].	XLAT		- - - - - - -