
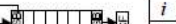


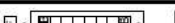



TRANSFER				Flags							
Name	Comment	Code	Operation	O	D	I	T	S	Z	A	P C
MOV	Move (copy)	MOV Dest,Source	Dest:=Source								
XCHG	Exchange	XCHG Op1,Op2	Op1:=Op2 , Op2:=Op1								
STC	Set Carry	STC	CF:=1								1
CLC	Clear Carry	CLC	CF:=0								0
CMC	Complement Carry	CMC	CF:=¬CF								±
STD	Set Direction	STD	DF:=1 (string op s downwards)		1						
CLD	Clear Direction	CLD	DF:=0 (string op s upwards)		0						
STI	Set Interrupt	STI	IF:=1			1					
CLI	Clear Interrupt	CLI	IF:=0			0					
PUSH	Push onto stack	PUSH Source	DEC SP, [SP]:=Source								
PUSHF	Push flags	PUSHF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL								
PUSHA	Push all general registers	PUSHA	AX, CX, DX, BX, SP, BP, SI, DI								
POP	Pop from stack	POP Dest	Dest:=[SP], INC SP								
POPF	Pop flags	POPF	O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL	±	±	±	±	±	±	±	±
POPA	Pop all general registers	POPA	DI, SI, BP, SP, BX, DX, CX, AX								
CBW	Convert byte to word	CBW	AX:=AL (signed)								
CWD	Convert word to double	CWD	DX:AX:=AX (signed)	±					±	±	±
CQDE	Conv word extendeddouble	CQDE 386	EAX:=AX (signed)								
IN <i>i</i>	Input	IN Dest, Port	AL/AX/EAX := byte/word/double of specified port								
OUT <i>i</i>	Output	OUT Port, Source	Byte/word/double of specified port := AL/AX/EAX								

*i* for more informations see instruction specifications


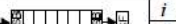
Flags: ±=affected by this instruction ?=undefined after this instruction

ARITHMETIC				Flags							
Name	Comment	Code	Operation	O	D	I	T	S	Z	A	P C
ADD	Add	ADD Dest,Source	Dest:=Dest+Source	±					±	±	±
ADC	Add with Carry	ADC Dest,Source	Dest:=Dest+Source+CF	±					±	±	±
SUB	Subtract	SUB Dest,Source	Dest:=Dest-Source	±					±	±	±
SBB	Subtract with borrow	SBB Dest,Source	Dest:=Dest-(Source+CF)	±					±	±	±
DIV	Divide (unsigned)	DIV Op	Op=byte: AL:=AX / Op AH:=Rest	?					?	?	?
DIV	Divide (unsigned)	DIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?					?	?	?
DIV 386	Divide (unsigned)	DIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?					?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=byte: AL:=AX / Op AH:=Rest	?					?	?	?
IDIV	Signed Integer Divide	IDIV Op	Op=word: AX:=DX:AX / Op DX:=Rest	?					?	?	?
IDIV 386	Signed Integer Divide	IDIV Op	Op=doublew.: EAX:=EDX:EAX / Op EDX:=Rest	?					?	?	?
MUL	Multiply (unsigned)	MUL Op	Op=byte: AX:=AL*Op if AH=0 ♦	±					?	?	?
MUL	Multiply (unsigned)	MUL Op	Op=word: DX:AX:=AX*Op if DX=0 ♦	±					?	?	?
MUL 386	Multiply (unsigned)	MUL Op	Op=double: EDX:EAX:=EAX*Op if EDX=0 ♦	±					?	?	?
IMUL <i>i</i>	Signed Integer Multiply	IMUL Op	Op=byte: AX:=AL*Op if AL sufficient ♦	±					?	?	?
IMUL	Signed Integer Multiply	IMUL Op	Op=word: DX:AX:=AX*Op if AX sufficient ♦	±					?	?	?
IMUL 386	Signed Integer Multiply	IMUL Op	Op=double: EDX:EAX:=EAX*Op if EAX sufficient ♦	±					?	?	?
INC	Increment	INC Op	Op:=Op+1 (Carry not affected !)	±					±	±	±
DEC	Decrement	DEC Op	Op:=Op-1 (Carry not affected !)	±					±	±	±

CMP	Compare	CMP Op1,Op2	Op1-Op2	±					±	±	±
SAL	Shift arithmetic left (=SHL)	SAL Op,Quantity		<i>i</i>					±	±	?
SAR	Shift arithmetic right	SAR Op,Quantity		<i>i</i>					±	±	?
RCL	Rotate left through Carry	RCL Op,Quantity		<i>i</i>							±
RCR	Rotate right through Carry	RCR Op,Quantity		<i>i</i>							±
ROL	Rotate left	ROL Op,Quantity		<i>i</i>							±
ROR	Rotate right	ROR Op,Quantity		<i>i</i>							±

*i* for more informations see instruction specifications

♦ then CF:=0, OF:=0 else CF:=1, OF:=1

LOGIC				Flags							
Name	Comment	Code	Operation	O	D	I	T	S	Z	A	P C
NEG	Negate (two-complement)	NEG Op	Op:=0-Op if Op=0 then CF:=0 else CF:=1	±					±	±	±
NOT	Invert each bit	NOT Op	Op:=¬Op (invert each bit)								
AND	Logical and	AND Dest,Source	Dest:=Dest∧Source	0					±	±	?
OR	Logical or	OR Dest,Source	Dest:=Dest∨Source	0					±	±	?
XOR	Logical exclusive or	XOR Dest,Source	Dest:=Dest⊕Source	0					±	±	?
SHL	Shift logical left (=SAL)	SHL Op,Quantity		<i>i</i>					±	±	?
SHR	Shift logical right	SHR Op,Quantity		<i>i</i>					±	±	?

This page may be freely distributed without cost provided it is not changed. Send comments to roger@jegerlehner.ch - www.jegerlehner.ch/intel/

MISCELLANEOUS				Flags								
Name	Comment	Code	Operation	O	D	I	T	S	Z	A	P	C
NOP	No operation	NOP	No operation									
LEA	Load effective adress	LEA Dest,Source	Dest := address of Source									
INT	Interrupt	INT Nr	interrupts current program, runs spec. int-program			0	0					

JUMPS (flags remain unchanged)				Name	Comment	Code	Operation
Name	Comment	Code	Operation	Name	Comment	Code	Operation
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(= JZ)	JNE	Jump if not Equal	JNE Dest	(= JNZ)
JZ	Jump if Zero	JZ Dest	(= JE)	JNZ	Jump if not Zero	JNZ Dest	(= JNE)
JCXZ	Jump if CX Zero	JCXZ Dest		JECXZ	Jump if ECX Zero	JECXZ Dest	386
JP	Jump if Parity (Parity Even)	JP Dest	(= JPE)	JNP	Jump if no Parity (Parity Odd)	JNP Dest	(= JPO)
JPE	Jump if Parity Even	JPE Dest	(= JP)	JPO	Jump if Parity Odd	JPO Dest	(= JNP)

Unsigned (Cardinal)				Signed (Integer)			
JA	Jump if Above	JA Dest	(= JNBE)	JG	Jump if Greater	JG Dest	(= JNLE)
JAE	Jump if Above or Equal	JAE Dest	(= JNB = JNC)	JGE	Jump if Greater or Equal	JGE Dest	(= JNL)
JB	Jump if Below	JB Dest	(= JNAE = JC)	JL	Jump if Less	JL Dest	(= JNGE)
JBE	Jump if Below or Equal	JBE Dest	(= JNA)	JLE	Jump if Less or Equal	JLE Dest	(= JNG)
JNA	Jump if not Above	JNA Dest	(= JBE)	JNG	Jump if not Greater	JNG Dest	(= JLE)
JNAE	Jump if not Above or Equal	JNAE Dest	(= JB = JC)	JNGE	Jump if not Greater or Equal	JNGE Dest	(= JL)
JNB	Jump if not Below	JNB Dest	(= JAE = JNC)	JNL	Jump if not Less	JNL Dest	(= JGE)
JNBE	Jump if not Below or Equal	JNBE Dest	(= JA)	JNLE	Jump if not Less or Equal	JNLE Dest	(= JG)
JC	Jump if Carry	JC Dest		JO	Jump if Overflow	JO Dest	
JNC	Jump if no Carry	JNC Dest		JNO	Jump if no Overflow	JNO Dest	
				JS	Jump if Sign (= negative)	JS Dest	
				JNS	Jump if no Sign (= positive)	JNS Dest	

INSTRUÇÃO	DESTINO	ORIGEM	COM BYTE	COM WORD
MOVE VETOR	ES:DI	DS:SI	MOVSB	MOVSW
COMPARA VETOR	ES:DI	DS:SI	CMPSB	CMPSW
ARMAZENA VETOR	ES:DI	AL ou AX	STOSB	STOSW
CARREGA VETOR	AL ou AX	DS:SI	LODSB	LODSW
PROCURA VETOR	ES:DI	AL ou AX	SCASB	SCASW