ニーモニック別命令一覧表

コマンド名	意　味	オペランド	使　用　例
AAA	ASCII Adjust for Addition	————	AAA
AAD	ASCII Adjust for Division	————	AAD
AAM	ASCII Adjust for Multiply	————	AAM
AAS	ASCII Adjust for Subtraction	————	AAS
ADC	Add with Carry	reg , reg reg , mem mem, reg reg , imm mem, imm acc , imm	ADC AX, SI ADC DX, [SI+100H] ADC [BX], DI ADC BX, 1024 ADC BYTE PTR [BX], 30H ADC AL, 5

ADD	Addition	reg , reg
		reg , mem
		mem, reg
		reg , imm
		mem, imm
		acc , imm
			ADD CX, DX
			ADD DI, [BX+100H]
			ADD [1000H], CL
			ADD CL, 2
			ADD BY [BX], 2
			ADD AX, 200H

AND	Logical And	reg , reg reg , mem mem, reg reg , imm mem, imm acc , imm	AND AL, BL AND CX, [123H] AND [00FFH+DI], AL AND CX, 0F0H AND BY [BX+5], 01H AND AX, 01010001B
CALL	Call a procedure	near-proc far-proc mem 16 reg 16 mem 32	CALL (NEAR) 100H CALL FAR 2000:3000H CALL [BX] CALL AX CALL FAR [1000]
CBW	Convert Byte to Word	————	CBW
CLC	Clear Carry flag	————	CLC
CLD	Clear Direction flag	————	CLD
CLI	Clear Interrupt flag	————	CLI
CMC	Complement Carry flag	————	CMC

CMP	Compare
	destination to source
		reg , reg
		reg , mem
		mem, reg
		reg , imm
		mem, imm
		acc , imm
			CMP BX, CX
			CMP DH, [11E0H]
			CMP [BP+2], SI
			CMP BL, 02H
			CMP WORD PTR [BX+DI], 3420H
			CMP AL, 00010111B

CMPS /CMPSB /CMPSW	Compare String /for Byte /for Word	(dst-string, src-string) (repeat dst-string, src-string)	CMPSB REPE CMPSW
CWD	Convert Word to Doubleword	————	CWD
DAA	Decimal Adjust for Addition	————	DAA
DAS	Desimal Adjust for Subtraction	————	DAS

コマンド名	意　味	オペランド	使　用　例
DEC	Decrememt by 1	reg 16 reg 8 mem	DEC AX DEC AL DEC BY [1000H+SI]
DIV	Division, unsigned	reg 8 reg 16 mem 8 mem 16	DIV CL DIV BX DIV BY [100H] DIV WO [300H+SI]
ESC	Escape	imm, mem imm, reg	ESC 6, [SI] ESC 20, AL
HLT	Halt	————	HLT
IDIV	Integer Division	reg 8 reg 16 mem 8 mem 16	IDIV BL IDIV CX IDIV BY [1000H] IDIV WO [BX]
IMUL	Integer Multiplication	reg 8 reg 16 mem 8 mem 16	IMUL CL IMUL BX IMUL BY [1000H] IMUL WO [BX]
IN	Input byte or word	acc, imm 16 acc, DX	IN AL, 0FFEAH IN AX, DX
INC	Increment by 1	reg 16 reg 8 mem	INC CX INC BL INC BY [1000H]
INT	Interrupt	imm 8 (type＝3) imm 8 (type≠3)	INT 3 INT 67H
INTO	Interrupt if Overflow	————	INTO
IRET	Interrupt Return	————	IRET
JA /JNBE	Jump if Above /Jump if Not Below nor Equal	short-label	JA short-label
JAE /JNB	Jump if Above or Equal /Jump if Not Below	short-label	JAE short-label
JB /JNAE	Jump if Below /Jump if Not Above nor Equal	short-label	JB short-label
JBE /JNA	Jump if Below or Equal /Jump if Not Above	short-label	JNA short-label
JC	Jump if Carry	short-label	JC short-label
JCXZ	Jump if CX is Zero	short-label	CXZ short-label
JE/JZ	Jump if Equal /Jump if Zero	short-label	JZ short-label
JG /JLNE	Jump if Greater /Jump if Not Less nor Equal	short-label	JG short-label
JGE /JNL	Jump if Greater or Equal /Jump if Not Less	short-label	JGE short-label
JL /JNGE	Jump if Less /Jump if Not Greater nor Equal	short-label	JL short-label
JLE /JNG	Jump if Less or Equal /Jump if Not Greater	short-label	JLE short-label
JMP	Jump	short-label near-label far-label mem 16 reg 16 mem 32	JMP short-label JMP near-label JMP far-label JMP [BX] JMP CX JMP FAR [SI+100H]
JNC	Jump if Not Carry	short-label	JNC short-label
JNE /JNZ	Jump if Not Equal /Jump if Not Zero	short-label	JNE short-label
JNO	Jump if Not Overflow	short-label	JNO short-label

コマンド名	意　味	オペランド	使　用　例
JNP /JPO	Jump if Not Parity /Jump if Parity Odd	short-label	JPO short-label
JNS	Jump if Not Sign	short-label	JNS short-label
JO	Jump if Overflow	short-label	JO short-label
JP /JPE	Jump if Parity /Jump if Parity Even	short-label	JPE short-label
JS	Jump if Sign	short-label	JS short-label
LAHF	Load AH from Flags	————	LAHF
LDS	Load pointer using DS	reg 16, mem 32	LDS SI, [DI+5]
LEA	Load Effective Address	reg 16, mem 16	LEA BX, [BP]
LES	Load pointer using ES	reg 16, mem 32	LES DI, [BX+5]
LOCK	Lock bus	————	LOCK
LODS	Load String	(src-string) (repeat src-string)	LODSB REP LODSW
LOOP	Loop	short-label	LOOP short-label
LOOPE /LOOPZ	Loop if Equal /Loop if Zero	short-label	LOOPE short-label
LOOPNE /LOOPNZ	Loop if Not Equal /Loop if Not Zero	short-label	LOOPNE short-label

MOV	Move	mem, acc
		acc , mem
		reg , reg
		reg , mem
		mem, reg
		reg , imm
		mem, imm
		sreg, reg 16
		sreg, mem 16
		reg 16, sreg
		mem, sreg
			MOV [1000H], AL
			MOV AX, [1000H]
			MOV AX, CX
			MOV BP, [200H]
			MOV [200H], CX
			MOV CL, 2
			MOV BY [BX+100], 2CH
			MOV ES, CX
			MOV DS, [100H]
			MOV BP, SS
			MOV [BX], CS

MOVS /MOVSB /MOVSW	Move String /for Byte /for Word	(dst-string, src-string) (repeat dst-string,src-string)	MOVSB REP MOVSW
MUL	Multiplication unsigned	reg 8 reg 16 mem 8 mem 16	MUL BL MUL CX MUL BY [100H] MUL WO [100H]
NEG	Negate	reg mem	NEG AL NEG BY [1234H]
NOP	No Operation	————	NOP
NOT	Logical Not	reg mem	NOT AX NOT WO [1234H]
OR	Logical inclusive OR	reg , reg reg , mem mem, reg acc , imm reg , imm mem, imm	OR AL, BL OR DX, [DI] OR [1234H], CL OR AL, 01101101B OR CX, 0CFH OR BY [BX], 0CFH
OUT	Output byte or word	imm 8, acc DX, acc	OUT 44, AX OUT DX, AL
POP	Pop word off stack	reg sreg (CS illegal) mem 16	POP DX POP DS POP [1000H]
POPF	Pop Flags off stack	————	POPF
PUSH	Push word onto stack	reg sreg (CS illegal) mem 16	PUSH SI PUSH ES PUSH [1000H]
PUSHF	Push Flags onto stack	————	PUSHF

コマンド名	意　味	オペランド	使　用　例
RCL	Rotate Left through Carry	reg , 1 reg , CL mem, 1 mem, CL	RCL CX, 1 RCL AL, CL RCL [BX], 1 RCL BY [BP+100H], CL
RCR	Rotate Right through Carry	reg , 1 reg , CL mem, 1 mem, CL	RCR BX, 1 RCR BL, CL RCR [BX], 1 RCR WO [BP+100H], CL
REP /REPZ	Repeat string operation /while Zero	————	REP MOVSB
REPNE /REPNZ	Repeat string operation while Equal /while Zero	————	REPNE SCASW
RET	Return from procedure	(intra-segment, no pop) (intra-segment, pop) (inter-segment, no pop) (inter-segment, pop)	RET RET 4 RET RET 2
ROL	Rotate Left	reg , 1 reg , CL mem, 1 mem, CL	ROL BX, 1 ROL DI, CL ROL [DI], 1 ROL BY [BP+100H], CL
ROR	Rotate Right	reg , 1 reg , CL mem, 1 mem, CL	ROR AL, 1 ROR BX, CL ROR [BX], 1 ROR WO [BP+100H], CL
SAHF	Store AH into Flags	————	SAHF
SAL /SHL	Shift Arithmetic Left /Shift logical Left	reg , 1 reg , CL mem, 1 mem, CL	SAL AL, 1 SHL DI, CL SHL [BX], 1 SAL WO [BP+100H], CL
SAR	Shift Arithmetic Right	reg , 1 reg , CL mem, 1 mem, CL	SAR DX, 1 SAR DI, CL SAR [DI], 1 SAR BY [BP+100H], CL
SBB	Subtract with Borrow	reg , reg reg , mem mem, reg acc , imm reg , imm mem, imm	SBB BX, CX SBB DI, [1000H] SBB [BX], AX SBB AX, 2 SBB CL, 1 SBB BY [SI+100H], 10
SCAS	Scan String	(dst-string) (repeat dst-string)	SCASW REPNE SCASB
SHR	Shift logical Right	reg , 1 reg , CL mem, 1 mem, CL	SHR SI, 1 SHR SI, CL SHR BY [100H], 1 SHR WO [100H], CL
STC	Set Carry flag	————	STC
STD	Set Direction flag	————	STD
STI	Set Interrupt enable flag	————	STI
STOS	Store byte or word String	(dst-string) (repeat dst-string)	STOSB REP STOSW

SUB	Subtraction	reg , reg
		reg , mem
		mem, reg
		acc , imm
		reg , imm
		mem, imm
			SUB CX, BX
			SUB DX, [SI+100H]
			SUB [BP+2], CL
			SUB AL, 10
			SUB SI, 5280
			SUB WO [BP], 1000H

TEST	Test or non-destructive logical and	reg , reg reg , mem acc , imm reg , imm mem, imm	TEST SI, DI TEST SI, [1000H] TEST AL, 00100000B TEST BX, 0CC4H TEST BY [1000H], 01H
WAIT	Wait while $\overline{\text{Test}}$ pin not asserted	————	WAIT
XCHG	Exchange	acc , reg 16 mem, reg reg , reg	XCHG AX, BX XCHG [BX], AX XCHG AL, BL
XLAT	Translate	————	XLAT
XOR	Logical exclusive OR	reg , reg reg , mem mem, reg acc , imm reg , imm mem, imm	XOR AL, AL XOR BL, [1000] XOR [1001], BH XOR AX, AAAAH XOR DX, 5555H XOR [BX], 1234

略語例）　WO: WORD PTR,BY: BYTE PTR