Instructions
du microprocesseur Intel 8086

Instruction Fields

In the descriptions that follow, the "Encoding" field shows the assembled code for the instruction. There are one to four bytes represented, depending on the particular instruction or form of the instruction.

The general format of an instruction is:

operation-code-byte [*addressing-mode-byte* [*disp(s)*]]

The *operation-code* determines if there is an *address-mode-byte* and this in turn determines if *displacement(s)* are added to the end of the instruction.

Operation Code Byte

See the individual instructions.

Addressing Mode Byte

This byte consists of a *mode field*, a *register field*, and a *register/memory field*.

Mode Field

If mod=00, then disp=0, disp-low and disp-high are absent (unless, mod=00 and r/m=110), then EA=disp-high and disp-low).

If mod=01, then disp=disp-low, sign extended to 16 bits, disp-high is absent.

If mod=10, then disp=disp-high and disp-low.

If mod=11, then r/m is a reg field.

Register Field

Each of the general 8-bit, general 16-bit, base (BX and BP), and index (SI and DI) registers can be used in arithmetic and logical operations.

The 3-bit binary code assignment for the general, base and index registers and the 2-bit binary code assignment for the segment registers are:

8-Bit Registers	16-Bit Registers	Segment Registers
AL = 000	AX = 000	ES = 00
CL = 001	CX = 001	CS = 01
DL = 010	DX = 010	SS = 10
BL = 011	BX = 011	DS = 11
AH = 100	SP = 100	
CH = 101	BP = 101	
DH = 110	SI = 110	
BH = 111	DI = 111	

Flag Registers

X X X X OF DF IF TF SF ZF X AF X PF X CF
15 0

X = Reserved

CF = Carry Flag	TF = Trap Flag
PF = Parity Flag	IF = Interrupt Flag
AF = Auxiliary Carry Flag	DF = Direction Flag
ZF = Zero Flag	OF = Overflow Flag
SF = Sign Flag	

Register/Memory Field

If r/m = 000, then EA = [BX] + [SI] + disp
If r/m = 001, then EA = [BX] + [DI] + disp
If r/m = 010, then EA = [BP] + [SI] + disp
If r/m = 011, then EA = [BP] + [DI] + disp
If r/m = 100, then EA = [SI] + disp
If r/m = 101, then EA = [DI] + disp
If r/m = 110, then EA = [BP] + disp (unless mod=00, then EA = disp-high and disp-low)
If r/m = 111, then EA = [BX] + disp

APPENDIX B. INSTRUCTION SET SUMMARY

Flag registers

AF=Auxiliary carry
CF=Carry
DF=Direction
IF=Interrupt
OF=Overflow
PF=Parity
SF=Sign
TF=Trap
ZF=Zero

Legend for flag conditions

A=Altered to reflect results of operation
R=Replaced from storage
U=Undefined
0=Unconditionally cleared to 0
1=Unconditionally set to 1
None=No flags affected

AAA (no operands) – ASCII adjust for addition			
Operands	**Bytes**	**Example**	**Flags**
none	1	AAA	AF=A PF=U CF=A SF=U OF=U ZF=U

AAD (no operands) – ASCII adjust for division			
Operands	**Bytes**	**Example**	**Flags**
none	2	AAD	AF=U SF=A CF=U ZF=A OF=U PF=A

AAM (no operands) – ASCII adjust for multiply			
Operands	**Bytes**	**Example**	**Flags**
none	1	AAM	AF=U PF=A CF=U SF=A OF=U ZF=A

AAS (no operands) — ASCII adjust for subtraction

Operands	Bytes	Example	Flags
none	1	AAS	AF=A PF=U CF=A SF=U OF=U ZF=U

ADC destination, source — Add with carry

Operands	Bytes	Example	Flags
register, register	2	ADC AX, SI	AF=A PF=A
register, memory	2-4	ADC DX, BETA [SI]	CF=A SF=A
memory, register	2-4	ADC ALPHA [BX] [SI], DI	OF=A ZF=A
register, immediate	3-4	ADC BX, 256	
memory, immediate	3-6	ADC GAMMA, 30H	
accumulator, immediate	2-3	ADC AL, 5	

ADD destination, source — Addition

Operands	Bytes	Example	Flags
register, register	2	ADD CX, DX	AF=A PF=A
register, memory	2-4	ADD DI, [BX], ALPHA	CF=A SF=A
memory, register	2-4	ADD TEMP, CL	OF=A ZF=A
register, immediate	3-4	ADD CL, 2	
memory, immediate	3-6	ADD ALPHA, 2	
accumulator, immediate	2-3	ADD AX, 200	

AND destination source — Logical AND

Operands	Bytes	Example	Flags
register, register	2	AND AL,BL	AF=U PF=A
register, memory	2-4	AND CX,FLAG WORD	CF=0 SF=A
memory, register	2-4	AND ASCII [DI],AL	OF=0 ZF=A
register, immediate	3-4	AND CX0,F0H	
memory, immediate	3-6	AND BETA, 01H	
accumulator, immediate	2-3	AND AX, 01010000B	

CALL target — Call a procedure

Operands	Bytes	Example	Flags
near-proc	3	CALL NEAR_PROC	none
far-proc	5	CALL FAR_PROC	
memptr 16	2-4	CALL PROC_TABLE [SI]	
regptr 16	2	CALL AX	
memptr 32	2-4	CALL [BX].TASK [SI]	

CBW (no operands) — Convert byte to word

Operands	Bytes	Example	Flags
none	1	CBW	none

CLC (no operands) — Clear carry flag

Operands	Bytes	Example	Flags
none	1	CLC	CF=0

CLD (no operands) — Clear direction flag

Operands	Bytes	Example	Flags
none	1	CLD	DF=0

CLI (no operands) — Clear interrupt flag

Operands	Bytes	Example	Flags
none	1	CLI	DF=0

CMC (no operands) — Complement carry flag

Operands	Bytes	Example	Flags
none	1	CMC	CF=A

CMP destination, source — Compare destination to source

Operands	Bytes	Example	Flags
register, register	2	CMP BX, CX	AF=A PF=A
register, memory	2-4	CMP DH. ALPHA	CF=A SF=A
memory, register	2-4	CMP [BP + 2], SI	OF=A ZF=A
register, immediate	3-4	CMP BL, 02H	
memory, immediate	3-6	CMP [BX] RADAR [DI], 3420H	
accumulator, immediate	2-3	CMP AL, 00010000B	

CMPS dest-string, source-string — Compare string

Operands	Bytes	Example	Flags
dest-string, source-string	1	CMPS BUFF1, BUFF2	AF=A PF=A
(repeat) dest-string, source-string	1	REPE CMPS ID, KEY	CF=A SF=A OF=A ZF=A

CWD (no operand) — Convert word to doubleword

Operands	Bytes	Example	Flags
none	1	CWD	none

DAA (no operand) — Decimal adjust for addition

Operands	Bytes	Example	Flags
none	1	DAA	AF=A PF=A CF=A SF=A OF=A ZF=A

DAS (no operand) — Decimal adjust for subtraction

Operands	Bytes	Example	Flags
none	1	DAS	AF=A PF=A CF=A SF=A OF=A ZF=A

DEC destination — Decrement by 1

Operands	Bytes	Example	Flags
reg16	1	DEC AX	AF=A SF=A
reg8	2	DEC AL	OF=A ZF=A
memory	2-4	DEC ARRAY [SI]	PF=A

DIV source — Division, unsigned

Operands	Bytes	Example	Flags
reg8	2	DIV CL	AF=U PF=U
reg16	2	DIV BX	CF=U SF=U
mem8	2-4	DIV ALPHA	OF=U ZF=U
mem16	2-4	DIV TABLE [SI]	

ESC external-op code, source — Escape

Operands	Bytes	Example	Flags
immediate, memory	2-4	ESC 6,ARRAY [SI]	none
immediate, register	2	ESC 20,AL	

HLT (no operands) — Halt

Operands	Bytes	Example	Flags
none	1	HLT	none

IDIV source — Integer division

Operands	Bytes	Example	Flags
reg8	2	IDIV BL	AF=U PF=U
reg16	2	IDIV CX	CF=U SF=U
mem8	2-4	IDIV DIVISOR BYTE [SI]	OF=U ZF=U
mem16	2-4	IDIV [BX],DIVISOR_WORD	

IMUL source — Integer multiplication

Operands	Bytes	Example	Flags
reg8	2	IMUL CL	AF=U PF=U
reg16	2	IMUL BX	CF=A SF=U
mem8	2-4	IMUL RATE_BYTE	OF=A ZF=U
mem16	2-4	IMUL RATE_WORD[BP] [DI]	

IN accumulator, port — Input byte or word

Operands	Bytes	Example	Flags
accumulator, immed8	2	IN AL, 0FFEAH	none
accumulator, DX	1	IN AX, DX	

INC destination — Increment by 1

Operands	Bytes	Example	Flags
reg16	1	INC CX	AF=A SF=A
reg8	2	INC BL	OF=A ZF=A
memory	2-4	INC ALPHA [DI] [BX]	PF=A

INT interrupt-type — Interrupt

Operands	Bytes	Example	Flags
immed8(type = 3)	1	INT 3	IF=0
immed8(type ≠ 3)	2	INT 67	TF=0

INTO (no operands) — Interrupt if overflow

Operands	Bytes	Example	Flags
none	1	INTO	IF=0 TF=0

IRET (no operands) — Interrupt return

Operands	Bytes	Example	Flags
none	1	IRET	AF=R PF=R CF=R SF=R DF=R TF=R IF=R ZF=R

JA/JNBE short-label — Jump if above/Jump if not below or equal

Operands	Bytes	Example	Flags
short-label	2	JA ABOVE	none

JAE/JNB short-label – Jump if above or equal/Jump if not below			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JAE ABOVE_EQUAL	none

JB/JNAE short-label – Jump if below/Jump if not above nor equal			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JB BELOW	none

JBE/JNA short-label – Jump if below or equal/Jump if not above			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNA NOT ABOVE	none

JC short-label – Jump if carry			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JC CARRY SET	none

JCXZ short-label – Jump if CX is zero			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JCXZ COUNT DONE	none

JE/JZ short-label – Jump if equal/Jump if zero			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JZ ZERO	none

JG/JNLE short-label – Jump if greater/Jump if not less nor equal			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JG GREATER	none

JGE/JNL short-label – Jump if greater or equal/Jump if not less			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JGE GREATER EQUAL	none

JL/JNGE short-label – Jump if less/Jump if not greater nor equal			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JL LESS	none

JLE/JNG short-label – Jump if less or equal/Jump if not greater			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNG NOT GREATER	none

JMP target – Jump			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JMP SHORT	none
near-label	3	JMP WITHIN SEGMENT	
far-label	5	JMP FAR LABEL	
memptr16	2-4	JMP [BX], TARGET	
regptr16	2	JMP CX	
memptr32	2-4	JMP OTHER SEG [SI]	

JNC short-label – Jump if not carry			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNC NOT CARRY	none

JNE/JNZ short-label – Jump if not equal/Jump if not zero			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNE NOT EQUAL	none

JNO short-label – Jump if not overflow			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNO NO OVERFLOW	none

JNP/JPO short-label – Jump if not parity/jump if parity ODD			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JPO ODD PARITY	none

JNS short-label – Jump if not sign			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JNS POSITIVE	none

JO short-label – Jump if overflow			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JO SIGNED_OVRFLW	none

JP/JPE short-label — Jump if parity/Jump if parity even			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JPE EVEN_PARITY	none

JS short-label — Jump if sign			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	JS NEGATIVE	none

LAHF (no operands) — Load AH from flags			
Operands	**Bytes**	**Example**	**Flags**
none	1	LAHF	none

LDS destination, source — Load pointer using DS			
Operands	**Bytes**	**Example**	**Flags**
reg16, mem32	2-4	LDS SI, DATA. SEG[DI]	none

LOCK (no operands) — Lock bus			
Operands	**Bytes**	**Example**	**Flags**
none	1	LOCK XCHG FLAG.AL	none

LODS source-string — Load string			
Operands	**Bytes**	**Example**	**Flags**
source-string (repeat) source-string	1 1	LODS CUSTOMER NAME REP LODS NAME	none

LOOP short-label — Loop			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	LOOP AGAIN	none

LOOPE/LOOPZ short-label — Loop if equal/Loop if zero			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	LOOPE AGAIN	none

LOOPNE/LOOPNZ short-label — Loop if not equal/Loop if not zero			
Operands	**Bytes**	**Example**	**Flags**
short-label	2	LOOPNE AGAIN	none

LEA destination, source — Load effective address			
Operands	**Bytes**	**Example**	**Flags**
reg16, mem16	2-4	LEA BX,[BP] [DI]	none

LES destination, source — Load pointer using ES			
Operands	**Bytes**	**Example**	**Flags**
reg16, mem32	2-4	LES DI,[BX].TEXT_BUFF	none

MOV destination, source — Move			
Operands	**Bytes**	**Example**	**Flags**
memory, accumulator	3	MOV ARRAY [SI], AL	none
accumulator, memory	3	MOV AX, TEMP_RESULT	
register, register	2	MOV AX, CX	
register, memory	2-4	MOV BP, STACK_TOP	
memory, register	2-4	MOV COUNT [DI], CX	
register, immediate	2-3	MOV CL, 2	
memory, immediate	3-6	MOV MASK [BX] [SI], 2CH	
seg-reg, reg16	2	MOV ES, CX	
seg-reg, mem16	2-4	MOV DS. SEGMENT_BASE	
reg16, seg-reg	2	MOV BP, SS	
memory, seg-reg	2-4	MOV [BX].SEG_SAVE, CS	

MOVS dest-string, source-string — MOVE string			
Operands	**Bytes**	**Example**	**Flags**
dest-string. source-string repeat) dest-string. source-string	1 1	MOVS LINE EDIT_DATA REP MOVS SCREEN. BUFFER	none

MOVSB/MOVSW no operands — Move string (byte/word)			
Operands	**Bytes**	**Example**	**Flags**
none (repeat) none	1 1	MOVSB REP MOVSW	none

MUL source — Multiplication, unsigned

Operands	Bytes	Example	Flags
reg8	2	MUL BL	AF=U PF=U
reg16	2	MUL CX	CF=A SF=U
mem8	2-4	MUL MONTH [SI]	OF=A ZF=U
mem16	2-4	MUL BAUD–RATE	

NEG destination — Negate

Operands	Bytes	Example	Flags
register	2	NEG AL	AF=A PF=A
memory	2-4	NEG MULTIPLIER	CF=1* SF=A
			OF=A ZF=A

*0 if destination = 0

NOP no operands — No operation

Operands	Bytes	Example	Flags
none	1	NOP	none

NOT destination — Logical not

Operands	Bytes	Example	Flags
register	2	NOT AX	none
memory	2-4	NOT CHARACTER	

OR destination, source — Logical inclusive OR

Operands	Bytes	Example	Flags
register, register	2	OR AL, BL	AF=U PF=A
register, memory	2-4	OR DX, PORT ID [DI]	CF=0 SF=A
memory, register	2-4	OR FLAG BYTE. CL	OF=0 ZF=A
accumulator, immediate	2-3	OR AL, 0110110B	
register, immediate	3-4	OR CX, 01FH	
memory, immediate	3-6	OR[BX] CMD WORD, 0CFH	

OUT port, accumulator — Output byte or word

Operands	Bytes	Example	Flags
immed8. accumulator	2	OUT 44. AX	none
DX. accumulator	1	OUT DX. AL	

POP destination — Pop word off stack

Operands	Bytes	Example	Flags
register	1	POP DX	none
seg-reg (CS illegal)	1	POP DS	
memory	2-4	POP PARAMETER	

POPF no operands — Pop flags off stack

Operands	Bytes	Example	Flags
none	1	POPF	AF=R PF=R
			CF=R SF=R
			DF=R TF=R
			IF=R ZF=R

PUSH source — Push word onto stack

Operands	Bytes	Example	Flags
register	1	PUSH SI	none
seg-reg (CS legal)	1	PUSH ES	
memory	2-4	PUSH RETURN CODE [SI]	

PUSHF (no operands) – Push flags onto stack

Operands	Bytes	Example	Flags
none	1	PUSHF	none

RCL destination, count — Rotate left through carry

Operands	Bytes	Example	Flags
register, 1	2	RCL CX, 1	CF=A
register, CL	2	RCL AL, CL	OF=A
memory, 1	2-4	RCL ALPHA, 1	
memory, CL	2-4	RCL [BP].PARM, CL	

RCR destination, count — Rotate right through carry

Operands	Bytes	Example	Flags
register, 1	2	RCR BX, 1	CF=A
register, CL	2	RCR BL, CL	OF=A
memory, 1	2-4	RCR [BX], STATUS, 1	
memory, CL	2-4	RCR ARRAY [DI], CL	

REP (no operands) — Repeat string operation

Operands	Bytes	Example	Flags
none	1	REP MOVS DEST.SRCE	none

REPE/REPZ (no operands) — Repeat string operation while equal/while zero

Operands	Bytes	Example	Flags
none	1	REPE CMPS DATA.KEY	none

REPNE/REPNZ (no operands) — Repeat string operation while not equal/not zero

Operands	Bytes	Example	Flags
none	1	REPNE SCAS INPUT LINE	none

RET optional-pop-value — Return from procedure

Operands	Bytes	Example	Flags
(intra-segment, no pop)	1	RET	none
(intra-segment, pop)	3	RET 4	
(inter-segment, no pop)	1	RET	
(inter-segment, pop)	3	RET 2	

ROL destination, count — Rotate left

Operands	Bytes	Example	Flags
register, 1	2	ROL BX, 1	CF=A
register, CL	2	ROL DI, CL	OF=A
memory, 1	2-4	ROL FLAG BYTE[DI], 1	
memory, CL	2-4	ROL ALPHA, CL	

ROR destination, count — Rotate right

Operands	Bytes	Example	Flags
register, 1	2	ROR AL, 1	CF=A
register, CL	2	ROR BX, CL	OF=A
memory, 1	2-4	ROR PORT STATUS, 1	
memory, CL	2-4	ROR CMD WORD, CL	

SAHF (no operands) — Store AH into flags

Operands	Bytes	Example	Flags
none	1	SAHF	AF=R SF=R CF=R ZF=R PF=R

SAL/SHL destination, count — Shift arithmetic left/Shift logical left

Operands	Bytes	Example	Flags
register, 1	2	SAL AL, 1	CF=A
register, CL	2	SHL DI, CL	OF=A
memory, 1	2-4	SHL [BX].OVERDRAW, 1	
memory, CL	2-4	SAL STORE_COUNT, CL	

SAR destination, source — Shift arithmetic right

Operands	Bytes	Example	Flags
register, 1	2	SAR DX, 1	AF=U PF=A
register, CL	2	SAR DI, CL	CF=A SF=A
memory, 1	2-4	SAR N BLOCKS, 1	OF=A ZF=A
memory, CL	2-4	SAR N BLOCKS, CL	

SBB destination, source — Subtract with borrow

Operands	Bytes	Example	Flags
register, register	2	SBB BX, CX	AF=A PF=A
register, memory	2-4	SBB DI, [BX].PAYMENT	CF=A SF=A
memory, register	2-4	SSB BALANCE, AX	OF=A ZF=A
accumulator, immediate	2-3	SBB AX, 2	
register, immediate	3-4	SBB CL, 1	
memory, immediate	3-6	SBB COUNT[SI], 10	

SCAS dest-string — Scan string

Operands	Bytes	Example	Flags
dest-string	1	SCAS INPUT_LINE	AF=A PF=A
(repeat) dest-string	1	REPNE SCAS BUFFER	CF=A SF=A OF=A ZF=A

SHR destination, count — Shift logical right

Operands	Bytes	Example	Flags
register, 1	2	SHR SI, 1	CF=A
register, CL	2	SHR SI, CL	OF=A
memory, 1	2-4	SHR ID BYTE [SI] [BX], 1	
memory, CL	2-4	SHR INPUT WORD, CL	

STC no operands — Set carry flag

Operands	Bytes	Example	Flags
none	1	STC	CF=1

STD no operand — Set direction flag

Operands	Bytes	Example	Flags
none	1	STD	DF=1

STI (no operands) — Set interrupt enable flag

Operands	Bytes	Example	Flags
none	1	STI	IF=1

STOS dest-string — Store byte or word string			
Operands	**Bytes**	**Example**	**Flags**
dest-string	1	STOS PRINT_LINE	none
(repeat) dest-string	1	REP STOS DISPLAY	

SUB destination, source — Subtraction			
Operands	**Bytes**	**Example**	**Flags**
register, register	2	SUB CX,*BX	AF=A PF=A
register, memory	2-4	SUB DX, MATH_TOTAL[SI]	CF=A SF=A
memory, register	2-4	SUB [BP +2], CL	OF=A ZF=A
accumulator, immediate	2-3	SUB AL, 10	
register, immediate	3-4	SUB SI, 5280	
memory, immediate	3-6	SUB [BP].BALANCE, 1000	

TEST destination, source — Test or non-destructive logical AND			
Operands	**Bytes**	**Example**	**Flags**
register, register	2	TEST SI, DI	AF=U PF=A
register, memory	2-4	TEST SI, END_COUNT	CF=0 SF=A
accumulator, immediate	2-3	TEST AL, 00100000B	OF=0 ZF=A
register, immediate	3-4	TEST BX, 0CC4H	
memory, immediate	3-6	TEST RETURN CODE, 01H	

WAIT (no operands) — Wait while TEST pin not asserted			
Operands	**Bytes**	**Example**	**Flags**
none	1	WAIT	none

XCHG destination, source — Exchange			
Operands	**Bytes**	**Example**	**Flags**
accumulator, reg16	1	XCHG AX, BX	none
memory, register	2-4	XCHG SEMAPHORE, AX	
register, register	2	XCHG AL, BL	

XLAT source-table — Translate			
Operands	**Bytes**	**Example**	**Flags**
source-table	1	XLAT ASCII_TAB	none

XOR destination, source — Logical exclusive OR			
Operands	**Bytes**	**Example**	**Flags**
register, register	2	XOR CX, BX	AF=U PF=A
register, memory	2-4	XOR CL, MASK BYTE	CF=0 SF=A
memory, register	2-4	XOR ALPHA [SI], DX	OF=0 ZF=A
accumulator, immediate	2-3	XOR AL, 01000010B	
register, immediate	3-4	XOR SI, 00C2H	
memory, immediate	3-6	XOR RETURN CODE.0D2H	

Data Pseudo Operations

ASSUME *seg-reg*:*seg-name* [,...]
or
ASSUME NOTHING

COMMENT *delimiter text delimiter*

variable-name DB *expression*

variable-name DW *expression*

variable-name DD *expression*

variable-name DQ *expression*

variable-name DT *expression*

END *expression*

name EQU *expression*

label = *expression*

EVEN (no argument)

EXTERN *name*:*type* [,...]

name GROUP *seg-name* [,...]

name LABEL *type*

NAME *module-name*

ORG *expression*

procedure-name PROC [NEAR]
or
procedure-name PROC [FAR]

PUBLIC *symbol* [,...]

PURGE *symbol*

.RADIX *expression*

recordname RECORD *fieldname*:*width* [=*exp*] [,...]

segname SEGMENT [*align*] [*combine*]
[*class*]

•
•
•

segname ENDS

structurename STRUC

•
•
•

structurename ENDS

Character Set (00-7F) Quick Reference

DECIMAL VALUE		0	16	32	48	64	80	96	112
	HEXA DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	!!	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

Character Set (80-FF) Quick Reference

DECIMAL VALUE		128	144	160	176	192	208	224	240
	HEXA DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	▒	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▒	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	a	╢	╞	╓	μ	÷
7	7	ç	ù	o	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	•
11	B	ï	¢	½	╗	╦	■	δ	√
12	C	î	£	¼	╝	╠	▄	∞	n
13	D	ì	¥	¡	╜	═	▌	φ	2
14	E	Ä	₧	«	╛	╬	▐	∈	■
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK FF