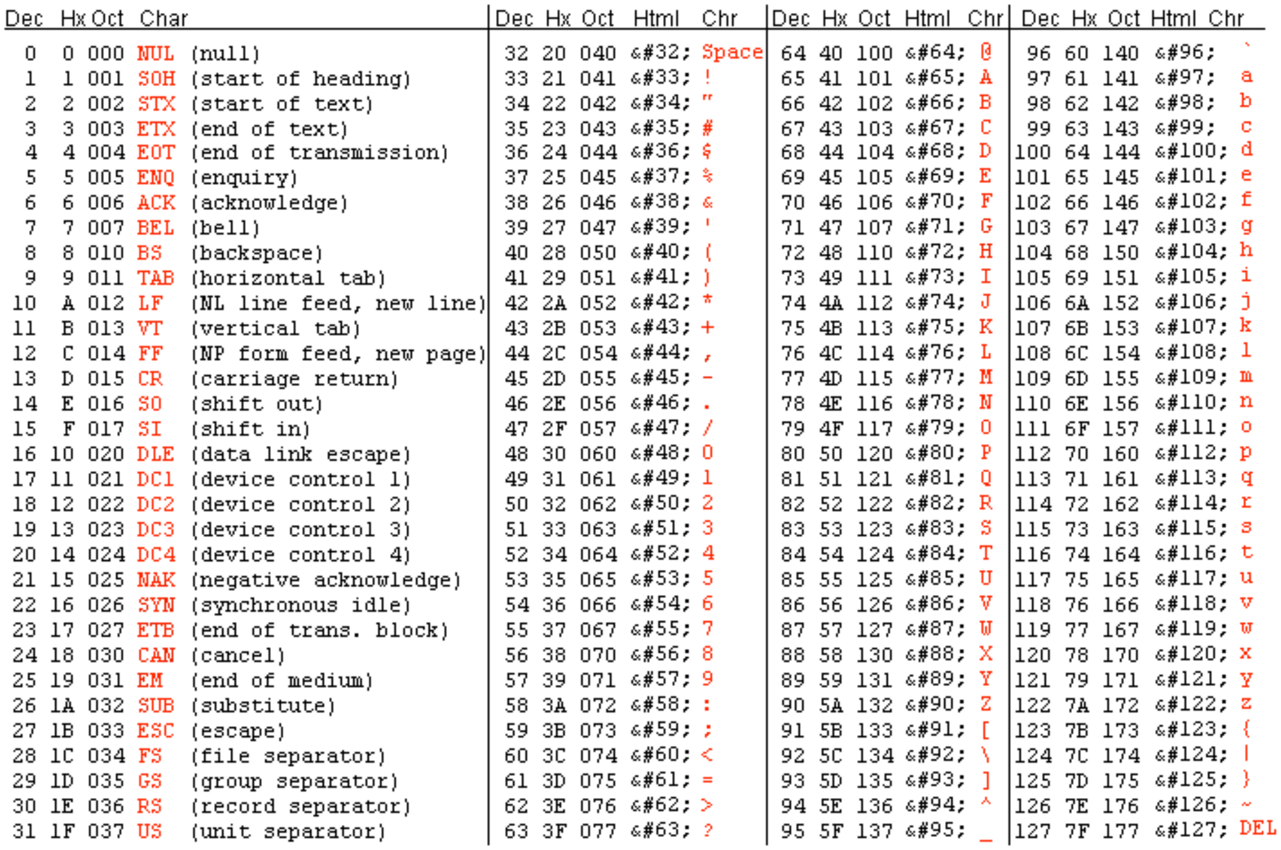
FOCAL POINT ACADEMY

Assembly for Reverse Engineers
ASCII Reference Guide



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Datarescue
Interactive Disassembler (IDA) Pro
Quick Reference Sheet
(http://www.datarescue.com)

Navigation

Jump to operand	Enter
Jump in new window	Alt+Enter
Jump to previous position	Esc
Jump to next position	Ctrl+Enter
Jump to address	G
Jump by name	Ctrl+L
Jump to function	Ctrl+P
Jump to segment	Ctrl+S
Jump to segment register	Ctrl+G
Jump to problem	Ctrl+Q
Jump to cross reference	Ctrl+X
Jump to xref to operand	X
Jump to entry point	Ctrl+E
Mark Position	Alt+M
Jump to marked position	Ctrl+M

Search

Next code	Alt+C
Next data	Ctrl+D
Next explored	Ctrl+A
Next unexplored	Ctrl+U
Immediate value	Alt+I
Next immediate value	Ctrl+I
Text	Alt+T
Next text	Ctrl+T
Sequence of bytes	Alt+B
Next sequence of bytes	Ctrl+B
Not function	Alt+U
Next void	Ctrl+V
Error operand	Ctrl+F

Graphing

Flow chart	F12
Function calls	Ctrl+F12

Open Subviews

Names	Shift+F4
Functions	Shift+F3
Strings	Shift+F12
Segments	Shift+F7
Segment registers	Shift+F8
Signatures	Shift+F5
Type libraries	Shift+F11
Structures	Shift+F9
Enumerations	Shift+F10

Data Format Options

ASCII strings style	Alt+A
Setup data types	Alt+D

File Operations

Parse C header file	Ctrl+F9
Create ASM file	Alt+F10
Save database	Ctrl+W

Debugger

Star process	F9
Terminate process	Ctrl+F2
Step into	F7
Step over	F8
Run until return	Ctrl+F7
Run to cursor	F4

Breakpoints

Breakpoint list	Ctrl+Alt+B

Watches

Delete watch	Del

Tracing

Stack trace	Ctrl+Alt+S

Miscellaneous

Calculator	Shift+/
Cycle through open views	Ctrl+Tab
Select tab	Alt + [1…N]
Close current view	Ctrl+F4
Exit	Alt+X
IDC Command	Shift+F2

Edit (Data Types – etc)

Copy	Ctrl+Ins
Begin selection	Alt+L
Manual instruction	Alt+F2
Code	C
Data	D
Struct variable	Alt+Q
ASCII string	A
Array	Num *
Undefine	U
Rename	N

Operand Type

Offset (data segment)	O
Offset (current segment)	Ctrl+O
Offset by (any segment)	Alt+R
Offset (user-defined)	Ctrl+R
Offset (struct)	T
Number (default)	Shift+3
Hexadecimal	Q
Decimal	H
Binary	B
Character	R
Segment	S
Enum member	M
Stack variable	K
Change sign	Shift+-
Bitwise negate	Shift+`
Manual	Alt+F1

Comments

Enter comment	Shift+;
Enter repeatable comment	;
Enter anterior lines	Ins
Enter posterior lines	Shift+Ins
Insert predefined comment	Shift+F1

Segments

Edit segment	Alt+S
Change segment register value	Alt+G

Structs

Struct var	Alt+Q
Force zero offset field	Ctrl+Z
Select union member	Alt+Y

Functions

Create function	P
Edit function	Alt+P
Set function end	E
Stack variables	Ctrl+K
Change stack pointer	Alt+K
Rename register	V
Set function type	Y

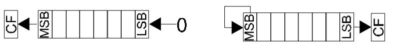
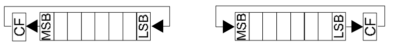
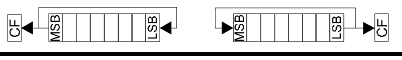
Intel Assembler 80186 and higher

CodeTable 1/2

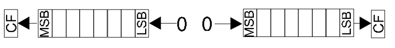
V 2.3 English. Also available in Spanish

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)	±				±	±	±	±	±
CWDE	Conv word extended double	CWDE 386	EAX:=AX (signed)									
IN *i*	Input	IN Dest, Port	AL/AX/EAX := byte/word/double of specified port									
OUT *i*	Output	OUT Port, Source	Byte/word/double of specified port := AL/AX/EAX									

i for more information 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*	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*				±	±	?	±	±
SAR	Shift arithmetic right	SAR Op,Quantity		*i*				±	±	?	±	±
RCL	Rotate left through Carry	RCL Op,Quantity		*i*								±
RCR	Rotate right through Carry	RCR Op,Quantity		*i*								±
ROL	Rotate left	ROL Op,Quantity		*i*								±
ROR	Rotate right	ROR Op,Quantity		*i*								±

i for more information 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				±	±	?	±	0
OR	Logical or	OR Dest,Source	Dest:=Dest∨Source	0				±	±	?	±	0
XOR	Logical exclusive or	XOR Dest,Source	Dest:=Dest (exor) Source	0				±	±	?	±	0
SHL	Shift logical left (≡ SAL)	SHL Op,Quantity		*i*				±	±	?	±	±
SHR	Shift logical right	SHR Op,Quantity		*i*				±	±	?	±	±

Intel Assembler 80186 and higher

CodeTable 2/2

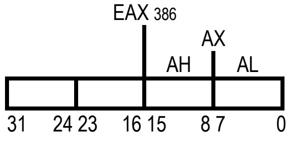
V 2.3 English. Also available in Spanish

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

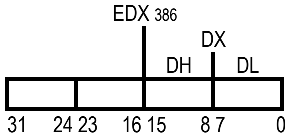
JUMPS (flags remain unchanged)							
Name	**Comment**	**Code**	**Operation**	**Name**	**Comment**	**Code**	**Operation**
CALL	Call subroutine	CALL Proc		RET	Return from subroutine	RET	
JMP	Jump	JMP Dest					
JE	Jump if Equal	JE Dest	(≡ JZ)	JNE	Jump if not Equal	JNE Dest	(≡ JNZ)
JZ	Jump if Zero	JZ Dest	(≡ 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)

JUMPS Unsigned (Cardinal)				**JUMPS 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	

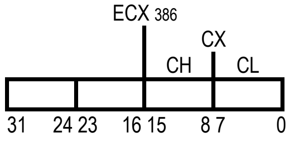
General Registers:



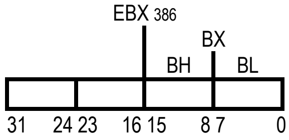
Accumulator



Data mul, div, IO



Count loop, shift



BaseX data ptr

Example:

```
        .DOSSEG             ; Demo program
        .MODEL SMALL
        .STACK 1024
Two     EQU 2               ; Const
        .DATA
VarB    DB ?                ; define Byte, any value
VarW    DW 1010b            ; define Word, binary
VarW2   DW 257              ; define Word, decimal
VarD    DD 0AFFFFh          ; define Doubleword, hex
S       DB "Hello !",0      ; define String
        .CODE
main:   MOV AX,DGROUP       ; resolved by linker
        MOV DS,AX           ; init datasegment reg
        MOV [VarB],42       ; init VarB
        MOV [VarD],-7       ; set VarD
        MOV BX,Offset[S]    ; addr of "H" of "Hello !"
        MOV AX,[VarW]       ; get value into accumulator
        ADD AX,[VarW2]      ; add VarW2 to AX
        MOV [VarW2],AX      ; store AX in VarW2
        MOV AX,4C00h        ; back to system
        INT 21h
        END main
```

KEEP IT SIMPLE, STUPID (AND SHORT) - KISS

Flags: | - | - | - | - | O | D | I | T | S | Z | - | A | - | P | - | C |

Control Flags (how instructions are carried out):

D: Direction	1 = string op's process down from high to low address
I: Interrupt	whether interrupts can occur. 1= enabled
T: Trap	single step for debugging

Status Flags (result of operations):

C: Carry	result of unsigned op. is too large or below zero. 1 = carry/borrow
O: Overflow	result of signed op. is too large or small. 1 = overflow/underflow
S: Sign	sign of result. Reasonable for Integer only. 1 = neg. / 0 = pos.
Z: Zero	result of operation is zero. 1 = zero
A: Aux. carry	similar to Carry but restricted to the low nibble only
P: Parity	1 = result has even number of set bits