CHAPTER 3

Symbols and Operators

Topical Cross-reference for Symbols

Date and Time Information

@Date
@Time

Environment Information

@Cpu
@Environ
@Interface
@Version

File Information

@FileCur
@FileName
@Line

Macro Functions

@CatStr
@InStr
@SizeStr
@SubStr

Miscellaneous

$	?	@@:
@B	@F	

Segment Information

@code	@CodeSize	@CurSeg
@data	@DataSize	@fardata
@fardata?	@Model	@stack
@WordSize		

Topical Cross-reference for Operators

Arithmetic

*	+	-
.	/	[]
MOD		

Control Flow

!	!=	&
&&	<	< =
= =	>	> =
‖		

Logical and Shift

AND	NOT	OR
SHL	SHR	XOR

Macro

!	%	&
;;	<>	

Miscellaneous

’’	“ ”	:
::	;	CARRY?
DUP	OVERFLOW?	PARITY?
SIGN?	ZERO?	

Record

MASK

WIDTH

Relational

EQ	GE	GT
LE	LT	NE

Segment

:
LROFFSET
OFFSET
SEG

Type

HIGH	HIGHWORD	LENGTH
LENGTHOF	LOW	LOWWORD
OPATTR	PTR	SHORT
SIZE	SIZEOF	THIS
TYPE		

Predefined Symbols

$
The current value of the location counter.

?
In data declarations, a value that the assembler allocates but does not initialize.

@@:
Defines a code label recognizable only between *label1* and *label2*, where *label1* is either start of code or the previous **@@:** label, and *label2* is either end of code or the next **@@:** label. See **@B** and **@F**.

@B
The location of the previous **@@:** label.

@CatStr(*string1* [[**,** *string2*...]] **)**
Macro function that concatenates one or more strings. Returns a string.

@code
The name of the code segment (text macro).

@CodeSize
0 for **TINY**, **SMALL**, **COMPACT**, and **FLAT** models, and 1 for **MEDIUM**, **LARGE**, and **HUGE** models (numeric equate).

@Cpu
A bit mask specifying the processor mode (numeric equate).

@CurSeg
The name of the current segment (text macro).

@data
The name of the default data group. Evaluates to DGROUP for all models except **FLAT**. Evaluates to **FLAT** under the **FLAT** memory model (text macro).

@DataSize
0 for **TINY**, **SMALL**, **MEDIUM**, and **FLAT** models, 1 for **COMPACT** and **LARGE** models, and 2 for **HUGE** model (numeric equate).

@Date
The system date in the format mm/dd/yy (text macro).

@Environ(*envvar* **)**
Value of environment variable *envvar* (macro function).

@F
The location of the next **@@:** label.

@fardata
The name of the segment defined by the **.FARDATA** directive (text macro).

@fardata?
The name of the segment defined by the **.FARDATA?** directive (text macro).

@FileCur
The name of the current file (text macro).

@FileName
The base name of the main file being assembled (text macro).

@InStr([[*position*]]**,** *string1***,** *string2* **)**
Macro function that finds the first occurrence of *string2* in *string1*, beginning at *position* within *string1*. If *position* does not appear, search begins at start of *string1*. Returns a position integer or 0 if *string2* is not found.

@Interface
Information about the language parameters (numeric equate).

@Line
The source line number in the current file (numeric equate).

@Model
1 for **TINY** model, 2 for **SMALL** model, 3 for **COMPACT** model, 4 for **MEDIUM** model, 5 for **LARGE** model, 6 for **HUGE** model, and 7 for **FLAT** model (numeric equate).

@SizeStr(*string* **)**
Macro function that returns the length of the given string. Returns an integer.

@SubStr(*string***,** *position* [[**,** *length*]] **)**
Macro function that returns a substring starting at *position*.

@stack
DGROUP for near stacks or STACK for far stacks (text macro).

@Time
The system time in 24-hour hh:mm:ss format (text macro).

@Version
610 in MASM 6.1 (text macro).

@WordSize
Two for a 16-bit segment or 4 for a 32-bit segment (numeric equate).

Operators

expression1 **+** *expression2*
Returns *expression1* plus *expression2*.

expression1 **–** *expression2*
Returns *expression1* minus *expression2*.

expression1 ***** *expression2*
Returns *expression1* times *expression2*.

expression1 **/** *expression2*
Returns *expression1* divided by *expression2*.

–*expression*
Reverses the sign of *expression*.

expression1 **[***expression2***]**
Returns *expression1* plus **[***expression2***]**.

*segment***:** *expression*
Overrides the default segment of *expression* with *segment*. The *segment* can be a segment register, group name, segment name, or segment expression. The *expression* must be a constant.

*expression***.** *field* **[[.** *field***]]**...
Returns *expression* plus the offset of *field* within its structure or union.

[*register***].** *field* **[[.** *field***]]**...
Returns value at the location pointed to by *register* plus the offset of *field* within its structure or union.

<*text***>**
Treats *text* as a single literal element.

"*text*"
Treats "*text*" as a string.

'*text*'
Treats '*text*' as a string.

!*character*
Treats *character* as a literal character rather than as an operator or symbol.

;*text*
Treats *text* as a comment.

;;*text*
Treats *text* as a comment in a macro that appears only in the macro definition. The listing does not show *text* where the macro is expanded.

%*expression*
Treats the value of *expression* in a macro argument as text.

&parameter**&**
Replaces *parameter* with its corresponding argument value.

ABS
See the **EXTERNDEF** directive.

ADDR
See the **INVOKE** directive.

expression1 **AND** *expression2*
Returns the result of a bitwise AND operation for *expression1* and *expression2*.

count **DUP** (*initialvalue* [[**,** *initialvalue*]]...)
Specifies *count* number of declarations of *initialvalue*.

expression1 **EQ** *expression2*
Returns true (–1) if *expression1* equals *expression2*, or returns false (0) if it does not.

expression1 **GE** *expression2*
Returns true (–1) if *expression1* is greater-than-or-equal-to *expression2*, or returns false (0) if it is not.

expression1 **GT** *expression2*
Returns true (–1) if *expression1* is greater than *expression2*, or returns false (0) if it is not.

HIGH *expression*
Returns the high byte of *expression*.

HIGHWORD *expression*
Returns the high word of *expression*.

expression1 **LE** *expression2*
Returns true (–1) if *expression1* is less than or equal to *expression2*, or returns false (0) if it is not.

LENGTH *variable*
Returns the number of data items in *variable* created by the first initializer.

LENGTHOF *variable*
Returns the number of data objects in *variable*.

LOW *expression*
Returns the low byte of *expression*.

LOWWORD *expression*
Returns the low word of *expression*.

LROFFSET *expression*
Returns the offset of *expression*. Same as **OFFSET**, but it generates a loader resolved offset, which allows Windows to relocate code segments.

expression1 **LT** *expression2*
Returns true (–1) if *expression1* is less than *expression2*, or returns false (0) if it is not.

MASK {*recordfieldname* | *record*}
Returns a bit mask in which the bits in *recordfieldname* or *record* are set and all other bits are cleared.

expression1 **MOD** *expression2*
Returns the integer value of the remainder (modulo) when dividing *expression1* by *expression2*.

expression1 **NE** *expression2*
Returns true (–1) if *expression1* does not equal *expression2*, or returns false (0) if it does.

NOT *expression*
Returns *expression* with all bits reversed.

OFFSET *expression*
Returns the offset of *expression*.

OPATTR *expression*
Returns a word defining the mode and scope of *expression*. The low byte is identical to the byte returned by **.TYPE**. The high byte contains additional information.

expression1 **OR** *expression2*
Returns the result of a bitwise OR operation for *expression1* and *expression2*.

type **PTR** *expression*
Forces the *expression* to be treated as having the specified *type*.

[[*distance*]] **PTR** *type*
Specifies a pointer to *type*.

SEG *expression*
Returns the segment of *expression*.

expression **SHL** *count*
Returns the result of shifting the bits of *expression* left *count* number of bits.

SHORT *label*
Sets the type of *label* to short. All jumps to *label* must be short (within the range –128 to +127 bytes from the jump instruction to *label*).

expression **SHR** *count*
Returns the result of shifting the bits of *expression* right *count* number of bits.

SIZE *variable*
Returns the number of bytes in *variable* allocated by the first initializer.

SIZEOF {*variable* | *type*}
Returns the number of bytes in *variable* or *type*.

THIS *type*
Returns an operand of specified *type* whose offset and segment values are equal to the current location-counter value.

.TYPE *expression*
See **OPATTR**.

TYPE *expression*
Returns the type of *expression*.

WIDTH {*recordfieldname* | *record*}
Returns the width in bits of the current *recordfieldname* or *record*.

expression1 **XOR** *expression2*
Returns the result of a bitwise XOR operation for *expression1* and *expression2*.

Run-Time Operators

The following operators are used only within **.IF**, **.WHILE**, or **.REPEAT** blocks and are evaluated at run time, not at assembly time:

expression1 **==** *expression2*
Is equal to.

expression1 **!=** *expression2*
Is not equal to.

expression1 **>** *expression2*
Is greater than.

expression1 **>=** *expression2*
Is greater than or equal to.

expression1 **<** *expression2*
Is less than.

expression1 **<=** *expression2*
Is less than or equal to.

expression1 **||** *expression2*
Logical OR.

expression1 **&&** *expression2*
Logical AND.

expression1 **&** *expression2*
Bitwise AND.

!*expression*
Logical negation.

CARRY?
Status of carry flag.

OVERFLOW?
Status of overflow flag.

PARITY?
Status of parity flag.

SIGN?
Status of sign flag.

ZERO?
Status of zero flag.