## What's Next

- Boolean and Comparison Instructions
- Conditional Jumps
- Conditional Loop Instructions
- Conditional Structures

**Conditional Control Flow Directives**

## Creating IF Statements

- Runtime Expressions
- Relational and Logical Operators
- MASM-Generated Code
- .REPEAT Directive
- .WHILE Directive

# Runtime Expressions

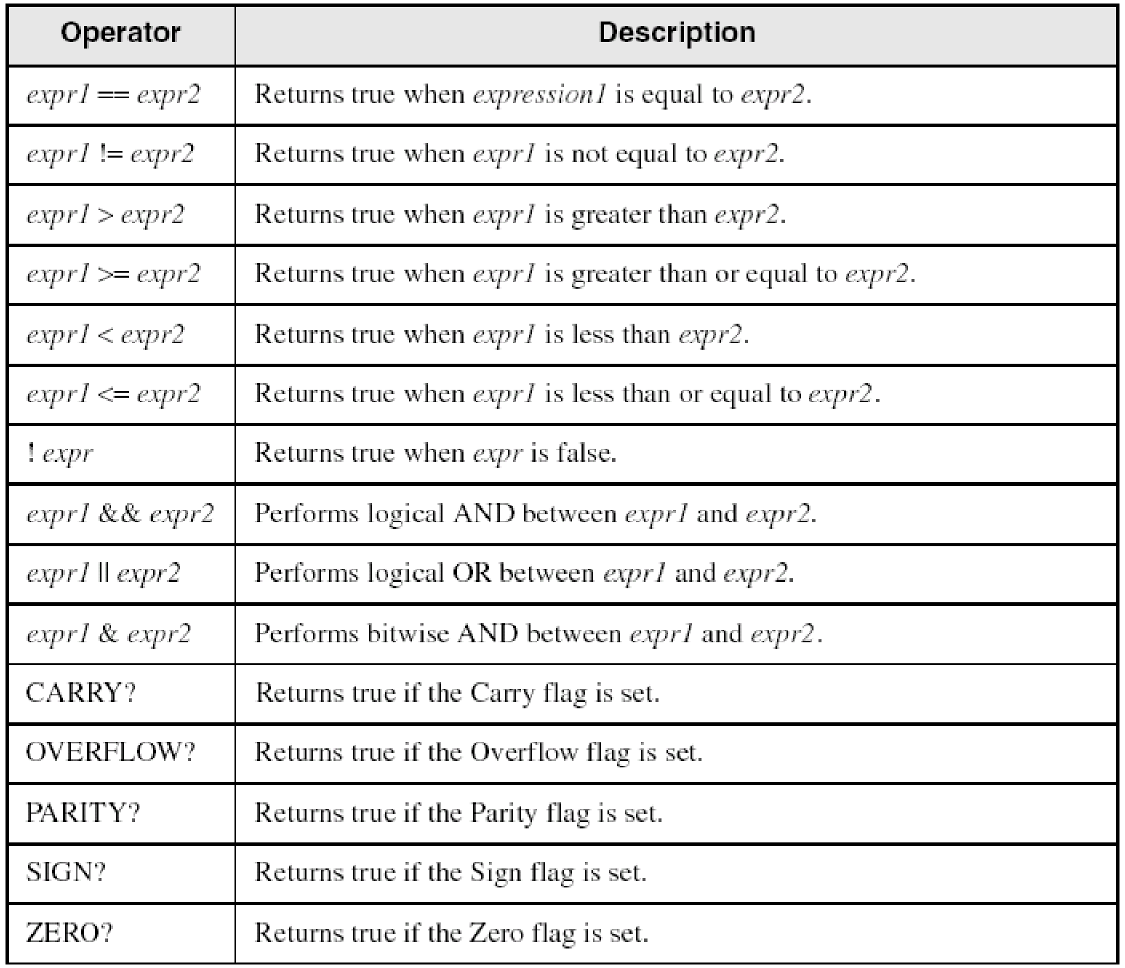
- .IF, .ELSE, .ELSEIF, and .ENDIF can be used to evaluate runtime expressions and create block-structured IF statements.
- Examples:

```
.IF eax > ebx
   mov edx,1
.ELSE
   mov edx,2
.ENDIF
```

```
.IF eax > ebx && eax > ecx
   mov edx,1
.ELSE
   mov edx,2
.ENDIF
```

- MASM generates "hidden" code for you, consisting of code labels, CMP and conditional jump instructions.

# Relational and Logical Operators



| Operator | Description |
|---|---|
| *expr1* == *expr2* | Returns true when *expression1* is equal to *expr2*. |
| *expr1* != *expr2* | Returns true when *expr1* is not equal to *expr2*. |
| *expr1* > *expr2* | Returns true when *expr1* is greater than *expr2*. |
| *expr1* >= *expr2* | Returns true when *expr1* is greater than or equal to *expr2*. |
| *expr1* < *expr2* | Returns true when *expr1* is less than *expr2*. |
| *expr1* <= *expr2* | Returns true when *expr1* is less than or equal to *expr2*. |
| ! *expr* | Returns true when *expr* is false. |
| *expr1* && *expr2* | Performs logical AND between *expr1* and *expr2*. |
| *expr1* \|\| *expr2* | Performs logical OR between *expr1* and *expr2*. |
| *expr1* & *expr2* | Performs bitwise AND between *expr1* and *expr2*. |
| CARRY? | Returns true if the Carry flag is set. |
| OVERFLOW? | Returns true if the Overflow flag is set. |
| PARITY? | Returns true if the Parity flag is set. |
| SIGN? | Returns true if the Sign flag is set. |
| ZERO? | Returns true if the Zero flag is set. |

## Signed and Unsigned Comparisons

```
.data
val1    DWORD 5
result DWORD ?
.code
mov eax,6
.IF eax > val1
  mov result,1
.ENDIF
```

Generated code:

```
    mov eax,6
    cmp eax,val1
    jbe @C0001
    mov result,1
@C0001:
```

MASM automatically generates an unsigned jump (JBE) because **val1** is unsigned.

## Signed and Unsigned Comparisons

```
.data
val1   SDWORD 5
result SDWORD ?
.code
mov eax,6
.IF eax > val1
  mov result,1
.ENDIF
```

Generated code:

```
    mov eax,6
    cmp eax,val1
    jle @C0001
    mov result,1
@C0001:
```

MASM automatically generates a signed jump (JLE) because **val1** is signed.

## Signed and Unsigned Comparisons

```
.data
result DWORD ?
.code
mov ebx,5
mov eax,6
.IF eax > ebx
  mov result,1
.ENDIF
```

Generated code:

```
    mov ebx,5
    mov eax,6
    cmp eax,ebx
    jbe @C0001
    mov result,1
@C0001:
```

MASM automatically generates an unsigned jump (JBE) when both operands are registers . . .

# Signed and Unsigned Comparisons

```
.data
result SDWORD ?
.code
mov ebx,5
mov eax,6
.IF SDWORD PTR eax > ebx
  mov result,1
.ENDIF
```

Generated code:

```
    mov ebx,5
    mov eax,6
    cmp eax,ebx
    jle @C0001
    mov result,1
@C0001:
```

. . . unless you prefix one of the register operands with the SDWORD PTR operator. Then a signed jump is generated.

## .REPEAT Directive

Executes the loop body before testing the loop condition associated with the .UNTIL directive.

Example:

```
; Display integers 1 – 10:

mov eax,0
.REPEAT
   inc eax
   call WriteDec
   call Crlf
.UNTIL eax == 10
```

## .WHILE Directive

Tests the loop condition before executing the loop body The .ENDW directive marks the end of the loop.

Example:

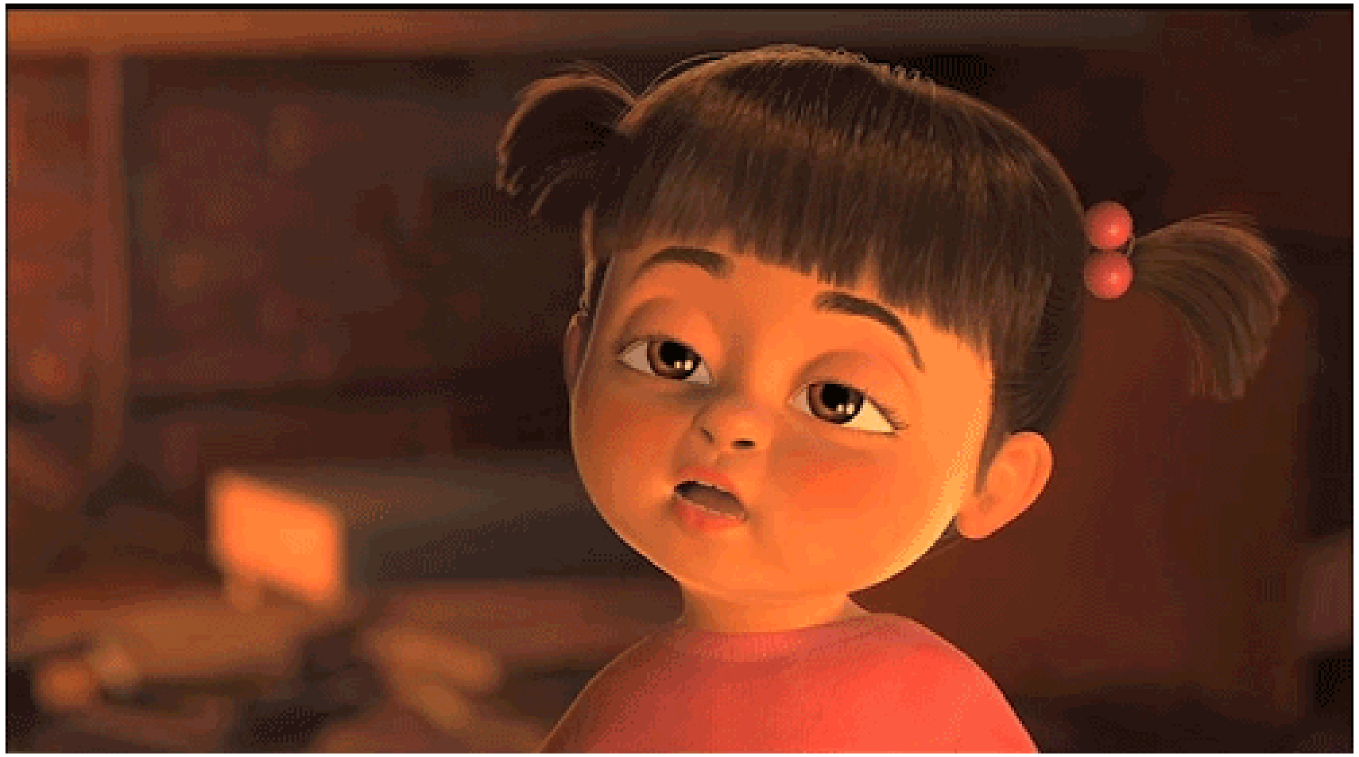
```
 ; Display integers 1 – 10:

mov eax,0
 .WHILE eax < 10
    inc eax
    call WriteDec
    call Crlf
 .ENDW
```

## Summary

- Bitwise instructions (AND, OR, XOR, NOT, TEST)
  - manipulate individual bits in operands
- CMP – compares operands using implied subtraction
  - sets condition flags
- Conditional Jumps & Loops
  - equality: JE, JNE
  - flag values: JC, JZ, JNC, JP, ...
  - signed: JG, JL, JNG, ...
  - unsigned: JA, JB, JNA, ...
  - LOOPZ, LOOPNZ, LOOPE, LOOPNE
- Flowcharts – logic diagramming tool
- Finite-state machine – tracks state changes at runtime

## Thanks a lot



If you are taking a Nap, **wake up**.......Lecture Over