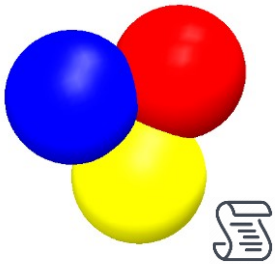

# ObjAsm

*x86 / x64*

# Macros

# 1. Introduction

**ObjAsm** is based on macros clustered over several files depending on their function. This document describes those macros. All files are provided in the **ObjAsm** package.

Implementation details can be checked at the source code level by reading the comments on each file.

# 2. Contents

# 3. Acknowledgements

I would like to express my very great appreciation to all whose valuable and constructive contributions made this work possible. Thank you!

Corrections, comments, suggestions, contributions, etc. may be sent to the MASM32 Forum, or directly mailed to:

ObjAsm@gmx.net

G. Friedrich,

August, 2022

# 4. Nomenclature

The following list describes the rules used to create the library:

1. X prefixes are used to denote a variable or register that can change according to the bitness assembly target. Example: xax means rax in 64 bits, while eax in 32 bits.
2. T file suffixes are used to denote a neutral string encoding.
3. X file suffixes are used to denote bitness-neutral code.
4. P file suffixes denote platform-independent code, usually leaf procedures.
5. Other file suffixes were used to identify the purpose of the code.

# 5. Abbreviations

BNC: Bitness Neutral Code
COM: Component Object Model
DLL: Dynamic Link Library
GUID: Globally Unique Identifier
HLL: High Level Language
ID: Identifier
IID: Interface Identifier
ZTC: Zero Terminating Character
→: Pointer to

# 6. Notes

The links in section 7 assume that this document is on the same drive as the **ObjAsm** installation and that it was done at the root level of the drive, e.g. D:\ObjAsm\...

# 7. Macros

```
Macro:      StringA / ($)CStrA / ($)DStrA / ($)JStrA
File:       \ObjAsm\Code\Macros\AStrings.inc
Purpose:    Place an ANSI string in the .const, .data or .code segment.
Arguments:  Arg1: Reference name (optional).
            Arg2: Quoted string text.
Return:     Nothing / Reference to the string.


Macro:      $Ofs(C/T/D/J)StrA
File:       \ObjAsm\Code\Macros\AStrings.inc
Purpose:    Place an ANSI string in the S_CONST, S_DATA or S_TEXT segment.
Arguments:  Arg1: Quoted string text.
Return:     String offset.


Macro:      BString / ($)CBStr / ($)TBStr / ($)DBStr / ($)JBStr
File:       \ObjAsm\Code\Macros\BStrings.inc
Purpose:    Place an BSTR string in the S_CONST, S_DATA or S_TEXT segment.
Arguments:  Arg1: Reference name (optional).
            Arg2: Quoted string text.
Return:     Nothing / Reference to the string.
Notes:      - Quotation marks can be used as usual. See example.
            - Partial input strings can be separated by commas.
            - Break input lines with "\".
            - Empty input strings ("" or '') causes an error.
            - Numeric inputs in word range are possible.
Example:    CBStr MyBStr, 'Note: "', "Director's cut", '"', 13, 10
            Resulting BStr:   Note: "Director's cut" + CRLF


Macro:      $Ofs(C/T/D/J)BStr
File:       \ObjAsm\Code\Macros\BStrings.inc
Purpose:    Place an BSTR string in the S_CONST, S_TEXT, S_DATA segment.
Arguments:  Arg1: Quoted string text.
Return:     String offset.


Macro:      CDLL_InsertAfter
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Insert an object after another in the linked list.
Arguments:  Arg1: → Member to insert after.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is rdx.


Macro:      CDLL_InsertBefore
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Insert an object before another in the linked list.
Arguments:  Arg1: → Member to insert before.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is rdx.


Macro:      LinkedList_InsertFirst
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Insert a new member to the end of the Linked List.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is rdx.
            Arg4: Optional auxiliar register


Macro:      LinkedList_InsertLast
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Insert a new member to the end of the Linked List.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is rdx.
            Arg4: Optional auxiliar register


Macro:      CDLL_Remove
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Remove a member from the linked list.
            If the removed item is the first, the second takes its place.
```

```
            If the list does not contain any item, the pFirst pointer is set to NULL.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to remove.
            Arg3: Auxiliar register, default is rcx.
            Arg4: Auxiliar register, default is rdx.


Macro:      CDLL_RemoveFirst
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Remove the first member from the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: Auxiliar register.
            Arg3: Auxiliar register.


Macro:      CDLL_RemoveLast
File:       \ObjAsm\Code\Macros\CDLL.inc
Purpose:    Remove the last member from the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: Auxiliar register.
            Arg3: Auxiliar register.


Macro:      DbgSaveContext
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Save CPU & FPU registers & flags onto stack and aligns it. This way, all subsequent
            invocations have an aligned stack and a cleared direction flag. Additionally, a
            Critacal Section is passed to avoid racing conditions.
Arguments:  Arg1: Handle the passed argument as a Value (FALSE) or a reference (TRUE).
            Arg2: Var to be saved in rbx/ebx/bx/bl or the address of Var in xbx.
Return:     Nothing.
Note:       On entry, the stack doesn't need to be aligned


Macro:      DbgLoadContext
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Load CPU & FPU registers & flags from stack and restores the original stack alignment.
Arguments:  None.
Return:     Nothing.


Macro:      DbgSetDestWnd
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Set the global symbol ??DbgDstWnd.
Arguments:  Arg1: Name of the child window of the Debug Center MDI target to which the information
            is directed.
Return:     Nothing.


Macro:      DbgShowSrcInfo
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output source line info on the debug device.
Arguments:  Arg1: Optional destination Window name.
Return:     Nothing.


Macro:      DbgShowTxtInfo
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output additional text info on the debug device.
Arguments:  Arg1: Text.
            Arg2: Optional destination Window name.
Return:     Nothing.


Macro:      DbgOutHex
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a number in hexadecimal on the debug output device.
Arguments:  Arg1: Text displayed befor the value.
            Arg2: Output color of PreText (defaut is black).
            Arg3: Symbol. Value or address must be stored in rbx/ebx/bx/bl.
            Arg4: Output color (default = black).
            Arg5: Optional destination Window name.
Note:       If register addressing is used, a size prefix is required.
            i.e.: DbgHexHex DWORD ptr [rcx]
            THIS MACRO TRASHES XDI & XSI


Macro:      DbgLine
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Draw a single line on the debug device.
Arguments:  Arg1: Optional destination Window name.
```

```
Macro:      DbgLine2
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Draw a double line on the debug device.
Arguments:  Arg1: Optional destination Window name.


Macro:      DbgText
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a text on the debug device.
Arguments:  Arg1: Text.
            Arg2: Optional destination Window name.


Macro:      DbgTextF
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output formated a text on the debug device.
Arguments:  Arg1: Optinal text color.
            Arg2: Optional destination Window name.
            Arg3: Quoted format string.
            Arg 4..n: optional arguments.
Example:    DbgTextF "Output", $RGB(000,000,000), "Data = '%s'", xax


Macro:      DbgWarning
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a text on the debug device on red color.
Arguments:  Arg1: Text.
            Arg2: Optional destination Window name.


Macro:      DbgWriteA
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a string on the debug device.
Arguments:  Arg1: → ANSI String.
            Arg2: Optional destination Window name.


Macro:      DbgWriteW
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a string on the debug device.
Arguments:  Arg1: → WIDE String.
            Arg2: Optional destination Window name.


Macro:      DbgWriteFA
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a string on the debug device.
Arguments:  Arg1: Optinal text color.
            Arg2: Optional destination Window name.
            Arg3: Quoted format string.
            Arg 4..n: optional arguments.
Note:       Dont use xbx as argument register. it is used internally


Macro:      DbgWriteFW
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a string on the debug device.
Arguments:  Arg1: Optinal text color.
            Arg2: Optional destination Window name.
            Arg3: Quoted format string.
            Arg 4..n: optional arguments.


Macro:      DbgStrA
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a specified string on the debug device.
Arguments:  Arg1: ANSI String.
            Arg2: Additional information.
            Arg3: Optional destination Window name.


Macro:      DbgStrW
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a specified WIDE string on the debug device.
Arguments:  Arg1: WIDE string.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
```

```
Macro:      DbgStrCA
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a limited ANSI string on the debug device.
Arguments:  Arg1: ANSI string.
            Arg2: Character count (fix number, like 8).
            Arg3: Additional information.
            Arg4: Optional destination Window name.


Macro:      DbgStrCW
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a limited WIDE string on the debug device.
Arguments:  Arg1: WIDE string.
            Arg2: Character count.
            Arg3: Additional information.
            Arg4: Optional destination Window name.


Macro:      DbgHex
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Display a number in hexadecimal format on the debug device.
Arguments:  Arg1: Number (register or symbol).
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Note:       If register addressing is used, a size prefix is required.
            i.e.: DbgHex DWORD ptr [rcx]


Macro:      DbgBin
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a number in binary format on the debug device.
Arguments:  Arg1: Number (register or symbol). Max 32 bits.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Note:       If register addressing is used, a size prefix is required.
            i.e.: DbgBin DWORD ptr [rcx]


Macro:      DbgDec
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a number in decimal format on the debug device.
Arguments:  Arg1: Number (register or symbol). Max 32 bits
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Note:       If register addressing is used, a size prefix is required.
            i.e.: DbgDec DWORD ptr [rcx]


Macro:      DbgFloat
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Display a floating point number in decimal format on the debug device.
Arguments:  Arg1: Floating point number (REAL4 or REAL8)
            Arg2: Additional information.
            Arg3: Optional destination Window name.


Macro:      DbgBmp
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a specified bitmap on the debug device.
Arguments:  Arg1: Bitmap HANDLE.
            Arg2: Optional destination Window name.


Macro:      DbgBmpFromDC
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a specified bitmap on the debug device.
Arguments:  Arg1: DC HANDLE.
            Arg2: Optional destination Window name.


Macro:      DbgMem
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output a memory block on the debug device.
Arguments:  Arg1: → memory block.
            Arg2: Size of memory block.
            Arg3: Output format (DBG_MEM_STR, DBG_MEM_[U]I??, DBG_MEM_R?, DBG_MEM_H??).
            Arg4: Additional information.
            Arg5: Optional destination Window name.


Macro:      DbgGlobalMemUsage
```

```
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Display global memory usage on the debug window.
Arguments:  Arg1: Additional information.
            Arg2: Optional destination Window name.


Macro:      DbgFPU
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output the content of the FPU registers on the debug device.
Arguments:  Arg1: Additional information.
            Arg2: Optional destination Window name.


Macro:      DbgMessage
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Translate a windows message to a string and output it to the debug device.
Arguments:  Arg1: Windows message.
            Arg2: Additional information.
            Arg3: Optional destination Window name.


Macro:      DbgApiError
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Translate an API error to a string and output it to the debug device.
Arguments:  Arg1: Additional information.
            Arg2: Optional destination Window name.


Macro:      DbgComError
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Translate a COM error to a string and output it to the debug device.
Arguments:  Arg1: COM error code.
            Arg2: Additional information.
            Arg3: Optional destination Window name.


Macro:      DbgObject
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output object variables to the debug device.
Arguments:  Arg1: Instance::ClassName.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Return:     Nothing.
Note:       ebp is assumed to nothing


Macro:      DbgVMT
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output offsets contained in a VMT.
Arguments:  Arg1: Instance::ClassName.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Return:     Nothing.
Note:       ebp is assumed to nothing


Macro:      DbgIMT
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Output offsets contained in a IMT.
Arguments:  Arg1: Instance::ClassName.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Return:     Nothing.
Note:       ebp is assumed to nothing


Macro:      DbgAttach
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Break execution and attaches the system debugger. If already loaded, nothing is done.
Arguments:  None.


Macro:      DbgBreak
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Break execution.
Arguments:  None.


Macro:      DbgCloseAll
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Close all child windows of the Debug window.
```

```
Arguments:  None.


Macro:      DbgCloseTxt
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Close specific text child window of the Debug window.
Arguments:  Arg1: Target Debug Window name.


Macro:      DbgCloseBmp
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Close specific bitmap child window of the Debug window.
Arguments:  Arg1: Target Debug Window name.


Macro:      DbgClearAll
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Clear the content of all child windows of the Debug window.
Arguments:  None.


Macro:      DbgClearTxt
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Clear the content of a specific text child window of the Debug window.
Arguments:  Arg1: Target Debug Window name.


Macro:      DbgClearBmp
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Clear the content of a specific bitmap child window of the Debug window.
Arguments:  Arg1: Target Debug Window name.


Macro:      Fix
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Reminder text echoed at compile time.
Arguments:  Arg1: (optional) Text.


Macro:      ASSERT
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Display a message if an argument value is FALSE.
Arguments:  Arg1: Value.
            Arg2: Additional information.
            Arg3: Optional destination Window name.
Note:       = as unequality comparison.


Macro:      ResGuard_Show
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Show the result of ResGuard system activity.
Arguments:  None.


Macro:      ResGuard_Start
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Start activity of the ResGuard system.
Arguments:  None.


Macro:      ResGuard_Stop
File:       \ObjAsm\Code\Macros\Debug.inc
Purpose:    Stop activity of the ResGuard system.
Arguments:  None.


Macro:      FlexArg
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Check the Arg parameter to determine if it is blank, contains NULL, a quoted text or
            an ordinal value to generate a proper segment memory allocation.
Arguments:  Arg1: Input.
Return:     Nothing.


Macro:      DIALOGEX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a DIALOGEX template structure.
Arguments:  Structure arguments.
Return:     Nothing.
```

```
Macro:      CONTROL
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a control template structure.
Arguments:  Structure arguments.
Return:     Nothing.


Macro:      PUSHBUTTON
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a PUSHBUTTON control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      DEFPUSHBUTTON
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a DEFPUSHBUTTON control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      PUSHBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a PUSHBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      AUTO3STATE
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a AUTO3STATE control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      STATE3
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a STATE3 control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      AUTOCHECKBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a AUTOCHECKBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      CHECKBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a CHECKBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      RADIOBUTTON
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a RADIOBUTTON control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      AUTORADIOBUTTON
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a AUTORADIOBUTTON control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      GROUPBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a GROUPBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      LTEXT
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a LTEXT control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
```

```
Return:     Nothing.


Macro:      RTEXT
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a RTEXT control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      CTEXT
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a CTEXT control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      COMBOBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a COMBOBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      LISTBOX
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a LISTBOX control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      EDITTEXT
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a EDITTEXT control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      SCROLLBAR
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a SCROLLBAR control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      ICON
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a ICON control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      TREEVIEW
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a TREEVIEW control template structure.
Arguments:  dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      LISTVIEW
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a LISTVIEW control template structure.
Arguments:  dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      HYPERLINK
File:       \ObjAsm\Code\Macros\DlgTmpl.inc
Purpose:    Create a HYPERLINK control template structure.
Arguments:  sTitle, dCtlID, wXPos, wYPos, wWidth, wHeight, dStyle, dExStyle, dHelpID, wExtra
Return:     Nothing.


Macro:      .try
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Instructions between this .try and the next .Catch or Finally will be protected.
Arguments:  Arg1: Name of the XFRAME which will deal with exceptions.
            Arg2: Flags for the exception handler.
            Arg3: → some informations for the exception handler.
```

```
Macro:      .catch
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    The following instructions will be executed if an exception is raised.
Arguments:  Arg1: Name of the Exception Handler which will deal with the exceptions.
            Arg2: (optional) Name of the exception record that will be filled on an exception.
Note:       If we reach this place, some register are filled with the following content:
            eax → EXCEPTION_RECORD.
            ecx = Exception code.
            edx = pInformation passed to the .try macro.


Macro:      .endcatch
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Follow immediately instructions guarded by a .catch.
Arguments:  Name of the handler which will deal with exceptions.


Macro:      .finally
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Following instructions will run even if an exception which can be handled
            by the exception handler is raised.
Arguments:  Name of the handler which will deal with exceptions.
Note:       If we reach this place, some register are filled with the following content:
            eax → EXCEPTION_RECORD or NULL if no exception was raised.
            ecx = Exception code or unchanged if no exception was raised.
            edx = pInformation passed to the .try macro or unchanged if no exception was raised.


Macro:      .endfinally
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Follow immediately instructions guarded by a .finally.


Macro:      xMsg
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Allow to define a message for an exception.
Arguments:  Arg1: Name of the handler which will deal with exceptions.
            Arg2: Message associated with the exception.


Macro:      Throw
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Raise an exception.
Arguments:  Arg1: Code of the exception to raise.
            Arg2: Name of a xMsg.
Notes:      If Msg is blank the exception is raised with the message szSehNoMsg in order to
            allowing the Top Level to know if the user provided a message or not.


Macro:      InstallTLH
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Install a Top Level Handler.
Arguments:  Arg1: → Top Level handler to install.
            Arg2: Var which will hold the pointer to the old handler.


Macro:      RestoreTLH
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Reinstall the old top level handler.
Arguments:  var holding the pointer to the old handler.


Macro:      $GetExceptionCode
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Retrieve the Exception Code.
Arguments:  None.


Macro:      $GetExceptionAddr
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Retrieve the Exception address.
Arguments:  None.


Macro:      CreateExceptionHandler
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Create a handler which can deal with one or more exceptions.
Arguments:  Arg1: Name of the handler to be created.
            Arg2: Address of Unwind code.
            Arg3: List of exceptions codes to deal with.
Notes:      The error handler MUST be declared as a C procedure for proper stack cleanup
```

```
Macro:      TopLevelHandler
File:       \ObjAsm\Code\Macros\Exception32.inc
Purpose:    Create a top level handler to deal with exceptions not catched.
Arguments:  Arg1: Name of the top level handler to be created.


Macro:      .try
File:       \ObjAsm\Code\Macros\Exception64.inc
Purpose:    Initialize a protected area.
Arguments:  None.
Return:     Nothing.


Macro:      .catch
File:       \ObjAsm\Code\Macros\Exception64.inc
Purpose:    Finalizes a protected area. The following code is executed in case of an exception.
Arguments:  None.
Return:     Nothing.
Note:       This macro can be ommited.


Macro:      .endcatch
File:       \ObjAsm\Code\Macros\Exception64.inc
Purpose:    32 bit compatibility macro.
Arguments:  None.
Return:     Nothing.
Note:       This macro can be ommited.


Macro:      .finally
File:       \ObjAsm\Code\Macros\Exception64.inc
Purpose:    The following code is always executed.
Arguments:  None.
Return:     Nothing.
Note:       This macro must always be present


Macro:      .finally
File:       \ObjAsm\Code\Macros\Exception64.inc
Purpose:    32 bit compatibility macro.
Arguments:  None.
Return:     Nothing.
Note:       This macro can be ommited.


Macro:      fildReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Load a CPU register containing an integer onto the FPU stack.
Arguments:  Arg1: Register name.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fistReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Store the integer part of the FPU st(0) register to a CPU register.
Arguments:  Arg1: Register name.
Return:     Nothing.


Macro:      fistpReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Store the integer part of the FPU st(0) register to a CPU register & pops it.
Arguments:  Arg1: Register name.
Return:     Nothing.


Macro:      fldReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Load a CPU register containing a REAL4 onto the FPU stack.
Arguments:  Arg1: Register name.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fstReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Store the FPU st(0) register to a CPU register in REAL4 format.
Arguments:  Arg1: Register name.
```

```
Return:     Nothing.


Macro:      fstpReg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Store and pops the FPU st(0) register to a CPU register in REAL4 format.
Arguments:  Arg1: Register name.
Return:     Nothing.


Macro:      fSgn
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compute the signum function of the content os st0.
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.
            Requires .586 option for the fcomip instruction.
            Uses rax register.


Macro:      fCheckStatus
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Check the status word of the FPU to detect fp computation errors.
Arguments:  Arg1: Status bit(s) to be checked.
Return:     Nothing.
Note:       The result affects the zero flag, that can be used to decide.


Macro:      fGetFlags
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Set EFlags with FPU StatusWord flags.
Arguments:  None.
Return:     Nothing.
Note:       Uses ax.


Macro:      fSetPrecision
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Set the precision bits in the control register.
Arguments:  Arg1: Precision (REAL4, REAL8, REAL10).
Return:     Nothing.
Notes:      00 = 24 bits (REAL4)
            01 = Not used
            10 = 53 bits (REAL8)
            11 = 64 bits (REAL10) (this is the initialized state)


Macro:      fMin
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compare two floats, returns the smaller value on the FPU.
Arguments:  Arg1: first value.
            Arg2: Second value.
Return:     Smallest value in st(0).
Note:       Uses ax.
            st7 must be empty.


Macro:      fMax
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compare two floats, returns the larger on the FPU stack.
Arguments:  Arg1: first value.
            Arg2: Second value.
Return:     Largest value in st(0).
Note:       Uses ax.
            st7 must be empty.


Macro:      fAbsMin
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compare two floats, returns the absolute smaller value on the FPU stack.
Arguments:  Arg1: first value.
            Arg2: Second value.
Return:     Smallest absolute value in st(0).
Note:       Uses ax.
            st6 and st7 must be empty.


Macro:      fAbsMax
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compare two floats, returns the absolute larger value on the FPU.
Arguments:  Arg1: first value.
```

```
            Arg2: Second value.
Return:     Largest absolute value in st(0).
Note:       Uses ax.
            st6 and st7 must be empty.


Macro:      fRnd
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Round the content of st0.
Arguments:  None.
Return:     Nothing.


Macro:      fRndUp
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Round up the content of st0.
Arguments:  None.
Return:     Nothing.


Macro:      fRndDn
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Round down the content of st0.
Arguments:  None.
Return:     Nothing.


Macro:      fInt
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate the integer part of the content of st0.
Arguments:  None.
Return:     Nothing.


Macro:      fFrac
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate the fractional part of the content of st0.
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fRound
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Round the content of st0 to x decimals.
Arguments:  Arg1: Decimals.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fExp2
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compute 2^x = 2^st0.
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fExpN
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compute st(0) = e^st0.
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fLogN
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate the neperian logarithm of st0 and store it in st0.
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fExpT
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compute st0 = 10^st0.
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.
```

```
Macro:      fLogT
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate the logarithm base 10 of st0 and sotres it in st0.
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fPower
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Compute st0 = st0^st1 (x^y).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fLog
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate the logarithm base st1 of st0 and store the result in st0.
Arguments:  None.
Return:     Nothing.
Note:       st6 and st7 must be empty.
            Log[st1](st0) = Log2(st0) / Log2(st1).


Macro:      fFitTrigRange
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Reduce the value in st0 to fit in the range of trig. functions: |x| < 2^63.
Arguments:  st0.
Return:     Nothing.
Note:       st6 and st7 must be empty.
            The result of fprem1 depends on the state of the FPU rounding bits.


Macro:      fTan
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = tan(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fSec
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Sec(st0).
Arguments:  None.
Return:     Nothing.


Macro:      fCsc
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Csc(st0).
Arguments:  None.
Return:     Nothing.


Macro:      fCot
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Cot(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcSin
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcSin(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCos
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCos(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.
```

```
Macro:      fArcTan
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcTan(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcSec
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcSec(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCsc
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCsc(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCot
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCot(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fSinh
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Sinh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fCosh
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Cosh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fTanh
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Tanh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fSech
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Sech(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fCsch
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Csch(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fCoth
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = Coth(st0).
Arguments:  None.
Return:     Nothing.
Note:       st5, st6 and st7 must be empty.


Macro:      fArcSinh
```

```
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcSinh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCosh
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCosh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcTanh
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcTanh(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcSech
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcSech(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCsch
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCsch(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fArcCoth
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Calculate st0 = ArcCoth(st0).
Arguments:  None.
Return:     Nothing.
Note:       st7 must be empty.


Macro:      fUnload
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Unload several stack registers
Arguments:  Number of FPU registers.
Return:     Nothing.


Macro:      fstStrA
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Create a string representation of the content of the st0 FPU register.
Arguments:  Arg1: → destionation buffer (at best 20 bytes + padding bytes).
            Arg2: Number of padding bytes.
            Arg3: Number of decimal places.
            Arg4: Format flag (f_NOR or f_SCI)
Return:     Operation code.
Note:       st4, st5, st6 and st7 must be empty.
            Destination buffer should have at least 20 bytes (add padding bytes)


Macro:      fLdStrA
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Load a string representation of a floating point number into
            the st0 FPU register.
Arguments:  Arg1: → string.
Return:     Operation code.
Note:       st4, st5, st6 and st7 must be empty.


Macro:      fjg
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates greater.
Arguments:  Arg1: Destination label.
Return:     Nothing.
```

```
Macro:      fjl
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates less.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fje
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates equal.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fjne
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates not equal.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fjge
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates greater
            or equal.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fjle
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates less
            or equal.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fjnc
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Jump to label if the result of the previous fcom instruction indicates that the
            arguments are non-comparable.
Arguments:  Arg1: Destination label.
Return:     Nothing.


Macro:      fcmp
File:       \ObjAsm\Code\Macros\fMath.inc
Purpose:    Fast way to compare REAL4 floats with sign using the CPU.
Arguments:  Arg1: first comparand.
            Arg2: Second comparand.
Return:     Modified flags.
Note:       This macro distinguishes between -0.0 and +0.0


Macro:      LDLL_InsertAfter
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Insert an object after another in the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert after.
            Arg3: → Member to insert.
            Arg4: Auxiliar register, default is xdx.


Macro:      LDLL_InsertBefore
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Insert an object before another in the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert before.
            Arg3: → Member to insert.
            Arg4: Auxiliar register, default is xdx.


Macro:      LinkedList_InsertFirst
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Insert a new member to the end of the Linked List.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is xdx.
```

```
Macro:      LinkedList_InsertLast
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Insert a new member to the end of the Linked List.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to insert.
            Arg3: Auxiliar register, default is xdx.


Macro:      LDLL_Remove
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Remove a member from the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: → Member to remove.
            Arg3: Auxiliar register, default is rcx.
            Arg4: Optional auxiliar register.


Macro:      LDLL_RemoveFirst
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Remove the first member from the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: Auxiliar register


Macro:      LDLL_RemoveLast
File:       \ObjAsm\Code\Macros\LDLL.inc
Purpose:    Remove the last member from the linked list.
Arguments:  Arg1: → Control structure or reference.
            Arg2: Auxiliar register


Macro:      LSLL.InsertAfter
File:       \ObjAsm\Code\Macros\LSLL.inc
Purpose:    Insert an object after another in the linked list.
Arguments:  Arg1: → Member to insert after.
            Arg2: → Member to insert.
            Arg3: Auxiliar register.


Macro:      LSLL_RemoveAfter
File:       \ObjAsm\Code\Macros\LSLL.inc
Purpose:    Remove a member from the linked list.
Arguments:  Arg1: → Member to remove after.
            Arg2: Auxiliar register, default is xcx.


Macro:      Static
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Allocate a variable in the Data segment
Arguments:  Arg1: Name.
            Arg2: Type.
            Arg3: Initial value.
Return:     Nothing.


Macro:      MemAlloc / $MemAlloc
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Allocate a memory block on global memory.
Arguments:  Arg1: Memory block size. For Windows muust be less than 7FFF8h.
            Arg2: Optional memory allocation flags:
            MEM_DEFAULT, MEM_INIT_ZERO, MEM_GENERATE_EXEPTION, MEM_NO_SERIALIZE.
Return:     xax → allocated memory block. NULL or exception if failed.
            UEFI returns an mem block aligned to 8.


Macro:      MemReAlloc
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Reallocate a block of memory.
Arguments:  Arg1: Memory POINTER obtained with MemAlloc/MemReAlloc.
            Arg2: New memory block size.
            Arg3: Optional memory allocation flags:
            MEM_INIT_ZERO, MEM_GENERATE_EXEPTION, MEM_NO_SERIALIZE, MEM_NO_MOVE
Return:     rax → allocated memory block. NULL or exception if failed.


Macro:      MemFree
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Free a memory block previously allocated with MemAlloc or MemReAlloc.
Arguments:  Arg1: → allocated memory block.
            Arg2: Optional memory allocation flags: MEM_NO_SERIALIZE, MEM_SAFE_FREE
Return:     eax = TRUE if succeeded, otherwise FALSE.
```

```
Macro:      MemCheck
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Validates a memory block on the process heap.
Arguments:  Arg1: → allocated memory block. If NULL, the whole heap is checked.
Return:     eax = non FALSE if succeeded, otherwise FALSE.


Macro:      (x)BitClear
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Reset a bit in a value.
Arguments:  Arg1: Destination.
            Arg2: Bit to be reset (ie: BIT12)
Return:     Nothing.


Macro:      (x)BitMask
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    AND a bit in a value.
Arguments:  Arg1: Destination.
            Arg2: Bit to be masked (ie: BIT12)
Return:     Nothing.


Macro:      BitSet
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Set a bit in a value.
Arguments:  Arg1: Destination.
            Arg2: Bit to be set (ie: BIT15)
Return:     Nothing.


Macro:      (x)BitToggle
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Toggle a bit in a value.
Arguments:  Arg1: Destination.
            Arg2: Bit to be toggled (ie: BIT15)
Return:     Nothing.


Macro:      .ifBitSet
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Start a conditional clauses, that must be terminated with .endif.
            Examine if any of the indicated bits are set.
Arguments:  Arg1: Bit holding variable.
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      .elseIfBitSet
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Continue a conditional clauses, that must be terminated with .endif.
            Examine if any of the indicated bits are set.
Arguments:  Arg1: Bit holding variable.
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      .ifBitClr
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Start a coditional clauses, that must be terminated with .endif.
            Examine if all of the indicated bits are not set.
Arguments:  Arg1: Bit holding variable.
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      .elseIfBitClr
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Start a coditional clauses, that must be terminated with .endif.
            Examine if all of the indicated bits are not set.
Arguments:  Arg1: Bit holding variable.
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      JmpIfBitSet
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Jump if a bit is set.
Arguments:  Arg1: Bit holding variable.
```

```
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      JmpIfBitClr
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Jump if a bit is not set.
Arguments:  Arg1: Bit holding variable.
            Arg2: Bit to be tested for (ie: BIT12)
Return:     Nothing.


Macro:      CloneGuid
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Copy the content of a GUID from a location to another.
Arguments:  Arg1: Destination GUID.
            Arg2: Source GUID.
            Arg3: optional register to be used.
Return:     Nothing.


Macro:      ($)StackAlloc
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Reserve a chunk of memory from stack.
Arguments:  Arg1: Requested memory size in bytes.
Return:     rax → Buffer begin.
Notes:      StackAlloc and StackFree must always be paired and no values ought remain in the stack
            between these macros.
            Stack probing is performed automatically.


Macro:      StackFree
File:       \ObjAsm\Code\Macros\Memory.inc
Purpose:    Release the reserved stack space with StackAlloc.
Arguments:  None.
Return:     Nothing.


Macro:      SysSetup
File:       \ObjAsm\Code\Macros\Model.inc
Purpose:    Load and configure standard modules.
Arguments:  Support level: OOP, WIN32/WIN64, CON32/CON64, NUI32/NUI64, LIB32/LIB64.
            Debug Switches : SHOWINFO, TRACE, RESGUARD, STKGUARD
            Source identification: "..." (also used as LOG file name)
            Default string type: ANSI_STRING or WIDE_STRING
Return:     Nothing.


Macro:      SysInit
File:       \ObjAsm\Code\Macros\Model.inc
Purpose:    Initialize the model and internal variables.
Arguments:  Windows:
            Arg1: Instance HANDLE. If not specified, the Module HANDLE is loaded into hInstance.
            Arg2: unused.
            UEFI:
            Arg1: Image HANDLE.
            Arg2: → SystemTable.


Macro:      SysDone
File:       \ObjAsm\Code\Macros\Model.inc
Purpose:    Finalize the model.
Arguments:  None.


Macro:      ObjNamespace
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Set the current object namespace.
Arguments:  Arg1: Space name.


Macro:      $ObjInst
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return the mangled object name instance.
Arguments:  Arg1: Object Expression     [Namespace:]ObjectName.
Example:    .data
            MyObject $ObjInst(Primer)


Macro:      $ObjMthd
File:       \ObjAsm\Code\Macros\Objects.inc
```

```
Purpose:    Return the mangled object method name.
Arguments:  Arg1: Object name.
Example:    .data
            MyObject $ObjMtd(pop)


Macro:      $Obj
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return the mangled object name.
Arguments:  Arg1: Object Expression    [Namespace:]ObjectName.


Macro:      $ObjPtr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return the mangled object pointer name.
Arguments:  Arg1: Object Expression: [Namespace:]ObjectName.


Macro:      $ObjTmpl
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return the mangled object template name.
Arguments:  Arg1: Object Expression: [Namespace:]ObjectName.


Macro:      ExterndefMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Declares an external defined method.
Arguments:  Arg1: Method Expression: ObjectName.MethodName.


Macro:      BuildObjInherPath
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to put in 1 symbol the whole interitance path starting from Primer to
            the final object and viceversa.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled ancestor name.
Return:     TRUE = OK, FALSE = failure.


Macro:      Object
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Start object definition. Here starts the magic
Arguments:  Arg1: Object Expression: Namespace:ObjectName.
            Arg2: Unique number to identificate the object type at run-time.
            Arg3: Ancestor Expression: [Namespace:]AncestorName. (single inheritance).


Macro:      ObjectEnd
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Terminate the declaration of an object and defines the following structures:
            ObjectName: Uninitialized object Template.
            ??ObjectName_VMT: Uninitialized Virtual Method Table (VMT).
            ??ObjectName_IMT: Uninitialized Interface Method Table (IMT).
            ??ObjectName_DMT: Uninitialized VMT + IMT.
            ??ObjectName_Events: Initialized ??EVENT_ENTRY structures list.
            TPL_ObjectName: Initialized ObjectName template.
Arguments:  None.


Macro:      Private
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Methods that follows this directive are defined with private scope.
Arguments:  None.


Macro:      PrivateEnd
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Methods that follows this directive are defined with public scope.
Arguments:  None.


Macro:      InterfaceAbstract
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a placeholder for an interface method that can be overridden.
            Calling an abstract method without overriding it, causes an GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      InterfaceMethod
File:       \ObjAsm\Code\Macros\Objects.inc
```

```
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      VirtualAbstract
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a placeholder for a virtual method that can be overridden.
            Calling an abstract method without overriding it, causes a GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      VirtualMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      DynamicAbstract
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a placeholder for a dynamic method that can be overridden.
            Calling an abstract method without overriding it, causes an GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      DynamicMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that is unique for each instance of the object.
            The implementation of the method must be placed in the code Segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      StaticMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that is common to each instance of the object and it is called
            directly, avoiding the indirection over the DMT. This implements early binding.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      InlineMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      RedefineMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Redefine a method (static, virtual, dynamic or interface) at compile-time.
Arguments:  Arg1: Method Name to be redefined.
            Arg2: List of new argument types.


Macro:      ObsoleteMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Invalidate a method definition (virtual or dynamic) at compile-time.
Arguments:  Arg1: Method name to be erased.


Macro:      DefineEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to define an event. Don't called directly.
```

```
Arguments:  Arg1: Mangled method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      Event
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that responds to an event, like a Windows message.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.
Note:       Inline Methods are not allowed.


Macro:      IDispEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a method that responds to the COM IDispatch.Invoke call.
Arguments:  Arg1: Method name.
            Arg2: DispID.
            Arg3: Flags (DISPATCH_METHOD, DISPATCH_PROPERTYGET, DISPATCH_PROPERTYPUT,
            DISPATCH_PROPERTYPUTREF).


Macro:      StaticEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    This is a shortcut for StaticMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      VirtualEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    This is a shortcut for VirtualMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      InterfaceEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    This is a shortcut for InterfaceMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      DynamicEvent
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    This is a shortcut for DynamicMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      DefineVariable
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define a variable (Unique to each instance of the object).
Arguments:  Arg1: Variable name.
            Arg2: Variable type.
            Arg3/4: Optional initial value/structure initial value(s).


Macro:      RedefineVariable
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Redefine a variables initial value.
Arguments:  Arg1: Variable name.
            Arg2: New initial value.


Macro:      LoadObjects
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Load all object structures from a precompiled file (ObjectName.lib).
Arguments:  Arg1: File name list (without extention) that contain the object data to load.


Macro:      MakeObjects
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Load all object data from a file.
Arguments:  Arg1: File name list (without extention) that contain the object data to compile.


Macro:      VirtualObjects
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Create the object data structures to reference an external object instance.
Arguments:  Arg1: File name list (without extention) that contain the object data definitions.
```

```
Macro:      PrototypeMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to perform the necessary method definitions for the compiler.
Arguments:  Arg1: Mangled object name
            Arg2: Method name.
            Arg3: List of argument types.


Macro:      CountInterfaceMethods
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that counts all interface methods belonging to an object.
Arguments:  Arg1: Mangled object name.
Return:     Count of Interface methods.


Macro:      CreateOMList
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create a list of methods to override.
Arguments:  Arg1: Mangled object name.
Return:     Count of methods to override.


Macro:      CreateOVList
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create a list of variables to override.
Arguments:  Arg1: Mangled object name.
Return:     Count of variables to override.


Macro:      CreateInterfaceMethodTable
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create the structure of the interface method table of an object.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) member initialization.


Macro:      CreateVirtualMethodTable
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create the structure of the virtual method table of an object.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) references.
Note:       This VMT is in reverse order to make place for the IMT


Macro:      CreateTemplate
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that defines the following structures:
            ??ObjectName_Init: Initialized template.
            ??ObjectName_DMT_Init: Initialized VMT + IMT.
            Additionally it creates the object data list in the .data segment.
Arguments:  Arg1: Mangled Object name.


Macro:      CreateDynamicTemplate
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create the dynamic section of the object template.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) references.


Macro:      CreateEventList
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to create the event list of an object.
Arguments:  Arg1: Mangled object name.


Macro:      GetMtdType
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return method type.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
            Retrun: ??MtdType: ??MTDTYPE_UNKNOWN, ..., ??MTDTYPE_DYNAMIC
            ??ImpObj: Mangled object name that implements the method.


Macro:      $IsObsoleteMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Check if a method of an object is obsolete or not.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
```

```
            Retrun: FALSE: is NOT an obsolete method, TRUE: it is an obsolete method.


Macro:      $IsPrivateMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Scan recursively in the inheritance path of a specified method to find out
            if it is a private method.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
            Retrun: TRUE: it is a virtual method, FALSE: is NOT a virtual method.


Macro:      $MethodAddr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Return a method address.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Notes:      - this macro uses the eax/rax register, except for static methods.
            - if the expression contains an instance POINTER, then the returned address is
            relative to this particular instance, otherwise to the object template.
            - can be used inside nested into OCall/ACall/etc. macros.
            - Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Obj.Mtd
            - Mtd


Macro:      New / $New
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Create a new instance of an object copying the object template.
            It is also suitable for object instances located on the stack using the LOCAL keyword.
Arguments:  Arg1: Expression of the form "Instance::Namespace:Object"
            "Instance" is an optional expression used for preallocated instances like when
            stack is used to hold the object.
            "Object" is a required expression, that identifies the object type to
            instantiate.
Return:     rax → created object instance.
            Possible syntax combinations:
            - Preallocated memory
            - Ins::Nsp:Obj
            - Ins::Obj
            - With memory allocation
            - Nsp:Obj
            - Obj


Macro:      Destroy
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Safely free the memory allocated for a particular object instance but calling first
            its destructor method. If not specified, "Done" is assumed.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Ins.Mtd
            - Ins


Macro:      Kill
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Free the memory allocated for a particular object instance but calling first its
            destructor method. If not specified, "Done" is assumed.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Ins.Mtd
            - Ins


Macro:      SetObject
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Assume a register to an object type.
Arguments:  Arg1: Register, i.e.: rax, rbx, ...
            Arg2: (optional) Object name.
            Arg3: (optional) Instance POINTER. Default is pSelf.


Macro:      SetOwner
File:       \ObjAsm\Code\Macros\Objects.inc
```

```
Purpose:    Link a register to an object owner object type and instance at compile-time.
Arguments:  Arg1: Register, i.e.: rax, rbx, ...
            Arg2: Object name.
            Arg3: (optional) Instance POINTER. Default is pSelf.


Macro:      ReleaseObject
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Free the link between a register and an object at compile-time.
Arguments:  Arg1: (Optional) Register, i.e.: rax, rbx, ...


Macro:      Override
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Set a new method address in the dynamic-, virtual-, interfacetable.
Arguments:  Arg1: Instance::Object.Method
            Arg2: Object.Method or procedure address.
Note:       Don't use r11 to pass a procedure address It is used internally.


Macro:      ObjectsInit
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to initialize all object templates calling their Startup procedure.
            No instance creation takes place.
Arguments:  None.


Macro:      ObjectsDone
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro to finalize all object templates calling their Shutdown procedure.
Arguments:  None.


Macro:      $MethodPrologue32
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Method prolog macro.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.
Return:     Total local space (byte count).


Macro:      MethodEpilogue32
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that defines the method epilog.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      $NoFramePrologue32
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Method prolog macro.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      NoFrameEpilogue32
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that defines the NOFRAME method epilog.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      Method
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Define an object method.
```

```
Arguments:  Arg1: Object method name (ObjectName.MethodName).
            Arg2: Method options like "uses".
            Arg3: Method arguments.


Macro:      MethodEnd
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Terminate an object method.
Arguments:  None.


Macro:      $Method
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Returns the mangled object template name.
Arguments:  Arg1: ObjectName.MethodName
Example:    $Method(Object.Method) macro pSelf, Arg1, Arg2
            ...
            endm


Macro:      ExitMethod
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Terminate an object method.
Arguments:  Arg1: Expression.
Note:       - More efficient is the use of i.e. "je @@EOM".
            - Attention with the = should be used.


Macro:      Embed
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Embed a named object instance into the template of another object.
Arguments:  Arg1: Instance name.
            Arg2: Object expression (Namespace:ObjectName).
Note:       - Never call Destroy to free an embedded object. Only call "Done".


Macro:      $PushMethodArgs
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that pushes all method arguments.
Arguments:  Arg1: Argument count.
            Arg2: Method argument list.


Macro:      $GetDeclaredParamCount
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that returns the parameter count of a method definition inside "Object".
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.


Macro:      $GetInstance32
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro for 32 bit that returns the instance pointer in a register.
Arguments:  Arg1: Instance name.
            Arg2: Used register.


Macro:      GetInstance64
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro for 64 bit that returns the instance pointer in the rcx register.
Arguments:  Arg1: Instance name.


Macro:      SetVarargPassedParams
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that sets the total parameter count (excluded pSelf) in the eax register.
Arguments:  Arg1: Method arguments.


Macro:      MethodInvoke
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro that calls a method.
Arguments:  Arg1: Invocation flag: TRUE = use the template, FALSE = use object instance.
            Arg2: Instance name.
            Arg3: Mangled object name.
            Arg4: Mangled method name.
            Arg5: Arguments.
Notes:      If a method has VARARGs, the number of arguments is passed in eax.


Macro:      PreParseExpr
```

```
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Get the positions of the separators of an expression like:
            "InsName::Namespace:ObjName.MtdName"
            | |       |
            ??Pos1 ??Pos2  ??Pos3
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     ??Pos1, ??Pos2 & ??Pos3.


Macro:      $ParseMtdExpr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Parse an input expression for address macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Obj.Mtd
            - Mtd


Macro:      $ParseCallExpr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Parse an input expression for ?Call macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Ins.Mtd
            - Mtd


Macro:      $ParseInstExpr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Parse an input expression for some Debug macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      Possible syntax combinations:
            - Ins::Nsp:Obj
            - Ins::Obj


Macro:      $ParseObjExpr
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Parse an input expression for Object macro.
Arguments:  Arg1: Expression of the form: "NameSpace:ObjName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??ObjExpr (mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Nsp:Obj
            - Obj


Macro:      OCall / $OCall
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Call an object method.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      ACall / $ACall
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Call the ancestor method of an object.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.
```

```
Macro:      TCall / $TCall
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Call a method using the address stored in the object template.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      DCall / $DCall
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Call an object method directly like a normal procedure (like a static method).
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      $GetParamCount
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Helper macro counts parameters in Args list.
Arguments:  Arg1: Method parameter list.


Macro:      $GetImpIfc
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Get the implementing interface name.
Arguments:  Arg1: Interface name.
            Arg2: Method name.


Macro:      ICall / $ICall
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Interface method call.
Arguments:  Arg1: InstancePointer::InterfaceName.MethodName.
            Arg2: Method arguments.


Macro:      AddErrMsg
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    Add an error message and creates the necessary Object Error Tables.
Arguments:  Arg1: Symbol name describing the error, e.g. OBJ_OUT_OF_MEMORY.
            Arg2: [Optional] Error description text.
            Arg3: [Optional] Error ID. Sets the first ID in a series of error messages.


Macro:      LockObjectAccess / UnlockObjectAccess
File:       \ObjAsm\Code\Macros\Objects.inc
Purpose:    These macros perform object locking for multithreaded access. The threads of a single
            process can use a ObjLock for mutual-exclusion synchronization. There is no guarantee
            about the order that threads obtain ownership of the ObjLock.
Arguments:  Arg1: Object instance. Must be a non volatile register.


Macro:      ObjNamespace
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Set the current object namespace.
Arguments:  Arg1: Space name.


Macro:      $ObjInst
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Return the mangled object name instance.
Arguments:  Arg1: Object Expression     [Namespace:]ObjectName.
Example:    .data
            MyObject $ObjInst(Primer)


Macro:      $ObjMthd
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Return the mangled object method name.
Arguments:  Arg1: Object name.
Example:    .data
            MyObject $ObjMtd(pop)


Macro:      $Obj
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Return the mangled object name.
Arguments:  Arg1: Object Expression    [Namespace:]ObjectName.


Macro:      $ObjPtr
File:       \ObjAsm\Code\Macros\Objects2.inc
```

```
Purpose:    Return the mangled object pointer name.
Arguments:  Arg1: Object Expression: [Namespace:]ObjectName.


Macro:      $ObjTmpl
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Return the mangled object template name.
Arguments:  Arg1: Object Expression: [Namespace:]ObjectName.


Macro:      ExterndefMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Declares an external defined method.
Arguments:  Arg1: Method Expression: ObjectName.MethodName.


Macro:      BuildObjInherPath
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to put in 1 symbol the whole interitance path starting from Primer to
            the final object and viceversa.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled ancestor name.
Return:     TRUE = OK, FALSE = failure.


Macro:      Object
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Start object definition. Here starts the magic
Arguments:  Arg1: Object Expression: Namespace:ObjectName.
            Arg2: Unique number to identificate the object type at run-time.
            Arg3: Ancestor Expression: [Namespace:]AncestorName. (single inheritance).


Macro:      ObjectEnd
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Terminate the declaration of an object and defines the following structures:
            ObjectName: Uninitialized object Template.
            ??ObjectName_VMT: Uninitialized Virtual Method Table (VMT).
            ??ObjectName_IMT: Uninitialized Interface Method Table (IMT).
            ??ObjectName_DMT: Uninitialized VMT + IMT.
            ??ObjectName_Events: Initialized ??EVENT_ENTRY structures list.
            TPL_ObjectName: Initialized ObjectName template.
Arguments:  None.


Macro:      Private
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Methods that follows this directive are defined with private scope.
Arguments:  None.


Macro:      PrivateEnd
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Methods that follows this directive are defined with public scope.
Arguments:  None.


Macro:      InterfaceAbstract
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a placeholder for an interface method that can be overridden.
            Calling an abstract method without overriding it, causes an GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      InterfaceMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      VirtualAbstract
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a placeholder for a virtual method that can be overridden.
            Calling an abstract method without overriding it, causes a GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.
```

```
Macro:      VirtualMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      DynamicAbstract
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a placeholder for a dynamic method that can be overridden.
            Calling an abstract method without overriding it, causes an GPF-Error.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      DynamicMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that is unique for each instance of the object.
            The implementation of the method must be placed in the code Segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      StaticMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that is common to each instance of the object and it is called
            directly, avoiding the indirection over the DMT. This implements early binding.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      InlineMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that is common to each instance of the object.
            The implementation of the method must be placed in the code segment like:
            Method <ObjectName>.<MethodName>, uses..., Argument1:...
            pSelf is the instance POINTER passed to the method to access instance data.
Arguments:  Arg1: Method name.
            Arg2: List of argument types.


Macro:      RedefineMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Redefine a method (static, virtual, dynamic or interface) at compile-time.
Arguments:  Arg1: Method Name to be redefined.
            Arg2: List of new argument types.


Macro:      ObsoleteMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Invalidate a method definition (virtual or dynamic) at compile-time.
Arguments:  Arg1: Method name to be erased.


Macro:      DefineEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to define an event. Don't called directly.
Arguments:  Arg1: Mangled method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      Event
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a method that responds to an event, like a Windows message.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.
Note:       Inline Methods are not allowed.


Macro:      IDispEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
```

```
Purpose:    Define a method that responds to the COM IDispatch.Invoke call.
Arguments:  Arg1: Method name.
            Arg2: DispID.
            Arg3: Flags (DISPATCH_METHOD, DISPATCH_PROPERTYGET, DISPATCH_PROPERTYPUT,
            DISPATCH_PROPERTYPUTREF).


Macro:      StaticEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    This is a shortcut for StaticMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      VirtualEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    This is a shortcut for VirtualMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      InterfaceEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    This is a shortcut for InterfaceMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      DynamicEvent
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    This is a shortcut for DynamicMethod and Event macros.
Arguments:  Arg1: Method name.
            Arg2: EventID list, like WM_CLOSE.


Macro:      DefineVariable
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define a variable (Unique to each instance of the object).
Arguments:  Arg1: Variable name.
            Arg2: Variable type.
            Arg3/4: Optional initial value/structure initial value(s).


Macro:      RedefineVariable
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Redefine a variables initial value.
Arguments:  Arg1: Variable name.
            Arg2: New initial value.


Macro:      LoadObjects
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Load all object structures from a precompiled file (ObjectName.lib).
Arguments:  Arg1: File name list (without extention) that contain the object data to load.


Macro:      MakeObjects
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Load all object data from a file.
Arguments:  Arg1: File name list (without extention) that contain the object data to compile.


Macro:      VirtualObjects
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Create the object data structures to reference an external object instance.
Arguments:  Arg1: File name list (without extention) that contain the object data definitions.


Macro:      PrototypeMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to perform the necessary method definitions for the compiler.
Arguments:  Arg1: Mangled object name
            Arg2: Method name.
            Arg3: List of argument types.


Macro:      CountInterfaceMethods
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that counts all interface methods belonging to an object.
Arguments:  Arg1: Mangled object name.
Return:     Count of Interface methods.
```

```
Macro:      CreateOMList
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create a list of methods to override.
Arguments:  Arg1: Mangled object name.
Return:     Count of methods to override.


Macro:      CreateOVList
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create a list of variables to override.
Arguments:  Arg1: Mangled object name.
Return:     Count of variables to override.


Macro:      CreateInterfaceMethodTable
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create the structure of the interface method table of an object.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) member initialization.


Macro:      CreateVirtualMethodTable
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create the structure of the virtual method table of an object.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) references.
Note:       This VMT is in reverse order to make place for the IMT


Macro:      CreateTemplate
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that defines the following structures:
            ??ObjectName_Init: Initialized template.
            ??ObjectName_DMT_Init: Initialized VMT + IMT.
            Additionally it creates the object data list in the .data segment.
Arguments:  Arg1: Mangled Object name.


Macro:      CreateDynamicTemplate
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create the dynamic section of the object template.
Arguments:  Arg1: Mangled object name.
            Arg2: with (TRUE) or without (FALSE) references.


Macro:      CreateEventList
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to create the event list of an object.
Arguments:  Arg1: Mangled object name.


Macro:      GetMtdType
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Return method type.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
            Retrun: ??MtdType: ??MTDTYPE_UNKNOWN, ..., ??MTDTYPE_DYNAMIC
            ??ImpObj: Mangled object name that implements the method.


Macro:      $IsObsoleteMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Check if a method of an object is obsolete or not.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
            Retrun: FALSE: is NOT an obsolete method, TRUE: it is an obsolete method.


Macro:      $IsPrivateMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Scan recursively in the inheritance path of a specified method to find out
            if it is a private method.
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.
            Retrun: TRUE: it is a virtual method, FALSE: is NOT a virtual method.


Macro:      $MethodAddr
File:       \ObjAsm\Code\Macros\Objects2.inc
```

```
Purpose:    Return a method address.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Notes:      - this macro uses the eax/rax register, except for static methods.
            - if the expression contains an instance POINTER, then the returned address is
            relative to this particular instance, otherwise to the object template.
            - can be used inside nested into OCall/ACall/etc. macros.
            - Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Obj.Mtd
            - Mtd


Macro:      New / $New
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Create a new instance of an object copying the object template.
            It is also suitable for object instances located on the stack using the LOCAL keyword.
Arguments:  Arg1: Expression of the form "Instance::Namespace:Object"
            "Instance" is an optional expression used for preallocated instances like when
            stack is used to hold the object.
            "Object" is a required expression, that identifies the object type to
            instantiate.
Return:     rax → created object instance.
            Possible syntax combinations:
            - Preallocated memory
            - Ins::Nsp:Obj
            - Ins::Obj
            - With memory allocation
            - Nsp:Obj
            - Obj


Macro:      Destroy
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Safely free the memory allocated for a particular object instance but calling first
            its destructor method. If not specified, "Done" is assumed.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Ins.Mtd
            - Ins


Macro:      Kill
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Free the memory allocated for a particular object instance but calling first its
            destructor method. If not specified, "Done" is assumed.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Ins.Mtd
            - Ins


Macro:      SetObject
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Assume a register to an object type.
Arguments:  Arg1: Register, i.e.: rax, rbx, ...
            Arg2: (optional) Object name.
            Arg3: (optional) Instance POINTER. Default is pSelf.


Macro:      SetOwner
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Link a register to an object owner object type and instance at compile-time.
Arguments:  Arg1: Register, i.e.: rax, rbx, ...
            Arg2: Object name.
            Arg3: (optional) Instance POINTER. Default is pSelf.


Macro:      ReleaseObject
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Free the link between a register and an object at compile-time.
Arguments:  Arg1: (Optional) Register, i.e.: rax, rbx, ...


Macro:      Override
File:       \ObjAsm\Code\Macros\Objects2.inc
```

```
Purpose:    Set a new method address in the dynamic-, virtual-, interfacetable.
Arguments:  Arg1: Instance::Object.Method
            Arg2: Object.Method or procedure address.
Note:       Don't use r11 to pass a procedure address It is used internally.


Macro:      ObjectsInit
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to initialize all object templates calling their Startup procedure.
            No instance creation takes place.
Arguments:  None.


Macro:      ObjectsDone
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro to finalize all object templates calling their Shutdown procedure.
Arguments:  None.


Macro:      $MethodPrologue32
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Method prolog macro.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.
Return:     Total local space (byte count).


Macro:      MethodEpilogue32
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that defines the method epilog.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      $NoFramePrologue32
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Method prolog macro.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      NoFrameEpilogue32
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that defines the NOFRAME method epilog.
Arguments:  Arg1: Procedure name
            Arg2: Flags.
            Arg3: Number of parameter bytes.
            Arg4: Number of local bytes.
            Arg5: Uses register list.
            Arg6: Additional macro arguments.


Macro:      Method
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Define an object method.
Arguments:  Arg1: Object method name (ObjectName.MethodName).
            Arg2: Method options like "uses".
            Arg3: Method arguments.


Macro:      MethodEnd
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Terminate an object method.
Arguments:  None.


Macro:      $Method
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Returns the mangled object template name.
```

```
Arguments:  Arg1: ObjectName.MethodName
Example:    $Method(Object.Method) macro pSelf, Arg1, Arg2
            ...
            endm


Macro:      ExitMethod
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Terminate an object method.
Arguments:  Arg1: Expression.
Note:       - More efficient is the use of i.e. "je @@EOM".
            - Attention with the = should be used.


Macro:      Embed
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Embed a named object instance into the template of another object.
Arguments:  Arg1: Instance name.
            Arg2: Object expression (Namespace:ObjectName).
Note:       - Never call Destroy to free an embedded object. Only call "Done".


Macro:      $PushMethodArgs
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that pushes all method arguments.
Arguments:  Arg1: Argument count.
            Arg2: Method argument list.


Macro:      $GetDeclaredParamCount
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that returns the parameter count of a method definition inside "Object".
Arguments:  Arg1: Mangled object name.
            Arg2: Mangled method name.


Macro:      $GetInstance32
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro for 32 bit that returns the instance pointer in a register.
Arguments:  Arg1: Instance name.
            Arg2: Used register.


Macro:      GetInstance64
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro for 64 bit that returns the instance pointer in the rcx register.
Arguments:  Arg1: Instance name.


Macro:      SetVarargPassedParams
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that sets the total parameter count (excluded pSelf) in the eax register.
Arguments:  Arg1: Method arguments.


Macro:      MethodInvoke
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro that calls a method.
Arguments:  Arg1: Invocation flag: TRUE = use the template, FALSE = use object instance.
            Arg2: Instance name.
            Arg3: Mangled object name.
            Arg4: Mangled method name.
            Arg5: Arguments.
Notes:      If a method has VARARGs, the number of arguments is passed in eax.


Macro:      PreParseExpr
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Get the positions of the separators of an expression like:
            "InsName::Namespace:ObjName.MtdName"
            | |       |
            ??Pos1 ??Pos2  ??Pos3
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     ??Pos1, ??Pos2 & ??Pos3.


Macro:      $ParseMtdExpr
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Parse an input expression for address macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
```

```
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Obj.Mtd
            - Mtd


Macro:      $ParseCallExpr
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Parse an input expression for ?Call macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Ins::Nsp:Obj.Mtd
            - Ins::Obj.Mtd
            - Nsp:Obj.Mtd
            - Ins.Mtd
            - Mtd


Macro:      $ParseInstExpr
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Parse an input expression for some Debug macros.
Arguments:  Arg1: Expression of the form: "InsName::NameSpace:ObjName.MtdName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??InsExpr, ??ObjExpr, ??MtdExpr. (??ObjExpr and ??MtdExpr are mangled).
Notes:      Possible syntax combinations:
            - Ins::Nsp:Obj
            - Ins::Obj


Macro:      $ParseObjExpr
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Parse an input expression for Object macro.
Arguments:  Arg1: Expression of the form: "NameSpace:ObjName"
Return:     TRUE if the parsing was successful, otherwise FALSE.
            ??ObjExpr (mangled).
Notes:      The challange from this macro is to avoid symbol expansion. E.g. GetObject is the name
            of a well known api. When we use it as the name of a method, it will get expanded to
            __imp_GetObjectA, which causes problems when parsing the input expression.
            Possible syntax combinations:
            - Nsp:Obj
            - Obj


Macro:      OCall / $OCall
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Call an object method.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      ACall / $ACall
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Call the ancestor method of an object.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      TCall / $TCall
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Call a method using the address stored in the object template.
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.


Macro:      DCall / $DCall
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Call an object method directly like a normal procedure (like a static method).
Arguments:  Arg1: "InsName::ObjName.MtdName" expression. See $ParseCallExpr.
            Arg2: Method arguments.
```

```
Macro:      $GetParamCount
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Helper macro counts parameters in Args list.
Arguments:  Arg1: Method parameter list.


Macro:      $GetImpIfc
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Get the implementing interface name.
Arguments:  Arg1: Interface name.
            Arg2: Method name.


Macro:      ICall / $ICall
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Interface method call.
Arguments:  Arg1: InstancePointer::InterfaceName.MethodName.
            Arg2: Method arguments.


Macro:      AddErrMsg
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    Add an error message and creates the necessary Object Error Tables.
Arguments:  Arg1: Symbol name describing the error, e.g. OBJ_OUT_OF_MEMORY.
            Arg2: [Optional] Error description text.
            Arg3: [Optional] Error ID. Sets the first ID in a series of error messages.


Macro:      LockObjectAccess / UnlockObjectAccess
File:       \ObjAsm\Code\Macros\Objects2.inc
Purpose:    These macros perform object locking for multithreaded access. The threads of a single
            process can use a ObjLock for mutual-exclusion synchronization. There is no guarantee
            about the order that threads obtain ownership of the ObjLock.
Arguments:  Arg1: Object instance. Must be a non volatile register.


Macro:      qMathInit
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Initialize internal trigonometric tables.
Arguments:  Arg1: temporary DWORD value.
Return:     Nothing.


Macro:      qSin
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the sin of a bittian value.
Arguments:  Arg1: Number of terms [1..3] used to compute the taylor series.
            Arg2: Temporary DWORD value.
Return:     sin value in st(0).


Macro:      qCos
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the cos of a bittian value.
Arguments:  Arg1: Number of terms [1..3] used to compute the taylor series.
            Arg2: Temporary DWORD value.
Return:     cos value in st(0).


Macro:      qSinCos
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the sin and cos of a bittian value.
Arguments:  Arg1: Number of terms [1..3] used to compute the taylor series.
            Arg2: Temporary DWORD value.
Return:     sin value in st(1).
            cos value in st(0).


Macro:      qCosSin
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the cos and sin of a bittian value.
Arguments:  Arg1: Number of terms [1..3] used to compute the taylor series.
            Arg2: Temporary DWORD value.
Return:     cos value in st(1).
            sin value in st(0).


Macro:      qRcpSqrt
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the reciprocal sqrt of st(0) value.
```

```
Arguments:  Arg1: Number of iterations [0..3]. -1 means using the fpu equivalent instructions.
            Arg2: temporary DWORD value.
Return:     1/sqrt in st(0).


Macro:      qSqrt
File:       \ObjAsm\Code\Macros\qMath.inc
Purpose:    Compute the sqrt of st(0) value.
Arguments:  Arg1: Number of iterations [0..3]. -1 means using the fpu equivalent instructions.
            Arg2: temporary DWORD value.
Return:     1/sqrt in st(0).


Macro:      $Quad
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Transform 2 DoubleWords into a QuadWord.
Arguments:  Arg1: High order DoubleWord.
            Arg2: Low order DoubleWord.
Return:     QUADWORD in edx::eax.


Macro:      qmov
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Move a QUADWORD.
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source QUADWORD.


Macro:      qdmov
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Move a DWORD to a QUADWORD.
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source dWord.


Macro:      qShiftL
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Shift a edx::eax left cx bits.
Arguments:  Arg1: edx::eax = QWORD
            Arg2: cx = bit shift count


Macro:      qneg
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Negate edx::eax
Arguments:  arg1: edx::eax = QWORD


Macro:      qdadd
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Add a DoubleWord to a QUADWORD (qWrd = qWrd + dWrd).
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source DoubleWord.


Macro:      qqadd
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Add 2 QuadWords (qWrd1 = qWrd1 + qWrd2).
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source QUADWORD.


Macro:      qdsub
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Subtract a DoubleWord from a QUADWORD (qWrd = qWrd - dWrd).
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source DoubleWord.


Macro:      qqsub
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Subtract 2 QuadWords (qWrd1 = qWrd1 - qWrd2).
Arguments:  Arg1: Destination QUADWORD.
            Arg2: Source QUADWORD.


Macro:      qdmul
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Multiply a QUADWORD by a DoubleWord.
Arguments:  ecx::edx::eax = edx::eax * ebx.
```

```
Macro:      qddiv
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Divide a QUADWORD by a DoubleWord. Uses ecx.
            Output: edx::eax = quotient of division of dividend by divisor
            ebx = remainder of division of dividend by divisor
Arguments:  edx::eax = dividend
            ebx = divisor


Macro:      qqdiv
File:       \ObjAsm\Code\Macros\QuadWord.inc
Purpose:    Divide 2 unsigned QuadWords. Uses edi, esi.
            Output: edx::eax = quotient of division of dividend by divisor
            ecx::ebx = remainder of division of dividend by divisor
Arguments:  edx::eax = dividend
            ecx::ebx = divisor


Macro:      SDLL_Init
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Initialize the control structure.
Arguments:  Arg1: Register → Sentinel structure.


Macro:      SDLL_InsertAfter
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Inserts an object after another in the linked list.
Arguments:  Arg1: Register → Member to insert after.
            Arg2: Register → Member to insert.
            Arg3: Auxiliar register. If not specified, RefReg is used.


Macro:      SDLL_InsertBefore
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Insert an object before another in the linked list.
Arguments:  Arg1: Register → Member to insert before.
            Arg2: Register → Member to insert.
            Arg3: Auxiliar register. If not specified, RefReg is used.


Macro:      SDLL_Remove
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Remove a member from the linked list.
Arguments:  Arg1: Register → Member to remove register or reference.
            Arg2: Auxiliar register 1.
            Arg3: Auxiliar register 2. If not specified, RefReg is used.


Macro:      SDLL_RemoveFirst
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Remove the first member from the linked list.
Arguments:  Arg1: Register → Sentinel structure.
            Arg2: Auxiliar register.


Macro:      SDLL_RemoveLast
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Remove the first member from the linked list.
Arguments:  Arg1: Register → Sentinel structure.
            Arg2: Auxiliar register 1.


Macro:      SDLL_GetFirst
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Return the first item in the list.
Arguments:  Arg1: Register → Sentinel structure.
            Arg2: Return register.
Note:       If SnlReg == [SnlReg].SDLL_ITEM.pNextItem, then the list is empty
Example:    SDLL_GetFirst ecx, eax; jz ...


Macro:      SDLL_GetLast
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Return the last item in the list.
Arguments:  Arg1: Register → Sentinel structure.
            Arg2: Return register.
Note:       If SnlReg == [SnlReg].SDLL_ITEM.pPrevItem, then the list is empty
Example:    SDLL_GetLast ecx, eax; jz ...
```

```
Macro:      SDLL_GetNext
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Return the next item in the list.
Arguments:  Arg1: Register → Reference item.
            Arg2: Register → Sentinel structure.
            Arg3: Return register. If not specified, the RefReg is used as return register.
Note:       If SnlReg == [RefReg].SDLL_ITEM.pNextItem, then the list is empty
Example:    SDLL_GetNext ecx, eax; jz ...


Macro:      SDLL_GetPrev
File:       \ObjAsm\Code\Macros\SDLL.inc
Purpose:    Return the previous item in the list.
Arguments:  Arg1: Register → Reference item.
            Arg2: Register → Sentinel structure.
            Arg3: Return register. If not specified, the RefReg is used as return register.
Note:       If SnlReg == [RefReg].SDLL_ITEM.pNextItem, then the list is empty
Example:    SDLL_GetNext ecx, eax; jz ...


Macro:      Vec4_SquareMagnitude_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Calculate the squared magnitude of a VEC4.
Arguments:  Arg1: xmm register holding the VEC4.
            Arg2: xmm auxiliary register.
Notes:      Output = Arg1.


Macro:      Vec4_Magnitude_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Calculate the magnitude of a VEC4.
Arguments:  Arg1: xmm register holding the VEC4.
            Arg2: xmm auxiliary register.
Notes:      Output = Arg1.


Macro:      Vec4_Normalize_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Normalize a VEC4.
Arguments:  Arg1: xmm register holding the VEC4.
            Arg2..3: xmm auxiliary registers.
Notes:      Output = Arg1.
Link:       http://www.3dbuzz.com/vbforum/showthread.php?104753-HowTo-Inline-Assembly-amp-SSE-
            Vector-normalization-done-fast


Macro:      Vec4_CrossProduct_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Calculate the cross product of 2 VEC4.
Arguments:  Arg1: xmm register holding the first VEC4.
            Arg2: xmm register holding the second VEC4.
            Arg3..4: xmm auxiliary registers.
Link:       http://neilkemp.us/src/sse_tutorial/sse_tutorial.html
Notes:      Output = Arg1.


Macro:      Vec4_DotProduct_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Calculate the dot product of 2 VEC4.
Arguments:  Arg1: xmm register holding the first VEC4.
            Arg2: xmm register holding the second VEC4.
            Arg3: xmm auxiliary register.
Notes:      Output = Arg1.


Macro:      Mat44_Load_A_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Load a matrix from aligned memory to xmm registers.
Arguments:  Arg1: → MAT44.
            Arg2..5: xmm registers that will recieve the MAT44 rows.


Macro:      Mat44_Load_U_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Load a matrix from unaligned memory to xmm registers.
Arguments:  Arg1: → MAT44.
            Arg2..5: xmm registers that will recieve the MAT44 rows.


Macro:      Mat44_Store_A_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
```

```
Purpose:    Store a matrix from xmm registers to aligned memory.
Arguments:  Arg1: → MAT44.
            Arg2..5: xmm registers that will recieve the MAT44 rows.


Macro:      Mat44_Store_U_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Store a matrix from xmm registers to unaligned memory.
Arguments:  Arg1: → MAT44.
            Arg2..5: xmm registers that will recieve the MAT44 rows.


Macro:      Mat44_Transpose_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Transpose the MAT44 loaded in xmm registers.
Arguments:  Arg1..4: xmm registers holding the MAT44 columns.
            Arg5: auxiliary xmm register.
Notes:      Output = Arg1..4 xmm registers holding the MAT44 rows.


Macro:      Mat44_Transpose_SSE_2
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Transpose the MAT44 loaded in xmm registers.
            This code is somewhat slower as Mat44_Transpose_SSE but can be compiled with ML6.15
Arguments:  Arg1..4: xmm registers holding the MAT44 columns.
            Arg5..6: auxiliary xmm registers.
Notes:      Output = Arg1..4 xmm registers holding the MAT44 rows.


Macro:      Mat44_Mult_Vec4_SSE
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Mutiplication of MAT44 by VEC4
Arguments:  Arg1..4: xmm registers holding the MAT44 columns.
            Arg5: xmm register holding the input VEC4.
            Arg6..8: auxiliary xmm registers.
Notes:      Output = Arg5.


Macro:      Real8ToXmm
File:       \ObjAsm\Code\Macros\SIMD_Math.inc
Purpose:    Moves Real8 values to an xmm register
Arguments:  Arg1: Real8 value
            Arg2: Real8 value
Example:    movaps xmm15, XMMWORD ptr Real8ToXmm(1.0, -1.0)


Macro:      $Esc
File:       \ObjAsm\Code\Macros\Strings.inc
Purpose:    Convert escape sequences to the corresponding character codes.
Arguments:  Arg1: Text.
Return:     Processed text.
Notes:      esc. seq.      code           symbol
            ------------------------------------------------
            \: 21h           ''
            \{ 28h           '('
            \} 29h           ')'
            \[ 3Ch           '<'
            \] 3Eh           '>'
            \= 22h           '"'
            \, 3Bh           ';'
            \0 0            zero character
            \n 0Dh, 0Ah      new line
            \r 0Dh           carrige return
            \l 0Ah           line feed
            \t 09h           horizontal tabulation

            Other combinations of the "\" character are treated as literals.


Macro:      $RepChr
File:       \ObjAsm\Code\Macros\Strings.inc
Purpose:    Repeat a character N times.
Arguments:  Arg1: Character to repeat
            Arg2: Character count.
Return:     Quoted text.
Example:    CStrW wBlanks, $RepChr(< >, 15)


Macro:      $Ofs(C/T/D/J)Str
File:       \ObjAsm\Code\Macros\Strings.inc
Purpose:    Place an ANSI or UNICODE string in the CONST, _DATA or _TEXT Segment.
```

```
Arguments:  Arg1: Quoted string text.
Return:     String offset.


Macro:      $PCG_RndChr
File:       \ObjAsm\Code\Macros\Strings.inc
Purpose:    Return at assembly-time a randomly generated (PCG) character value.
Arguments:  None.
Return:     Character value.


Macro:      $PCG_RndStr
File:       \ObjAsm\Code\Macros\Strings.inc
Purpose:    Return at assembly-time a randomly generated (PCG) string in const segment.
            On each run of the macro, a new string is generated
Arguments:  Arg1: String length in chars.
Return:     String location.
Example:    DbgWriteF ,, "¦ST", offset $PCG_RndStr(64)


Macro:      $ArgRev
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a reversed order version of a symbol list.
Arguments:  Arg1: Symbol list.
Return:     Reversed symbol list.


Macro:      PushAll
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Push recursively all arguments on stack.
Arguments:  Arg1: List of arguments.
Return:     Nothing.


Macro:      PopAll
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Pop a list of arguments of the stack.
Arguments:  Arg1: List of arguments.
Return:     Nothing.


Macro:      PushAllRev
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Push all arguments on stack in reverse order.
Arguments:  Arg1: List of arguments.
Return:     Nothing.


Macro:      PopAllRev
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Pop recursively all arguments on stack in reverse order.
Arguments:  Arg1: List of arguments.
Return:     Nothing.


Macro:      c2m
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Move a constant value to a QWORD, DWORD, WORD or BYTE memory address.
Arguments:  Arg1: destination memory symbol.
            Arg2: source constant value.
            Arg3: register used to move the memory content. If not specified, r13 is used.
Return:     Nothing.


Macro:      m2m
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Move a QWORD, DWORD, WORD or BYTE value from a memory address to another.
            Auxiliary register may not be used.
Arguments:  Arg1: Destination memory symbol.
            Arg2: Source memory symbol.
            Arg3: Auxiliary register used to move the memory content.
            If not specified, the stack may be used.
Return:     Nothing.


Macro:      mrm
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Move a QWORD, DWORD, WORD or BYTE value from a memory address to another using
            always the auxiliary register.
Arguments:  Arg1: destination memory symbol.
            Arg2: source memory symbol.
```

```
            Arg3: register used to move the memory content.
Return:     Nothing.


Macro:      m2z
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Zero the content of a memory location.
Arguments:  Arg1: Memory location.
Return:     Nothing.
Note:       On older CPUs, like a PIII, "and DstMem, 0" is faster than "mov DstMem, 0".


Macro:      s2s
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Clone a structure to another structure of the same type.
Arguments:  Arg1: Destination structure.
            Arg2: Source structure. Structure size must match the destination structure
            Arg3: Available registers for the copy operation.
Return:     Nothing.
Note:       Use only AVX istructions to avoid AVX/SSE transition penalties.
            Try using vmovdqu later.


Macro:      JumpOn
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Create a jump table and executes a jump to a label according the content of a
            register.
Arguments:  Arg1: Case register.
            Arg2: Jump labels.
Return:     Nothing.
Example:    JumpOn eax, @@10, @@20, @@30


Macro:      FillStringA
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill an ANSI string with text using a series of mov DWORD/WORD/BYTE instructions.
            It doesn't work with QWORDs.
Arguments:  Arg1: String to be filled.
            Arg2: Text.
Return:     Nothing.
Example:    FillStringA myString, <Hello>


Macro:      FillStringB
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill a BSTR string with text using a series of mov DWORD/WORD/BYTE instructions.
            It doesn't work with QWORDs.
Arguments:  Arg1: String to be filled.
            Arg2: Text.
Return:     Nothing.
Example:    FillStringB myString, <Hello>


Macro:      FillStringW
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill a UNICODE string with text using a series of mov DWORD/WORD/BYTE
            instructions.
Arguments:  Arg1: String to be filled.
            Arg2: Text.
Return:     Nothing.
Example:    FillStringW myString, <Hello>


Macro:      FillWordA
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill with a sequence of ANSI characters using a series of mov DWORD/WORD/BYTE.
            It doesn't work with QWORDs.
Arguments:  Arg1: String to be filled.
            Arg2: Text.
Return:     Nothing.
Example:    FillWordA myBuffer, <Hello>


Macro:      FillWordW
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill with a sequence of WIDE characters using a series of mov DWORD/WORD.
            It doesn't work with QWORDs.
Arguments:  Arg1: String to be filled.
            Arg2: Text.
Return:     Nothing.
Example:    FillWordW myBuffer, <Hello>
```

```
Macro:      DoesWordMatchA?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare a sequence of ANSI characters with a text using a series of
            cmp DWORD/WORD/BYTE instructions.
Arguments:  Arg1: Location containing the word to be compared.
            Arg2: Text.
Return:     In case that NoMatchLabel is not specified use the flags.
            ZERO? indicates that the word matches.
Example:    DoesWordMatchA? myWord, <Hello>
            DoesWordMatchA? myWord, <  >


Macro:      DoesWordMatchW?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare a sequence of WIDE characters with a text using a series of
            cmp DWORD/WORD instructions.
Arguments:  Arg1: Location containing the word to be compared.
            Arg2: Text.
Return:     In case that NoMatchLabel is not specified use the flags.
            ZERO? indicates that the word matches.
Example:    DoesWordMatchW? myWord, <Hello>
            DoesWordMatchW? myWord, <  >


Macro:      DoesStringMatchA?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare a string of ANSI characters with a text using a series of
            cmp DWORD/WORD/BYTE instructions. The ZTC is included in the comparison.
Arguments:  Arg1: Location containing the string to be compared.
            Arg2: Text.
Return:     In case that NoMatchLabel is not specified use the flags.
            Zero indicates that the word matches.
Example:    DoesStringMatchA? myString, <Hello>


Macro:      DoesStringMatchW?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare a string of WIDE characters with a text using a series of
            cmp DWORD/WORD instructions. The ZTC is included in the comparison.
Arguments:  Arg1: Location containing the string to be compared.
            Arg2: Text.
Return:     In case that NoMatchLabel is not specified use the flags.
            Zero indicates that the word matches.
Example:    DoesStringMatchW? myString, <Hello>


Macro:      WriteF
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Write a formated string to memory. If format includes format specifiers (subsequences
            beginning with ¦), the additional arguments following format are formatted and
            inserted
            in the resulting string replacing their respective specifiers.
            Escape sequences are supported to overcome MASM limitations. Escape sequences begin
            with a \ character.
Arguments:  Arg1: Non-volatile register pointing to the beginning of the output buffer.
            It is used internally and its value is modified upon return.
            Arg2: Single or double Quoted format string. Only ANSI characters are allowed.
            format specifiers start with the "¦" character, followed by 2 characters
            specifying the type (and the decimal places for floating-point numbers) of the
            subsequent arguments.
            Arg3-N: Additional arguments. Must be non-volatile registers, FPU registers or
            memory symbols.
Notes:      - Format specifiers:
            SB signed BYTE as decimal
            SW signed WORD as decimal
            SD signed DWORD as decimal
            SQ signed QWORD as decimal
            SX signed DWORD/QWORD as decimal
            UB unsigned BYTE as decimal
            UW unsigned WORD as decimal
            UD unsigned DWORD as decimal
            UQ unsigned QWORD as decimal
            UX unsigned DWORD/QWORD as decimal
            Fn Floating Point with n decimals, regular notation. n ranges from 0 to F
            En Floating Point with n decimals, scientific notation. n ranges from 0 to F
            H4 DWORD as hexadecimal
            H8 QWORD as hexadecimal
            HX DWORD/QWORD as hexadecimal
            SA ANSI string
```

```
            SW WIDE string
            ST ANSI/WIDE string
            GD GUID
            WE Windows API Error as description string
            CE COM Error as description string
            UE UEFI Error as description string
            MT Move to character position
            AT fill with spaces up to character position

            - Escape sequences:
            n Carriage Return and Line Feed
            r Carriage Return
            l Line Feed
            0 ZTC
            t Horizontal Tab
            [ "<" character
            ] ">" character
            : "" character
            ' "'" character
            \ "\" character

            - Format specifiers and escape sequences are case sensitive.

Example:    WriteF xdi, 'Memory at ¦H8h: ¦F3", rsi, REAL4 ptr [xsi]


Macro:      IsCharTypeA?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare an ANSI character with a pedefined type of characters.
Arguments:  Arg1: Character to be evaluated.
            Arg2: Character type.
            Arg3: (optional) Character table
Return:     Flags. Zero indicates that the character is NOT of the char type.
            On return, al contains the tested character.
Example:    IsCharTypeA? [myString+5], CharTypeText
            .if $IsCharTypeA?([myString+5], CharTypeText)


Macro:      ($)Choose
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Choose returns a value from a list of choices based on an index value.
Arguments:  Arg1: Index.
            Arg2: First choise.
            Arg3: Rest of choices.
Return:     Choosen value.


Macro:      (C/D)Real$$
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Declare a named floating point variable in the data segment.
Arguments:  Arg1: Name of the floating point variable.
            Arg2: Value of the floating point variable. May be ?.


Macro:      $CReal??
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Create in memory a REAL4 constant. Repeated declarations use the same memory location.
Arguments:  Arg1: Value of the floating point variable.
Return:     Symbol of the declared REAL4 value.
Note:       The $CReal?? macro are intended to avoid the creation of multiple instances of the
            same
            floating point constant.


Macro:      $DReal??
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Declare a floating point variable as a vaiable.
Arguments:  Arg1: Value of the floating point variable.
Return:     Symbol of the floting point value.


Macro:      sMax / $sMax
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the maximum of 2 signed values.
Arguments:  Arg1: First signed word.
            Arg2: Second signed word.
            Arg3: Optional destiantion register. Default is xax.
Return:     DstReg/rax/eax/ax/al = Biggest signed value.
```

```
Macro:      sMin / $sMin
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the minimum of 2 signed values.
Arguments:  Arg1: First signed word.
            Arg2: Second signed word.
            Arg3: Optional destiantion register. Default is xax.
Return:     DstReg/rax/eax/ax/al = Smallest signed value.


Macro:      uMax / $uMax
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the maximum of 2 values.
Arguments:  Arg1: First word.
            Arg2: Second word.
            Arg3: Optional destiantion register. Default is xax.
Return:     DstReg/rax/eax/ax/al = Biggest value.


Macro:      uMin / $uMin
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the minimum of 2 values.
Arguments:  Arg1: First value.
            Arg2: Second value.
            Arg3: Optional destiantion register. Default is xax.
Return:     DstReg/rax/eax/ax/al = Smallest value.


Macro:      $uMini / uMini
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the smallest of 2 values.
Arguments:  Arg1: Default rax = a, rcx = b
Return:     Reg1/rax/eax/ax/al = $uMini(a,b)
            Uses: Default are rax, rcx, rdx (Reg2 and AuxReg are trashed).


Macro:      $uMaxi / uMaxi
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the maximum of 2 values.
Arguments:  Arg1: Default rax = a, rcx = b
Return:     Reg1/rax/eax/ax/al = $uMaxi(a,b)
            Uses: Default are rax, rcx, rdx (Reg2 and AuxReg are trashed).


Macro:      $sMaxi / sMaxi
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the maximum of 2 signed values.
Arguments:  rax = a, rdx = b
Return:     rax/eax/ax/al = min(a,b), rdx/edx/dx/dl = max(a,b)
            Uses: rax, rdx.


Macro:      uMiniMaxi
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the minimum and maximun of 2 values.
Arguments:  Reg1, Reg2
Return:     Reg1 = min(Reg1, Reg2), Reg2 = max(Reg1, Reg2)
            Uses: Reg1, Reg2, AuxReg1, AuxReg2


Macro:      sMean / $sMean
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute (Reg1 + Reg2)/2 (signed values).
Arguments:  Arg1: Reg1.
            Arg2: Reg2.
            Arg3: AuxReg.
Return:     rax/eax/ax/al = (Reg1 + Reg2)/2.
            Uses: Default rax, rcx, rdx (AuxReg is trashed).
Note:       This algorithm can not overflow


Macro:      LoadRegIfLessThan
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compare Reg with a value and load it according to the result.
Arguments:  Arg1: Register to be comparend.
            Arg2: Than value or register.
            Arg3: Then value or register.
            Arg4: Otherwise value or register.
            Arg5: [optional] auxiliar register. Default edx.
Return:     Register.
```

```
Macro:      HiWord / $HiWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Retrieve the high WORD from a DWORD.
Arguments:  Arg1: DWORD.
Return:     eax = High WORD.


Macro:      LoWord / $LoWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Retrieve the low WORD from a DWORD.
Arguments:  Arg1: DWORD.
Return:     eax = Low WORD.


Macro:      $addr
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Get the runtime address of the operand. Used in those cases where "addr" is not
            allowed.
Arguments:  Arg1: Operand.
Return:     Memory address of operand.


Macro:      PushArgsFor
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Push all arguments on stack in reverse order.
Arguments:  Arg1: API name.
            Arg2: List of arguments.
Return:     Nothing.


Macro:      $RGB
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Calculate a RGB value of the specified R/G/B/Alpha arguments.
Arguments:  Arg1: Red component.
            Arg2: Green component.
            Arg3: Blue component.
            Arg4: Alpha component (if any).
Return:     RGB DWORD value (CRGB).


Macro:      $BGR
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Calculate a RGB value of the specified B/G/R/Alpha arguments.
Arguments:  Arg1: Blue component.
            Arg2: Green component.
            Arg3: Red component.
            Arg4: Alpha component (if any).
Return:     RGB DWORD value (RGBQUAD).


Macro:      RGB2BGR
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Convert the color from RGB to BGR as it is usually stored in memory.
Arguments:  Arg1: 32 bit register containeing the RGB/BRG color.


Macro:      $Lower
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Convert a text symbol to lowercase.
Arguments:  Arg1: String symbol.


Macro:      $Upper
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Convert a text symbol to uppercase.
Arguments:  Arg1: Text symbol.


Macro:      $IsFloat
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Check if a passed text is a floating point value.
Arguments:  Arg1: Text representing a float.
Link:       http://masm32.com/board/index.php?topic=4709.15
Return:     TRUE for any REAL4/8/10 variable or decimal FP literal.
            Autor: qWord


Macro:      $invoke
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Accelerator macro that calls a procedure and returns its return value in eax.
```

```
Arguments:  Arg1: Procedure name to be called.
            Arg2: Procedure arguments.
Return:     Return value of the procedure in eax.


Macro:      $call
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Call invocation that returns eax.
Arguments:  Arg1: Procedure name.
Return:     Nothing.


Macro:      $MakeWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute a WORD value.
Arguments:  Arg1: High order BYTE.
            Arg2: Low order BYTE.
Return:     DWORD value.


Macro:      $MakeDWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute a DWORD value.
Arguments:  Arg1: High order WORD.
            Arg2: Low order WORD.
Return:     DWORD value.


Macro:      IsPositive
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return TRUE if the signed integer argument is >= 0, otherwise FALSE.
Arguments:  Arg1: SDWORD
Return:     TRUE or FALSE.


Macro:      IsNegative
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return TRUE if the signed integer argument is < 0, otherwise FALSE.
Arguments:  Arg1: SDWORD
Return:     TRUE or FALSE.


Macro:      MSB32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute the most significant "1" bit in a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.


Macro:      LSB32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute the least significant "1" bit in a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.


Macro:      NLPo2
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compute the Next Largest Power of 2 of a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.


Macro:      PopulationCount32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Count the number of "1"s in a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.


Macro:      TZC32 Trailing Zero Count
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Count the number of trailing "0"s in a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.
```

```
Macro:      ReverseBits32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Reverse the bits in a DWORD.
Arguments:  Arg1: Argument value in a register.
            Arg2: (optional) Auxiliar register.
Return:     Result in input register.


Macro:      ReverseBitsShort32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Reverse the bits in a DWORD.
Arguments:  Arg1: Input register value.
            Arg2: Output regiter value.
Return:     Result in input register.
Note:       Uses ecx.


Macro:      IsInRange?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Check if a specified value is in the range [Base...Top].
Arguments:  Arg1: Value (register)
            Arg2: Bottom.
            Arg3: Top.
            Arg4: (optional) Auxiliar register.
Return:     Value (Arg) is 0 if it is in range, otherwise -1.
Note:       Add an inc to Arg to the end of the macro to return TRUE or FALSE.


Macro:      IsNotBetween?
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Check if a specified value is in between the range [Bot...Top], excluding the limits.
Arguments:  Arg1: Value (register)
            Arg2: Bottom
            Arg3: Top
            Arg4: (optional) Auxiliar register
Return:     Value (Arg) is 0 if it is in range, otherwise -1.
Note:       Add an inc to Arg to the end of the macro to return TRUE or FALSE.


Macro:      ClearLocals
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Clear all locals.
Note:       Source of Edgar Harris (Donkey).


Macro:      ht
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Insert a hint BYTE (hint taken) for conditional jumps.
Note:       Must be inserted immediately before the conditional jump.
            The rules for static prediction are:
            - A forward branch defaults to not taken.
            - A backward branch defaults to taken.
            Once the processor has enough information, dynamic rules take over.


Macro:      hnt
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Insert a hint BYTE (hint not taken) for conditional jumps.
Note:       Must be inserted immediately before the conditional jump.


Macro:      _Pause_
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Emulate the pause instruction.
Note:       http://siyobik.info/index.php?module=x86&id=232


Macro:      FillMemZero
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill a memory range with zeros.
Arguments:  Arg1: Mem to be filled.
            Arg2: Mem size.
Return:     Nothing.
Example:    FillMemZero myString, 5


Macro:      FillMemByte
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill a memory range with a BYTE value.
Arguments:  Arg1: Mem to be filled.
```

```
            Arg2: Mem size.
            Arg3: BYTE value.
Return:     Nothing.


Macro:      FillMemWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Fill a memory range with a WORD value.
Arguments:  Arg1: Mem to be filled.
            Arg2: Mem size.
            Arg3: WORD value.
Return:     Nothing.


Macro:      SaveFpuContext
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Save the FPU context that can be restored using LoadFpuContext. FPU is reinitialized.
Arguments:  None.
Return:     Nothing.


Macro:      LoadFpuContext
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Reload the FPU context that was stored using SaveFpuContext.
Arguments:  None.
Return:     Nothing.


Macro:      $SubWord
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a subword according to the requested size.
Arguments:  Arg1: Word type (signed or unsigned).
            Arg2: Required size [1, 2, 4, 8].
Return:     Word.


Macro:      $SubReg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a subregister according to the requested size.
Arguments:  Arg1: Register name.
            Arg2: Required register size [1, 2, 4, 8] in bytes.
Return:     Register.


Macro:      $SubRegMM
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a mm subregister according to the requested size.
Arguments:  Arg1: Register name.
            Arg2: Required register size [16, 32, 64] (xmm, ymm, zmm) in bytes.
Return:     Register.


Macro:      $ChrReg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a subregister according to the CHR size.
Arguments:  Arg1: Register name.
Return:     Register.


Macro:      $64, $32, $16, $8
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return a subregister according to the destination size.
Arguments:  Arg1: Register name.
Return:     Register.


Macro:      GetInterruptTicks
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the interrupt tick count since system start.
            Interrupt ticks are triggered each 100 ns The count is monotone.
            The count is gattered from a system shared memory area known as KUSER_SHARED_DATA,
            which is always at the same location (7FFE0000h) on Windows systems since NT.
Arguments:  Arg1: (optional) Destination memory where the count should be stored (QWORD).
Return:     rax = tick count.


Macro:      $Log2
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Return the Log2 of a value at compile time.
Arguments:  Arg1: Input value.
Return:     Log value.
```

```
Macro:      StackAlign
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Align down the stack to a specified boundary.
Arguments:  Boundary in bytes.
Note:       Uses an auxiliary register, default is edx.


Macro:      StackRestore
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Restore the stack to the previous boundary.
Arguments:  None.


Macro:      $CurSeg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Returns the simplified current segment name.
Arguments:  None.


Macro:      EchoOpAttr
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Echo OpAttr value.
Arguments:  None.


Macro:      ANNOTATION
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Pass a hint to external tools.
Arguments:  Hints, separated by commas.
            Examples: ANNOTATION prv:rdi rsi
            ANNOTATION use:Point Arg1 xdi


Macro:      $ToStr
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Used to echo a numeric symbol.
Arguments:  Symbol name.
            Use: %echo This is a test using $ToStr(%Number1)


Macro:      ?mov
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compile mov instruction if TARGET_BITNESS = 32.
Arguments:  Arg1: First mov.
            Arg2: Second mov.


Macro:      ??mov
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Compile mov instruction if TARGET_BITNESS = 64.
Arguments:  Arg1: First mov.
            Arg2: Second mov.


Macro:      LocReg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Declares the use of a symbol as a register or as a local.
Arguments:  Expression of the form SymbolName1:Register1, SymbolName2:Register2...


Macro:      $LocReg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Replaces the Expression with the corresponding local or register.
Arguments:  SymbolName declared with LocReg.


Macro:      $LocReg32
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Replaces the Expression with the corresponding local or 32 bit register.
Arguments:  SymbolName declared with LocReg.


Macro:      ArgReg
File:       \ObjAsm\Code\Macros\System.inc
Purpose:    Declares the use of a symbol as a register or as an argument.
Arguments:  Expression of the form SymbolName1:Register1, SymbolName2:Register2...


Macro:      $ArgReg
```

File: \ObjAsm\Code\Macros\System.inc
Purpose: Replaces the Expression with the corresponding argument or register.
Arguments: SymbolName.

Macro: $ArgReg32
File: \ObjAsm\Code\Macros\System.inc
Purpose: Replaces the Expression with the corresponding argument or 32 bit register.
Arguments: SymbolName.

Macro: pushfx
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile pushfd or pushfq according to TARGET_BITNESS.
Arguments: None.

Macro: popfx
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile popfd or popfq according to TARGET_BITNESS.
Arguments: None.

Macro: pushaq
File: \ObjAsm\Code\Macros\System.inc
Purpose: Simil to pushad but for 64 bit. Order of pushed values equal to pushad.
Arguments: None.
Note: rsp is not the value of the instruction begin, like it is using pushad.

Macro: popaq
File: \ObjAsm\Code\Macros\System.inc
Purpose: Simil to popaq but for 64 bit.
Arguments: None.

Macro: pushax
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile pushad or pushaq according to TARGET_BITNESS.
Arguments: None.

Macro: popax
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile popad or popaq according to TARGET_BITNESS.
Arguments: None.

Macro: stosx
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile stosd or stosq according to TARGET_BITNESS.
Arguments: None.

Macro: stosc
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile stosb or stosw according to TARGET_STR_TYPE.
Arguments: None.

Macro: movsc
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile movsb or movsw according to TARGET_STR_TYPE.
Arguments: None.

Macro: scasc
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile scasb or scasw according to TARGET_STR_TYPE.
Arguments: None.

Macro: @Random
File: \ObjAsm\Code\Macros\System.inc
Purpose: Compile-time random number generator.
Arguments: Arg1: Range.
Return: Integer [0..Range-1]

Macro: @Err
File: \ObjAsm\Code\Macros\System.inc

```
Purpose:    Compile-time error generation.
Arguments:  Arg1: Message
Return:     Nothing.


Macro:      DispatchEvent
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Search for a specified EventID in the "Event Translation Table". If found,
            the matching method is called, otherwise the default proc is invoked.
Arguments:  Default procedure name and common parameters.
Return:     Nothing.
Note:       - uses xsi → object instance.
            - the event handler receives the event ID (WM_XXX, ...) in eax.


Macro:      Subclass
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Subclass a system window.
Arguments:  Arg1: Object name.
            Arg2: Optional index to allow multiple subclassings.
Return:     Nothing.
Note:       xsi has to point to the object instance


Macro:      Unsubclass
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Unsubclass a system window.
Arguments:  Arg1: Object name.
            Arg2: Optional index to allow multiple subclassings.
Return:     Nothing.
Note:       xsi has to point to the object instance


Macro:      ($)GetSubclassingInst
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Return the instance POINTER of the object that subclassed a system window.
Arguments:  Arg1: Object name.
            Arg2: Window HANDLE.
            Arg2: Optional index to allow multiple subclassings.
Return:     rax → Instance.
Note:       xsi has to point to the object instance


Macro:      CloneRect
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Copy the coordinates from the source to the destination RECT.
Arguments:  Arg1: Destination RECT structure.
            Arg2: Source RECT structure.
            Arg3: optional registers to be used to copy the RECT content.
Return:     Nothing.


Macro:      GrowRect
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Change the size of a RECT by a specified value.
Arguments:  Arg1: RECT structure.
            Arg2: X size to grow.
            Arg2: (optional) Y size to grow.
Return:     Nothing.


Macro:      MoveRect
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Move a RECT by a specified X/Y value.
Arguments:  Arg1: → RECT structure.
            Arg2: X size to move.
            Arg3: (optional) Y size to move.
Return:     Nothing.


Macro:      $MakePointS
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Return a POINTS structure (DWORD) containing the coordinates indicated in args.
Arguments:  Arg1: X position.
            Arg2: Y position.
Return:     POINT structure.


Macro:      EnableDlgControl
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Enable or disable a dialog control.
```

```
Arguments:  Arg1: Dialog HANDLE.
            Arg2: Control ID.
            Arg3: Action (TRUE=Enable, FALSE=Disable).
Return:     Nothing.


Macro:      EnableDlgCtrlNotif
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Enable the parent notification machanism of a dialog control, which was disabled
            by default.
Arguments:  Arg1: Dialog HANDLE.
            Arg2: Control ID.


Macro:      PntS2Pnt
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Convert the POINTS contained in PntS into a POINT contained in Pnt.
Arguments:  POINT, POINTS
Return:     Nothing.


Macro:      PntS2Regs
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Convert the POINTS contained in PntS to DWORD values in the eax and ecx registers.
Arguments:  POINTS, Reg32
Return:     eax = x, Reg32 = y.


Macro:      Pnt2Reg32
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Convert a POINT to a Reg32.
Arguments:  Arg1: Reg32.
            Arg2: POINT.


Macro:      $MakeLangID
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Create a resource language ID.
Arguments:  Arg1: Primary language.
            Arg2: Sublanguage.
Return:     Language ID.


Macro:      $PrimaryLangID
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Return the primary language from a Language ID.
Arguments:  Language ID.
Return:     Primary Language ID.


Macro:      $SubLangID
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Return the sublanguage from a Language ID.
Arguments:  Language ID.
Return:     Sublanguage ID.


Macro:      $Dlu2PixX
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Converts X Dialog Logic Units to pixel.
Arguments:  Arg1: X DLU value.
            Arg2: X DBU value obtained from GetDlgBaseUnits.
Return:     eax = Pixel value.
Note:       Uses xdx.


Macro:      $Dlu2PixY
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Converts Y Dialog Logic Units to pixel.
Arguments:  Arg1: Y DLU value.
            Arg2: Y DBU value obtained from GetDlgBaseUnits.
Return:     eax = Pixel value.
Note:       Uses xdx.


Macro:      SetWndStyle
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Set the style flag of a window.
Arguments:  Arg1: Style flag.
Note:       Uses volatile registers.
```

```
Macro:      ClrWndStyle
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Clear the style flag of a window.
Arguments:  Arg1: Style flag.
Note:       Uses volatile registers.


Macro:      SetWndStyleEx
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Set the extended style flag of a window.
Arguments:  Arg1: Extended style flag.
Note:       Uses volatile registers.


Macro:      ClrWndStyleEx
File:       \ObjAsm\Code\Macros\WinHelpers.inc
Purpose:    Clear the extended style flag of a window.
Arguments:  Arg1: Extenden style flag.
Note:       Uses volatile registers.


Macro:      StringW / ($)CStrW / ($)TStrW / ($)DStrW / ($)JStrW
File:       \ObjAsm\Code\Macros\WStrings.inc
Purpose:    Place an WIDE string in the .const, .text, .data or .code segment.
Arguments:  Arg1: Reference name (optional).
            Arg2: Quoted string text.
Return:     Nothing / Reference to the string.
Notes:      - Quotation marks can be used as usual. See example.
            - Partial input strings can be separated by commas.
            - Break input lines with "\".
            - Empty input strings ("" or '') causes an error.
            - Numeric inputs in word range are possible.
            - sizeof and length of directives work with this macro.
Example:    CStrW MyStrW, 'Note: "', "Director's cut", '"', 13, 10
            Resulting WIDE string:  Note: "Director's cut" + CRLF


Macro:      $Ofs(C/T/D/J)WStr
File:       \ObjAsm\Code\Macros\WStrings.inc
Purpose:    Place an WIDE string in the S_CONST, S_TEXT, S_DATA segment.
Arguments:  Arg1: Quoted string text.
Return:     String offset.


Macro:      IsSurrogateHigh
File:       \ObjAsm\Code\Macros\WStrings.inc
Purpose:    Check if the WIDE char is a high surrogate.
Arguments:  Arg1: Wide character.
Return:     ZERO? if character is a high surrogate.


Macro:      IsSurrogateLow
File:       \ObjAsm\Code\Macros\WStrings.inc
Purpose:    Check if the WIDE char is a low surrogate.
Arguments:  Arg1: Wide character.
Return:     ZERO? if character is a low surrogate.


Macro:      xLSLL.InsertAfter
File:       \ObjAsm\Code\Macros\xLSLL.inc
Purpose:    Insert an object after another in the linked list.
Arguments:  Arg1: → Member to insert after.
            Arg2: → Member to insert.
            Arg3: Auxiliar register.
Note:       eax is trashed.


Macro:      xLSLL_RemoveAfter
File:       \ObjAsm\Code\Macros\xLSLL.inc
Purpose:    Remove a member from the linked list.
Arguments:  Arg1: → Member to remove after.
            Arg2: Auxiliar register, default is ecx.
Note:       eax is trashed.
```