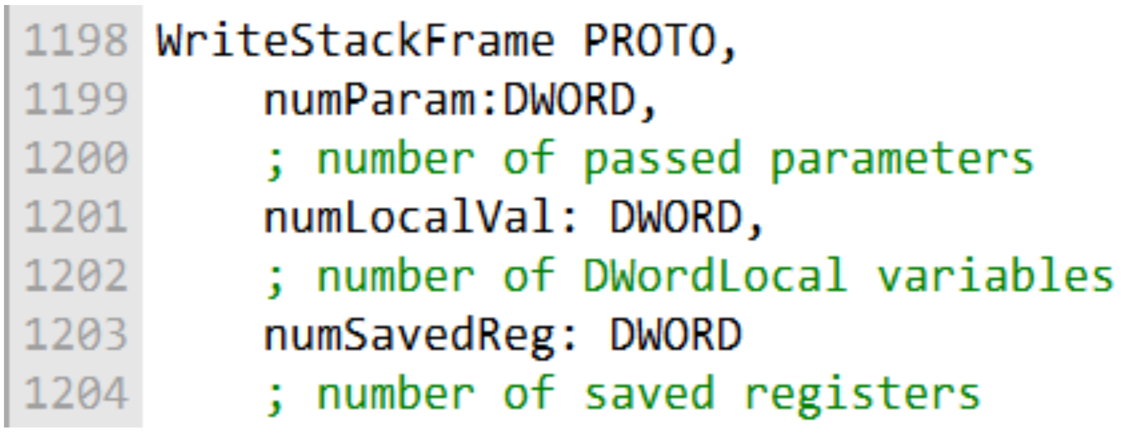
WireStackFrame Procedure

Here is a more in-depth explanation of the WriteStackFrame and WriteStackFrameName procedures:

The Irvine32 library contains a useful procedure named WriteStackFrame that displays the contents of the current procedure’s stack frame. It shows the procedure’s stack parameters, return address, local variables, and saved registers.



```
WriteStackFrame PROTO,
    numParam:DWORD,
    ; number of passed parameters
    numLocalVal: DWORD,
    ; number of DWordLocal variables
    numSavedReg: DWORD
    ; number of saved registers
```

The **WriteStackFrame procedure** displays the contents of the current stack frame, which contains the stack parameters, local variables, saved registers, and return address for the current procedure.

It takes 3 parameters:

- **numParam -** The number of parameters passed to the current procedure. This determines how many DWORDs to show for the parameters at the top of the stack.
- **numLocalVal -** The number of DWORD local variables allocated on the stack for the current procedure.
- **numSavedReg -** The number of registers saved on the stack for the current procedure. Typically this is 2 for EAX and EBX.

It displays the stack contents by starting at EBP and moving downward to ESP. For each DWORD it displays the offset from EBP and the hex value stored there.

The parameters passed to the procedure are displayed first at the highest offsets from EBP. Then the return address, saved EBP, local variables, and saved registers are displayed in descending offset order.

ESP points to the last used stack location, so the display stops when it reaches ESP.

WriteStackFrameName does the same thing, but takes an additional parameter:

- **procName** - A pointer to a null-terminated string containing the name of the current procedure. This is displayed at the top of the output.

So WriteStackFrameName allows you to identify which procedure's stack frame is being displayed. This is useful when multiple procedures call WriteStackFrame/Name.

In summary, these procedures give visibility into the stack contents at any point within a procedure. This helps debug issues with stack parameters, local variables, register saving, etc.

Here is an explanation of the MASM code example that was shown in the original text:

```
; In main procedure
main PROC
    mov eax, 0EAEAEAEAh   ; Save test value in EAX
    mov ebx, 0EBEBEBEBh   ; Save test value in EBX
    INVOKE myProc, 1111h, 2222h ; Call myProc, passing 2 parameters
    exit main             ; Exit program

main ENDP
; In myProc procedure
myProc PROC
    ; Procedure uses EAX and EBX, so they will be saved
    USES eax ebx
    x: DWORD, y:DWORD     ; Declare parameter variables
    LOCAL a:DWORD, b:DWORD  ; Declare local variables
    PARAMS = 2            ; 2 parameters
    LOCALS = 2            ; 2 local DWORD variables
    SAVED_REGS = 2        ; 2 saved registers (EAX and EBX)
    mov a,0AAAAh          ; Load value into local variable a
    mov b,0BBBBh          ; Load value into local variable b
    ; Display stack frame contents
    INVOKE WriteStackFrame, PARAMS, LOCALS, SAVED_REGS
myProc ENDP
```

The following sample output was produced by the call:

```
Stack Frame
00002222 ebp+12 (parameter)
00001111 ebp+8 (parameter)
00401083 ebp+4 (return address)
0012FFF0 ebp+0 (saved ebp) <--- ebp
0000AAAA ebp-4 (local variable)
0000BBBB ebp-8 (local variable)
EAEAEAEA ebp-12 (saved register)
EBEBEBEB ebp-16 (saved register) <--- esp
```

A second procedure, named WriteStackFrameName, has an additional parameter that holds the name of the procedure owning the stack frame:

```
WriteStackFrameName PROTO,
     numParam:DWORD,
     ; number of passed parameters
     numLocalVal:DWORD,
     ; number of DWORD local variables
     numSavedReg:DWORD,
     ; number of saved registers
     procName:PTR BYTE
     ; null-terminated string
```

The main procedure:

- Loads some sample values into EAX and EBX to be saved on the stack later.
- Calls the myProc procedure, passing two DWORD arguments (1111h and 2222h).
- Exits the program.

The myProc procedure:

- Uses EAX, EBX registers so they will be saved on the stack.
- Declares x and y parameters and a and b local variables.

- Loads sample values into the local variables.
- Calls WriteStackFrame, passing:

  - 2 for the number of parameters
  - 2 for the number of local DWORD variables
  - 2 for the number of saved registers (EAX and EBX)

This displays the contents of myProc's stack frame, including:

- **The 1111h and 2222h parameters**
- **The return address back to main**
- **The saved EBP from main**
- **The local variables a and b**
- **The saved EAX and EBX registers from main**

So this demonstrates how WriteStackFrame can display a procedure's stack contents to help understand and debug the stack usage.

You can find the source code for the Irvine32 library in the \Examples\Lib32 directory of our book’s install directory (usually C:\Irvine). Look for the file named Irvine32.asm.