Considerations for Writing Prolog/Epilog Code

Article • 08/03/2021

Microsoft Specific

Before writing your own prolog and epilog code sequences, it is important to understand how the stack frame is laid out. It is also useful to know how to use the `__LOCAL_SIZE` symbol.

Stack Frame Layout

This example shows the standard prolog code that might appear in a 32-bit function:

```
push        ebp                 ; Save ebp
mov         ebp, esp            ; Set stack frame pointer
sub         esp, localbytes     ; Allocate space for locals
push        <registers>         ; Save registers
```

The `localbytes` variable represents the number of bytes needed on the stack for local variables, and the `<registers>` variable is a placeholder that represents the list of registers to be saved on the stack. After pushing the registers, you can place any other appropriate data on the stack. The following is the corresponding epilog code:

```
pop         <registers>   ; Restore registers
mov         esp, ebp      ; Restore stack pointer
pop         ebp           ; Restore ebp
ret                       ; Return from function
```

The stack always grows down (from high to low memory addresses). The base pointer (`ebp`) points to the pushed value of `ebp`. The locals area begins at `ebp-4`. To access local variables, calculate an offset from `ebp` by subtracting the appropriate value from `ebp`.

__LOCAL_SIZE

The compiler provides a symbol, `__LOCAL_SIZE`, for use in the inline assembler block of function prolog code. This symbol is used to allocate space for local variables on the stack frame in custom prolog code.

The compiler determines the value of `__LOCAL_SIZE`. Its value is the total number of bytes of all user-defined local variables and compiler-generated temporary variables. `__LOCAL_SIZE` can be used only as an immediate operand; it cannot be used in an expression. You must not change or redefine the value of this symbol. For example:

```
mov        eax, __LOCAL_SIZE           ;Immediate operand--Okay
mov        eax, [ebp - __LOCAL_SIZE]   ;Error
```

The following example of a naked function containing custom prolog and epilog sequences uses the `__LOCAL_SIZE` symbol in the prolog sequence:

C++

```cpp
// the__local_size_symbol.cpp
// processor: x86
__declspec ( naked ) int main() {
   int i;
   int j;

   __asm {      /* prolog */
      push   ebp
      mov      ebp, esp
      sub      esp, __LOCAL_SIZE
      }

   /* Function body */
   __asm {   /* epilog */
      mov      esp, ebp
      pop      ebp
      ret
      }
}
```

END Microsoft Specific

See also

Naked Function Calls

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A