Rules and Limitations for Naked Functions

Article • 08/03/2021

Microsoft Specific

The following rules and limitations apply to naked functions:

- The `return` statement is not permitted.
- Structured Exception Handling and C++ Exception Handling constructs are not permitted because they must unwind across the stack frame.
- For the same reason, any form of `setjmp` is prohibited.
- Use of the `_alloca` function is prohibited.
- To ensure that no initialization code for local variables appears before the prolog sequence, initialized local variables are not permitted at function scope. In particular, the declaration of C++ objects is not permitted at function scope. There may, however, be initialized data in a nested scope.
- Frame pointer optimization (the /Oy compiler option) is not recommended, but it is automatically suppressed for a naked function.
- You cannot declare C++ class objects at the function lexical scope. You can, however, declare objects in a nested block.
- The `naked` keyword is ignored when compiling with /clr.
- For __fastcall naked functions, whenever there is a reference in C/C++ code to one of the register arguments, the prolog code should store the values of that register into the stack location for that variable. For example:

C++

```
// nkdfastcl.cpp
// compile with: /c
// processor: x86
__declspec(naked) int __fastcall  power(int i, int j) {
   // calculates i^j, assumes that j >= 0

   // prolog
   __asm {
      push ebp
      mov ebp, esp
      sub esp, __LOCAL_SIZE
     // store ECX and EDX into stack locations allocated for i and j
     mov i, ecx
     mov j, edx
   }

   {
      int k = 1;   // return value
      while (j-- > 0)
         k *= i;
      __asm {
         mov eax, k
      };
   }
```

```
    // epilog
    __asm {
       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