# asm declaration

[[edit]](http://en.cppreference.com/mwiki/index.php?title=Template:cpp/language/declarations/navbar_content&action=edit)

*asm-declaration* gives the ability to embed assembly language source code within a C++ program. This declaration is conditionally-supported and implementation defined, meaning that it may not be present and, even when provided by the implementation, it does not have a fixed meaning.

# [[edit](http://en.cppreference.com/mwiki/index.php?title=cpp/language/asm&action=edit&section=1)]Syntax

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | |
| **asm (** *string\_literal* **)** **;** |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | |

# [[edit](http://en.cppreference.com/mwiki/index.php?title=cpp/language/asm&action=edit&section=2)]Explanation

The *string\_literal* is typically a short program written in assembly language, which is executed whenever this declaration is executed. Different C++ compilers have wildly varying rules for asm-declarations, and different conventions for the interaction with the surrounding C++ code.

As other [block declarations](http://en.cppreference.com/w/cpp/language/declarations), this declaration can appear inside a block (a function body or another compound statement), and, as all other declarations, this declaration can also appear outside a block.

|  |  |
| --- | --- |
|  | This section is incomplete Reason: write a note on GCC extended assembly syntax, since it is now supported by Intel, IBM, Sun (as of v12), etc |

# [[edit](http://en.cppreference.com/mwiki/index.php?title=cpp/language/asm&action=edit&section=3)]Example

Demonstrates two kinds of inline assembly syntax offered by the GCC compiler. This program will only work correctly on x86\_64 platform under Linux.

#include <iostream>

extern "C" int func();

// the definition of func is written in assembly language

asm(".globl func\n\t"

".type func, @function\n\t"

"func:\n\t"

".cfi\_startproc\n\t"

"movl $7, %eax\n\t"

"ret\n\t"

".cfi\_endproc");

int main() {

int n = func();

// extended inline assembly

asm ("leal (%0,%0,4),%0"

: "=r" (n)

: "0" (n));

[std::cout](http://en.cppreference.com/w/cpp/io/cout) << "7\*5 = " << n << [std::endl](http://en.cppreference.com/w/cpp/io/manip/endl); // flush is intentional

// standard inline assembly

asm ("movq $60, %rax\n\t" // the exit syscall number on Linux

"movq $2,  %rdi\n\t" // this program returns 2

"syscall");

}

Output:

7\*5 = 35

# END