

**Nadeau** software consulting

C/C++ tip: How to detect the processor type using compiler predefined macros

February 9, 2012

Topics: [C/C++](#)

Processor macros are predefined by all C/C++ compilers to enable `#if/#endif` sets to wrap processor-specific code, such as in-line assembly for SSE instructions on x86 processors. But there are no standards for processor macros. The same compiler may have different macros on different operating systems, and different compilers for the same processor may have different macros. **This article surveys common compilers and shows how to use predefined macros to detect common desktop and server processors at compile time.**

Table of Contents

[How to list predefined macros](#)[How to detect the processor type](#)[Itanium](#)[POWER](#)[SPARC](#)[x86 and x86-64](#)[Further reading](#)[Related articles at NadeauSoftware.com](#)[Web articles](#)

How to list predefined macros

See [How to list compiler predefined macros](#) for instructions on getting a list of macros for the compilers referenced here.

How to detect the processor type

Throughout the following sections note:

- **Red text** indicates deprecated macros that don't start with an underscore. C++ compilers, and C compilers in standards compliance mode, do not define them.
- **Green text** indicates recommended macros that are well-supported and useful for detecting a specific OS.

Itanium

A.K.A.: IA64**Developer:** Intel**Processors:** Itanium, Itanium 2, Itanium 2 9000/9100/9300, etc.

```
#if defined(__ia64) || defined(__itanium__) || defined(_M_IA64)
    /* Itanium ----- */
#endif
```

Itanium

	GNU GCC/G++		HP C/aC++	Intel ICC/ICPC	Microsoft Visual Studio
	<i>BSD,</i>				
Macro	<i>Linux</i>	<i>HP-UX</i>	<i>HP-UX</i>	<i>Linux</i>	<i>Windows</i>
ia64				yes	
__ia64	yes	yes	yes	yes	
__ia64__	yes	yes		yes	
__IA64__		yes			
__itanium__	yes	yes		yes	
__M_IA64					yes

Notes:

- "IA64" is the old name for the processor architecture. Intel now prefers "Itanium".
- There is no single Itanium processor macro defined by all compilers on all OSes. An `#if/#endif` that checks multiple macros is required.
- Microsoft's support for Itanium ended after Visual Studio 2010 and Windows Server 2008.
- Clang/LLVM currently does not support Itanium processors.

POWER

A.K.A.: PowerPC

Developer: IBM, Freescale

Processors: PowerPC, POWER 1/2/3/4/5/6/7, G1, G2, G3, G4, G5, etc.

```
#if defined(__powerpc__) || defined(__ppc__) || defined(__PPC__)
/* POWER ----- */

#if defined(__powerpc64__) || defined(__ppc64__) || defined(__PPC64__) || \
    defined(__64BIT__) || defined(__LP64__) || defined(__LP64__)
/* POWER 64-bit ----- */

#else
/* POWER 32-bit ----- */

#endif
#endif
```

POWER 32-bit

	Clang/LLVM						GNU GCC/G++			IBM XL C/C++					
Macro	BSD, Linux, OSX						AIX	FreeBSD	Linux	NetBSD	OpenBSD	OSX	AIX	Linux	
_ARCH_PPC	yes														
_POWER	yes														
powerpc							yes		yes						
__powerpc							yes		yes	yes					yes
__powerpc__	yes	yes		yes		yes		yes	yes					yes	
__PowerPC__							yes								
__POWERPC__	yes						yes								
PPC							yes		yes	yes					
__ppc__	yes						yes		yes						
__PPC							yes		yes	yes					yes

<code>__PPC__</code>	yes	yes	yes	yes	yes	yes
----------------------	-----	-----	-----	-----	-----	-----

POWER 64-bit

Macro	Clang/LLVM GNU GCC/G++						IBM XL C/C++		
	<i>BSD, Linux, OSX</i>	<i>AIX</i>	<i>FreeBSD</i>	<i>Linux</i>	<i>NetBSD</i>	<i>OpenBSD</i>	<i>OSX</i>	<i>AIX</i>	<i>Linux</i>
<code>_ARCH_PPC</code>	yes								
<code>_ARCH_PPC64</code>	yes								
<code>_POWER</code>		yes							
<code>__powerpc</code>				yes				yes	yes
<code>__powerpc__</code>	yes	yes	yes	yes	yes			yes	yes
<code>__powerpc64__</code>	yes		yes	yes				yes	yes
<code>__PowerPC__</code>			yes						
<code>__POWERPC__</code>	yes						yes		
<code>__ppc__</code>	yes		yes						
<code>__ppc64</code>								yes	
<code>__ppc64__</code>	yes						yes		
<code>__PPC</code>				yes		yes		yes	yes
<code>__PPC__</code>		yes	yes	yes		yes		yes	yes
<code>__PPC64__</code>			yes	yes				yes	yes

Notes:

- There is no single POWER processor macro defined by all compilers on all OSes. An `#if/#endif` that checks multiple macros is required.
- GCC for AIX, NetBSD, and OpenBSD defines the same macros for 32-bit and 64-bit POWER processors. For AIX, `__64BIT__` is defined for 64-bit POWER. For OpenBSD, `__LP64` and `__LP64__` are defined for 64-bit POWER. For NetBSD, GCC doesn't provide a macro to check for 64-bit use.
- Apple's OSX support for POWER processors ended after OSX 10.5 Leopard in 2007. The open source [Darwin](#) distribution, on which OSX is based, is still available for POWER processors.

SPARC**Developer:** Oracle, Fujitsu, Sun**Processors:** UltraSPARC I/II/III/IV/T1/T2, SPARC T3/T4, etc.

```
#if defined(__sparc)
/* SPARC ----- */
#endif
```

SPARC

Macro	Clang/LLVM	GNU GCC/G++		Oracle Solaris Studio
	BSD, Linux, Solaris	BSD	Linux, Solaris	Solaris
<code>sparc</code>	yes	yes	yes	yes
<code>__sparc</code>	yes	yes	yes	yes
<code>__sparc__</code>	yes	yes	yes	

<code>__sparc64__</code>	yes
--------------------------	-----

Notes:

- GCC defines processor name macros depending upon the value of the `-march` command-line option. These include `__sparclite__`, `__sparclet__`, `__sparc_v8__`, `__sparc_v9__`, `__supersparc__`, `__hypersparc__`, and so forth. However, other compilers don't provide this level of detail and writing code that depends upon these macros is probably a bad idea.

x86 and x86-64

A.K.A. (32-bit): IA-32, i386, x86, x86-32

A.K.A. (64-bit): AMD64, EM64T, IA-32e, Intel64, x64, x86-64

Developers: AMD, Intel

Processors: Athlon, Atom, Core, Core 2, Core i3/i5/i7, Opteron, Pentium, Phenom, Sempron, Turion, etc.

```
#if defined(__x86_64__) || defined(_M_X64)
/* x86 64-bit ----- */

#elif defined(__i386) || defined(_M_IX86)
/* x86 32-bit ----- */

#endif
```

x86 32-bit

Macro									Oracle Solaris Studio	Microsoft Visual Studio
	Clang/LLVM			GNU GCC/G++		Intel ICC/ICPC		Portland PGCC/PGCPP		
	BSD, Cygwin, Linux, OSX, Solaris	MinGW	Windows	BSD, Linux, OSX, Solaris	Cygwin, MinGW, Windows	Linux, OSX	Windows	Linux, OSX, Windows	Linux, Solaris	Windows
i386	yes	yes	yes	yes	yes	yes		yes	yes	
__i386	yes	yes	yes	yes	yes	yes		yes	yes	
__i386__	yes	yes	yes	yes	yes	yes		yes		
_M_IX86			yes				yes			yes
X86		yes			yes					

x86 64-bit

Macro									Oracle Solaris Studio	Microsoft Visual Studio
	Clang/LLVM			GNU GCC/G++		Intel ICC/ICPC		Portland PGCC/PGCPP		
	BSD, Linux, MinGW, OSX, Solaris	Windows		BSD, Linux, MinGW, OSX, Solaris	Linux, OSX	Windows		Linux, OSX, Windows	Linux, Solaris	Windows
__x86_64	yes	yes	yes		yes				yes	
__x86_64__	yes	yes	yes		yes			yes	yes	
__amd64	yes	yes	yes						yes	
__amd64__	yes	yes	yes					yes	yes	
_M_AMD64		yes				yes				yes
_M_X64		yes				yes				yes

Notes:

- There is no single x86 processor macro defined by all compilers on all OSes. An `#if/#endif` that checks multiple macros is required.
- 64-bit instructions for the x86 architecture originated with AMD and were later adopted by Intel. The `_amd64`, `_amd64__`, and `_M_AMD64` macros are for legacy support, while the newer `__x86_64`, `__x86_64__`, and `_M_X64` are vendor-generic.
- With appropriate command-line options, Clang/LLVM and GCC can build 32-bit and 64-bit binaries for Windows instead of POSIX. The macros they define differ between POSIX and Windows.
- GCC and Clang/LLVM define a variety of processor name macros, depending upon the value of the `-march` command-line option. These include `__i486__`, `__i586__`, `__pentium__`, `__pentiumpro__`, `__athlon__`, `__atom__`, `__core2__`, `__corei7__`, `__k8__`, and so forth. However, other compilers do not provide this level of detail and writing code that depends upon these macros is probably a bad idea.

Further reading

Related articles at NadeauSoftware.com

[C/C++ tip: How to list compiler predefined macros](#) explains how to get a compiler's macros by using command-line options and other methods.

[C/C++ tip: How to detect the compiler name and version using compiler predefined macros](#) provides `#if/#endif` sets for detecting common compilers.

[C/C++ tip: How to detect the operating system type using compiler predefined macros](#) provides `#if/#endif` sets for detecting desktop and server operating systems using compiler macros.

Web articles

[Architectures](#) at Sourceforge.net provides a list of current and obsolete processors and some of the macros used to detect them. Unfortunately, the list is out of date, occasionally wrong, doesn't include several compilers, and doesn't show differences between compilers and OSes.

[List of CPU architectures](#) at Wikipedia.org provides categories of processor architectures and links to further information about each. No information is provided on predefined macros.

[Pre-defined C/C++ Compiler Macros](#) at beefchunk.com has a brief list of processors and predefined macros. However, the list is incomplete and doesn't show differences among compilers and operating systems.

Comments

Copyright © Nadeau Software Consulting (Dr. David R. Nadeau). All rights reserved.