

Xcode equivalent of ' __asm int 3 / DebugBreak() / Halt?

Asked 16 years, 3 months ago Modified 12 years, 3 months ago

Viewed 19k times



23



What's the instruction to cause a hard-break in Xcode?
For example under Visual Studio I could do '`__asm int 3`' or '`DebugBreak()`'. Under some GCC implementations it's `asm("break 0")` or `asm("trap")`.

I've tried various combos under Xcode without any luck.
(inline assembler works fine so it's not a syntax issue).

For reference this is for an assert macro. I don't want to use the definitions in `assert.h` both for portability, and because they appear to do an `abort()` in the version XCode provides.

John - Super, cheers. For reference the `int 3` syntax is the one required for Intel Macs and iPhone.

Chris - Thanks for your comment but there are many reasons to avoid the standard `assert()` function for codebases ported to different platforms. If you've gone to the trouble of rolling your own assert it's usually because you have additional functionality (logging, stack unwinding, user-interaction) that you wish to retain.

Your suggestion of attempting to replace the handler via an implementation of '__assert' or similar is not going to be portable. The standard 'assert' is usually a macro and while it may map to __assert on the Mac it doesn't on other platforms.

xcode

macos

debugbreak

Share

Improve this question

Follow

edited Aug 31, 2011 at 18:12



gcamp

14.7k ● 4 ● 56 ● 86

asked Sep 1, 2008 at 0:18



Andrew Grant

58.8k ● 22 ● 131 ● 144

7 Answers

Sorted by:

Highest score (default)



24

http://developer.apple.com/documentation/DeveloperTools/Conceptual/XcodeProjectManagement/090_Running_Programs/chapter_11_section_3.html



```
asm {trap}                ; Halts a program running on PPC
__asm {int 3}              ; Halts a program running on IA-
```





-
- 7 With the GCC/clang ASM syntax, this becomes
`__asm__("int $3")` for Intel Macs and (probably)
`__asm__("trap")` for iDevices. – [zneak](#) Oct 28, 2012 at 22:01
-
- 2 The link above appears to be dead now. – [rstackhouse](#) Jan 21, 2015 at 17:35
-



12

You can just insert a call to `Debugger()` — that will stop your app in the debugger (if it's being run under the debugger), or halt it with an exception if it's not.



Also, **do not avoid** `assert()` for "portability reasons" — portability is why it exists! It's part of Standard C, and you'll find it wherever you find a C compiler. What you really want to do is define a new *assertion handler* that does a debugger break instead of calling `abort()`; virtually all C compilers offer a mechanism by which you can do this.



Typically this is done by simply implementing a function or macro that follows this prototype:

```
void __assert(const char *expression, const char *file
```

It's called when an assertion expression fails. Usually it, not `assert()` itself, is what performs "the `printf()` followed by `abort()`" that is the default documented behavior. By customizing this function or macro, you can change its behavior.

Share Improve this answer

edited Sep 1, 2008 at 22:36

Follow

answered Sep 1, 2008 at 22:28



Chris Hanson

55k ● 8 ● 74 ● 104

-
- 2 What is a portable way to drop into the debugger on the failed line? `assert()` drops you into library code...
– [JBRWilkinson](#) Oct 5, 2011 at 17:41
-
- 1 Thanks Chris, but first - `Debugger()` is marked "deprecated" in the headers, without any direction for a replacement, Second - I don't know what to link against, in order to have it - normal "Cocoa" and "CoreFoundation" frameworks don't have it - I have a link error for a missing symbol on `_Debugger` – [Motti Shneur](#) Jan 14, 2016 at 8:29
-



```
__builtin_trap();
```

6



Since `Debugger()` is depreciated now this should work instead.





https://developer.apple.com/library/mac/technotes/tn2124/_index.html#//apple_ref/doc/uid/DTS10003391-CH1-SECCONTROLLEDCRASH

Share Improve this answer

answered Sep 7, 2012 at 16:16

Follow



Tod

4,678 ● 5 ● 34 ● 23

- 1 `__builtin_trap` crashes your program with a `SIGILL`, though (instead of barely interrupt it). – [zneak](#) Oct 28, 2012 at 21:58



3



For posterity: I have some code for generating halts at the correct stack frame in the debugger and (optionally) pausing the app so you can attach the debugger just-in-time. Works for simulator and device (and possibly desktop, if you should ever need it). Exhaustively detailed post at <http://iphone.m20.nl/wp/2010/10/xcode-iphone-debugger-halt-assertions/>



Share Improve this answer

edited Apr 27, 2011 at 8:02

Follow

answered Apr 27, 2011 at 7:45



Steven Kramer

8,503 ● 2 ● 39 ● 43



I found the following in an [Apple Forum](#):

2



Xcode doesn't come with any symbolic breaks built in - but they're quick to add. Go to the breakpoints window and add:

```
-[NSException raise]
```

Share Improve this answer

Follow

answered Sep 1, 2008 at 0:22



[GateKiller](#)

75.8k ● 75 ● 175 ● 204



```
kill(getpid(), SIGINT);
```

2

Works in the simulator and the device.



Share Improve this answer

Follow

answered Jun 1, 2011 at 17:54



[Sonny Saluja](#)

7,287 ● 2 ● 28 ● 39

First of all, = `raise(SIGINT)` . Second, these have pesky tendency to get forwarded into the main thread, so if you do it on another thread, you don't actually stop where you raised it. – [Ilya](#) Jul 26, 2015 at 11:26



There is also the following function that is available as cross platform straight `Halt()` alternative:

1



```
#include <stdlib.h>
```

```
void abort(void);
```



We use it in our cross platform engine for the iPhone implementation in case of fatal asserts. Cross platform across Nintendo DS/Wii/XBOX 360/iOS etc...

Share Improve this answer

Follow

edited Oct 30, 2011 at 22:05



logancautrell

8,772 ● 3 ● 40 ● 50

answered Apr 6, 2009 at 6:29



Peter

