Will an app built with only armeabi run on armeabi-v7a devices?

Asked 11 years, 5 months ago Modified 10 years, 5 months ago

Viewed 8k times Part of Mobile Development Collective

12

The Xamarin documentation is a bit unclear on this. If I build my app with only armeabi ticked in the build preferences, will my app:

1. Be available for v7a devices in the Play Store?
2. Run on v7a devices?

If it does run, are there any features like using threads that will lead to unexpected behaviour or crashes?

I've got a simple app and am trying to keep it small. Also, I don't have a v7a device to do a quick experiment.

Clarification:

While there seems to be clear acceptance that it is "safe, but not so performant" to compile an Android app with only the amreabi library (see this excellent post: Why use armeabi-v7a code over armeabi code?), the Xamarin docs on CPU architecture, that I assume applies to their compiled .so libraries, says:

> it is important to remember that the armeabi runtime used by Xamarin.Android is thread safe. If an application that has armeabi support is deployed to an armeabi-v7a device, many strange and unexplainable exceptions will occur.

I have since been able to test my app that is just compiled with armeabi on a v7a device and haven't run into any "strange and unexplainable exceptions" YET.

Update:

Looks like the Xamarin docs has since been updated and now (2014-07-14) reads:

> it is important to remember that the armeabi runtime used by Xamarin.Android is *not thread safe*. If an application that has armeabi support is deployed to an armeabi-v7a device, many strange and unexplainable exceptions will occur.

MD

android build xamarin.android xamarin cpu-architecture

Share

Improve this question

Follow

edited May 23, 2017 at 12:33

Community Bot

1 ●1

asked Jul 2, 2013 at 15:06

Jannie Theunissen
30.1k ● 21 ● 106 ● 136

5 Unless Xamarin screwed something up, `armeabi` code will run just fine on ARM v7a devices. It certainly does for normal Android NDK development. – CommonsWare Jul 5, 2013 at 22:43

4 Answers

Sorted by: Highest score (default)

8

According to the Xamarin Android documentation, armeabi code will crash in unexpected ways on a multi-core armeavi-v7 device.

http://docs.xamarin.com/guides/android/advanced_topics/cpu_architecture

> **Section 1.1**
>
> Note: Xamarin.Android’s armeabi code is not thread safe and should not be used on multi-CPU armeabi-v7a devices (described below). Using aremabi code on a single-core armeabi-v7a device is safe.

The reason that Xamarin Android make it a requirement to include armeabi-v7a has to do with thread safe memory access. Put simply the armeabi instruction set lacks the instructions necessary to safely lock memory on SMP devices.

The most thorough discussion of the issue can be found in this bug report: https://bugzilla.xamarin.com/show_bug.cgi?id=7013

> *Jonathan Pryor 2012-09-20 11:41:45 EDT*
>
> As far as I can determine, it is (nearly) IMPOSSIBLE to SAFELY use an armeabi library on a SMP armeabi-v7a device. This is because armeabi lacks the CPU instructions necessary to safely lock data on SMP devices, so if the armeabi library contains data that must be protected against access from multiple threads, it's busted, and libmonodroid.so is such a library. This may be fixable by creating a libmonodroid.so which dynamically determines the runtime CPU, allowing it to use either armeabi or armeabi-v7a lock instructions accordingly, but this has not been done yet, and the implementation timeframe is unknown.
>
> Thus, if your app will be running on SMP hardware, you should include the armeabi-v7a runtime with your app. This can be done in the Project Options dialog.

These crashes are rare but catastrophic and very hard to debug as you experience random memory corruption and segmentation faults.

I was able to reproduce the issue reliably on a Galaxy S3. Some example code that demonstrates the crash is in this bug report: https://bugzilla.xamarin.com/show_bug.cgi?id=7167

It's unknown whether or not this bug affects other NDK applications on Android. But it definitely affects Xamarin Android.

Share Improve this answer Follow

answered Oct 1, 2013 at 6:27
Jared Kells
7,012 ● 4 ● 40 ● 44

Excellent, authoritative light on the subject, Jared. Thanks a lot. – Jannie Theunissen Oct 1, 2013 at 7:02

7

+50

I clicked through and read the Xamarin comments. Based on reading them, I think you are asking the wrong question. The answer to the question you asked is (as CommonsWare stated in his comment), "yes, unless Xamarin screwed something up". Unfortunately, their docs indicate that they think that they did, arguably, screw something up. There are some typos in their documentation that confuse things a bit, specifically in one place (Section 1.1) they say "is thread safe" when they clearly mean "is NOT thread safe". They restate this correctly in Section 1.2:

> Note: Xamarin.Android’s armeabi code is not thread safe and should not be used on multi-CPU armeabi-v7a devices (described below). Using aremabi code on a single-core armeabi-v7a device is safe.

I think if you combine information from sections 1.2 and 1.1, it becomes clear what Xamarin is telling you. To be clear I'm just restating what their documentation says, not making any assertions about the veracity of their documentation. That is, in the case where the armeabi libs (which are not thread safe) get loaded on a multi-core or multi-processor device, bad things may happen. This case can arise due to a bug in ICS (4.0.0-4.0.3). Therefore:

> applications built by using Xamarin.Android 4.2 or lower should explicitly specify the armeabi-v7a as the sole ARM-based ABI

Here is the actual info from their docs (formatting added) rearranged into an order that may help make it clearer:

> **From Section 1.2.1**
>
> Note: Xamarin.Android’s armeabi code is **not thread safe** and should not be used on **multi-CPU** armeabi-v7a devices (described below).

> Using aremabi code on a single-core armeabi-v7a device is safe.
>
> **From Section 1.1**
>
> Due to a bug in Android 4.0.0, 4.0.1, 4.0.2, and 4.0.3, the native libraries will be picked up from the armeabi directory even though there is an armeabi-v7a directory present and the device is an armeabi-v7a device.
>
> Note: Because of these reasons, it is strongly recommended that applications built by using Xamarin.Android 4.2 or lower should explicitly specify the armeabi-v7a as the sole ARM-based ABI.

I think that based on the rest of the doc, this is what the first paragraph in Section 1.1 should say (bold edits are mine):

> The Application Binary Interface will be discussed in detail below, but it is important to remember that the armeabi runtime used by Xamarin.Android is **not** thread safe. If an application that has armeabi support is deployed to **multi-CPU** armeabi-v7a device, many strange and unexplainable exceptions will occur.

Share Improve this answer edited Jun 20, 2020 at 9:12

Follow

Community Bot
1 ● 1

answered Jul 11, 2013 at 15:41

ajh158
1,467 ● 1 ● 13 ● 32

Great interpretation and quite convincing. Thanks a lot!
– Jannie Theunissen Jul 11, 2013 at 17:03

The number of cores should not have any impact on thread safety. Android can switch threads at any time even on a single core device, and this can be in the middle of a memory write. The statements above do not give me any confidence in the stability of Xamarin on any hardware. – Danny Parker Jul 12, 2013 at 10:56

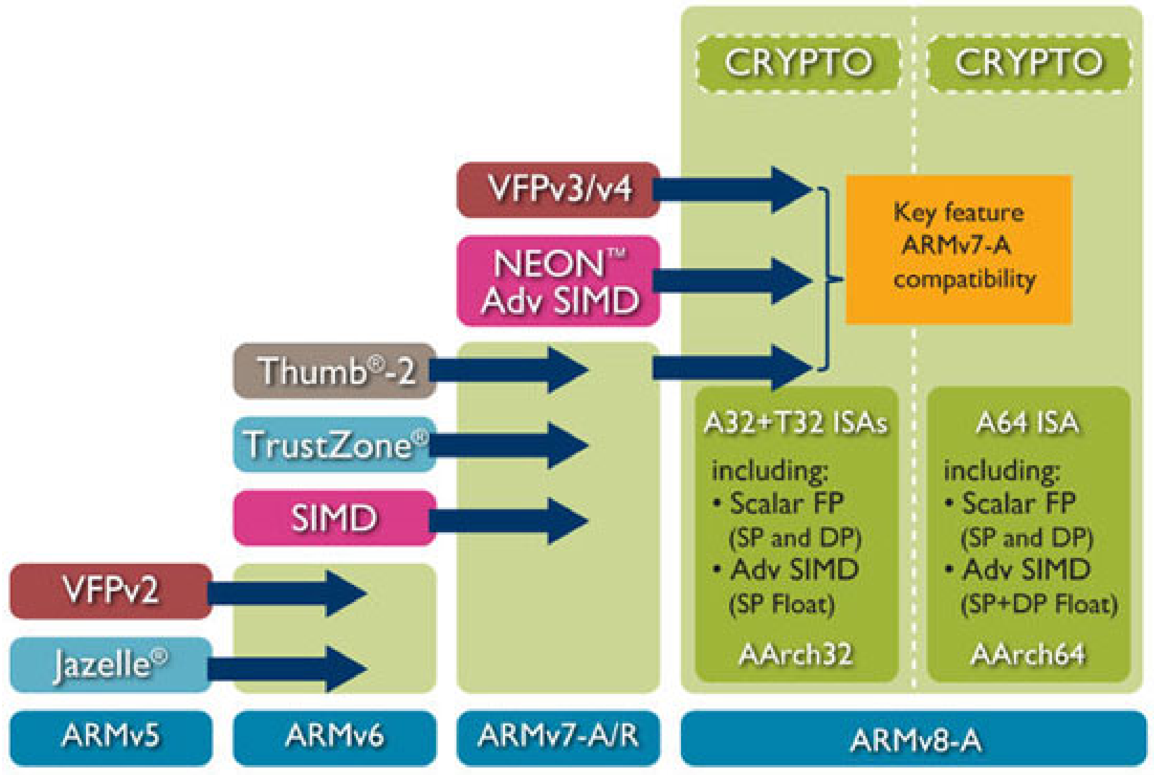
This is one of the reasons that I pointed out that my answer is intended to help clarify the information that the Xamarin docs are attempting to convey - not to make any assertions about the accuracy of the docs. If you read on to near the end of section 1.3.2, it indicates that "This can also result in obscure run-time errors, as armeabi is not SMP safe." My read on JannieT's question was "what is it that this Xamarin doc is trying to tell me?"; it certainly seems like Xamarin found an issue and is trying to warn their users about it (poorly), so my intention was to help interpret the doc. – ajh158 Jul 12, 2013 at 12:10

It might make sense for JannieT to edit their question to make it Xamarin specific; it's tagged as such, but the Question itself is general where it should be specific.
– ajh158 Jul 12, 2013 at 12:11

http://www.arm.com/products/processors/instruction-set-architectures/index.php

3

If you look at this diagram it explains the ethos of ARM processor design.


CRYPTO
CRYPTO
VFPv3/v4
NEON™ Adv SIMD
Key feature ARMv7-A compatibility
Thumb®-2
TrustZone®
SIMD
VFPv2
Jazelle®
A32+T32 ISAs including: • Scalar FP (SP and DP) • Adv SIMD (SP Float)
AArch32
A64 ISA including: • Scalar FP (SP and DP) • Adv SIMD (SP+DP Float)
AArch64
ARMv5
ARMv6
ARMv7-A/R
ARMv8-A

New iterations extend the base feature set, but don't change it. NEON and SIMD need to be directly referenced to be used, so cannot be referenced from an ARMv5 binary. Unless your binary is huge (that's the actual executable, not the whole APK), I would compile both and get the best of both worlds. See this question for details.

Regardless, I would contact Xamarin to clarify that slightly loaded 'unexplainable exceptions' statement. If they perceive issues with their code running on multiple processors then their code is inherently not thread safe regardless of the number of cores.

Share Improve this answer Follow

edited May 23, 2017 at 11:45

Community Bot
1 ●1

answered Jul 11, 2013 at 13:35

Danny Parker
1,773 ● 18 ● 32

Thanks Danny. If I include all the base libraries to be completely save and compatible, the very simplest and smallest compiled app would be bigger than 8Mb. I want to asses the risk I take of publishing something smaller and while your answer cast a bit more light it doesn't relieve my fears and doubts. – Jannie Theunissen Jul 11, 2013 at 14:48

This is the nature of basing your app on top of Xamarin I am afraid. If they are not providing multiple .so and telling you which ones you need to include for specific features of Xamarin this will lead to app bloat. You are basically including a bunch of code that your app may not need because of how they are bundling their platform. – Nick Palmer Jul 12, 2013 at 9:44

0

Yes.

armeabi is the general base and armeabi-v7a includes some additional instructions not found in the armeabi instruction set. v7a has support for hardware floating point operations, which can make your code much faster if it is doing any floating point operations. Android will attempt to load an armeabi-v7a library first if the hardware supports that, but if not, it will fall back to the armeabi version.

Share Improve this answer

answered Jul 10, 2013 at 3:41

Follow

Nick Palmer
2,743 ● 2 ● 27 ● 35

How do you know this, Nick? See my clarification.
– Jannie Theunissen Jul 10, 2013 at 6:24

Xamarin's comment implies that there code may not be thread safe to me, and I would contact them for clarification, since that statement from them is very vague. Regardless, they SHOULD be shipping .so for both architecture, and for x86 now as well. – Nick Palmer Jul 12, 2013 at 9:42