David Whaley [author]
david.whaley@arm.com
Nathan Li 李陈鲁 [译]
nathan.li@arm.com

64 位移动计算白皮书 [64-bit Computing for Mobile]

序言

谨以此文献给希望了解为何 Arm 64 位指令集架构（AArch64）是移动设备中不可或缺之构成要件的安卓应用开发者或技术决策人。本文亦将探讨与 64 位安卓应用开发相关的优势，裨益，与思考，并向读者介绍如何进一步获取更为详尽的（技术）指南。

一般而言，概念上“64 位”通常与一台设备的中央处理器（CPU）紧密相连。一颗 64 位的 CPU 被设计用于操作 64 位字长的整型数据。相较于 32 位设备，通常意味着它能更有效地处理更大的数据块。在计算机体系架构的概念中，“64 位”同时也指用于寻址和数据存储的寄存器宽度。理论上，64 位宽的地址寄存器可以访问极其巨大的数字空间，上至 2 的 64 次方个独特的内存地址单元，而相应的 32 位寄存器只能涵盖 2 的 32 次方地址范围。对应于实际数字，通常这意味着高达 18 艾（180 千亿兆）字节的海量存储，远超 32 位环境下仅仅 4 千兆字节的（容量）上限。同等条件下，一颗 64 位的处理器亦能比 32 位处理器更有效率地在更广阔的地址空间内处理（有效范围）更大的数据类型。尽管实际观测到的性能提升常常会被各种因素所左右，但整体而言，64 位处理器已被证实代表着更快的运行速度，更低延时的数据吞吐，以及更迅捷的用户响应（依托于出色的软件实现）。

Arm 何以 64 位？ Why Arm 64-bit?

64 位 CPU 体系架构在当代计算（学）范畴内，意味着一系列的优越特性。Arm 的“Armv8-A”架构于 2011 年问世，其所配备的宽寻址寄存器允许处理器无障碍访问远大于 4 千兆字节的地址空间。因此，一颗 64 位处理器相较于 32 位处理器，能以显著更快的速度处理更丰富的内存数据集合，进而颇为有效地管理海量数据结构（对象），大型的数组队列，与更宽裕的存储空间。

在移动计算领域内，Arm 始终精益求精，开拓创新，持续不断优化 32 位与 64 位的处理器架构以及相关核心设计，通常广为人知的有 AArch32 与 AArch64 架构。随着安卓内核成功移植到 64 位，其余的操作系统核心组件，程序库，和应用程序如今都能完美运行于 32 位或 64 位两种体系下。Armv8 架构向下兼容过往的 32 位 Arm 架构产品，然而对于前沿的算力挑战，如人工智能（AI），机器学习（ML），3D 游戏，以及 4K 超高清显示等等而言， 伴随 32 位指令集（ISA）而生的种种限制为人们诟病久已！

英雄总有迟暮时，AArch32 的架构实现历经多年演化，余下的改进空间日趋有限。在过去 10 年中，曾凭借出色的能效比带给我们无数欢乐时光的 32 位架构，已愈发难以被进一步提升，Arm 倾力探索并实施的种种优化措施所能带来的增益也日渐式微。而与之相对应，投入同样的工程资源到 64 位指令集的研发与优化上却意味着巨大的潜在收益。

今天有无数运行于 Arm CPU 上的高效能移动应用，一个 64 位的体系架构将能保障它们未来的可持续发展，并孕育显著的创新机遇。以安卓生态的演进与谷歌推动的变革为一致愿景，业界同仁正在投入巨大的努力来确保安卓原生应用的开发能如期迈入全面支持 64 位体系的时代。

来自谷歌开发者博客的讯息 Google’s Developer Blog Messages

一篇发表于 2017 年 12 月份的（谷歌）开发者博客郑重宣布，“*In anticipation of future Android devices that support 64-bit code only, the Play Console will require that new apps and app updates with native libraries provide 64-bit versions in addition to their 32-bit versions. This can be within a single APK or as one of the multiple APKs published.*” – “因预期到未来的安卓设备将只支持 64 位（指令集），（Google） Play 应用商店将要求所有包含原生程序库的新上架的应用和（针对现有）应用的版本更新，在原先的 32 位构建基础上，（必须）额外提交 64 位构建。这一举措可以通过发布（同时包含 32 位和 64 位可执行文件的）单一或多个（独立）的安卓应用安装包（APK）来实施”。

这篇标志性的开发者博客同时也揭示了原生应用向 64 位迁移的各个关键时间节点，“*… We are not removing 32-bit support. Google Play will continue to support 32-bit apps and devices. Apps that do not include native code are unaffected. This change will come into effect in August 2019 …*” – “我们并非要舍弃 32 位支持，Google Play（商店）将一如既往地支持 32 位应用及设备。任何不涉及原生程序库的（移动）应用都不此受影响，这一（重大）革新将于 2019 年 8 月 起生效”。

一篇随后发布的（谷歌）开发者博客 “Get your apps-ready for 64-bit” 于 2019 年 1 月就此话题展开了进一步的讨论，详尽介绍了谷歌要如何将安卓生态迁移至 64 位的整体计划，包括重要时间点，对应程序（库）版本，及相关支持:

“*Starting **August 1, 2019 （2019 年 8 月 1 日起）**:*

- *All new apps and app updates that include native code are required to provide 64-bit versions in addition to 32-bit versions when publishing to Google Play.* 新上架的原生应用和应用更新需在 32 位构建基础上提交 64 位构建。
- *Extension: Google Play will continue to accept 32-bit only updates to existing games that use Unity 5.6 or older until August 2021.* 使用 Unity 5.6（或以下版本）开发的 32 位游戏，豁免期延长至 2021 年 8 月（即额外两年）。

*Starting **August 1, 2021 （2021 年 8 月 1 日起）**:*

- *Google Play will stop serving apps without 64-bit versions on 64-bit capable devices, meaning they will no longer be available in the Play Store on those devices.* Google Play 将不再向 64 位兼容设备提供不含 64 位构建的应用。
- *This will include games built with Unity 5.6 or older.*” 使用 Unity 5.6（或以下版本）开发的游戏不再享受豁免。

谷歌的再次郑重声明不仅非常值得一读，而且也包含了关于 64 位移植的（具体）建议与工具（介绍）- **阅读原文**

关于安卓生态的思考 Android Ecosystem Considerations

64 位计算的技术基础以及相关能力建设，多年前就已相继就绪，而今谷歌对开发者即刻开始 64 位迁移的建议驱动着整个安卓生态系统不断向前。许多兼顾 iOS 和安卓的跨平台开发者，受益于早先开发 64 位 iOS 应用获取的经验，“在需要的时候（安卓）64 位移植工作将会进行得（相对更为）简单平顺” 。

对于很多应用来说，创建所必须的 64 位程序库并非难事，因为许多的开源库已经是类型安全的并经历了多年的反复考验。大部分使用 Arm NEON 指令的代码可以无需修改就能通过 64 位编译。编译器领域的长足进步，通常能自动生成更加出色和性能更好的（可执行）编码。今天，如果一个安卓应用是完全由 Java 语言写成的，那么目前的安卓运行时（ART）就可以保证其无需调整便能完美运行（于 64 位平台上）。需要特别注意的是，如果你的 Java 程序中使用了任何原生程序库或者原生 SDK（以及 NDK），都必须正确地包含相应模块的 64 位版本（而非 32 位）以便成功编译。以 64 位原生程序的移植为主题，Arm 为开发者准备了一系列的博客和文档 - **更多阅读**

其他的安卓生态系统 Additional Android Ecosystems

某种意义上，就谷歌究竟能在多大程度上真正影响亚洲地区重量级应用商店生态的发展，存在不同声音，考虑到其在中国（大陆地区）甚至未能提供 Play 商店服务。因而谷歌的 64 位迁移计划，在这些显然不容忽视的市场里，也许从某些观点看来，并非那么不可或缺。然而，Arm 通过与这些（自成一体的）安卓生态中，那些最具影响力成员们的接触与交流，确信他们都将拥抱 64 位指令集生态。与此同时，谷歌也公布了在 Google Play 应用商店提交应用，所需的全新安卓应用程序接口（API）等级，并且这一举措获得了来自中国的应用商店的积极配合与支持 - **更多阅读**

64 位（计算）的技术本质与裨益 Technical Rationale and Benefits of 64-bit

64 位安卓系统运行时比起它的 32 位对应副本的确需要更多的内存空间，这就意味着系统的最低内存要求有所上升，具体幅度也取决于设备的屏幕分辨率和像素密度等因素。一个 32 位版本的安卓系统通常只需要大约 1 千兆字节（GB）的内存就能启动，而与此同时，在 64 位 CPU 上运行时的最低内存容量要求可能会相应上升到 1.8 甚至 2 千兆字节。

最后一版仅支持 32 位 CPU 的安卓系统 “KitKat” 发布于 2013 年，以 2014 年诞生的 “Lollipop” 为起点，安卓系统开始同时支持 32 位和 64 位应用程序，这么说也许至少隐瞒了安卓系统（在兼容性方面）的部分潜力。受益于此，Arm 得以在 Armv8 系列 64 位处理器架构上完美向下兼容所有现存的 32 位安卓应用程序。

对于安卓设备而言，有太多理由转向 64 位代码体系，例如：

- Enhanced security; 增强的安全性
- Better performance; 更高的性能
- More and larger registers; 数量更多且字长更宽的寄存器
- Greater precision in 64-bit numbers; 更高精度表现的 64 位数据操作
- A richer instruction set, and 愈加丰富的指令集，以及
- The advantage of new and modern features not available with 32-bit code. 其他 32 位代码无法利用的先进特性

64 位编码中包含大量位宽更高的指针变量，因而导致了应用程序的可执行二进制文件大小增长，当然用户同时也获得了访问更大范围内存地址空间的能力，进而能够操作更大的数据块以更快的速度载入文件到内存中。配合经过优化的代码实现，这一特性允许处理器事实上以更低的能耗，更快的速度来完成（全部）操作，从而整体上节约电力。

纯 64 位设备 64-bit Only Devices

伴随着安卓生态系统中 64 位应用数量的日渐庞大，在不断降低设备元器件成本的市场驱动力作用下，终有一天安卓平台上会诞生一大批的纯 64 位设备。对开发者而言，这意味着更低的研发成本（无需兼顾 32 位与 64 位），与此外各类纯 64 位设备所能带来的种种益处，包括更低的（硬件）复杂度，针对单一架构的深度优化，都无疑有助于一个更加强壮，健康，以及鲁棒的系统得以问世。

这里列出的，是一些力挺纯 64 位指令集安卓设备的观点：

- Additional AArch32 support significantly affects micro-architectural complexity. - 额外的 32 位支持导致（处理器）微架构复杂度明显提升
- Further complexity leads to added validation costs for testing for OEM partners. - 由复杂性导致的最终分摊至 OEM 厂商的额外系统测试及验证成本
- Armv8.x security features for safe authentication, data protection, exploit mitigation, and secure content delivery are only available in AArch64. - Armv8.x 的安全特性包括 安全验证，数据保护，利用缓解，以及安全内容交付等，都只能在 AArch64 架构上实现
- Greater performance is available for emerging mobile workloads such as the mixed reality (AR and VR), AI/ML, and web applications. – 在不断涌现和演化的移动应用场景（如混合现实，人工智能，机器学习，和网络应用）中具备更好的性能表现
- Supporting one runtime requires less testing and maintenance. – 单一运行时意味着更少的测试和维护工作量
- AArch64-only CPUs will be more efficient. – 仅支持 AArch64 架构的 CPU 整体上更为高效

如今的安卓智能手机同时支持 32 位与 64 位应用，这帮助我们实现了对过往安卓设备（和应用）的向下兼容性，从而确保了最为广泛的潜在受众群体。然而，与之相伴的成本和劣势正在不断累积，下面是一份（负面）特性的清单:

- Support of multiple ISAs requires pre-decoding in the Instruction Cache. –（同时）支持多个指令集导致必须在指令缓存中进行预解码
- Increased Flash overhead for Multilib APK builds. – 保持多个版本安卓应用安装包（APK）导致的闪存空间浪费
- 100+MB dual Android Zygote RAM overhead. – 双份的安卓“接合子”管理器实现所消耗的额外 100 多兆内存
- Larger security attack surfaces. – 更复杂的系统安全环境意味着更容易被攻击
- Additional validation and cost for OEMs. – OEM 厂商所面临的更高的验证成本
- Decoding older T32 code multiplies decoder area by two. – 兼容 T32 代码导致解码器占用的（晶圆）面积翻倍
- Decoding Thumb16 is the critical timing path of an app’s decoding stage. – Thumb16（某旧指令集）的解码通常构成应用代码解码阶段的关键路径（性能短板）
- Aliasing of NEON™ registers in AArch32 requires double tagging for register renaming. – 在 AArch32 架构体系上进行 NEON™寄存器的重命名需要双倍的标注量
- 30 percent of the NEON renaming table area – accessed on every cycle on NEON / FP execution. – 导致 NEON 重命名表（NEON / FP 执行期间的每个时钟周期都会访问）占用 30%的额外面积
- 30 percent increase of source operand storage and forwarding network in NEON Issue Queues. – 导致 NEON 的（指令）发射队列需要 30%的额外源操作数存储空间与传送网络
- 50 percent of the power in forwarding logic and register matching logic of the NEON Issue Queues – IQs represent a third of the dynamic power of a Core. – NEON（指令）发射队列中的传送逻辑与寄存器匹配逻辑（实现）需要消耗额外 50%的电力 – 指令发射队列是单个（处理器）核心中三分之一动态功耗的源头

Arm 立足当下开拓进取 What is Arm doing?

Arm 正与合作伙伴们通力协作，以期尽可能地洞察是否在生态，技术，以及商务层面上存在任何不利于 64 位移植的（潜在）关键议题。当然，我们从来不认为在这一进程中，有任何必经的问题或意外。迄今为止，Arm 已经从安卓应用开发者，游戏引擎开发者，芯片制造商，与操作系统供应商等处，获取了大量的积极建议。我们将持续密切关注安卓生态的发展，权衡 64 位应用的可用比例（与权重），从而保证这一决定性的变革在正确的方向上不断前进。

Arm 一直以来都在与游戏引擎开发者们紧密合作，以保证各个游戏工作室能在 2019 年 8 月前，有充裕的时间构建，测试，并发布（64 位的）安卓游戏产品。我们坚信游戏引擎开发者们始终引领着（技术）创新，并推动着整个安卓生态系统的性能包线延申至新的疆域。目前，正在进行 64 位移植的引擎合作伙伴，凭借尚未完全优化的初期产品形态就已观测到了至少 10%的性能提升。

Arm 始终专注于构建由一些列性能最优的产品组合而成的高能效套装。我们目前致力于，为那些在安卓平台上被广泛应用的关键程序库，运行时，浏览器，和引擎，引入更多的（软件）优化。在可以遇见的未来，通过向安卓应用开发者们提供更多支持，必将令整个安卓生态获益匪浅。

安卓应用的未来 What is next for my Android application?

如果应用原先是（完全）由 Java™语言开发的, 那么开发者无须（为 64 位）采取任何行动，所有非原生程序均不受影响。 倘若应用是基于原生安卓程序库，那么也许需要一点额外的工作，取决于原先程序的书写方式。即使只是（为 64 位平台）重编译一份完美编码的 32 位程序，也或多或少需要修改其中的一部分。因此，对于游戏开发者（特别是有自研引擎项目或代码涉及底层硬件的），及早验证以防在整个生态加速跃迁时遭遇困难，显得尤为重要。

Arm 坚信这场以 64 位体系架构为核心的生态升级，完全符合原生应用开发者们的最佳利益。我们已经做好准备向广大开发者社区提供支持与指导。Arm 倾注于 64 位处理器架构的未来，今后相关的性能提升将只能够在 64 位的系统环境中达成。作为这场变革的一部分，我们鼓励应用开发者立即检视产品是否需要为 64 位迁移做任何额外的工作！

更多的资源 Additional Resources

如果开发者正在搜寻关于 64 位移植的信息，Arm 推荐访问以下的在线资源（英文），并愿尽可能提供更多帮助。

Armv8-A programming:
Armv8-A Programmer's Guide
Contained within this link are presentations and tutorials from events since the introduction of AArch64. New and updated materials will be posted as they become available.
Arm architecture tutorials for app developers

Armv8 Architecture Reference Manual
developer.arm.com

Android programming article:
codeproject.com/Making-sure-your-Android-game-is-ready-for-64bit

Specific Arm Cortex-A processor guides:
Arm Cortex-A Series Processors

Arm Technology Overview:
Arm Technologies Overview

Suggested Google Android resources:
Android Developer Site
Android Source Code - 64-bit Builds
Android Developer NDK Guides - 64-bit ABIs

For the latest Android information from Google:
Android Developers Blog
Updated information on the Google Play 64-bit requirement is contained in the January 15, 2019 blog.