International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

AOSP Based Custom ROM for OnePlus Nord

Akshay Kakatkar[1], Kunal Mane[2], Shreyash Kale[3], Dr. D. K. Shedge[4]

[1,2]Electronics and Telecommunication, AISSMS Institute of Information Technology, Maharashtra, India
[3,4]Professor, AISSMS Institute of Information Technology, Maharashtra, India

Abstract

In our project, we have developed a Custom ROM Based on AOSP for OnePlus Nord device. Our project can be directly flashed on OnePlus Nord without any fatal issues. We are using AOSP Source Code provided by Google to build the AOSP ROM for OnePlus Nord. The kernel source is based on the kernel source provided by CodeLinaro (Qualcomm) for their chips based on Linux Kernel 4.19 (msm-4.19) and includes the proprietary drivers released by OnePlus with GPL Licensing for our device.

During our research of similar available products, we found out that there are multiple AOSP Based Custom ROM Projects available for multiple devices. But there was no AOSP Based Custom ROM available for the OnePlus Nord. Hence, we have built AOSP Based Custom ROM for the OnePlus Nord. The latest Android Version available for the OnePlus Nord by OnePlus is Android 12 whereas the latest version available by Google is Android 13 with May 2023 security patch. Which is also the version that we have built for OnePlus Nord.

The objectives of this project are to provide stable and secure software for OnePlus Nord that will last even after the software update cycle from the OEM is over. This includes new Android versions, security patches, and kernel upstream.

Keywords: AOSP, CustomROM, OnePlusNord, avicii.

1. Introduction

The mobile phone industry is one of the biggest industries in the modern world. Android based devices have the highest market share in this industry with over 71% of the devices using the Android operating system.

However, the life cycle of an Android device when compared to an iOS device is very short due to the average software updates life cycle of 2 years compared to 5 years for iOS. This results in the device becoming obsolete much earlier than it can last as the fatal security patches are no longer provided to the device so it becomes a security risk to continue using it.

Our project's purpose is to develop an AOSP based Custom ROM for the OnePlus Nord that can last even after the OEM software update life cycle is over and make it possible for users to get the latest security patches from Google.

The objectives of this work are:

- To build stable software for the device.
- Include the latest security patches from Google.
- Include the latest kernel upstream from CodeLinaro.

- Include the latest kernel security patches from ASB.

The resulting ROM build contains the following main components:

- Linux Kernel.
- Kernel Drivers.
- Hardware Abstraction Layers.
- Android Runtimes.
- Native Libraries.
- Java Frameworks.
- System apps.
- Different configuration files.

2. Literature Survey

● Chirag Kanthed and Yagyapal Yadav [1] discuss strategies for reducing RAM usage in Custom ROMs. They explore methods to increase RAM availability and develop delay-free interfaces. Their work likely involves optimizing resource utilization, improving memory management algorithms, and implementing efficient performance tuning techniques to minimize RAM footprint.

● Shreyas S, T Tejo Veena Sri, Poshith N, Shesha Sai Charan R, Ravi Kumar V [2] present their research on developing a Custom ROM specifically tailored for the Mi A1 device. They outline the benefits of using their Custom ROM and provide a detailed explanation of the installation process. Their work likely includes customizing the ROM to enhance device-specific features, optimizing performance, and ensuring compatibility with Mi A1 hardware.

● Kumar Vimal & Aditya Trevedi [3] propose a memory management schematic that aids in determining which applications can be eliminated from memory to optimize resource utilization. They also focus on decreasing response times, potentially by implementing efficient memory allocation algorithms, task scheduling techniques, and caching mechanisms.

● Saurabh Manjrekar and Ramesh Bhati [4] provide insights into the process of installing Custom ROMs, including the pros and cons associated with such customization. They also discuss rooting methods, which allow users to gain full control over their device's system. Their work likely includes explaining the installation steps, highlighting the risks and benefits of Custom ROMs, and offering guidance on rooting procedures.

● Parikshit Rajput, Vinay Koraganti, Biswajeet Champaty [5] compare Custom ROMs to Stock ROMs specifically for Lenovo A7000 devices. They discuss the advantages of using Custom ROMs, which may include improved performance, additional customization options, and access to the latest Android updates. Their work likely involves evaluating the specific benefits of Custom ROMs for the Lenovo A7000, addressing any potential drawbacks, and providing insights into the decision-making process for device users.

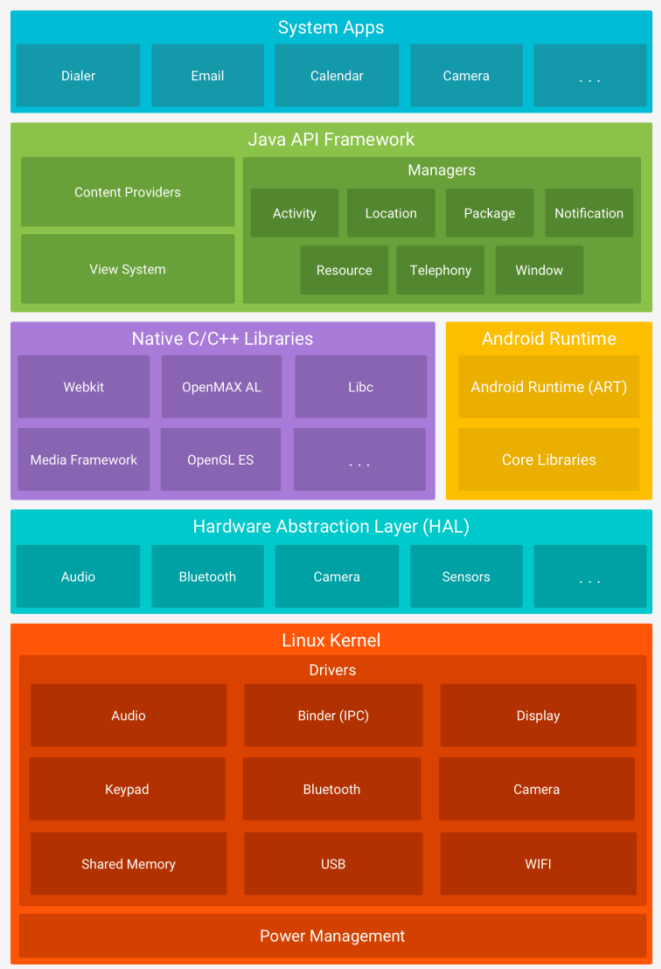
Furthermore, the literature survey mentions certain challenges related to outdated research papers that may not cover the installation or optimization process for newer devices using dynamic partitioning or A/B partitioning styles. It also highlights that recoveries listed in earlier research papers are now deprecated, suggesting the use of recoveries compiled while building the Custom ROM itself. The survey further suggests referencing Google's basic build guide for Pixel devices, which provides

instructions for setting up a Linux Build environment for building Android. However, it notes that Google does not provide instructions for constructing the necessary repositories required to build AOSP Based Custom ROMs for non-Google devices. In such cases, the public repositories of the latest Pixel devices can serve as a reference for creating device-specific repositories.

Additionally, the survey mentions the availability of AOSP Based Custom ROMs for multiple devices, with public device repositories that can be used as references for building one's own Custom ROM. Notable organizations specializing in AOSP Based Custom ROMs, such as LineageOS, Paranoid Android, and PixelExperience.

3. Architecture

Figure 1: Android Architecture



Android Architecture is split in different parts as shown in Figure 1.

1. **Linux Kernel:-** This is the most basic foundation for Android Platform which includes different drivers for the hardware used in the device as well as different drivers needed by the Platform. The Linux Kernel serves as the basis for the Android Platform because the Android Runtime (ART) depends on it for essential functions like threading and low-level memory management.

1. **Hardware Abstraction Layer (HAL):-** The Hardware Abstraction Layer offers shared interfaces lower level Java API Frameworks that expose various hardware features like Audio,

Bluetooth, Camera, Sensors, etc. The Hardware Abstraction Layer is made up of various library modules, each of which implements an interface for a particular kind of hardware part, such as a Bluetooth module or camera. The Android system loads the library module for that hardware component when a framework API calls to access device hardware.

2. **Android Runtimes:-** Android Runtime is a managed Runtime used by applications and system services on Android, ART includes processor dalvik, Zygote as well as core libraries. By using different Dalvik and Zygote the system performance can differ.

3. **Native C/C++ Libraries:-** Many Android features and services are created using native code (ART, HAL, etc.), which necessitates the use of C/C++ native libraries. The Android platform offers Java framework APIs so that apps can access the functionality of these native libraries as needed. For instance, OpenGL ES can be accessible through the Java OpenGL API Framework to give the programme capability for rendering and manipulating 2D and 3D graphics.

4. **Java API Framework:-** The full feature set of the Android Platform is accessible to users via Java-written APIs. By making it easier to reuse fundamental, modular system components and services, such as the ones listed below, these APIs can be utilized as building blocks to develop Android apps.

   a. View System:- A user interface for an app, including lists, grids, text fields, buttons, and even an embeddable web browser, can be created using the View System.

   b. Resource Manager:- Access to non-code resources such localized text, images, and layout files is made possible by Resource Manager.

   c. Activity Manager:- The entire application lifecycle and activity stack are under the control of the activity manager.

   d. Content Providers:- Applications can publish and share data with other applications thanks to content providers.

   e. Notification Manager:- Applications can show alerts and notifications to users with the help of the notification manager.

5. **System Applications:-** System apps are applications that come with the Android operating system and provide services like email, SMS messaging, calendars, internet browsing, and so on. The platform's apps have no unique status among the apps installed by the user. As a result, third-party apps might become the user's default apps for functions such as Web browser, SMS Messaging, keyboard, and so on (with some exclusions, such as the Settings app). The system apps serve as apps for consumers, providing crucial functionality that developers can use from their own apps. e.g. If you wish to send an SMS message, you don't have to implement the capability yourself; instead, you can use any SMS app the user has loaded.

4. Methodology

To build AOSP Based Custom ROM for OnePlus Nord we require different code repositories. We first sync the AOSP Source from Android GIT and then place our repositories in the correct path. The following are the repositories that we do not use directly from AOSP or have our own changes over AOSP.

1. **device_oneplus_avicii** :- This is the repository brought from scratch and consists of device specific configurations that are needed to build and boot AOSP based Custom ROM on the device. This repository also consists of the necessary Makefiles that are needed to call this specific device product in

order to start the build. In this repository, we have the Build Specifications of the OnePlus Nord (avicii) along with its different config files.

2. **manifest :-** The initial repository of AOSP project, it contains the list of repositories tracked from android.googlesource.com and additional repositories tracked from our own organization. This repository is used to download the source code of AOSP. Since there are more than 1000 repositories that are required for building android, rather than cloning each repository individually, the 'repo' command enables us to sync these repositories in one go. The 'repo' package finds a default.xml on the given remote in order to start syncing the repositories, this default.xml can track other XMLs for more additional repositories that may be required for device specific configurations.
3. **kernel_oneplus_sm7250 :-** This repository consists of the Linux kernel source code that is used to boot the avicii device. The kernel source code is forked and merged with the Qualcomm Code Aurora Forum (CAF) Tag on top of it, along with the needed changes and extra drivers that are needed to boot the 'avicii' device.
4. **vendor_oneplus_avicii :-** This repository consists of the proprietary binaries that are provided by Stock OEM with no source code, so they are copied directly after cleaning up the unneeded libraries, these unneeded repositories consists of the unused frameworks and libraries that are either left unused or are not required by the Android userspace.
5. **frameworks :-** It consists of the application framework, written in C++ and java. It contains all of the system services and core implementation of Android.
6. **bionic :-** This repository consists of Android's C library like libc, dynamic linker and math.
7. **art :-** This repository contains the Android Runtime related packages and libraries.
8. **build :-** This repository contains the build related makefiles that can be inherited. It also includes the scripts to be run during building and signing of the builds.

AOSP Based Custom ROM compiled on Linux PC/Servers. The build environment is set up using instructions provided on Google Developers guide. The code is synced from Android GIT using the "repo" tool provided by google and then using the "lunch" command to set up the devices. Then "make" command is used to make a recovery flashable .zip file.

In order to install the Custom ROM on OnePlus Nord we follow the steps as given below.

1. Turn on Developer Options in Settings by tapping on OxygenOS Version 5-7 times in the "About Phone" menu in the Settings app.
2. Go to Settings -> System -> Developer Options, Turn on USB Debugging and Allow OEM Unlocking.
3. Reboot the device to bootloader mode using "fastboot reboot bootloader" or by holding Vol Up and Vol Down keys while powering on the device.
4. Unlock the bootloader using "fastboot oem unlock". This unlocks the bootloader so that builds not signed via OnePlus proprietary signing keys can boot.
5. Note that this will format all data on the device and may void the warranty for some OEMs (It does not in case of the OnePlus Nord). Some OEMs may also lose their Widevine L1 certification as do OnePlus Nord and many other OnePlus devices.
6. Flash dtbo.img and vbmeta.img from out directory of the Custom ROM in order to allow the recovery to boot. Use the below commands for that.

"fastboot flash dtbo dtbo.img"
"fastboot flash vbmeta vbmeta.img"
This flashes the dtbo and vbmeta into the respective partitions.

7. Flash the recovery using "fastboot flash recovery recovery.img". This flashes the recovery into the recovery partition which can be used to install the Custom ROM.
8. Click "Apply Update" and then Click on "Apply Update via ADB". This puts the recovery into sideload mode in order to transfer and flash the custom ROM onto the device. Figure 2, Figure 3 include screenshots of the recovery and sideload page respectively.

Figure 2: Recovery

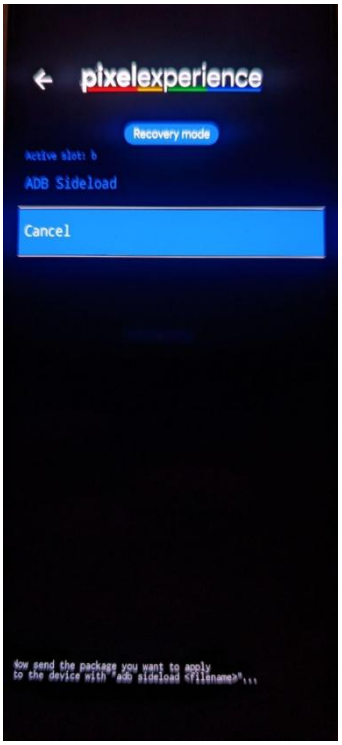
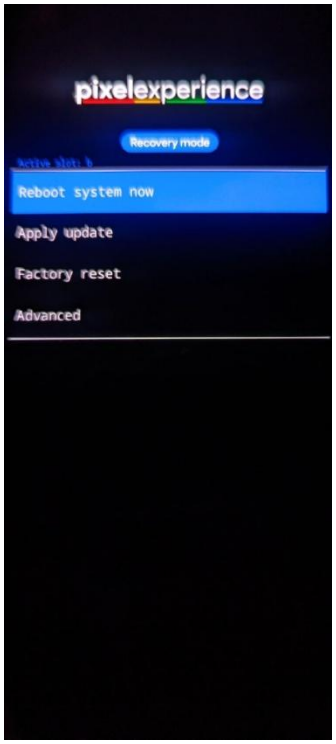


Figure 3: Sideload



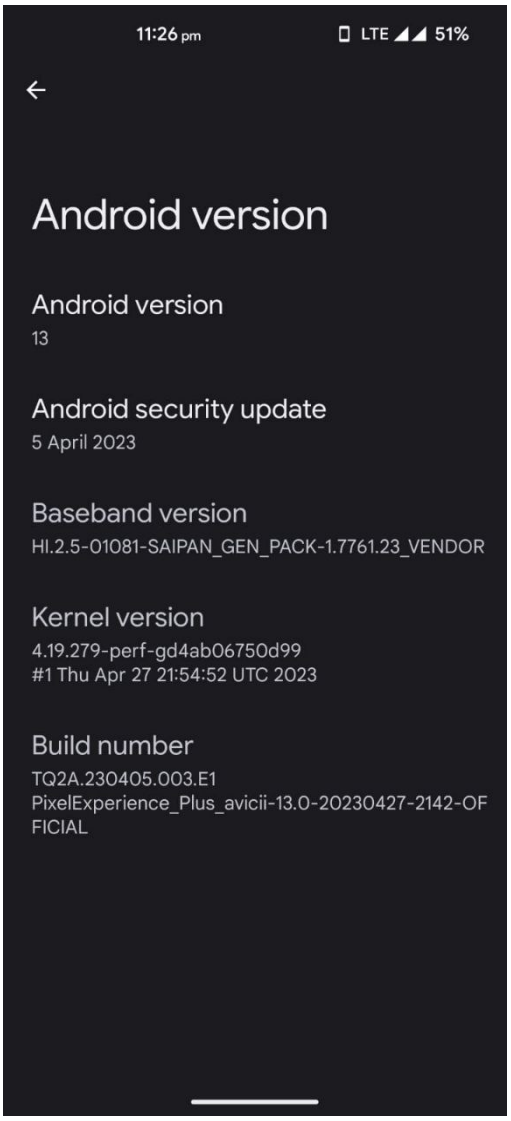
9. After the recovery enters sideload mode you can transfer and flash the Custom ROM from the PC using the command "adb sideload ROM.zip". Here replace ROM.zip with the correct zip file of the Custom ROM file.
10. After the Flashing Process is done, go back to the recovery home page and select "Wipe Data/Factory Reset" in order to erase the userdata partition.
11. Reboot device and it will successfully boot into AOSP Based Custom ROM.

Note that the above are the installation instructions specific to OnePlus Nord. It may differ for other devices.

5. Results and Discussion

AOSP Based Custom ROM once installed on our OnePlus Nord (avicii) via the AOSP Recovery is completely stable and does not have any noticeable bugs or glitches. Below is a screenshot of the "About Device" screen of OnePlus Nord running AOSP Based Custom ROM.

Figure 4: About Device



After installing Custom ROM we have noticed 4-10% improvement in the performance of the device in benchmarks for both CPU and GPU compared to OxygenOS. The overall stability of the device is as good as or better than that of OxygenOS. However, we have noticed that on OnePlus Nord after installing Custom ROM the device loses its Widevine Level 1 certification.

Below is a table comparing the performance of OxygenOS and Custom ROM. We have attached the proofs of the benchmarks in the appendix.

Table 1: Benchmark Results

Benchmark	OxygenOS	Custom ROM	Improvement
Geekbench CPU	763	811	6%

(Single Core)			
Geekbench CPU (Multi Core)	1857	1929	4%
Geekbench GPU (OpenCL)	1009	1096	9%
Geekbench GPU (Vulkan)	1016	1121	10%

Here, we used the benchmarking tools provided by Primate Labs Inc. which are publicly available on Google Play store in order to measure the performance of both OxygenOS provided by OnePlus and AOSP Based Custom ROM built by us. The Single Core and Multi Core scores are benchmarks measuring CPU performance while the compute benchmark is used to measure the graphics performance on OpenCL and Vulkan APIs. The Geekbench ML app is used to measure the Machine Learning performance of the device.

All of these tools are publicly available on Google Play store and can be used by anyone to measure the performance of their device and compare how it ranks compared to other devices.

6. Conclusion

In conclusion, we have developed a stable and performant AOSP Based Custom ROM for the OnePlus Nord device. The ROM is based on the latest Android Security patches and incorporates the most recent version of the Linux Kernel. This ensures that users can enjoy a secure and seamless user experience on their devices. The ROM has been validated through benchmarking tests, which show a noticeable improvement in performance when compared to the default OxygenOS provided by OnePlus.

The ROM was developed by leveraging our own meticulously constructed code repositories. The platform source code was synchronized from the Android GIT repository using the industry-standard "repo" tool provided by Google. This ensured that the ROM is based on the latest Android Security patches, offering enhanced security and protection for the device.

The ROM also incorporates the most recent version of the Linux Kernel, which brings various improvements and optimizations to the device's overall performance and stability. By integrating the latest kernel version, we aim to deliver an enhanced user experience and ensure compatibility with the latest hardware and software technologies available.

The performance of the ROM was validated through benchmarking tests using renowned tools such as Geekbench. The results clearly indicate a noticeable improvement in performance when compared to the default OxygenOS provided by OnePlus. For instance, in the CPU benchmark, our Custom ROM achieved a 6% performance improvement in Single Core performance and a 4% improvement in Multi Core performance. Similarly, in the compute benchmarks utilizing OpenCL and Vulkan APIs, our Custom ROM showcased respective performance gains of 9% and 10% compared to OxygenOS.

Overall, we are confident that our AOSP Based Custom ROM for the OnePlus Nord device is a reliable and performant option for users who are looking for a secure and seamless user experience.

7. Future Scope

As part of our research project, we have identified several areas for future exploration and development. One potential direction for this project is to expand the custom ROM to include further optimizations of the kernel. The kernel is the core component of the Android operating system, and optimizing it can lead to improved performance and stability.

Additionally, we believe that our custom ROM has the potential to be made more generic and compatible with multiple devices. This would require extensive testing and modifications to ensure that the ROM can work seamlessly across a range of devices. However, by making our custom ROM more generic, we can broaden its potential user base and contribute to the open-source community by providing a more versatile and adaptable ROM.

Moreover, during our research project, we attempted to create a custom ROM that is compatible with both the OnePlus Nord (avicii) and the OnePlus Nord CE (ebba). While we received some initial success with the kernel, we were unable to perform further testing on the OnePlus Nord CE device as we do not own it. However, this area presents an exciting opportunity for future research to develop a custom ROM that can work on multiple devices and offer a more versatile and flexible solution.

8. Appendix

Figure 5: Geekbench CPU (OxygenOS)

Figure 6: Geekbench CPU(Custom ROM)

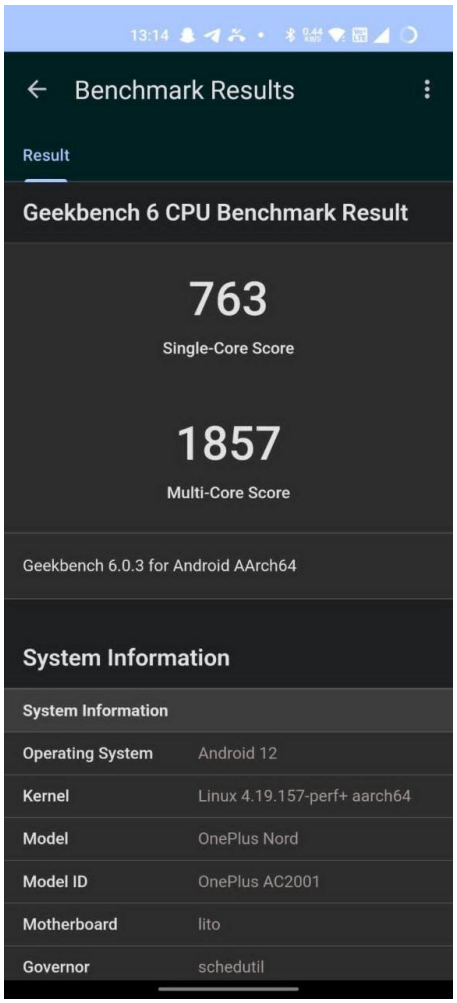


Figure 7: OpenCL (OxygenOS)

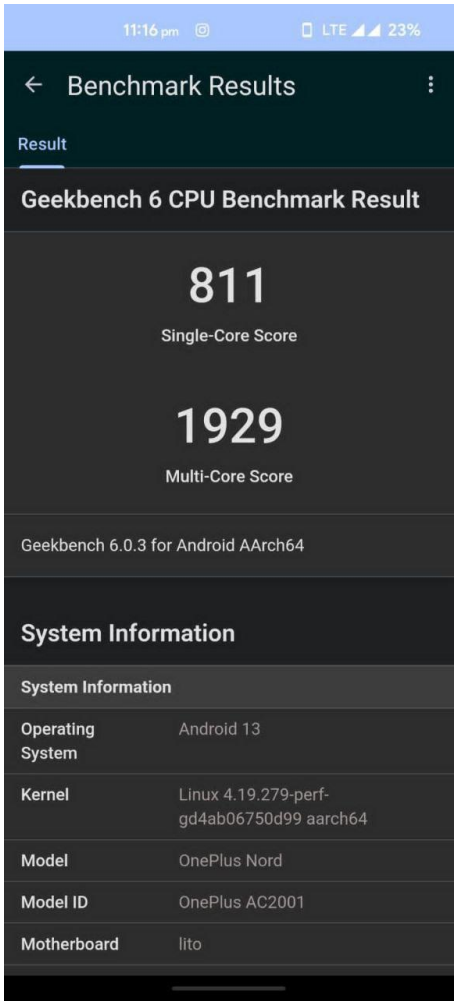


Figure 8: OpenCL (Custom ROM)

Figure 9: Vulkan (OxygenOS)

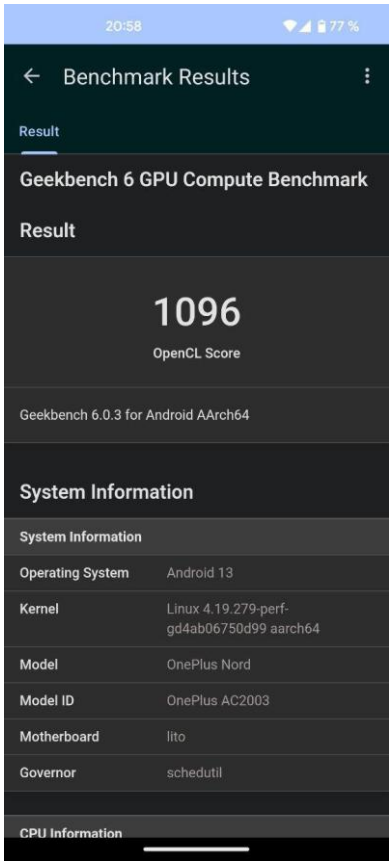
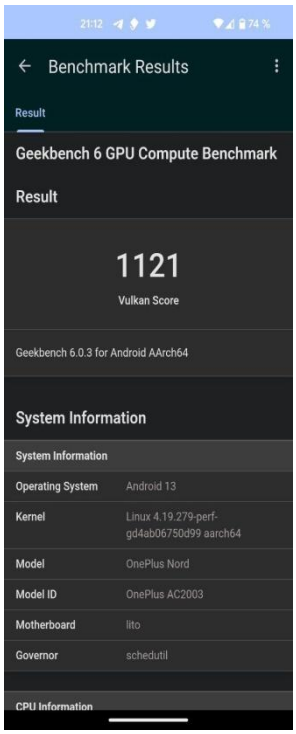


Figure 10: Vulkan (Custom ROM)



9. References

1. Chirag Kanthed, Yagyapal Yadav, “Building Custom ROM using AOSP and Improving RAM usage in it”, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, October 2017, 2 (5), 145-149.
2. S. Shreyas, “Developing Custom ROM based on Android using AOSP”, International Journal for Research in Applied Science and Engineering Technology, August 2020, 8 (8), 709-714.
3. K. Vimal and A. Trivedi, "A memory management scheme for enhancing performance of applications on Android," IEEE Recent Advances in Intelligent Computational Systems (RAICS), December 2015, 162-166.
4. Saurabh Manjrekar, Ramesh Bhati, “Custom ROM- A Prominent Aspects of Android”, International Journal of Advanced Research in Computer Engineering and Technology, 2012.
5. Parikshit Rajput, Vinay Koraganti, Biswajeet Champaty, “Custom ROM”, International Journal of Advances in Science Engineering and Technology, October 2018, 6 (4), 140-143.