**KOCAELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**BİTİRME TEZİ**

**Mobil Cihazlar için Test Uygulamaları**

**CENK CAMKIRAN VE FERHAT ÇAKIR**

**5,2 cm**

**ÖNSÖZ ve TEŞEKKÜR**

Benchmark yazılımları, her bir cihazın ve genel sistemin seviyesini değerlendirmek ve tahmin etmek için yararlıdırlar, bu nedenle daha yüksek performans elde etmek için donanım seçmemize veya sistem değişkenlerini ayarlamamıza yardımcı olabilirler. Platform ortak ve aynı yazılım yeteneğine sahip olsa bile, gerçekteki performansı, donanım ve diğer yazılım bileşenlerine göre değişir. Bu yüzden bütün donanım ve yazılım geliştiricileri daha yüksek performans elde etmek için büyük çabalar harcar. İstenen ürünü elde edebilmek için geliştiriciler ürünün performansını en ince ayrıntısına kadar incelerler. Böylece ürünü doğrularlar ya da zayıf noktaları varsa bunları tespit edip güçlendirirler.

Benchmark yazılımları konusunda bize çalışma fırsatı veren değerli hocamıza teşekkür ederiz. Bizi destekleyen arkadaşlarımıza ve danışman hocamıza sonsuz minnet duygularımızı sunarız.

Mayıs – 2019 Cenk Camkıran – Ferhat Çakır

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR**1**

İÇİNDEKİLER**3**

ŞEKİLLER DİZİNİ**5**

ÖZET**7**

ABSTRACT**8**

GİRİŞ**9**

1. GENEL BİLGİLER**10**

1.1. ÇALIŞMANIN AMACI10

1.2. İLGİLİ ÇALIŞMALAR10

1.3. ANDROİD İŞLETİM SİSTEMİ11

1.4. ANDROİD MİMARİSİ 12

1.5. ANDROİD İŞLETİM SİSTEMİNİN TARİHÇESİ13

2. PERFORMANS ANALİZİ YAPAN ARAÇLAR**17**

2.1. ANDROİD SDK ARAÇLARI17

2.2. BENCHMARK17

3. BENCHMARK KRİTERLERİ**21**

3.1. CPU21

3.2. GPU24

3.3. MİKRO-ÖLÇÜT TEKNİKLERİ ÜZERİNDE YAPILAN BAZI ÇALIŞMALAR25

3.3.1. Geometry İnstancing with Skinned Animation25

3.3.2. Screen-Space Ambient Occlusion25

3.3.3. Tiled Deferred Shading26

3.3.4. Occlusion Culling27

3.3.5. Shadow MappingAnimation27

3.3.6. Order-Independent TransparencyAnimation28

3.3.7. Cube Mapping and Planar Reflection Animation29

3.3.8. BVH Update and Visualizationtion29

3.3.9. Displacement Mapping with Tessellation Animation30

3.4. DEPOLAMA31

4. UYGULAMA TESTLERİ**34**

4.1. TEST CİHAZLARI34

4.2. SAMSUN GALAXY NOTE 4’ÜN ELDE ETTİĞİ SONUÇLAR35

4.3. SAMSUN GALAXY NOTE J3’ÜN ELDE ETTİĞİ SONUÇLAR38

4.4. SAMSUN GALAXY NOTE WİN’İN ELDE ETTİĞİ SONUÇLAR40

5. PROJE DETAYLARI**43**

5.1. CPU DETAYLARI43

5.1. GPU DETAYLARI50

5.1. RAM DETAYLARI53

5.1. DEPOLAMA DETAYLARI56

6. SONUÇLAR**59**

KAYNAKÇA**60**

ŞEKİLLER DİZİNİ

Şekil 1. Android Mimarisi **12**

Şekil 2. AnTuTu Benchmark **18**

Şekil 3. Android BaseMark **18**

Şekil 4. GFXBench **19**

Şekil 5. 3DMark SlingShot ES3.1 **19**

Şekil 6. 3DMark Giriş Ekranı **19**

Şekil 7. AndroBench **20**

Şekil 8. Geometry Instancing with Skinned Animation **25**

Şekil 9. Screen-Space Ambient Occlusion **26**

Şekil 10. Tiled Deferred Shading **27**

Şekil 11. Occlusion Culling **27**

Şekil 12. Shadow Mapping **28**

Şekil 13. Order-Independent Transparency **28**

Şekil 14. Cube Mapping and Planar Reflection **29**

Şekil 15. BVH Update and Visualization **30**

Şekil 16. Displacement Mapping with Tessellation **30**

Şekil 17. Mobile Storage Analyzer **31**

Şekil 18. Test Edilen Cihazlar GPU-Ekran **34**

Şekil 19. Test Edilen Cihazlar RAM-Depolama**34**

Şekil 20. Test Edilen Cihazlar CPU **36**

Şekil 21. Linpack Benchmark **36**

Şekil 22. JAVA OpenGL Benchmark **37**

Şekil 23. JAVA Draw Benchmark **37**

Şekil 24. MBytes/Second Transferring 4 Byte Words **38**

Şekil 25. Reading Speed in MBytes/Seconds **38**

Şekil 26. Heap Stack Allocation **39**

Şekil 27. MBytes/Second Cinsinden Okuma-Yazma Hızları **39**

Şekil 28. Android için Elde Edilen Sonuçlar **39**

Şekil 29. Requery için Elde Edilen Sonuçlar **39**

Şekil 30. Linpack Benchmark **40**

Şekil 31. JAVA OpenGL Benchmark **40**

Şekil 32. JAVA Draw Benchmark **40**

Şekil 33. MBytes/Second Transferring 4 Byte Words **41**

Şekil 34. Reading Speed in MBytes/Seconds **41**

Şekil 35. Heap Stack Allocation **41**

Şekil 36. MBytes/Second Cinsinden Okuma-Yazma Hızları **41**

Şekil 37. Android için Elde Edilen Sonuçlar **41**

Şekil 38. Requery için Elde Edilen Sonuçlar **41**

Şekil 39. Linpack Benchmark **42**

Şekil 40. JAVA OpenGL Benchmark **42**

Şekil 41. JAVA Draw Benchmark **42**

Şekil 42. MBytes/Second Transferring 4 Byte Words **43**

Şekil 43. Reading Speed in MBytes/Seconds **43**

Şekil 44. Heap Stack Allocation **43**

Şekil 45. MBytes/Second Cinsinden Okuma-Yazma Hızları **43**

Şekil 46. Android için Elde Edilen Sonuçlar **43**

Şekil 47. Requery için Elde Edilen Sonuçlar **43**

Şekil 48. CPU Örnek Çalışma-1 **49**

Şekil 49. CPU Örnek Çalışma-2 **50**

Şekil 50. GPU Örnek Çalışma-1 **52**

Şekil 51. GPU Örnek Çalışma-2 **53**

Şekil 52. RAM Örnek Çalışma-1 **55**

Şekil 53. RAM Örnek Çalışma-2 **56**

Şekil 54. Depolama Örnek Çalışma-1 **58**

Şekil 55. Depolama Örnek Çalışma-2 **59**

# ÖZET

Android, en yaygın mobil platformlardan birisidir. Birçok kişi Android mobil cihazları kullanıyor ve çoğu geliştirici şu anda Android uygulamaları geliştiriyor. Android platformu, akıllı mobil cihazlar ve dahil gömülü sistemler için yaygın olarak kullanıldığından, sistematik performans analizi için ihtiyaçlar önemli ölçüde artmıştır. Mobil cihazların önemli ölçüde evriminin ve mobil cihaz kullanımının artmasına rağmen, pek çok mobil kriter hala tam olarak araştırılmamıştır.

Mobil uygulamalar, masaüstü odaklı uygulamalarla benzer özellikler paylaşsa da, farklı programlama ortamları ve kullanıcı kullanımına göre farklı özellikler sunar. Ne zaman yeni bir telefon piyasaya girse ve çoğu zaman ondan önce bile, bu telefonun işlem gücünü gösteren kriterler sayısız benchmark’lerden geçiyor. İnsanlar gerçek hayatta nadiren telefonların performansı hakkında konuşurlar ve eğer yaparlarsa, bu konuşmalar ya kendileri kullanırken gördükleri hakkında, örneğin (“Bu telefon hızlı çalışıyor.” , “Ara sıra donuyor.” , “Uygulamalar hemen açılıyor.”) ya da YouTube'da gördükleri karşılaştırmalarda telefonların birbirleriyle kıyaslanmaları sonucunda edindikleri bilgiler hakkında oluyor. Bu konuşulan durumlar sistemin performansından kaynaklanıyor.

Sistem performansı genellikle benchmark yazılımları ile ölçülüyor. Bildiğimiz kadarıyla Google Play'de performans testi uygulamaları olarak Antutu Benchmark, GeekBench, GFXBench, BaseMark, AndroBench vb. mevcuttur. Bu yazıda, Android platformları için açık kaynak kodlu bir mobil multimedya benchmark yazılımı sunulmuştur. Anlatılan benchmark yazılımı, Android platformlarında çalışan çeşitli multimedya benchmark testlerinden (Micro-Benchmark) oluşuyor.

# 

# ABSTRACT

Android is one of the most common mobile platforms. Many people are using Android mobile devices, and most developers are currently creating Android apps. Since the Android platform is widely used for embedded systems, including smart mobile devices, the needs for systematic performance analysis have increased significantly. Despite the substantial evolution of mobile devices and increased mobile device use, many mobile benchmarks are still not fully explored.

Although mobile applications share similar features with desktop-oriented applications, they offer different features depending on different programming environments and user usage. When a new phone enters the market, and most of the time, even before that, it goes through countless tests for benchmarks showing the processing power of this phone. People rarely talk about the performance of phones in real life, and if they do, these conversations about what they see when they use themselves, for example (“This phone works fast”, “Occasionally freezes”, “Apps are opening immediately.”) or what they see on YouTube comparisons, it is about the information that the phones have been compared with each other. This is due to the performance of the system.

System performance is generally measured by benchmark softwares. As far as we know, the performance test applications on Google Play include Antutu Benchmark, GeekBench, GFXBench, BaseMark, AndroBench and so on. It is available for everyone. In this article, we present an open source mobile multimedia benchmark for Android platforms. Our benchmark consists of a variety of multimedia benchmark tests running on Android platforms.

# GİRİŞ

Akıllı telefonların popülaritesi büyük bilgisayar paradigmalarını değiştiriyor. Mobil platformlar, geleneksel masaüstü uygulamalarından farklı türde uygulamalara sahiptir. Bununla birlikte, ölçütlerin ve mimarilerin özelliklerini anlamak için çok ölçüt yoktur. Android, dünyanın en popüler mobil platformudur. Android mobil cihazlar çok sayıda kişi tarafından kullanılır ve günümüzde, birçok geliştirici Android uygulamaları geliştirmeye devam ediyor. Android'in yapısı Linux çekirdeğine dayanır ve üstünde Java framework vardır. Bu nedenle, Android uygulama geliştiricileri Java programlama dilini kullanarak kolayca Android uygulamaları geliştirebilirler. Android uygulamaları geliştirmek için, Google, bizlere birçok kütüphane ve kullanışlı araçlara sahip olan Android SDK’yı sağlıyor. Android açık kaynak kodlu bir yazılımdır ve geliştiricilere kendi uygulamalarını yapmak için ücretsiz bir platform sunduğundan, birçok donanım satıcısı Android'i tercih eder ve bu yüzden pazar payı da gittikçe artmaktadır.

MiBench [1] ve SPEC benchmark suite [2] gibi geleneksel benchmarklar, Android ve iOS gibi mobil cihaz platformlarıyla uyumluluk sorunu nedeniyle mobil cihazların performansını karakterize etmek için yeterli değildir. Bu yeni sistemlerin tümü, Java ve Objective-C gibi yeni nesne yönelimli program dillerinde yazılmış, pek çok diğer masaüstü uygulamasından çok farklı program özellikleri gösteren programları çalıştırıyor. Quadrant Standard Edition ve SmartBench, AnTuTu gibi genel performansı ölçebilen Android cihazlar için iyi bilinen benchmark uygulamalarıdır. CF-bench, GLBenchmark, Linpack, BenchmarkPI gibi diğer benchmark uygulamaları, örneğin CPU veya grafik alt sistemi gibi belirli bir sistem alanının performansını ölçmek için kullanılır.

Bu yazıda, sistemin performansını ölçmek için benchmark yazılımı kullanılmıştır. Benchmark yazılımları, her bir cihazın ve genel sistemin seviyesini değerlendirmek ve tahmin etmek için yararlıdırlar, bu nedenle daha yüksek performans elde etmek için donanım seçmemize veya sistem değişkenlerini ayarlamamıza yardımcı olabilirler. Benchmark yazılımı tanıtıldıktan sonra, basit birkaç performans analizi sonucu gösterilecektir. Bunlar, ilgili çalışmada kapsamlı performans analizi yazılımı geliştirmesi için bir temel olacaktır.

# GENEL BİLGİLER

Benchmark uygulamaları, her bir cihazın ve genel sistemin göreceli seviyesini değerlendirmek ve tahmin etmek için kullanılan araçlardır, bu sayede daha yüksek performans elde etmek için donanım seçmemize veya sistem değişkenlerini ayarlamamıza yardımcı olabilirler. Buna rağmen, hangi parçanın performansı etkilediğini veya daha iyi performans için hangi parçanın manipüle edilmesi gerektiğini belirtmek zordur. Detaylı yazılım performans analizi de benchmark içinde mevcut değildir.

# ÇALIŞMANIN AMACI

Çalışmanın amacı, Android İşletim Sistemi yüklü cihazın CPU, GPU ve depolamanın performansını ölçmek ve birden çok farklı cihaz arasındaki performanslarını görmek ve bunları etkileyen değişkenleri bulup analiz etmektir.

Bu makalede, geçmiş zamanlarda yazılmış, üzerinde çalışmalar yapılmış benchmark yazılımları araştırılıp, bunlardan ilham alınarak geliştirilen benchmark yazılımını sizlere sunmak istenmiştir.

Bu yazıda öncelikle, daha önce yapılmış çalışmalardan bazıları sizlere tanıtılacak, ardından benchmark yazılımındaki kriterler incelenecektir.

# İLGİLİ ÇALIŞMALAR

# AndroStep: Android Storage Performance Analysis Tool [32]:

Bu çalışmada, Android İşletim Sistemine sahip cihazlarda bulunan depolama aygıtları hakkında temel teorik bilgiler anlatılmış, depolama aygıtının hangi kriterler altında test edilmesi gerektiği ve birden fazla cihazdaki depolama aygıtlarının çeşitli kriterler altında yazma-okuma hızları test edilmiş ve ortaya çıkan sonuçlar karşılaştırılmıştır.

# The AM-Bench: An Android Multimedia Benchmark Suite [33]:

Bu çalışmada, Android mimarisinin genel yapısı hakkında bilgi verilmiş, Android İşletim Sistemi Yüklü cihazlarda bulunan CPU aygıtının hangi kriterler altında test edilmesi gerektiği anlatılmıştır. Birden fazla cihaz bu kriterler altında test edilmiş ve ortaya çıkan sonuçlar karşılaştırılmıştır.

# L-Bench: An Android benchmark set for low-power mobile GPUs [34]:

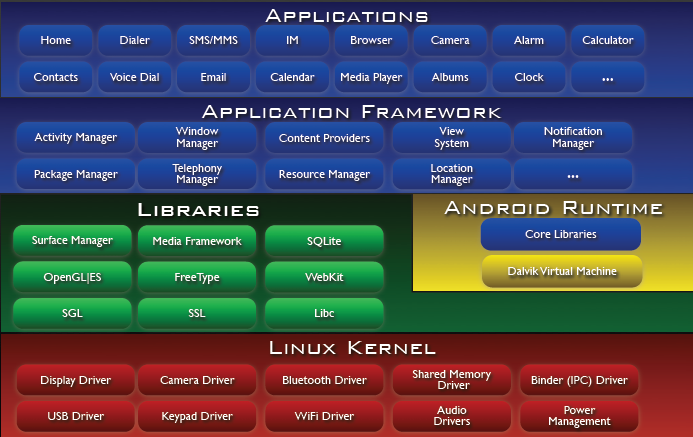
Bu çalışmada, Android İşletim Sistemi yüklü cihazlarda bulunan GPU aygıtının mimarisi hakkında kısa bilgi verilmiş, GPU’nun geçmişten günümüze olan gelişimi ve öneminden bahsedilmiş, piyasada çok kullanılan, GPU’yu test eden benchmark yazılımları sınıflandırılmış ve bu yazılımlar hakkında kısa bilgiler verilmiştir. Ve en sonda, cihazda bulunan GPU bileşeninin hangi kriterler altında ve GPU’nun hangi bileşenlerinin test edilmesi gerektiği anlatılmıştır. Kendi yaptıkları benchmark yazılımıyla farklı cihazların GPU performansları elde edilmiş ve grafik üzerinde elde edilen performans sonuçları karşılaştırılmıştır.

# ANDROİD İŞLETİM SİSTEMİ

Android işletim sistemi, OHA (Open Handset Alliance) [3] tarafından yönetilen, tüketiciler için daha iyi bir telefon oluşturmayı amaçlayan bir açık kaynak kodlu proje olarak geliştirildi ve sürdürülmeye devam etmektedir. Android İşletim Sistemi Linux tabanlı ve açık kaynak kodlu bir işletim sistemidir. 2008 yılında piyasaya sürüldüğünde birçok Android İşletim Sistemi Apache Free-Software ve Açık Kaynak Kodu lisansıyla geliştirilmeye açık hale gelmiştir.

Android İşletim Sisteminde ara katman yazılımı kütüphaneler ve API C diliyle yazılırken, uygulama yazılımları ise Harmony üzerine kurulu, Java uyumlu kütüphanelerden oluşan uygulama iskeleti üzerinden çalışır, derlenmiş Java kodunu çalıştırmak için dinamik çevirmeli (JIT) Dalvik Sanal Makinesi (DVM) kullanılır. Android İşletim Sistemi, Android uygulamaları geliştirmek isteyenlere de gerekli olan Android SDK hizmetini sunmakla beraber yeni çıkan versiyonlarıyla güncellemektedir.

* 1. **ANDROİD MİMARİSİ**



Şekil 1: Android Mimarisi [35]

Android İşletim Sistemi temel olarak 5 kısımdan oluşur.

1. **İşletim Sistemi Çekirdeği:** Android, Linux Çekirdeğine dayanır ve altta yatan donanımla iletişimi kurar. Bu katman Android’in bellek yönetimi programları, güvenlik ayarları, güç yönetimi yazılımı, birkaç sürücü, dosya sistemi erişimi, ağ iletişimi ve süreçler arası iletişimi içeriyor.
2. **Android Runtime:** Android Çalışma Zamanı, Java kütüphaneleri için gereken temel kütüphaneleri ve bir kayıt tabanlı sanal makine olan Dalvik VM'yi içerir. Dalvik VM, düşük bellek gereksinimleri için optimize edilmiştir ve çalışma zamanı belleğini verimli bir şekilde kullanarak optimize edilmiş bytecode yorumlayıcısıyla (interpreter) cihaz başına çoklu sanal makine işlemlerini desteklemek için tasarlanmıştır. Dalvik VM ayrıca, işlemlerin izolasyonunu, bellek yönetimini ve altta yatan işletim sisteminden gelen threading’i desteklemektedir.
3. **Kütüphaneler:** Veritabanı, web tarayıcı ve grafik arayüzlerine ait kütüphaneleri içerir.
4. **Uygulama Çatısı:** Arkaplan servislerinin ve bildirimlerinin yer aldığı ayrıca geliştiricilere sunulan süreçlerin bulunduğu kısımdır.
5. **Uygulama Katmanı:** Uygulama (Application) katmanı, mimarideki en yüksek katmandır ve Telefon Çevirici, E-posta, Kişiler (Rehber), Kamera, Galeri ve diğerleri gibi yerleşik uygulamaları içerir. Uygulama framework katmanı, activity manager, paket yöneticisi (package manager), pencere yöneticisi (window manager) ve diğerleri gibi temel platform hizmetlerini sağlar. Bu katman, donanım erişimine soyut bir arayüz sağlar ve ayrıca kullanıcı arayüzünü veya uygulama kaynaklarını yönetmenin bir yolunu sağlar. Uygulama framework katmanları, geliştiricilerin bir Android uygulaması oluşturmak için hizmetleri bir araya getirmelerini sağlar. Middleware katmanı, C / C ++ çekirdek kütüphanelerini ve Android Çalışma Zamanını içerir. C / C ++ çekirdek kütüphaneleri dört kategoride sınıflandırılabilir; biyonik libc, fonksiyon kütüphaneleri, yerel sunucular ve donanım soyutlama kütüphaneleri (hardware abstraction libraries). Bu çekirdek kütüphaneler, Android'e özgü yerleşik hizmetleri ve standart video, ses ve diğer veri depolamayı destekler.
   1. **ANDROİD İŞLETİM SİSTEMİNİN TARİHÇESİ**

Android İşletim Sistemi’nin temelleri ilk olarak 2003 yılında Kaliforniya’da Andy Rubin, Rich Miner, Nick Sears ve Chris White tarafından atıldı. Bu dört mühendis Linux tabanlı bir İşletim Sistemi geliştirmeye başladı. Temmuz 2005'te Google, Android Inc.'i almış ve ufak bir başlangıç şirketini Palo Alto'da kurmuştur. Android'in kurucuları, Google'a çalışmak için giden Andy Rubin (Danger'in kurucusu), Rich Miner (WildFire'ın kurucusu), Nick Sears ve Chris White'dır. (WebTV'de çalışan ilk mühendislerden biri). Bu arada, Android Inc.'in işlevleri daha önce yaptıkları mobil işletim sistemleri kadar iyi bilinmemektedir ve bu da Google'ın Mobil Cihazlar pazarına gireceği söylentisini başlatmıştır. Buna karşın, bu süreç içinde pazara ne çeşit ürünle çıkacakları tam olarak belli olmamıştır.

Google'da, ekip (Rubin tarafından izin verilen) Linux tabanlı, taşınabilir cihazlar için geliştirdikleri, esnek, güncelleştirilebilir işletim sistemini yazmışlardır. Raporlara göre Google çoktan donanım serilerini, yazılım partnerlerini ve taşınabilir cihazlarda çeşitli derecelerde birlikte çalışacağı sinyalini vermiştir. Google, Android'i satın aldıktan sonra telefon üreticilerini yanına alarak Open Handset Alliance (OHA) birliğini kurmuştur. Android'in gelişimi OHA eliyle yönetilmeye başlanmıştır. Android 1.0 HTC Dream'in işletim sistemi olarak telefonlara ilk adımını atmıştır. Bu sürüm; kamera desteği Wi-Fi, bluetooth, klasörler, bir web tarayıcısı, uyarılar, sesli arama, YouTube, alarmlı saat, galeri gibi özellikler sunmaktadır.

**Android İşletim Sistemi’nin gelişim tarihini kronojik olarak sıralarsak;**

1. 2008 yılında piyasaya sürüldüğünde birçok Android İşletim Sistemi Apache Free-Software ve Açık Kaynak Kodu lisansıyla geliştirilmeye açık hale gelmiştir.
2. 9 Şubat 2009 da Android 1.1 yayınlanmıştır. Bu sürüm bir önceki sürümdeki sistem hatalarını düzeltmeyi ve API'yı geliştirmeyi amaçlamıştır.
3. 30 Nisan 2009'da Android 1.5 (Cupcake) yayınlanmıştır. Bluetooth desteği, kamera kaydı, video gösterimi, Picasa, sık kullanılanlar klasörü ve Widget'ları kişiselleştirebilme, yazı tahmin edebilen klavye ve animasyonlu ekran özellikleri Android'e kazandırılmıştır.

1. 15 Eylül 2009'da Android 1.6 (Donut) yayınlanmıştır. Ekran çözünürlüğü yükseltilmiştir. Doğru çeviri yapma ve okutma özelliği, geliştirilmiş Android market, galeride çoklu silme, sesli arama, WVGA ekran çözünürlük desteği Android'e kazandırılmıştır.
2. 29 Ekim 2009'da Android 2.0 ve 2.1 (Eclair) yayınlanmıştır. HTML 5 ve Bluetooth 2.1 desteği kazandırılmıştır. Sanal klavye ve yüksek ekran çözünürlüğü desteği ile özelleştirilebilen arayüz özelliği eklenmiştir. Geliştirilmiş Google Maps, hareketli duvar kağıtları ve dijital yakınlaştırma özelliği de kazandırılmıştır.
3. Mayıs 2010'da Android 2.2 (Froyo) yayınlanmıştır. 720p ekran çözünürlüğü desteği, USB bağlantı, Wi-Fi tarayıcı özelliği, Flash Player 10.1 desteği eklenmiştir.

1. Şubat 2011'de Android 2.3 (Gingerbread) yayınlanmıştır. Çoklu kamera ve çoklu dokunmatik desteği eklenmiştir. XGA (1366×768) ekran çözünürlük desteği ve video arama özelliği eklenmiştir.

1. Yine Şubat 2011'de Android 3.0 (Honeycomb) yayınlanmıştır. Android işletim sistemi, tabletlerle uyumlu hale getirilmiştir.
2. Ekim 2011'de Android 4.0 (Ice Cream Sandwich) yayınlanmıştır. Yüz tanıma fonksiyonları ve NFC ile dosya paylaşımı özellikleri eklenmiştir.

1. Mart 2012'de Android Market'in adı Google Play Store olarak değiştirilmiştir.

1. Temmuz 2012'de 4.1 (Jelly Bean) yayınlanmıştır. Aynı anda iki uygulama açma özelliği Android'e kazandırılmıştır. Butter project (takılmadan kayan ekran) ve enerji verimliliği sağlanmıştır.

1. Ekim 2012'de 4.2 (Jelly Bean Plus) yayınlanmıştır. Çoklu kullanıcı, 360 derece panoramik fotoğraf ve Swype klavye kazandırılmıştır.

1. 11 Şubat 2013'te 4.2.2 (Jelly Bean Plus) yayınlanmıştır. Bu güncelleme "performansı ve kararlılığı artırır" sloganıyla çıkarılmıştır. Isınma ve donma problemleri giderilmiştir.

1. 24 Temmuz 2013'te 4.3 (Jelly Bean) yayınlanmıştır. Çoklu kullanıcı için ebeveyn kontrolü, oyunlar için OpenGL ES 3.0 desteği, diğer teknolojik aletlere bağlanmak için ilk akıllı Bluetooth teknolojisi, Wi-Fi'ı açıp kapatmak yerine otomatik açılıp kapanabilen akıllı Wi-Fi teknolojisi eklenmiştir. Ayrıca uygulama izinleri bu sürümle birlikte kontrol edilebilir hale gelmiştir.

1. 31 Ekim 2013'te çıkması beklenen 5 (Key Lime Pie) yerine Android 4.4 (KitKat) yayınlanmıştır. Görsel arayüzde birçok değişiklik yapılmıştır. RAM optimizasyonu sayesinde 512 MB RAM'e sahip cihazlarda da çalışabilmektedir. ART (Android Run Time) sanal makinesi Android 4.4 Kitkat Nexus ile deneysel olarak sunulmuştur.

1. 24 Şubat 2014'te MWC fuarında Sony Xperia Z2 ve Samsung Galaxy S5, Android 4.4.2 KitKat kullanmıştır. Cihazlar bu işletim sistemiyle satışa sunulacaktır.

1. 3 Kasım 2014 tarihinde Google Android 5.0 Lollipop sürümünü çıkarttı.

1. 2 Aralık 2014 tarihinde Google Android 5.0.1 güncellemesi yayınlandı.

1. 19 Aralık 2014 tarihinde Google bazı hata düzeltmeleri içeren Android 5.0.2 sürümünü yayınladı.

1. 29 Mayıs 2015 tarihinde Google yaptığı I/O 2015 konferansında Android M 6.0'ı duyurdu ve önizleme sürümünü yayınladı.

1. 19 Ağustos 2015 tarihinde Google Android 6.0'ın kod adını Marshmallow olarak duyurdu ve 3.önizleme sürümünü yayınladı.
2. 29 Eylül 2015'de Android 6.0 Marshmallow bulunan Nexus 5X ve 6P tanıtılmıştır.
3. **PERFORMANS ANALİZİ YAPAN ARAÇLAR**

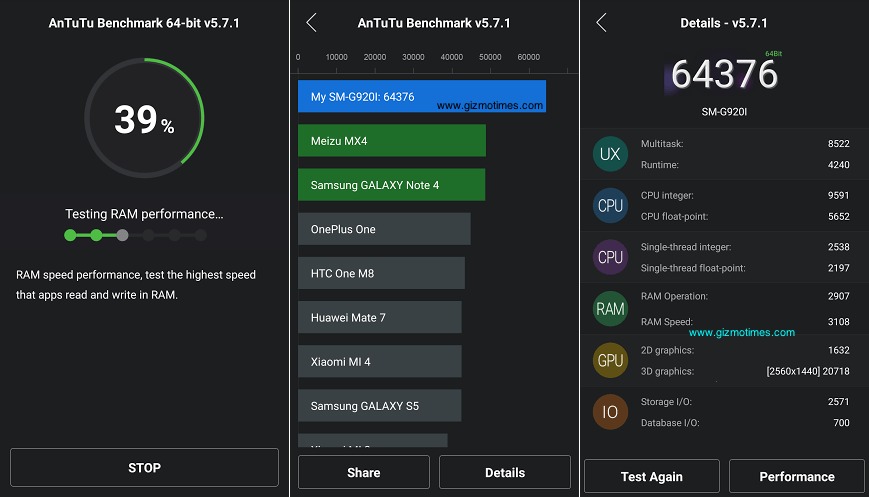
Değerlendirme ve sistem ya da uygulamaların performansını çözümlemek için birçok araç vardır. Uygulama geliştiricileri için bazı benchmark’ları ve performans ölçüm yazılımları incelenip test edilmiştir. Bunlar Android mağazalarından veya açık web sitelerinden ücretsiz olarak satın alınabilirler.

* 1. **ANDROİD SDK ARAÇLARI**

Android yazılım geliştirme kiti (SDK), SDK araçları, yapı araçları ve Platform araçları da dahil olmak üzere farklı bileşenleri içerir. SDK araçları öncelikle Stok Android Emülatörü, Hiyerarşi Görüntüleyici, SDK Yöneticisi ve ProGuard’ı içerir. Yapı Araçları öncelikle aapt (Android paketleme aracı oluşturmak için vardır. Apk), dx (dönüştüren Android aracı .java dosyaları için .dex dosyaları)’ı içerir. Platform Araçları’nda Android Shell Debug, Sqlite3 ve Systrace vardır.

* 1. **BENCHMARK**

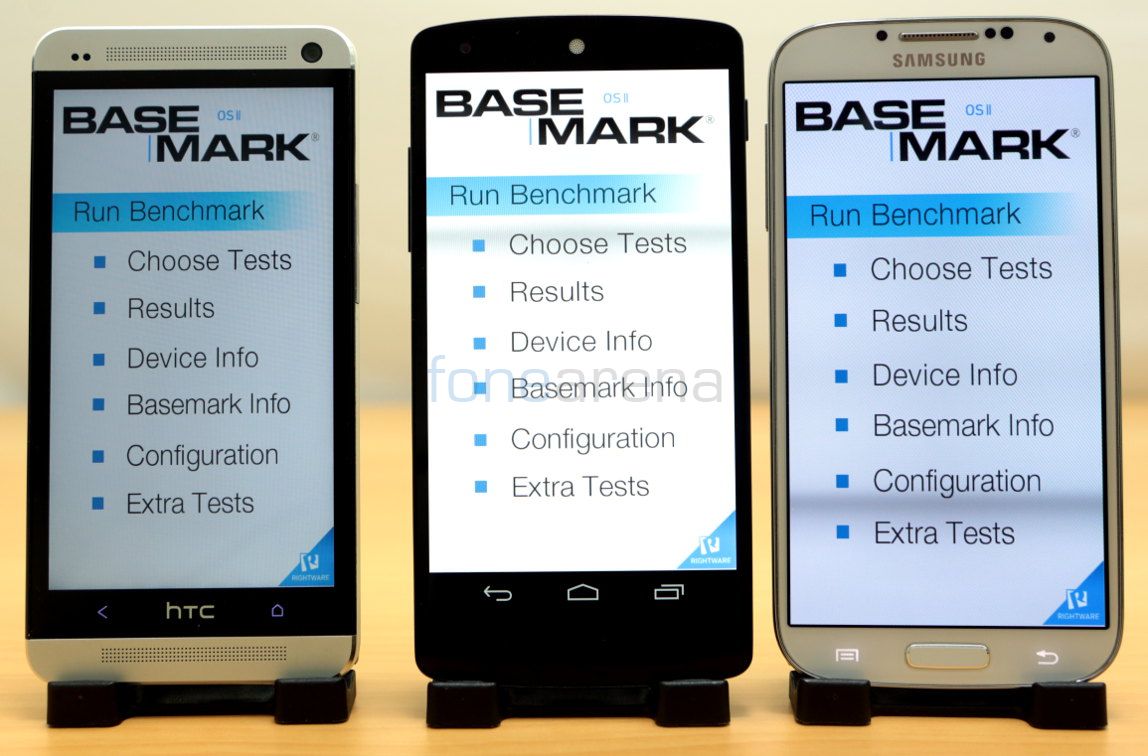
Benchmark aracı, bir sistemin performansını değerlendiren veya ölçen bir programlama uygulamasıdır. Hedef cihaz ve sistemde özel bir program çalıştırılır, performans verilerini toplar ve nicel bir değer olarak gösterir. Şekil 2 iyi bilinen bir benchmark uygulaması olan, Antutu-Benchmark’ın sonuçlarını gösterir. “Bellek performansı”, “CPU Tamsayı Performansı”, “CPU Kayan Nokta Performansı”, “2D 3D Grafik Performansı”, “SD Kart Okuma/Yazma Hızı” ve “Veritabanı IO Performansı” testi ile istenilen bir sistemin tam bir testini yapabilirsiniz. Son puan, test edilen sistemin göreceli değerini temsil eder ve diğer cihazların sonuçlarıyla karşılaştırılabilir. Şekil 2'de 2.resim aynı tür cihazlar arasında sıralamayı gösterir ve Şekil 2'de 3.resim test edilen cihazın Multitasking, CPU, GPU, RAM ve IO performansını test ederek sonuçlarını göstermektedir. Bu sayede cihazların aynı tür donanım ayar durumu ve sistem yazılım sürümüne göre çok farklı performans gösterdiğini görebiliriz.



Şekil 2: Antutu Benchmark [36]

Mobil cihazın GPU bileşenini test eden benchmark yazılımları da mevcuttur.

İki tür mobil GPU benchmark’i vardır: yüksek seviye benchmark’lar ve düşük seviye benchmark’lar. Şekil 3.’te Basemark ES 3.1 [4], Şekil 4’te GFXBench 4.0 [5], Şekil 5 ve Şekil 6’da 3DMark Sling Shot Benchmark [6] gibi üst seviye benchmark’ler genellikle oyun benzeri sahneler oluşturur ve bir cihazdaki kare hızlarını ölçer (veya puanlarını hesaplar). Bu benchmark türleri genel GPU performansını öğrenmek için faydalı olabilir. Buna karşılık, düşük seviyeli benchmark’lar GPU'daki her bir bileşenin belirli özelliklerini veya performansını ölçmeyi amaçlamaktadır. GFXBench'te bulunan düşük seviye testler, tesellation, ALU, drive over-head ve texturing performansını ölçer.



Şekil 3: Android BaseMark [37]



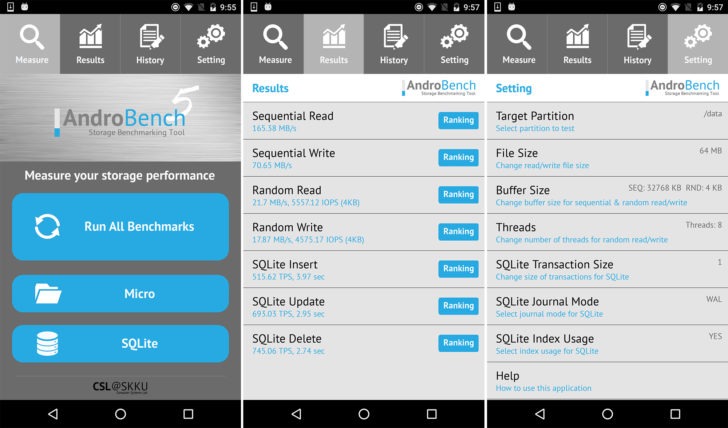
*Şekil 4: GFXBench [38]*





*Şekil 5: 3DMark SlingShot [39] Şekil 6: 3DMark Giriş Ekranı [40]*

Mobil cihazın depolama bileşenini test eden benchmark yazılımları da mevcuttur. Şekil 7’de Androbench, Android İşletim Sisteminde çalışan bir depolama test aracıdır. Androbench'in kritik özelliği, SQLite işlemlerini ölçmek için seçenekler içermesidir. Androbench, mobil cihazdaki dosya ve SQLite işlemlerinin performansını ölçmek için basit ve güçlü bir benchmark (test) aracıdır. Her ne kadar Androbench, mobil cihazda dosya ve SQLite işlemlerinin performansını ölçmek için basit ama güçlü bir benchmark aracı olsa da, Android tabanlı cihazların çeşitli IO özelliklerini ölçmede üç adet sınırlaması vardır. İlk olarak, Androbench O SYNC, O DIRECT ve mmap gibi senkronizasyon seçeneklerinin değiştirilmesine izin vermez. Sadece mevcut seçenek O SYNC. İkincisi, fsync () çağrılarının performans etkisini ölçemez. Son olarak, Androbench çoklu iş parçacığı benchmark ortamını desteklemiyor.



Şekil 7: AndroBench [41]

Mobil cihazın diğer bileşenlerini de test eden benchmark yazılımları da mevcuttur. Bildiğimiz kadarıyla, Android tabanlı mobil cihazlar için özel olarak hazırlanmış bir benchmark yazılımı henüz önerilmemiştir. Bununla birlikte, mobil cihazlar için bazı benchmark yazılımları mevcuttur, bunlar genellikle mobil cihazlarda belirli bir özelliği hedefler. Ayrıca, MiBench [7] veya PARSEC [8] gibi geleneksel benchmark yazılımları, IOS veya Android gibi işletim sistemleriyle uyumsuzluk sorunu nedeniyle mobil cihazların performansını karakterize etmek için yeterli değildir. DENBench [9], AES (Gelişmiş Şifreleme Standardı), DES (Dijital Şifreleme Standardı) ve diğerleri gibi mobil uygulamalarda yaygın olarak kullanılan algoritmaları içeren yazılım test araçlarıdır. Ancak, DENBench sadece medya kütüphanesini göz önüne alır. DENBench açık kaynak kodlu veya ticari olarak ücretsiz değildir. GPSBenchmark [10], yalnızca mobil cihazların GPS performansını inceler ve kaynak kodu kullanılamaz. JBenchmark [11], mobil telefonlar ve PDA'lar için grafiksel bir benchmark yazılım aracıdır. JBenchmark, animasyonda ağır yük, mapping testleri, kullanıcı arayüzü simülasyonu gibi çeşitli özelliklere sahiptir. Ancak, JBenchmark yalnızca J2ME özellikli cihazlarla uyumludur. GLBenchmark [12], Android, iOS, Linux, Symbian ve Windows Mobile gibi çeşitli işletim sistemlerinde OpenGL ES için tasarlanmış bir 3D test paketidir. Bu araç, kaynak kodunun değiştirilmesine izin vermeyen lisanslı bir ticari 3D test aracıdır. Son zamanlarda, BBench modern akıllı telefonlar için önerilmiştir. Ancak BBench, sadece Web tarayıcılarını hedeflemektedir.

1. **BENCHMARK KRİTERLERİ** 
   1. **CPU**

İşlemci için, günümüz dünyasında söz konusu cihaz -masaüstü, dizüstü bilgisayar, mobil ve diğerleri - için tüm sahneyi temel olarak çalıştıran ‘bir aracın motoru’ gibi olduğu söylenebilir. Akıllı telefon veya PC’deki işlemci ne kadar güçlüyse söz konusu cihaz daha hızlı ve aynı anda birden çok işi daha rahat yapabilir.

Tek çekirdekli bir işlemci, sadece bir işlemci bir gerçek merkezi işleme birimi (CPU) içerir ve bu bir yonga adı verilen tek bir bileşen üzerine kurulu bir işlemcidir.

Çift çekirdekli işlemci tek bir bilgisayar bileşenidir, ya da başka bir deyişle sadece tek bir bilgisayar yongasıdır, ama bu bir bileşenin içinde iki bağımsız merkezi işleme birimi vardır. Aslında sadece tek çekirdekliye benzer, ancak 1 yerine 2 CPU'ya sahiptir.

Benzer şekilde, dört çekirdekli bir işlemci, tek bir bilgisayar bileşeni içinde, 4 bağımsız gerçek merkezi işleme birimi ( CPU'nun) olan tek bir bilgisayar yongasıdır. Ve bir sekiz çekirdekli işlemci, sekiz bağımsız gerçek merkezi işleme birimi ( CPU'nun) olan tek bir bilgisayar bileşenidir.

Ayrıca, hexa-core olarak da bilinen 6 çekirdekli işlemci ve ayrıca deca-core olarak da bilinen 10 çekirdek ile birlikte gelen akıllı telefonlar da vardır.

Daha fazla işlemci çekirdeği, daha fazla verim ve daha az ısınma sorunu anlamına gelmektedir. Bunun nedeni, birden çok çekirdekle, PC'nizdeki veya telefonunuzdaki farklı görevlerin çekirdekler arasında dağıtılmasıdır. Birden çok çekirdekli işlemci, cihazınız tarafından tüketilen pil miktarını azaltmaya yardımcı olabilir ve ayrıca telefonunuzun veya bilgisayarınızın daha sorunsuz çalışmasına ve daha ağır görevleri göreceli olarak kolaylıkla idare etmesine yardımcı olabilir.

Birden fazla çekirdek, telefonunuzda veya PC'nizde aynı anda birden fazla şey veya görev yapmanız gerektiğinde en yararlı olanıdır. Buna multitasking(Aynı anda birden fazla işi yapma) denir. Çok çekirdekli işlemcilerde, genellikle her çekirdek diğerinden bağımsızdır. Işık kullanımı sırasında, normal ışık kullanım yükünü işlemek için bir seferde yalnızca bir çekirdek çalışabilir ve daha ağır kullanıma geçtiğinizde, yükü hafifletmeye yardımcı olmak için daha fazla çekirdek devreye girer.

İşlemci çekirdeklerini 2 katına çıkarmak aynı kullanım senaryosunda 2 katı hıza ulaşmamızı sağlamaz. Bir 2 çekirdekli işlemci telefonunuzu tek çekirdekli işlemciye göre 2 katı hızla çalıştırmaz. Çekirdek sayısını katlamak o kadar kat hıza ulaştırmaz.

Çekirdek sayısının yanı sıra, işlemciler 1.4 GHz, 1.8 GHz, 2.2 GHz vb. hızlarla derecelendirilmiştir. Sayı ne kadar yüksek olursa, tek çekirdeğin performansı ve işlemcisi o kadar hızlı olur. 1.8 GHz tek çekirdekli işlemci, yalnızca bir işlemcinin ele alındığı tek bir senaryoda 1.4 GHz çift çekirdekli işlemciden daha güçlü ve daha hızlıdır, ancak ikincisi, birden fazla görevi aynı anda çalıştırmayı içeren senaryolarda daha iyi performans sağlayacaktır.

Bu sadece normal CPU performans karşılaştırmasıdır ve hiçbir akıllı telefon hatta hiçbir elektronik cihaz bir diğeriyle sadece performans açısından tam olarak aynı şartlar altında karşılaştırılamaz. CPU'nun optimizasyonuna etki eden pek çok faktör vardır, mevcut RAM miktarı, GPU, en önemlisi hangi çekirdeğin en iyi optimizasyonların ne zaman sağlanacağına karar veren yazılımlar. Dolayısıyla yazılımlar tüm bunları hesaplar ve dolayısıyla yükü nispeten böler, yazılımlar olmadan dört çekirdekli bir işlemci, çift çekirdekli bir işlemci gibi iş yapabilir, dolayısıyla tamamen farklı sonuçlar ortaya çıkabilir [13].

**CPU’nun Çalıştığı Örnek Uygulamalar**

**1) FBReader [14]:** FBReader en popüler ücretsiz Android e-kitap okuyucu uygulamasıdır. Mobil cihazlarda e-kitap yükleme işlemini ve metin arama işlemlerini değerlendirmek üzere FBReader seçilmiştir. FBReader, aramalar veya metin yeniden boyutlandırmada hızlı hesaplamalar yapmak için bir dosya sisteminden belleğe eksiksiz bir e-kitap yükler. FBReader, metin araması için karmaşık bir algoritma uygulamamaktadır, ancak metni aramak için metin modelindeki tüm paragraflar ve her paragraftaki metin blokları üzerinde yinelenmektedir. Sonuç olarak, arama işlemi bellek kapasitesinin mobil cihazların performansını nasıl etkileyeceğini inceleyecektir.

**2) Barkod Tarayıcı [15]:** Barkod Tarayıcı, CD'lerde, kitaplarda ve diğer ürünlerde barkodları tarar. Barkod Tarayıcı, açık kaynaklı bir barkod görüntü işleme kütüphanesi olan ZXing kütüphanesini kullanır. Mobil cihazlarda yerleşik kamerayı kullanarak barkodları çözer. Desteklenen formatlar UPC-A / E, EAN-8/13, QR kodu ve diğerleridir. Barkod Tarayıcı, QR kod dedektörü için kendi algoritmasını uygular. Algoritmanın temel unsurları aşağıdaki gibidir.

• Binary veriyi alabilmek için Gray-Scale verisi üzerinde eşik (threshold) kararları almak

• Ham görüntü piksellerini modüllere örnekleme; Her QR kod modülü ham görüntü pikselleriyle yayılmalıdır.

• Hizalama düzenlerinin (pattern) tespiti.

• Görüntü döndürme ve doğru perspektif bozulma.

**3) libGDX 2D [16]:** LibGDX, Java / C / C ++ ile yazılmış, ticari olarak bedava bir oyun geliştirme framework programıdır ve birçok Android oyun uygulaması libGDX motorunu temel alır. Mobil cihazlarda 2D grafiklerin ve ilgili hesaplamaların performansını incelemek için LibGDX 2D seçilmiştir.

2B animasyon, mobil cihazların ekranında 2B görüntüler çizmenin temel yeteneklerini inceler. Sprite ve Sprite Batch, 2D hesaplamanın maksimum kapasitesini kontrol etmek için tam stres hesaplaması yapar. Matrix Hesaplaması, JNI kullanarak Java kaynak kodundan ve C yerel (native) kaynak kodundan 4'e 4 matris çarpımının yürütme süresini karşılaştırır. Dahil edilen hesaplamalar ise temel bir matris çarpımı, bir vektör çarpımı ve bir matrisin tersini alma işlemleridir. Parallex, yan yana hareket eden bir animasyonu test eder. Toplama (Culling), ekranda görünen nesneleri belirleme yeteneğini inceler. Vertex Buffer Object, vertex, normal, color ve belirli bir dizideki diğer bilgiler gibi verileri mobil cihazlardaki video bileşenine yüklemek için yöntemleri test eder.

**5) Kamera [17]:** Kullanıcılar mobil cihazı günlük yaşamlarında sık sık taşıdıklarından, mobil cihazlarda yerleşik bir kamera sıklıkla fotoğraf çekmek için kullanılır. Kamera, fotoğraf çekebilen veya video kaydedebilen bir Android yerleşik uygulamasıdır. AM-Bench, yerleşik kameraların mobil cihazlardaki performansını test etmek için bir Kamera benchmark yazılımı içerir. Kamera benchmark, yerleşik kameraların dört ana işlemini mobil cihazlarda inceler: başlatma, gecikme, JPEG call back ve değiştirme (switch) modu.

**6) Galeri [17]:** Galeri, mobil cihazlarda multimedya içeriği görüntüleyen bir Android yerel uygulamasıdır. Küçük Resim (Thumbnail) Yükleme kriter olarak alındığında, kullanıcılar ilk kez bir albüm seçtiklerinde yüklenen toplam resim sayısı, mobil cihazların ekran boyutuna bağlıdır. Böylece, yürütme süresinin ortalaması hesaplanır.

* 1. **GPU**

Mobil cihazlarda yüksek kaliteli grafiklere olan artan taleple, GPU'lar artık akıllı telefonlar ve tabletler gibi mobil cihazlar için vazgeçilmez birimler olmuştur. Özellikle, yüksek çözünürlüklü ekrana sahip mobil cihazlar için daha güçlü GPU'lar gerekmektedir. Sonuç olarak, GPU'lar genellikle geniş bir uygulama alanında kullanılır ve GPU performansı, mobil aygıtları seçmek için önemli ölçütlerden biri haline gelmiştir. Bununla birlikte, GPU performans analizi kolay bir iş değildir çünkü GPU, geometri birimleri, ALU'lar, doku haritalama (mapping) birimleri, raster işlem pipeline’lar (ROP'lar), önbellek / bellek birimleri ve tessellatorler gibi çok karmaşık bileşenlerden oluşur. Tabii ki, GPU sağlayıcıları her birimin en yüksek performansını sağlar (örneğin, triangle rate, FLOP'lar, the texel rate, the fill rate, en yüksek bellek bant genişliği, vb.). Ancak, bu en yüksek performans değerleri, farklı GPU mimarileri arasında karşılaştırılamayabilir, çünkü gerçek performans, donanım organizasyonu, mimari özellikler, programlama politikaları, sürücü desteği ve benzeri gibi çeşitli faktörlerle değişebilir. Bu yüzden uygun GPU benchmark yazılımlarına ihtiyaç duyulmuştur.

Daha önce benchmark başlığı altında bahsedildiği gibi, iki tür mobil GPU benchmark’ı vardır: yüksek seviye benchmark’lar ve düşük seviye benchmark’lar. GFXBench 4.0 [5], Basemark ES 3.1 [4] ve 3DMark Sling Shot Benchmark [6] gibi üst seviye benchmark’lar genellikle oyun benzeri sahneler oluşturur ve bir cihazdaki kare hızlarını ölçer (veya puanlarını hesaplar). Bu benchmark türleri genel GPU performansını öğrenmek için faydalı olabilir. Buna karşılık, düşük seviyeli benchmark’lar GPU'daki her bir bileşenin belirli özelliklerini veya performansını ölçmeyi amaçlamaktadır. GFXBench'te bulunan düşük seviye testler, tesellation, ALU, drive over-head ve texturing performansını ölçer.

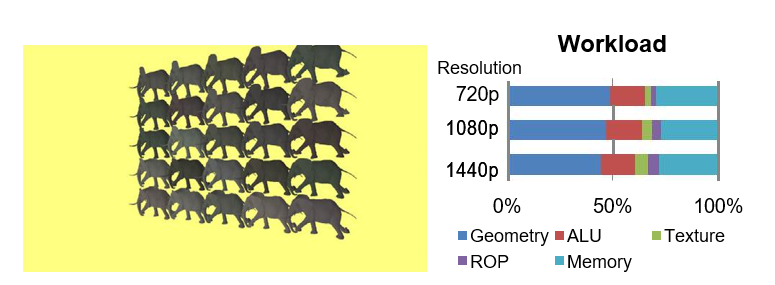
Mevcut benchmark’ların sınırlandırılması, sonuçları derinlemesine analiz etmede büyük zorluklar çıkartır. Örneğin, aynı seviyedeki GPU'lar üst seviye benchmark’larda farklı kare hızları gösteriyorsa, sonuçların nedenini bulmak zordur.

Diğer bir endişe, gerçek dünya senaryolarıyla olan korelasyondur; üst seviye benchmark yazılımları genellikle yalnızca bir veya iki sahne gösterdiğinden, GPU tedarikçilerinin belirli benchmark yazılımları için bazı sürücü optimizasyonları, benchmark yazılımlarında büyük hızlanmalar sağlayabilir ve bu sonuçlar gerçek dünya senaryolarındaki gerçek performanstan farklı olabilir.

Benchmark’da GPU’yu test etmek için kullanılan mikro-ölçüt (MicroBenchmark) teknikleri vardır.

* 1. **MİKRO-ÖLÇÜT (MİCRO-BENCHMARK) TEKNİKLERİ ÜZERİNDE YAPILAN BAZI ÇALIŞMALAR**
     1. **Geometry Instancing with Skinned Animation**

Geometri örneklemesi (Geometry instancing), geometri verilerinin birden fazla kez kullanılmasını kolaylaştırır ve OpenGL ES 3.0' daki yeni özelliklerden birisidir. Bu özelliğin test edilebilmesi için, bu özellik benchmark programına dâhil edilmiştir. Önce OpenGL ES 3.0 Programlama Kılavuzu'ndaki [18] basit bir geometri örnekleme kodu, temel kod olarak kullanılmış, ondan sonra ayrıca ‘skinned animation’ adlı koda ‘per-vertex lighting to the vertex shader’ tekniği uygulanmıştır. Şekil 8’de görüldüğü gibi, sahne üzerinde 25 adet fil oluşturulmuştur ve her fil nesnesi 84K tane üçgenden oluşuyor. Böylece, sahnede toplam 2.1M üçgen oluşturulmuştur. Sonuç olarak, bu uygulama temel olarak geometri işlemeye odaklanır.

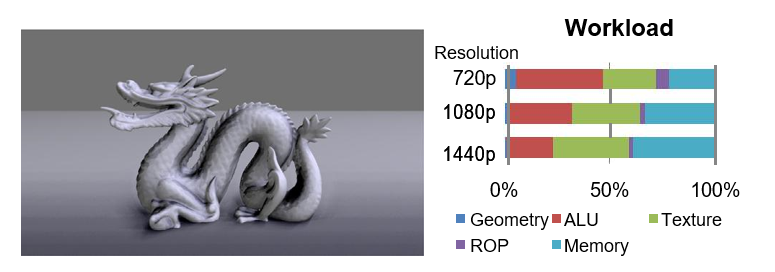


Şekil 8: Geometry Instancing with Skinned Animation [34]

* + 1. **Screen-Space Ambient Occlusion**

Screen-Space Ambient Occlusion (SSAO) [19,20] ortam aydınlatması için yaklaşık bir yöntemdir ve şu anda çeşitli masaüstü / konsol oyunlarında yaygın olarak kullanılmaktadır. Bu teknik, ekran boşluğunda bitişik piksellerin derinlik ve normal değerlerini kullanarak ortam aydınlatma etkisini yaklaşık olarak gösterir, böylece OpenGL ES 3.0'da çoklu renderleme hedefleri (MRT'ler) kullanılarak uygulanabilir.

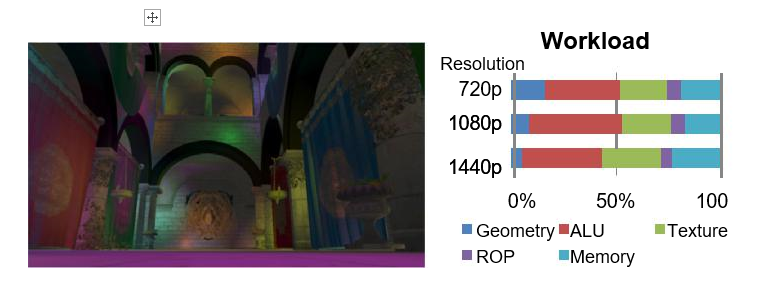
Temel kaynak kodu olarak, OpenGL SuperBible'da [21] bulunan Dragon Scene (100K üçgenleri) ile örnek bir kod kullanılmıştır. Bu kod iki geçiş gerçekleştirir, render ve SSAO, renk verilerini transfer etmek için kayan noktalı MRT kullanılır ve derinlik / normal verileri render geçişten SSAO geçişine aktarma yapılır. Bununla birlikte, kayan nokta oluşturma hedefleri OpenGL ES 3.1'de resmi olarak desteklenmemektedir. Bu engelin üstesinden gelmek için [22] 'da kayan nokta kodlama / kod çözme tekniği ile iki kanallı RG dokular kullanılmış; veriler 8+8 bit sabit nokta gösterimi olarak ele alınmıştır.



Şekil 9: Screen-Space Ambient Occlusion [34]

* + 1. **Tiled Deferred Shading**

Birçok ışık kaynağı ile döşenmiş ertelenmiş gölgeleme (Tiled deferred shading) [23], MRT'lerin temsili bir uygulamasıdır. Bu tekniğin amacı, gereksiz aydınlatma işlemlerini ayırmaktır. Bunu başarmak için önce geometri, yoğun G-tamponlarına dönüştürülür. Daha sonra, ekranı çoklu döşemelere bölerek bir CPU üzerine, bir ekran boşluk ızgarası inşa edilir. Her döşemenin minimum ve maksimum derinlik değerleri ilk önce hesaplanır (min - maks. Küçültme), bundan sonra sahnedeki tüm ışık kaynakları, etkilenen her bir 2,5D (2B derinlik aralığı) ızgara hücresine atanır. Son aydınlatma işlemleri daha sonra G tamponları ve ışık ızgarası kullanılarak gerçekleştirilir.

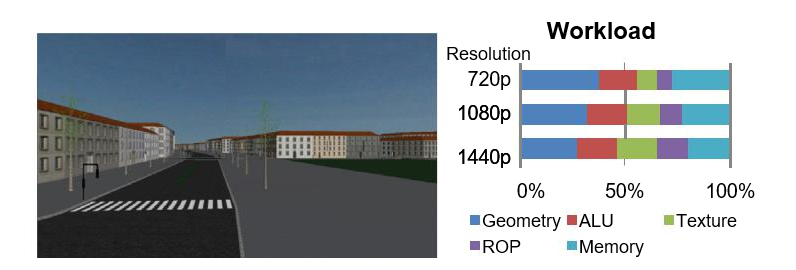


Şekil 10: Tiled Deferred Shading [34]

* + 1. **Occlusion Culling**

Tıkanma (Occlusion) sorguları OpenGL ES 3.0'da desteklenir ve büyük sahnelerde tıkanma engelleme için kullanılabilir. Masaüstü platformlarda çok ünlü bir teknik olduğu için, bu teknik benchmark yazılımında yer verilmiştir. CHC++ algoritması, arka arkaya sırayla sınırlayıcı bir hacim hiyerarşisini (BVH) geçer ve düğümler üzerindeki oklüzyon (Occlusion) sorgularını kullanarak düğümlerin görünürlüğünü kontrol eder.

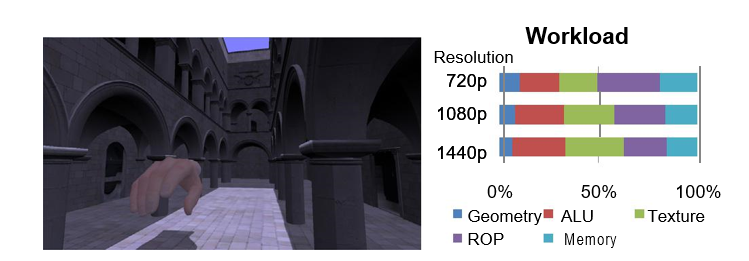
Sonuç olarak, tıkalı nesneler toplanır (Şekil 11).



Şekil 11: Occlusion Culling [34]

* + 1. **Shadow Mapping**

Gölge eşleme, raster tabanlı GPU'larda gölgeler oluşturmak için kullanılan geleneksel bir tekniktir. Bu teknikte, ışığın görünümündeki derinlik değerleri ilk önce bir gölge haritada (shadow map) saklanır. Bundan sonra, ışık ile kamera görüntüsündeki her nokta arasındaki mesafe, gölge haritada kaydedilen mesafe ile karşılaştırılır; önceki değer ikinci değerden yüksekse, nokta gölgelidir çünkü ışık ile nokta arasında bir tıkayıcı (veya tıkayıcı) vardır (Şekil 12).

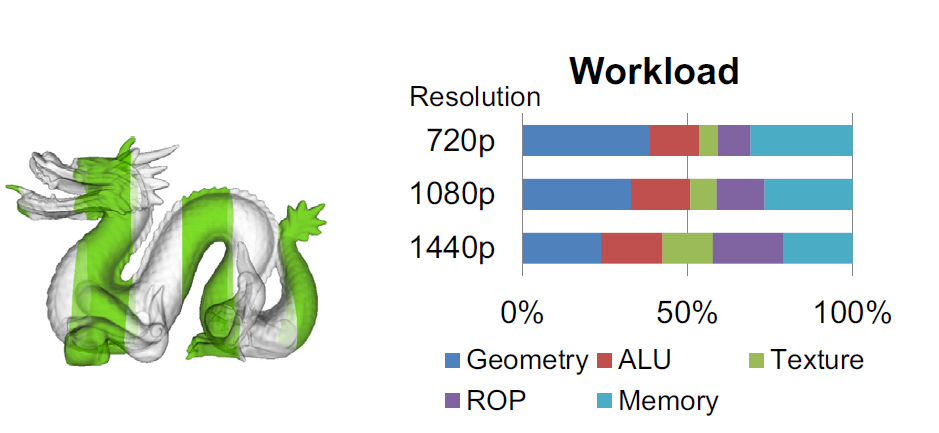


Şekil 12: Shadow Mapping [34]

* + 1. **Order-Independent Transparency**

Order-independent transparency (OIT), raster tabanlı GPU'larda zorlu konulardan birisidir, çünkü basit alfa harmanlama (blending), sunulan geometri sırasına göre farklı görüntülerle sonuçlanabilir.

Ağırlıklı harmanlanmış OIT [24] son zamanlarda tanıtılan hafif bir algoritmadır. Bu yaklaşım, bu parçalar karıştırıldığında derinlik değerlerine göre parçalara farklı ağırlıklar verir. Yaklaşım için sadece sınırlı hafızaya sahip klasik harmanlama işlemleri gereklidir, ancak harmanlanmış görüntüler makuldür.



Şekil 13: Order-Independent Transparency [34]

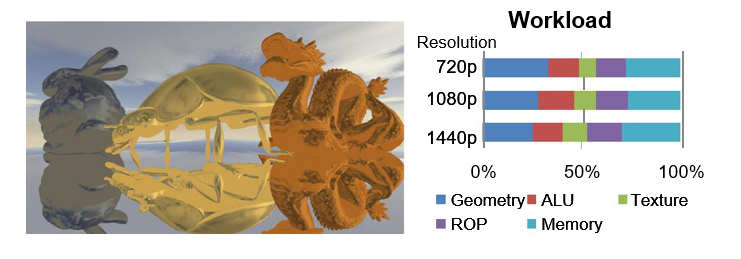
* + 1. **Cube Mapping and Planar Reflection**

Küp haritalama [25], yansıma etkileri için yaygın olarak kullanılan bir tekniktir.

Bu yöntemde, bir küp görüntüsünün altı yüzüne bir ortam görüntüsü yansıtılır ve yansıtıcı nesneler oluşturulduğunda, olay ve normal vektörler ayarlanarak harita alınır.

Dinamik küp haritası oluşturma maliyeti yüksek olduğu için, küp haritalaması genellikle gökyüzü ve dağlar gibi statik ortamları temsil etmek için kullanılır. Bu nedenle, gerçek zamanlı görüntülemede dinamik yansıma olması durumunda, genellikle şablon tamponu [26] kullanan düzlemsel yansımalar kullanılır.

Bu teknik, yansıtıcı malzemeleri içeren düzlemsel bir nesneye (örneğin bir zemin) yansıtılmış bir görüntü verir ve şablon değerlerini test ederek yansıtılan alanı kısıtlar (Şekil 14).



Şekil 14: Cube Mapping and Planar Reflection [34]

* + 1. **BVH Update and Visualization**

OpenGL ES 3.1, hesaplama gölgelendiricilerini geliştiricilere sunmaktadır.

Bu alt bölümdeki BVH güncelleme ve görselleştirme testi, bilgisayar gölgelendiricilerin mevcut grafik uygulamalarına nasıl entegre edileceğine bir örnektir. Bu micro benchmark, bir sahnenin BVH'sini anında günceller ve BVH'yi ekranda görselleştirir.

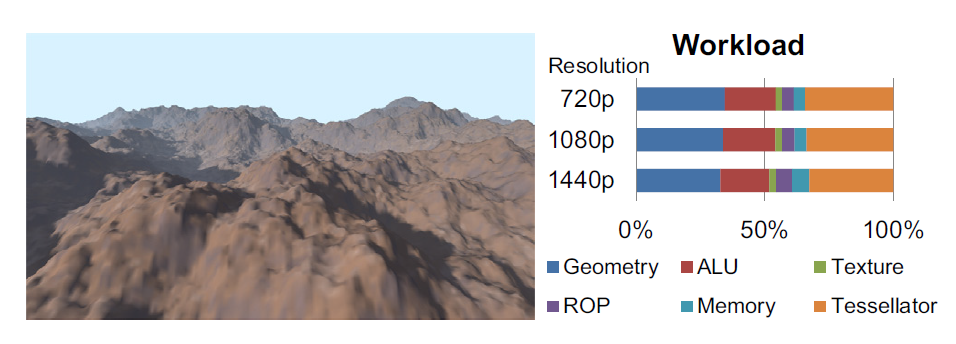
BVH, GPRS’te paralel olarak gProximity [27] 'te açıklanan seviye seviyesine göre güncellenmiştir. Bu BVH güncellemesi, dinamik sahnelerde BVH tabanlı çarpışma tespiti [27], ışın izleme (ray tracing) [28] ve tıkanma temizleme (occlusion culling) [29] için kullanılabilir (Şekil 15).



Şekil 15: BVH Update and Visualization [34]

* + 1. **Displacement Mapping with Tessellation**

Mozaik (tesellation) gölgelendirici, AEP'nin temel özelliklerinden biridir. Mozaikleme kontrolü ve değerlendirme gölgelendiricileri arasında, bir donanım tessellator’u bulunur ve performansını bu yer değiştirme haritalandırma micro benchmark’i içinde test edilmiştir. Bu micro benchmark için kaynak kodu ve sahne OpenGL SuperBible'dan; bir yer değiştirme sahnesi (Şekil 16), yer değiştirme haritalaması ve taşıt latasyonu kullanılarak gerçekleştirilir. İşlem kısaca şu şekilde tarif edilmiştir: İlk önce 64x64 yaması girilir. Bundan sonra, mozaikleme seviyeleri, mozaikleme kontrol gölgelendiricisinde bir kerede hesaplanır. Daha sonra, tessellator'dan elde edilen sonuç, daha sonra mozaik değerlendirme gölgelendiricisine gönderilir ve yer değiştirme eşlemesi, bir yükseklik haritası kullanılarak gölgelendiricide gerçekleştirilir. Son olarak, doku eşlemesi, parça gölgelendiricisinde gerçekleştirilir.



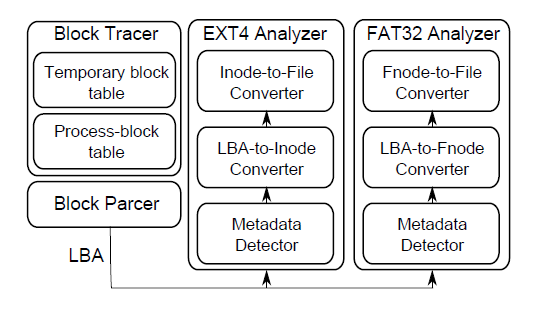
Şekil 16: Displacement Mapping with Tessellation [34]

* 1. **DEPOLAMA**

Çoğu mobil cihaz, sınırlı miktarda RAM, dâhili flash depolama ve harici bir SD kart yuvası ile birlikte verilir. Ek olarak, bazı cihazlarda (örneğin, LGG2X telefon) telefonun içinde çıkartılamayan bir SD kart bulunur; bu depolama hala harici depolama olarak kabul edilir.

Dâhili flaş depolama birimi; önyükleyici ve çekirdek bölümleri, kurtarma, sistem ayarları, önceden kurulmuş sistem uygulamaları ve kullanıcı tarafından yüklenmiş uygulama verileri dâhil tüm önemli sistem bölümlerini içerir. Harici depolama birimi, öncelikle medya dosyaları (yani, şarkılar, filmler ve fotoğraflar), belgeler ve yedek görüntüler gibi kullanıcı içeriğini depolamak için kullanılır. [30]

Mobil Depolama Analiz Cihazı (Mobile Storage Analyzer), mobil cihazlarda depolama birimini analiz etmek için kullanılan bir yazılımdır. Mobil Depolama Analiz Cihazı (MOST) şunlardan oluşur: (i) IO'lar için işlemleri ve dosya ile ilgili bilgileri koruyan değiştirilmiş bir Linux çekirdeği; (ii) belirli bir blok için bir dosyanın tanımlanmasını sağlayan bir blok analizcisi ve (iii) blktrace yardımcı programı. Şekil 17’de, MOST'un bir şeması bulunmaktadır.



Şekil 17: Mobile Storage Analyzer [32]

Modern bir IO alt sisteminin katmanlı yapısı nedeniyle, oturumla (session) ilgili bilgilerin blok cihazı düzeyinde tanımlanması mümkün değildir. Bir I/O isteği, katmanlar arasında, örneğin dosya sisteminden blok aygıtı katmanına geçildiğinde, oturuma ilişkin bilgiler (yani, dosya kimliği ve işlem kimliği) kaybolur. Bloklar, ilgili dosyalar ve işlemler arasındaki ilişkileri analiz edebilmek için bilgilerin farklı katmanlardan toplanması gerekir. MOST, dosya bilgilerinden sonuç çıkarmak için IO izini, blok aygıtı sürücü düzeyinde toplar ve ardından her bir blok için bilgileri işler. MOST üç ters eşleme (reverse mapping) sorununu ele alır: LBA-dosya eşlemesi, LBA-işlem eşlemesi ve geriye dönük LBA eşlemesi.

LBA-dosya eşlemesi için MOST, disk bloğunu ait olduğu dosyaya ters eşleyebilir. Bir mantıksal blok numarasını bir giriş olarak kabul eder ve bir dosya adı oluşturur. MOST, EXT4 dosya sistemindeki bloğu ve FAT32 dosya sistemi için bir dâhili modülü eşlemek için debugfs [31] kullanır.

MOST, belirli bir IO yayınlayan orijinal süreci tanımlar. Android'de, mmcqd daemon, mmc kart aygıt sürücüsünü yönetir ve tüm blok IO'ların yayınlanmasından sorumludur. Herhangi bir değişiklik yapılmadan, blktrace, ilgilendiğimiz bilgiler olmayan, mmcqd arka plan programı tarafından başlatılan tüm blok IO’ları bildirir. Android Çekirdeğinde bir işlemden-bloğa (işlem-blok) eşleme (mapping) tablosu oluşturuluyor. Tablonun girişi <LBA, işlem kimliği> 'dir. IO zamanlayıcısı IO isteğini sıraya eklediğinde, MOST <LBA, işlem kimliği> bilgisini işlemden-bloğa (işlem-blok) eşleme tablosuna ekler. MOST daha sonra LBA ile işlem kimliğini almak için işlemden-bloğa (işlem-blok) eşleştirme tablosuna daha sonra atıfta bulunur.

MOST, geriye dönük LBA haritalamasına izin verir. Android'de, birçok dosyanın kısa ömürlü olduğunu, SQLite tarafından oluşturulduğunu ve hızla silindiği görülmüştür. Bu geçici dosyalar SQLite tarafından yönetsel amaçlar için yaratılmıştır, örneğin, depolama aygıtında değişikliklerin güncellenmesi ve yapılması için geçici bir veritabanı dosyası oluşturulması gibi. Bu dosyaların nasıl kullanıldığını tam olarak anlayabilmek çok önemlidir. Kısa ömürlü olmalarına rağmen, her dosya NAND depolamasına göre fsync () 'dir ve bu da sistem performansını büyük ölçüde etkiler. Bir iz kaydedildiğinde verilen bir LBA için dosya bilgisine ihtiyacımız var, sonradan analiz edildiğinde değil. MOST verilen bir LBA için analiz prosedürünü başlattığında, bloğun ait olduğu geçici bir dosya silinmiş olabilir ve bu nedenle bulunamıyor olabilir. Bu sorunu çözmek için, MOST, Android çekirdeğinde bir dosyadan bloğa eşleme tablosu oluşturur. Dosya-blok eşleme tablosu, <LBA, dosya> dizisidir. IO zamanlayıcısı LBA'da zamanlayıcı sırasına takıldığında, MOST dosya-blok eşleme tablosuna <LBA, dosya> girişi ekler. Daha sonra, MOST verilen bir LBA için dosya bilgisini elde etmek üzere bu tabloya başvurur.Tablo boyutunu küçültmek için MOST, yalnızca geçici dosyalar için bir <LBA, dosya> girişi ekler, yani dosya uzantısı .db-journal, .db-mjxxxx, .bak veya tmp gibi. Blktrace izleme dosyası için bir günlük oluşturduğunda, verilen bloğun ilgili IO'yu tetikleyen geçici dosyalara ait olup olmadığını belirlemek için geçici blok tablosuna başvurur.

MOST, mantıksal blokları üç tipte sınıflandırır: Metadata (Meta veri), Journal ve Veri.  
EXT4 dosya sisteminde, Meta veri blokları bir süper blok, grup tanımlayıcısı, veri bloğu (data block) bitmap, inode bitmap ve inode tablosunu barındıran bloklardır. FAT32 dosya sisteminde Meta Veri blokları, bir önyükleme kaydını içeren bloklara ve Dosya Ayırma Tablosuna (FAT) karşılık gelir. Günlük, EXT4 dosya sisteminin günlük bloğudur. Veri blokları, dosya verilerini ve dizin girişlerini barındıran bloklardır.

1. **UYGULAMA TESTLERİ**

**4.1. Test Cihazları**

|  |  |  |
| --- | --- | --- |
| Test edilen Cihazlar | GPU | Ekran |
| Samsung Galaxy Note 4  (Android 6.0.1 Marshmallow) | Mali-T760 MP6  700 MHz  OpenGL ES 3.1  GPU Çekirdekleri = 6 | 5.7 inc  1440 x 2560 piksel  518 ppi |
| Samsung Galaxy J3  (Android 5.1.1 Lollipop) | Mali-400 MP  GPU Çekirdekleri = 2  OpenGL ES 2.0 | 5.0 inc  720 x 1280 piksel  294 ppi |
| Samsung Galaxy Win  (Android 4.1.2 Jelly Bean) | Adreno 203  245 MHz  GPU Çekirdekleri = 2  OpenGL ES 2.0 | 4.7 inc  480 x 800 piksel  199 ppi |

Şekil 18: Test Edilen Cihazlar GPU-Ekran

|  |  |  |
| --- | --- | --- |
| Test edilen Cihazlar | RAM | Depolama |
| Samsung Galaxy Note 4  (Android 6.0.1 Marshmallow) | 3 GB RAM  LPDDR3  Double Channel  825 MHz | 32 GB |
| Samsung Galaxy J3  (Android 5.1.1 Lollipop) | 1.5 GB RAM  LPDDR3  Single Channel  533 MHz | 8 GB |
| Samsung Galaxy Win  (Android 4.1.2 Jelly Bean) | 1 GB  LPDDR2  Single-channel  300 MHz | 8 GB |

Şekil 19: Test Edilen Cihazlar RAM-Depolama

|  |  |
| --- | --- |
| Test Edilen Cihazlar | CPU |
| Samsung Galaxy Note 4  (Android 6.0.1 Marshmallow) | Exynos 5433 Octa-core (4x1.3 GHz Cortex-A53 & 4x1.9 GHz Cortex-A57)  Instruction Set: ARMv8-A 64-bit  Level 1 (L1) Cache Memory: 32 KB + 32 KB  Level 2 (L2) Cache Memory: 2048 KB |
| Samsung Galaxy J3  (Android 5.1.1 Lollipop) | Spreadtrum SC9830I Quad Core (4x ARM Cortex-A7 1.50 GHz)  Level 0 (L0) Cache Memory: 4 KB + 4 KB  Level 1 (L1) Cache Memory: 16 KB + 16 KB  Level 2 (L2) Cache Memory: 2048 KB  Instruction Set: ARMv7-A 32-bit |
| Samsung Galaxy Win  (Android 4.1.2 Jelly Bean) | Qualcomm MSM8625Q Snapdragon 200  Quad-core 1.2 GHz Cortex-A5  Level 2 (L2) Cache Memory: 1024 KB |

Şekil 20: Test Edilen Cihazlar CPU

**4.2. Samsun Galaxy Note 4 ‘ün Elde Ettiği Sonuçlar**

**CPU (Çalışma Süreleri):**

|  |  |  |
| --- | --- | --- |
| Integer Calculation(1000. Sayı) | Matrix Multiplication | Matrix Inverse |
| 3.18 saniye | 0.261 milisaniye | 0.27 saniye |

|  |  |  |
| --- | --- | --- |
| Matrix Equations | Matrix RRF | Floating Point Calculation |
| 0.099 milisaniye | 0.031 milisaniye | DegtoRad: 0.016 milisaniye  RadtoDeg: 0.009 milisaniye  N. inci Kök: 0.3255 milisaniye |

|  |  |  |
| --- | --- | --- |
| Quicksort | Dijkstra | FFT |
| 0.469 milisaniye | 0.027 milisaniye | 0.0145 ms per iteration |

|  |  |  |
| --- | --- | --- |
| HTML Parser | AES | QR Code Scan |
| 9.19 saniye | Encrypt: 0.012 saniye  Decrypt: 0.789 milisaniye | 0.17 saniye |

|  |  |
| --- | --- |
| Canny Edge Detection | Gaussian Blur |
| 0.061 saniye | 0.059 saniye |

|  |  |
| --- | --- |
| MFLOPS | 376.91 |

Şekil 21: Linpack Benchmark

Linpack Benchmark, Jack Dongarra tarafından bir doğrusal cebir rutini paketi ile üretildi. 1980'li yılların ortalarından itibaren süper bilgisayar performansına eğimli bilimsel bilgisayarlar için temel kriterlerden biri haline geldi.Burada kullanılan orijinal çift duyarlıklı C versiyonu 100x100 matris üzerinde çalışmaktadır. Performans, dy [i] = dy [i] + da \* dx [i]'a bağlı olan daxpy işlevindeki bir iç döngü tarafından yönetilir.Millions of Floating Point Operation Per Second (MFLOPS) cinsinden ölçülür.Bu bölüm Java'da ana kod olarak Android Native Development Kit kullanır.

**GPU Sonuçları:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Triangles | WireFrame | Shaded | Shaded+ | Textured |
| 9000+ | 24.95 | 25.44 | 18.89 | 16.57 |
| 18000+ | 13.34 | 13.34 | 11.37 | 9.58 |
| 36000+ | 6.48 | 6.81 | 6.29 | 4.58 |

Şekil 22: Java OpenGL Benchmark

Bu benchmark, Frames Per Second (FPS) grafik performansını ölçmek için OpenGL ile tüm Java kodlarını kullanır. Dört test, önce wireframes olarak 50 küpten oluşan bir arka plan çizer, ardından gölgelendirilir(colour shaded). Üçüncü test, döner plakaları da içeren, oluklu kenarları ve tavanı olan bir tünelin içine ve dışına küpleri görüntüler. Son test küplere ve plakalara dokular ekler. 50 küp, rasgele konumlar, renkler ve dönme ayarlarıyla, 15, 30 ve 60 kez yeniden çizilir. "Küp başına 6 x 2 üçgen ile üç test kümesi için çerçeve başına minimum üçgenler 9000, 18000 ve 36000'dir. Sonuçlar Frames Per Second olarak gösterilmektedir.

|  |  |  |
| --- | --- | --- |
| Test | Frames | FPS |
| Display PNG Bitmap Twice | 175 | 17.40 |
| Plus 2 SweepGradient Circles | 105 | 10.45 |
| Plus 200 Random Small Circles | 92 | 9.19 |
| Plus 320 Long Lines | 70 | 6.96 |
| Plus 4000 Random Small Circles | 38 | 3.80 |

Şekil 23: Java Draw Benchmark

Bu benchmark, Frames Per Second (FPS) grafik performansını ölçer. Beş test, sürekli değişen renk tonlarının arka planını çizer. Test 1, her kare için hareket eden bitmapler iki kez bir PNG dosyası yükler. Test 2, tekrar hareket eden 2 SweepGradient daire oluşturur. Test 3, ekranın ortasına 200 rastgele küçük daire çizer. Test 4, her bir tarafın ortasından diğer tarafa 80 renk çizer, yine değişen renkler ile. Test 5, Test 3 ile aynı küçük rasgele daireleri çizer ancak 4000 adet ile ekranı doldurur. Her test yaklaşık 10 saniye sürer.

**RAM Sonuçları:**

Aşağıda 2 ayri benchmark sonuçları verilmiştir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memory | Serial | Serial | Random | Random |
| KBytes | Read | Read/Write | Read | Read/Write |
| 16 | 3406 | 3354 | 3990 | 3398 |
| 32 | 4102 | 3429 | 3426 | 3287 |
| 64 | 4058 | 3430 | 2303 | 2228 |
| 128 | 4096 | 3353 | 1821 | 1648 |
| 256 | 4049 | 3379 | 1510 | 1029 |
| 512 | 3933 | 3362 | 1332 | 851 |
| 1024 | 3968 | 3268 | 1234 | 797 |
| 4096 | 2631 | 2012 | 273 | 265 |
| 16384 | 2756 | 2257 | 138 | 117 |
| 65536 | 2763 | 2272 | 45 | 43 |

Şekil 24: MBytes/Second Transferring 4 Byte Words

Bu benchmark, MBytes Per Second'de önbellek ve bellekten veri aktarım hızları elde etmek için artan veri boyutlarında dört test gerçekleştirir. Aynı program yapısını kullanarak, 32 bit tamsayıları kullanarak okuma ve yazma testleriyle seri ve rasgele adres seçimleri kullanılır. Asıl amaç, rasgele erişim kullanarak performansın ne kadar yavaş olabileceğini göstermektir. Burada hız, verilerin çoğunun kullanılmadığı burst'lerde okuma ve yazma ve önceki önbelleklerin boyutundan önemli ölçüde etkilenebilir.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Memory |  |  |  |  |  |  |
| KBytes | Double | Single | Integer | Double | Single | Integer |
| 16 | 3459 | 1780 | 1789 | 3856 | 1782 | 1948 |
| 32 | 3143 | 1678 | 1778 | 3701 | 1756 | 1918 |
| 64 | 3362 | 1765 | 1779 | 3917 | 1777 | 1975 |
| 128 | 3364 | 1835 | 1820 | 3989 | 1808 | 2007 |
| 256 | 3425 | 1811 | 1822 | 3968 | 1765 | 2070 |
| 512 | 3822 | 2017 | 2045 | 4467 | 2010 | 2211 |
| 1024 | 3643 | 2000 | 1992 | 4107 | 1969 | 2154 |
| 4096 | 3318 | 1901 | 1912 | 3565 | 1806 | 2030 |
| 16384 | 3382 | 1929 | 1913 | 3634 | 1869 | 2053 |
| 65536 | 3299 | 1864 | 1922 | 3539 | 1796 | 2019 |

Şekil 25: Reading Speed in MBytes/Second

Bu benchmark, 2 x 8 KB - 2 x 32 MB büyüklüğündeki önbellek ve RAM verilerinde hesaplamalar yaparak saniyede Mega Bayt cinsinden veri okuma hızlarını ölçer. Hesaplamalarda x [m] = x [m] + s \* y [m] ve x [m] = x [m] + y [m] çift ve tek duyarlı Floating Point kullanılır ve x [m] = x [m] + s + y [m] ve x [m] = x [m] + y [m] tamsayıdır. Millions of Floating Point Operation Per Second (MFLOPS) hızı, çift duyarlı için MB / saniyeyi 8 ve 16'ya bölerek hesaplanabilir tek için ve 4 ve 8'e bölünür. Tamsayı testlerinin listelerinde Millions of Instruction Per Second'ın MB / saniyeyi 0,78'e 2 eklenerek ve 0,66 ile çarparak bulunabileceğini göstermektedir. Önbellek boyutları, bellek kullanımı değiştikçe değişen performansla gösterilir.

|  |  |
| --- | --- |
| Heap Allocation | Stack Allocation |
| 0.009 milisaniye | 0.011 milisaniye |

Şekil 26: Heap-Stack Allocation

**Depolama Sonuçları:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MB | Write1 | Write2 | Write3 | Read1 | Read2 | Read3 |
| 8 | 208.1 | 258.6 | 276.0 | 882.7 | 905.5 | 913.5 |
| 16 | 278.9 | 307.9 | 290.4 | 1067.2 | 1131.6 | 1162.2 |

Şekil 27: Mbytes/Second Cinsinden Okuma-Yazma Hızları

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 124.6 | 472.6 | 80256.82 | 21159.543 |

Şekil 28: Android için Elde Edilen Sonuçlar

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 195.4 | 726.4 | 51177.074 | 13766.52 |

Şekil 29: Requery için Elde Edilen Sonuçlar

**4.3. Samsun Galaxy J3‘ün Elde Ettiği Sonuçlar**

**CPU (Çalışma Süreleri):**

|  |  |  |
| --- | --- | --- |
| Integer Calculation(1000. Sayı) | Matrix Multiplication | Matrix Inverse |
| 4.46 saniye | 0.122 milisaniye | 0.407 saniye |

|  |  |  |
| --- | --- | --- |
| Matrix Equations | Matrix RRF | Floating Point Calculation |
| 0.152 milisaniye | 0.030 milisaniye | DegtoRad: 0.016 milisaniye  RadtoDeg: 0.009 milisaniye  N. inci Kök: 1.861 milisaniye |

|  |  |  |
| --- | --- | --- |
| Quicksort | Dijkstra | FFT |
| 0.213 milisaniye | 0.030 milisaniye | 0.0225 ms per iteration |

|  |  |  |
| --- | --- | --- |
| HTML Parser | AES | QR Code Scan |
| 20.92 saniye | Encrypt: 0.012 saniye  Decrypt: 0.427 milisaniye | 0.251 saniye |

|  |  |
| --- | --- |
| Canny Edge Detection | Gaussian Blur |
| 0.1 saniye | 0.144 saniye |

|  |  |
| --- | --- |
| MFLOPS | 89.41 |

Şekil 30: Linpack Benchmark

**GPU Sonuçları:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Triangles | WireFrame | Shaded | Shaded+ | Textured |
| 9000+ | 16.92 | 17.29 | 13.02 | 10.48 |
| 18000+ | 8.74 | 8.91 | 7.66 | 5.98 |
| 36000+ | 4.48 | 4.55 | 4.20 | 3.23 |

Şekil 31: Java OpenGL Benchmark

|  |  |  |
| --- | --- | --- |
| Test | Frames | FPS |
| Display PNG Bitmap Twice | 308 | 30.79 |
| Plus 2 SweepGradient Circles | 228 | 22.71 |
| Plus 200 Random Small Circles | 189 | 18.89 |
| Plus 320 Long Lines | 116 | 11.57 |
| Plus 4000 Random Small Circles | 39 | 3.83 |

Şekil 32: Java Draw Benchmark

**RAM Sonuçları:**

Aşağıda 2 ayri benchmark sonuçları verilmiştir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memory | Serial | Serial | Random | Random |
| KBytes | Read | Read/Write | Read | Read/Write |
| 16 | 1782 | 1747 | 1722 | 1616 |
| 32 | 1706 | 1670 | 1311 | 1337 |
| 64 | 1661 | 1632 | 786 | 875 |
| 128 | 1707 | 1652 | 630 | 719 |
| 256 | 1661 | 1639 | 576 | 659 |
| 512 | 1438 | 1218 | 227 | 283 |
| 1024 | 1204 | 1087 | 89 | 114 |
| 4096 | 1130 | 1089 | 55 | 72 |
| 16384 | 1130 | 1088 | 49 | 66 |
| 65536 | 1108 | 1079 | 44 | 58 |

Şekil 33: MBytes/Second Transferring 4 Byte Words

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Memory |  |  |  |  |  |  |
| KBytes | Double | Single | Integer | Double | Single | Integer |
| 16 | 804 | 451 | 621 | 1100 | 556 | 736 |
| 32 | 801 | 450 | 619 | 1094 | 554 | 733 |
| 64 | 749 | 433 | 588 | 1000 | 528 | 688 |
| 128 | 736 | 428 | 581 | 978 | 522 | 678 |
| 256 | 731 | 426 | 576 | 972 | 518 | 673 |
| 512 | 576 | 366 | 474 | 718 | 433 | 539 |
| 1024 | 363 | 269 | 319 | 415 | 304 | 348 |
| 4096 | 324 | 248 | 287 | 365 | 277 | 315 |
| 16384 | 322 | 246 | 285 | 362 | 275 | 313 |
| 65536 | 312 | 246 | 285 | 361 | 275 | 312 |

Şekil 34: Reading Speed in MBytes/Second

|  |  |
| --- | --- |
| Heap Allocation | Stack Allocation |
| 0.030 milisaniye | 0.030 milisaniye |

Şekil 35: Heap-Stack Allocation

**Depolama Sonuçları:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MB | Write1 | Write2 | Write3 | Read1 | Read2 | Read3 |
| 8 | 208.6 | 210.0 | 100.8 | 489.1 | 520.6 | 535.8 |
| 16 | 209.1 | 45.5 | 35.8 | 569.1 | 595.0 | 563.3 |

Şekil 36: Mbytes/Second Cinsinden Okuma-Yazma Hızları

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 283.2 | 802.6 | 35285.816 | 124459.507 |

Şekil 37: Android için Elde Edilen Sonuçlar

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 395.0 | 1281.4 | 25316.455 | 7803.9644 |

Şekil 38: Requery için Elde Edilen Sonuçlar

**4.3. Samsun Galaxy Win‘in Elde Ettiği Sonuçlar**

**CPU (Çalışma Süreleri):**

|  |  |  |
| --- | --- | --- |
| Integer Calculation(1000. Sayı) | Matrix Multiplication | Matrix Inverse |
| 1.92 saniye | 0.865 milisaniye | 0.47 saniye |

|  |  |  |
| --- | --- | --- |
| Matrix Equations | Matrix RRF | Floating Point Calculation |
| 0.208 milisaniye | 0.068 milisaniye | DegtoRad: 0.016 milisaniye  RadtoDeg: 0.008 milisaniye  N. inci Kök: 0.041 milisaniye |

|  |  |  |
| --- | --- | --- |
| Quicksort | Dijkstra | FFT |
| 2.226 milisaniye | 0.205 milisaniye | 0.0036 ms per iteration |

|  |  |  |
| --- | --- | --- |
| HTML Parser | AES | QR Code Scan |
| 5.97 saniye | Encrypt: 0.0035 saniye  Decrypt: 0.016 saniye | 0.13 saniye |

|  |  |
| --- | --- |
| Canny Edge Detection | Gaussian Blur |
| 0.09 saniye | 0.022 saniye |

|  |  |
| --- | --- |
| MFLOPS | 56.06 |

Şekil 39: Linpack Benchmark

**GPU Sonuçları:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Triangles | WireFrame | Shaded | Shaded+ | Textured |
| 9000+ | 19.15 | 20.85 | 15.20 | 12.60 |
| 18000+ | 10.31 | 11.10 | 9.38 | 7.48 |
| 36000+ | 5.29 | 5.63 | 5.19 | 4.08 |

Şekil 40: Java OpenGL Benchmark

|  |  |  |
| --- | --- | --- |
| Test | Frames | FPS |
| Display PNG Bitmap Twice | 291 | 29.07 |
| Plus 2 SweepGradient Circles | 202 | 20.17 |
| Plus 200 Random Small Circles | 160 | 15.97 |
| Plus 320 Long Lines | 118 | 11.78 |
| Plus 4000 Random Small Circles | 32 | 3.14 |

Şekil 41: Java Draw Benchmark

**RAM Sonuçları:**

Aşağıda 2 ayri benchmark sonuçları verilmiştir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memory | Serial | Serial | Random | Random |
| KBytes | Read | Read/Write | Read | Read/Write |
| 16 | 673 | 1245 | 1310 | 1184 |
| 32 | 1125 | 1108 | 855 | 900 |
| 64 | 829 | 842 | 442 | 521 |
| 128 | 781 | 780 | 349 | 408 |
| 256 | 764 | 754 | 311 | 367 |
| 512 | 720 | 580 | 124 | 159 |
| 1024 | 685 | 407 | 62 | 71 |
| 4096 | 644 | 342 | 40 | 47 |
| 16384 | 669 | 381 | 34 | 41 |
| 65536 | 587 | 397 | 28 | 35 |

Şekil 42: MBytes/Second Transferring 4 Byte Words

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Memory |  |  |  |  |  |  |
| KBytes | Double | Single | Integer | Double | Single | Integer |
| 16 | 536 | 329 | 469 | 732 | 421 | 557 |
| 32 | 500 | 326 | 444 | 665 | 397 | 518 |
| 64 | 429 | 296 | 389 | 551 | 352 | 446 |
| 128 | 413 | 288 | 377 | 526 | 288 | 429 |
| 256 | 411 | 285 | 371 | 520 | 339 | 425 |
| 512 | 398 | 279 | 352 | 503 | 328 | 414 |
| 1024 | 389 | 274 | 352 | 480 | 313 | 400 |
| 4096 | 414 | 293 | 295 | 465 | 318 | 427 |
| 16384 | 415 | 274 | 371 | 512 | 346 | 429 |
| 65536 | 415 | 294 | 379 | 512 | 347 | 428 |

Şekil 43: Reading Speed in MBytes/Second

|  |  |
| --- | --- |
| Heap Allocation | Stack Allocation |
| 22.15 milisaniye | 0.011 milisaniye |

Şekil 44: Heap-Stack Allocation

**Depolama Sonuçları:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MB | Write1 | Write2 | Write3 | Read1 | Read2 | Read3 |
| 8 | 153.7 | 160.2 | 50.9 | 276.4 | 330.3 | 385.6 |
| 16 | 147.8 | 30.1 | 20.4 | 349.8 | 311.1 | 329.7 |

Şekil 45: Mbytes/Second Cinsinden Okuma-Yazma Hızları

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 325.4 | 1485.6 | 30731.406 | 6731.287 |

Şekil 46: Android için Elde Edilen Sonuçlar

|  |  |  |  |
| --- | --- | --- | --- |
| Read AVG | Write AVG | Rows/second | Inserts/second |
| 257.2 | 1841.4 | 38880.246 | 5430.651 |

Şekil 47: Requery için Elde Edilen Sonuçlar

1. **PROJE DETAYLARI**

Yapılan mobil benchmark uygulaması Android üzerinde Java ve C programlama dili kullanılarak JNI ile geliştirilmiştir ve test edilen cihazın 4 ayrı donanımını test etmektedir:

**1.)CPU**

**2.)GPU**

**3.)RAM**

**4.)Depolama**

**5.1. CPU Detayları**

Android tabanlı mobil cihazın CPU donanımın hızını ve performansını test etmek için çeşitli algoritmalar kullanılmıştır. Bu algoritmaların işlevleri aşağıda açıklanmış:

**1.)Integer Hesaplaması:** Integer hesaplaması, günümüz işlemciler için tartışılmaz bir gerekliliktir. İster Masaüstü PC’lerde ister gömülü sistemlerde ister mobil cihazlarda, çeşitli platformlar için geliştirilen yazılımlar Integer verilerle alakalı hesaplamaları içermektedir. Integer hesaplaması, işlemci için vazgeçilmez bir ihtiyaçtır ve bu yüzden her bir benchmark uygulaması içinde Integer Hesaplaması hakkında bir micro benchmark’ın geliştirilmesi artık bir zorunluluk haline gelmiştir.

Integer Hesaplaması, temel olarak 1000. Asal sayıyı bulmayı amaçlamaktadır ve işlemcinin bu işlemi tamamlaması için geçen süre ekran üzerinde gösterilmiştir.

**2.)Matris İşlemleri:** Günümüz işlemciler için en stresli işlemlerden bir tanesi matrislerle yapılan işlemlerdir. Ve bunlardan bir tanesi ise matris çarpımı işlemidir. Matris çarpımı işlemi sırasında çok fazla matematiksel işlem ve bundan dolayı da çok fazla CPU kaynağı kullanıldığı için günümüz benchmark uygulamalarında yeri vazgeçilmez olmuştur.

Bu işlemde, elemanları rastgele atılmış 7x7’lik 2 matrisin çarpımı gerçekleştirilmiş ve işlemcinin bu işlemi tamamlaması için geçen süre ekran üzerinde gösterilmiştir.

Diğer kullanılan matris işlemleri ise; matris tersini alma, matris kullanarak normal ve karmaşık denklemleri çözme ve RRF algoritmasıdır.

**3.)Floating Point Hesaplaması:** Floating Point Hesaplaması, günümüz uygulamalarda çokça kullanılan işlemlerdir. Ve hangi platform olursa olsun günümüz işlemciler için tartışılmaz bir gerekliliktir.

Dereceden radyana, radyandan dereceye ve n’inci kök işlemleri ile birlikte işlemcinin performansı ölçülmüş ve işlemcinin bu işlemleri çalıştırırken harcadığı zaman ekranda gösterilmiştir.

**4.)Quicksort algoritması:** Verinin hafızada sıralı tutulması için geliştirilen [sıralama algoritmalarından (sorting algorithms)](http://www.bilgisayarkavramlari.com/2008/08/09/siralama-algoritmalari-sorting-algorithms/) bir tanesidir. Basitçe sıralanacak olan dizideki orta noktada (mean) bulunan bir sayıyı seçerek diğer bütün sayıları bu orta sayıdan büyük veya küçük diye sınıflayarak sıralama yapmayı hedeflemektedir. [42]

Bilgisayar bilimlerinde, çok sayıda sıralama algoritması mevcuttur. Bu algoritmayı kullanılmasının herhangi bir özel nedeni yoktur. Amaç, işlemcinin bir veriyi sıralarken nasıl bir performans gösterdiğini test etmektir.

**5.)Dijkstra Algroitması:** Bilgisayar bilimlerinde kullanılan ve algoritmayı literatüre kazandıran kişinin ismini taşıyan dijkstra algoritması, verilen bir [şekilde (graph)](http://www.bilgisayarkavramlari.com/2008/08/01/graf-sekil-graph/) en kısa yolu (shortest path) bulmak için kullanılır.[43]

Bilgisayar bilimlerinde, çok sayıda kısa yok bulma algoritması mevcuttur. Ve Djikstra algoritması çokça kullanılan bir algoritma olduğu için, uygulamada işlemcinin kısa yol bulma problemini çözerken nasıl bir performans gösterdiğini test etmektir.

**6.)FFT (Fast Fourier Transform):** Hızlı Fourier dönüşümü titreşim analizinde kullanılan, istatistik tabanlı, matematiksel bir işlemdir. Karışık sinyal yumaklarını ayrıştırır ve hangi frekansta ne şiddette bir titreşim olduğunu gösterir. Kısaca FFT sinyallerimizi zaman alanından frekans alanına geçirirken kullandığımız bir işlemdir.[44]

FFT algoritması, çok fazla CPU kaynaklarını harcadığı için, günümüz işlemciler için bir micro benchmark halini almıştır. İşlemcinin bu algoritmayı çalıştırırken sergilemiş olduğu performans ekranda gösterilmiştir.

**7.)HTML Parser:** Günümüz mobil uygulamalarda çok HTML Parse yani bir web sitesinden veri çekme işlemi çok sık kullanılmaktadır. Örneğin bir haber sitesi, yayınladığı haberlerde bulunan verilerin mobil uygulamalar tarafından çekilebilmesi için özel bir servis sunmadığı durumlarda HTML Parse işlemi kullanılır. HTML Parse işleminde, elde edilen HTML etiketlerindeki veriler temizlenir bu veriler ekranda gösterilir. İşlemcinin bu veri çekme ve temizleme işlemi sırasında sergilediği performans test edilen mobil cihaz üzerinde gösterilmiştir.

**8.)AES:** Bilişim Sektöründe Siber Güvenlik büyük bir önem arz etmektedir. Siber Güvenliğin bir parçası olan Şifreleme algoritmaları ise gün geçtikçe geliştirilmiş ve Bilgisayar Bilimlerinin vazgeçilmez bir konusu olmuştur. Şifreleme Algoritmaları karmaşık matematiksel işlemler içerdiği için masaüstü ve mobil alanda geliştirilmiş benchmark uygulamalarında yerini almış ve günümüz modern işlemcilerin şifreleme performansını ölçmek için vazgeçilmez bir kıyaslama aracı olmuştur.

Mobil cihazın şifreleme performansını ölçmek için AES Şifreleme algoritması kullanılmış ve işlemcinin sergilemiş olduğu performans ekranda gösterilmiştir.

**9.) QR Code Scan:** Barkod Tarayıcı, CD'lerde, kitaplarda ve diğer ürünlerde barkodları tarar. Barkod Tarayıcı, açık kaynaklı bir barkod görüntü işleme kütüphanesi olan ZXing kütüphanesini kullanır. Mobil cihazlarda yerleşik kamerayı kullanarak barkodları çözer. Desteklenen formatlar UPC-A / E, EAN-8/13, QR kodu ve diğerleridir. Barkod Tarayıcı, QR kod dedektörü için kendi algoritmasını uygular. Algoritmanın temel unsurları aşağıdaki gibidir.

• Binary veriyi alabilmek için Gray-Scale verisi üzerinde eşik (threshold) kararları almak

• Ham görüntü piksellerini modüllere örnekleme; Her QR kod modülü ham görüntü pikselleriyle yayılmalıdır.

• Hizalama düzenlerinin (pattern) tespiti.

• Görüntü döndürme ve doğru perspektif bozulma. [45]

**10.) Canny Edge Detection ve Gaussian Blur:** Görüntü İşleme, Bilişim sektörünün her alanında kullanılmaktadır. Piyasada birçok görüntü işleme kütüphanesi bulunmaktadır. Popüler görüntü işleme kütüphanesi olan OpenCV, hem mobil hem masaüstü platformlarda kullanıcılara birçok görüntü işleme fonksiyonu sunmaktadır. Görüntü işlemenin temelinde birçok karmaşık matematiksel işlemler yapıldığı için çok fazla işlemci kaynağı harcanmaktadır. Dolayısıyla modern işlemcilerin performansını ölçmek için vazgeçilmez bir kıyaslama tekniği haline gelmiştir.

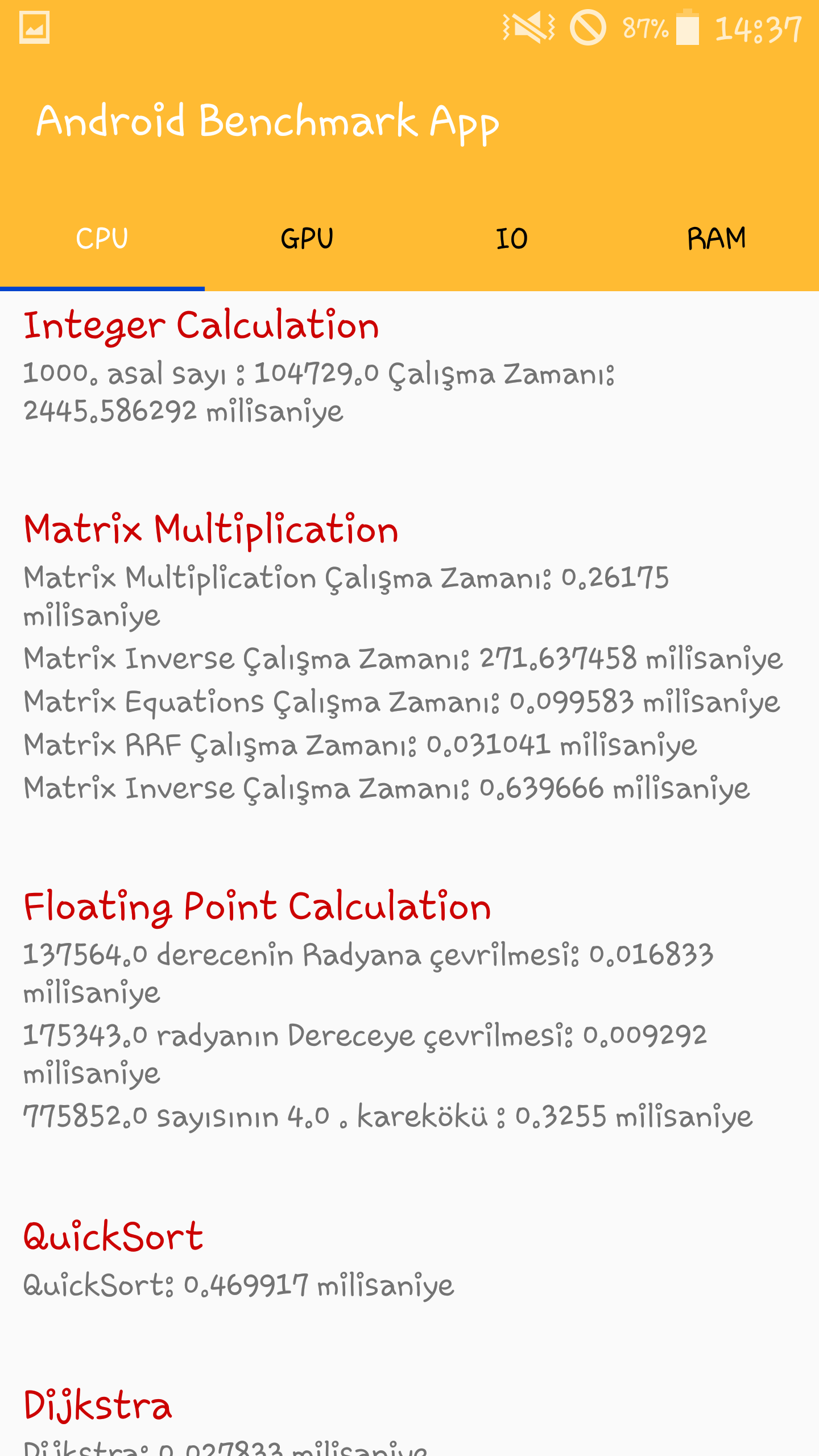
Burada işlemcinin performansını ölçmek için Canny Edge Detection ve Gaussian Blur tekniği kullanılmış ve elde edilen test sonuçları mobil cihazın ekranında gösterilmiştir.

**11.) Linpack Benchmark:** Linpack Benchmark, Jack Dongarra tarafından bir doğrusal cebir rutini paketi ile üretildi. 1980'li yılların ortalarından itibaren süper bilgisayar performansına eğimli bilimsel bilgisayarlar için temel kriterlerden biri haline geldi. Burada kullanılan orijinal çift duyarlıklı C versiyonu 100x100 matris üzerinde çalışmaktadır. Performans, dy [i] = dy [i] + da \* dx [i]'a bağlı

olan daxpy işlevindeki bir iç döngü tarafından yönetilir. Millions of Floating Point Operation Per Second (MFLOPS) cinsinden ölçülür. Bu bölüm Java'da ana kod olarak Android Native Development Kit kullanır.



Şekil 48: CPU Örnek Çalışma-1



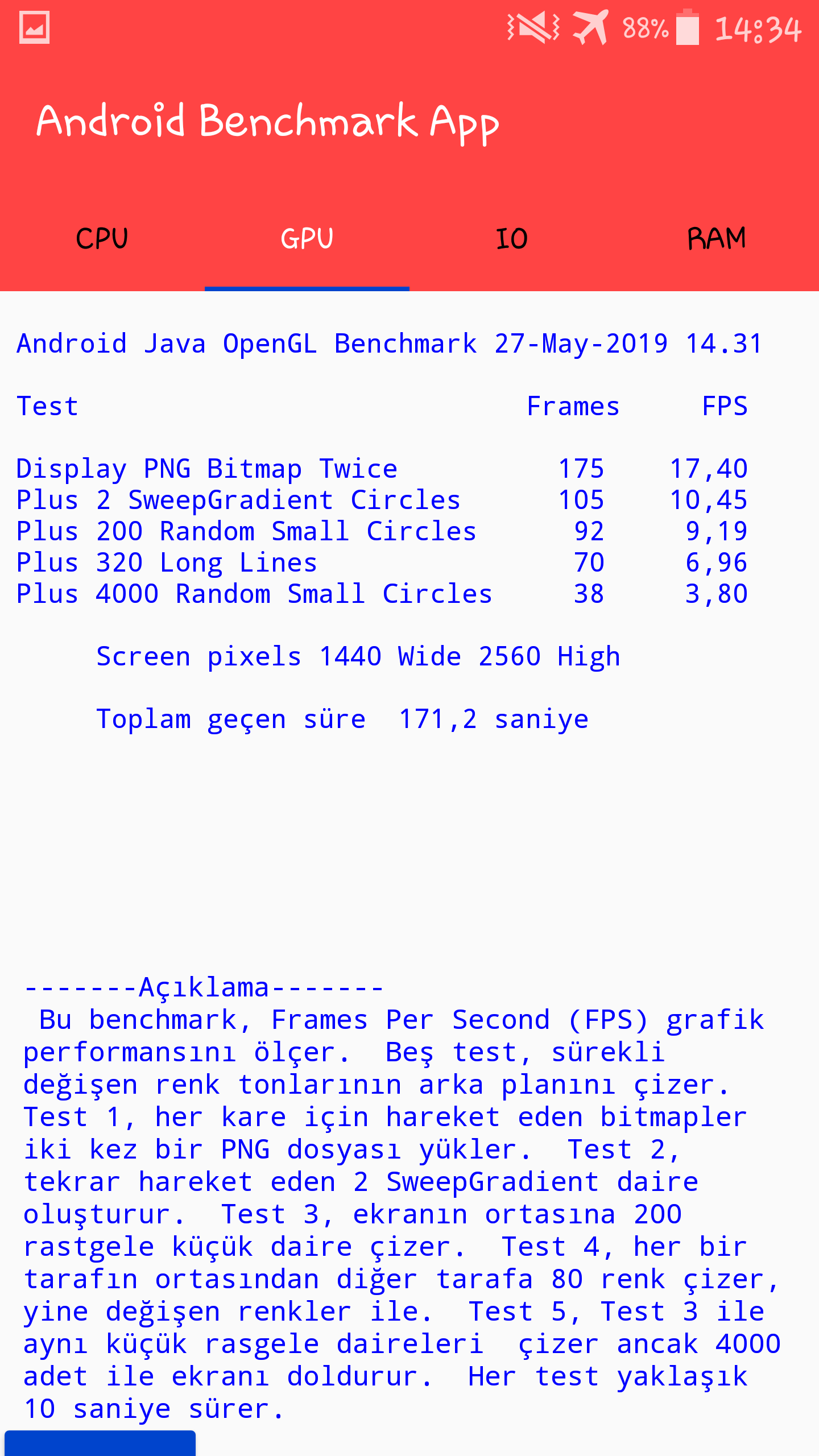
Şekil 49: CPU Örnek Çalışma-2

**5.2. GPU Detayları**

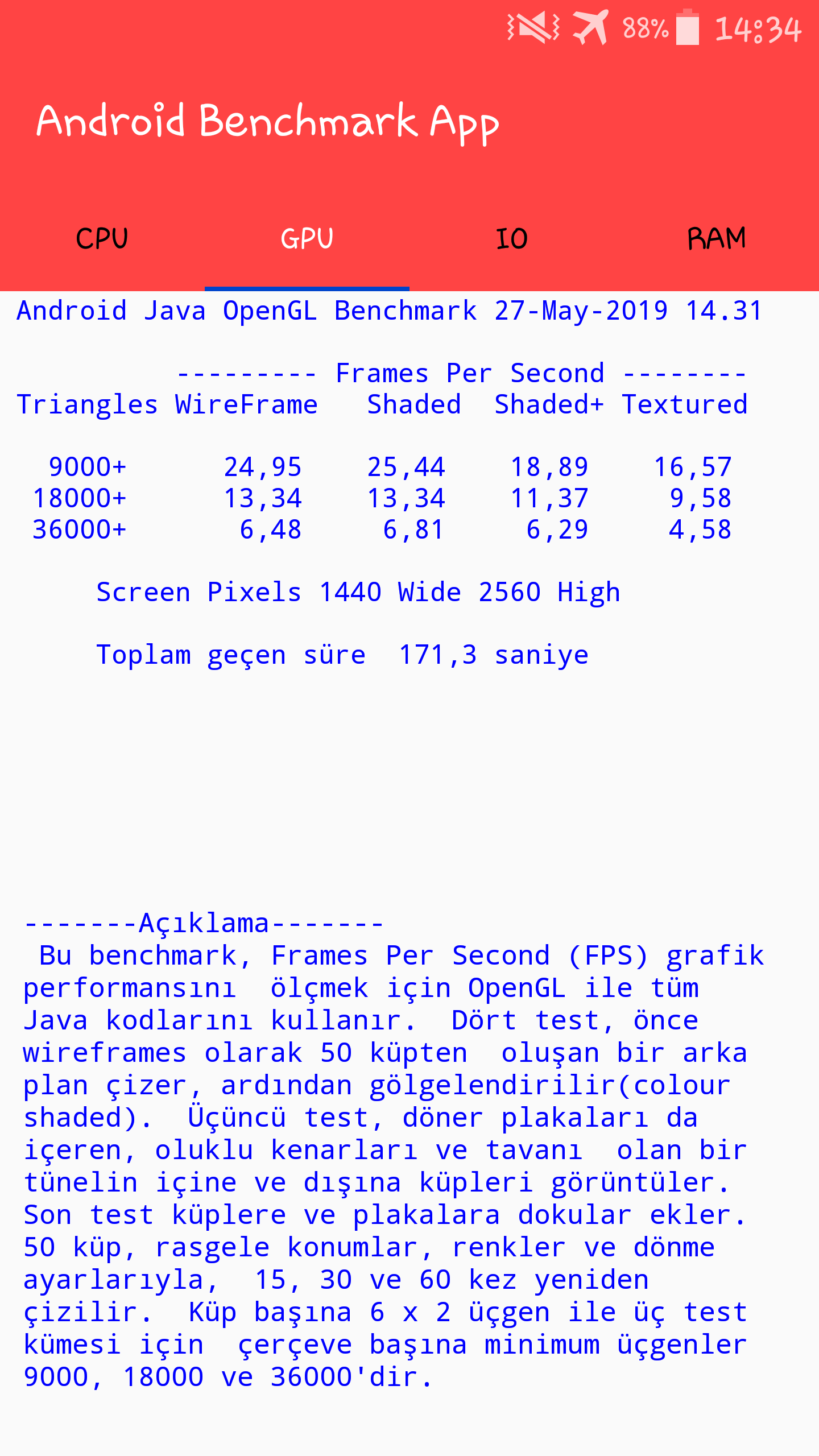
Mobil cihazlarda yüksek kaliteli grafiklere olan artan taleple, GPU'lar artık akıllı telefonlar ve tabletler gibi mobil cihazlar için vazgeçilmez birimler olmuştur. Özellikle, yüksek çözünürlüklü ekrana sahip mobil cihazlar için daha güçlü GPU'lar gerekmektedir. Sonuç olarak, GPU'lar genellikle geniş bir uygulama alanında kullanılır ve GPU performansı, mobil aygıtları seçmek için önemli ölçütlerden biri haline gelmiştir.

**1.) Java OpenGL Benchmark:** Bu benchmark, Frames Per Second (FPS) grafik performansını ölçmek için OpenGL ile tüm Java kodlarını kullanır. Dört test, önce wireframes olarak 50 küpten oluşan bir arka plan çizer, ardından gölgelendirilir(colour shaded). Üçüncü test, döner plakaları da içeren, oluklu kenarları ve tavanı olan bir tünelin içine ve dışına küpleri görüntüler. Son test küplere ve plakalara dokular ekler. 50 küp, rasgele konumlar, renkler ve dönme ayarlarıyla, 15, 30 ve 60 kez yeniden çizilir. Küp başına 6 x 2 üçgen ile üç test kümesi için çerçeve başına minimum üçgenler 9000, 18000 ve 36000'dir.

**2.)Java Draw Benchmark:** Bu benchmark, Frames Per Second (FPS) grafik performansını ölçer. Beş test, sürekli değişen renk tonlarının arka planını çizer. Test 1, her kare için hareket eden bitmapler iki kez bir PNG dosyası yükler. Test 2, tekrar hareket eden 2 SweepGradient daire oluşturur. Test 3, ekranın ortasına 200 rastgele küçük daire çizer. Test 4, her bir tarafın ortasından diğer tarafa 80 renk çizer, yine değişen renkler ile. Test 5, Test 3 ile aynı küçük rasgele daireleri çizer ancak 4000 adet ile ekranı doldurur. Her test yaklaşık 10 saniye sürer.



Şekil 50: GPU Örnek Çalışma-1



Şekil 51: CPU Örnek Çalışma-2

**5.3. RAM Detayları**

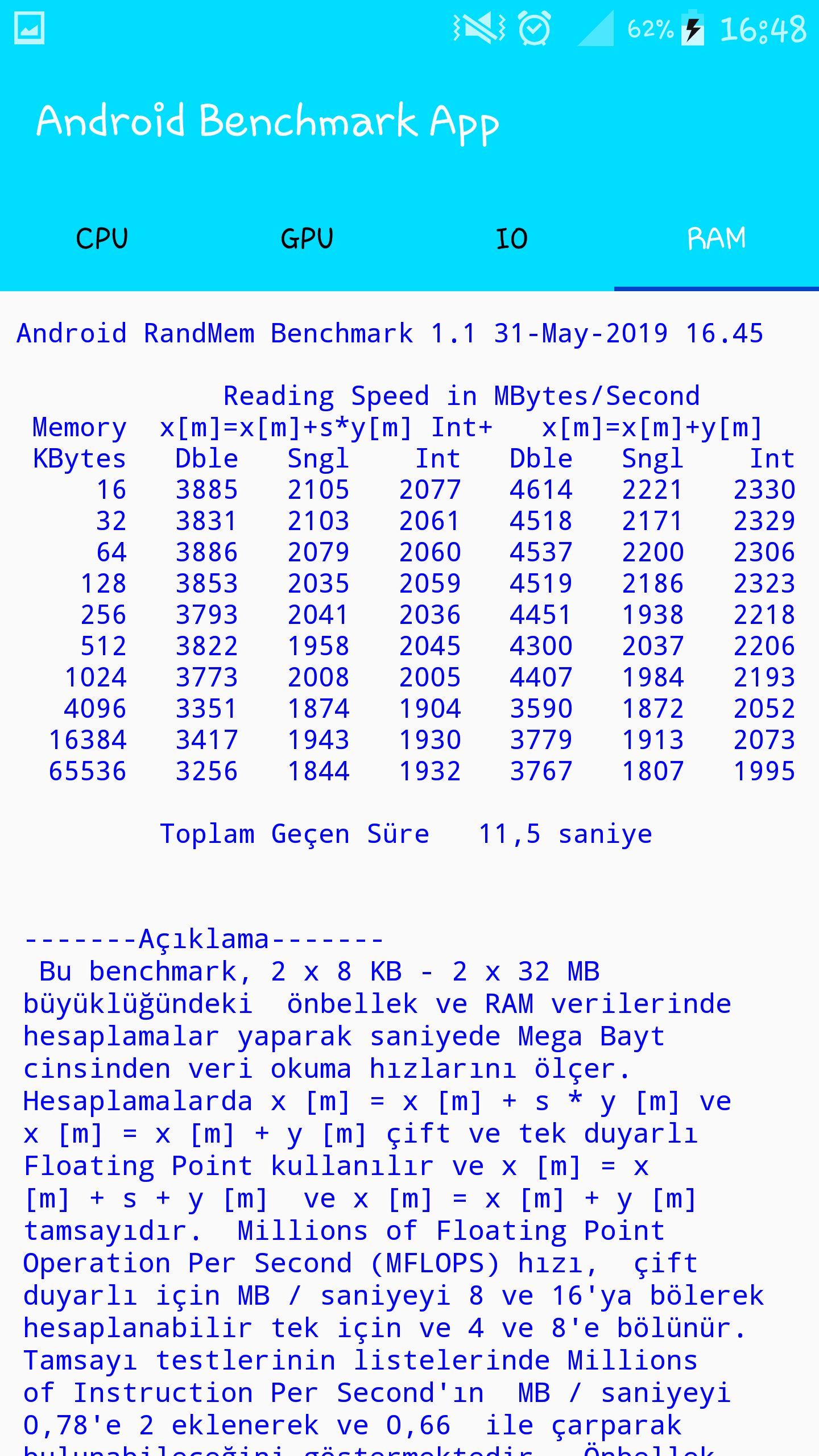
**1.)RandMem Benchmark 1:** İnsanda beyin, makinede motor hangi görevi yapıyorsa bilgisayarda da RAM o işlevi görür. Bir başka deyişle bilgisayarda çalışılan verilerin hızlı erişilebilirliğini artırmak için bilgisayar hafızasında tutmaya yarayan bir donanımdır[46].

Mobil ve masaüstü cihazların vazgeçilmez bir birimi olan RAM, her benchmark uygulamasında yerini almış ve RAM’in performansını ölçmek için çeşitli metodlar kullanılmaktadır.

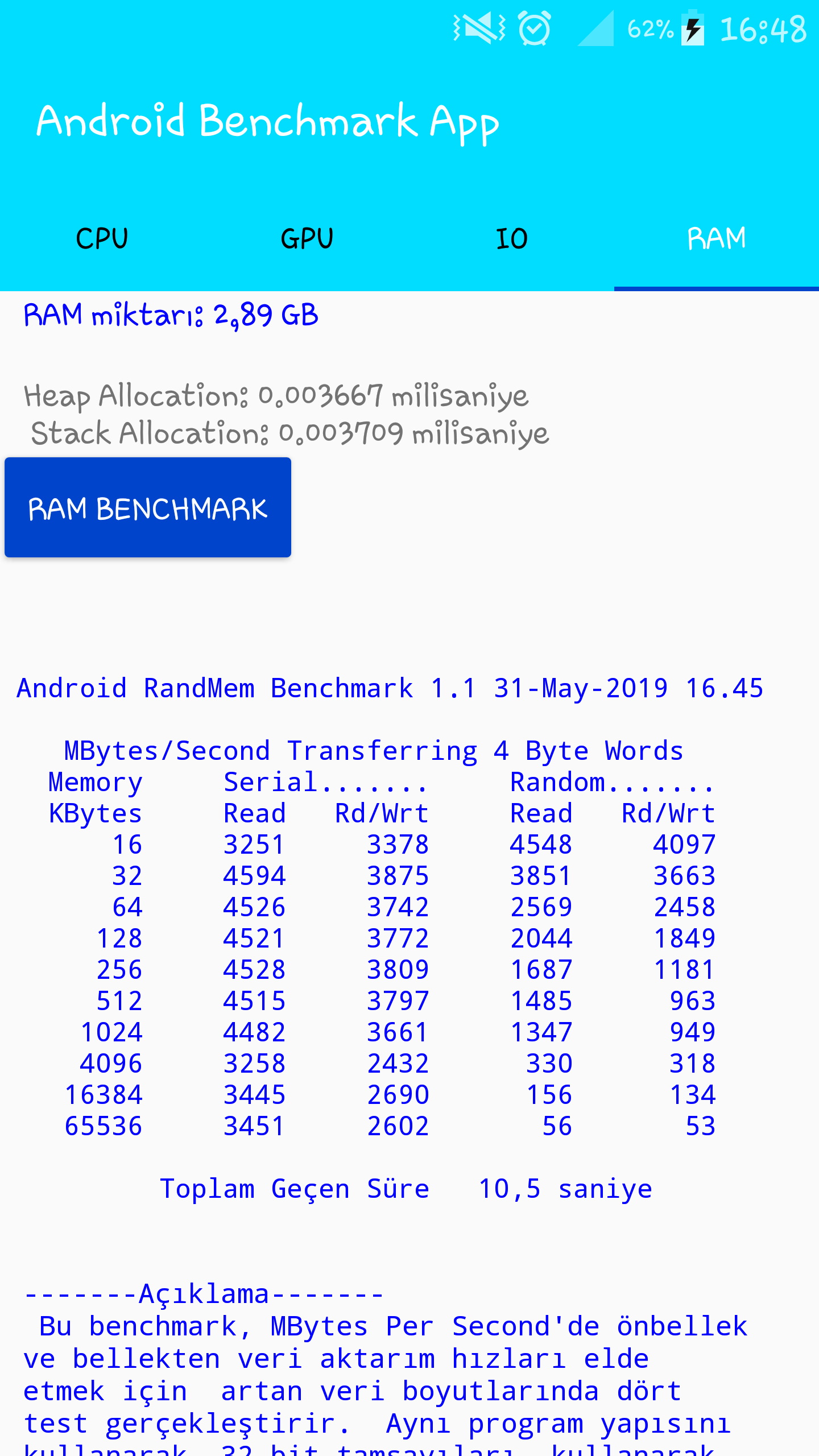
Bu benchmark, MBytes Per Second'de önbellek ve bellekten veri aktarım hızları elde etmek için artan veri boyutlarında dört test gerçekleştirir. Aynı program yapısını kullanarak, 32 bit tamsayıları kullanarak okuma ve yazma testleriyle seri ve rasgele adres seçimleri kullanılır. Asıl amaç, rasgele erişim kullanarak performansın ne kadar yavaş olabileceğini göstermektir. Burada hız, verilerin çoğunun kullanılmadığı burst'lerde okuma ve yazma ve önceki önbelleklerin boyutundan önemli ölçüde etkilenebilir.

**2.)RandMem Benchmark 2:** Bu benchmark, 2 x 8 KB - 2 x 32 MB büyüklüğündeki önbellek ve RAM verilerinde hesaplamalar yaparak saniyede Mega Bayt cinsinden veri okuma hızlarını ölçer. Hesaplamalarda x [m] = x [m] + s \* y [m] ve x [m] = x [m] + y [m] çift ve tek duyarlı Floating Point kullanılır ve x [m] = x [m] + s + y [m] ve x [m] = x [m] + y [m] tamsayıdır. Millions of Floating Point Operation Per Second (MFLOPS) hızı, çift duyarlı için MB / saniyeyi 8 ve 16'ya bölerek hesaplanabilir tek için ve 4 ve 8'e bölünür. Tamsayı testlerinin listelerinde Millions of Instruction Per Second'ın MB / saniyeyi 0,78'e 2 eklenerek ve 0,66 ile çarparak bulunabileceğini göstermektedir. Önbellek boyutları, bellek kullanımı değiştikçe değişen performansla gösterilir.

**1.)Heap Stack Allocation**: RAM üzerinde heap ve stack bölümlerinde veri için ayrılan yerin ne kadar sürede tamamlandığını ölçer.



Şekil 52: RAM Örnek Çalışma-1



Şekil 53: RAM Örnek Çalışma-2

**5.4. Depolama Detayları**

**1.)Depolama Birimi Benchmark:** Test 1 - 3 adet 8 ve 16 MB boyutlarındaki dosyaları yazıp okumaktadır.

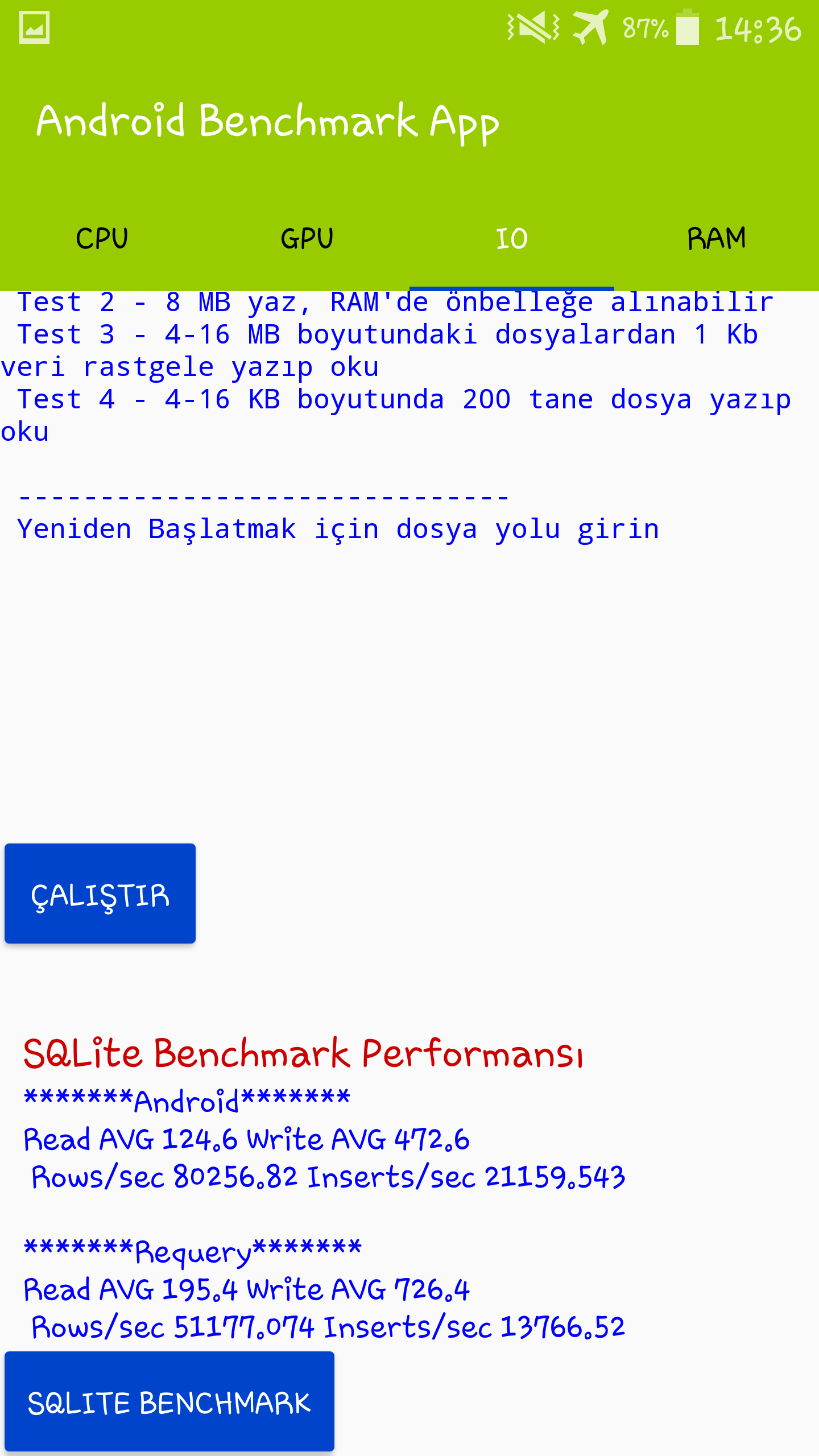
Test 2 – ‘8 MB yaz, RAM'de önbelleğe alınabilir’ işlemini yapmaktadır.

Test 3 - 4-16 MB boyutundaki dosyalardan 1 Kb veri rastgele yazıp okumaktadır.

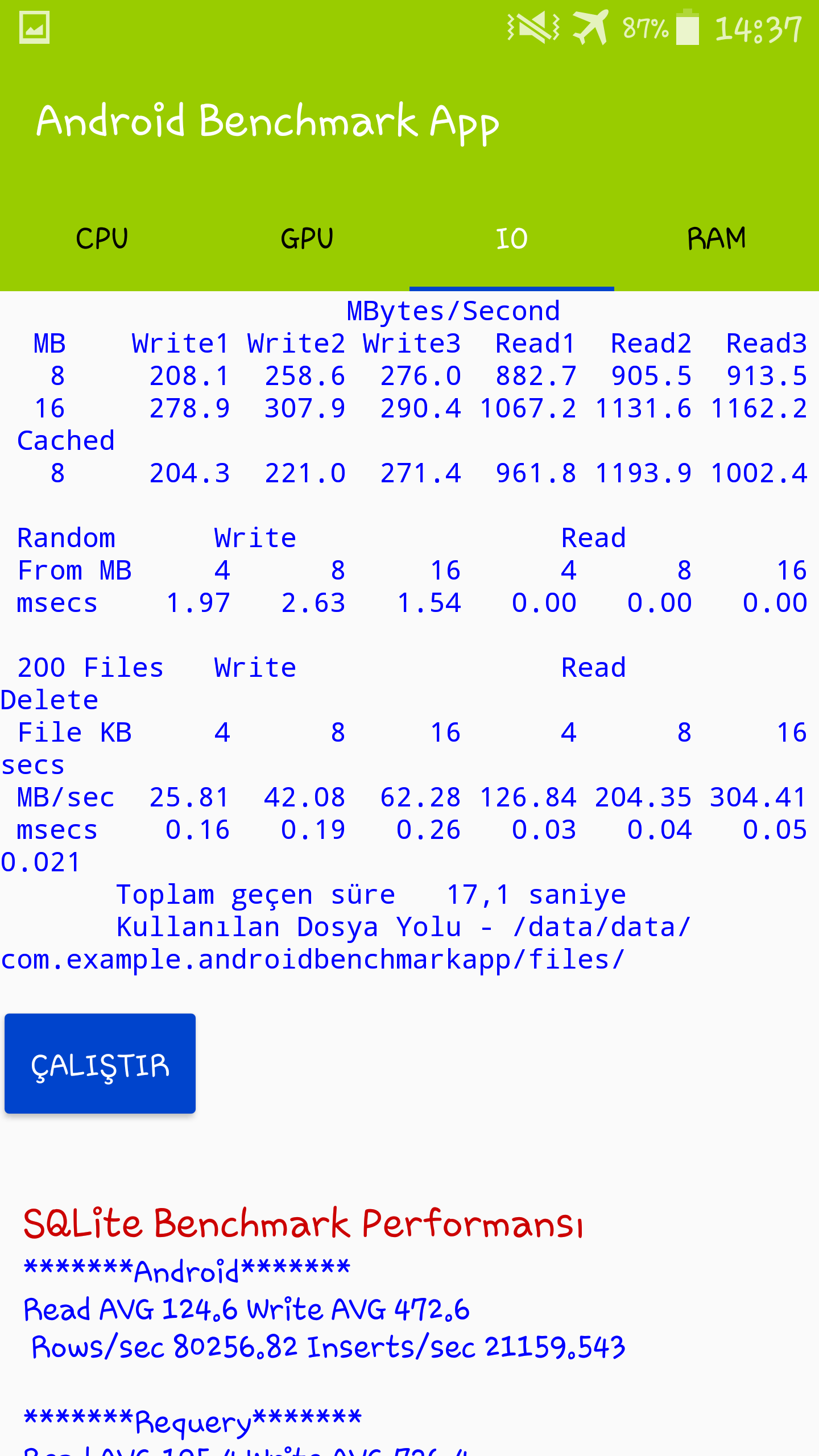
Test 4 - 4-16 KB boyutunda 200 tane dosya yazıp okumaktadır.

**2.)SQLite Benchmark:** Verilerin düzenli olarak saklandığı veritabanlarında, veriler okunurken ve yazılırken harddisk (depolama birimi) kullanılır. Dolayısıyla herhangi bir veritabanından veri okunurken veya veritabanına veri yazılırken elde edilen okuma-yazma hızı, üzerinde çalışılan makinenin sahip olduğu hard diskin okuma-yazma hızına bağlıdır. Bu yüzden hem mobil hem masaüstü platformlarda veritabanı teknolojisi sıkça kullanıldığı için, veritabanında bulunan veriler üzerinde okuma-yazma işlemleri mobil platformlar için vazgeçilmez bir micro benchmark konumuna gelmiştir.

Bu çalışmada, bir veritabanında bulunan veriler üzerinde okuma-yazma işlemleri gerçekleştirilmiş ve mobil cihazın sergilemiş olduğu performans sonuçları cihazın ekranında gösterilmiştir.



Şekil 54: Depolama Örnek Çalışma-1



Şekil 55: Depolama Örnek Çalışma-2

1. **SONUÇLAR**

Bu çalışma, günümüzde piyasada yaygın olan benchmark yazılımlarını ve bu yazılımların iyi-kötü yanlarını, Android İşletim Sistemi yüklü cihazlarda bulunan CPU, GPU ve depolama aygıtlarının sahip olduğu bileşenler ve bu aygıtların hangi kriterler altında test edilmesi gerektiği konusu hakkında geçmişte yapılmış çalışmaları ve bu konu hakkında yazılmış makaleleri sizlere aktarmış bulunmaktadır. Makale yazarları, kendi benchmark yazılımlarını en iyisi ve tam olarak tanımlamasalar da, bir benchmark yazılımında olması gereken temel kriterleri, yazılım geliştiricilerine rehber olması amacıyla aktarmaktadırlar. CPU , GPU , RAM , Depolama olmak üzere 4 farklı alanda SAMSUNG Galaxy Note 4, J3 ve WİN olmak üzere 3 değişik telefon kullanarak bu telefonlar arasında karşılaştırmalar yapılmaktadır. Bu karşılaştırmalar sonucunda telefonların birbirlerine göre daha farklı sonuçlar verdiği tespit edilip gerekli bilgiler resimler halinde gösterilmektedir. Sonuç olarak cihazlar arasındaki farklılıkların donanımlarıyla birlikte arttığı tespit edilip bu tespitler cihazın kullanımına göre az bir farkla değişiklikler göstermektedirler.

Gelecek çalışmalarda ise, Android İşletim Sistemi yüklü cihazlarda bulunan pil ve cihazın tarayıcı (JavaScript, HTML performansı vb.) performansı test edilecektir. Bu aygıtlar test edilip sonuçları karşılaştırılacak ve bu aygıtların hangi bileşenlerinin hangi kriterler altında test edilmesi gerektiği anlatılacaktır.

**KAYNAKLAR**

[1] Guthaus M., Ringenberg J., Ernst D., Austin T., Mudge T., Brown R.,

*Mibench: A free, commercially representative embedded benchmark suite,”* in Workload Characterization, 2001. WWC-4. 2001IEEE International Workshop on, 3 – 14, Aralık 2001

[2] URL-1: http://www.specbench.org/, Welcome to SPEC, The Standard Performance Evaluation Corporation, (25 Aralık 2018)

[3] URL-2: https://www.openhandsetalliance.com/, OHA Open Handset Alliance, (20 Aralık 2018)

[4] URL-3: https://www.basemark.com/product-catalog/ basemark-es-3-1/, BasemarkLtd.BasemarkES3.1, (2015)

[5] URL-4: https://gfxbench.com/, KishontiInformatics.GFXBench4.0, (2016)

[6] URL-5: http://www.future mark.com/benchmarks/3dmark/android, Futuremark Corporation 3DMark SlingShot Benchmark, (2015).

[7] Guthaus M., Ringenberg J., Ernst D., Austin T., Mudge T., Brown R., *Mibench: A free, commercially representative embedded benchmark suite,”* in Workload Characterization, 2001. WWC-4. 2001IEEE International Workshop on, 3 – 14, Aralık 2001

[8] Bienia C., Kumar S., Singh J. P., Li K., *The PARSEC Benchmark*

*Suite: Characterization and Architectural Implications*, Princeton

University, Tech. Rep., TR-811-08, 2008.

[9] Levy M., *Evaluating digital entertainment system performance*, Computer,

cilt-no. 38, basım sayısı. 7, 68 – 72, Temmuz 2005.

[10] URL-6: http://www.gpsbenchmark.com/the-mobile-app1, Mobile gps benchmark

[11] URL-7: http://www.jbenchmark.com/, Jbechmark, Kishonti Information Ltd.

[12] URL-8: http://www.kandroid.org/online-pdk/guide/instrumentation testing.html, GLBenchmark, Kishonti Information Ltd.

[13] Singh H., What does a processor do in a smartphone and what is it?, https://www.quora.com/What-does-a-processor-do-in-a-smartphone-and-what-is-it, (30 Aralık 2018)

[14] URL-9: http://www.fbreader.org/, FBReader

[15] URL-10: http://code.google.com/p/zxing/, ZXing (”Zebra Crossing”)

[16] URL-11: http://libgdx.badlogicgames.com/, libGDX

[17] URL-12: http://developer.android.com, Android Developers, Google

[18] Ginsburg D., Purnomo B., Shreiner D., Munshi A., *OpenGL ES 3.0 programming guide*, 2nd ed. Addison-Wesley Professional, 169-172, 2014

[19] Mittring M., *Finding next gen: Cryengine 2*, In: ACM SIGGRAPH 2007 courses, 97–121, 2007

[20] Shanmugam P., Arikan O., *Hardware accelerated ambient occlusion techniques on GPUs*, Proceedings of the 2007 symposium on interactive 3D graphics and games, ACM;, 73–80, 2007

[21] Sellers G., Wright RS., Haemel N., *OpenGL SuperBible: comprehensive tutorial and reference*, 7th ed, Addison-Wesley, 624-663, 2015.

[22] Gerasimov P., *Omnidirectional shadow mapping*, In: GPU Gems: programming techniques, tips, and tricks for real-time graphics, Oxford University Press, 193–204 [chap. 12], 2004

[23] [Olsson O., Assarsson U., Tiled shading, *J Graph GPU Game Tools,* 2011;15(4):235–51.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref4)

[24] [McGuire M., Bavoil L., Weighted blended order-independent transparency](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref7), [*J Comput Graph Technol,* 2013;2(2):122–41.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref7)

[25] [Greene N., Environment mapping and other applications of world projections](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref8), [*IEEE Comput Graph Appl,* 1986;6(11):21–9.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref8)

[26] McReynolds T., Blythe D., Fowle C., Grantham B., Hui S., Womack P., *Programming with OpenGL: Advanced rendering*, SIGGRAPH, cilt-no 97, 144–153, 1997

[27] [Lauterbach C., Mo Q., Manocha D., gProximity: hierarchical GPU-based opera-tions for collision and distance queries, *Comput Graph Forum,* *(EURO*-*GRAPHICS* *2010*), 2010;29(2):419–28.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref9)

[28] [Nah JH., Kim JW., Park J., Lee WJ., Park JS., Jung SY., HART: a hybrid](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref10) [architecture for ray tracing animated scenes, *IEEE Trans Vis Comput Graph*](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref10), [2015;21(3):389–401.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref10)

[29] [Mattausch O., Bittner J., Wimmer M., CHC](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref5)++[: coherent hierarchical culling](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref5) [revisited, *Comput Graph Forum (EUROGRAPHICS 2008),* 2008;27(2):221–30.](http://refhub.elsevier.com/S0097-8493(16)30111-X/sbref5)

[30] Hyojun K., Nitin A., Cristian U., *Examining Storage Performance on Mobile Devices,* ACM 978-1-4503-0980, MobiHeld '11, 2-3, 23 Ekim 2011

[31] URL-13: http://linux.die.net/man/8/debugfs, Theodore Ts’o, Debugfs.

[32] Sooman J., Kisung L., Jungwoo H., Seongjin L., Youjip W., AndroStep: Android Storage Performance Analysis Tool, Hanyang University, Dept. of Electronics and Computer Engineering, South Korea, Ocak 2013

[33] Chayong L., Euna K., Hyesoon K., The AM-Bench: An Android Multimedia

Benchmark Suite, Georgia Institute of Technology, School of Computer Science, United States of America, 2012

[34] Jae-Ho N a., Youngsun S b., Yeongkyu L a,\*., L-Bench: An Android benchmark set for low-power mobile GPUs, a LG Electronics 56 Digital-ro10-gil Geumcheon-gu Seoul, b LG Electronics 19 Yangjae-daero11-gilSeocho-gu Seoul, South Korea, 12 Eylül 2016

[35] URL-13: http://huseyinbodur.net/?tag=android-mimarisi, (19 Aralık 2018)

[36] URL-14: https://www.gizmotimes.com/what-how/antutu-benchmark/478,

(15 Aralık 2018)

# [37] Srivatsan S., Basemark OS II Benchmark Performance – Android vs iPhone vs Windows Phone, https://www.fonearena.com/blog/94251/basemark-os-ii-benchmark-performance-android-vs-iphone-vs-windows-phone.html (21 Aralık 2018)

# [38] Surur, GFXBench 3.0 Benchmark comes to Windows Phone, https://mspoweruser.com/gfxbench-3-0-benchmark-comes-to-windows-phone/ (22 Aralık 2018)

# [39] FoneTech, Honor 9 vs. Huawei P10 vs. Huawei P10 Plus - 3DMark Benchmark Test, https://www.youtube.com/watch?v=bTMkXQjKq8Q (23 Aralık 2018)

# [40] Rene Fürst, Samsung Galaxy S7 vs S6 Edge 3DMark Slingshot Benchmark, https://www.youtube.com/watch?v=dAX9U1o6m5k (28 Aralık 2018)

## [41] Ryan W., Androbench updated to v5.0 with a less ugly UI and fixed Pixel benchmarks, https://www.androidpolice.com/2016/10/28/androbench-updated-v5-0-less-ugly-ui-fixed-pixel-benchmarks/ (1 Ocak 2019)

# [42]<http://bilgisayarkavramlari.sadievrenseker.com/2008/08/09/hizli-siralama-algoritmasi-quick-sort-algorithm/>

# [43]<http://bilgisayarkavramlari.sadievrenseker.com/2010/05/13/dijkstra-algoritmasi-2/>]

# [44] https://en.wikipedia.org/wiki/Fast\_Fourier\_transform

# [45] http://code.google.com/p/zxing/, ZXing (”Zebra Crossing”)

# [46] <https://bilgihanem.com/ram-nedir-nasil-calisir/>].